Package 'lakefetch'

October 9, 2026

Title Calculate Fetch and Wave Exposure for Lake Sampling Points

Version 0.1.14

Description Calculates fetch (open water distance) and wave exposure metrics for lake sampling points. Downloads lake boundaries from 'OpenStreetMap', calculates directional fetch using a ray-casting approach, and optionally integrates National Hydrography Dataset ('NHD') data <https://www.usgs.gov/national-hydrography> for hydrological context including outlet and inlet locations. Can estimate lake depth from surface area using empirical relationships, and integrate historical weather data for cumulative wave energy calculations. Includes an optional interactive 'shiny' application for visualization.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

LazyData true

Depends R (>= 4.1.0)

RoxygenNote 7.3.3

Imports sf (>= 1.0-0), osmdata (>= 0.2.0), ggplot2 (>= 3.0.0)

Suggests hydrogeofetch, jsonlite, shiny, leaflet, base64enc, parallel, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://docs.ropensci.org/lakefetch/, https://github.com/ropensci/lakefetch

BugReports https://github.com/ropensci/lakefetch/issues

NeedsCompilation no

Author Jeremy Lynch Farrell [aut, cre]

Maintainer Jeremy Lynch Farrell <farrej2@rpi.edu>

Repository CRAN

Date/Publication 2026-10-09 02:50:02 UTC

Contents

lakefetch-package . . . 2
add_lake_context . . . 3
add_lake_depth . . . 4
add_weather_context . . . 5
adirondack_sites . . . 6
assign_sites_to_lakes . . . 7
create_ray_geometries . . . 8
example_lake . . . 9
fetch_app . . . 10
fetch_app_upload . . . 11
fetch_calculate . . . 12
get_lake_boundary . . . 14
get_lake_depth . . . 16
lakefetch_options . . . 17
lakefetch_reset_options . . . 19
load_sites . . . 19
plot_fetch_bars . . . 20
plot_fetch_map . . . 21
plot_fetch_rose . . . 22
sanitize_filename . . . 23
wisconsin_lakes . . . 23

Index **25**

`lakefetch-package` *lakefetch: Calculate Fetch and Wave Exposure for Lake Sampling Points*

Description

The lakefetch package provides tools for calculating fetch (open water distance) and wave exposure metrics for lake sampling points. It downloads lake boundaries from OpenStreetMap, calculates directional fetch using ray-casting, and optionally integrates with NHD for hydrological context.

Main Functions

`fetch_calculate` Main entry point for fetch calculation

`load_sites` Load and validate site data

`get_lake_boundary` Get lake boundary from OSM or file

`add_lake_context` Add NHD hydrological context

`fetch_app` Launch interactive Shiny app

Visualization

`plot_fetch_map` Map of sites colored by exposure

`plot_fetch_bars` Bar chart of effective fetch

`plot_fetch_rose` Rose diagram for single site

`create_ray_geometries` Create ray lines for mapping

Configuration

`lakefetch_options` Get/set package options

`lakefetch_reset_options` Reset options to defaults

Author(s)

Maintainer: Jeremy Lynch Farrell <farrej2@rpi.edu>

See Also

Useful links:

- https://docs.ropensci.org/lakefetch/
- https://github.com/ropensci/lakefetch
- Report bugs at https://github.com/ropensci/lakefetch/issues

---

`add_lake_context` *Add Lake Context from NHD*

Description

Add hydrological context to fetch results using the National Hydrography Dataset (NHD). Includes outlet/inlet locations, watershed area, connectivity classification, and stream order.

Usage

```
add_lake_context(fetch_results, lake_polygons, utm_epsg)
```

Arguments

`fetch_results`	sf object with fetch calculation results
`lake_polygons`	sf object with lake polygons
`utm_epsg`	EPSG code for UTM projection

Details

Requires the hydrogeofetch package. If not available, returns the input with NA columns added for consistent output format.

Added columns include:

- nhd_permanent_id: NHD permanent identifier
- nhd_gnis_name: GNIS name from NHD
- nhd_areasqkm: Area in square kilometers from NHD
- outlet_dist_m: Distance to outlet in meters
- outlet_bearing: Compass direction to outlet
- inlet_nearest_dist_m: Distance to nearest inlet
- inlet_nearest_bearing: Compass direction to nearest inlet
- inlet_count: Number of inlets
- connectivity_class: Headwater/Drainage/Terminal/Isolated
- outlet_stream_order: Strahler stream order at outlet
- watershed_area_ha: Watershed area in hectares
- lake_watershed_ratio: Lake area / watershed area

Value

sf object with additional columns for NHD context

Examples

```
csv_path <- system.file("extdata", "sample_sites.csv", package = "lakefetch")
sites <- load_sites(csv_path)
lake <- get_lake_boundary(sites)
results <- fetch_calculate(sites, lake)
results_with_context <- add_lake_context(results$results, results$lakes, lake$utm_epsg)
```

---

add_lake_depth *Add Depth Information to Fetch Results*

Description

Looks up or estimates depth for each lake in the fetch results and adds depth columns.

Usage

```
add_lake_depth(fetch_results, lakes, user_depths = NULL)
```

Arguments

fetch_results	sf object with fetch results
lakes	sf object with lake polygons
user_depths	Named vector of user-provided depths (names = lake IDs)

Value

fetch_results with added depth columns

Examples

```
data(adirondack_sites)
sites <- load_sites(adirondack_sites)
lake <- get_lake_boundary(sites)
results <- fetch_calculate(sites, lake)

# Add depth estimates
results$results <- add_lake_depth(results$results, results$lakes)

# Or provide known depths using an actual lake_osm_id from results
lake_id <- results$lakes$osm_id[1]
depths <- setNames(15.5, lake_id)
results$results <- add_lake_depth(results$results, results$lakes, user_depths = depths)
```

---

add_weather_context *Add Weather Context to Fetch Results*

Description

Adds historical weather metrics and cumulative wave energy to fetch calculation results. For each site, the function queries the Open-Meteo historical-weather API for wind speed and direction in the days leading up to the sample's `datetime`, combines those winds with the site's directional fetch to estimate wave height (Sverdrup-Munk-Bretschneider equations), and integrates wave energy across the requested look-back window(s).

Usage

```
add_weather_context(
  fetch_results,
  datetime_col = "datetime",
  windows_hours = c(24, 72, 168),
  depth_m = NULL
)
```

Arguments

fetch_results	sf object with fetch results (must have datetime column)
datetime_col	Name of the datetime column
windows_hours	Numeric vector of look-back windows in hours over which to integrate cumulative wave energy. The default c(24, 72, 168) produces metrics for the 1-day, 3-day, and 7-day windows ending at each site's sampling datetime. For each window the function adds columns named wave_energy_24h, wave_energy_72h, wave_energy_168h, etc.
depth_m	Water depth for orbital velocity calculation

Details

The input data must have a datetime column in POSIXct format or a format that can be parsed (ISO 8601, or common date-time formats). Network access to the Open-Meteo API is required; sites are queried sequentially with a short pause between calls to respect the public API rate limit.

Value

sf object with additional weather columns

Examples

```
csv_path <- system.file("extdata", "sample_sites.csv", package = "lakefetch")
sites <- load_sites(csv_path)
lake <- get_lake_boundary(sites)
results <- fetch_calculate(sites, lake)

# Add datetime to results
results$results$datetime <- as.POSIXct("2024-07-15 14:00:00")

# Add weather context
results_with_weather <- add_weather_context(
  results$results,
  datetime_col = "datetime"
)
```

---

adirondack_sites *Adirondack Lake Sampling Sites*

Description

A dataset containing example lake sampling sites from the Adirondack region of New York State. These synthetic but realistic coordinates demonstrate typical multi-lake sampling scenarios.

Usage

```
adirondack_sites
```

Format

A data frame with 12 rows and 5 variables:

Site Unique site identifier

lake.name Name of the lake

latitude Latitude in decimal degrees (WGS84)

longitude Longitude in decimal degrees (WGS84)

datetime Date and time of sampling (POSIXct)

Details

The dataset includes sites from four Adirondack lakes:

- Blue Mountain Lake (3 sites)
- Raquette Lake (4 sites)
- Long Lake (2 sites)
- Tupper Lake (3 sites)

Source

Synthetic data for demonstration purposes

Examples

```
# Load the dataset
data(adirondack_sites)

# View structure
str(adirondack_sites)


# Use with lakefetch (requires internet connection)
sites <- load_sites(adirondack_sites)
lake_data <- get_lake_boundary(sites)
results <- fetch_calculate(sites, lake_data)
```

---

assign_sites_to_lakes *Assign Sites to Lakes*

Description

Perform spatial join to assign each site to its containing lake polygon.

Usage

```
assign_sites_to_lakes(sites_sf, water_polygons, tolerance_m = NULL)
```

Arguments

`sites_sf`	sf object with site points
`water_polygons`	sf object with lake polygons
`tolerance_m`	Buffer distance in meters for matching sites that fall just outside lake polygons (e.g., due to GPS noise or coarse OSM boundaries). Default is the value from `lakefetch_options()` (50 m). For datasets with appreciable GPS error or for very coarsely mapped shorelines, try 100-500 m.

Value

sf object with sites and added columns for lake_osm_id, lake_name, lake_area_km2

Examples

```
csv_path <- system.file("extdata", "sample_sites.csv", package = "lakefetch")
sites <- load_sites(csv_path)
lake_data <- get_lake_boundary(sites)

# Assign sites to their containing lakes. Default tolerance (50 m) is
# appropriate for sites with accurate coordinates that fall inside the
# lake polygon. If your sites are near the shoreline or your GPS error is
# larger, increase tolerance_m (e.g., 200-500 m).
sites_assigned <- assign_sites_to_lakes(
  lake_data$sites,
  lake_data$all_lakes,
  tolerance_m = 200
)

# Check assignments
table(sites_assigned$lake_name)
```

`create_ray_geometries` *Create Ray Geometries for Map Visualization*

Description

Create line geometries representing fetch rays from each site. Useful for detailed visualization of the ray-casting results.

Usage

```
create_ray_geometries(fetch_data)
```

Arguments

`fetch_data`	Results from `fetch_calculate`

Value

An sf object with ray line geometries

Examples

```
csv_path <- system.file("extdata", "sample_sites.csv", package = "lakefetch")
sites <- load_sites(csv_path)
lake <- get_lake_boundary(sites)
results <- fetch_calculate(sites, lake)
rays <- create_ray_geometries(results)

# Plot rays for a specific site
site_name <- results$results$Site[1]
site_rays <- rays[rays$Site == site_name, ]
ggplot2::ggplot() + ggplot2::geom_sf(data = site_rays, ggplot2::aes(color = Distance))
```

---

example_lake *Blue Mountain Lake Polygon (Example Lake)*

Description

The OpenStreetMap boundary polygon for Blue Mountain Lake in Hamilton County, New York. Bundled with the package so that fetch examples can run offline (no internet or Overpass API call required) and pkgdown pages can render plot output. The coordinates match the sites in `system.file("extdata", "sample_sites.csv", package = "lakefetch")`, so the two datasets can be used together end-to-end.

Usage

```
example_lake
```

Format

An sf object with 1 row and 3 fields plus geometry:

osm_id OSM relation identifier

name Lake name ("Blue Mountain Lake")

area_km2 Surface area in square kilometers

geometry MULTIPOLYGON geometry in UTM Zone 18N (EPSG:32618)

Source

Downloaded from OpenStreetMap https://www.openstreetmap.org/. See `data-raw/create_example_data.R` for the exact query.

Examples

```
data(example_lake)
print(example_lake)

# Plot the lake
library(ggplot2)
ggplot(example_lake) + geom_sf()

# Load matching sample sites (they lie inside this polygon) and
# compute fetch end-to-end without touching OSM. First convert
# example_lake into the multi-lake list format that fetch_calculate()
# expects:
sites <- load_sites(system.file("extdata", "sample_sites.csv",
                                package = "lakefetch"))
sites_sf <- sf::st_transform(
  sf::st_as_sf(sites, coords = c("longitude", "latitude"), crs = 4326,
               remove = FALSE),
  sf::st_crs(example_lake)
)
lake_data <- list(all_lakes = example_lake,
                  sites = sites_sf,
                  utm_epsg = sf::st_crs(example_lake)$epsg)
results <- fetch_calculate(sites, lake_data, add_context = FALSE)
sf::st_drop_geometry(results$results)[, c("Site", "fetch_effective",
                                          "exposure_category")]
```

`fetch_app` *Launch Interactive Fetch App*

Description

Launch a Shiny app for interactive exploration of fetch calculation results. Click on site markers to view fetch rays and detailed information. Click anywhere on the map to analyze a new point.

Usage

```
fetch_app(fetch_data, title = NULL)
```

Arguments

`fetch_data`	Results from `fetch_calculate`
`title`	Optional app title

Details

Requires the shiny, leaflet, and base64enc packages (suggested dependencies).

The app displays:

- Interactive map with satellite imagery
- Site markers colored by exposure category
- Click markers to see fetch rays
- Popup with rose diagram and metrics
- Click anywhere on the map to analyze a new point

Value

Launches a Shiny app (does not return)

Examples

```
if (interactive()) {
  sites <- load_sites("my_sites.csv")
  lake <- get_lake_boundary(sites)
  results <- fetch_calculate(sites, lake)
  fetch_app(results)
}
```

---

fetch_app_upload *Launch Interactive Fetch App with File Upload*

Description

Launch a standalone Shiny app where users can upload a CSV file with GPS coordinates, and the app will automatically download lake boundaries, calculate fetch, and display interactive results.

Usage

```
fetch_app_upload(title = "Lake Fetch Calculator")
```

Arguments

title	Optional app title (default: "Lake Fetch Calculator")

Details

Requires the shiny, leaflet, and base64enc packages (suggested dependencies).

The app workflow:

1. Upload a CSV file with latitude/longitude columns
2. App downloads lake boundaries from OpenStreetMap
3. Calculates fetch for all uploaded points
4. Displays interactive map with results
5. Click anywhere on a lake to analyze additional points

6. Download results as CSV or GeoPackage

CSV file requirements:

- Must have columns starting with "lat" and "lon" (case-insensitive)
- Optional "Site" column for point names
- Additional columns are preserved in output

Value

Launches a Shiny app (does not return)

Examples

```
if (interactive()) {
  # Launch the upload app
  fetch_app_upload()
}
```

`fetch_calculate` *Calculate Fetch for Lake Sampling Sites*

Description

Main entry point for fetch calculation. Takes sites and lake boundaries, calculates directional fetch using ray-casting, and returns results with exposure metrics.

Usage

```
fetch_calculate(
  sites,
  lake,
  depth_m = NULL,
  fetch_method = NULL,
  add_context = TRUE,
  find_max_fetch = FALSE
)
```

Arguments

`sites`	Data frame or sf object with site locations
`lake`	Lake boundary data from `get_lake_boundary`
`depth_m`	Mean water depth in meters for orbital velocity calculation (used in the SMB wave hindcast equations). Can be a single value (applied to all sites), a vector (one per site), or NULL to use depth from sites data or the default from `lakefetch_options`. Mean depth is preferred over maximum depth as it better represents conditions across the water column for wave attenuation estimates.

`fetch_method` Method for calculating effective fetch. Options:

"top3" Mean of the 3 highest directional fetch values (default)

"max" Maximum directional fetch value

"cosine" SPM/CERC cosine-weighted average. Uses 9 radials centered on the direction of maximum fetch at 6-degree intervals, weighted by cosine of angle from center. Based on Shore Protection Manual (1984).

If NULL, uses the value from `lakefetch_options`.

`add_context` Logical; add NHD context if available (default TRUE)

`find_max_fetch` Logical; if TRUE, finds the location in each lake with the maximum possible fetch using a longest-internal-chord algorithm. The result is returned as a `$max_fetch` element in the output list. Default FALSE.

Details

For each site, the function:

1. Assigns the site to its containing lake polygon
2. Buffers the site inward from shore (GPS accuracy adjustment)
3. Casts rays in all directions at specified angle resolution
4. Measures distance to shore in each direction
5. Calculates summary metrics (mean, max, effective fetch)
6. Calculates orbital velocity using depth
7. Derives exposure category (Sheltered/Moderate/Exposed)

Exposure thresholds can be configured via `lakefetch_options`.

Value

A list with elements:

`results` sf object with fetch results for each site

`lakes` sf object with lake polygons used

`angles` Vector of angles used for fetch calculation

`max_fetch` (only if `find_max_fetch = TRUE`) sf object with one row per lake containing the maximum fetch location, chord length (meters), and chord bearing (degrees)

References

Shore Protection Manual (1984). U.S. Army Corps of Engineers, Coastal Engineering Research Center. 4th Edition.

Examples

```
csv_path <- system.file("extdata", "sample_sites.csv", package = "lakefetch")
sites <- load_sites(csv_path)
lake <- get_lake_boundary(sites)
results <- fetch_calculate(sites, lake)

# With explicit depth
results <- fetch_calculate(sites, lake, depth_m = 5)

# Using cosine-weighted effective fetch (SPM method)
results <- fetch_calculate(sites, lake, fetch_method = "cosine")

# Access results
results$results  # sf with all fetch data
results$lakes    # lake polygons

# Find the location with maximum fetch in each lake
results <- fetch_calculate(sites, lake, find_max_fetch = TRUE)
results$max_fetch  # sf with max fetch location per lake
```

---

get_lake_boundary *Get Lake Boundary*

Description

Get lake boundary polygon(s) either from OpenStreetMap or from a local file.

Usage

```
get_lake_boundary(
  sites,
  file = NULL,
  timeout = 90,
  simplify_tolerance_m = 0,
  total_timeout_s = 300
)
```

Arguments

sites	A data.frame with latitude and longitude columns, or an sf object.
file	Optional file path to a shapefile or geopackage with lake boundaries.
timeout	Integer; Overpass API query timeout in seconds. Default is 90. Increase for very large lakes (e.g., `timeout = 300` for Mälaren or Great Lakes) or when server load is high.

simplify_tolerance_m
: Numeric; if greater than 0, simplify lake polygons with `sf::st_simplify(dTolerance = simplify_tolerance_m)` (in meters, applied in the UTM projection). Useful for very large or complex lakes where an exact coastline is not needed and a coarser polygon greatly speeds up fetch ray-casting. Typical values: 50-500 m for large lakes (e.g., Mälaren, Vättern). Default is 0 (no simplification).

total_timeout_s
: Numeric; soft wall-clock budget in seconds on the total time `get_lake_boundary()` will spend downloading from OSM. The budget is consulted at natural breakpoints (between Overpass query types, between clusters for spread-out sites, and between name-filtered queries) and aborts further work when exceeded. It is NOT a hard cap on a single `osmdata::osmdata_sf()` call: if Overpass returns HTTP 429, osmdata does its own 60-second-per-retry backoff loop internally and we cannot safely interrupt that from R without risking a segfault on Windows. So on a heavily throttled server a single call may still exceed `total_timeout_s` by several minutes. For a hard ceiling, wrap the call in `R.utils::withTimeout()` yourself, or supply a local boundary file via the `file` argument. Default 300 seconds (5 minutes); set to `Inf` to disable.

Details

If `file` is provided, the lake boundary is loaded from the file. Otherwise, the function downloads lake boundaries from OpenStreetMap based on the bounding box of the provided sites.

For very large lakes (> ~500 km^2), the default 90-second Overpass API timeout may be exceeded. Use `timeout = 300` or higher in those cases. For lakes with very complex shorelines (e.g., Mälaren, Vättern, Võrtsjärv), additionally pass `simplify_tolerance_m = 100` (or higher) to coarsen the polygon and speed up downstream fetch calculations.

Value

A list with elements:

`all_lakes`	sf object with lake polygons in UTM projection
`sites`	sf object with site points in UTM projection
`utm_epsg`	EPSG code for the UTM projection used

Examples

```
csv_path <- system.file("extdata", "sample_sites.csv", package = "lakefetch")
sites <- load_sites(csv_path)
lake_data <- get_lake_boundary(sites)

# For very large lakes, increase the timeout
lake_data <- get_lake_boundary(sites, timeout = 300)

# For large/complex lakes, also coarsen the shoreline
lake_data <- get_lake_boundary(sites, timeout = 300,
                               simplify_tolerance_m = 100)
```

`get_lake_depth` *Get Lake Depth Estimates*

Description

Retrieves or estimates lake depth for wave calculations. Uses user-provided depth if available, otherwise estimates from lake surface area using empirical relationships.

Usage

```
get_lake_depth(
  lake_polygon,
  site_coords = NULL,
  user_depth = NULL,
  method = "auto"
)
```

Arguments

`lake_polygon`	sf polygon of the lake
`site_coords`	Coordinates of the sample site (optional, for future bathymetry grid support)
`user_depth`	User-provided depth in meters (highest priority)
`method`	Method for depth estimation: "auto" or "empirical"

Details

Depth estimation methods:

1. User-provided: Direct input, highest confidence
2. Empirical: Estimated from lake surface area using published relationships

The empirical method uses the relationship from Cael et al. (2017): mean_depth ~ 10.3 * area_km2^0.25

Value

A list with elements:

`depth_mean`	Estimated mean depth in meters
`depth_max`	Estimated maximum depth in meters (if available)
`source`	Source of the estimate ("user" or "empirical")
`confidence`	Confidence level ("high", "medium", "low")

References

Messager, M.L., Lehner, B., Grill, G., Nedeva, I., Schmitt, O. (2016): Estimating the volume and age of water stored in global lakes using a geo-statistical approach. Nature Communications, 7: 13603.

Cael, B.B., Heathcote, A.J., Seekell, D.A. (2017): The volume and mean depth of Earth's lakes. Geophysical Research Letters, 44: 209-218.

Examples

```
data(example_lake)

# With user-provided depth
depth <- get_lake_depth(example_lake, user_depth = 8.5)

# Estimate from lake area
depth <- get_lake_depth(example_lake)
```

---

lakefetch_options *Get or Set lakefetch Package Options*

Description

Get or set options that control the behavior of lakefetch functions.

Usage

```
lakefetch_options(...)
```

Arguments

...	Named arguments to set options. If empty, returns all current options.

Details

Available options:

buffer_distance_m GPS accuracy buffer in meters (default: 10)

angle_resolution_deg Direction resolution in degrees (default: 5)

max_fetch_m Maximum fetch distance in meters (default: 50000)

validation_buffer_m Shore detection validation buffer (default: 10)

default_wind_speed_ms Default wind speed in m/s (default: 10)

default_depth_m Default water depth in meters (default: 10)

gps_tolerance_m Buffer for matching sites to lakes (default: 100)

fetch_method Effective fetch calculation method: "top3" (mean of 3 highest directional fetches, default), "max" (maximum directional fetch), or "cosine" (SPM/CERC cosine-weighted average across 9 radials at 6-degree intervals; see Shore Protection Manual, 1984)

exposure_sheltered_m Fetch threshold below which sites are classified as "Sheltered" (default: 2500 m). This is a practical default; no universal standard exists in the literature. Adjust based on your study system.

exposure_exposed_m Fetch threshold above which sites are classified as "Exposed" (default: 5000 m). Sites between thresholds are "Moderate". See Mason et al. (2018) for Great Lakes exposure mapping methodology.

exposure_relative_sheltered Proportion of lake maximum fetch below which sites are classified as "Sheltered" in the relative exposure system (default: 0.25). Sites are classified relative to the lake's longest internal chord (maximum possible fetch).

exposure_relative_exposed Proportion of lake maximum fetch above which sites are classified as "Exposed" in the relative exposure system (default: 0.50). Sites between thresholds are "Moderate".

use_parallel Use parallel processing for multi-lake (default: TRUE)

use_nhd Use NHD integration if available (default: TRUE)

Value

If no arguments, returns a list of all current options. If arguments provided, sets those options and returns invisible NULL.

References

Shore Protection Manual (1984). U.S. Army Corps of Engineers, Coastal Engineering Research Center. 4th Edition.

Mason, L. A., Riseng, C. M., Laber, A. L., & Rutherford, E. S. (2018). Effective fetch and relative exposure index maps for the Laurentian Great Lakes. Scientific Data, 5, 180295.

Examples

```
# Get all options
lakefetch_options()

# Get specific option
lakefetch_options()$buffer_distance_m

# Set options
lakefetch_options(buffer_distance_m = 20, angle_resolution_deg = 10)
```

lakefetch_reset_options *Reset lakefetch Options to Defaults*

Description

Reset all lakefetch package options to their default values.

Usage

```
lakefetch_reset_options()
```

Value

Invisible NULL

Examples

```
lakefetch_reset_options()
```

load_sites *Load Sites from CSV or Data Frame*

Description

Load and validate site data for fetch calculation. Automatically detects coordinate columns (latitude/longitude) and cleans the data.

Usage

```
load_sites(x, lat_col = NULL, lon_col = NULL, site_col = NULL, lake_col = NULL)
```

Arguments

x	Either a file path to a CSV file or a data.frame with site data.
lat_col	Optional character string specifying the name of the latitude column. If NULL (default), auto-detects columns starting with "lat".
lon_col	Optional character string specifying the name of the longitude column. If NULL (default), auto-detects columns starting with "lon".
site_col	Optional character string specifying the name of the site identifier column. If NULL (default), auto-detects a column named "site".
lake_col	Optional character string specifying the name of the lake name column. If NULL (default), auto-detects common lake name patterns.

Details

The function:

- Detects latitude/longitude columns (names starting with "lat"/"lon")
- Cleans coordinate values (removes non-numeric characters)
- Creates Site column if not present
- Removes rows with invalid or missing coordinates
- Detects location name from data columns or filename

Column names can be specified explicitly using the `lat_col`, `lon_col`, `site_col`, and `lake_col` arguments. This is useful when your data uses non-standard column names that the auto-detection cannot find.

Value

A data.frame with columns Site, latitude, longitude, and any additional columns from the input. Includes attributes "location_name" and "location_column" if a location was detected.

Examples

```
# Load from data frame
df <- data.frame(
  Site = c("A", "B", "C"),
  latitude = c(43.42, 43.43, 43.41),
  longitude = c(-73.69, -73.68, -73.70)
)
sites <- load_sites(df)

# Load with custom column names
df2 <- data.frame(
  sample_id = c("A", "B"),
  y_coord = c(43.42, 43.43),
  x_coord = c(-73.69, -73.68),
  reservoir = c("Lake One", "Lake One")
)
sites <- load_sites(df2, lat_col = "y_coord", lon_col = "x_coord",
                    site_col = "sample_id", lake_col = "reservoir")
```

`plot_fetch_bars` *Plot Fetch Bar Chart*

Description

Create a bar chart showing effective fetch by site.

Usage

```
plot_fetch_bars(fetch_data, title = "Effective Fetch by Site")
```

Arguments

fetch_data	Results from fetch_calculate
title	Optional plot title

Value

A ggplot2 object

Examples

```
csv_path <- system.file("extdata", "sample_sites.csv", package = "lakefetch")
sites <- load_sites(csv_path)
lake <- get_lake_boundary(sites)
results <- fetch_calculate(sites, lake)
plot_fetch_bars(results)
```

---

plot_fetch_map *Plot Fetch Map*

Description

Create a map showing site locations colored by exposure category.

Usage

```
plot_fetch_map(fetch_data, title = "Fetch Analysis - Site Locations")
```

Arguments

fetch_data	Results from fetch_calculate
title	Optional plot title

Value

A ggplot2 object

Examples

```
# Use the bundled example lake (Blue Mountain Lake, NY) and the
# matching sample sites to compute and plot fetch offline.
data(example_lake)
sites_df <- load_sites(system.file("extdata", "sample_sites.csv",
                                   package = "lakefetch"))
sites_sf <- sf::st_transform(
  sf::st_as_sf(sites_df,
               coords = c("longitude", "latitude"), crs = 4326,
               remove = FALSE),
```

```
  sf::st_crs(example_lake)
)
lake_data <- list(all_lakes = example_lake,
                  sites = sites_sf,
                  utm_epsg = sf::st_crs(example_lake)$epsg)
results <- fetch_calculate(sites_df, lake_data, add_context = FALSE)
plot_fetch_map(results)
```

`plot_fetch_rose` *Plot Fetch Rose Diagram*

Description

Create a rose diagram showing directional fetch for a single site.

Usage

```
plot_fetch_rose(fetch_data, site, title = NULL)
```

Arguments

`fetch_data`	Results from `fetch_calculate`
`site`	Site name (character) or row index (integer) to plot
`title`	Optional plot title (defaults to site name)

Value

Invisible NULL (creates base R plot)

Examples

```
# Compute fetch offline against the bundled Blue Mountain Lake polygon.
data(example_lake)
sites_df <- load_sites(system.file("extdata", "sample_sites.csv",
                                   package = "lakefetch"))
sites_sf <- sf::st_transform(
  sf::st_as_sf(sites_df,
               coords = c("longitude", "latitude"), crs = 4326,
               remove = FALSE),
  sf::st_crs(example_lake)
)
lake_data <- list(all_lakes = example_lake,
                  sites = sites_sf,
                  utm_epsg = sf::st_crs(example_lake)$epsg)
results <- fetch_calculate(sites_df, lake_data, add_context = FALSE)
plot_fetch_rose(results, 1)
```

sanitize_filename *Sanitize a String for Use in Filenames*

Description

Remove or replace invalid filename characters.

Usage

```
sanitize_filename(name)
```

Arguments

name	Character string to sanitize

Value

A sanitized string safe for use as a filename

Examples

```
sanitize_filename("Lake O'Brien (2024)")
```

wisconsin_lakes *Wisconsin Lake Sampling Sites*

Description

A dataset containing example sampling sites from well-known Wisconsin lakes. These coordinates are useful for testing with real lake boundaries from OpenStreetMap.

Usage

```
wisconsin_lakes
```

Format

A data frame with 8 rows and 4 variables:

Site Unique site identifier

lake.name Name of the lake

latitude Latitude in decimal degrees (WGS84)

longitude Longitude in decimal degrees (WGS84)

Details

The dataset includes sites from three Wisconsin lakes:

- Lake Mendota (3 sites) - Madison's largest lake, well-studied
- Lake Monona (2 sites) - Connected to Mendota via Yahara River
- Geneva Lake (3 sites) - Popular recreational lake in SE Wisconsin

Source

Synthetic data based on real lake locations

Examples

```
# Load the dataset
data(wisconsin_lakes)

# View the data
head(wisconsin_lakes)


# Use with lakefetch (requires internet connection)
sites <- load_sites(wisconsin_lakes)
lake_data <- get_lake_boundary(sites)
results <- fetch_calculate(sites, lake_data)
```

Index

∗ **datasets**
adirondack_sites, 6
example_lake, 9
wisconsin_lakes, 23

add_lake_context, *2*, 3
add_lake_depth, 4
add_weather_context, 5
adirondack_sites, 6
assign_sites_to_lakes, 7

create_ray_geometries, *3*, 8

example_lake, 9

fetch_app, *2*, 10
fetch_app_upload, 11
fetch_calculate, *2*, *8*, *10*, 12, *21, 22*

get_lake_boundary, *2*, *12*, 14
get_lake_depth, 16

lakefetch *(*lakefetch-package*)*, 2
lakefetch-package, 2
lakefetch_options, *3*, *12, 13*, 17
lakefetch_reset_options, *3*, 19
load_sites, *2*, 19

plot_fetch_bars, *3*, 20
plot_fetch_map, *3*, 21
plot_fetch_rose, *3*, 22

sanitize_filename, 23

wisconsin_lakes, 23