Package 'circuitscaper'

November 6, 2026

Title 'Circuitscape' and 'Omniscape' Connectivity Analysis via 'Julia'

Version 0.2.0

Author Matthew Kling [aut, cre, cph]

Maintainer Matthew Kling <mattkling@berkeley.edu>

Description Provides an R-native interface to the 'Circuitscape.jl' and 'Omniscape.jl' 'Julia' packages for landscape connectivity modeling using circuit theory. Users work entirely in R with familiar objects (SpatRaster, file paths) while 'Julia' handles computation invisibly. Supports all four 'Circuitscape' modes (pairwise, one-to-all, all-to-one, advanced) and 'Omniscape' moving-window analysis. Methods are described in McRae (2006) <doi:10.1111/j.0014-3820.2006.tb00500.x> and Landau et al. (2021) <doi:10.21105/joss.02829>.

License MIT + file LICENSE

URL https://github.com/matthewkling/circuitscaper, https://matthewkling.github.io/circuitscaper/

BugReports https://github.com/matthewkling/circuitscaper/issues

Depends R (>= 4.0)

Imports terra, JuliaConnectoR (>= 1.1.3), rjson, tools, utils

Suggests testthat (>= 3.0.0), withr, knitr, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

Config/testthat/edition 3

SystemRequirements Julia (>= 1.10), Circuitscape.jl, Omniscape.jl

Config/roxygen2/version 8.0.0

NeedsCompilation no

Repository CRAN

Date/Publication 2026-11-06 16:40:27 UTC

Contents

cs_advanced 2
cs_all_to_one 4
cs_install_julia 6
cs_julia_available 7
cs_one_to_all 8
cs_pairwise 10
cs_setup 13
os_condition 14
os_run 16

Index **19**

`cs_advanced` *Advanced Circuitscape Analysis*

Description

Solve a single circuit with user-specified source and ground layers.

Usage

```
cs_advanced(
  resistance,
  source,
  ground,
  resistance_is = "resistances",
  ground_is = "resistances",
  use_unit_currents = FALSE,
  use_direct_grounds = FALSE,
  short_circuit = NULL,
  source_ground_conflict = "keepall",
  four_neighbors = FALSE,
  avg_resistances = FALSE,
  solver = "cg+amg",
  output_dir = NULL,
  verbose = FALSE
)
```

Arguments

`resistance`	A terra::SpatRaster or file path. The resistance (or conductance) surface. Higher values represent greater resistance to movement.
`source`	A terra::SpatRaster or file path. Source current strengths (amps per cell). Cells with value 0 or NA are not sources.

`ground` A terra::SpatRaster or file path. Ground node values. Interpretation depends on `ground_is`: resistances to ground (default) or conductances to ground. Cells with value 0 or NA are not grounds.

`resistance_is` Character. Whether the resistance surface represents `"resistances"` (default) or `"conductances"`.

`ground_is` Character. Whether the ground raster values represent `"resistances"` (default) or `"conductances"` to ground.

`use_unit_currents`
Logical. If `TRUE`, all current sources are set to 1 amp regardless of the values in the source raster. Default `FALSE`.

`use_direct_grounds`
Logical. If `TRUE`, all ground nodes are tied directly to ground (zero resistance), regardless of the values in the ground raster. Default `FALSE`.

`short_circuit` Optional terra::SpatRaster or file path. Raster identifying short-circuit regions (aka polygons). Cells sharing the same positive integer value are treated as short-circuit regions with zero resistance between them. Default `NULL` (no short-circuit regions).

`source_ground_conflict`
Character. How to resolve cells that appear in both the source and ground rasters: `"keepall"` (default, keep both), `"rmvsrc"` (remove source), `"rmvgnd"` (remove ground), or `"rmvall"` (remove both).

`four_neighbors` Logical. Use 4-neighbor (rook) connectivity instead of 8-neighbor (queen). Default `FALSE`.

`avg_resistances`
Logical. When using 8-neighbor connectivity, compute the resistance of diagonal connections as the average of the two cells rather than their sum. Default `FALSE` (Circuitscape default). Ignored when `four_neighbors = TRUE`.

`solver` Character. Solver to use: `"cg+amg"` (default) or `"cholmod"`.

`output_dir` Optional character path. If provided, output files persist there. Default `NULL` uses a temporary directory.

`verbose` Logical. Print Circuitscape solver output. Default `FALSE`.

Details

Unlike the other Circuitscape modes, advanced mode does not iterate over focal nodes. Instead, the user provides explicit source current and ground conductance rasters, and a single circuit is solved. This gives full control over the current injection pattern and is useful for modeling specific scenarios such as directional movement between a defined source area and destination.

Value

A terra::SpatRaster with the following layers:

current Current density at each cell.

voltage Voltage at each cell. Voltage is analogous to movement probability and decreases with distance from sources.

References

McRae, B.H. (2006). Isolation by resistance. *Evolution*, 60(8), 1551–1561. doi:10.1111/j.0014-3820.2006.tb00500.x

Circuitscape.jl: https://docs.circuitscape.org/Circuitscape.jl/latest/

See Also

cs_pairwise(), cs_one_to_all(), cs_all_to_one(), cs_setup()

Examples

```
library(terra)
res <- rast(system.file("extdata/resistance.tif", package = "circuitscaper"))
origin <- rast(system.file("extdata/source.tif", package = "circuitscaper"))
dest <- rast(system.file("extdata/ground.tif", package = "circuitscaper"))
result <- cs_advanced(res, origin, dest, ground_is = "conductances")
plot(result)
```

---

cs_all_to_one *All-to-One Circuitscape Analysis*

Description

For each focal node in turn, inject current at all other focal nodes and ground that single node.

Usage

```
cs_all_to_one(
  resistance,
  locations,
  resistance_is = "resistances",
  four_neighbors = FALSE,
  avg_resistances = FALSE,
  short_circuit = NULL,
  included_pairs = NULL,
  write_voltage = FALSE,
  cumulative_only = TRUE,
  source_strengths = NULL,
  solver = "cg+amg",
  output_dir = NULL,
  verbose = FALSE
)
```

Arguments

`resistance`	A terra::SpatRaster or file path. The resistance (or conductance) surface. Higher values represent greater resistance to movement. Use the `resistance_is` argument if your surface represents conductances instead.
`locations`	Focal node locations, provided as any of: • A terra::SpatRaster (or `raster::RasterLayer`) with positive integer IDs identifying each node. Cells with value 0 or `NA` are not treated as focal nodes. • A file path to a raster file (e.g., `.tif`, `.asc`). • A two-column matrix or data.frame of x/y coordinates. Each row becomes a focal node, auto-assigned IDs 1, 2, 3, ... in row order. Coordinates are snapped to the nearest cell of the `resistance` raster.
`resistance_is`	Character. Whether the resistance surface represents `"resistances"` (default) or `"conductances"`.
`four_neighbors`	Logical. Use 4-neighbor (rook) connectivity instead of 8-neighbor (queen). Default `FALSE`.
`avg_resistances`	Logical. When using 8-neighbor connectivity, compute the resistance of diagonal connections as the average of the two cells rather than their sum. Default `FALSE` (Circuitscape default). Ignored when `four_neighbors = TRUE`.
`short_circuit`	Optional terra::SpatRaster or file path. Raster identifying short-circuit regions (aka polygons). Cells sharing the same positive integer value are treated as short-circuit regions with zero resistance between them. Default `NULL` (no short-circuit regions).
`included_pairs`	Optional character file path. A text file specifying which pairs of focal nodes to include or exclude from analysis. See the Circuitscape documentation for the file format. Default `NULL` (all pairs).
`write_voltage`	Logical. Write voltage maps. Default `FALSE`. When `TRUE`, per-iteration voltage layers (named `voltage_1`, `voltage_2`, ...) are included in the output raster.
`cumulative_only`	Logical. If `TRUE` (default), only the cumulative current map is returned. If `FALSE`, per-iteration current layers (named `current_1`, `current_2`, ...) are also included. Use with caution for large numbers of focal nodes, as this can produce many layers.
`source_strengths`	Optional. Variable current injection strengths for each focal node. Can be: • A numeric vector with one value per focal node (in the same order as the locations input). Node IDs are assigned 1, 2, 3, ... matching the order. • A character file path to a tab-delimited text file with two columns: node ID and strength in amps. Nodes not listed default to 1 amp. Default `NULL` (all nodes inject 1 amp).
`solver`	Character. Solver to use: `"cg+amg"` (default) or `"cholmod"`.
`output_dir`	Optional character path. If provided, output files persist there. Default `NULL` uses a temporary directory that is cleaned up automatically.
`verbose`	Logical. Print Circuitscape solver output. Default `FALSE`.

Details

All-to-one mode iterates over each focal node. In each iteration, all other focal nodes are injected with 1 amp of current each, and the focal node is connected to ground. This produces a current map showing how current converges on that node from across the landscape.

This mode is useful for identifying the most accessible or reachable sites in the network, emphasizing current flow toward each ground node. The cumulative map sums across all iterations and highlights cells that are important for connectivity across the full set of nodes.

Value

A terra::SpatRaster with the following layers:

cumulative_current Current flow summed across all iterations.

current_*N* Per-node current map for focal node *N*, where *N* is the integer node ID from the `locations` raster. One layer per focal node.

References

McRae, B.H. (2006). Isolation by resistance. *Evolution*, 60(8), 1551–1561. doi:10.1111/j.0014-3820.2006.tb00500.x

Circuitscape.jl: https://docs.circuitscape.org/Circuitscape.jl/latest/

See Also

`cs_pairwise()`, `cs_one_to_all()`, `cs_advanced()`, `cs_setup()`

Examples

```
library(terra)
res <- rast(system.file("extdata/resistance.tif", package = "circuitscaper"))
coords <- matrix(c(10, 40, 40, 40, 10, 10, 40, 10), ncol = 2, byrow = TRUE)
result <- cs_all_to_one(res, coords)
plot(result)
```

`cs_install_julia` *Install Julia and Required Packages*

Description

Downloads and installs Julia, Circuitscape.jl, and Omniscape.jl. This is the recommended first step after installing the circuitscaper R package.

Usage

```
cs_install_julia(force = FALSE, version = "latest")
```

Arguments

force	Logical. If TRUE, download and install Julia even if a Julia installation is already present, and reinstall the Julia packages. Use this to replace a Julia installation that is too old. Default FALSE.
version	Character. Julia version to install, e.g. "1.12.1". Default "latest", the latest stable release. Must be at least 1.10.

Details

In interactive sessions, prompts for confirmation before downloading. In non-interactive sessions (e.g., CI), proceeds without prompting.

If no suitable Julia is found (see cs_setup() for where it looks), or if force = TRUE, the official Julia release for your platform is downloaded from julialang.org, checked against its published size and checksum, and unpacked into circuitscaper's user data directory (tools::R_user_dir("circuitscaper", "data")). That Julia is then used by cs_setup() in later sessions. Julia installed by other means (e.g., juliaup or the official installers) works too; in that case cs_install_julia() just adds the Julia packages.

The Julia packages are installed into Julia's default package depot (usually ~/.julia), as with any Julia installation.

Value

Invisibly returns TRUE on success, FALSE if cancelled.

Examples

```
## Not run:
cs_install_julia()

## End(Not run)
```

---

cs_julia_available *Check if Julia and Required Packages Are Available*

Description

Tests whether Julia is installed and the Circuitscape and Omniscape Julia packages can be loaded. This is a lightweight check that does not initialize a full Julia session. It is used internally by example code and can be called by users to verify their setup before running analyses.

Usage

```
cs_julia_available()
```

Value

TRUE if Julia is found (in the places cs_setup() searches) and the 'Circuitscape' and 'Omniscape' Julia packages load successfully, FALSE otherwise.

Examples

```
cs_julia_available()
```

---

cs_one_to_all *One-to-All Circuitscape Analysis*

Description

For each focal node in turn, inject current at that node and ground all other focal nodes simultaneously.

Usage

```
cs_one_to_all(
  resistance,
  locations,
  resistance_is = "resistances",
  four_neighbors = FALSE,
  avg_resistances = FALSE,
  short_circuit = NULL,
  included_pairs = NULL,
  write_voltage = FALSE,
  cumulative_only = TRUE,
  source_strengths = NULL,
  solver = "cg+amg",
  output_dir = NULL,
  verbose = FALSE
)
```

Arguments

resistance	A terra::SpatRaster or file path. The resistance (or conductance) surface. Higher values represent greater resistance to movement. Use the resistance_is argument if your surface represents conductances instead.
locations	Focal node locations, provided as any of: • A terra::SpatRaster (or raster::RasterLayer) with positive integer IDs identifying each node. Cells with value 0 or NA are not treated as focal nodes. • A file path to a raster file (e.g., .tif, .asc).

- A two-column matrix or data.frame of x/y coordinates. Each row becomes a focal node, auto-assigned IDs 1, 2, 3, ... in row order. Coordinates are snapped to the nearest cell of the `resistance` raster.

`resistance_is` Character. Whether the resistance surface represents `"resistances"` (default) or `"conductances"`.

`four_neighbors` Logical. Use 4-neighbor (rook) connectivity instead of 8-neighbor (queen). Default `FALSE`.

`avg_resistances`
Logical. When using 8-neighbor connectivity, compute the resistance of diagonal connections as the average of the two cells rather than their sum. Default `FALSE` (Circuitscape default). Ignored when `four_neighbors = TRUE`.

`short_circuit` Optional terra::SpatRaster or file path. Raster identifying short-circuit regions (aka polygons). Cells sharing the same positive integer value are treated as short-circuit regions with zero resistance between them. Default `NULL` (no short-circuit regions).

`included_pairs` Optional character file path. A text file specifying which pairs of focal nodes to include or exclude from analysis. See the Circuitscape documentation for the file format. Default `NULL` (all pairs).

`write_voltage` Logical. Write voltage maps. Default `FALSE`. When `TRUE`, per-iteration voltage layers (named `voltage_1`, `voltage_2`, ...) are included in the output raster.

`cumulative_only`
Logical. If `TRUE` (default), only the cumulative current map is returned. If `FALSE`, per-iteration current layers (named `current_1`, `current_2`, ...) are also included. Use with caution for large numbers of focal nodes, as this can produce many layers.

`source_strengths`
Optional. Variable current injection strengths for each focal node. Can be:

- A numeric vector with one value per focal node (in the same order as the locations input). Node IDs are assigned 1, 2, 3, ... matching the order.
- A character file path to a tab-delimited text file with two columns: node ID and strength in amps. Nodes not listed default to 1 amp. Default `NULL` (all nodes inject 1 amp).

`solver` Character. Solver to use: `"cg+amg"` (default) or `"cholmod"`.

`output_dir` Optional character path. If provided, output files persist there. Default `NULL` uses a temporary directory that is cleaned up automatically.

`verbose` Logical. Print Circuitscape solver output. Default `FALSE`.

Details

One-to-all mode iterates over each focal node. In each iteration, the focal node is injected with 1 amp of current and all remaining focal nodes are simultaneously connected to ground. This produces a current map showing how current spreads from that node through the landscape to reach the others.

This mode is useful for mapping how well each site is connected to the rest of the focal node network, emphasizing current dispersal from each source. The cumulative map sums across all iterations and highlights cells that are important for connectivity across the full set of nodes.

Value

A terra::SpatRaster with the following layers:

cumulative_current Current flow summed across all iterations.

current_*N* Per-node current map for focal node *N*, where *N* is the integer node ID from the `locations` raster. One layer per focal node.

References

McRae, B.H. (2006). Isolation by resistance. *Evolution*, 60(8), 1551–1561. doi:10.1111/j.0014-3820.2006.tb00500.x

Circuitscape.jl: https://docs.circuitscape.org/Circuitscape.jl/latest/

See Also

`cs_pairwise()`, `cs_all_to_one()`, `cs_advanced()`, `cs_setup()`

Examples

```
library(terra)
res <- rast(system.file("extdata/resistance.tif", package = "circuitscaper"))
coords <- matrix(c(10, 40, 40, 40, 10, 10, 40, 10), ncol = 2, byrow = TRUE)
result <- cs_one_to_all(res, coords)
plot(result)
```

---

cs_pairwise *Pairwise Circuitscape Analysis*

Description

Compute pairwise effective resistances and cumulative current flow between all pairs of focal nodes.

Usage

```
cs_pairwise(
  resistance,
  locations,
  resistance_is = "resistances",
  four_neighbors = FALSE,
  avg_resistances = FALSE,
  short_circuit = NULL,
  included_pairs = NULL,
  write_voltage = FALSE,
  cumulative_only = TRUE,
  source_strengths = NULL,
  solver = "cg+amg",
```

```
  output_dir = NULL,
  verbose = FALSE
)
```

Arguments

resistance	A terra::SpatRaster or file path. The resistance (or conductance) surface. Higher values represent greater resistance to movement. Use the `resistance_is` argument if your surface represents conductances instead.
locations	Focal node locations, provided as any of: • A terra::SpatRaster (or `raster::RasterLayer`) with positive integer IDs identifying each node. Cells with value 0 or `NA` are not treated as focal nodes. • A file path to a raster file (e.g., `.tif`, `.asc`). • A two-column matrix or data.frame of x/y coordinates. Each row becomes a focal node, auto-assigned IDs 1, 2, 3, ... in row order. Coordinates are snapped to the nearest cell of the `resistance` raster.
resistance_is	Character. Whether the resistance surface represents `"resistances"` (default) or `"conductances"`.
four_neighbors	Logical. Use 4-neighbor (rook) connectivity instead of 8-neighbor (queen). Default `FALSE`.
avg_resistances	Logical. When using 8-neighbor connectivity, compute the resistance of diagonal connections as the average of the two cells rather than their sum. Default `FALSE` (Circuitscape default). Ignored when `four_neighbors = TRUE`.
short_circuit	Optional terra::SpatRaster or file path. Raster identifying short-circuit regions (aka polygons). Cells sharing the same positive integer value are treated as short-circuit regions with zero resistance between them. Default `NULL` (no short-circuit regions).
included_pairs	Optional character file path. A text file specifying which pairs of focal nodes to include or exclude from analysis. See the Circuitscape documentation for the file format. Default `NULL` (all pairs).
write_voltage	Logical. Write voltage maps. Default `FALSE`. When `TRUE`, per-iteration voltage layers (named `voltage_1`, `voltage_2`, ...) are included in the output raster.
cumulative_only	Logical. If `TRUE` (default), only the cumulative current map is returned. If `FALSE`, per-iteration current layers (named `current_1`, `current_2`, ...) are also included. Use with caution for large numbers of focal nodes, as this can produce many layers.
source_strengths	Optional. Variable current injection strengths for each focal node. Can be: • A numeric vector with one value per focal node (in the same order as the locations input). Node IDs are assigned 1, 2, 3, ... matching the order. • A character file path to a tab-delimited text file with two columns: node ID and strength in amps. Nodes not listed default to 1 amp. Default `NULL` (all nodes inject 1 amp).

`solver`	Character. Solver to use: `"cg+amg"` (default) or `"cholmod"`.
`output_dir`	Optional character path. If provided, output files persist there. Default `NULL` uses a temporary directory that is cleaned up automatically.
`verbose`	Logical. Print Circuitscape solver output. Default `FALSE`.

Details

Pairwise mode iterates over every unique pair of focal nodes. For each pair, one node is injected with 1 amp of current and the other is connected to ground. The effective resistance between the pair is recorded, and the resulting current flow is accumulated across all pairs into a cumulative current map that highlights important movement corridors.

This is the most common Circuitscape mode and is typically used to quantify connectivity between discrete habitat patches or populations. The resistance matrix can be used as a distance metric in analyses such as isolation by resistance.

Value

A named list with:

current_map A terra::SpatRaster. By default contains a single `cumulative_current` layer (current flow summed across all pairs). When `cumulative_only = FALSE`, additional per-pair layers are included (e.g., `current_1_2`, `current_1_3`). When `write_voltage = TRUE`, per-pair voltage layers are included (e.g., `voltage_1_2`, `voltage_1_3`).

resistance_matrix A symmetric numeric matrix of pairwise effective resistances between focal nodes, with node IDs as row and column names.

References

McRae, B.H. (2006). Isolation by resistance. *Evolution*, 60(8), 1551–1561. doi:10.1111/j.0014-3820.2006.tb00500.x

Circuitscape.jl: https://docs.circuitscape.org/Circuitscape.jl/latest/

See Also

`cs_one_to_all()`, `cs_all_to_one()`, `cs_advanced()`, `cs_setup()`

Examples

```
library(terra)
res <- rast(system.file("extdata/resistance.tif", package = "circuitscaper"))
coords <- matrix(c(10, 40, 40, 40, 10, 10, 40, 10), ncol = 2, byrow = TRUE)
result <- cs_pairwise(res, coords, cumulative_only = FALSE)
plot(result$current_map)
result$resistance_matrix
```

`cs_setup` *Set Up Julia and Load Circuitscape/Omniscape*

Description

Initialize the Julia session and load the Circuitscape and Omniscape Julia packages. This is called automatically on first use of any `cs_*` or `os_*` function. Call explicitly to control the Julia path, number of threads, or pre-warm the session.

Usage

```
cs_setup(julia_home = NULL, threads = 1L, quiet = TRUE)
```

Arguments

`julia_home`	Character. Path to the Julia `bin/` directory. If `NULL` (default), Julia is looked for in: the `JULIA_HOME` option or environment variable, the `JULIA_BINDIR` environment variable, a Julia installed by `cs_install_julia()` (or by `JuliaCall::install_julia()`), the system PATH (including a login shell's PATH on macOS and Linux), and common Windows install locations.
`threads`	Integer. Number of Julia threads. Default `1L`. If Julia is already running with fewer threads, it is restarted with `threads` threads (see Threading, below). This setting controls parallelism for `os_run()` only; Circuitscape functions (`cs_*`) run single-threaded regardless of this value.
`quiet`	Logical. Suppress Julia startup messages. Default `TRUE`.

Details

`cs_setup()` does **not** install Julia or Julia packages. If Julia is not found or the required packages are missing, it throws an informative error directing you to `cs_install_julia()`.

`cs_setup()` will:

- Verify that Julia is installed and accessible, and start it. Julia runs in its own process, connected to R by 'JuliaConnectoR'.
- Verify that the Circuitscape and Omniscape Julia packages are installed.
- Load both packages and warm up the JIT compiler.

Once Julia is initialized, it stays warm for the R session. Subsequent calls to `cs_setup()` return immediately. If the Julia process has stopped in the meantime (for example, it crashed or ran out of memory, or was stopped with `JuliaConnectoR::stopJulia()`), the next `cs_setup()` call, or the next `cs_*()` or `os_*()` call, starts and sets up Julia again.

Threading:

Multi-threading is used by `os_run()` when `parallelize = TRUE`. Circuitscape functions (`cs_pairwise`, `cs_one_to_all`, etc.) do not benefit from multiple threads.

```
cs_setup(threads = 4)
os_run(resistance, radius = 50, parallelize = TRUE)
```

Julia's thread count is fixed when Julia starts, so if Julia is already running with fewer threads, `cs_setup(threads = )` restarts it. This takes as long as the first setup, since Circuitscape and Omniscape are loaded and warmed up again, and it discards anything else in this R session's Julia (e.g., objects created through 'JuliaConnectoR' by other code). Asking for fewer threads than Julia has doesn't restart it.

Julia is started with `JULIA_NUM_THREADS` set to `"N,0"`: N threads and no interactive threads. Omniscape releases before 0.6.3 fail in parallel when Julia has an interactive thread, so `cs_setup(threads = )` also restarts a Julia that has one, and `os_run()` falls back to a serial run with a warning if it finds one.

Value

Invisibly returns `TRUE` on success.

References

McRae, B.H. (2006). Isolation by resistance. *Evolution*, 60(8), 1551–1561. doi:10.1111/j.0014-3820.2006.tb00500.x

Landau, V.A., Shah, V.B., Anantharaman, R. & Hall, K.R. (2021). Omniscape.jl: Software to compute omnidirectional landscape connectivity. *Journal of Open Source Software*, 6(57), 2829. doi:10.21105/joss.02829

Circuitscape.jl: https://docs.circuitscape.org/Circuitscape.jl/latest/

Omniscape.jl: https://docs.circuitscape.org/Omniscape.jl/latest/

See Also

`cs_install_julia()`, `cs_pairwise()`, `os_run()`

Examples

```
cs_setup()
cs_setup(threads = 4)
cs_setup(julia_home = "/usr/local/julia/bin")
```

---

os_condition *Define a Condition for Conditional Omniscape Connectivity*

Description

Creates a condition specification for use with the `condition` argument of `os_run()`. Conditional connectivity restricts which source cells are connected to each target, based on how their values on a condition layer (e.g., climate, land cover class) compare. Conditions can compare present-day values at both sources and targets, or present-day values at sources against future values at targets (e.g., for climate-tracking connectivity).

Usage

```
os_condition(
  present,
  future = NULL,
  type = c("equal", "within"),
  lower = NULL,
  upper = NULL
)
```

Arguments

present	A single-layer terra::SpatRaster or file path giving present-day condition values. Must align with the resistance surface.
future	Optional single-layer terra::SpatRaster or file path giving future condition values. If supplied, each source's *present* value is compared against the target's *future* value. If NULL (default), present values are used for both sources and targets.
type	Character. How source and target values are compared: "equal" (connect only sources whose value equals the target's value; intended for categorical data) or "within" (connect only sources whose value falls within a range relative to the target's value; see lower and upper). If not specified, defaults to "within" when lower and upper are supplied and "equal" otherwise.
lower, upper	Numeric. Required when type = "within", and not used otherwise. A source is connected to a target only if target + lower <= source <= target + upper, or equivalently if source - target falls in [lower, upper]. For example, lower = -1 and upper = 0.5 connect sources whose values are up to 1 unit below or 0.5 units above the target's value. Either bound may be infinite (e.g., upper = Inf for a one-sided comparison).

Details

Omniscape evaluates each moving-window target against every source cell in the window, and sets the strength of any source that fails the comparison to zero for that target. The target's value is summarized over its block (see block_size in os_run()): the median for "within", and the mode for "equal". When block_size = 1 this is simply the target cell's value.

Omniscape requires a condition value at every source cell. Source cells with a missing (NA) present-day condition value are *not* filtered out; they remain sources for every target. os_run() warns if it detects this.

"equal" compares values exactly, so it is only meaningful for categorical (integer-valued) layers; a warning is issued if a SpatRaster supplied with type = "equal" contains non-integer values.

Up to two conditions can be combined by passing a list of two os_condition() objects to os_run(); a source must satisfy both to be connected.

Value

An object of class "os_condition": a list with elements present, future, type, lower, and upper.

References

Omniscape.jl conditional connectivity options: https://docs.circuitscape.org/Omniscape.jl/latest/usage/

See Also

os_run()

Examples

```
library(terra)
temp_now <- rast(nrows = 20, ncols = 20, vals = rep(1:20, each = 20))
temp_future <- temp_now + 2

# Climate-tracking: connect sources to targets whose future temperature is
# between 1 unit cooler and 3 units warmer than the source's present value
# (i.e., source - target_future within [-3, 1])
os_condition(temp_now, future = temp_future, lower = -3, upper = 1)

# Categorical: connect only cells of the same land cover class
landcover <- rast(nrows = 20, ncols = 20, vals = sample(1:3, 400, TRUE))
os_condition(landcover, type = "equal")
```

---

os_run *Run Omniscape Moving-Window Connectivity Analysis*

Description

Performs an Omniscape analysis, computing omnidirectional landscape connectivity using a moving window approach based on circuit theory.

Usage

```
os_run(
  resistance,
  radius,
  source_strength = NULL,
  block_size = 1L,
  source_threshold = 0,
  r_cutoff = Inf,
  resistance_is = "resistances",
  calc_normalized_current = TRUE,
  calc_flow_potential = TRUE,
  condition = NULL,
  condition_type = NULL,
  parallelize = FALSE,
  julia_threads = 2L,
```

```
  solver = "cg+amg",
  output_dir = NULL,
  verbose = FALSE
)
```

Arguments

resistance	A terra::SpatRaster or file path. The resistance (or conductance) surface. Higher values represent greater resistance to movement.
radius	Numeric. Moving window radius in pixels. This determines the maximum distance over which connectivity is evaluated from each source pixel.
source_strength	Optional terra::SpatRaster or file path. Source strength weights, often derived from habitat quality or suitability, where higher values indicate stronger sources of movement. If NULL (default), source strength is set to the inverse of resistance (i.e., all non-nodata pixels become sources, weighted by conductance). Use r_cutoff to exclude high-resistance cells from acting as sources in that case.
block_size	Integer. Aggregation block size for source points. Default 1 (no aggregation). A block_size of e.g. 3 coarsens the source grid into 3x3 blocks, reducing the number of solves (and thus computation time) substantially with typically negligible effects on results.
source_threshold	Numeric. Minimum source strength to include a pixel. Default 0.
r_cutoff	Numeric. Maximum resistance value for a cell to be included as a source when source_strength = NULL. Cells with resistance above this value are excluded as sources. Default Inf (no cutoff). Only relevant when source_strength is not provided.
resistance_is	Character. Whether the resistance surface represents "resistances" (default) or "conductances".
calc_normalized_current	Logical. Compute normalized current flow. Default TRUE.
calc_flow_potential	Logical. Compute flow potential. Default TRUE.
condition	Optional conditional connectivity specification, which restricts each target to sources with compatible values on one or two condition layers. Either an os_condition() object, a list of two os_condition() objects (sources must satisfy both), or a single terra::SpatRaster or file path, which is shorthand for os_condition(x, type = "equal"). Use os_condition() for range-based ("within") comparisons and for comparing present-day source values against future target values. Default NULL (no conditions).
condition_type	Deprecated. Use condition = os_condition(x, type = ...) instead. Only "equal" is still accepted here.
parallelize	Logical. Use Julia multithreading. Default FALSE. Set the number of threads with cs_setup()'s threads argument, which restarts Julia if it is already running with fewer threads. If the running Julia session cannot run Omniscape in parallel (it has only one thread, or it was started with an interactive thread,

	which Omniscape releases before 0.6.3 do not support), `os_run()` warns and runs serially.
`julia_threads`	Integer. Number of Julia threads to start if `parallelize = TRUE` and Julia is not yet running. Default 2. Has no effect once Julia is running (`os_run()` doesn't restart Julia); Omniscape then uses all of Julia's threads.
`solver`	Character. Solver to use: `"cg+amg"` (default) or `"cholmod"`.
`output_dir`	Optional character path. If provided, output files persist there, in Omniscape's `omniscape_output/` subdirectory, as ASCII grids (`.asc`). Default `NULL` uses a temporary directory. To save the results in another format, write the returned raster with `terra::writeRaster()`.
`verbose`	Logical. Print Omniscape output. Default `FALSE`.

Value

A terra::SpatRaster with the following layers (depending on options):

cumulative_current Raw cumulative current flow. Always present. Higher values indicate cells that carry more current across all moving-window iterations.

flow_potential Expected current under homogeneous resistance (if `calc_flow_potential = TRUE`). Reflects the spatial configuration of sources independently of landscape resistance.

normalized_current Cumulative current divided by flow potential (if `calc_normalized_current = TRUE`). Values greater than 1 indicate cells where connectivity is higher than expected given the source geometry; values less than 1 indicate relative barriers. This is typically the most informative layer for identifying corridors and pinch points.

References

Landau, V.A., Shah, V.B., Anantharaman, R. & Hall, K.R. (2021). Omniscape.jl: Software to compute omnidirectional landscape connectivity. *Journal of Open Source Software*, 6(57), 2829. doi:10.21105/joss.02829

Omniscape.jl: https://docs.circuitscape.org/Omniscape.jl/latest/

See Also

`os_condition()`, `cs_pairwise()`, `cs_setup()`

Examples

```
library(terra)
res <- rast(system.file("extdata/resistance.tif", package = "circuitscaper"))
result <- os_run(res, radius = 10)
plot(result)

# Conditional connectivity: only connect cells in the same zone
# (here, the left and right halves of the landscape)
zones <- (init(res, "col") > ncol(res) / 2) + 1
result_cond <- os_run(res, radius = 10, condition = zones)
plot(result_cond)
```

Index

cs_advanced, 2
cs_advanced(), *6*, *10*, *12*
cs_all_to_one, 4
cs_all_to_one(), *4*, *10*, *12*
cs_install_julia, 6
cs_install_julia(), *13, 14*
cs_julia_available, 7
cs_one_to_all, 8
cs_one_to_all(), *4*, *6*, *12*
cs_pairwise, 10
cs_pairwise(), *4*, *6*, *10*, *14*, *18*
cs_setup, 13
cs_setup(), *4*, *6–8*, *10*, *12*, *17, 18*

os_condition, 14
os_condition(), *17, 18*
os_run, 16
os_run(), *13–16*

terra::SpatRaster, *2, 3*, *5, 6*, *8–12*, *15*, *17, 18*
terra::writeRaster(), *18*