

Package ‘catchmentACS’

October 6, 2026

Title Isochrone-Based Area-Weighted ACS Aggregation

Version 0.6.0

Description Computes American Community Survey (ACS) estimates for the area within a given drive time of each of a set of points, such as the locations of pre-kindergarten classrooms. Such an area is called an isochrone. The drive-time areas come from a routing service, either the 'Open Source Routing Machine' ('OSRM', <https://project-osrm.org/>) or 'openrouteservice' (<https://openrouteservice.org/>), and the ACS 5-year estimates of census tracts come from the Census Bureau (<https://www.census.gov/data/developers.html>) through the 'tidycensus' package, one state at a time. The tracts that overlap an area are combined by area weighting, which uses two weights: the share of each tract's area that lies inside, for counts and rates, and each tract's share of the overlapping area, for the medians and per-person values of three ACS tables (median household income, median home value, and per capita income). A median or per-person value from any other table is added up like a count. Counts, medians, per-person values, and five rates are returned with a margin of error at a chosen confidence level, 90 percent by default, and with a record of the settings, inputs, and package versions behind the run. The five rates are the poverty rate, the shares of households receiving Supplemental Nutrition Assistance Program (SNAP) benefits and Supplemental Security Income, the unemployment rate, and the labor force participation rate. The margins of error of counts and rates use the approximation formulas in chapter 8 of U.S. Census Bureau (2020) ``Understanding and Using American Community Survey Data: What All Data Users Need to Know" <https://www.census.gov/programs-surveys/acs/library/handbooks/general.html>. For a count the formula for a sum is applied to the weighted tract estimates, with the weights treated as fixed. For a rate the default is the formula for a ratio. Its margin of error is at least as wide as that of the formula for a proportion, which the handbook gives for ratios whose numerator is part of the denominator, as it is in all five rates. The margins of error of medians and per-person values are an approximation made by the package.

License MIT + file LICENSE

Copyright see file COPYRIGHTS

Encoding UTF-8

Language en-US

LazyData true

Depends R (>= 4.1.0)

Imports cli, digest, dplyr, httr2, jsonlite, rlang, sf (>= 1.0-12),
tibble, tidycensus (>= 1.6), tidyr, tigris

Suggests classInt, htmltools, knitr (>= 1.35), leaflet,
openrouteservice, osrm (>= 4.0), rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr, rmarkdown

Config/Needs/coverage covr

Config/Needs/website pkgdown

Config/testthat/edition 3

URL <https://github.com/joonho112/catchmentACS>,
<https://joonho112.github.io/catchmentACS/>

BugReports <https://github.com/joonho112/catchmentACS/issues>

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author JoonHo Lee [aut, cre, cph] (ORCID:
<<https://orcid.org/0009-0006-4019-8703>>)

Maintainer JoonHo Lee <jlee296@ua.edu>

Repository CRAN

Date/Publication 2026-10-06 07:20:08 UTC

Contents

as_tibble.cacs_run_result	3
cacs_acs_default_rates	4
cacs_acs_default_vars	5
cacs_acs_prefetch	6
cacs_acs_validate	9
cacs_alabama_sites	11
cacs_cache_dir	12
cacs_cache_status	14
cacs_capture_conditions	15
cacs_clear_cache	17
cacs_derive_rates	19
cacs_describe	23
cacs_get_cache_state	25
cacs_intersect_weight	26

cacs_isochrone	31
cacs_moe_to_se	36
cacs_plot_site_intersection	37
cacs_plot_site_isochrone	39
cacs_plot_site_pipeline	40
cacs_plot_site_rates	44
cacs_plot_site_weighted	46
cacs_propagate_moe	48
cacs_rings_to_cumulative	51
cacs_run	53
cacs_set_cache	61
cacs_se_to_moe	62
cacs_summary_as_markdown	63
cacs_validate_iso	64
cacs_validate_osrm_endpoint	66
catchmentACS-conditions	67
format.cacs_site_plot_pipeline	70
group_by.cacs_run_result	71
print.cacs_run_result	72
print.cacs_run_summary	74
print.cacs_site_plot_pipeline	75
summary.cacs_run_result	76
Index	78

as_tibble.cacs_run_result
<i>Convert the result of cacs_run() to a plain tibble</i>

Description

Removes the class `cacs_run_result` from a result of `cacs_run()`, so that the table prints as an ordinary tibble, and keeps the other attributes. The rows keep their order in `x` unless `rate_first` is `TRUE`; the rows are then sorted by `site_id`, `drive_time_min`, and `variable`, with the rows of the five rates before the other rows within each site and drive time.

Usage

```
## S3 method for class 'cacs_run_result'
as_tibble(x, ..., rate_first = NULL)
```

Arguments

- `x` A `cacs_run_result` object, as returned by `cacs_run()`.
- `...` Passed to the data frame method of `tibble::as_tibble()`, such as `.name_repair`.

rate_first A logical value: TRUE puts the rate rows first and FALSE keeps the order of the rows. With NULL (the default), the option `catchmentACS.rate_first_default`, FALSE unless it has been set, decides in the calls described above, and other calls keep the order. A result without a variable column, such as the list-column form, keeps its order either way.

Details

With `rate_first = NULL` (the default), the option `catchmentACS.rate_first_default` decides, but only when `as_tibble()` is called from code run in the global environment, such as the console, a script, a document, or a function written in one of them. A conversion made inside another package, such as `dplyr`, keeps the order, and so does a call that passes the function itself, as in `lapply(results, tibble::as_tibble)`. The first time the option puts the rate rows first in an R session, a message says so.

Value

A tibble of class `c("tbl_df", "tbl", "data.frame")` with the rows, columns, and attributes of `x`.

See Also

Other result summaries: `cacs_describe()`, `cacs_summary_as_markdown()`, `group_by.cacs_run_result()`, `print.cacs_run_result()`, `print.cacs_run_summary()`, `summary.cacs_run_result()`

Examples

```
out <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                           package = "catchmentACS"))$run_result

# By default the rows keep their order in out: the 14 ACS variables, then
# the five rates
tibble::as_tibble(out)$variable

# The rates first
tibble::as_tibble(out, rate_first = TRUE)$variable
```

`cacs_acs_default_rates`

Numerator and denominator ACS codes of the five built-in rates

Description

The numerator and denominator codes of each of the five rates that `cacs_derive_rates()` and `cacs_run()` compute. The codes are American Community Survey (ACS) variable codes. The list is the default, and the only accepted value, of the `rates` argument of both functions.

Usage

`cacs_acs_default_rates`

Format

A named list of five character vectors, one for each rate. Each vector has the elements num and den, the ACS codes of the numerator and the denominator.

Details

The five rates are:

`poverty_rate` People whose income in the past 12 months was below the poverty level (B17001_002) divided by people for whom poverty status is determined (B17001_001).

`snap_rate` Households that received Food Stamps or the Supplemental Nutrition Assistance Program (SNAP) in the past 12 months (B22003_002) divided by all households (B22003_001).

`ssi_rate` Households with Supplemental Security Income (SSI) in the past 12 months (B19056_002) divided by all households (B19056_001).

`unemp_rate` Unemployed people in the civilian labor force (B23025_005) divided by the civilian labor force (B23025_003), among people 16 years and over.

`labor_force_participation` People in the labor force, including the armed forces (B23025_002), divided by people 16 years and over (B23025_001).

See Also

`cacs_derive_rates()` computes the rates and their margins of error, and `vignette("theory-derived-rates", package = "catchmentACS")` (<https://joonho112.github.io/catchmentACS/articles/theory-derived-rates.html>) explains the calculations.

Other rates and margins of error: `cacs_moe_to_se()`, `cacs_se_to_moe()`

Examples

```
names(cacs_acs_default_rates)

# The codes of the numerator and the denominator of the poverty rate
cacs_acs_default_rates$poverty_rate
```

`cacs_acs_default_vars` *ACS variable codes used by default*

Description

The 14 American Community Survey (ACS) variable codes that `cacs_acs_prefetch()` and `cacs_run()` use when `variables` is NULL (the default). Ten of them are the numerators and denominators of the five rates in `cacs_acs_default_rates`. The other four are total population (B01003_001), total households (B11001_001), median household income (B19013_001), and per capita income (B19301_001).

The names are short labels for the codes, such as `pov_below` for B17001_002. The package uses only the codes, so results name each variable by its code.

Usage

```
data("cacs_acs_default_vars")
```

Format

A named character vector of length 14. Each value is an ACS variable code: B and five digits for the table, an underscore, and three digits, such as B17001_002.

Source

The codes are from the ACS detailed tables of the U.S. Census Bureau. The variables and their names were chosen for this package.

Examples

```
data("cacs_acs_default_vars", package = "catchmentACS")
cacs_acs_default_vars
names(cacs_acs_default_vars)
length(cacs_acs_default_vars) # 14
```

cacs_acs_prefetch

Download ACS estimates for the census tracts of one state

Description

Downloads American Community Survey (ACS) 5-year estimates for the census tracts of one state, with the tract boundaries, using `tidycensus::get_acs()`. Each estimate comes with the margin of error published with it, the half-width of its 90 percent confidence interval. The function removes water tracts unless `drop_water_tracts = FALSE`, and it returns negative margins of error as NA whatever the other arguments are (see the "Water tracts" and "Missing estimates and margins of error" sections). Unless the cache is turned off, the result is also saved in the cache folder and reused by later calls with the same request (see the "Cache behavior" section).

Usage

```
cacs_acs_prefetch(
  state,
  year = 2023L,
  variables = NULL,
  survey = "acs5",
  geography = "tract",
  cache_dir = NULL,
  write_gpkg = FALSE,
  force_refresh = FALSE,
  drop_water_tracts = TRUE,
  verbose = TRUE
)
```

Arguments

state	A string giving one state, the District of Columbia, or Puerto Rico, as a two-letter USPS abbreviation (such as "AL") or a two-digit FIPS code (such as "01"). A vector of states, or a lowercase code such as "al", gives an error.
year	A single whole number giving the last year of the ACS 5-year estimates, from 2009 to 2024; the default is 2023 (the 2019-2023 estimates).
variables	A character vector of ACS variable codes, or NULL (the default) for cacs_acs_default_vars . Each code must have the form of "B19013_001" (the letter B, five digits, an underscore, and three digits); other codes, such as "B01001A_001" or "C17002_001", give an error. "core" and "extended" select the same variables as NULL. For codes that are not in the ACS variable list for year, see the "Downloading from the Census Bureau" section. The Details of cacs_intersect_weight() say how each code is combined; a median or per-person value from a table other than B19013, B25077, or B19301 is added up like a count.
survey	A string giving the survey. Only "acs5" (the default), the ACS 5-year estimates, is accepted.
geography	A string giving the geographic level. Only "tract" (the default), census tracts, is accepted.
cache_dir	A path to the cache folder, or NULL (the default) to use cacs_cache_dir() ; see the "Cache behavior" section. A folder outside the temporary folder of the R session is tidied as described in cacs_cache_dir() . cacs_clear_cache() clears only the folder that cacs_cache_dir() returns.
write_gpkg	A logical value, FALSE (the default) or TRUE. If TRUE, a downloaded result is also written to a GeoPackage file (.gpkg) in the acs folder of the cache folder (cache_dir, or cacs_cache_dir() when cache_dir is NULL), for use in other GIS software. The file is written even when the cache is turned off, but not when a saved result is read. It is named after the cache key, attr(result, "cacs_provenance")\$cache_key, with the extension .gpkg. By default the cache folder is inside the temporary folder of the R session, so the file is deleted when the session ends; in a cache folder that lasts between sessions it is deleted together with the saved result (see cacs_cache_dir()), and cacs_clear_cache() also deletes it. To keep a copy, write the result with sf::st_write() .
force_refresh	A logical value. If TRUE, the data are downloaded even when a saved result exists, and the new result replaces it unless the cache is turned off. With FALSE (the default), a saved result is read when there is one; see the "Cache behavior" section.
drop_water_tracts	A logical value. If TRUE (the default), tracts numbered 9900 or higher and tracts whose boundary has no area are removed; if FALSE, they are kept unless they were already removed from a saved result. See the "Water tracts" section.
verbose	A logical value. With TRUE (the default), the function shows a message before the download and a summary line when it finishes, and other messages report reading a saved result, writing files, and removing water tracts. FALSE turns off these messages; warnings are still given. See the "Progress messages" section of cacs_run() for how to turn the summary line off.

Value

An sf data frame with one row for each tract and variable, and the columns GEOID (the 11-digit tract identifier: two digits for the state, three for the county, and six for the tract), NAME (the tract name), variable (the ACS variable code), estimate, moe (the margin of error), and geometry (the tract boundary, in NAD83, EPSG:4269).

The attributes `cacs_provenance` and `cacs_acs_provenance` hold the same list, a record of how the result was produced. It gives the request: the state, the year, the survey, the geography, and the variable codes. It counts the variables, tracts, rows, missing estimates, and margins of error set to NA. It also holds the cache key, the time the result was created (in UTC), and the tidycensus and sf versions. A result read from the cache keeps the list from the original download. The attribute `cacs_schema_version` is the version label ("1.0") of the column layout.

Downloading from the Census Bureau

Reading a saved result needs neither a Census API key nor an internet connection. The data are downloaded when no saved result is found or when `force_refresh = TRUE`. A download needs a Census API key in the `CENSUS_API_KEY` environment variable, which `tidycensus::census_api_key()` can set; if it is missing or empty, the function stops before any request is made. Variable codes that are not in the ACS variable list for the year are skipped with a warning, and the function stops if none is left. A failed download is tried up to three times, except that an HTTP 4xx error, such as a bad request, stops the function at once. The downloaded table is checked with the same rules as `cacs_acs_validate()`.

Water tracts

When `drop_water_tracts = TRUE` (the default), two kinds of tracts are removed right after the download. The first are tracts numbered 9900 or higher (a GEOID whose last six digits begin with 99), which the package treats as water or special-purpose tracts. The second are tracts whose boundary has zero, negative, or non-finite area, such as an empty boundary, because an area weight cannot be computed for them. A message lists the removed GEOIDs (the first five, followed by the number of others) unless `verbose = FALSE`.

The same rule is applied again when a saved result is read from the cache, but tracts removed before the result was saved stay removed even with `drop_water_tracts = FALSE`. They come back only with `force_refresh = TRUE` or after the saved result is deleted, for example with `cacs_clear_cache()`. With `drop_water_tracts = FALSE`, `cacs_intersect_weight()` skips tracts with zero area, with a warning.

Missing estimates and margins of error

Negative values in the margin-of-error column, which the Census Bureau's data API uses as codes rather than margins of error, are returned as NA; the function does not change estimates. For example, -55555555 means that a margin of error is not appropriate because the estimate is controlled to an independent population or housing estimate (U.S. Census Bureau, "Notes on ACS Estimate and Annotation Values"). A warning is given when more than 10 percent of the rows have a missing estimate, and another when more than 10 percent have -55555555 in place of a margin of error. Step 5 in the Details of `cacs_intersect_weight()` describes how these NA values carry into the estimates for the drive-time areas.

Cache behavior

By default the result is saved in the acs folder of the cache folder (cache_dir, or `cacs_cache_dir()` when cache_dir is NULL), with a small checksum file next to it. The default cache folder lasts only for the R session; `cacs_cache_dir()` says how to keep saved results between sessions and how long they are kept. A later call reads the saved result when it asks for the same state value, year, survey, geography, and variables (in any order) and the installed versions of R, catchmentACS, tidycensus, tigris, and sf have not changed. Otherwise the data are downloaded again (state = "AL" and state = "01", for example, are saved separately). A saved result that fails its checksum check is deleted and downloaded again.

A saved result that is read back with fewer than 1,000 rows gives a warning that it may be incomplete, once per session and only for copies with at least six variables (the catchmentACS.stale_min_variables option). The row limit is set with the option catchmentACS.stale_threshold_rows (0 turns the check off) or for one state value with an option such as catchmentACS.stale_threshold_WY.

See Also

`cacs_acs_validate()` checks ACS data obtained another way, and `cacs_get_cache_state()`, `cacs_set_cache()`, and `cacs_clear_cache()` show, turn on or off, and clear the cache.

Other steps of the calculation: `cacs_derive_rates()`, `cacs_intersect_weight()`, `cacs_isochrone()`, `cacs_propagate_moe()`, `cacs_run()`

Examples

```
library(sf)
# The rows of 60 census tracts around a site in Birmingham, Alabama, taken
# from the 2019-2023 estimates that cacs_acs_prefetch() downloaded for the
# state and kept in a file that comes with the package: one row for each
# tract and variable
acs_bhm <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                             package = "catchmentACS"))$acs_sf

head(acs_bhm)

# Downloads the estimates for every tract in Alabama, which needs a Census
# API key.
## Not run:
al <- cacs_acs_prefetch(state = "AL", year = 2023)

## End(Not run)
```

cacs_acs_validate

Check the form of ACS data for census tracts

Description

Checks that American Community Survey (ACS) estimates for census tracts, such as a table from tidycensus::get_acs() with geometry = TRUE, have the form that `cacs_intersect_weight()` and the acs argument of `cacs_run()` require. `cacs_run()` checks at the start only that acs is an sf object, and makes the other checks at its intersection step, after the routing step.

Usage

```
cacs_acs_validate(x)
```

Arguments

x An sf object of ACS estimates with one row for each tract and variable, or any other object to check.

Details

The checks are made in this order, and the first one that fails stops the function with an error:

1. `x` is an sf object.
2. Its coordinate reference system is NAD83 (EPSG:4269).
3. It has the columns GEOID, NAME, variable, estimate, moe (the margin of error), and geometry.
4. It has at least one row.
5. Every GEOID has 11 digits, like a census tract code.
6. estimate and moe are numeric.
7. No tract has two rows for the same variable.

Whether variable holds ACS codes, and the values of estimate and moe, are not checked. A name in place of an ACS code, such as "pop" from `tidycensus::get_acs(variables = c(pop = "B01003_001"))`, gets NA values, and `cacs_run()` then stops with an error about ACS codes that are not combined. Census Bureau annotation codes, such as -555555555, pass the checks. `cacs_intersect_weight()` treats them as missing, and its warning counts them, except a margin-of-error code next to a missing estimate; the example data below have only such codes and give no warning. A tract with no row for one of the variables is not reported, and it is left out of that variable's total or average without a warning. Whether every tract has one row for each variable can be checked with `all(table(acs$GEOID, acs$variable) == 1)`.

Value

TRUE, invisibly, when every check passes. Otherwise the function stops with an error of class `catchmentACS_error_schema` at the first check that fails.

See Also

`cacs_acs_prefetch()` downloads ACS data in this form.

Other validation and conditions: `cacs_capture_conditions()`, `cacs_validate_iso()`, `cacs_validate_osrm_endpoint`, `catchmentACS-conditions`

Examples

```
acs <- readRDS(system.file("extdata", "sample_alabama_subset.rds",
                           package = "catchmentACS"))
cacs_acs_validate(acs)

# tidycensus downloads the data, which needs a Census API key.
```

```
## Not run:
my_acs <- tidycensus::get_acs(
  geography = "tract", variables = "B01003_001",
  state = "AL", year = 2023, geometry = TRUE, output = "tidy"
)
cacs_acs_validate(my_acs)

## End(Not run)
```

cacs_alabama_sites	<i>Ten example sites near Alabama city centers</i>
--------------------	--

Description

Ten made-up site locations in Alabama, as an `sf` object of points. Each point lies near the center of an Alabama city. The points were not taken from the locations of Pre-K classrooms or of any other facility, and the data contain no administrative records.

The sites are the `sites` input of examples that need a routing service, such as the one in `cacs_run()`, and the default `sites_df` of the map functions, such as `cacs_plot_site_isochrone()`. The bundled files `legacy_2025_sites.rds` and `legacy_2025_isochrones.rds`, used by several examples that run offline, give the same `site_id` values to other points, 130 to 440 kilometers away.

Usage

```
data("cacs_alabama_sites")
```

Format

An `sf` object (a data frame with a geometry column) with 10 rows and 5 columns. The points are in longitude and latitude on WGS 84 (EPSG:4326).

`site_id` A string, "AL_SITE_01" to "AL_SITE_10".

`site_name` A string, "Sample Site 1" to "Sample Site 10".

`county_fips` A string giving the five-digit FIPS code of the county that contains the point, such as "01073" (Jefferson County) for the point near Birmingham.

`region_label` A string naming the part of the state, such as "Birmingham Metro" or "North"; each site has a different one.

`geometry` The point.

Source

Made for this package from public coordinates of the centers of ten Alabama cities: Birmingham, Mobile, Huntsville, Montgomery, Tuscaloosa, Auburn, Decatur, Florence, Dothan, and Gadsden, in the order of `site_id`. Each point was then moved by a random amount of up to 0.007 degrees of longitude and of latitude (less than 1 kilometer in all), drawn with a fixed random seed.

Examples

```
library(sf)
data("cacs_alabama_sites", package = "catchmentACS")
print(cacs_alabama_sites)
nrow(cacs_alabama_sites)      # 10
sf::st_crs(cacs_alabama_sites)$epsg  # 4326

# Needs a Census API key, and routes three sites on the public OSRM demo
# server with the osrm package; that server limits the requests it accepts.
## Not run:
library(sf)
result <- cacs_run(
  sites      = cacs_alabama_sites[1:3, ],
  state      = "AL",
  year       = 2023,
  drive_times = c(5, 10, 15),
  variables  = "core",
  provider   = "osrm",
  weight_method = "area",
  output     = "long",
  verbose    = FALSE
)
head(result)

## End(Not run)
```

cacs_cache_dir	<i>Get the path of the cache folder</i>
----------------	---

Description

Returns the path of the cache folder, where the package saves results on disk for reuse. By default, [cacs_isochrone\(\)](#), [cacs_acs_prefetch\(\)](#), and [cacs_intersect_weight\(\)](#) each save their results in a subfolder of it (isochrone, acs, and intersect), and a later call that matches an earlier one reads the saved result instead of computing or downloading it again. The help page of each function says what must match.

Usage

```
cacs_cache_dir(create = FALSE)
```

Arguments

create	A logical value. If FALSE (the default), the path is returned without creating the folder. If TRUE, the folder is created, together with any missing parent folders, when it does not exist, and an error is given if it cannot be created.
--------	---

Details

The cache folder is the first of these that is set and not empty:

1. the option `catchmentACS.cache_dir`;
2. the environment variable `CACS_CACHE_DIR`;
3. a folder named `catchmentACS` inside the temporary folder of the R session (see [tempdir\(\)](#)), which R deletes when the session ends.

The folder is worked out again at every call, so a change to the option or the environment variable takes effect at once.

Value

A string giving the path of the cache folder.

How saved results are checked

Each saved result is an `.rds` file with a checksum file (`.fingerprint`) next to it. When a saved result is read, its checksum is computed again and compared with the one in the checksum file. If the checksum file is missing or not in the expected form, if the `.rds` file cannot be read, or if the checksums differ, both files are deleted and the result is computed or downloaded again. When saved American Community Survey (ACS) data are read, [cacs_acs_prefetch\(\)](#) also warns if they look incomplete.

A cache folder that lasts between sessions

Results saved in the default folder are deleted when the R session ends. To keep them for later sessions, set the option or the environment variable to a folder that lasts, for example with this line in the R startup file (see [Startup](#)):

```
options(catchmentACS.cache_dir = tools::R_user_dir("catchmentACS", "cache"))
```

[tools::R_user_dir\(\)](#) gives a folder for the cache files of one package, inside the user's cache folder for R; where that is depends on the operating system.

A cache folder outside the temporary folder of the session is tidied once per session, the first time a result is read from it or saved in it while the cache is on. These files are deleted:

- results not used for 30 days, with their checksum and GeoPackage files. Reading a saved result counts as using it. The option `catchmentACS.cache_max_age_days` sets another number of days, and `Inf` keeps results however old they are. Its value when the folder is first used in the session is the one that counts, so set it in the R startup file next to `catchmentACS.cache_dir`, for example `options(catchmentACS.cache_max_age_days = 90)`; a value that is not a single positive number counts as 30;
- GeoPackage files as old as that without a result;
- files left by an interrupted write, and results or checksum files without their partner, when they are more than a day old.

Only files named the way the package names its saved files, in the subfolders `isochrone`, `acs`, `acs_test`, and `intersect`, are deleted.

Versions 0.5.1 and earlier saved results in the user cache folder of the operating system: `~/Library/Caches/catchmentACS` on macOS, `~/.cache/catchmentACS` on Linux, and `catchmentACS/catchmentACS/Cache` inside the folder named by the environment variable `LOCALAPPDATA` on Windows. The package no longer reads or deletes that folder; delete it yourself if it is not needed.

See Also

Other cache and configuration: [cacs_cache_status\(\)](#), [cacs_clear_cache\(\)](#), [cacs_get_cache_state\(\)](#), [cacs_set_cache\(\)](#)

Examples

```
# The path of the cache folder, without creating the folder
cacs_cache_dir(create = FALSE)
```

<code>cacs_cache_status</code>	<i>Summarize the files in the cache folder</i>
--------------------------------	--

Description

Counts the files in each subfolder of the cache folder returned by [cacs_cache_dir\(\)](#) and reports their total size and their oldest and newest modification times. The folder is not created if it does not exist. For a folder given in the `cache_dir` argument of another function, first set the option to it, as in `options(catchmentACS.cache_dir = "path/to/folder")`.

Usage

```
cacs_cache_status()
```

Value

A tibble with one row for each subfolder (`isochrone`, `acs`, `acs_test`, and `intersect`; see [cacs_clear_cache\(\)](#)) and these columns:

`namespace` The subfolder.

`n_entries` The number of saved results (`.rds` files).

`orphan_tmp` The number of temporary `.rds.tmp` files, which are left when a write is interrupted and are never read.

`n_fingerprints` The number of checksum files (`.fingerprint` files; see [cacs_cache_dir\(\)](#)).

`orphan_fingerprint_tmp` The number of temporary `.fingerprint.tmp` files, left in the same way.

`legacy_entries` The number of `.rds` files without a checksum file, such as those saved by versions of the package before 0.3.0. Such a file is deleted, and the result computed again, when a later call looks for it.

`total_size_mb` The total size of these files and of the GeoPackage files written by `cacs_acs_prefetch(write_gpkg = TRUE)`, in megabytes (1024^2 bytes).

`oldest, newest` The earliest and latest modification times of the same files, or NA when there are none. Reading a saved result updates its modification time.

Other files are not counted.

See Also

Other cache and configuration: [cacs_cache_dir\(\)](#), [cacs_clear_cache\(\)](#), [cacs_get_cache_state\(\)](#), [cacs_set_cache\(\)](#)

Examples

```
cacs_cache_status()
```

```
cacs_capture_conditions
```

Capture the package's messages and warnings in a table

Description

Evaluates an expression and returns a tibble listing the standardized catchmentACS messages and warnings given while it runs, that is, those with the class `catchmentACS_condition` (see [catchmentACS-conditions](#)). The captured messages and warnings are not shown.

Usage

```
cacs_capture_conditions(expr, classes = NULL, return_value = NULL)
```

Arguments

<code>expr</code>	An expression to evaluate, such as a call to cacs_run() .
<code>classes</code>	A character vector of the classes to capture, or NULL (the default) for all messages and warnings of the package. Each element can be a full class name, such as <code>"catchmentACS_message_progress_summary"</code> , or <code>"message"</code> or <code>"warning"</code> for all messages or all warnings of the package. It can also be the part of a class name after <code>catchmentACS_message_</code> , <code>catchmentACS_warning_</code> , or <code>catchmentACS_error_</code> , such as <code>"water_tract_filter"</code> or <code>"progress"</code> . A name that matches no class captures nothing, without an error.
<code>return_value</code>	A string giving what to return: <code>"conditions"</code> for the table of captured conditions, or <code>"both"</code> for a list with the value of <code>expr</code> as well. NULL (the default) uses the option <code>catchmentACS.capture_return_value</code> , which is <code>"conditions"</code> unless it has been changed.

Details

The conditions are captured with `withCallingHandlers()`, so `expr` runs to the end. Messages and warnings that are not captured, because they come from other packages or do not match classes, are shown as usual. Errors are not captured: an error in `expr` stops `cacs_capture_conditions()`, and the conditions captured before it are not returned.

Value

With `return_value = "conditions"`, a tibble with one row for each captured condition, in the order in which they were given, and these columns:

`class` The most specific class of the condition, such as `"catchmentACS_message_progress_summary"`.

`message` The text of the condition, as returned by `conditionMessage()`.

`phase` The step that gave the condition: `"acs"` for `cacs_acs_prefetch()`, `"isochrone"` for `cacs_isochrone()`, `"intersect"` for `cacs_intersect_weight()`, `"moe"` for `cacs_propagate_moe()`, `"rates"` for `cacs_derive_rates()`, `"run"` for `cacs_run()` itself, and `"cache"` for `cacs_set_cache()` and `cacs_clear_cache()`. Messages and warnings about saved results have the value of their step (some of those about ACS data saved in test mode have `"acs_test"`; see `namespace_mode` in `cacs_get_cache_state()`). A progress line or summary line has the label of its step, such as `"Intersect+weight"` (see the "Progress messages" section of `cacs_run()`). The warning of `cacs_validate_osrm_endpoint()` has `"isochrone"`, and the message and warning of the maps about the nearest site have the name of the function, such as `"cacs_plot_site_rates"`. The warnings of `cacs_intersect_weight()` about repaired geometries and the message of `as_tibble.cacs_run_result()` have NA. The values are not the names of the elements of the `cacs_run_warnings` attribute of a `cacs_run()` result.

`timestamp` The time at which the condition was given, as a date-time in UTC.

`call` The call in which the condition was given, as text, or NA when there is none. For a condition given while `cacs_run()` runs a step, the text can include the data passed to the step and be very long.

With `return_value = "both"`, a list with two elements: `result`, the value of `expr`, and `conditions`, the tibble above.

Keeping the value of the expression

With `return_value = "conditions"`, the value of `expr` is not returned. An assignment made with `<-` inside `expr` does not keep it either, because `expr` is evaluated in a new environment: `cacs_capture_conditions(acs <- cacs_acs_prefetch("AL"))` leaves `acs` as it was. With `return_value = "both"`, the value is kept as the result element of the list. An assignment made with `<<-` inside `expr` also keeps it, in the first variable of that name found by searching from the environment in which `cacs_capture_conditions()` is called (see [assignOps](#)).

See Also

Other validation and conditions: [cacs_acs_validate\(\)](#), [cacs_validate_iso\(\)](#), [cacs_validate_osrm_endpoint\(\)](#), [catchmentACS-conditions](#)

Examples

```

# Turn the cache off while this example runs (see ?cacs_set_cache).
old <- options(catchmentACS.cache_enabled = FALSE)

# Example data bundled with the package: the drive-time areas are circles
# with a radius of 1 km per minute, and the ACS data are made up.
library(sf)
iso <- readRDS(system.file("extdata", "legacy_2025_isochrones.rds",
                           package = "catchmentACS"))
acs <- readRDS(system.file("extdata", "sample_alabama_subset.rds",
                           package = "catchmentACS"))
iso_07 <- iso[iso$site_id == "AL_SITE_07" & iso$drive_time_min == 10, ]
site_07 <- data.frame(site_id = "AL_SITE_07", lon = -85.365, lat = 31.655)

# The messages of a run, captured instead of shown; out$result is the
# result of cacs_run()
out <- cacs_capture_conditions(
  cacs_run(site_07, state = "AL", drive_times = 10,
           precomputed_isochrones = iso_07, acs = acs),
  return_value = "both"
)
out$conditions[, c("class", "message")]

# Only the warnings: the range check on rates, turned on here, warns
# about two rates of the made-up data
old_audit <- options(catchmentACS.audit_rates = TRUE)
warned <- cacs_capture_conditions(
  cacs_run(site_07, state = "AL", drive_times = 10,
           precomputed_isochrones = iso_07, acs = acs, verbose = FALSE),
  classes = "warning"
)
warned[, c("class", "message")]
options(old_audit)

options(old)

# The download from the Census Bureau needs a Census API key.
## Not run:
out <- cacs_capture_conditions(
  cacs_acs_prefetch(state = "AL", year = 2023),
  classes = "water_tract_filter",
  return_value = "both"
)
al <- out$result
out$conditions$message

## End(Not run)

```

Description

Deletes the saved results chosen by namespace, with their checksum files, any temporary files left by an interrupted write, and the GeoPackage files written by `cacs_acs_prefetch(write_gpkg = TRUE)`, from the cache folder returned by `cacs_cache_dir()`. Other files in the folder and the subfolders themselves are kept. A folder given in the `cache_dir` argument of another function is not changed; to clear such a folder, first set the option to it, as in `options(catchmentACS.cache_dir = "path/to/folder")`.

Usage

```
cacs_clear_cache(
  namespace = c("all", "isochrone", "acs", "acs_test", "intersect"),
  confirm = interactive()
)
```

Arguments

namespace	A string naming the kind of saved result to delete, which is also the name of its subfolder: "all" (the default) All four kinds. "isochrone" Drive-time areas saved by <code>cacs_isochrone()</code>. "acs" American Community Survey (ACS) data saved by <code>cacs_acs_prefetch()</code>. "acs_test" ACS data saved in test mode (see <code>namespace_mode</code> in <code>cacs_get_cache_state()</code>). "intersect" Results saved by <code>cacs_intersect_weight()</code>.
confirm	A logical value. If TRUE, a question is asked first, and files are deleted only if the answer is yes; in a non-interactive session nobody can answer, so nothing is deleted. If FALSE, files are deleted without a question. The default, <code>interactive()</code>, is TRUE only in an interactive session, so a script run with Rscript deletes without asking. The question is also skipped when the environment variable <code>CACS_NO_CONFIRM</code> is "1".

Details

After deleting, `cacs_clear_cache()` sets the hit and miss counts of the cleared subfolders to zero (see `cacs_get_cache_state()`). If the cache folder does not exist, a message says so and nothing is deleted.

Value

A tibble, returned invisibly, with one row for each subfolder cleared and the columns `namespace`, `n_removed` (the number of files deleted, counting a saved result, its checksum file, and its GeoPackage file as separate files), and `bytes_freed` (their total size in bytes). It has no rows when the cache folder does not exist or the answer to the question is no.

See Also

Other cache and configuration: `cacs_cache_dir()`, `cacs_cache_status()`, `cacs_get_cache_state()`, `cacs_set_cache()`

Examples

```
# Use a temporary cache folder, so that a cache folder you have set is
# left alone
old <- options(catchmentACS.cache_dir = file.path(tempdir(), "cacs-example"))
cacs_cache_status()
cacs_clear_cache("isochrone", confirm = FALSE)
options(old)
```

cacs_derive_rates	<i>Compute rates and their margins of error for drive-time areas</i>
-------------------	--

Description

Computes the five rates in [cacs_acs_default_rates](#), such as the poverty rate, for each site and drive-time pair in the output of [cacs_propagate_moe\(\)](#), each with a margin of error (MOE). A margin of error is the half-width of a confidence interval. The rates are added after the input rows, as five new rows for each pair.

Usage

```
cacs_derive_rates(
  weighted_acs,
  rates = cacs_acs_default_rates,
  formula_dispatch = "general_ratio_conservative",
  verbose = TRUE,
  ...
)
```

Arguments

- | | |
|-------------------------------|--|
| <code>weighted_acs</code> | A tibble returned by cacs_propagate_moe() , from which rows may have been removed (for example with <code>[]</code> or <code>dplyr::filter()</code>) but whose columns and attributes are kept, including <code>cacs_aggregation_carriers</code> . A tibble returned by cacs_intersect_weight() can also be used; the margins of error of the rates are then at the 90 percent level. The rates are computed from <code>cacs_aggregation_carriers</code> for each site and drive-time pair that still has a row, so removing the rows of a rate's codes does not change the rate. |
| <code>rates</code> | A named list giving, for each rate, the American Community Survey (ACS) codes of the numerator and the denominator. Only cacs_acs_default_rates (the default), the list of the five built-in rates, is accepted; any other list, including a subset or a reordering of it, gives an error. |
| <code>formula_dispatch</code> | A string choosing the margin-of-error formula for the rates: <code>"general_ratio_conservative"</code> (the default), <code>"auto"</code> , or <code>"proportion_subset"</code> . The last two give the same result: the proportion formula for <code>poverty_rate</code> and <code>labor_force_participation</code> only, and the ratio formula for the other three (see Details). |

`verbose` A logical value. With TRUE (the default), the function shows a one-line summary when the step finishes (see the "Progress messages" section of `cacs_run()`).

`...` Not used: any argument given here, such as `level = 0.95`, gives an error. The confidence level of the rates is set by the `level` argument of `cacs_propagate_moe()`.

Details

Each rate is the ratio of two weighted counts of ACS estimates. The two counts and their variances are read from the `cacs_aggregation_carriers` attribute that `cacs_intersect_weight()` attaches. The margin of error is computed from them at the confidence level that `cacs_propagate_moe()` records in the `cacs_confidence_level` attribute (the 90 percent level if there is none). The result does not keep `cacs_aggregation_carriers`, so `cacs_derive_rates()` gives an error if it is run on its own result.

The numerator of each of the five rates is part of its denominator, so the rates are proportions, for which the Census Bureau's handbook gives the proportion formula, C1 (U.S. Census Bureau 2020, chapter 8). By default, the margin of error of every rate is computed with the ratio formula, C2, which for the same rate is never narrower than the margin the proportion formula gives. `formula_dispatch` can choose the proportion formula for two of the five rates. The table gives the formula used for each rate with each value of `formula_dispatch`.

Rate	Default	"auto"	"proportion_subset"
<code>poverty_rate</code>	ratio	proportion	proportion
<code>labor_force_participation</code>	ratio	proportion	proportion
<code>snap_rate</code>	ratio	ratio	ratio
<code>ssi_rate</code>	ratio	ratio	ratio
<code>unemp_rate</code>	ratio	ratio	ratio

"auto" and "proportion_subset" give the same rates and margins of error. "proportion_subset" also gives a warning naming the three rates that keep the ratio formula, and lists them in the `formula_downgraded_rates` element of the `cacs_rate_provenance` attribute, a list recording how the rates were computed. The warning calls those three rates ineligible for the proportion formula. That wording describes the package's list of rates and not the data: the numerator of each of the three is part of its denominator. Both formulas are given in the "MOE formula families" section of `cacs_propagate_moe()`.

Under "auto" and "proportion_subset", the three rates in the table keep the ratio formula whatever the values are; the choice is built into the package and does not depend on the data. `unemp_rate` keeps it to reproduce the 2025 analysis the package was first written for, and the package gives no reason for `snap_rate` and `ssi_rate`. `vignette("theory-derived-rates", package = "catchmentACS")` describes how the proportion formula can be computed for them from the estimate and moe of the rows of their two ACS codes.

When the value under the square root of the proportion formula is negative, the row uses the ratio formula instead (see `cacs_propagate_moe()`), with `moe_fallback = TRUE` and `moe_fallback_reason = "negative_variance"`.

Value

The tibble `weighted_acs` with its rows unchanged and in the same order, followed by the rate rows. On the rate rows:

`variable`, `estimand_family` The name of the rate, such as "poverty_rate", and "derived_rate".
`estimate`, `moe` The rate (not a percentage) and its margin of error.

`moe_formula_requested`, `moe_formula_effective` The formula chosen for the rate (see the table in Details) and the formula used, which differs only where the ratio formula replaced the proportion formula. With `formula_dispatch = "proportion_subset"`, the three rates that keep the ratio formula have "general_ratio_conservative" in both columns.

`moe_fallback`, `moe_fallback_reason` TRUE with "negative_variance" or "zero_denominator" when the chosen formula could not be used (see Details and the "Rates that are NA" section), and FALSE with "n/a" otherwise, including when `failure_origin` is "carrier".

`weight_sum` The smaller of the sums of the coverage weights for the numerator and for the denominator (a tract's coverage weight is the share of its area inside the drive-time area); NA when `failure_origin` is "carrier".

`n_tracts`, `n_tracts_num`, `n_tracts_den` See the "Tract counts for rates" section.

`failure_origin` "none", or "carrier" when a count or margin of error that the rate needs is missing (see the "Rates that are NA" section).

The other columns are described in the Value section of `cacs_run()`.

The attributes of `weighted_acs` are kept, except `cacs_aggregation_carriers`, and `cacs_confidence_level` is set to the confidence level used. Two attributes are added:

`cacs_rate_provenance` A list recording how the rates were computed. It gives the rates and the formula chosen for each, the value of `formula_dispatch`, and the rates that "proportion_subset" left on the ratio formula. It also counts the rate rows: all of them, and those with a substituted formula, a zero denominator, `failure_origin = "carrier"`, or a rate outside the check ranges. The last fields are the confidence level and its z value, the time the rates were computed, and the package version.

`cacs_rate_audit` The result of the range check described in the "Checking rates against ranges" section.

Rates that are NA

A rate and its margin of error are NA when the numerator or the denominator, or the margin of error of either, is missing for the pair. This happens when a code of the rate has no rows in the ACS data, when a tract in the area has a missing estimate or margin of error for one of the two codes (including a Census Bureau code that `cacs_intersect_weight()` sets to NA), and when no tract is left for the pair. The codes of all five rates are in `cacs_acs_default_vars`, the default variables of `cacs_acs_prefetch()` and `cacs_run()`. Rates that are NA for these reasons have "carrier" in `failure_origin`, the column that records the step at which a row failed; the value is named after the `cacs_aggregation_carriers` attribute, which holds the weighted totals. A rate whose denominator is zero is also NA, as is its margin of error, with "zero_denominator" in `moe_fallback_reason` and TRUE in `moe_fallback`.

One warning at the end counts these rows and the rows where the ratio formula replaced the proportion formula. Its class is `catchmentACS_warning_runtime`; when it counts only rows with `failure_origin = "carrier"`, it also has the class `catchmentACS_warning_carrier_missing`.

Tract counts for rates

On rate rows, `n_tracts_num` and `n_tracts_den` give the numbers of tracts combined for the numerator and for the denominator (the values of `n_tracts` on the rows of those two ACS codes), and `n_tracts` is NA. On the rows of the ACS variables, `n_tracts_num` and `n_tracts_den` are NA, and they are also NA on rate rows with `failure_origin = "carrier"`.

When, in the ACS data, a tract in the area has a row for one of the two codes of a rate and not for the other, for example after rows with a missing estimate have been removed, the numerator and the denominator are totals over different sets of tracts. The rate is then no longer the share of one population: it can be much larger or smaller than the share for the area, and even above 1, and no warning is given. The two counts then differ unless the two codes lack the same number of tracts, so equal counts do not rule this out. Rows whose estimate is missing, kept as NA rows rather than removed, make the rate NA, with a warning (see the "Rates that are NA" section). `cacs_acs_validate()` shows how to check that every tract has one row for each variable.

Checking rates against ranges

With `options(catchmentACS.audit_rates = TRUE)`, each rate is compared with a fixed range: 0 to 0.6 for `poverty_rate`, 0 to 0.5 for `snap_rate`, 0 to 0.15 for `ssi_rate`, 0 to 0.3 for `unemp_rate`, and 0.3 to 0.85 for `labor_force_participation`. For each rate with values outside its range, a warning of class `catchmentACS_warning_rate_out_of_range` gives the number of those rows and the smallest and largest of their values; the rates are not changed. The `cacs_rate_audit` attribute records whether the check was on (enabled), the number of rate rows outside the ranges (`n_out_of_bounds`), and those rows (`rows`). It is attached even when the check is off (the default), with `enabled = FALSE` and no rows.

References

U.S. Census Bureau (2020). *Understanding and Using American Community Survey Data: What All Data Users Need to Know*. Chapter 8, Calculating Measures of Error for Derived Estimates.

See Also

For the rates and their margins of error at more length, see `vignette("theory-derived-rates", package = "catchmentACS")` (<https://joonho112.github.io/catchmentACS/articles/theory-derived-rates.html>).

Other steps of the calculation: `cacs_acs_prefetch()`, `cacs_intersect_weight()`, `cacs_isochrone()`, `cacs_propagate_moe()`, `cacs_run()`

Examples

```
# Turn the cache off while this example runs (see ?cacs_set_cache).
old <- options(catchmentACS.cache_enabled = FALSE)

# Example data bundled with the package: the drive-time areas are circles
# with a radius of 1 km per minute, and the ACS data are made up.
library(sf)
iso <- readRDS(system.file("extdata", "legacy_2025_isochrones.rds",
                           package = "catchmentACS"))
acs <- readRDS(system.file("extdata", "sample_alabama_subset.rds",
```

```

      package = "catchmentACS"))
iso_07 <- iso[iso$site_id == "AL_SITE_07" & iso$drive_time_min == 10, ]
agg <- cacs_intersect_weight(iso_sf = iso_07, acs_sf = acs,
                           verbose = FALSE)
prop <- cacs_propagate_moe(agg, verbose = FALSE)

# The five rates, with margins of error at the 90 percent level
out <- cacs_derive_rates(prop, verbose = FALSE)
is_rate <- out$estimand_family == "derived_rate"
out[is_rate, c("variable", "estimate", "moe", "moe_formula_effective")]

# Margins of error with the default and with formula_dispatch = "auto"
out_auto <- cacs_derive_rates(prop, formula_dispatch = "auto",
                             verbose = FALSE)
tibble::tibble(variable = out$variable[is_rate],
               moe_default = out$moe[is_rate],
               moe_auto = out_auto$moe[is_rate],
               formula_auto = out_auto$moe_formula_effective[is_rate])

options(old)

```

cacs_describe

Describe how a result was produced

Description

Prints a description of an object made by the package, drawn from the attributes that record how it was produced, and returns the description invisibly as a list. It describes the results of `cacs_run()` and of the functions that `cacs_run()` runs: `cacs_isochrone()`, `cacs_acs_prefetch()`, `cacs_intersect_weight()`, `cacs_propagate_moe()`, and `cacs_derive_rates()`. Their Value sections describe the attributes.

Usage

```

cacs_describe(x, ...)

## S3 method for class 'cacs_description'
print(x, ...)

```

Arguments

<code>x</code>	An object to describe (see Details). For <code>print()</code> , a list returned by <code>cacs_describe()</code> .
<code>...</code>	Not used.

Details

The output starts with a title and the kind of object, which is also kept in `object_type` and decides the sections that follow:

"cacs_run_result" A result of `cacs_run()`. For a result with `output = "both"`, its element `long` is described. The sections are "Object", "Run", "Rates", and "Cache". "Run" shows the routing service and profile, the drive times, the numbers of sites, and the running time, as `print.cacs_run_result()` does, and "Rates" the lines of "Rate Derivation" below. For the list-column form, "Rate rows" is 0, because the rates are inside its `derived_rates` column. "Cache" shows values of `cacs_get_cache_state()` for the R session at the time of the call.

"weighted_seam", "propagated_seam", "derived_rates" The results of `cacs_intersect_weight()`, `cacs_propagate_moe()`, and `cacs_derive_rates()`, told apart by their attributes; another tibble is `"tbl_df"`. After "Object", "Carrier Table" summarizes the `cacs_aggregation_carriers` attribute, the table of weighted totals and means, with their variances, that `cacs_intersect_weight()` keeps for the later steps: the number of rows, the numbers of distinct variables and sites, how many rows have a missing total, and the range of `weight_sum`. `cacs_derive_rates()` removes that attribute, and for its result the section reads Carrier attribute: absent or already consumed. "MOE Propagation" shows values of `cacs_moe_provenance`, the record of how the margins of error (MOE) were computed. "Rate Derivation" shows the number of rate rows and values of `cacs_rate_provenance`, and "Aggregation" values of `cacs_aggregation_provenance`.

"isochrone_sf", "acs_sf", "sf" sf objects told apart by their columns: drive-time areas, such as a result of `cacs_isochrone()`, American Community Survey (ACS) data, such as a result of `cacs_acs_prefetch()`, and others. "Spatial Provenance" shows values of `cacs_isochrone_provenance` and `cacs_res_param`, and "ACS Provenance" values of `cacs_acs_provenance`.

"unknown" Any other object, with one section saying that there is nothing to describe.

A data frame that is not a tibble is described as a tibble only if it has one of the package's attributes or one of the columns `site_id`, `drive_time_min`, `variable`, `estimand_family`, `ring_topology`, or `failure_origin`. A value that is not recorded is shown as `n/a`. Two sections differ: "Rates computed" lists the five rates even for an object without rate rows, and "Carrier Table" says that the attribute is absent or already consumed.

The description does not count the rate rows whose chosen formula for the margin of error could not be used (`moe_fallback = TRUE`). The lines about rates in "MOE Propagation", such as "C1 to C2 fallbacks", are always 0 (see `cacs_propagate_moe()`). "Formula downgrades" is the number of rates that `formula_dispatch = "proportion_subset"` left on the ratio formula. In "Aggregation", "Input sites" is the number of site and drive-time pairs, and "Sites with data" is the number of rows with tracts in the `cacs_intersect_weight()` result, which has one row for each pair and variable.

The description is printed as R messages, so `suppressMessages()` hides it.

Value

`cacs_describe()` returns, invisibly, a list of class `cacs_description` with these elements:

`object_type` A string naming the kind of object (see Details).

`sections` A named list with one character vector of printed lines for each section.

`data` A named list of the attributes that the sections are drawn from, such as `rate_provenance`, the `cacs_rate_provenance` attribute of `x` or `NULL`. For a result of `cacs_run()`, it also holds `cache_state`, from `cacs_get_cache_state()`.

`attributes_present` A character vector of the names of the package's attributes that `x` has.

`print()` prints `x` in the same way and returns it invisibly.

See Also

Other result summaries: `as_tibble.cacs_run_result()`, `cacs_summary_as_markdown()`, `group_by.cacs_run_result()`, `print.cacs_run_result()`, `print.cacs_run_summary()`, `summary.cacs_run_result()`

Examples

```
# A bundled cacs_run() result for a 10-minute area in Birmingham, Alabama
out <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                           package = "catchmentACS"))$run_result
desc <- cacs_describe(out)

# The package's attributes that out has
desc$attributes_present
```

`cacs_get_cache_state` *Show the cache settings and how often saved results were used*

Description

Returns a one-row table with the cache settings, the number of times a saved result was used (a hit) or not used (a miss) in this R session, and a summary of the files in the cache folder.

Usage

```
cacs_get_cache_state()
```

Value

A tibble with one row and these columns:

`enabled` Whether the cache is on (see `cacs_set_cache()`). This column does not show a kind of saved result that is turned off on its own.

`cache_dir` The path of the cache folder, as returned by `cacs_cache_dir()`.

`namespace_mode` "test" in test mode, which is on, for example, while `testthat` runs tests and no cache folder is set with the option `catchmentACS.cache_dir` or the environment variable `CACS_CACHE_DIR`. American Community Survey (ACS) data are then kept in the subfolder `acs_test` instead of `acs`. Otherwise "production".

`fingerprint_algorithm` The checksum method of the checksum files, "sha256" (see `cacs_cache_dir()`).

`counters` A list holding a tibble with one row for each kind of saved result (`isochrone`, `acs`, and `intersect`; see `cacs_clear_cache()`). Its columns are `namespace`, `hit`, `miss`, and `hit_rate`, which is `hit / (hit + miss)` or `NaN` before the first lookup. Its `acs` row adds up the subfolders `acs` and `acs_test`.

`hits, misses` Lists each holding a named integer vector of the same counts for each subfolder: `isochrone`, `acs`, `acs_test`, and `intersect`.

`status` A list holding the table returned by `cacs_cache_status()`.

`session_started_at` The time, in UTC, at which the package was loaded in this session.

Hits and misses

Each time `cacs_isochrone()`, `cacs_acs_prefetch()`, or `cacs_intersect_weight()` looks for a saved result, the lookup counts as a hit if the saved result is used and as a miss if the result is computed or downloaded instead. A miss happens when nothing is saved for the inputs or the saved result fails its checksum check (see `cacs_cache_dir()`), and, for the first two functions, when the cache is off. `cacs_intersect_weight()` does not look while the cache is off, and `cacs_acs_prefetch()` does not look with `force_refresh = TRUE`. Lookups in a folder given in a `cache_dir` argument are counted too. The counts start when the package is loaded, and `cacs_clear_cache()` sets them back to zero for the subfolders it clears.

See Also

Other cache and configuration: `cacs_cache_dir()`, `cacs_cache_status()`, `cacs_clear_cache()`, `cacs_set_cache()`

Examples

```
cacs_get_cache_state()

# The numbers of hits and misses for each kind of saved result
cacs_get_cache_state()$counters[[1]]
```

`cacs_intersect_weight` *Aggregate ACS tract estimates to drive-time areas by area weighting*

Description

Combines the American Community Survey (ACS) estimates of the census tracts that overlap each drive-time area (isochrone) into an estimate and a margin of error for every site, drive time, and variable. Counts, such as total population (B01003_001), are added up with each tract weighted by the share of its area inside the drive-time area (its coverage weight), which assumes that whatever a variable counts is spread evenly over each tract's area. The assumption is made for each variable separately, and it is a stronger one for a subgroup, such as the people below the poverty level, than for the population as a whole. Medians and per-person values, such as median household income (B19013_001) and per capita income (B19301_001), are averaged over the overlapping tracts with weights proportional to the area each tract shares with the drive-time area. This average stands in for the median or per-person value of the drive-time area, but its weights follow area and ignore how many people live in each tract. It can therefore be far from that value when the overlapping tracts differ in population density.

Usage

```
cacs_intersect_weight(
  iso_sf,
  acs_sf,
  bg_pop_sf = NULL,
  weight_method = c("area", "population"),
```

```

    min_weight = 1e-06,
    verbose = TRUE,
    keep_tract_audit = FALSE,
    cache_dir = NULL
  )

```

Arguments

<code>iso_sf</code>	An sf object of drive-time areas as returned by cacs_isochrone() , with one row for each site and drive time. Data from other sources must have the same columns, which cacs_validate_iso() checks, and each area must contain the shorter ones (<code>ring_topology = "cumulative"</code> ; see cacs_rings_to_cumulative()).
<code>acs_sf</code>	An sf object of ACS estimates for census tracts as returned by cacs_acs_prefetch() , with one row for each tract and variable. Data from other sources must have the same columns, which cacs_acs_validate() checks.
<code>bg_pop_sf</code>	An sf object of block-group population estimates, or NULL (the default). It has no effect on the result (see <code>weight_method</code>).
<code>weight_method</code>	A string giving the weighting method: "area" (the default) or "population", which is not implemented yet; using it gives an error before any other work is done.
<code>min_weight</code>	A single number in [0, 1); the default is 1e-6. Tracts whose coverage weight is at or below it are dropped (see step 4 of Details).
<code>verbose</code>	A logical value. If TRUE (the default), progress messages are shown: a summary line when the step finishes and, with five or more site and drive-time pairs, also a line as each pair finishes. See the "Progress messages" section of cacs_run() for how to turn them off or show the lines for any number of pairs.
<code>keep_tract_audit</code>	A logical value, FALSE (the default) or TRUE. If TRUE, the result has a <code>cacs_tract_audit</code> attribute: a table with one row for each tract used in each drive-time area. Its columns are <code>site_id</code> , <code>drive_time_min</code> , GEOID, the tract's coverage weight (<code>area_wt</code>), and the areas of the overlap (<code>int_area_m2</code>) and of the tract (<code>tract_area_m2</code>) in square meters.
<code>cache_dir</code>	A path to the cache folder, or NULL (the default) to use cacs_cache_dir() . The result is saved in, and looked for in, its <code>intersect</code> subfolder. A folder outside the temporary folder of the R session is tidied as described in cacs_cache_dir() .

Details

Like the ACS margins they are combined from, the margins of error are half-widths of 90 percent confidence intervals, and they treat the weights as fixed and the tract estimates as independent (see [cacs_propagate_moe\(\)](#)). Rates, such as the poverty rate, are computed later by [cacs_derive_rates\(\)](#), each as the ratio of two of these weighted counts.

The function works in these steps.

1. Checks the inputs. `iso_sf` must be in EPSG:4326 (longitude and latitude on WGS 84) and `acs_sf` in EPSG:4269 (NAD83), as returned by [cacs_isochrone\(\)](#) and [cacs_acs_prefetch\(\)](#), and two rows of `acs_sf` for the same tract and variable give an error. Both must lie within

a box around the contiguous United States and the District of Columbia, so data for Alaska, Hawaii, or Puerto Rico give an error. The codes that the Census Bureau's data API puts in place of some estimates and margins of error (-222222222, -333333333, -555555555, -666666666, -888888888, and -999999999) are set to NA, with a warning that counts them; a margin-of-error code next to a missing estimate is not counted. Other negative values are used as they are. A warning is also given when the bounding box of the drive-time areas extends beyond that of the tracts (see below).

2. Transforms both inputs to EPSG:5070 (NAD83 / Conus Albers), an equal-area projection, and measures all areas there, in square meters. Invalid geometries are repaired, with a warning. For `iso_sf`, the change from WGS 84 to NAD83 is the one that PROJ chooses. Without datum grid files, PROJ treats the two as the same; with them, it can move the areas by a meter or two. Results can therefore differ between computers in the fourth or fifth significant digit.
3. Skips, with a warning, the tracts whose area is zero or not finite, such as a water tract with an empty boundary, and stops with an error if every tract is like that. The skipped tracts are listed in the `skipped_geoids` attribute.
4. For each site and drive time, finds the tracts that overlap the area. A tract's coverage weight is the area of its overlap divided by the tract's area. Tracts with a coverage weight at or below `min_weight`, including tracts that only touch the edge of the area, are dropped. Each area is used whole, so the 10-minute results include the tracts of the 5-minute area.
5. Combines the remaining tracts for each variable. A count is the sum of the tract estimates multiplied by their coverage weights, and a median or per-person value is the average of the tract estimates weighted by the area of each overlap. The margin of error is $\sqrt{\sum_i (w_i M_i)^2}$, where M_i is the margin of error of tract i and w_i its weight. For a count the weight is the coverage weight, and for a median or per-person value it is the area share (see the "Coverage weights and area shares" section). A missing estimate in any of the tracts, including a code set to NA in step 1, makes the variable's estimate and margin of error NA; a missing margin of error makes only the margin of error NA. A rate that uses the variable is NA in both cases (see `cacs_derive_rates()`). `cacs_acs_prefetch()` sets negative margins of error to NA.

Whether a variable is a count, a median, or a per-person value is decided by its ACS code. Tables B19013 and B25077 are medians, and table B19301 is a per-person value. Every other code of the form B, five digits, an underscore, and three digits (such as B17001_002) is treated as a count, so a median or per-person value from another table is added up like a count. Other codes, such as those of tables whose names begin with C or S or end with a letter (such as B17001A), are not combined and get NA values.

Only the tracts in `acs_sf` are used: any part of a drive-time area outside them (for example, across a state line) adds nothing, so counts come out too small, and rates and medians come from the remaining tracts. The only warning about this, in step 1, compares the bounding box of all the areas in `iso_sf` with the bounding box of all the tracts in `acs_sf`: it is given when the first extends more than 0.05 degrees (about 5 km) beyond the second on some side. The tracts of a neighboring state can be included in `acs_sf`, for example by combining two `cacs_acs_prefetch()` results with `dplyr::bind_rows()`; the combined data no longer record the ACS year, so `acs_year` in the result is NA.

A site and drive time can have no tract left after step 4: no tract overlaps the area, the area is empty or appears twice in `iso_sf`, or every tract is at or below `min_weight`. Such a pair gives one row with `variable = NA`, NA values, `n_tracts = 0`, and `failure_origin = "intersection"`. No warning is given. The same row is given when the overlap calculation for a pair fails, so one failing pair does not stop the others.

Results are cached (see `cacs_set_cache()`) in the folder given by `cache_dir` or `cacs_cache_dir()`: a call that matches an earlier one returns the saved result without repeating steps 2 to 5 or their warnings. The drive-time areas are matched by the `cache_key` that `cacs_isochrone()` attaches to its result together with their contents: the geometries, the coordinate reference system, and the columns `site_id`, `drive_time_min`, `provider`, `profile`, `osm_snapshot_date`, and `ring_topology`. A subset or an edited copy of a `cacs_isochrone()` result is therefore computed again, while the same areas in another row order use the saved result. `options(catchmentACS.cache_intersect = FALSE)` turns this cache off.

Value

A tibble with one row for each site, drive time, and ACS variable, sorted by `site_id`, `drive_time_min`, and `variable`, followed by one row for each site and drive-time pair with no tract left (see Details). It has the columns that `cacs_run()` returns in its long form, described in its Value section, except `est_total`, `var_total_raw`, `est_mean`, and `var_mean_raw`, which `cacs_propagate_moe()` adds. The main columns are:

`estimate`, `moe` The estimate and its margin of error, at the 90 percent level (see Details).

`weight_sum`, `n_tracts` The sum of the coverage weights of the tracts combined, also on the rows for medians and per-person values, and the number of those tracts, including any with a missing estimate.

`estimand_family`, `weight_basis` The kind of quantity and the weights used: "spatial_total" (a count) with "coverage"; "median_proxy" (a median) or "area_weighted_scalar_proxy" (a per-person value) with "area_mean"; or "metadata_only" (a code that is not combined) with "none", and `cacs_propagate_moe()` gives an error for such rows.

`failure_origin` "none", or "intersection" on the row of a pair with no tract left.

The result has these attributes:

`cacs_aggregation_carriers` A table of the weighted sums and averages of the tract estimates, with their variances, for each site, drive time, and variable. `cacs_propagate_moe()` and `cacs_derive_rates()` read it, and `cacs_derive_rates()` removes it.

`cacs_aggregation_provenance` A list recording `weight_method`, `min_weight`, `acs_year`, the time the result was computed, the versions of `catchmentACS`, `sf`, `GEOS`, and `PROJ`, and counts of variables and of missing estimates. `n_sites_input` is the number of site and drive-time pairs, and `n_sites_with_data` and `n_sites_empty` are the numbers of rows of the result with and without tracts.

`skipped_geoids` The GEOID of each row of `acs_sf` skipped in step 3 of Details, so a skipped tract appears once for each of its variables; an empty character vector when nothing is skipped.

It also has `cacs_schema_version`, the version label ("1.0") of the column layout, and, with `keep_tract_audit = TRUE`, `cacs_tract_audit` (see that argument).

Coverage weights and area shares

For one site, drive time, and variable, let a_i be the area of the overlap between tract i and the drive-time area and A_i the area of the tract. The tracts are those kept in step 4 of Details that have a row for the variable in `acs_sf`; a tract without such a row is left out, without a warning. A count then does not include that tract, and `n_tracts` on the rows of that variable is smaller than on the rows of

the variables that have a row for the tract (`cacs_acs_validate()` describes a check). Counts are combined with the coverage weights (`weight_basis = "coverage"`)

$$c_i = \frac{a_i}{A_i},$$

and medians and per-person values with the area shares (`weight_basis = "area_mean"`)

$$s_i = \frac{a_i}{\sum_k a_k},$$

which sum to one.

Totals kept for later steps

The `cacs_aggregation_carriers` attribute has the columns `site_id`, `drive_time_min`, `variable`, `estimand_family`, `weight_sum`, and `n_tracts`, with the same values as in the result. It has four more: `est_total`, the sum $\sum_i c_i X_i$ of the tract estimates X_i ; `est_mean`, the average $\sum_i s_i X_i$; and `var_total_raw` and `var_mean_raw`, their variances (the squares of their standard errors). The sum and the average are computed for every variable; `estimate` and `moe` come from the sum for counts and from the average for medians and per-person values.

See Also

`cacs_describe()` prints a summary drawn from the attributes of the result. Area weighting is explained at more length in `vignette("theory-spatial-aggregation", package = "catchmentACS")`, online at <https://joonho112.github.io/catchmentACS/articles/theory-spatial-aggregation.html>.

Other steps of the calculation: `cacs_acs_prefetch()`, `cacs_derive_rates()`, `cacs_isochrone()`, `cacs_propagate_moe()`, `cacs_run()`

Examples

```
# Turn the cache off while this example runs (see ?cacs_set_cache).
old <- options(catchmentACS.cache_enabled = FALSE)

# Example data bundled with the package: the drive-time areas are circles
# with a radius of 1 km per minute, and the ACS data are made up.
library(sf)
iso <- readRDS(system.file("extdata", "legacy_2025_isochrones.rds",
                           package = "catchmentACS"))
acs <- readRDS(system.file("extdata", "sample_alabama_subset.rds",
                           package = "catchmentACS"))
iso_07 <- iso[iso$site_id == "AL_SITE_07" & iso$drive_time_min == 10, ]
agg <- cacs_intersect_weight(iso_sf = iso_07, acs_sf = acs,
                             keep_tract_audit = TRUE, verbose = FALSE)

# The coverage weight and the area share of each tract in the area (the
# three tracts have the same area, so here the area shares are the
# coverage weights divided by their sum)
tracts <- attr(agg, "cacs_tract_audit")
tracts$area_share <- tracts$int_area_m2 / sum(tracts$int_area_m2)
```

```

tracts[, c("GEOID", "area_wt", "area_share")]

# A count, a median, and a per-person value, with the weights each one uses
agg[agg$variable %in% c("B01003_001", "B19013_001", "B19301_001"),
     c("variable", "estimate", "moe", "weight_basis", "n_tracts")]

options(old)

```

cacs_isochrone

Build drive-time areas around sites with a routing service

Description

Builds a drive-time area around each site for each drive time, using a routing service: the Open Source Routing Machine (OSRM, the default) or openrouteservice. Each area, also called an isochrone, covers the places that the routing service finds reachable from the site within that many minutes, so the 10-minute area includes the 5-minute area.

Usage

```

cacs_isochrone(
  sites,
  drive_times = c(5, 10, 15),
  provider = c("osrm", "ors", "mapbox", "r5r"),
  profile = "car",
  osrm_mode = c("demo", "docker"),
  ors_api_key = Sys.getenv("ORS_API_KEY"),
  mapbox_token = Sys.getenv("MAPBOX_TOKEN"),
  r5r_core = NULL,
  osm_snapshot_date = NULL,
  cache_dir = NULL,
  verbose = TRUE,
  ...
)

```

Arguments

sites	An sf object of points in EPSG:4326, or a data frame with numeric columns lon and lat (longitude and latitude). An sf object in another coordinate reference system, including NAD83 (EPSG:4269), gives an error. Either form needs a column site_id with a different value for each site; a numeric site_id becomes a string in the result. Without a site_id column, a column point_id is used as site_id, with a warning.
drive_times	A numeric vector of drive times in minutes: at most six values, each a whole number from 1 to 60; the default is c(5, 10, 15). The values can be in any order, and a repeated value is used once.

provider	A string giving the routing service: "osrm" (the default) for OSRM or "ors" for openrouteservice; see the "Provider status" section. "mapbox" and "r5r" are also accepted, but they are not implemented yet; using them gives an error of class <code>catchmentACS_error_credential</code> , after the other arguments are checked and before any request is sent.
profile	A string giving the routing profile, such as "car" (the default). With OSRM, the public demo server offers "car", "bike", and "foot"; with a profile that is not available, routing fails for every site and the error is recorded in <code>failure_reason</code> . With openrouteservice, the accepted values are the openrouteservice profiles "driving-car", "driving-hgv", "cycling-regular", "cycling-road", "cycling-mountain", "cycling-electric", "foot-walking", "foot-hiking", and "wheelchair", and the short names "car", "hgv", "bike" (for "cycling-regular"), "foot", and "walk" (both for "foot-walking"); any other value stops the function with the R error "subscript out of bounds".
osrm_mode	A string giving where OSRM requests are sent: "demo" (the default) for the public OSRM demo server or "docker" for a local OSRM server. It also sets the default res; see the OSRM sections below.
ors_api_key	A string giving the openrouteservice API key, by default the value of the <code>ORS_API_KEY</code> environment variable. It is used only with provider = "ors"; see the "Provider status" section.
mapbox_token	A string, by default the value of the <code>MAPBOX_TOKEN</code> environment variable. It is not used (see provider), but a value that is not a single string, such as NULL, gives an error.
r5r_core	An r5r core object, or NULL (the default). It is not used (see provider), but its class is checked.
osm_snapshot_date	A date to record as the date of the OpenStreetMap data used by the routing service: a Date object or a string in the form year-month-day, such as "2025-04-01", or NULL (the default). The date is only recorded in the result; it is not sent to the routing service and does not change the areas. A call with a different date does not reuse a saved result.
cache_dir	A path to the cache folder, or NULL (the default) to use <code>cacs_cache_dir()</code> . A folder outside the temporary folder of the R session is tidied as described in <code>cacs_cache_dir()</code> .
verbose	A logical value. With TRUE (the default), the function shows a summary line when the step finishes and, with five or more sites, a line as each site finishes. See the "Progress messages" section of <code>cacs_run()</code> for how to turn them off or show the lines for any number of sites. The function shows the message that a saved result is returned, and the notices about the default res, even when verbose is FALSE.
...	Additional named arguments, which depend on provider. With OSRM, the accepted names are res, which sets how detailed the areas are (see the OSRM sections below); osrm.server, the address of an OSRM server to use instead of the one chosen by osrm_mode; and osrm.profile or osrm.profile_name, a profile name to use in the requests instead of profile. With openrouteservice, the accepted names are attributes, area_units, and smoothing, which are

passed on to `openrouteservice::ors_isochrones()`. Other arguments give a warning and are not used.

Details

Routing for a site is tried up to three times, and not again after an HTTP 4xx error such as a bad request. A site for which routing fails keeps its rows, with an empty geometry and the error message in `failure_reason`; warnings report the number of rows that failed or have no area. If the routing service answers that its request limit has been reached (HTTP status 429), the function stops with an error as soon as that answer comes, sends no requests for the remaining sites, and returns no result. It does the same when the service refuses access (HTTP status 401 or 403), for example because an API key is not accepted.

The requests have no time limit of their own: a server that accepts the connection but does not answer makes the function wait, and a server that cannot be reached is tried three times for each site before the site is reported as failed. `cacs_validate_osrm_endpoint()` checks an OSRM server with a time limit and can be called first.

Unless the cache is turned off with `cacs_set_cache()`, the result is saved in the cache folder (`cache_dir`, or `cacs_cache_dir()`), which by default lasts only for the R session. A later call with the same arguments (other than `verbose`) and option settings returns the saved result without contacting the routing service, as long as the installed versions of `sf`, `PROJ`, and `GEOS` have not changed. A result is not saved when routing failed for a site because the service did not answer, timed out, or answered with an HTTP 5xx error, since a later call may succeed; a saved result with such a failure is deleted, and its sites are routed again. A site that the service refused with another HTTP 4xx error, such as a bad request, is saved with its `failure_reason`, and a warning is given each time the saved result is used. `cacs_clear_cache()` with `namespace = "isochrone"` removes only the results saved in the folder returned by `cacs_cache_dir()`, whose help page describes the cache folder, how long saved results are kept, and how they are checked.

Value

An `sf` tibble with one row for each site and drive time, in the order of sites and then by increasing drive time, and 16 columns:

- `site_id`: the site identifier, converted to a string.
- `drive_time_min`: the drive time in minutes.
- `geometry`: the drive-time area, of geometry type `POLYGON` or `MULTIPOLYGON`, in longitude and latitude (EPSG:4326); empty when there is no area.
- `provider`: the routing service used, `"osrm"` or `"ors"`.
- `profile`: the routing profile, such as `"car"`.
- `osm_snapshot_date`: the date given in `osm_snapshot_date`, as a string, or `"unknown"`.
- `routing_engine_version`: `"osrm-pkg/"` or `"openrouteservice-pkg/"` followed by the version of the R package that sent the requests (not the version of the routing server).
- `polygon_simplification_tolerance`: always `NA`.
- `provider_requested`: the service asked for, always the same as `provider`.
- `provider_downgrade`: whether another service was used instead, always `FALSE`.

- `generated_at`: the time the areas were built; a result read from the cache keeps the time of the original call.
- `isochrone_empty`: TRUE when the row has no area, because routing failed or because the service returned no area for that drive time (with OSRM, for example, when the site is too far from the road network).
- `osm_snapshot_status`: "user_supplied" when `osm_snapshot_date` was given, otherwise "unknown_best_effort".
- `failure_reason`: NA when routing succeeded, otherwise the error message from the last attempt.
- `retry_count`: the number of attempts made for the site, counting the first (1 to 3).
- `ring_topology`: always "cumulative", meaning that each area includes the areas of the shorter drive times.

The attributes `cacs_isochrone_provenance` and `cacs_provenance` hold the same list, a record of how the result was produced: the cache key, provider, profile, `osrm_mode`, the snapshot date and status, `ring_topology`, and, for OSRM, the `res` value, the OSRM server and profile, and an estimate of the requests per site (`osrm_request_budget`). The attribute `cacs_res_param` holds the `res` value (NA for openrouteservice). Rows selected from the result keep these attributes, including the cache key that `cacs_intersect_weight()` uses to recognize the areas in its own cache.

Provider status

`provider = "osrm"` (the default) uses OSRM through the `osrm` package and needs no API key; `osrm_mode` chooses the server (see the OSRM sections below). `provider = "ors"` uses openrouteservice through the `openrouteservice` package and needs an API key, given in `ors_api_key`; if the key is empty, the function stops with an error of class `catchmentACS_error_credential` before any request is sent. The openrouteservice route has been checked only with simulated responses from openrouteservice, not with the service itself. The two services compute the areas in different ways, so the area for the same site and drive time can differ between them.

OSRM servers

With `osrm_mode = "demo"`, the requests go to the public OSRM demo server at <https://routing.openstreetmap.de/>; with `osrm_mode = "docker"`, they go to <http://0.0.0.0:5000/>, or to the address set in the option `catchmentACS.osrm_docker_server`. An address given in `osrm.server` (see ...) is used instead with either mode. With `osrm_mode = "demo"`, the `osrm` package's own option `osrm.server` is also used. The `osrm` package sets that option to the demo server when it is loaded, for example by `library(osrm)`, replacing a value set earlier; a value set before a call to this function is kept, even when the call loads `osrm`.

OSRM grid resolution

The `osrm` package draws the areas of a site from travel times to a grid of `res` by `res` points around it, sized for the longest drive time, so the areas for shorter drive times rest on fewer points. A larger `res` gives more detailed areas but needs more requests and more time. On the demo server, the `osrm` package waits one second after every 75 grid points that it sends, which comes to about 10 seconds for each site at `res = 30L` and about a minute at `res = 70L`. There is no such wait with other servers.

If `res` is not given, it is 70L with `osrm_mode = "docker"`. The public demo server is protected by resolving `res` to 30L with `osrm_mode = "demo"`. With the lower value, fewer requests are sent for each site, so the server is less likely to answer that its request limit has been reached (HTTP status 429). With options(`catchmentACS.osrm_demo_budget_protect = FALSE`), `res` is 70L with either mode. A message reports the value used, once per R session; its class is `catchmentACS_message_demo_budget_protected` for 30L and `catchmentACS_message_res_default_changed` for 70L. When `res` is given, such as `res = 30L`, `res = 50L`, or `res = 70L`, that value is used and neither message is shown.

From version 5.0.0, the `osrm` package may show the message "'res' is deprecated, use 'n' instead." for each site. The `res` value is still used, and `n` is not accepted by `cacs_isochrone()`.

Areas built with another server, other OpenStreetMap data, or another `res` can differ, and so can the estimates computed from them.

See Also

`cacs_validate_osrm_endpoint()` checks whether an OSRM server is accepting requests, and `cacs_validate_iso()` checks whether drive-time areas made with other tools are in the form that the package expects. The routing services, and how to use a local OSRM server, are described in `vignette("providers", package = "catchmentACS")` (<https://joonho112.github.io/catchmentACS/articles/providers.html>).

Other steps of the calculation: `cacs_acs_prefetch()`, `cacs_derive_rates()`, `cacs_intersect_weight()`, `cacs_propagate_moe()`, `cacs_run()`

Examples

```
# The 10-minute area that cacs_isochrone() built for one site in
# Birmingham, Alabama, with the public OSRM demo server, kept in a file that
# comes with the package
iso_bhm <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                             package = "catchmentACS"))$iso_sf
sf::st_drop_geometry(iso_bhm)[, c("site_id", "drive_time_min", "provider",
                                   "isochrone_empty", "failure_reason")]

# A table with no rows: the area has the form that the package expects
cacs_validate_iso(iso_bhm)

# These calls send several dozen requests to the public OSRM demo server,
# which limits how many it accepts.
## Not run:
library(sf)
# The site at the center of the drive-time areas for AL_SITE_07 that come
# with the package (used in the examples of cacs_run() and others)
site_07 <- data.frame(site_id = "AL_SITE_07", lon = -85.365, lat = 31.655)

# 5-, 10-, and 15-minute areas from the public OSRM demo server
iso <- cacs_isochrone(site_07, drive_times = c(5, 10, 15))
st_drop_geometry(iso)[, c("site_id", "drive_time_min", "isochrone_empty")]

# A finer grid gives more detailed areas but takes longer
iso_fine <- cacs_isochrone(site_07, drive_times = c(5, 10, 15), res = 50L)
```

```
## End(Not run)
```

cacs_moe_to_se

Convert a margin of error to a standard error

Description

Divides a margin of error (MOE) by the normal quantile z for the level, giving the standard error (SE): $SE = MOE/z$. A margin of error is the half-width of a confidence interval. When level is 0.90, the level of published American Community Survey (ACS) margins of error, the function uses $z = 1.645$, the value the Census Bureau uses. At other levels it uses $qnorm(1 - (1 - level) / 2)$. The function is the inverse of [cacs_se_to_moe\(\)](#) at the same level.

Usage

```
cacs_moe_to_se(moe, level = 0.9)
```

Arguments

moe	A numeric vector of margins of error.
level	A single number between 0 and 1 (not a percentage) giving the confidence level at which the margins of error were computed. The default is 0.9. The value is not checked: level = 90, for example, gives NaN with a warning.

Details

ACS 1-year margins of error for 2005 and earlier were published with $z = 1.65$; dividing them by 1.65 gives the standard error. The values of moe are not checked. The negative codes that the Census Bureau's data API puts in place of some margins of error, such as -55555555, are divided like any other number and give negative standard errors. [cacs_acs_prefetch\(\)](#) returns these codes as NA.

Value

A numeric vector of standard errors, the same length as moe.

References

U.S. Census Bureau (2020). *Understanding and Using American Community Survey Data: What All Data Users Need to Know*. Chapter 7, Understanding Error and Determining Statistical Significance.

See Also

[cacs_propagate_moe\(\)](#) computes margins of error for drive-time area estimates.

Other rates and margins of error: [cacs_acs_default_rates](#), [cacs_se_to_moe\(\)](#)

Examples

```
cacs_moe_to_se(c(8.225, 16.45, NA), level = 0.90)
```

```
cacs_plot_site_intersection
```

Map the tracts that overlap a drive-time area (map 2 of 4)

Description

Draws an interactive leaflet map of one drive-time area of a site and the census tracts it overlaps. The part of each tract inside the area is shaded by the tract's coverage weight, labeled `area_wt` on the map: the share of the tract's area that lies inside the drive-time area. The color scale runs from 0 to 1 on every map, so a weight has the same color on all of them. The outline of the area and the site's point are drawn on top.

Usage

```
cacs_plot_site_intersection(
  site_id = NULL,
  lat = NULL,
  lon = NULL,
  site_name = NULL,
  iso_sf,
  tract_sf,
  sites_df = NULL,
  tiles = "OpenStreetMap",
  padding_km = 5,
  drive_time_min = NULL,
  ...
)
```

Arguments

<code>site_id</code>	A string giving the <code>site_id</code> of the site to map, used to find its point in <code>sites_df</code> and its drive-time areas in <code>iso_sf</code> .
<code>lat, lon</code>	Single numbers giving the latitude and longitude of a point in degrees (WGS 84), used when <code>site_id</code> is <code>NULL</code> . Both must be given. The point is marked at that location, and the areas of every site in <code>iso_sf</code> are used.
<code>site_name</code>	A string giving the name of the site, compared with the <code>site_name</code> column of <code>sites_df</code> including case, and used when <code>site_id</code> , <code>lat</code> , and <code>lon</code> are <code>NULL</code> . It must match one row.
<code>iso_sf</code>	An <code>sf</code> object of drive-time areas in EPSG:4326 (longitude and latitude), such as the result of <code>cacs_isochrone()</code> , with the columns <code>site_id</code> and <code>drive_time_min</code> .
<code>tract_sf</code>	An <code>sf</code> object of census tract polygons with a <code>GEOID</code> column, such as the result of <code>cacs_acs_prefetch()</code> . When a tract has several rows, as in that result (one for each variable), the first row is used.

sites_df	An sf object of site points in longitude and latitude with a site_id column, or NULL (the default) for <code>cacs_alabama_sites</code> . The bundled file <code>legacy_2025_isochrones.rds</code> uses the same site_id values for other places, so with its areas the default puts the point and the initial view far from them.
tiles	A string naming the background map, one of the tile providers in <code>leaflet::providers</code> ; the default is "OpenStreetMap", the standard OpenStreetMap map, which needs no API key. The map is built offline, and the tiles are downloaded when it is displayed. The CARTO tiles, such as "CartoDB.Positron", need an API key, which these functions do not send, so they show a notice asking for one.
padding_km	A single positive number. The initial view shows at least this many kilometers on each side of the site; the default is 5.
drive_time_min	A single number giving the drive time, in minutes, of the area to map, or NULL (the default) for the longest drive time of the site in <code>iso_sf</code> . It must be one of the drive times in <code>iso_sf</code> .
...	Not used. Any argument given here is ignored.

Details

`cacs_intersect_weight()` adds up counts with the same coverage weights, and combines medians and per-person values with area shares instead (each tract's share of the total overlap area). This function computes the overlaps itself from `iso_sf` and `tract_sf`, measuring areas in the equal-area projection EPSG:5070 as `cacs_intersect_weight()` does, but without its `min_weight` cut-off.

Clicking a tract shows its GEOID (the tract's Census identifier), its coverage weight, and the area of its part inside the drive-time area in square kilometers (`intersection_km2`).

With `lat` and `lon` instead of `site_id`, the rows of `iso_sf` are not narrowed to one site. The drive time is then chosen among the drive times of all the sites in `iso_sf`, and the area of every site for that drive time is drawn, so a map of one area needs an `iso_sf` that holds the rows of one site. `cacs_plot_site_rates()` and `cacs_plot_site_pipeline()` differ: they map the site in `sites_df` nearest to the point.

Value

A leaflet map (an HTML widget).

See Also

`vignette("visual-walkthrough", package = "catchmentACS")` (<https://joonho112.github.io/catchmentACS/articles/visual-walkthrough.html>) builds all four maps for one site, one section each.

Other maps of one site: `cacs_plot_site_isochrone()`, `cacs_plot_site_pipeline()`, `cacs_plot_site_rates()`, `cacs_plot_site_weighted()`, `format.cacs_site_plot_pipeline()`, `print.cacs_site_plot_pipeline()`

Examples

```
if (requireNamespace("leaflet", quietly = TRUE)) {
  # Example data bundled with the package for one site in Birmingham,
  # Alabama: a 10-minute drive-time area from the OSRM routing service,
  # 2019-2023 ACS estimates for nearby tracts, and the cacs_run() result.
```

```

fx <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                          package = "catchmentACS"))
cacs_plot_site_intersection(site_id = "AL_BHM_01", iso_sf = fx$iso_sf,
                           tract_sf = fx$tract_sf,
                           sites_df = fx$sites_df)
}

```

cacs_plot_site_isochrone

Map a site and its drive-time areas (map 1 of 4)

Description

Draws an interactive leaflet map of one site and its drive-time areas (isochrones), the areas reachable from the site within each drive time. It is the first of the four maps that [cacs_plot_site_pipeline\(\)](#) builds.

Usage

```

cacs_plot_site_isochrone(
  site_id = NULL,
  lat = NULL,
  lon = NULL,
  site_name = NULL,
  iso_sf,
  sites_df = NULL,
  tiles = "OpenStreetMap",
  padding_km = 5,
  ...
)

```

Arguments

site_id	A string giving the site_id of the site to map, used to find its point in sites_df and its drive-time areas in iso_sf.
lat, lon	Single numbers giving the latitude and longitude of a point in degrees (WGS 84), used when site_id is NULL. Both must be given. The point is marked at that location, and the areas of every site in iso_sf are used.
site_name	A string giving the name of the site, compared with the site_name column of sites_df including case, and used when site_id, lat, and lon are NULL. It must match one row.
iso_sf	An sf object of drive-time areas in EPSG:4326 (longitude and latitude), such as the result of cacs_isochrone() , with the columns site_id and drive_time_min.
sites_df	An sf object of site points in longitude and latitude with a site_id column, or NULL (the default) for cacs_alabama_sites . The bundled file legacy_2025_isochrones.rds uses the same site_id values for other places, so with its areas the default puts the point and the initial view far from them.

tiles	A string naming the background map, one of the tile providers in <code>leaflet::providers</code> ; the default is "OpenStreetMap", the standard OpenStreetMap map, which needs no API key. The map is built offline, and the tiles are downloaded when it is displayed. The CARTO tiles, such as "CartoDB.Positron", need an API key, which these functions do not send, so they show a notice asking for one.
padding_km	A single positive number. The initial view shows at least this many kilometers on each side of the site; the default is 5.
...	Not used. Any argument given here is ignored.

Details

The site is drawn as a point. Each drive time in the `drive_time_min` column of `iso_sf` has its own color, shown in a legend; without that column, all the areas are drawn in one color.

Value

A leaflet map (an HTML widget).

See Also

`vignette("visual-walkthrough", package = "catchmentACS")` (<https://joonho112.github.io/catchmentACS/articles/visual-walkthrough.html>) builds all four maps for one site, one section each.

Other maps of one site: `cacs_plot_site_intersection()`, `cacs_plot_site_pipeline()`, `cacs_plot_site_rates()`, `cacs_plot_site_weighted()`, `format.cacs_site_plot_pipeline()`, `print.cacs_site_plot_pipeline()`

Examples

```
if (requireNamespace("leaflet", quietly = TRUE)) {
  # Example data bundled with the package for one site in Birmingham,
  # Alabama: a 10-minute drive-time area from the OSRM routing service,
  # 2019-2023 ACS estimates for nearby tracts, and the cacs_run() result.
  fx <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                           package = "catchmentACS"))
  cacs_plot_site_isochrone(site_id = "AL_BHM_01", iso_sf = fx$iso_sf,
                          sites_df = fx$sites_df)
}
```

cacs_plot_site_pipeline

Build all four maps for one site

Description

Builds, for one site, the four maps that show step by step how the package computes its estimates from American Community Survey (ACS) data, and returns them in one list. The maps come from [cacs_plot_site_isochrone\(\)](#) (the site and its drive-time areas), [cacs_plot_site_intersection\(\)](#) (the census tracts that overlap one area, shaded by coverage weight), [cacs_plot_site_weighted\(\)](#) (one ACS variable over those tracts), and [cacs_plot_site_rates\(\)](#) (the five rates). A tract's coverage weight is the share of its area inside the drive-time area.

Usage

```
cacs_plot_site_pipeline(
  site_id = NULL,
  lat = NULL,
  lon = NULL,
  site_name = NULL,
  iso_sf,
  tract_sf,
  acs_sf,
  run_result,
  sites_df = NULL,
  variable = "B17001_002",
  variable_family = "spatial_total",
  tiles = "OpenStreetMap",
  padding_km = 5,
  drive_time_min = NULL,
  ...
)
```

Arguments

site_id	A string giving the site_id of the site to map, used to find its point in sites_df and its drive-time areas in iso_sf.
lat, lon	Single numbers giving the latitude and longitude of a point in degrees (WGS 84), used when site_id is NULL. The nearest site in sites_df is found as in cacs_plot_site_rates() , with one message, and used for all four maps.
site_name	A string giving the name of the site, compared with the site_name column of sites_df including case, and used when site_id, lat, and lon are NULL. It must match one row.
iso_sf	An sf object of drive-time areas in EPSG:4326 (longitude and latitude), such as the result of cacs_isochrone() , with the columns site_id and drive_time_min.
tract_sf	An sf object of census tract polygons with a GEOID column, such as the result of cacs_acs_prefetch() . When a tract has several rows, as in that result (one for each variable), the first row is used.
acs_sf	A data frame or sf object of tract estimates in the long form of the result of cacs_acs_prefetch() , with the columns GEOID, variable, estimate, and moe. It is often the same object as tract_sf. The estimates are mapped as

	they are: unlike <code>cacs_intersect_weight()</code> , the map does not set Census Bureau annotation codes, such as -6666666666, to NA, and it draws a tract with two rows for the variable twice instead of giving an error.
<code>run_result</code>	A data frame of rates with the columns <code>site_id</code> , <code>variable</code> , <code>estimate</code> , <code>moe</code> , <code>n_tracts_num</code> , <code>n_tracts_den</code> , and <code>moe_fallback</code> , such as the long form of a <code>cacs_run()</code> result or the result of <code>cacs_derive_rates()</code> . The rows of the chosen site for the five rates are used, and there must be at least one.
<code>sites_df</code>	An sf object of site points in longitude and latitude with a <code>site_id</code> column, or NULL (the default) for <code>cacs_alabama_sites</code> . The bundled file <code>legacy_2025_isochrones.rds</code> uses the same <code>site_id</code> values for other places, so with its areas the default puts the point and the initial view far from them.
<code>variable</code>	A string giving the ACS variable code to map, one of the values in the <code>variable</code> column of <code>acs_sf</code> . The default is "B17001_002", the number of people whose income in the past 12 months was below the poverty level.
<code>variable_family</code>	A string giving the kind of quantity that <code>variable</code> is, as in the <code>estimand_family</code> column of <code>cacs_run()</code> results. It only chooses the ColorBrewer palette: "Greens" for "spatial_total" (a count; the default), "PuOr" for "median_proxy" (a median), "YlOrRd" for "area_weighted_scalar_proxy" (a per-person value), and "RdYlBu" for "derived_rate" (a rate). NA and unrecognized values give "YlGnBu", and "metadata_only" gives an error. The value is not checked against <code>variable</code> .
<code>tiles</code>	A string naming the background map, one of the tile providers in <code>leaflet::providers</code> ; the default is "OpenStreetMap", the standard OpenStreetMap map, which needs no API key. The map is built offline, and the tiles are downloaded when it is displayed. The CARTO tiles, such as "CartoDB.Positron", need an API key, which these functions do not send, so they show a notice asking for one.
<code>padding_km</code>	A single positive number. The initial view shows at least this many kilometers on each side of the site; the default is 5.
<code>drive_time_min</code>	A single number giving the drive time, in minutes, of the area in the second and third maps, or NULL (the default) for the longest drive time of the site in <code>iso_sf</code> . The first map draws all the areas of the site, and the fourth their outline.
<code>...</code>	Passed to the four map functions, which do not use them.

Details

The site is looked up once, before any map is built, and used for all four. `iso_sf`, `sites_df`, `tiles`, `padding_km`, and `...` are passed to all four functions, `tract_sf` and `drive_time_min` to the second and third, `acs_sf`, `variable`, and `variable_family` to the third, and `run_result` to the fourth.

The function does not contact a routing service or the Census Bureau. Its inputs are results computed beforehand: the drive-time areas by `cacs_isochrone()`, the tract estimates by `cacs_acs_prefetch()`, and the rates by `cacs_run()`.

`cacs_plot_site_rates` *Map a site with its rates and margins of error (map 4 of 4)*

Description

Draws an interactive leaflet map of a site and the outline of its drive-time areas in `iso_sf`, merged into one. Clicking the site's point opens a pop-up that lists the site's values from `run_result` for the five rates in `cacs_acs_default_rates`.

Usage

```
cacs_plot_site_rates(
  site_id = NULL,
  lat = NULL,
  lon = NULL,
  site_name = NULL,
  iso_sf,
  run_result,
  sites_df = NULL,
  tiles = "OpenStreetMap",
  padding_km = 5,
  ...
)
```

Arguments

<code>site_id</code>	A string giving the <code>site_id</code> of the site to map, used to find its point in <code>sites_df</code> and its drive-time areas in <code>iso_sf</code> .
<code>lat, lon</code>	Single numbers giving the latitude and longitude of a point in degrees (WGS 84), used when <code>site_id</code> is <code>NULL</code> . Both must be given. The site in <code>sites_df</code> nearest to the point is mapped, and a message of class <code>catchmentACS_message_resolve_site</code> gives its <code>site_id</code> and its distance from the point. If that site has no rows in <code>run_result</code> , the function stops with an error. If it is more than 5 km from the point, a warning of class <code>catchmentACS_warning_resolve_site_distant</code> is also given.
<code>site_name</code>	A string giving the name of the site, compared with the <code>site_name</code> column of <code>sites_df</code> including case, and used when <code>site_id</code> , <code>lat</code> , and <code>lon</code> are <code>NULL</code> . It must match one row.
<code>iso_sf</code>	An <code>sf</code> object of drive-time areas in EPSG:4326 (longitude and latitude), such as the result of <code>cacs_isochrone()</code> , with the columns <code>site_id</code> and <code>drive_time_min</code> .
<code>run_result</code>	A data frame of rates with the columns <code>site_id</code> , <code>variable</code> , <code>estimate</code> , <code>moe</code> , <code>n_tracts_num</code> , <code>n_tracts_den</code> , and <code>moe_fallback</code> , such as the long form of a <code>cacs_run()</code> result or the result of <code>cacs_derive_rates()</code> . The rows of the chosen site for the five rates are used, and there must be at least one.

sites_df	An sf object of site points in longitude and latitude with a site_id column, or NULL (the default) for <code>cacs_alabama_sites</code> . The bundled file <code>legacy_2025_isochrones.rds</code> uses the same site_id values for other places, so with its areas the default puts the point and the initial view far from them.
tiles	A string naming the background map, one of the tile providers in <code>leaflet::providers</code> ; the default is "OpenStreetMap", the standard OpenStreetMap map, which needs no API key. The map is built offline, and the tiles are downloaded when it is displayed. The CARTO tiles, such as "CartoDB.Positron", need an API key, which these functions do not send, so they show a notice asking for one.
padding_km	A single positive number. The initial view shows at least this many kilometers on each side of the site; the default is 5.
...	Not used. Any argument given here is ignored.

Details

For each rate, the pop-up shows the estimate and its margin of error (MOE, the half-width of a 90 percent confidence interval unless another level was used for `run_result`), both as proportions to four decimal places. It also shows the numbers of tracts combined for the numerator and for the denominator (`n_tracts_num` and `n_tracts_den`, filled in by `cacs_derive_rates()`), the MOE formula used (`moe_formula_effective`), and whether the chosen formula could not be used (`moe_fallback`). A rate with `moe_fallback = TRUE` also has an asterisk before its name: either the ratio formula was used in place of the proportion formula, or the denominator is zero and the rate is NA (see `cacs_derive_rates()`).

When `run_result` has rates for more than one drive time of the site, each rate is listed once for each drive time, and the pop-up does not show the drive time.

Value

A leaflet map (an HTML widget).

See Also

`vignette("visual-walkthrough", package = "catchmentACS")` (<https://joonho112.github.io/catchmentACS/articles/visual-walkthrough.html>) builds all four maps for one site, one section each.

Other maps of one site: `cacs_plot_site_intersection()`, `cacs_plot_site_isochrone()`, `cacs_plot_site_pipeline()`, `cacs_plot_site_weighted()`, `format.cacs_site_plot_pipeline()`, `print.cacs_site_plot_pipeline()`

Examples

```
if (requireNamespace("leaflet", quietly = TRUE)) {
  # Example data bundled with the package for one site in Birmingham,
  # Alabama: a 10-minute drive-time area from the OSRM routing service,
  # 2019-2023 ACS estimates for nearby tracts, and the cacs_run() result.
  fx <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                           package = "catchmentACS"))
  # Clicking the site's point opens the pop-up with the five rates
  cacs_plot_site_rates(site_id = "AL_BHM_01", iso_sf = fx$iso_sf,
                      run_result = fx$run_result, sites_df = fx$sites_df)
```

```
}
```

```
cacs_plot_site_weighted
```

Map an ACS variable by tract in a drive-time area (map 3 of 4)

Description

Draws an interactive leaflet map of one American Community Survey (ACS) variable over the census tracts that overlap a drive-time area of a site. The part of each tract inside the area is shaded by the tract's estimate as it is in `acs_sf`, before any weighting. The outline of the area and the site's point are drawn on top.

Usage

```
cacs_plot_site_weighted(
  site_id = NULL,
  lat = NULL,
  lon = NULL,
  site_name = NULL,
  iso_sf,
  tract_sf,
  acs_sf,
  sites_df = NULL,
  variable = "B17001_002",
  tiles = "OpenStreetMap",
  padding_km = 5,
  drive_time_min = NULL,
  variable_family = "spatial_total",
  ...
)
```

Arguments

<code>site_id</code>	A string giving the <code>site_id</code> of the site to map, used to find its point in <code>sites_df</code> and its drive-time areas in <code>iso_sf</code> .
<code>lat, lon</code>	Single numbers giving the latitude and longitude of a point in degrees (WGS 84), used when <code>site_id</code> is <code>NULL</code> . Both must be given. The point is marked at that location, and the areas of every site in <code>iso_sf</code> are used.
<code>site_name</code>	A string giving the name of the site, compared with the <code>site_name</code> column of <code>sites_df</code> including case, and used when <code>site_id</code> , <code>lat</code> , and <code>lon</code> are <code>NULL</code> . It must match one row.
<code>iso_sf</code>	An <code>sf</code> object of drive-time areas in EPSG:4326 (longitude and latitude), such as the result of <code>cacs_isochrone()</code> , with the columns <code>site_id</code> and <code>drive_time_min</code> .
<code>tract_sf</code>	An <code>sf</code> object of census tract polygons with a <code>GEOID</code> column, such as the result of <code>cacs_acs_prefetch()</code> . When a tract has several rows, as in that result (one for each variable), the first row is used.

acs_sf	A data frame or sf object of tract estimates in the long form of the result of <code>cacs_acs_prefetch()</code> , with the columns GEOID, variable, estimate, and moe. It is often the same object as <code>tract_sf</code> . The estimates are mapped as they are: unlike <code>cacs_intersect_weight()</code> , the map does not set Census Bureau annotation codes, such as -6666666666, to NA, and it draws a tract with two rows for the variable twice instead of giving an error.
sites_df	An sf object of site points in longitude and latitude with a <code>site_id</code> column, or NULL (the default) for <code>cacs_alabama_sites</code> . The bundled file <code>legacy_2025_isochrones.rds</code> uses the same <code>site_id</code> values for other places, so with its areas the default puts the point and the initial view far from them.
variable	A string giving the ACS variable code to map, one of the values in the <code>variable</code> column of <code>acs_sf</code> . The default is "B17001_002", the number of people whose income in the past 12 months was below the poverty level.
tiles	A string naming the background map, one of the tile providers in <code>leaflet::providers</code> ; the default is "OpenStreetMap", the standard OpenStreetMap map, which needs no API key. The map is built offline, and the tiles are downloaded when it is displayed. The CARTO tiles, such as "CartoDB.Positron", need an API key, which these functions do not send, so they show a notice asking for one.
padding_km	A single positive number. The initial view shows at least this many kilometers on each side of the site; the default is 5.
drive_time_min	A single number giving the drive time, in minutes, of the area to map, or NULL (the default) for the longest drive time of the site in <code>iso_sf</code> . It must be one of the drive times in <code>iso_sf</code> .
variable_family	A string giving the kind of quantity that <code>variable</code> is, as in the <code>estimand_family</code> column of <code>cacs_run()</code> results. It only chooses the ColorBrewer palette: "Greens" for "spatial_total" (a count; the default), "PuOr" for "median_proxy" (a median), "YlOrRd" for "area_weighted_scalar_proxy" (a per-person value), and "RdYlBu" for "derived_rate" (a rate). NA and unrecognized values give "YlGnBu", and "metadata_only" gives an error. The value is not checked against <code>variable</code> .
...	Not used. Any argument given here is ignored.

Details

Clicking a tract shows the values that `cacs_plot_site_intersection()` shows: GEOID, the coverage weight `area_wt` (the share of the tract's area inside the drive-time area), and `intersection_km2` (the area of that part in square kilometers), with the variable code. It also shows the tract's estimate, its margin of error (moe, the half-width of the 90 percent confidence interval published with the estimate), and the coverage weight times the estimate, labeled "area_wt x estimate". For a count, the products add up to the estimate that `cacs_intersect_weight()` gives for the area, which is NA when any tract's estimate is missing. For a median or a per-person value, that estimate is instead an average of the tract estimates weighted by area shares (see `cacs_intersect_weight()`), and the products do not add up to it.

The estimates are grouped into color classes by natural breaks when the `classInt` package is installed and there are at least five different estimates, and by `pretty()` otherwise. A tract with a missing estimate, or with no row for `variable` in `acs_sf`, is gray.

With `lat` and `lon` instead of `site_id`, the rows of `iso_sf` are not narrowed to one site. The drive time is then chosen among the drive times of all the sites in `iso_sf`, and the area of every site for that drive time is drawn, so a map of one area needs an `iso_sf` that holds the rows of one site. `cacs_plot_site_rates()` and `cacs_plot_site_pipeline()` differ: they map the site in `sites_df` nearest to the point.

Value

A leaflet map (an HTML widget).

See Also

`vignette("visual-walkthrough", package = "catchmentACS")` (<https://joonho112.github.io/catchmentACS/articles/visual-walkthrough.html>) builds all four maps for one site, one section each.

Other maps of one site: `cacs_plot_site_intersection()`, `cacs_plot_site_isochrone()`, `cacs_plot_site_pipeline()`, `cacs_plot_site_rates()`, `format.cacs_site_plot_pipeline()`, `print.cacs_site_plot_pipeline()`

Examples

```
if (requireNamespace("leaflet", quietly = TRUE)) {
  # Example data bundled with the package for one site in Birmingham,
  # Alabama: a 10-minute drive-time area from the OSRM routing service,
  # 2019-2023 ACS estimates for nearby tracts, and the cacs_run() result.
  fx <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
    package = "catchmentACS"))
  # Median household income, a median
  cacs_plot_site_weighted(site_id = "AL_BHM_01", iso_sf = fx$iso_sf,
    tract_sf = fx$tract_sf, acs_sf = fx$acs_sf,
    variable = "B19013_001",
    variable_family = "median_proxy",
    sites_df = fx$sites_df)
}
```

<code>cacs_propagate_moe</code>	<i>Compute margins of error for drive-time area estimates</i>
---------------------------------	---

Description

Computes a margin of error (MOE) for each estimate in the output of `cacs_intersect_weight()`, at the confidence level given by `level`. A margin of error is the half-width of a confidence interval. The formula depends on the kind of estimate and is recorded in the `moe_formula_effective` column. American Community Survey (ACS) margins of error are published at the 90 percent level; at the default `level = 0.9`, the margins of error of counts, medians, and per-person values are the same as those that `cacs_intersect_weight()` returns.

Usage

```
cacs_propagate_moe(data, formula = NULL, level = 0.9, verbose = TRUE, ...)
```

Arguments

data	A tibble returned by <code>cacs_intersect_weight()</code> , from which rows may have been removed (for example with <code>[]</code> or <code>dplyr::filter()</code>) but whose columns and attributes are kept, including <code>cacs_aggregation_carriers</code> (a table of the weighted totals and means with their variances, which this function reads). Rows for ACS codes that <code>cacs_intersect_weight()</code> does not combine (<code>estimand_family</code> is "metadata_only"), such as codes of tables whose names begin with C or S, make the function stop with an error; it runs once those rows are removed. The row kept for a site and drive-time pair with no tracts is returned unchanged.
formula	A string, a list of strings named by values of <code>variable</code> , or NULL (the default), giving the formula for each row: "weighted_sum", "weighted_mean", "proportion_subset", or "general_ratio_conservative" (see the "MOE formula families" section). NULL gives each row the formula for its kind of estimate, and a string is used for every row. In a list, the rows not named keep the formula they would get with NULL. Each row accepts only the formula for its kind of estimate; any other formula, a name not found in <code>variable</code> , or an unknown formula gives an error.
level	A single number between 0 and 1 (not a percentage) giving the confidence level of the margins of error. The default is 0.9. Any other value, such as 0, 1, 90, NA, "0.9", or <code>c(0.9, 0.95)</code> , gives an error.
verbose	A logical value. With TRUE (the default), the function shows a one-line summary when the step finishes (see the "Progress messages" section of <code>cacs_run()</code>).
...	Not used. The name <code>fallback_chain_max</code> is accepted with a warning and has no effect; any other argument gives an error.

Details

The margins of error are combined as if the census tract estimates were independent. The Census Bureau's handbook for ACS data users notes that its approximation formulas leave out the covariance between estimates, so a margin of error can be too small or too large depending on the correlation between them (U.S. Census Bureau 2020, chapter 8). The margin of error of a count, a median, or a per-person value is too small if the tract estimates are positively correlated. For a rate, errors that move the numerator and the denominator in the same direction partly offset each other in the ratio, and the package does not compute the net effect for a given rate. The area-based weights of the tracts are treated as fixed, so the margins of error leave out two more sources of error. One is the assumption that whatever a variable counts is spread evenly over each tract's area. The other, for medians and per-person values, is the weighting of tracts by area.

Value

The tibble data with the same rows in the same order, in which `moe` (the margin of error) is recomputed at `level`. The columns that record how it was computed are rewritten: `moe_formula_requested` and `moe_formula_effective` (the formula chosen and the one used), and `moe_fallback` and `moe_fallback_reason` (whether the chosen formula could not be used, and why). Count, median, and per-person rows keep `moe_fallback` = FALSE even when their margin of error is NA. Four columns are added from the `cacs_aggregation_carriers` attribute: `est_total` and `var_total_raw`, the weighted sum of the tract estimates and its variance (used for counts), and `est_mean` and `var_mean_raw`, the weighted average and its variance (used for medians and per-person values).

With `options(cacs.return_se = TRUE)`, an `se` column is added too: the standard error, `moe` divided by 1.645 at the 90 percent level or by the normal quantile for another level.

The attributes of data are kept, and two are added. `cacs_confidence_level` is the value of `level`, which `cacs_derive_rates()` uses for the rates. `cacs_moe_provenance` is a list recording the confidence level, the formula argument, and the numbers of rows computed with Families A and B (see the "MOE formula families" section). Its counts for rates are always 0, because the rates are added later by `cacs_derive_rates()`, which counts them in its `cacs_rate_provenance` attribute.

MOE formula families

By default, each row gets the formula for the kind of estimate in its `estimand_family` column. The four formulas are called Families A, B, C1, and C2; the letters also appear in warnings and in the row counts of the `cacs_moe_provenance` attribute. The formulas give the margin of error at the 90 percent level, the level of the tract margins of error M_i ; at another level, the result is multiplied by $z/1.645$, where z is the normal quantile for the level, `qnorm(1 - (1 - level) / 2)`.

- Family A, recorded as "weighted_sum", is used for counts ("spatial_total"). The margin of error is $\sqrt{\sum_i (w_i M_i)^2}$, where w_i is the coverage weight of tract i , the share of its area inside the drive-time area. This is the handbook's formula for a sum, applied to the weighted tract estimates.
- Family B, recorded as "weighted_mean", is used for medians and per-person values ("median_proxy" and "area_weighted_scalar_proxy"). It is the same formula, with w_i proportional to the area that tract i shares with the drive-time area and summing to one. This is an approximation made by the package rather than one given in the handbook.
- Family C1, recorded as "proportion_subset", is the handbook's formula for a proportion, a ratio whose numerator is part of its denominator. For the rate $\hat{p} = \hat{X}_{\text{num}} / \hat{X}_{\text{den}}$ of two weighted counts with Family A margins of error M_{num} and M_{den} , the margin of error is $\sqrt{M_{\text{num}}^2 - \hat{p}^2 M_{\text{den}}^2} / \hat{X}_{\text{den}}$.
- Family C2, recorded as "general_ratio_conservative", is the handbook's formula for a ratio whose numerator is not part of its denominator: $\sqrt{M_{\text{num}}^2 + \hat{p}^2 M_{\text{den}}^2} / |\hat{X}_{\text{den}}|$. The package divides by the absolute value of the denominator; for the five rates, whose denominators are counts, that is the denominator itself. Like the other formulas it leaves out the correlation between tract estimates (see Details).

C1 and C2 are used for the rows of the five rates in `cacs_acs_default_rates` ("derived_rate"). By default every rate uses C2, the ratio formula. The numerator of each of the five rates is part of its denominator, so the rates are proportions, for which the handbook gives C1, the proportion formula. The two formulas differ by $2\hat{p}^2 M_{\text{den}}^2$ under the square root, so C2 never gives the narrower margin; how much wider it is depends on $\hat{p} M_{\text{den}}$ relative to M_{num} and varies from rate to rate.

If the value under the square root of C1 is negative, the row uses C2 instead, as the handbook advises (U.S. Census Bureau 2020, chapter 8). The substitution is recorded in the row: "general_ratio_conservative" in `moe_formula_effective`, TRUE in `moe_fallback`, and "negative_variance" in `moe_fallback_reason`. How rates that cannot be computed are marked is described in the "Rates that are NA" section of `cacs_derive_rates()`.

The output of `cacs_intersect_weight()` has no rate rows, so this function applies only Families A and B; `cacs_derive_rates()` applies C1 and C2 to the rates, as chosen by its `formula_dispatch` argument, at the level set here.

References

U.S. Census Bureau (2020). *Understanding and Using American Community Survey Data: What All Data Users Need to Know*. Chapter 8, Calculating Measures of Error for Derived Estimates.

See Also

`cacs_se_to_moe()` and `cacs_moe_to_se()` convert between margins of error and standard errors. The mathematics behind these formulas is in `vignette("theory-moe-propagation", package = "catchmentACS")` (<https://joonho112.github.io/catchmentACS/articles/theory-moe-propagation.html>).

Other steps of the calculation: `cacs_acs_prefetch()`, `cacs_derive_rates()`, `cacs_intersect_weight()`, `cacs_isochrone()`, `cacs_run()`

Examples

```
# Turn the cache off while this example runs (see ?cacs_set_cache).
old <- options(catchmentACS.cache_enabled = FALSE)

# Example data bundled with the package: the drive-time areas are circles
# with a radius of 1 km per minute, and the ACS data are made up.
library(sf)
iso <- readRDS(system.file("extdata", "legacy_2025_isochrones.rds",
                           package = "catchmentACS"))
acs <- readRDS(system.file("extdata", "sample_alabama_subset.rds",
                           package = "catchmentACS"))
iso_07 <- iso[iso$site_id == "AL_SITE_07" & iso$drive_time_min == 10, ]
agg <- cacs_intersect_weight(iso_sf = iso_07, acs_sf = acs,
                            verbose = FALSE)

# Margins of error at the default 90 percent level and at 95 percent
prop <- cacs_propagate_moe(agg, verbose = FALSE)
prop_95 <- cacs_propagate_moe(agg, level = 0.95, verbose = FALSE)
tibble::tibble(variable = prop$variable, estimate = prop$estimate,
               moe_90 = prop$moe, moe_95 = prop_95$moe,
               formula = prop$moe_formula_effective)

options(old)
```

cacs_rings_to_cumulative

Combine drive-time bands into cumulative drive-time areas

Description

Some routing tools return drive-time bands, also called rings: for each site, the area reachable in 0 to 5 minutes, the area reachable in 5 to 10 minutes, and so on. The package uses cumulative areas instead, in which the 10-minute area covers everything reachable within 10 minutes and so contains the 5-minute area. This function makes them by joining each band with the bands of the shorter drive times (their geometric union).

Usage

```
cacs_rings_to_cumulative(iso_sf, drive_times = NULL)
```

Arguments

iso_sf	An sf object of polygons in EPSG:4326 (longitude and latitude), with one row for each site and band and a site_id column of strings. The columns isomin and isomax give the drive times, in whole minutes, at which each band starts and ends: isomin must be 0 or more and isomax above isomin. A ring_topology column, if present, must hold only "annulus" (bands) or "cumulative"; it does not change the result.
drive_times	A numeric vector of drive times in minutes, each one of the isomax values, or NULL (the default) for all of them. Every site must have a band that ends at each of these drive times.

Details

For each site and drive time, the bands that end at or before that drive time are joined. If all of them start at 0, the rows are taken to be cumulative areas already; otherwise they must run from 0 to that drive time without a gap in minutes, or the function gives an error. The geometries are not compared, so bands that overlap or leave gaps between them are joined as they are.

Value

An sf object with one row for each site and drive time, sorted by site_id and then by drive time. The geometry, in a column named geometry, is the union of the bands up to that drive time. The other columns are copied from the band that ends at that drive time, except four: isomin is set to 0, isomax and drive_time_min to the drive time, and ring_topology to "cumulative". [cacs_intersect_weight\(\)](#) and [cacs_run\(\)](#) also need the other columns of a [cacs_isochrone\(\)](#) result; [cacs_validate_iso\(\)](#) lists any that are missing.

See Also

[cacs_isochrone\(\)](#), which returns cumulative areas, and [cacs_validate_iso\(\)](#), which checks drive-time areas from other sources.

Examples

```
# Two bands for one site: 0 to 5 minutes and 5 to 10 minutes
poly <- function(xmin, xmax, ymin, ymax) {
  sf::st_polygon(list(rbind(
    c(xmin, ymin),
    c(xmax, ymin),
    c(xmax, ymax),
    c(xmin, ymax),
    c(xmin, ymin)
  )))
}

bands <- sf::st_sf(
```

```

    site_id = c("A", "A"),
    isomin = c(0L, 5L),
    isomax = c(5L, 10L),
    geometry = sf::st_sfc(
      poly(0, 1, 0, 1),
      poly(1, 2, 0, 1),
      crs = 4326
    )
  )
)

# The 10-minute area is the union of the two bands
cumulative <- cacs_rings_to_cumulative(bands, drive_times = c(5, 10))
sf::st_drop_geometry(cumulative)[
  , c("site_id", "isomin", "isomax", "drive_time_min", "ring_topology")
]
unique(cumulative$ring_topology)

```

cacs_run

Compute ACS estimates for drive-time areas around sites

Description

Builds drive-time areas around each site, one for each drive time, and combines the American Community Survey (ACS) estimates of the census tracts that overlap each area. These areas are also called isochrones. Returns a table with an estimate and its margin of error (the half-width of a confidence interval) for every site, drive time, and variable, plus rows for five rates such as the poverty rate ([cacs_acs_default_rates](#)). The function runs [cacs_acs_prefetch\(\)](#), [cacs_isochrone\(\)](#), [cacs_intersect_weight\(\)](#), [cacs_propagate_moe\(\)](#), and [cacs_derive_rates\(\)](#) in that order; each of them can also be called on its own. Supplying `precomputed_isochrones` or `acs` skips the corresponding step, and with both supplied no routing service or Census API key is needed.

Usage

```

cacs_run(
  sites,
  state,
  year = 2023,
  drive_times = c(5, 10, 15),
  variables = NULL,
  provider = c("osrm", "ors", "mapbox", "r5r"),
  weight_method = c("area", "population"),
  moe_formula = NULL,
  output = c("long", "list_column", "both"),
  precomputed_isochrones = NULL,
  cache_dir = NULL,
  acs = NULL,
  bg_pop_sf = NULL,

```

```

rates = cacs_acs_default_rates,
formula_dispatch = "general_ratio_conservative",
iso_args = list(),
acs_args = list(),
weight_args = list(),
moe_args = list(),
rate_args = list(),
verbose = TRUE
)

```

Arguments

sites	A data frame with the columns <code>site_id</code> , <code>lon</code> , and <code>lat</code> , or an <code>sf</code> object of points with a <code>site_id</code> column, such as cacs_alabama_sites . The coordinates are longitude and latitude on WGS 84 (an <code>sf</code> object must be in EPSG:4326), and the <code>site_id</code> values must be unique.
state	A string giving one state, the District of Columbia, or Puerto Rico, as a two-letter USPS abbreviation (such as "AL") or a two-digit FIPS code (such as "01"). Lowercase codes, such as "al", give an error unless the ACS data are supplied through <code>acs</code> . Only the tracts of this state are downloaded, so the part of a drive-time area in another state adds nothing to the estimates unless <code>acs</code> also holds the tracts of that state (see the Details of cacs_intersect_weight() , which also say what happens to <code>acs_year</code>). Data for Alaska, Hawaii, or Puerto Rico give an error at the intersection step.
year	A single whole number giving the last year of the ACS 5-year estimates, from 2009 to 2024; the default is 2023 (the 2019-2023 estimates).
drive_times	A numeric vector of drive times in minutes; the default is <code>c(5, 10, 15)</code> . When the areas are built, up to six whole numbers from 1 to 60 are accepted.
variables	A character vector of ACS variable codes, or <code>NULL</code> (the default) for cacs_acs_default_vars ; see cacs_acs_prefetch() for the accepted codes. Codes outside cacs_acs_default_vars are combined as counts, except those of tables B19013 and B25077 (medians) and B19301 (per capita income), so a median or per-person value from any other table is added up over the tracts like a count, without a warning (see cacs_intersect_weight()). Rates whose codes in cacs_acs_default_rates are left out are <code>NA</code> , with a warning.
provider	A string giving the routing service used to build the drive-time areas. "osrm" (the default) uses the Open Source Routing Machine (OSRM) through a public server that needs no key; it requires the <code>osrm</code> package. "ors" uses <code>openrouteservice</code> , which requires the <code>openrouteservice</code> package and an API key in the <code>ORS_API_KEY</code> environment variable or in <code>iso_args = list(ors_api_key = ...)</code> . "mapbox" and "r5r" are accepted names, but building areas with them is not implemented yet. Using them gives an error at the routing step, so the ACS data are downloaded first unless <code>acs</code> is supplied. When <code>precomputed_isochrones</code> is supplied, no areas are built, and <code>provider</code> is only recorded in the <code>cacs_run_provenance</code> attribute and shown when the result is printed.
weight_method	A string giving the weighting method: "area" (the default) or "population". With "area", counts are added up with each tract weighted by the share of its

	area inside the drive-time area, which assumes that whatever a variable counts is spread evenly over each tract's area. The rates are ratios of these counts. Medians and per-person values are averaged with weights proportional to each tract's area inside the drive-time area. These weights ignore how many people live in each tract, so the average can be far from the value for the drive-time area when the tracts differ in population density (see cacs_intersect_weight()). "population" is not implemented yet and gives an error before any step runs.
moe_formula	A string, or a list of strings named by ACS variable code, passed to cacs_propagate_moe() as its formula argument, or NULL (the default). Each ACS variable accepts only the formula for its kind ("weighted_sum" for a count, "weighted_mean" for a median or per-person value), and any other choice gives an error. The formulas for the rates are chosen by formula_dispatch.
output	A string choosing the form of the result, as described in the Value section: "long" (the default), "list_column", or "both".
precomputed_isochrones	An sf object of drive-time areas as returned by cacs_isochrone() (see cacs_validate_iso() for data from other sources), or NULL (the default) to build them with provider. When supplied, the routing step is skipped and every area in it is used. sites and drive_times do not select areas from it; they are recorded in the cacs_run_provenance attribute and shown when the result is printed. A site in sites that has no area in it gets no rows and no warning. If precomputed_isochrones has areas for sites that are not in sites, the numbers of sites shown when the result is printed are wrong (see print.cacs_run_result()). Areas with isochrone_empty = TRUE or a failure_reason are treated as routing failures (see Details).
cache_dir	A path to the cache folder, or NULL (the default) to use cacs_cache_dir(). It is used for the ACS data, drive-time areas, and intersection results that cacs_acs_prefetch(), cacs_isochrone(), and cacs_intersect_weight() save. A folder outside the temporary folder of the R session is tidied as described in cacs_cache_dir().
acs	An sf object of ACS tract estimates as returned by cacs_acs_prefetch() (see cacs_acs_validate() for data from other sources), or NULL (the default) to download them with tidycensus; downloading needs a Census API key (see cacs_acs_prefetch()). When supplied, the download step is skipped, every variable in it is used (variables is ignored), and state and year are only recorded in the cacs_run_provenance attribute. Census Bureau annotation codes, such as -555555555, are treated as missing, so a rate that uses such a value is NA for the areas that include the tract, and two rows for the same tract and variable give an error (see cacs_intersect_weight(), which also says when a warning is given). When a tract has a row for one code of a rate and not for the other, for example after rows with a missing estimate were removed, the rate is computed over different tracts without a warning (see the "Tract counts for rates" section of cacs_derive_rates()). Variable codes that cacs_intersect_weight() does not combine, such as "C17002_002" or "B19013A_001", make cacs_propagate_moe() stop with an error.
bg_pop_sf	An sf object of block-group population estimates, or NULL (the default). It has no effect on the result (see weight_method).
rates	A named list giving the numerator and denominator ACS codes of each rate. Only cacs_acs_default_rates (the default), the list of the five built-in rates,

is accepted; any other list, including a subset or a reordering of it, gives an error.

formula_dispatch	A string choosing the margin-of-error formula for the five rates. "general_ratio_conservative" (the default) uses the ratio formula for every rate. "auto" and "proportion_subset" use the proportion formula for poverty_rate and labor_force_participation and keep the ratio formula for snap_rate, ssi_rate, and unemp_rate; "proportion_subset" also gives a warning naming those three.
iso_args	A named list of arguments passed on to cacs_isochrone() when the areas are built, or an empty list (the default). The accepted names are osrm_mode, ors_api_key, mapbox_token, r5r_core, profile, osm_snapshot_date, and res; other names give a warning and are dropped. Because osrm.server is not accepted, another OSRM server is used by setting osrm_mode = "docker" and the option catchmentACS.osrm_docker_server (see the "OSRM servers" section of cacs_isochrone()).
acs_args	A named list of arguments passed on to cacs_acs_prefetch() when acs is NULL, or an empty list (the default). The accepted names are survey, geography, force_refresh, and write_gpkg; other names give a warning and are dropped.
weight_args	A named list of arguments passed on to cacs_intersect_weight() , or an empty list (the default). The accepted names are min_weight and keep_tract_audit; other names give a warning and are dropped.
moe_args	A named list of arguments passed on to cacs_propagate_moe() , or an empty list (the default). The accepted names are level, which sets the confidence level of all the margins of error (0.9 by default), and fallback_chain_max, which has no effect and gives a warning (see cacs_propagate_moe()). The name formula gives an error (moe_formula sets it), and other names give a warning and are dropped.
rate_args	A named list of arguments for cacs_derive_rates() , or an empty list (the default). No names are accepted: formula_dispatch gives an error (the argument formula_dispatch sets it), and any other name gives a warning and is dropped.
verbose	A logical value, passed to each step that cacs_run() runs. If TRUE (the default), messages report which steps run and how each step progresses; see the "Progress messages" section.

Details

Margins of error are at the 90 percent level unless moe_args sets another level, and they treat the tract estimates as independent (see [cacs_propagate_moe\(\)](#)). By default, every rate uses the formula that the Census Bureau's handbook gives for a ratio of two estimates. The five rates are proportions, for which the handbook gives a separate formula whose margins are never wider; formula_dispatch can select it for poverty_rate and labor_force_participation.

Site and drive-time pairs for which the routing service returns no area stay in the result with NA values and failure_origin = "isochrone" (failure_origin records the step at which a row failed), and a warning reports how many pairs failed. A routing failure stops the function in two cases: no pair succeeds, or the routing service refuses a request because its request limit has been reached or an API key is not accepted.

Value

With `output = "long"` (the default), a tibble that also has the class `cacs_run_result`, with one row for each site, drive time, and variable, where the variable is an ACS variable or one of the five rates. With `output = "list_column"`, a tibble of the same class with one row for each site and drive time. With `output = "both"`, a plain list with the elements `long` and `list_column`, one of each form.

The long form has these columns:

`site_id`, `drive_time_min`, `variable` The site, the drive time in minutes, and the ACS variable code or rate name.

`ring_topology` Always "cumulative": each drive-time area contains the shorter ones.

`estimate`, `moe` The estimate and its margin of error, at the confidence level given by the `cacs_confidence_level` attribute. For an ACS variable, both are NA when a tract's estimate is missing, and `moe` is NA when a tract's margin of error is missing.

`weight_sum` The sum of the coverage weights of the tracts combined (a tract's coverage weight is the share of its area inside the drive-time area); for a rate, the smaller of the sums for its numerator and denominator, and NA on a rate row whose `failure_origin` is "carrier".

`n_tracts` The number of tracts combined; NA on rate rows.

`n_tracts_num`, `n_tracts_den` On rate rows, the numbers of tracts combined for the numerator and for the denominator; NA on other rows and on a rate row whose `failure_origin` is "carrier" (see [cacs_derive_rates\(\)](#)).

`provider`, `profile`, `osm_snapshot_date` The routing service, the routing profile (such as "car"), and the date of the OpenStreetMap data, as recorded on the drive-time areas.

`acs_year` The ACS year from the `cacs_provenance` attribute that [cacs_acs_prefetch\(\)](#) puts on the ACS data; NA when the data have no such attribute, as with the sample ACS data in the package.

`weight_method`, `weight_basis` The weighting method ("area") and the weights used (see the `weight_method` argument): "coverage" for counts and rates, and "area_mean" for medians and per-person values.

`estimand_family` The kind of quantity: "spatial_total" (a count), "median_proxy" (a median), "area_weighted_scalar_proxy" (a per-person value such as per capita income), or "derived_rate" (a rate).

`moe_formula_requested`, `moe_formula_effective` The margin-of-error formula chosen, through `moe_formula` or `formula_dispatch` (see [cacs_derive_rates\(\)](#) for the rates), and the one used: "weighted_sum", "weighted_mean", "proportion_subset" (the proportion formula), or "general_ratio_conservative" (the ratio formula). A rate row whose `failure_origin` is "carrier" has no margin of error, and both columns then name the formula that was chosen for it rather than one that was used.

`moe_fallback`, `moe_fallback_reason` For a rate, whether the chosen formula could not be used, and why: "negative_variance" (the ratio formula was used instead) or "zero_denominator" (the rate and its margin of error are NA); "n/a" otherwise.

`failure_origin` The step at which the row failed: "none"; "isochrone" (the routing service returned no area; the row's estimates and most other values are NA); "intersection" (no tract is left for the area; the pair has one such row, with `variable = NA` and `n_tracts = 0`); or "carrier" (the rate is NA because its numerator or denominator, or the margin of error of either, is missing, as for the rates of a pair with no tract).

`weight_uncertainty_propagated` FALSE: the margins of error treat the weights as fixed.

The columns `est_total`, `var_total_raw`, `est_mean`, and `var_mean_raw` hold the weighted sums and averages of the tract estimates, with their variances, from which `estimate` and `moe` are computed (see `cacs_propagate_moe()`). They are NA on rate rows. With `options(cacs.return_se = TRUE)`, a column `se` (the standard error) follows; it is also NA on rate rows.

The list-column form has these columns:

`site_id` The site.

`lon`, `lat` The values of the columns of the same names in `sites`, or NA when `sites` has no such columns, even if it is an `sf` object such as `cacs_alabama_sites`; the point coordinates are not used.

`drive_time_min` The drive time in minutes.

`isochrone` The drive-time area, as a one-row `sf` object.

`acs_estimates`, `derived_rates` Tibbles of the rows of the long result for that site and drive time: the ACS variables and the rates.

`metadata` A list of `provider`, `profile`, `osm_snapshot_date`, `acs_year`, `weight_method`, `failure_origin`, and `n_tracts`, taken from the first row of `acs_estimates`.

Both forms have these attributes:

`cacs_schema_version` The version label ("1.0") of the column layout.

`cacs_run_provenance` A list recording how the result was produced. It gives the settings of the call: `state`, `year`, `drive_times`, `provider`, `weight_method`, `moe_formula`, `formula_dispatch`, `output_format`, `level` (the confidence level), and `min_weight`. It records which steps were skipped because their input was supplied, in `bypass_iso`, `bypass_acs`, and `execution_path`. It also holds counts of sites and of pairs that succeeded or failed, the time of the run, and the package and R versions.

`cacs_run_warnings` A list of the warnings given by each step, in the elements `acs_prefetch`, `isochrone`, `intersect_weight`, `propagate_moe`, and `derive_rates`; when pairs failed at the routing step, the element `orchestrator` has an entry whose `failed_pairs` lists them, with the reason.

`cacs_aggregation_provenance`, `cacs_moe_provenance`, `cacs_rate_provenance` The settings and counts recorded by `cacs_intersect_weight()`, `cacs_propagate_moe()`, and `cacs_derive_rates()`; the last includes the formula chosen for each rate.

`cacs_confidence_level` The confidence level of the margins of error (0.9 unless `moe_args` sets another level).

`cacs_run_result_metadata` A list that `print()` and `summary()` use: the time the result was created, the package version, the routing service and profile, and the drive times. It also holds counts of sites, the running time (`wall_clock_seconds`), `skipped_geoids`, and counts of rate rows by `moe_fallback_reason` (`moe_fallback_summary`).

The long form also keeps `skipped_geoids`, the tracts skipped because they have no area (see `cacs_intersect_weight()`), and `cacs_rate_audit`, the result of the check of each rate against a fixed range that `options(catchmentACS.audit_rates = TRUE)` turns on (see `cacs_derive_rates()`). With `weight_args = list(keep_tract_audit = TRUE)`, it also keeps `cacs_tract_audit`, a table of the tracts in each area with their coverage weights. `cacs_describe()` prints a summary drawn from these attributes.

Progress messages

The five steps report their progress with messages, whether they are run by `cacs_run()` or called on their own. Each step counts its work in units: sites for `cacs_isochrone()`, site and drive-time pairs for `cacs_intersect_weight()`, and a single unit for the other three steps. A step can show a summary line when it finishes, with how many units succeeded out of how many, the time taken, and any failures. It can also show a progress line as each unit finishes. An example is `Intersect+weight: 21/60 (ETA 0:01) - AL_SITE_07/15min`, where ETA is the estimated time left. `cacs_propagate_moe()` and `cacs_derive_rates()` show at most the summary line.

The first of these rules that applies decides what a step shows:

1. With `verbose = FALSE`, no progress messages.
2. If the environment variable `CACS_QUIET` is "1" (other values are ignored), no progress messages.
3. If `getOption("catchmentACS.progress")` is "off", no progress messages; if it is "force", the summary line and the progress lines, whatever the number of units.
4. Otherwise, as with "auto" or when the option is not set (the default), the summary line alone for fewer than five units, and the summary line and the progress lines for five or more.

With more than 50 units, only the first progress line, every tenth line after it, and the last line are shown; `options(catchmentACS.progress_throttle = k)` changes ten to `k`. A step whose result is read from the cache shows no progress messages. These settings do not affect warnings.

With `verbose = TRUE`, `cacs_run()` also shows messages of its own. A message at the start says which steps will run, and one message is shown for each step skipped because `acs` or `precomputed_isochrones` was supplied. At the end, a message gives the numbers of site and drive-time pairs that succeeded and failed. In the same way, `cacs_intersect_weight()` shows a message before it starts, and `cacs_acs_prefetch()` one before it downloads. Rules 2 and 3 leave these messages on; `verbose = FALSE` turns them off. With `output = "list_column"` or "both", a message about the isochrone column is shown on every call, whatever `verbose` is.

All of these are ordinary R messages, so `suppressMessages()` also hides them. Progress lines have the condition class `catchmentACS_message_progress_tick` and summary lines the class `catchmentACS_message_progress_summary`. Both also have the class `catchmentACS_message_progress`, as do the messages of `cacs_run()` and `cacs_intersect_weight()` that `verbose = FALSE` turns off, so `cacs_capture_conditions()` with that class in `classes` collects all of them. The condition classes of the package are listed in `catchmentACS-conditions`.

References

U.S. Census Bureau (2020). *Understanding and Using American Community Survey Data: What All Data Users Need to Know*. Chapter 8, Calculating Measures of Error for Derived Estimates.

See Also

The help pages `print.cacs_run_result()`, `summary.cacs_run_result()`, and `as_tibble.cacs_run_result()` describe how the result is printed, summarized, and converted to a plain tibble. `vignette("getting-started", package = "catchmentACS")` (<https://joonho112.github.io/catchmentACS/articles/getting-started.html>) walks through a first run, and `vignette("methodology", package = "catchmentACS")`

(<https://joonho112.github.io/catchmentACS/articles/methodology.html>) describes the calculations.

Other steps of the calculation: `cacs_acs_prefetch()`, `cacs_derive_rates()`, `cacs_intersect_weight()`, `cacs_isochrone()`, `cacs_propagate_moe()`

Examples

```
# Turn the cache off while this example runs (see ?cacs_set_cache).
old <- options(catchmentACS.cache_enabled = FALSE)

# Example data bundled with the package: the drive-time areas are circles
# with a radius of 1 km per minute, and the ACS data are made up. The
# routing columns of the areas, such as provider, hold fixed values that
# do not come from a routing service, and the result repeats some of them.
library(sf)
iso <- readRDS(system.file("extdata", "legacy_2025_isochrones.rds",
                           package = "catchmentACS"))
acs <- readRDS(system.file("extdata", "sample_alabama_subset.rds",
                           package = "catchmentACS"))
iso_07 <- iso[iso$site_id == "AL_SITE_07" & iso$drive_time_min == 10, ]
# The site at the center of these areas
site_07 <- data.frame(site_id = "AL_SITE_07", lon = -85.365, lat = 31.655)

# With both the drive-time areas and the ACS data supplied, no routing
# service, Census API key, or internet connection is needed.
out <- cacs_run(site_07, state = "AL", drive_times = 10,
                precomputed_isochrones = iso_07, acs = acs,
                verbose = FALSE)

# The five rates for the 10-minute area of AL_SITE_07
tibble::as_tibble(out) |>
  dplyr::filter(variable %in% names(cacs_acs_default_rates)) |>
  dplyr::select(variable, estimate, moe)

options(old)

# Needs a Census API key, and builds the areas of five sites on the public
# OSRM demo server, which limits the requests it accepts.
## Not run:
library(sf)
out_live <- cacs_run(cacs_alabama_sites[1:5, ], state = "AL", year = 2023,
                    drive_times = c(5, 10, 15), provider = "osrm")

# The poverty rate for each site and drive time, highest first
tibble::as_tibble(out_live) |>
  dplyr::filter(variable == "poverty_rate") |>
  dplyr::arrange(dplyr::desc(estimate)) |>
  dplyr::select(site_id, drive_time_min, estimate, moe)

## End(Not run)
```

cacs_set_cache	<i>Turn the cache on or off</i>
----------------	---------------------------------

Description

Turns on or off the saving and reuse of results on disk (the cache; see [cacs_cache_dir\(\)](#)) by [cacs_isochrone\(\)](#), [cacs_acs_prefetch\(\)](#), and [cacs_intersect_weight\(\)](#), and so by [cacs_run\(\)](#). Turning the cache off does not delete saved results.

Usage

```
cacs_set_cache(
  enabled = TRUE,
  scope = c("session", "global"),
  confirm = interactive()
)
```

Arguments

enabled	A logical value: TRUE (the default) turns the cache on and FALSE turns it off. NA or any other value gives an error.
scope	A string: "session" (the default) sets the option for the current R session, and "global" also shows the line to add to the R startup file for later sessions (see Details).
confirm	Not used. It is kept so that code written for earlier versions still runs.

Details

`cacs_set_cache()` sets the option `catchmentACS.cache_enabled` for the rest of the R session and changes no file. With `scope = "global"`, it also shows a line to add to the R startup file (see [Startup](#)) so that later sessions start with the same setting. For `enabled = FALSE`, the line sets this option. For `enabled = TRUE`, the line sets a cache folder that lasts between sessions instead, because the cache is on by default and its default folder is deleted when the session ends.

If an environment variable decides whether the cache is on (see the next section), the message says so.

Versions 0.3.0 to 0.5.1 wrote the setting to the startup file themselves, between the lines `# catchmentACS v0.3 cache control` and `# end catchmentACS v0.3 cache control`. R still runs these lines when a session starts, so the setting in them still applies. The package no longer reads or removes them; delete them by hand if they are not wanted.

Value

TRUE, invisibly.

Environment variables and options

These environment variables and options are read at every call, so a change takes effect at once. The rules are checked in this order, so CACS_NO_CACHE and CACS_CACHE_ENABLED take precedence over cacs_set_cache():

1. If the environment variable CACS_NO_CACHE is "1", "true", "yes", or "on" (ignoring case), the cache is off.
2. If the environment variable CACS_CACHE_ENABLED is set and not empty, the cache is on when its value is one of those four and off otherwise.
3. If the option catchmentACS.cache_enabled is TRUE or FALSE, it decides. Other values are ignored.
4. Otherwise, the cache is on (the default).

When the cache is on, one kind of saved result can be turned off on its own. The option catchmentACS.cache_isochrone, catchmentACS.cache_acs, or catchmentACS.cache_intersect set to FALSE turns off the cache of [cacs_isochrone\(\)](#), [cacs_acs_prefetch\(\)](#), or [cacs_intersect_weight\(\)](#). The environment variables CACS_CACHE_ISOCHRONE, CACS_CACHE_ACS, and CACS_CACHE_INTERSECT do the same when set to a value other than the four above, unless the matching option is TRUE.

See Also

Other cache and configuration: [cacs_cache_dir\(\)](#), [cacs_cache_status\(\)](#), [cacs_clear_cache\(\)](#), [cacs_get_cache_state\(\)](#)

Examples

```
# Turn the cache off for this session, check the setting, and restore it
old <- options("catchmentACS.cache_enabled")
cacs_set_cache(FALSE)
cacs_get_cache_state()$enabled

# Also show the line that keeps the cache off in later sessions;
# no file is changed
cacs_set_cache(FALSE, scope = "global")
options(old)
```

cacs_se_to_moe

Convert a standard error to a margin of error

Description

Multiplies a standard error (SE) by the normal quantile z for the level, giving the margin of error (MOE): $MOE = z \cdot SE$. A margin of error is the half-width of a confidence interval. When level is 0.90, the level of published American Community Survey (ACS) margins of error, the function uses $z = 1.645$, the value the Census Bureau uses. At other levels it uses $qnorm(1 - (1 - level) / 2)$, for example about 1.96 at level = 0.95.

Usage

```
cacs_se_to_moe(se, level = 0.9)
```

Arguments

<code>se</code>	A numeric vector of standard errors.
<code>level</code>	A single number between 0 and 1 (not a percentage) giving the confidence level of the margin of error. The default is 0.9. The value is not checked: <code>level = 90</code> , for example, gives NaN with a warning.

Value

A numeric vector of margins of error, the same length as `se`.

References

U.S. Census Bureau (2020). *Understanding and Using American Community Survey Data: What All Data Users Need to Know*. Chapter 7, Understanding Error and Determining Statistical Significance.

See Also

[cacs_propagate_moe\(\)](#) computes margins of error for drive-time area estimates.

Other rates and margins of error: [cacs_acs_default_rates](#), [cacs_moe_to_se\(\)](#)

Examples

```
cacs_se_to_moe(c(5, 10, NA), level = 0.90)
```

```
cacs_summary_as_markdown
```

Format a summary table as a Markdown table

Description

Formats a data frame as a Markdown pipe table for Quarto or R Markdown documents and GitHub pages, using [knitr::kable\(\)](#). Numbers are rounded to three decimals unless `digits` is given. The `knitr` package is required.

Usage

```
cacs_summary_as_markdown(summary_tbl, ...)
```

Arguments

`summary_tbl` A data frame, such as one of the elements `rates_breakdown`, `rates_per_site`, and `rates_per_site_moe` of a summary made by `summary.cacs_run_result()`.

... Arguments passed to `knitr::kable()`, such as `digits`, `caption`, `col.names`, or `align`. The format is always "pipe", so `format` gives an error.

Details

A table with a `site_id` column and a column for at least one of the five rates, such as `rates_per_site` or `rates_per_site_moe`, is changed in two ways. Its columns are put in the order `site_id`, `drive_time_min` (if present), the rates in the order of `cacs_acs_default_rates`, and any other columns. A caption is added unless `caption` is given. When the cells are text, as in `rates_per_site_moe`, the caption says that they show estimates with margins of error (half-widths of confidence intervals), at the confidence level in the table's attribute `cacs_confidence_level`, which `summary.cacs_run_result()` sets. For a table without that attribute, the caption gives no level.

Value

A character vector of class `knitr_kable` with the lines of the table, and of the caption if there is one. Printing it shows the table.

See Also

Other result summaries: `as_tibble.cacs_run_result()`, `cacs_describe()`, `group_by.cacs_run_result()`, `print.cacs_run_result()`, `print.cacs_run_summary()`, `summary.cacs_run_result()`

Examples

```
if (requireNamespace("knitr", quietly = TRUE)) {
  # A made-up table shaped like the rates_breakdown element of a summary
  summary_tbl <- tibble::tibble(
    variable = "poverty_rate",
    mean = 0.1234,
    sd = 0.0567,
    n_NA = 0L
  )
  cacs_summary_as_markdown(summary_tbl)
}
```

`cacs_validate_iso`

Check drive-time areas and list the problems found

Description

Checks whether an object can be used as drive-time areas (isochrones) by `cacs_intersect_weight()`, or by `cacs_run()` through its `precomputed_isochrones` argument, and returns a table with one row for each problem found. Those two functions stop with an error at the first of these problems. `cacs_run()` checks at the start only that `precomputed_isochrones` is an `sf` object, and makes the other checks after the download step.

Usage

```
cacs_validate_iso(iso_sf)
```

Arguments

iso_sf An sf object of drive-time areas with one row for each site and drive time, or any other object to check.

Details

The object must be an sf object; if it is not, that is the only problem reported. It must have the 16 columns of the result of [cacs_isochrone\(\)](#), with `ring_topology` equal to "cumulative" in every row, meaning that each area contains the areas of the shorter drive times. An `isomin` column, if present, must be 0 in every row (see [cacs_rings_to_cumulative\(\)](#)). The coordinate reference system must be EPSG:4326, and the geometry type POLYGON or MULTIPOLYGON. `provider` and `provider_requested` must be "osrm", "ors", "mapbox", or "r5r", and `site_id` must hold strings that are neither missing nor empty. `drive_time_min` must be above 0, `drive_time_min` and `retry_count` must be stored as integers, and `isochrone_empty` and `provider_downgrade` must be logical.

The areas themselves are not compared, so a 5-minute area larger than the 10-minute area of the same site passes. So does a site and drive time that appears in two rows, which [cacs_intersect_weight\(\)](#) turns into a row of NA values. The function gives an error only for an sf object without a usable geometry column, such as one made by selecting rows with `[]` before the sf package is loaded.

Areas made with another tool, such as the result of [cacs_rings_to_cumulative\(\)](#), can be used once the missing columns hold the values that [cacs_isochrone\(\)](#) gives an area built without a problem (see its Value section). NA in `isochrone_empty` and "" in `failure_reason` pass the checks here, but [cacs_run\(\)](#) treats an area as a routing failure unless its `isochrone_empty` is FALSE and its `failure_reason` is NA.

Value

A tibble with one row for each problem, and no rows when every check passes. Its seven columns are strings:

severity Always "error".

check A short name for the check, such as "iso_crs".

col The column concerned, or NA.

actual, expected What was found, and what is required.

fix_hint, example A suggested fix, and a line of R code for it, written for an object named `iso_sf`.

See Also

Other validation and conditions: [cacs_acs_validate\(\)](#), [cacs_capture_conditions\(\)](#), [cacs_validate_osrm_endpoint_catchmentACS-conditions](#)

Examples

```
# The drive-time areas bundled with the package: no problems, no rows
iso <- readRDS(system.file("extdata", "legacy_2025_isochrones.rds",
                           package = "catchmentACS"))

cacs_validate_iso(iso)

# Drive times stored as decimal numbers instead of integers
iso$drive_time_min <- as.numeric(iso$drive_time_min)
cacs_validate_iso(iso)
```

```
cacs_validate_osrm_endpoint
```

Check whether an OSRM server answers a routing request

Description

Sends one small request to an Open Source Routing Machine (OSRM) server and reports whether the server answered it with HTTP status 200. `cacs_isochrone()` sends many requests for each site, so this shows beforehand whether the server can be reached and is accepting requests.

Usage

```
cacs_validate_osrm_endpoint(server = NULL, timeout = 5)
```

Arguments

server	A string giving the address of the OSRM server, or NULL (the default). With NULL or "", the address is the value of the option <code>osrm.server</code> at the time of the call, or the public OSRM demo server <code>https://routing.openstreetmap.de/</code> when the option is not set. Loading the <code>osrm</code> package, for example with <code>library(osrm)</code> , sets that option to the demo server and replaces a value set before (see the "OSRM servers" section of <code>cacs_isochrone()</code>). The option <code>catchmentACS.osrm_docker_server</code> is not used, so the local server that <code>cacs_isochrone()</code> uses with <code>osrm_mode = "docker"</code> is checked only when its address is given here.
timeout	A single positive number giving the number of seconds to wait for an answer; the default is 5.

Details

When the server answers with status 429, meaning that its limit on requests has been reached, the function gives a warning of class `catchmentACS_warning_provider_quota_exhausted` and still returns its result. Other statuses give no warning. A server that cannot be reached, or does not answer within `timeout` seconds, gives `quota_ok = FALSE` and `http_status = NA` without a warning or an error.

The request is for a route between two fixed points near Birmingham, Alabama, so a server whose map data do not cover them may not answer with status 200. It is sent to `route/v1/driving/-86.8,33.5;-86.7,33.4` under the server address; on the public demo server, `routed-car/`, `routed-bike/`, or `routed-foot/` comes first, following the option `osrm.profile`.

Value

A tibble with one row and four columns:

`endpoint` The server address used.

`quota_ok` TRUE when the server answered with HTTP status 200, and FALSE otherwise, including when there was no answer.

`response_ms` The time taken, in milliseconds, including any time spent waiting for an answer that did not come.

`http_status` The HTTP status of the answer, as an integer, or NA when there was no answer.

See Also

Other validation and conditions: [cacs_acs_validate\(\)](#), [cacs_capture_conditions\(\)](#), [cacs_validate_iso\(\)](#), [catchmentACS-conditions](#)

Examples

```
# A local port where no server is expected to answer: quota_ok is FALSE
# and http_status is NA, without an error
cacs_validate_osrm_endpoint("http://localhost:9", timeout = 1)

# Contacts the public OSRM demo server, which limits the requests it
# accepts, and then builds a drive-time area there.
## Not run:
site_07 <- data.frame(site_id = "AL_SITE_07", lon = -85.365, lat = 31.655)

# Check the server before building the drive-time areas
check <- cacs_validate_osrm_endpoint()
check
if (check$quota_ok) {
  iso <- cacs_isochrone(site_07, drive_times = 10)
} else if (identical(check$http_status, 429L)) {
  # Request limit reached: build the areas with a local OSRM server
  iso <- cacs_isochrone(site_07, drive_times = 10, osrm_mode = "docker")
} else {
  stop("The OSRM server is not accepting requests; see check$http_status.")
}

## End(Not run)
```

catchmentACS-conditions

Condition classes for messages, warnings, and errors

Description

The messages, warnings, and errors of catchmentACS are conditions (R's term for all three) with classes that say what happened. Code can therefore select them by class instead of by the text of the message, with [cacs_capture_conditions\(\)](#) or with the base R functions described in the "Handling conditions by class" section.

Details

The class vector of a condition starts with a class of its own, such as `catchmentACS_warning_partial`. It is followed by the class of its kind (`catchmentACS_message`, `catchmentACS_warning`, or `catchmentACS_error`), then `catchmentACS_condition`, and last the classes that `rlang` and `R` add, such as `rlang_warning`, `warning`, and `condition`. Some conditions also have a class shared by a group of related conditions, between their own class and the class of their kind (see the lists below). A handler for a class receives every condition that has it, so a handler for `catchmentACS_warning` receives every warning of the package.

A few conditions do not have these classes. Some argument checks give a plain `R` error, such as a value of output in `cacs_run()` that is not one of the choices, and a missing `osrm` or `openrouteservice` package gives an error of class `rlib_error_package_not_found` from `rlang`. Messages and warnings from other packages keep their own classes.

Messages

`catchmentACS_message_progress` Messages that report on the steps as they run. Progress lines also have the class `catchmentACS_message_progress_tick`, and summary lines `catchmentACS_message_progress` (see the "Progress messages" section of `cacs_run()`).

`catchmentACS_message_cache` Messages about the cache, such as those saying that a saved result was read or saved, the message that `cacs_acs_prefetch()` shows before it downloads, the messages of `cacs_set_cache()`, and the messages of `cacs_clear_cache()` that report its result. When a saved result fails the check described in `cacs_cache_dir()` and is deleted, the message also has the class `catchmentACS_message_cache_legacy_invalidated` if its checksum file is missing, and `catchmentACS_message_cache_fingerprint_mismatch` otherwise. This message is given while the saved result is looked for, inside a call to `suppressMessages()`, so it is never shown, and neither a handler nor `cacs_capture_conditions()` receives it. `catchmentACS_message_cache_enabled_announce` is for a notice, off by default, that the cache is on. It is hidden in the same way when it is given during a lookup.

`catchmentACS_message_water_tract_filter` The message of `cacs_acs_prefetch()` that lists the water tracts it removed (see its "Water tracts" section).

`catchmentACS_message_demo_budget_protected` The message of `cacs_isochrone()` that res is 30 because it was not given (see its "OSRM grid resolution" section).

`catchmentACS_message_res_default_changed` The same message when res is 70.

`catchmentACS_message_perf_fix_applied` The message of `cacs_isochrone()`, with `verbose = TRUE`, that the Open Source Routing Machine (OSRM) server in use is not the public demo server, so there is no wait between requests.

`catchmentACS_message_listcol_iso_filled` The message of `cacs_run()` about the isochrone column, with output = "list_column" or "both".

`catchmentACS_message_rate_first_changed` The message, once per `R` session, that `as_tibble.cacs_run_result()` has put the rate rows first because of `options(catchmentACS.rate_first_default = TRUE)`. A call with `rate_first = TRUE` gives no such message.

`catchmentACS_message_resolve_site` The message of `cacs_plot_site_rates()` and `cacs_plot_site_pipeline()` that, given lat and lon instead of `site_id`, they use the nearest site.

Warnings

`catchmentACS_warning_runtime` A warning about a problem that does not stop the step, for example sites whose routing failed, geometry that was repaired, or a Census download that is tried again. The warnings of `cacs_propagate_moe()` and `cacs_derive_rates()` that count the rows whose margin of error used another formula or is NA also have this class. The "Rates that are NA" section of `cacs_derive_rates()` says when its warning also has the class `catchmentACS_warning_carrier_missing`.

`catchmentACS_warning_rate_out_of_range` The warnings of the range check on rates that `options(catchmentACS.audit = TRUE)` turns on (see `cacs_derive_rates()`).

`catchmentACS_warning_partial` The warning of `cacs_run()` when some, but not all, site and drive-time pairs have no drive-time area because routing failed or gave an empty area. The rows of those pairs are NA, and the `cacs_run_warnings` attribute of the result lists the pairs.

`catchmentACS_warning_geometry_skip` The warning of `cacs_intersect_weight()` that it skipped tracts whose area is zero or not finite (step 3 in its Details).

`catchmentACS_warning_provenance` A warning that an argument or a column was dropped, ignored, or renamed, such as an unknown name in `iso_args` or another list of arguments of `cacs_run()`. `cacs_intersect_weight()` also gives this class to its warning that the drive-time areas extend beyond the tracts in the American Community Survey (ACS) data (step 1 in its Details).

`catchmentACS_warning_variable` The warning of `cacs_acs_prefetch()` that it skipped variable codes that are not in the ACS variable list for the year.

`catchmentACS_warning_cache_stale_suspect` The warning that saved ACS data read from the cache have fewer rows than expected (see the "Cache behavior" section of `cacs_acs_prefetch()`). It also has the class `catchmentACS_warning_cache`.

`catchmentACS_warning_resolve_site_distant` The warning, given with `catchmentACS_message_resolve_site`, that the nearest site is more than 5 km from lat and lon.

`catchmentACS_warning_provider_quota_exhausted` The warning of `cacs_validate_osrm_endpoint()` that the OSRM server answered that its request limit has been reached (HTTP status 429).

Errors

`catchmentACS_error_schema` An argument or an input table is not in the expected form, such as a value of the wrong type, a missing column, or a value out of range. Most errors from argument checks have this class.

`catchmentACS_error_annulus_input` The drive-time areas are bands between two drive times, with `isomin` above 0 or `ring_topology = "annulus"`, which `cacs_intersect_weight()` and `cacs_run()` do not accept (`cacs_rings_to_cumulative()` converts them). These errors also have the class `catchmentACS_error_schema` and two older classes, `cacs_error_annulus_input` and `cacs_error_schema`, kept so that code written with them still works. No other condition has the older classes.

`catchmentACS_error_credential` A Census or openrouteservice API key is missing, or a routing service refused the requests as unauthorized (HTTP status 401 or 403). The same class is given for choices that are not implemented yet: `provider = "mapbox"` or `"r5r"`, and `weight_method = "population"`.

`catchmentACS_error_network` The Census download failed three times.

`catchmentACS_error_variable` None of the requested variable codes is in the ACS variable list for the year.

`catchmentACS_error_operator` The work cannot go on for a reason other than the form of the arguments. For example, the Census API or the routing service refused the requests, the cache folder cannot be created, the data lie outside the area that `cacs_intersect_weight()` accepts, or no site and drive-time pair has a drive-time area in `cacs_run()`.

`catchmentACS_error_geometry` The area weights cannot be computed: every tract has zero or non-finite area, the overlap of a tract and a drive-time area cannot be computed, or invalid geometry cannot be repaired.

`catchmentACS_error_missing_suggest` A package that the function needs, but that catchmentACS only suggests, is not installed: leaflet for the `cacs_plot_site_*`() map functions, or knitr for `cacs_summary_as_markdown()`.

Handling conditions by class

`cacs_capture_conditions()` records the messages and warnings of the package in a table instead of showing them. The base R functions for conditions take the same class names (see `conditions`). A handler for a class in `withCallingHandlers()` lets the code go on, and one in `tryCatch()` stops the code at the first condition of the class. `suppressMessages()` and `suppressWarnings()` can hide only the messages or warnings of the classes named in their `classes` argument, as in `suppressMessages(expr, classes = "catchmentACS_message_progress")`. `cacs_run()` also keeps the warnings of each step in the `cacs_run_warnings` attribute of its result.

See Also

Other validation and conditions: `cacs_acs_validate()`, `cacs_capture_conditions()`, `cacs_validate_iso()`, `cacs_validate_osrm_endpoint()`

Examples

```
# The error that cacs_acs_validate() gives when the ACS data are not an sf
# object, kept here to show its classes
err <- tryCatch(cacs_acs_validate(data.frame(GEOID = "01001020100")),
               error = function(e) e)
class(err)

# A handler for one class of the package
tryCatch(
  cacs_acs_validate(data.frame(GEOID = "01001020100")),
  catchmentACS_error_schema = function(e) "not in the expected form"
)
```

format.cacs_site_plot_pipeline

Describe the four maps in one line

Description

Returns a one-line description of the list that names its class and the `site_id` of its site, such as "<cacs_site_plot_pipeline: 4 widgets for site AL_BHM_01>", for use in text (for example, `sprintf("Maps: %s", format(x))`).

Usage

```
## S3 method for class 'cacs_site_plot_pipeline'
format(x, ...)
```

Arguments

`x` A list of class "cacs_site_plot_pipeline", as returned by `cacs_plot_site_pipeline()`.
`...` Not used.

Value

A string.

See Also

Other maps of one site: `cacs_plot_site_intersection()`, `cacs_plot_site_isochrone()`, `cacs_plot_site_pipeline()`, `cacs_plot_site_rates()`, `cacs_plot_site_weighted()`, `print.cacs_site_plot_pipeline()`

Examples

```
if (requireNamespace("leaflet", quietly = TRUE)) {
  fx <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                           package = "catchmentACS"))
  maps <- cacs_plot_site_pipeline(
    site_id = "AL_BHM_01", iso_sf = fx$iso_sf, tract_sf = fx$tract_sf,
    acs_sf = fx$acs_sf, run_result = fx$run_result, sites_df = fx$sites_df
  )
  sprintf("Maps: %s", format(maps))
}
```

```
group_by.cacs_run_result
```

Group the result of cacs_run()

Description

Groups a result of `cacs_run()` when it is passed to `dplyr::group_by()`. The method removes the class `cacs_run_result` and then groups the rows as those of an ordinary tibble, keeping their order. The grouped table keeps the other attributes of `.data`, and tables computed from it, for example with `dplyr::summarise()`, are ordinary tibbles.

Usage

```
## S3 method for class 'cacs_run_result'
group_by(.data, ..., .add = FALSE, .drop = NULL)
```

Arguments

.data	A cacs_run_result object, as returned by cacs_run() .
...	Variables or computations to group by, as in dplyr::group_by() .
.add, .drop	Passed to dplyr::group_by() . With .drop = NULL (the default), dplyr's default is used (dplyr::group_by_drop_default()).

Value

A grouped tibble, of class c("grouped_df", "tbl_df", "tbl", "data.frame").

See Also

Other result summaries: [as_tibble.cacs_run_result\(\)](#), [cacs_describe\(\)](#), [cacs_summary_as_markdown\(\)](#), [print.cacs_run_result\(\)](#), [print.cacs_run_summary\(\)](#), [summary.cacs_run_result\(\)](#)

Examples

```
out <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                           package = "catchmentACS"))$run_result

# The number of rows for each kind of quantity
out |>
  dplyr::group_by(estimand_family) |>
  dplyr::summarise(rows = dplyr::n())
```

```
print.cacs_run_result Print the result of cacs_run()
```

Description

Prints a result of [cacs_run\(\)](#) as a header about the run, two tables of the five rates, and a preview of the rows.

Usage

```
## S3 method for class 'cacs_run_result'
print(x, ..., n_head = 5, n_tail = 3)
```

Arguments

x	A cacs_run_result object, as returned by cacs_run() .
...	Passed to the print method for tibbles (tibble::print.tbl()) for the preview, such as n, the number of rows to show.
n_head, n_tail	Single numbers that have no effect (n in ... sets the number of rows).

Details

Each part of the output starts with a title line:

catchmentACS run result The time the result was created and the version of catchmentACS that created it, the routing service and the drive times given to `cacs_run()`, and the routing profile of the drive-time areas. The drive times are the ones given in the call, so when the areas come from `precomputed_isochrones` the line can name a drive time that no row of the result has. The line "Sites" gives the number of sites with at least one drive-time area (success), the number of sites given to `cacs_run()` in `sites (total)`, and `total` minus `success`, but not below 0 (failed). If `precomputed_isochrones` has areas for sites that are not in `sites`, `success` counts them too, so it can exceed `total` and `failed` can miss sites without a drive-time area. When tracts were skipped because they have no area, one more line gives the length of the `skipped_geoids` attribute, which lists each such tract once for each of its variables.

Top 5 rates (cross-site mean +/- sd) One row for each of the five rates, highest mean first. The columns are the unweighted mean (`mean_estimate`) and the standard deviation (`sd_estimate`) of the rate over all site and drive-time pairs, leaving out missing values, and `n`, the number of pairs, including those where the rate is missing.

Rates per site The estimates of the five rates, with one row for each site and drive time. The table is shown when the result has at most `getOption("catchmentACS.summary_per_site_max")` sites (12 by default). With 0 it is never shown, and with `Inf` always.

Tibble preview The rows, printed as a tibble. Before this title, the line "Wall clock" gives the time that `cacs_run()` took, in seconds. The last line names `as_tibble(x)`, which returns a plain tibble (see `as_tibble.cacs_run_result()`), and `x[]`, which is the same object and prints in the same way.

The rate tables need the columns `variable`, `estimate`, and `moe`, so they are not shown for the list-column form of the result. Subsets made with `[]` or with `dplyr` verbs such as `dplyr::filter()`, and tables made with `dplyr::count()` or `dplyr::distinct()`, keep the class and are printed in the same way, with the header of the whole result. Printing one that has those three columns but no `site_id` column gives an error. Without the `cacs_run_result_metadata` attribute, the result is printed as a plain tibble.

Value

`x`, invisibly.

Capturing the output

The tables are ordinary printed output, and every other line is an R message written with the `cli` package. These calls keep different parts:

```
capture.output(print(x))           # the tables
capture.output(print(x), type = "message") # the cli output
testthat::capture_messages(print(x)) # the cli output, in tests
suppressMessages(print(x))         # prints only the tables
```

See Also

Other result summaries: [as_tibble.cacs_run_result\(\)](#), [cacs_describe\(\)](#), [cacs_summary_as_markdown\(\)](#), [group_by.cacs_run_result\(\)](#), [print.cacs_run_summary\(\)](#), [summary.cacs_run_result\(\)](#)

Examples

```
# A bundled cacs_run() result for a 10-minute area in Birmingham, Alabama
out <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                           package = "catchmentACS"))$run_result

# One site and one drive time, so the standard deviations are NA
print(out, n = 5)
```

```
print.cacs_run_summary
```

Print the summary of a cacs_run() result

Description

Prints the list returned by [summary.cacs_run_result\(\)](#). It starts with a header giving the time the result was created, the routing service and profile, and the numbers of sites, as in [print.cacs_run_result\(\)](#). It then shows the numbers of rows, sites, variables, and drive times, the five-number summary `n_tracts_summary` from [stats::fivenum\(\)](#) (its hinges are labeled q1 and q3), the rate tables chosen by the `breakdown` argument of [summary\(\)](#), and `moe_fallback_rate` as a percentage. The table for each site shows the estimates only (the element `rates_per_site`).

Usage

```
## S3 method for class 'cacs_run_summary'
print(x, ...)
```

Arguments

<code>x</code>	A <code>cacs_run_summary</code> list, as returned by summary.cacs_run_result() .
<code>...</code>	Not used.

Details

The tables are ordinary printed output, and every other line is a message, as with [print.cacs_run_result\(\)](#).

Value

`x`, invisibly.

See Also

Other result summaries: [as_tibble.cacs_run_result\(\)](#), [cacs_describe\(\)](#), [cacs_summary_as_markdown\(\)](#), [group_by.cacs_run_result\(\)](#), [print.cacs_run_result\(\)](#), [summary.cacs_run_result\(\)](#)

Examples

```
out <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                           package = "catchmentACS"))$run_result

# With breakdown = "per_site", the only rate table is the one for each site
print(summary(out, breakdown = "per_site"))
```

```
print.cacs_site_plot_pipeline
```

Print a summary of the four maps

Description

Prints the site (its `site_id` and name), the variable of the third map, and the four elements of the list, `x$isochrone`, `x$intersection`, `x$weighted`, and `x$rates`, with a short note on each. No map is drawn; printing one of the elements draws that map.

Usage

```
## S3 method for class 'cacs_site_plot_pipeline'
print(x, ...)
```

Arguments

<code>x</code>	A list of class "cacs_site_plot_pipeline", as returned by cacs_plot_site_pipeline() .
<code>...</code>	Not used.

Details

The summary is written with the `cli` package as messages, so `capture.output(print(x), type = "message")` captures it and `capture.output(print(x))` does not.

Value

`x`, invisibly.

See Also

Other maps of one site: [cacs_plot_site_intersection\(\)](#), [cacs_plot_site_isochrone\(\)](#), [cacs_plot_site_pipeline\(\)](#), [cacs_plot_site_rates\(\)](#), [cacs_plot_site_weighted\(\)](#), [format.cacs_site_plot_pipeline\(\)](#)

Examples

```
if (requireNamespace("leaflet", quietly = TRUE)) {
  fx <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                           package = "catchmentACS"))
  maps <- cacs_plot_site_pipeline(
    site_id = "AL_BHM_01", iso_sf = fx$iso_sf, tract_sf = fx$tract_sf,
    acs_sf = fx$acs_sf, run_result = fx$run_result, sites_df = fx$sites_df
  )
  # Lists the four maps and the variable of the third, without drawing a map
  print(maps)
}
```

```
summary.cacs_run_result
```

Summarize the result of cacs_run()

Description

Summarizes a result of `cacs_run()` in a list of counts and tables of the five rates, which `print.cacs_run_summary()` prints.

Usage

```
## S3 method for class 'cacs_run_result'
summary(object, ..., breakdown = c("cross_site", "per_site", "both"))
```

Arguments

object	A result of <code>cacs_run()</code> in the long form (output = "long", the default). The list-column form gives an error.
...	Not used.
breakdown	A string choosing the rate tables that <code>print.cacs_run_summary()</code> shows. With "cross_site" (the default), it shows rates_breakdown, and also rates_per_site when the result has at most <code>getOption("catchmentACS.summary_per_site_max")</code> sites, as in <code>print.cacs_run_result()</code> . "per_site" shows only rates_per_site, and "both" shows both. The elements of the list do not depend on this choice.

Value

A list of class `c("cacs_run_summary", "list")` with the elements below, and an attribute `breakdown` that keeps the breakdown argument for printing.

metadata The `cacs_run_result_metadata` attribute of object, or NULL if it has none: the values shown in the header of `print.cacs_run_result()`, the time that `cacs_run()` took in seconds (`wall_clock_seconds`), `skipped_geoids`, and counts of rate rows by `moe_fallback_reason` (`moe_fallback_summary`).

- `n_rows`, `n_sites`, `n_variables`, `n_drive_times` The number of rows, and the numbers of distinct values of `site_id`, `variable`, and `drive_time_min`.
- `n_tracts_summary` The five-number summary of `n_tracts` from `stats::fivenum()`: minimum, lower hinge, median, upper hinge, and maximum. Rows where `n_tracts` is NA, such as rate rows, are left out.
- `rates_breakdown` A tibble with one row for each of the five rates (`variable`). The columns are the unweighted mean (`mean`) and the standard deviation (`sd`) of the rate over all site and drive-time pairs, leaving out missing values, and `n_NA`, the number of pairs where the rate is missing. `print.cacs_run_result()` shows the same means and standard deviations as "Top 5 rates", under the names `mean_estimate` and `sd_estimate`. The last column of that table, `n`, counts all the pairs, including those where the rate is missing.
- `rates_per_site` A tibble of the estimates of the five rates, with one row for each site and drive time (`site_id`, `drive_time_min`).
- `rates_per_site_moe` The same table with each cell a string giving the estimate and its margin of error (the half-width of its confidence interval), rounded to three decimals. When object has the attribute `cacs_confidence_level`, the table has it as well, and `cacs_summary_as_markdown()` names that level in its caption.
- `moe_fallback_rate` The number of rows whose `moe_fallback` is TRUE (see `cacs_run()`), divided by the number of all rows.

See Also

Other result summaries: `as_tibble.cacs_run_result()`, `cacs_describe()`, `cacs_summary_as_markdown()`, `group_by.cacs_run_result()`, `print.cacs_run_result()`, `print.cacs_run_summary()`

Examples

```
# A bundled cacs_run() result for a 10-minute area in Birmingham, Alabama
out <- readRDS(system.file("extdata", "visual_walkthrough_fixture.rds",
                           package = "catchmentACS"))$run_result

s <- summary(out)
s

# The rates with their margins of error
s$rates_per_site_moe
```

Index

- * **cache and configuration**
 - cacs_cache_dir, 12
 - cacs_cache_status, 14
 - cacs_clear_cache, 17
 - cacs_get_cache_state, 25
 - cacs_set_cache, 61
- * **datasets**
 - cacs_acs_default_vars, 5
 - cacs_alabama_sites, 11
- * **maps of one site**
 - cacs_plot_site_intersection, 37
 - cacs_plot_site_isochrone, 39
 - cacs_plot_site_pipeline, 40
 - cacs_plot_site_rates, 44
 - cacs_plot_site_weighted, 46
 - format.cacs_site_plot_pipeline, 70
 - print.cacs_site_plot_pipeline, 75
- * **rates and margins of error**
 - cacs_acs_default_rates, 4
 - cacs_moe_to_se, 36
 - cacs_se_to_moe, 62
- * **result summaries**
 - as_tibble.cacs_run_result, 3
 - cacs_describe, 23
 - cacs_summary_as_markdown, 63
 - group_by.cacs_run_result, 71
 - print.cacs_run_result, 72
 - print.cacs_run_summary, 74
 - summary.cacs_run_result, 76
- * **steps of the calculation**
 - cacs_acs_prefetch, 6
 - cacs_derive_rates, 19
 - cacs_intersect_weight, 26
 - cacs_isochrone, 31
 - cacs_propagate_moe, 48
 - cacs_run, 53
- * **validation and conditions**
 - cacs_acs_validate, 9
 - cacs_capture_conditions, 15
 - cacs_validate_iso, 64
 - cacs_validate_osrm_endpoint, 66
 - catchmentACS-conditions, 67
- as_tibble.cacs_run_result, 3
- as_tibble.cacs_run_result(), 16, 25, 59, 64, 68, 72–74, 77
- assignOps, 16
- cacs_acs_default_rates, 4, 5, 19, 36, 44, 50, 53–55, 63, 64
- cacs_acs_default_vars, 5, 7, 21, 54
- cacs_acs_prefetch, 6
- cacs_acs_prefetch(), 5, 10, 12, 13, 16, 18, 21–24, 26–28, 30, 35–37, 41, 42, 46, 47, 51, 53–57, 59–62, 68, 69
- cacs_acs_validate, 9
- cacs_acs_validate(), 8, 9, 16, 22, 27, 30, 55, 65, 67, 70
- cacs_alabama_sites, 11, 38, 39, 42, 45, 47, 54, 58
- cacs_cache_dir, 12
- cacs_cache_dir(), 7, 9, 14, 15, 18, 25–27, 29, 32, 33, 55, 61, 62, 68
- cacs_cache_status, 14
- cacs_cache_status(), 14, 18, 25, 26, 62
- cacs_capture_conditions, 15
- cacs_capture_conditions(), 10, 59, 65, 67, 68, 70
- cacs_clear_cache, 17
- cacs_clear_cache(), 7–9, 14–16, 25, 26, 33, 62, 68
- cacs_derive_rates, 19
- cacs_derive_rates(), 4, 5, 9, 16, 23, 24, 27–30, 35, 42, 44, 45, 50, 51, 53, 55–60, 69
- cacs_describe, 23
- cacs_describe(), 4, 30, 58, 64, 72, 74, 77
- cacs_get_cache_state, 25

- cacs_get_cache_state(), 9, 14–16, 18, 24, 62
- cacs_intersect_weight, 26
- cacs_intersect_weight(), 7–10, 12, 16, 18–24, 26, 34, 35, 38, 42, 47–56, 58–62, 64, 65, 69, 70
- cacs_isochrone, 31
- cacs_isochrone(), 9, 12, 16, 18, 22–24, 26, 27, 29, 30, 37, 39, 41, 42, 44, 46, 51–53, 55, 56, 59–62, 65, 66, 68
- cacs_moe_to_se, 36
- cacs_moe_to_se(), 5, 51, 63
- cacs_plot_site_intersection, 37
- cacs_plot_site_intersection(), 40, 41, 43, 45, 47, 48, 71, 75
- cacs_plot_site_isochrone, 39
- cacs_plot_site_isochrone(), 11, 38, 41, 43, 45, 48, 71, 75
- cacs_plot_site_pipeline, 40
- cacs_plot_site_pipeline(), 38–40, 45, 48, 68, 71, 75
- cacs_plot_site_rates, 44
- cacs_plot_site_rates(), 38, 40, 41, 43, 48, 68, 71, 75
- cacs_plot_site_weighted, 46
- cacs_plot_site_weighted(), 38, 40, 41, 43, 45, 71, 75
- cacs_propagate_moe, 48
- cacs_propagate_moe(), 9, 16, 19, 20, 22–24, 27, 29, 30, 35, 36, 53, 55, 56, 58–60, 63, 69
- cacs_rings_to_cumulative, 51
- cacs_rings_to_cumulative(), 27, 65, 69
- cacs_run, 53
- cacs_run(), 3–5, 7, 9–11, 15, 16, 20–24, 27, 29, 30, 32, 35, 42, 44, 47, 49, 51, 52, 61, 64, 65, 68–73, 76, 77
- cacs_se_to_moe, 62
- cacs_se_to_moe(), 5, 36, 51
- cacs_set_cache, 61
- cacs_set_cache(), 9, 14–16, 18, 25, 26, 29, 33, 68
- cacs_summary_as_markdown, 63
- cacs_summary_as_markdown(), 4, 25, 70, 72, 74, 77
- cacs_validate_iso, 64
- cacs_validate_iso(), 10, 16, 27, 35, 52, 55, 67, 70
- cacs_validate_osrm_endpoint, 66
- cacs_validate_osrm_endpoint(), 10, 16, 33, 35, 65, 69, 70
- catchmentACS-conditions, 67
- catchmentACS_condition
(catchmentACS-conditions), 67
- catchmentACS_error
(catchmentACS-conditions), 67
- catchmentACS_message
(catchmentACS-conditions), 67
- catchmentACS_warning
(catchmentACS-conditions), 67
- conditions, 70
- dplyr::bind_rows(), 28
- dplyr::count(), 73
- dplyr::distinct(), 73
- dplyr::filter(), 73
- dplyr::group_by(), 71, 72
- dplyr::group_by_drop_default(), 72
- dplyr::summarise(), 71
- format.cacs_site_plot_pipeline, 70
- format.cacs_site_plot_pipeline(), 38, 40, 43, 45, 48, 75
- group_by.cacs_run_result, 71
- group_by.cacs_run_result(), 4, 25, 64, 74, 77
- knitr::kable(), 63, 64
- print.cacs_description(cacs_describe), 23
- print.cacs_run_result, 72
- print.cacs_run_result(), 4, 24, 25, 55, 59, 64, 72, 74, 76, 77
- print.cacs_run_summary, 74
- print.cacs_run_summary(), 4, 25, 64, 72, 74, 76, 77
- print.cacs_site_plot_pipeline, 75
- print.cacs_site_plot_pipeline(), 38, 40, 43, 45, 48, 71
- sf::st_write(), 7
- Startup, 13, 61
- stats::fivenum(), 74, 77
- summary.cacs_run_result, 76
- summary.cacs_run_result(), 4, 25, 59, 64, 72, 74

`tempdir()`, [13](#)
`tibble::as_tibble()`, [3](#)
`tibble::print.tbl()`, [72](#)
`tools::R_user_dir()`, [13](#)