Package: dci (via r-universe)

October 23, 2026

Title Calculate the Dendritic Connectivity Index in River Networks

Version 1.0.3

Maintainer Alex Arkilanian <a.arkilanian@gmail.com>

Description Calculate and analyze ecological connectivity across the watercourse of river networks using the Dendritic Connectivity Index.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/aarkilanian/dci

BugReports https://github.com/aarkilanian/dci/issues

Suggests furrr, future, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports rlang, dplyr, magrittr, units, sf, tidygraph, sfnetworks, igraph, tidyselect

Depends R (>= 3.50)

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libglpk-dev libicu-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://aarkilanian.r-universe.dev

Date/Publication 2026-09-23 22:00:10 UTC

RemoteUrl https://github.com/aarkilanian/dci

RemoteRef HEAD

RemoteSha 95c10b62b5f7b0653ce85f02f1caa2ad30a4496a

Contents

calculate_dci 2
enforce_dendritic 3
export_dci 4
import_points 5
import_rivers 6
river_net 7
yamaska_barriers 8
yamaska_net 8
yamaska_outlet 9
yamaska_rivers 9

Index **11**

`calculate_dci` *Calculate DCI for a* `river_net` *Object*

Description

Calculates the potamodromous and diadromous forms of the Dendritic Connectivity Index (DCI) for a `river_net` object.

Usage

```
calculate_dci(
  net,
  form,
  pass = NULL,
  weight = NULL,
  threshold = NULL,
  parallel = FALSE,
  quiet = FALSE
)
```

Arguments

`net`	A `river_net` object.
`form`	A string specifying the DCI form to calculate. Options are: "pot" for potamodromous or "dia" for diadromous.
`pass`	The name of a column in the nodes table of `net` containing numeric passability values. If `NULL`, all barriers are assumed to have 0 passability.
`weight`	The name of a column in the edges table of `net` containing numeric weights for river lengths. If `NULL`, DCI is calculated using river length only.
`threshold`	Optional numeric value specifying a dispersal limit in map units. If `NULL` (default), no limit is applied.

parallel	Logical. If FALSE, the default, all operations are performed in series. If TRUE parallel operation is performed using furrr::future_pmap(). Specify the number of workers and strategy using future::plan().
quiet	Logical. If FALSE, prints the global DCI and a plot of river segments to the console. Defaults to TRUE.

Details

Passability values are probabilities between 0 and 1, where 0 indicates a fully impassable barrier and 1 indicates full passability. If values in the specified passability column fall outside this range, they will be normalized.

Weighting values should also be probabilities between 0 and 1. River segments with weights of 0 or NA will be excluded from the DCI calculation.

Upon successful calculation, the global DCI value for the river network will be printed to the console unless quiet = TRUE.

Value

An sf object of the river network with new columns specifying segmental DCI values and relative DCI values. These are each segment's contribution to the global DCI score which is printed. The relative values are simply those values normalized.

Examples

```
# For the potamodromous DCI
res <- calculate_dci(net = yamaska_net, form = "pot", pass = "pass_1",
weight = "weight")
```

---

enforce_dendritic *Enforce dendritic river topology*

Description

Identifies and optionally corrects features in a river network that violate a strictly dendritic topology.

Usage

```
enforce_dendritic(rivers, correct = TRUE, quiet = FALSE, max_iter = 10)
```

Arguments

rivers	A rivers object returned by import_rivers().
correct	Logical. If FALSE (default), no changes are made and topological issues are identified only. If TRUE, issues are automatically corrected.
quiet	Logical. If FALSE, the function prints a summary including the global DCI and a map of segments. Defaults to TRUE.

`max_iter` An integer indicating the maximum number of correction iterations to run. As some topological errors are corrected new ones can can arise requiring multiple passes. In some cases, an automated correction choice can lead to a recursive correction that eliminates most rivers. In this case, some manual corrections may help avoid this.

Details

In a dendritic network, two upstream rivers converge into a single downstream river at each confluence. This function can enforce this dendritic topology in a river network by detecting (and optionally correcting) two types of topological errors: (1) divergences, where a single river splits into multiple downstream branches (commonly forming loops or braided channels), and (2) complex confluences, where more than two upstream rivers meet at a single point.

If errors are being corrected manually, rerun this function again until no errors remain as correcting divergences can lead to other topological errors that need to be corrected

Value

If `correct = FALSE`, returns a `sf` object with the columns `"divergent"` and `"complex"` indicating topological errors. These columns contain integer identifiers indicating which features are part of the same divergent or complex structure. If `correct = TRUE`, returns a `rivers` object with the topological issues corrected.

Examples

```
# Import rivers
rivers_in <- import_rivers(yamaska_rivers, quiet = TRUE)

# Correct errors automatically
rivers_cor <- enforce_dendritic(rivers_in, correct = TRUE)

# Return highlighted topological errors for manual correction
rivers_uncor <- enforce_dendritic(rivers_in, correct = FALSE)

# For large river networks it may be better to specify a smaller number of
# correction sweeps.
rivers_cor <- enforce_dendritic(rivers_in, correct = TRUE, max_iter = 3)
```

---

export_dci *Export DCI Results to Spatial Format*

Description

Exports the output of `calculate_dci()` as a spatial object with DCI values joined to the relevant features in the river network.

Usage

```
export_dci(net, results, type = "rivers", relative = FALSE, quiet = TRUE)
```

Arguments

net	A river_net object.
results	A dci_results object, or a list of such objects, as returned by calculate_dci().
type	A character string specifying which component of the river network the results should be exported for. Valid options are "rivers" (default), or "bars".
relative	A logical value indicating whether relative DCI values should be returned in addition to raw values. Defaults to FALSE.
quiet	Logical. If FALSE, prints the global DCI and a plot of river segments to the console. Defaults to TRUE.

Value

An sf object containing the corresponding DCI results joined to the selected network component. If multiple results are supplied, result columns are appended by a number corresponding to the index of the associated results.

Examples

```
res_pot <- calculate_dci(net = yamaska_net, form = "pot", pass = "pass_1",
quiet = TRUE)
res_dia <- calculate_dci(net = yamaska_net, form = "dia", pass = "pass_1",
quiet = TRUE)

# Export segment-level potamodromous DCI results to rivers
riv_results <- export_dci(net = yamaska_net, results = res_pot,
type = "rivers")

# Can also be run quietly to keep from plotting results
riv_results <- export_dci(net = yamaska_net, results = res_pot,
type = "rivers", quiet = TRUE)

# Results can also be exported to barrier points
bar_results <- export_dci(net = yamaska_net, results = res_pot,
type = "bars")

# If multiple results are calculated these can be combined together
all_res <- export_dci(net = yamaska_net, results = list(res_pot, res_dia),
type = "rivers")
```

---

import_points *Prepare point data for connectivity analyses*

Description

Reads and prepares geospatial point data for use with river_net().

Usage

```
import_points(pts, type)
```

Arguments

pts	A character string specifying the path to a shapefile of points, or an sf object containing point features.
type	A character string indicating the type of points. Must be one of: "bars" for barriers or "out" for the outlet.

Value

An object of class barriers or outlet depending on type, prepared for use with river_net().

Examples

```
import_points(yamaska_barriers, type = "bars")
import_points(yamaska_outlet, type = "out")
```

---

import_rivers *Prepare rivers for connectivity analyses*

Description

Reads and prepares geospatial river line data for use in river_net(). Only the largest fully connected component of the network is retained; river lines that are part of disconnected secondary networks are discarded.

Usage

```
import_rivers(rivers, quiet = FALSE)
```

Arguments

rivers	A character string specifying the path to a shapefile of river lines, or an sf object representing river geometries.
quiet	Logical. If FALSE (default), plots the imported river lines in black over the original lines in red so that removed rivers are highlighted.

Value

An object of class rivers, suitable for use with enforce_dendritic() or as input to river_net().

Examples

```
rivers_in <- import_rivers(yamaska_rivers)

# This can also be done quietly to omit plotting river lines after importing
rivers_in <- import_rivers(yamaska_rivers, quiet = TRUE)
```

river_net *Create a* river_net *Object*

Description

Constructs a river_net object, a geospatial network structure built on top of the sfnetworks::sfnetwork() class. This object integrates river lines, barriers and outlets allowing for connectivity analyses with calculate_dci() or other network tools.

Usage

```
river_net(
  rivers,
  barriers,
  outlet,
  check = TRUE,
  tolerance = NULL,
  max_iter = 10
)
```

Arguments

rivers	A rivers object returned by import_rivers().
barriers	A barriers object returned by import_points() with type = "bars".
outlet	An outlet object returned by import_points() with type = "out".
check	Logical. If TRUE (default), dendritic topology is enforced using enforce_dendritic().
tolerance	A numeric value specifying the snapping distance (in map units) to align points to the river network. Defaults to NULL, meaning no snapping.
max_iter	An integer indicating the maximum number of correction iterations to run. As some topological errors are corrected new ones can can arise requiring multiple passes. In some cases, an automated correction choice can lead to a recursive correction that eliminates most rivers. In this case, some manual corrections may help avoid this.

Value

An object of class river_net representing the river network formed from the provided spatial inputs.

Examples

```
riv_in <- import_rivers(yamaska_rivers, quiet = TRUE)
bar_in <- import_points(yamaska_barriers, type = "bars")
out_in <- import_points(yamaska_outlet, type = "out")

# For large river networks it may be better to specify a smaller number of
# correction sweeps.
yam_net <- river_net(rivers = riv_in, barriers = bar_in,
outlet = out_in, max_iter = 3)
```

yamaska_barriers *Barrier point features for the Yamaska watershed*

Description

An sf object of the POINT geometries that make up imagined barriers on the Yamaska watershed of Southern Québec. Features are projected to CRS:32198.

Usage

```
yamaska_barriers
```

Format

An sf object with 620 LINESTRING features:

pass_1 the passability of nodes in the network from 0 (impassable) to 1

pass_2 the binary passability of nodes in the network, 0 if a barrier and 1 otherwise

geometry the geometry list column of river POINT features

yamaska_net *River network for the Yamaska watershed*

Description

A spatial river_net object extending the tidygraph representation of spatial graphs. Nodes represent confluences, river endpoints, barriers, and the outlet. Edges represent stream reaches. Spatial features are projected to EPSG:32198.

Usage

```
yamaska_net
```

Format

A river_net object with:

Nodes **geometry** the geometry list column of the node point features
- **pass_1** the passability of nodes in the network from 0 (impassable) to 1
- **pass_2** the binary passability of nodes in the network, 0 if a barrier and 1 otherwise
- **type** the type of the node: topological, barrier, or outlet

Edges **from** the row index of the origin node of the edge feature
- **to** the row index of the destination node of the edge feature
- **qual** a simulated weighting based on habitat quality of features
- **riv_length** length of edge features in meters
- **rivID** unique river feature integer ID
- **geometry** the geometry list column of edge line features

Source

https://www.donneesquebec.ca/recherche/dataset/grhq

https://www.donneesquebec.ca/recherche/dataset/structure

---

yamaska_outlet *Outlet point feature for the Yamaska watershed*

Description

An sf object of the POINT geometry of the outlet of the Yamaska watershed of Southern Québec. The feature is projected to CRS:32198.

Usage

```
yamaska_outlet
```

Format

An sf object with 1 POINT feature:

geometry the geometry list column of the outlet POINT feature

Source

https://www.donneesquebec.ca/recherche/dataset/grhq

https://www.donneesquebec.ca/recherche/dataset/structure

---

yamaska_rivers *River line features for the Yamaska watershed*

Description

An sf object of the LINESTRING geometries that make up the rivers of the Yamaska watershed of Southern Québec. The rivers make up the edges of the yamaska_net object. Features are projected to CRS:32198.

Usage

```
yamaska_rivers
```

Format

An sf object with 620 features:

qual a simulated weighting based on habitat quality of features

riv_length length of edge features in meters

geometry the geometry list column of edge line features

Source

https://www.donneesquebec.ca/recherche/dataset/grhq

https://www.donneesquebec.ca/recherche/dataset/structure

Index

∗ **datasets**
yamaska_barriers, 8
yamaska_net, 8
yamaska_outlet, 9
yamaska_rivers, 9

calculate_dci, 2
calculate_dci(), *5*, *7*

enforce_dendritic, 3
enforce_dendritic(), *6, 7*
export_dci, 4

furrr::future_pmap(), *3*
future::plan(), *3*

import_points, 5
import_points(), *7*
import_rivers, 6
import_rivers(), *3*, *7*

river_net, *2*, *5*, 7, *7*
river_net(), *5, 6*

sfnetworks::sfnetwork(), *7*

yamaska_barriers, 8
yamaska_net, 8
yamaska_outlet, 9
yamaska_rivers, 9