Package ‘osrm.backend’

December 3, 2025

Title Bindings for 'Open Source Routing Machine'

Version 0.1.0

Description Install and control 'Open Source Routing Machine' ('OSRM') backend executables to prepare routing data and run/stop a local 'OSRM' server. For computations with the running server use the 'osrm' R package (<https://cran.r-project.org/package=osrm>).

License MIT + file LICENSE

URL https://github.com/e-kotov/osrm.backend, https://www.ekotov.pro/osrm.backend/

BugReports https://github.com/e-kotov/osrm.backend/issues

Imports digest, httr2, jsonlite, processx, ps

Suggests knitr, osrm, quarto, sf, testthat (>= 3.0.0)

VignetteBuilder quarto

Config/testthat/edition 3

Encoding UTF-8

Language en

RoxygenNote 7.3.3

SystemRequirements OSRM backend binaries (>= v5.27.0) <https://github.com/Project-OSRM/osrm-backend>; package downloads binaries automatically if not found

NeedsCompilation no

Author Egor Kotov [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-6690-5345>)

Maintainer Egor Kotov <kotov.egor@gmail.com>

Repository CRAN

Date/Publication 2025-12-03 21:00:02 UTC

Contents

`osrm_check_available_versions`
Check for Available OSRM Versions

Description

Queries the GitHub API to get a list of all available version tags for the OSRM backend that have binaries for the current platform.

Usage

```
osrm_check_available_versions(prereleases = FALSE)
```

Arguments

`prereleases`	A logical value. If `TRUE`, include pre-release versions in the returned list. Defaults to `FALSE`.

Value

A character vector of available version tags.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  # Get all stable versions with binaries for this platform
  osrm_check_available_versions()
}
```

`osrm_check_latest_version` *Check for the Latest Stable OSRM Version*

Description

Queries the GitHub API to find the most recent stable (non-pre-release) version tag for the OSRM backend that has binaries available for the current platform.

Usage

```
osrm_check_latest_version()
```

Value

A string containing the latest version tag (e.g., `"v5.27.1"`).

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  # Get the latest stable version number of OSRM backend
  osrm_check_latest_version()
}
```

`osrm_cleanup` *Clean Up OSRM Files in a Directory*

Description

Remove OSRM-generated files from a directory. This is useful when switching between algorithms (CH and MLD) or when you want to start fresh.

Usage

```
osrm_cleanup(path, keep_osm = TRUE, dry_run = FALSE, quiet = FALSE)
```

Arguments

`path`	A string. Path to an OSRM file or directory containing OSRM files. If a file path is provided (e.g., `data.osm.pbf` or `data.osrm.hsgr`), the base name will be extracted and all related `.osrm.*` files will be removed.
`keep_osm`	Logical. If `TRUE` (default), keeps the original `.osm.pbf` (or `.osm`, `.osm.bz2`) file. If `FALSE`, removes it as well.
`dry_run`	Logical. If `TRUE`, shows what would be deleted without actually deleting. Default is `FALSE`.
`quiet`	Logical. If `TRUE`, suppresses messages. Default is `FALSE`.

Details

OSRM creates many `.osrm.*` files during the extract, contract, partition, and customize stages. This function helps clean up these files.

Important: The CH and MLD algorithms cannot safely coexist in the same directory because the MLD partition stage modifies some extract-stage files. Use this function to clean up before switching algorithms.

Value

Invisibly returns a character vector of removed file paths.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # Stage a temporary workspace with placeholder OSRM files
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  file.create(
    file.path(osrm_dir, "cur.osrm.timestamp"),
    file.path(osrm_dir, "cur.osrm.hsgr"),
    file.path(osrm_dir, "cur.osrm.mldgr"),
    file.path(osrm_dir, "cur.osrm.partition")
  )

  # Preview what would be deleted
  osrm_cleanup(osrm_dir, dry_run = TRUE)

  # Clean up OSRM artifacts (keep the OSM file)
  osrm_cleanup(osrm_dir)
  # Remove everything including the OSM source
  osrm_cleanup(osrm_dir, keep_osm = FALSE)

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

osrm_clear_path *Clear OSRM Path from Project's .Rprofile*

Description

Scans the `.Rprofile` file in the current project's root directory and removes any lines that were added by `osrm_install()` to modify the `PATH`.

Usage

```
osrm_clear_path(quiet = FALSE)
```

Arguments

`quiet`	A logical value. If `TRUE`, suppresses messages. Defaults to `FALSE`.

Value

`TRUE` if the file was modified, `FALSE` otherwise.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  # Clean up a temporary project's .Rprofile
  old <- setwd(tempdir())
  on.exit(setwd(old), add = TRUE)
  writeLines(
    c(
      "#added-by-r-pkg-osrm.backend",
      'Sys.setenv(PATH = paste("dummy", Sys.getenv("PATH"), sep = .Platform$path.sep))'
    ),
    ".Rprofile"
  )
  osrm_clear_path(quiet = TRUE)
  unlink(".Rprofile")
}
```

osrm_contract *Contract OSRM Graph for Contraction Hierarchies (CH)*

Description

Run the `osrm-contract` tool to contract an OSRM graph for the CH pipeline. After running, a companion `<base>.osrm.hsgr` file must exist to confirm success.

Usage

```
osrm_contract(
  input_osrm,
  threads = 8L,
  verbosity = c("INFO", "NONE", "ERROR", "WARNING", "DEBUG"),
  segment_speed_file = NULL,
  turn_penalty_file = NULL,
  edge_weight_updates_over_factor = 0,
  parse_conditionals_from_now = 0,
  time_zone_file = NULL,
  quiet = FALSE,
  verbose = FALSE,
  spinner = TRUE,
  echo_cmd = FALSE
)
```

Arguments

`input_osrm`	A string. Path to a `.osrm.timestamp` file, the base path to the `.osrm` files (without extension), or a directory containing exactly one `.osrm.timestamp` file.
`threads`	An integer. Number of threads to use; default `8`.
`verbosity`	A string. Log verbosity level passed to `-l/--verbosity` (one of `"NONE"`,`"ERROR"`,`"WARNING"`,`"INFO"`,`'` default `"INFO"`.
`segment_speed_file`	A string or `NULL`. Path to nodeA,nodeB,speed CSV; default `NULL`.
`turn_penalty_file`	A string or `NULL`. Path to from_,to_,via_nodes,penalties CSV; default `NULL`.
`edge_weight_updates_over_factor`	A numeric. Threshold for logging large weight updates; default `0`.
`parse_conditionals_from_now`	A numeric. UTC timestamp for conditional restrictions; default `0`.
`time_zone_file`	A string or `NULL`. GeoJSON file for time zone boundaries; default `NULL`.
`quiet`	A logical. Master switch that suppresses package messages and process output when `TRUE`; default `FALSE`.
`verbose`	A logical. When `TRUE` and `quiet = FALSE`, streams stdout and stderr from the underlying `processx::run` calls.

`spinner`	A logical. When `TRUE` and `quiet = FALSE`, shows a spinner instead of live logs; default `TRUE`.
`echo_cmd`	A logical. When `TRUE` and `quiet = FALSE`, prints each command before running; default `FALSE`.

Value

An object of class `osrm_job` with the following elements:

osrm_job_artifact The path to the contracted `.osrm.hsgr` file.

osrm_working_dir The directory containing all OSRM files.

logs The `processx::run` result object.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # Prepare a small graph then contract it for the CH pipeline
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  profile <- osrm_find_profile("car.lua")

  extract_job <- osrm_extract(
    input_osm = tmp_pbf,
    profile = profile,
    overwrite = TRUE,
    threads = 1L
  )

  ch_graph <- osrm_contract(extract_job, threads = 1L, verbose = TRUE)
  ch_graph$osrm_job_artifact

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

`osrm_customize` *Customize OSRM Graph for Multi-Level Dijkstra (MLD)*

Description

Run the `osrm-customize` tool to customize a partitioned OSRM graph for the MLD pipeline. After running, a companion `<base>.osrm.mldgr` file must exist to confirm success.

Usage

```
osrm_customize(
  input_osrm,
  threads = 8L,
  verbosity = c("INFO", "NONE", "ERROR", "WARNING", "DEBUG"),
  segment_speed_file = NULL,
  turn_penalty_file = NULL,
  edge_weight_updates_over_factor = 0,
  parse_conditionals_from_now = 0,
  time_zone_file = NULL,
  quiet = FALSE,
  verbose = FALSE,
  spinner = TRUE,
  echo_cmd = FALSE
)
```

Arguments

`input_osrm`	A string. Path to a `.osrm.partition` file, the base path to the partitioned `.osrm` files (without extension), or a directory containing exactly one `.osrm.partition` file.
`threads`	An integer. Number of threads to use; default `8` (osrm-customize's default).
`verbosity`	A string. Log verbosity level passed to `-l/--verbosity` (one of `"NONE"`,`"ERROR"`,`"WARNING"`,`"INFO"`,`'` default `"INFO"`.
`segment_speed_file`	A string or `NULL`. Path to nodeA,nodeB,speed CSV; default `NULL`.
`turn_penalty_file`	A string or `NULL`. Path to from_,to_,via_nodes,penalties CSV; default `NULL`.
`edge_weight_updates_over_factor`	A numeric. Factor threshold for logging large weight updates; default `0`.
`parse_conditionals_from_now`	A numeric. UTC timestamp from which to evaluate conditional turn restrictions; default `0`.
`time_zone_file`	A string or `NULL`. GeoJSON file with time zone boundaries; default `NULL`.
`quiet`	A logical. Master switch that suppresses package messages and process output when `TRUE`; default `FALSE`.

`verbose`	A logical. When `TRUE` and `quiet = FALSE`, streams stdout and stderr from the underlying `processx::run` calls.
`spinner`	A logical. When `TRUE` and `quiet = FALSE`, shows a spinner instead of live logs; default `TRUE`.
`echo_cmd`	A logical. When `TRUE` and `quiet = FALSE`, prints each command before running; default `FALSE`.

Value

An object of class `osrm_job` with the following elements:

osrm_job_artifact The path to the customized `.osrm.mldgr` file.

osrm_working_dir The directory containing all OSRM files.

logs The `processx::run` result object.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # Partition then customize a graph for the MLD pipeline
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  profile <- osrm_find_profile("car.lua")

  extract_job <- osrm_extract(
    input_osm = tmp_pbf,
    profile = profile,
    overwrite = TRUE,
    threads = 1L
  )
  partition_job <- osrm_partition(extract_job, threads = 1L)

  mld_graph <- osrm_customize(partition_job, threads = 1L, verbose = TRUE)
  mld_graph$osrm_job_artifact

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

`osrm_extract` *Extract OSM into OSRM Graph Files*

Description

Run the `osrm-extract` tool to preprocess an OSM file (`.osm`, `.osm.bz2`, or `.osm.pbf`) into the base `.osrm` graph files using a specified Lua profile. After running, a companion <base>`.osrm.timestamp` file must exist to confirm success.

Usage

```
osrm_extract(
  input_osm,
  profile = osrm_find_profile("car.lua"),
  threads = 8L,
  overwrite = FALSE,
  verbosity = c("INFO", "NONE", "ERROR", "WARNING", "DEBUG"),
  data_version = NULL,
  small_component_size = 1000L,
  with_osm_metadata = FALSE,
  parse_conditional_restrictions = FALSE,
  location_dependent_data = NULL,
  disable_location_cache = FALSE,
  dump_nbg_graph = FALSE,
  quiet = FALSE,
  verbose = FALSE,
  spinner = TRUE,
  echo_cmd = FALSE
)
```

Arguments

`input_osm`	A string. Path to the input OSM file (`.osm`, `.osm.bz2`, or `.osm.pbf`) or a directory containing exactly one OSM file with a supported extension.
`profile`	A string. Path to the OSRM Lua profile (e.g. returned by `osrm_find_profile("car.lua")`).
`threads`	An integer. Number of threads for `-t`/`--threads`; default `8` (OSRM's default).
`overwrite`	A logical. If `FALSE` (default), stops when any existing `.osrm*` files matching the base name are found alongside `input_osm`. Set to `TRUE` to proceed regardless.
`verbosity`	A string. Log verbosity level passed to `-l`/`--verbosity` (one of `"NONE"`, `"ERROR"`, `"WARNING"`, `"INFO"`, ' default `"INFO"`.
`data_version`	A string or `NULL`. Passed to `-d`/`--data_version`; default `NULL`, in which case the option is omitted.

`small_component_size`
: An integer. For `--small-component-size`; default `1000` (OSRM's default).

`with_osm_metadata`
: A logical. If `TRUE`, adds `--with-osm-metadata`; default `FALSE`.

`parse_conditional_restrictions`
: A logical. If `TRUE`, adds `--parse-conditional-restrictions`; default `FALSE`.

`location_dependent_data`
: A string or `NULL`. Path to GeoJSON, passed to `--location-dependent-data`; default `NULL`, in which case the option is omitted.

`disable_location_cache`
: A logical. If `TRUE`, adds `--disable-location-cache`; default `FALSE`.

`dump_nbg_graph` A logical. If `TRUE`, adds `--dump-nbg-graph`; default `FALSE`.

`quiet` A logical. Master switch that suppresses package messages and process output when `TRUE`; default `FALSE`.

`verbose` A logical. When `TRUE` and `quiet` = `FALSE`, streams stdout and stderr from the underlying `processx::run` calls.

`spinner` A logical. When `TRUE` and `quiet` = `FALSE`, shows a spinner instead of live logs; default `TRUE`.

`echo_cmd` A logical. When `TRUE` and `quiet` = `FALSE`, prints each command before running; default `FALSE`.

Value

An object of class `osrm_job` with the following elements:

osrm_job_artifact The path to the generated `.osrm.timestamp` file.

osrm_working_dir The directory containing all OSRM files.

logs The `processx::run` result object.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  # Install OSRM (temporary, session PATH)
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # copy example OSM PBF into a temporary workspace to avoid polluting pkg data
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)

  # Find the path to the profile first
  car_profile <- osrm_find_profile("car.lua")
  # extract OSRM graph files
  result <- osrm_extract(
    input_osm                = tmp_pbf,
    profile                  = car_profile,
    overwrite                = TRUE,
    threads                  = 1L
  )
  # path to generated .osrm files (specifically, the .osrm.timestamp file)
  result$osrm_job_artifact

  # Clean up binaries and workspace
  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

`osrm_find_profile` *Locate an OSRM Lua profile (e.g. car.lua) in a host installation*

Description

By default OSRM ships profiles for "car", "bike" and "foot" in a `profiles/` directory alongside the binaries. This function will try to locate `osrm-routed` on the `PATH`, resolve symlinks, and look first for a `profiles/` directory next to the binary (as placed there by `osrm_install()`). If that fails, it looks for sibling directories `share/osrm/profiles` and `share/osrm-backend/profiles`. IF that fails, it will try to fall back on `/usr/local/share/osrm/profiles`,`/usr/local/share/osrm-backend/profiles`, `/usr/share/osrm/profiles`, and `/usr/share/osrm-backend/profiles`.

Usage

```
osrm_find_profile(profile = "car.lua")
```

Arguments

`profile`	A single string, the name of the Lua profile file (e.g. `"car.lua"`). Defaults to `"car.lua"`.

Value

The normalized filesystem path to the profile.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )
  osrm_find_profile("car.lua")
  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
}
```

`osrm_install` *Install OSRM Backend Binaries*

Description

Downloads and installs pre-compiled binaries for the OSRM backend from the official GitHub releases. The function automatically detects the user's operating system and architecture to download the appropriate files. Only the latest v5 release (`v5.27.1`) and `v6.0.0` were manually tested and are known to work well; other releases available on GitHub can be installed but are not guranteed to function correctly.

Usage

```
osrm_install(
  version = "latest",
  dest_dir = NULL,
  force = FALSE,
  path_action = c("session", "project", "none"),
  quiet = FALSE
)
```

Arguments

`version`	A string specifying the OSRM version tag to install. Defaults to `"latest"`. Use `"latest"` to automatically find the most recent stable version (internally calls `osrm_check_latest_version()`). Versions other than `v5.27.1` and `v6.0.0` will trigger a warning but are still attempted if binaries are available.
`dest_dir`	A string specifying the directory where OSRM binaries should be installed. If `NULL` (the default), a user-friendly, persistent location is chosen via `tools::R_user_dir("osrm.backend` `which = "cache")`, and the binaries are installed into a subdirectory named after the OSRM version (e.g. `.../cache/v6.0.0`).

`force`	A logical value. If `TRUE`, reinstall OSRM even if it's already found in `dest_dir`. If `FALSE` (default), the function will stop if an existing installation is detected.
`path_action`	A string specifying how to handle the system `PATH`. One of: • `"session"` (default): Adds the OSRM bin directory to the `PATH` for the current R session only. • `"project"`: Modifies the `.Rprofile` in the current project to set the `PATH` for all future sessions in that project. • `"none"`: Does not modify the `PATH`.
`quiet`	A logical value. If `TRUE`, suppresses installer messages and warnings. Defaults to `FALSE`.

Details

The function performs the following steps:

1. Queries the GitHub API to find the specified release of `Project-OSRM/osrm-backend`.
2. Identifies the correct binary (`.tar.gz` archive) for the user's OS (Linux, macOS, or Windows) and architecture (x64, arm64).
3. Downloads the archive to a temporary location.
4. Extracts the archive and locates the OSRM executables (e.g., `osrm-routed`, `osrm-extract`).
5. Copies these executables to a local directory (defaults to `file.path(tools::R_user_dir("osrm.backend", which =`
6. Downloads the matching Lua profiles from the release tarball and installs them alongside the binaries.
7. Optionally modifies the `PATH` environment variable for the current session or project.

macOS users should note that upstream OSRM v6.x binaries are built for macOS 15.0 (Sequoia, Darwin 24.0.0) or newer. `osrm_install()` automatically blocks v6 installs on older macOS releases and, when `version = "latest"`, selects the most recent v5 build instead while warning about the requirement. Warnings include both the marketing version and Darwin kernel so you'll see messages like `macOS 13 Ventura [Darwin 22.6.0]`.

When installing OSRM v6.x for Windows, the upstream release omits the Intel Threading Building Blocks (TBB) runtime and a compatible `bz2` DLL. To keep the executables runnable out of the box, `osrm_install()` fetches TBB from oneTBB v2022.3.0 and the BZip2 runtime from bzip2-windows v1.0.8.0, verifying their SHA-256 checksums before extraction. Without these extra libraries, the OSRM v6 binaries shipped for Windows cannot start.

On macOS, OSRM v6.x binaries also miss the bundled TBB runtime. The installer reuses the libraries from release `v5.27.1` to keep the binaries functional and patches their `libbz2` linkage using `install_name_tool` so that they load the system-provided BZip2 runtime.

Power users (including package authors running cross-platform tests) can override the auto-detected platform by setting the R options `osrm.backend.override_os` and `osrm.backend.override_arch` (e.g., `options(osrm.backend.override_os = "linux", osrm.backend.override_arch = "arm64")`) before calling `osrm_install()`. Overrides allow requesting binaries for any OS and CPU combination that exists on the GitHub releases.

Value

The path to the installation directory.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  old <- setwd(tempdir())
  on.exit(setwd(old), add = TRUE)

  # Install the default stable version and set PATH for this session
  install_dir <- osrm_install(path_action = "session", quiet = TRUE)

  # Install for a project non-interactively (e.g., in a script)
  osrm_install(path_action = "project", quiet = TRUE, force = TRUE)

  # Clean up the project's .Rprofile and uninstall binaries
  osrm_clear_path(quiet = TRUE)
  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
}
```

`osrm_partition` *Partition OSRM Graph for Multi-Level Dijkstra (MLD)*

Description

Run the `osrm-partition` tool to partition an OSRM graph for the MLD pipeline. After running, a companion <base>`.osrm.partition` file must exist to confirm success.

Usage

```
osrm_partition(
  input_osrm,
  threads = 8L,
  verbosity = c("INFO", "NONE", "ERROR", "WARNING", "DEBUG"),
  balance = 1.2,
  boundary = 0.25,
  optimizing_cuts = 10L,
  small_component_size = 1000L,
  max_cell_sizes = c(128, 4096, 65536, 2097152),
  quiet = FALSE,
  verbose = FALSE,
  spinner = TRUE,
  echo_cmd = FALSE
)
```

Arguments

`input_osrm`	A string. Path to a `.osrm.timestamp` file, the base path to the `.osrm` files (without extension), or a directory containing exactly one `.osrm.timestamp` file.
`threads`	An integer. Number of threads to use; default `8` (osrm-partition's default).
`verbosity`	A string. Log verbosity level passed to `-l/--verbosity` (one of `"NONE","ERROR","WARNING","INFO",'` default `"INFO"`.
`balance`	A numeric. Balance for left and right side in single bisection; default `1.2`.
`boundary`	A numeric. Percentage of embedded nodes to contract as sources and sinks; default `0.25`.
`optimizing_cuts`	An integer. Number of cuts to use for optimizing a single bisection; default `10`.
`small_component_size`	An integer. Size threshold for small components; default `1000`.
`max_cell_sizes`	A numeric vector. Maximum cell sizes starting from level 1; default `c(128,4096,65536,2097152)`.
`quiet`	A logical. Master switch that suppresses package messages and process output when `TRUE`; default `FALSE`.
`verbose`	A logical. When `TRUE` and `quiet = FALSE`, streams stdout and stderr from the underlying `processx::run` calls.
`spinner`	A logical. When `TRUE` and `quiet = FALSE`, shows a spinner instead of live logs; default `TRUE`.
`echo_cmd`	A logical. When `TRUE` and `quiet = FALSE`, prints each command before running; default `FALSE`.

Value

An object of class `osrm_job` with the following elements:

osrm_job_artifact The path to the partitioned `.osrm.partition` file.

osrm_working_dir The directory containing all OSRM files.

logs The `processx::run` result object.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # Prepare a small graph then partition it for the MLD pipeline
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  profile <- osrm_find_profile("car.lua")

  extract_job <- osrm_extract(
    input_osm = tmp_pbf,
    profile = profile,
    overwrite = TRUE,
    threads = 1L
  )

  partition_job <- osrm_partition(extract_job, threads = 1L, verbose = TRUE)
  partition_job$osrm_job_artifact

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

osrm_prepare_graph *Prepare OSRM Graph for Routing (Extract + Partition/Contract)*

Description

High-level wrapper that first runs `osrm-extract` on an OSM file to produce the base `.osrm` graph, then prepares it for routing via either the MLD pipeline (`osrm-partition` + `osrm-customize`) or the CH pipeline (`osrm-contract`).

Usage

```
osrm_prepare_graph(
  input_osm,
  profile = osrm_find_profile("car.lua"),
  threads = 8L,
  overwrite = FALSE,
  verbosity = c("INFO", "NONE", "ERROR", "WARNING", "DEBUG"),
  data_version = NULL,
  small_component_size = 1000L,
  with_osm_metadata = FALSE,
  parse_conditional_restrictions = FALSE,
  location_dependent_data = NULL,
  disable_location_cache = FALSE,
  dump_nbg_graph = FALSE,
  algorithm = c("mld", "ch"),
  balance = 1.2,
  boundary = 0.25,
  optimizing_cuts = 10L,
  max_cell_sizes = c(128, 4096, 65536, 2097152),
  quiet = FALSE,
  verbose = FALSE,
  spinner = TRUE,
  echo_cmd = FALSE
)
```

Arguments

`input_osm`	A string. Path to the input OSM file (`.osm`, `.osm.bz2`, or `.osm.pbf`) or a directory containing exactly one OSM file with a supported extension.
`profile`	A string. Path to the OSRM Lua profile (e.g. returned by `osrm_find_profile("car.lua")`).
`threads`	An integer. Number of threads for extract and partition/contract; default `8`.
`overwrite`	A logical. If `FALSE`, stops if any existing `.osrm*` files matching the base name are found alongside `input_osm`. Set to `TRUE` to overwrite them.
`verbosity`	A string. Log verbosity for extract/partition/contract (one of `"NONE"`,`"ERROR"`,`"WARNING"`,`"INFO"`,`"DEB` default `"INFO"`.
`data_version`	A string or `NULL`. Passed to `osrm-extract` via `-d`; default `NULL`.
`small_component_size`	An integer. For extract & partition; default `1000`.
`with_osm_metadata`	A logical. Adds `--with-osm-metadata` during extract; default `FALSE`.
`parse_conditional_restrictions`	A logical. Adds `--parse-conditional-restrictions`; default `FALSE`.
`location_dependent_data`	A string or `NULL`. Path to GeoJSON for extract; default `NULL`.
`disable_location_cache`	A logical. Adds `--disable-location-cache`; default `FALSE`.
`dump_nbg_graph`	A logical. Adds `--dump-nbg-graph`; default `FALSE`.
`algorithm`	A string. One of `"mld"` (default) or `"ch"`.
`balance`	A numeric. Balance for `osrm-partition`; default `1.2`.
`boundary`	A numeric. Boundary percentage for `osrm-partition`; default `0.25`.
`optimizing_cuts`	An integer. Optimizing cuts for `osrm-partition`; default `10`.
`max_cell_sizes`	A numeric vector. Max cell sizes for `osrm-partition`; default `c(128,4096,65536,2097152)`.
`quiet`	A logical. Master switch that suppresses package messages and process output when `TRUE`; default `FALSE`.
`verbose`	A logical. When `TRUE` and `quiet = FALSE`, streams stdout and stderr from the underlying `processx::run` calls.
`spinner`	A logical. When `TRUE` and `quiet = FALSE`, shows a spinner instead of live logs; default `TRUE`.
`echo_cmd`	A logical. When `TRUE` and `quiet = FALSE`, prints each command before running; default `FALSE`.

Value

An object of class `osrm_job` with the following elements:

osrm_job_artifact The path to the final routing-ready graph file (`.osrm.hsgr` for CH or `.osrm.mldgr` for MLD).

osrm_working_dir The directory containing all OSRM files.

logs A list of `processx::run` results for each stage: `extract`, `partition`/`contract`, and `customize` (if MLD).

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # Prepare a routing-ready graph with the default MLD pipeline
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)

  graph <- osrm_prepare_graph(
    input_osm = tmp_pbf,
    overwrite = TRUE,
    threads = 1L,
    algorithm = "mld"
  )
  graph$osrm_job_artifact

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

`osrm_servers` *List OSRM servers started via this package*

Description

Returns a snapshot of servers registered by `osrm_start_server()` or `osrm_start()`. You can stop one by passing its `id`, `port`, or `pid` to `osrm_stop()`.

Usage

```
osrm_servers()
```

Value

A data.frame with columns: `id`, `pid`, `port`, `algorithm`, `started_at`, `alive`, `has_handle`.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # copy example OSM PBF into a temporary workspace to avoid polluting pkg data
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  graph <- osrm_prepare_graph(tmp_pbf, overwrite = TRUE, threads = 1L)

  srv <- osrm_start_server(graph$osrm_job_artifact, port = 6000, threads = 1L)
  osrm_servers()
  osrm_stop(srv)

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

`osrm_start` *Start an OSRM Server with Automatic Setup*

Description

A high-level, "one-shot" function to start an OSRM server that automatically handles OSRM installation and graph preparation. This is the recommended function for most users to get a server running quickly with minimal steps.

Usage

```
osrm_start(
  path,
  algorithm = c("mld", "ch"),
  quiet = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`path`	A string. Path to the input data. Can be one of: • A path to an OSM file (e.g., `/path/to/data.osm.pbf`). • A path to a directory containing OSRM graph files or an OSM file. • A direct path to a final graph file (`.osrm.mldgr` or `.osrm.hsgr`).
`algorithm`	A string specifying the routing algorithm to use for graph preparation, either `"mld"` (Multi-Level Dijkstra, default) or `"ch"` (Contraction Hierarchies). This is only used when `osrm_prepare_graph` is automatically called.
`quiet`	A logical value. If `TRUE`, suppresses installer messages and warnings. Defaults to `FALSE`.
`verbose`	A logical. If `FALSE` (default), suppresses detailed console output from backend commands. If `TRUE`, shows all output, which is useful for debugging.
`...`	Additional arguments passed on to `osrm_prepare_graph()` (e.g., `overwrite = TRUE`) and `osrm_start_server()` (e.g., `port = 5001`).

Details

This function is designed for convenience and automates the entire setup process. By default, it is not verbose and only prints high-level status messages.

1. **Check for OSRM Installation:** It first verifies if the `osrm-routed` executable is available in the system's `PATH`. If not, it automatically calls `osrm_install()` to download and install the latest stable version.
2. **Process Input Path and Prepare Graph:** The function intelligently handles the `path` argument to find or create the necessary graph files. If the graph files do not exist, it automatically runs `osrm_prepare_graph()` to build them, which may take some time.
3. **Start Server:** Once the graph files are located or prepared, it launches the `osrm-routed` server and prints a confirmation message with the server's PID and port.

For advanced users or for debugging, set `verbose = TRUE` to see the detailed console output from the installation and graph preparation steps. For full manual control, use the lower-level functions like `osrm_prepare_graph()` and `osrm_start_server()` directly.

Value

A `processx::process` object for the running server.

See Also

osrm_stop(), osrm_start_server() for manual server startup.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # copy example OSM PBF into a temporary workspace to avoid polluting pkg data
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  local_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = local_pbf, overwrite = TRUE)

  # Start the server with one command.
  # It will quietly install OSRM and prepare the graph if needed.
  osrm_process <- osrm_start(local_pbf)

  # Stop the server when done.
  osrm_stop()

  # To see all backend logs during setup, use verbose = TRUE
  osrm_process_verbose <- osrm_start(local_pbf, verbose = TRUE)
  osrm_stop()

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

---

osrm_start_server   *Start an OSRM MLD/CH server with* `osrm-routed`

Description

Launches an `osrm-routed` process pointing at a localized OSRM graph (either `.osrm.mldgr` for MLD or `.osrm.hsgr` for CH/CoreCH).

Usage

```
osrm_start_server(
  osrm_path,
  version = FALSE,
  help = FALSE,
  verbosity = c("INFO", "ERROR", "WARNING", "NONE", "DEBUG"),
  trial = FALSE,
  ip = "0.0.0.0",
  port = 5001L,
  threads = 8L,
  shared_memory = FALSE,
  memory_file = NULL,
  mmap = FALSE,
  dataset_name = NULL,
  algorithm = NULL,
  max_viaroute_size = 500L,
  max_trip_size = 100L,
  max_table_size = 100L,
  max_matching_size = 100L,
  max_nearest_size = 100L,
  max_alternatives = 3L,
  max_matching_radius = -1L,
  quiet = FALSE,
  verbose = FALSE,
  echo_cmd = FALSE
)
```

Arguments

`osrm_path`	Character(1). Path to the `.osrm.mldgr` or `.osrm.hsgr` file
`version`	Logical; if `TRUE`, prints version and exits
`help`	Logical; if `TRUE`, prints help and exits
`verbosity`	Character; one of `"NONE"`,`"ERROR"`,`"WARNING"`,`"INFO"`,`"DEBUG"`
`trial`	Logical or integer; if `TRUE` or >0, quits after initialization (default: `FALSE`)
`ip`	Character; IP address to bind (default: `"0.0.0.0"`)
`port`	Integer; TCP port to listen on (default: `5001`)
`threads`	Integer; number of worker threads (default: `8`)
`shared_memory`	Logical; load graph from shared memory (default: `FALSE`)
`memory_file`	Character or NULL; DEPRECATED (behaves like `mmap`)
`mmap`	Logical; memory-map data files (default: `FALSE`)
`dataset_name`	Character or NULL; name of shared memory dataset
`algorithm`	Character or NULL; one of `"CH"`,`"CoreCH"`,`"MLD"`. If `NULL` (default), auto-selected based on file extension
`max_viaroute_size`	Integer (default: `500`)

`max_trip_size`	Integer (default: `100`)
`max_table_size`	Integer (default: `100`)
`max_matching_size`	Integer (default: `100`)
`max_nearest_size`	Integer (default: `100`)
`max_alternatives`	Integer (default: `3`)
`max_matching_radius`	Integer; use `-1` for unlimited (default: `-1`)
`quiet`	Logical; when `TRUE`, suppresses package messages.
`verbose`	Logical; reserved for controlling future server verbosity, included for interface consistency with `osrm_start()`. Defaults to `FALSE`.
`echo_cmd`	Logical; echo command line to console before launch (default: `FALSE`)

Details

The server's standard output and error streams can be controlled via R options for non-interactive use or persistent logging. By default, they are captured as pipes, which allows for reading output directly within the R session (e.g., via `osrm_server$read_output_lines()`).

To redirect the server's output to one or more files, you can set the `osrm.server.log_file` R option before calling this function:

- **Combined Log:** To send both `stdout` and `stderr` to a single file, provide a file path: `options(osrm.server.log_fil` `= "path/to/osrm.log")`
- **Separate Logs:** To send `stdout` and `stderr` to separate files, provide a named list: `options(osrm.server.log_file` `= list(stdout = "out.log", stderr = "err.log"))`
- **Default Behavior:** To restore the default behavior of using pipes, set the option to `NULL`: `options(osrm.server.log_file = NULL)`

You can override the `osrm-routed` executable via `options(osrm.routed.exec = "/full/path/to/osrm-routed")`.

Value

A `processx::process` object for the running server (also registered internally).

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # Build a graph then launch an OSRM server on a custom port
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)

  graph <- osrm_prepare_graph(
    input_osm = tmp_pbf,
    overwrite = TRUE,
    threads = 1L,
    algorithm = "mld"
  )

  server <- osrm_start_server(
    osrm_path = graph$osrm_job_artifact,
    port = 6000,
    threads = 1L
  )

  # Later, stop the server again
  osrm_stop(server)

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

osrm_stop *Stop an OSRM Server*

Description

Terminates an `osrm-routed` process launched by `osrm_start()` or `osrm_start_server()`.

Usage

```
osrm_stop(
  server = NULL,
  id = NULL,
  port = NULL,
  pid = NULL,
  wait = 1000L,
  quiet = FALSE
)
```

Arguments

`server`	Optional `processx::process` object returned by `osrm_start_server()`.
`id`	Optional character id from `osrm_servers()`.
`port`	Optional integer TCP port.
`pid`	Optional integer process id.
`wait`	Integer milliseconds to wait for clean shutdown (default `1000`).
`quiet`	Logical; suppress messages (default `FALSE`).

Details

This function provides a flexible way to stop a running OSRM process. If no arguments are specified, it defaults to stopping the most recently started server that is still alive.

You can also stop a specific server by providing:

- The `processx::process` object returned by `osrm_start()` or `osrm_start_server()`.
- The server's `id`, `port`, or `pid` (use `osrm_servers()` to find these).

Value

A list with fields `id`, `pid`, `port`, `stopped` (logical).

See Also

`osrm_start()`, `osrm_servers()`, `osrm_stop_all()`

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # copy example OSM PBF into a temporary workspace to avoid polluting pkg data
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  graph <- osrm_prepare_graph(tmp_pbf, overwrite = TRUE, threads = 1L)

  srv <- osrm_start_server(graph$osrm_job_artifact, port = 6000, threads = 1L)

  # Stop by passing the process object
  osrm_stop(srv)

  # Or stop by port after the process is registered
  osrm_stop(port = 6000)
  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

`osrm_stop_all` *Stop all running OSRM servers started via this package*

Description

Stop all running OSRM servers started via this package

Usage

```
osrm_stop_all()
```

Value

The number of servers attempted.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  graph <- osrm_prepare_graph(tmp_pbf, overwrite = TRUE, threads = 1L)

  srv <- osrm_start_server(graph$osrm_job_artifact, port = 6000, threads = 1L)
  stopped <- osrm_stop_all()
  stopped

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

osrm_uninstall *Uninstall OSRM Backend Binaries*

Description

Removes the OSRM backend binaries and optionally clears the PATH configuration from the project's .Rprofile.

Usage

```
osrm_uninstall(
  dest_dir = NULL,
  clear_path = TRUE,
  quiet = FALSE,
  all = FALSE,
  force = FALSE
)
```

Arguments

dest_dir	A string specifying the directory from which to remove OSRM binaries. If NULL (the default), the function looks for an installation in the per-version subdirectories inside tools::R_user_dir("osrm.backend", which = "cache") and removes it. When multiple versions are installed, interactive sessions that are not quiet will be prompted (with a numbered menu and 0 to cancel) to choose a directory; otherwise, dest_dir must be supplied. Ignored if all = TRUE.
clear_path	A logical value. If TRUE (default), also removes the PATH configuration from the project's .Rprofile by calling osrm_clear_path().
quiet	A logical value. If TRUE, suppresses informational messages and confirmation prompts. Defaults to FALSE.
all	A logical value. If TRUE, removes all OSRM installations found in the default cache directory. Will prompt for confirmation unless force = TRUE. Defaults to FALSE. When TRUE, the dest_dir parameter is ignored.
force	A logical value. If TRUE, skips all confirmation prompts, enabling non-interactive usage. Defaults to FALSE.

Value

TRUE if one or more directories were successfully removed, and FALSE otherwise.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  # Install OSRM temporarily
  install_dir <- osrm_install(path_action = "session", quiet = TRUE)

  # Uninstall that specific version and clear PATH changes
  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )

  # If multiple installs exist, remove them all
  osrm_uninstall(all = TRUE, force = TRUE, quiet = TRUE)
}
```

`osrm_which` *Locate the OSRM Installation Used by* `osrm.backend`

Description

Resolves the `osrm-routed` executable available on the current `PATH` (or the override provided via `options(osrm.routed.exec)`). Runs `osrm-routed --version` to verify availability, then prints the directory containing the executable together with the backend version reported by `osrm-routed` so you know what will be used in the current session.

Usage

```
osrm_which(quiet = FALSE)
```

Arguments

`quiet`	Logical; if `FALSE` (default), prints information about the located installation. If `TRUE`, suppresses printed output and only returns the information as a list.

Value

A list with components `executable` (full path to `osrm-routed`), `directory` (its parent folder), `osrm_version` (character vector of non-empty lines emitted by `osrm-routed --version`), and the raw `processx::run` result in `logs`.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # check which OSRM installation will be used
  osrm_which()

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
}
```

Index