Package 'accessibility'

October 12, 2022

Type Package

Title Transport Accessibility Measures

Version 1.0.1

Description A set of fast and convenient functions to calculate multiple transport accessibility measures. Given a pre-computed travel cost matrix and a land use dataset (containing the location of jobs, healthcare and population, for example), the package allows one to calculate active and passive accessibility levels using multiple accessibility measures, such as: cumulative opportunities (using either travel cost cutoffs or intervals), minimum travel cost to closest N number of activities, gravity-based (with different decay functions) and different floating catchment area methods.

License MIT + file LICENSE

URL https://github.com/ipeaGIT/accessibility

BugReports https://github.com/ipeaGIT/accessibility/issues

Depends R (>= 3.5.0)

Imports checkmate, data.table, Rdpack (>= 0.7), utils

Suggests covr, ggplot2, knitr, rmarkdown, sf, stats, testthat

VignetteBuilder knitr

RdMacros Rdpack

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.2.1

Author Rafael H. M. Pereira [aut] (<https://orcid.org/0000-0003-2125-7465>), Daniel Herszenhut [aut, cre] (<https://orcid.org/0000-0001-8066-1105>), Ipea - Institute for Applied Economic Research [cph, fnd]

Maintainer Daniel Herszenhut <dhersz@gmail.com>

Repository CRAN

Date/Publication 2022-10-06 16:40:02 UTC

R topics documented:

`cost_to_closest` *Minimum travel cost to closest N number of opportunities*

Description

Calculates the minimum travel cost to the closest N number of opportunities.

This function is generic over any kind of numeric travel cost, such as distance, time and money.

Usage

```
cost_to_closest(
  travel_matrix,
  land_use_data,
  opportunity,
  travel_cost,
  n = 1,
  group_by = character(0),
  active = TRUE,
  fill_missing_ids = TRUE
)
```

Arguments

`travel_matrix`	A data frame. The travel matrix describing the costs (i.e. travel time, distance, monetary cost, etc.) between the origins and destinations in the study area. Must contain the columns `from_id`, `to_id` and any others specified in `travel_cost_col`.
`land_use_data`	A data frame. The distribution of opportunities within the study area cells. Must contain the columns `id` and any others specified in `opportunity_col`.
`opportunity`	A string. The name of the column in `land_use_data` with the number of opportunities/resources/services to be considered when calculating accessibility levels.

`travel_cost` A string. The name of the column in `travel_matrix` with the travel cost between origins and destinations. Defaults to `"travel_time"`.

`n` A `numeric`. A number indicating the minimum number of opportunities that should be considered. Defaults to 1.

`group_by` A `character` vector. When not `character(0)` (the default), indicates the `travel_matrix` columns that should be used to group the accessibility estimates by. For example, if `travel_matrix` includes a departure time column, that specifies the departure time of each entry in the data frame, passing `"departure_time"` to this parameter results in accessibility estimates grouped by origin and by departure time.

`active` A logical. Whether to calculate active accessibility (the of opportunities that can be reached from a given origin, the default) or passive accessibility (by how many people each destination can be reached).

`fill_missing_ids`
A `logical`. Calculating minimum travel cost to closest N number of opportunities may result in missing ids in the output if they cannot reach the specified amount of opportunities across all destinations they can reach. For example, estimating the minimum travel time that an origin that can only reach 4 opportunities takes to reach 5 opportunities resulting in such origin not being included in the output. When `TRUE` (the default), the function identifies which ids would be left out from the output and fill their respective minimum travel costs with `Inf`, which incurs in a performance penalty.

Value

A data frame containing the accessibility estimates for each origin/destination (depending if `active` is `TRUE` or `FALSE`) in the travel matrix.

Examples

```
data_dir <- system.file("extdata", package = "accessibility")
travel_matrix <- readRDS(file.path(data_dir, "travel_matrix.rds"))
land_use_data <- readRDS(file.path(data_dir, "land_use_data.rds"))

df <- cost_to_closest(
  travel_matrix,
  land_use_data,
  n = 1,
  opportunity = "schools",
  travel_cost = "travel_time"
)
head(df)

df <- cost_to_closest(
  travel_matrix,
  land_use_data,
  n = 2,
  opportunity = "schools",
  travel_cost = "travel_time"
```

```
)
head(df)
```

`cumulative_cutoff` *Cumulative access based on a travel cost cutoff*

Description

Calculates the number of opportunities accessible under a given specified travel cost cutoff.

This function is generic over any kind of numeric travel cost, such as distance, time and money.

Usage

```
cumulative_cutoff(
  travel_matrix,
  land_use_data,
  opportunity,
  travel_cost,
  cutoff,
  group_by = character(0),
  active = TRUE,
  fill_missing_ids = TRUE
)
```

Arguments

`travel_matrix`	A data frame. The travel matrix describing the costs (i.e. travel time, distance, monetary cost, etc.) between the origins and destinations in the study area. Must contain the columns `from_id`, `to_id` and any others specified in `travel_cost_col`.
`land_use_data`	A data frame. The distribution of opportunities within the study area cells. Must contain the columns `id` and any others specified in `opportunity_col`.
`opportunity`	A string. The name of the column in `land_use_data` with the number of opportunities/resources/services to be considered when calculating accessibility levels.
`travel_cost`	A string. The name of the column in `travel_matrix` with the travel cost between origins and destinations. Defaults to `"travel_time"`.
`cutoff`	A `numeric`. A number indicating the travel cost cutoff.
`group_by`	A `character` vector. When not `character(0)` (the default), indicates the `travel_matrix` columns that should be used to group the accessibility estimates by. For example, if `travel_matrix` includes a departure time column, that specifies the departure time of each entry in the data frame, passing `"departure_time"` to this parameter results in accessibility estimates grouped by origin and by departure time.

`active` A logical. Whether to calculate active accessibility (the of opportunities that can be reached from a given origin, the default) or passive accessibility (by how many people each destination can be reached).

`fill_missing_ids` A `logical`. Calculating cumulative accessibility may result in missing ids if the they cannot reach any of the destinations within the specified travel cost cutoff. For example, using a travel time cutoff of 20 minutes, when estimating the accessibility of origin `A` that can only reach destinations with more than 40 minutes results in id `A` not being included in the output. When `TRUE` (the default), the function identifies which origins would be left out and fills their respective accessibility values with 0, which incurs in a performance penalty.

Value

A data frame containing the accessibility estimates for each origin/destination (depending if `active` is `TRUE` or `FALSE`) in the travel matrix.

See Also

Other cumulative access: `cumulative_interval()`

Examples

```
data_dir <- system.file("extdata", package = "accessibility")
travel_matrix <- readRDS(file.path(data_dir, "travel_matrix.rds"))
land_use_data <- readRDS(file.path(data_dir, "land_use_data.rds"))

# active accessibility: number of schools accessible from each origin
df <- cumulative_cutoff(
  travel_matrix = travel_matrix,
  land_use_data = land_use_data,
  cutoff = 30,
  opportunity = "schools",
  travel_cost = "travel_time"
)
head(df)

# passive accessibility: number of people that can reach each destination
df <- cumulative_cutoff(
  travel_matrix = travel_matrix,
  land_use_data = land_use_data,
  cutoff = 30,
  opportunity = "population",
  travel_cost = "travel_time",
  active = FALSE
)
head(df)
```

`cumulative_interval` *Cumulative access based on maximum travel time interval*

Description

Calculates the average or median number of opportunities that can be reached considering multiple maximum travel cost thresholds within a given travel cost interval specified by the user. The time interval cumulative accessibility measures was originally proposed by Tomasiello et al. (2022).

This function is generic over any kind of numeric travel cost, such as distance, time and money.

Usage

```
cumulative_interval(
  travel_matrix,
  land_use_data,
  opportunity,
  travel_cost,
  interval,
  interval_increment = 1,
  summary_function = stats::median,
  group_by = character(0),
  active = TRUE
)
```

Arguments

`travel_matrix`	A data frame. The travel matrix describing the costs (i.e. travel time, distance, monetary cost, etc.) between the origins and destinations in the study area. Must contain the columns `from_id`, `to_id` and any others specified in `travel_cost_col`.
`land_use_data`	A data frame. The distribution of opportunities within the study area cells. Must contain the columns `id` and any others specified in `opportunity_col`.
`opportunity`	A string. The name of the column in `land_use_data` with the number of opportunities/resources/services to be considered when calculating accessibility levels.
`travel_cost`	A string. The name of the column in `travel_matrix` with the travel cost between origins and destinations. Defaults to `"travel_time"`.
`interval`	A `numeric` vector of length 2. Indicates the start and end points of the interval of travel cost thresholds to be used. The first entry must be lower than the second.
`interval_increment`	A `numeric`. How many travel cost units separate the cutoffs used to calculate the accessibility estimates which will be used to calculate the summary estimate within the specified interval. Should be thought as the resolution of the distribution of travel costs within the interval. Defaults to 1.

summary_function
: A function. This function is used to summarize a distribution of accessibility estimates within a travel cost interval as a single value. Can be any function that takes an arbitrary number of numeric values as as input and returns a single number as output. Defaults to `stats::median()`.

group_by
: A `character` vector. When not `character(0)` (the default), indicates the `travel_matrix` columns that should be used to group the accessibility estimates by. For example, if `travel_matrix` includes a departure time column, that specifies the departure time of each entry in the data frame, passing `"departure_time"` to this parameter results in accessibility estimates grouped by origin and by departure time.

active
: A logical. Whether to calculate active accessibility (the of opportunities that can be reached from a given origin, the default) or passive accessibility (by how many people each destination can be reached).

Value

A data frame containing the accessibility estimates for each origin/destination (depending if `active` is `TRUE` or `FALSE`) in the travel matrix.

References

Tomasiello DB, Santos DHM, Oliveira JLA, Braga CKV, Pereira RHM (2022). "A time interval metric for cumulative opportunity accessibility." *SocArXiv*. doi:10.31235/osf.io/ux5ah, https://osf.io/preprints/socarxiv/ux5ah/.

See Also

Other cumulative access: `cumulative_cutoff()`

Examples

```
data_dir <- system.file("extdata", package = "accessibility")
travel_matrix <- readRDS(file.path(data_dir, "travel_matrix.rds"))
land_use_data <- readRDS(file.path(data_dir, "land_use_data.rds"))

df <- cumulative_interval(
  travel_matrix = travel_matrix,
  land_use_data = land_use_data,
  interval = c(20, 30),
  opportunity = "schools",
  travel_cost = "travel_time"
)
head(df)

df <- cumulative_interval(
  travel_matrix = travel_matrix,
  land_use_data = land_use_data,
  interval = c(40, 80),
  opportunity = "jobs",
```

```
  travel_cost = "travel_time"
)
head(df)
```

`decay_binary` *Binary (a.k.a. step) decay function*

Description

Returns a binary weighting function (frequently used to calculate cumulative opportunities measures) to be used inside accessibility calculating functions.

This function is generic over any kind of numeric travel cost, such as distance, time and money.

Usage

```
decay_binary(cutoff)
```

Arguments

`cutoff`	A `numeric`. A number indicating the travel cost cutoff.

Value

A `function` that takes a generic travel cost vector (`numeric`) as an input and returns a vector of weights (`numeric`).

See Also

Other decay functions: `decay_exponential()`, `decay_linear()`, `decay_power()`, `decay_stepped()`

Examples

```
weighting_function <- decay_binary(cutoff = 30)

weighting_function(20)

weighting_function(35)
```

decay_exponential *Negative exponential decay function*

Description

Returns a negative exponential weighting function to be used inside accessibility calculating functions.

This function is generic over any kind of numeric travel cost, such as distance, time and money.

Usage

```
decay_exponential(decay_value)
```

Arguments

decay_value	A numeric. The calibration parameter that, when multiplied by the travel cost, is used as the exponent of e in the negative exponential function.

Value

A function that takes a generic travel cost vector (numeric) as an input and returns a vector of weights (numeric).

See Also

Other decay functions: decay_binary(), decay_linear(), decay_power(), decay_stepped()

Examples

```
weighting_function <- decay_exponential(decay_value = 0.1)

weighting_function(20)

weighting_function(35)
```

decay_linear *Linear decay function*

Description

Returns a linear weighting function to be used inside accessibility calculating functions.

This function is generic over any kind of numeric travel cost, such as distance, time and money.

Usage

```
decay_linear(cutoff)
```

Arguments

cutoff	A numeric. A number indicating the travel cost cutoff until which the weighting factor decays linearly. From this point onward the weight is equal to 0.

Value

A function that takes a generic travel cost vector (numeric) as an input and returns a vector of weights (numeric).

See Also

Other decay functions: decay_binary(), decay_exponential(), decay_power(), decay_stepped()

Examples

```
weighting_function <- decay_linear(cutoff = 30)

weighting_function(20)

weighting_function(35)
```

---

decay_power *Inverse power decay function*

Description

Returns an inverse power weighting function to be used inside accessibility calculating functions.

This function is generic over any kind of numeric travel cost, such as distance, time and money.

Usage

```
decay_power(decay_value)
```

Arguments

decay_value	A numeric. The calibration parameter to be used as the exponent in the inverse power function.

Value

A function that takes a generic travel cost vector (numeric) as an input and returns a vector of weights (numeric).

See Also

Other decay functions: decay_binary(), decay_exponential(), decay_linear(), decay_stepped()

Examples

```
weighting_function <- decay_power(decay_value = 0.1)

weighting_function(20)

weighting_function(35)
```

`decay_stepped` *Stepped decay function*

Description

Returns a stepped weighting function to be used inside accessibility calculating functions.

This function is generic over any kind of numeric travel cost, such as distance, time and money.

Usage

```
decay_stepped(steps, weights)
```

Arguments

`steps`	A `numeric` vector. The travel cost steps, in ascending order. Please do not include travel cost 0 as a step: this is already handled by the function.
`weights`	A `numeric` vector with same length as `steps`. The values, between 0 and 1, that the function assumes at each step. Please do not include weight 1 as the first value: this is already handled by the function. The function considers the steps' intervals "open on the right", meaning that the function assumes the step value at the actual step, not after it. Please see the illustrative examples for effects of this assumption on the results.

Value

A `function` that takes a generic travel cost vector (`numeric`) as an input and returns a vector of weights (`numeric`).

See Also

Other decay functions: `decay_binary()`, `decay_exponential()`, `decay_linear()`, `decay_power()`

Examples

```
weighting_function <- decay_stepped(
  c(10, 20, 30, 40),
  weights = c(0.75, 0.5, 0.25, 0)
)

weighting_function(c(5, 25, 35, 45))
```

```
# intervals are open on the right, so the values change exactly at each step
weighting_function(c(0, 10, 20, 30, 40))
```

`floating_catchment_area` *Floating catchment area accessibility*

Description

Calculates accessibility accounting for the competition of resources using a measure from the floating catchment area (FCA) family. Please see the details for the available FCA measures.

This function is generic over any kind of numeric travel cost, such as distance, time and money.

Usage

```
floating_catchment_area(
  travel_matrix,
  land_use_data,
  opportunity,
  travel_cost,
  demand,
  method,
  decay_function,
  group_by = character(0),
  fill_missing_ids = TRUE
)
```

Arguments

`travel_matrix`	A data frame. The travel matrix describing the costs (i.e. travel time, distance, monetary cost, etc.) between the origins and destinations in the study area. Must contain the columns `from_id`, `to_id` and any others specified in `travel_cost_col`.
`land_use_data`	A data frame. The distribution of opportunities within the study area cells. Must contain the columns `id` and any others specified in `opportunity_col`.
`opportunity`	A string. The name of the column in `land_use_data` with the number of opportunities/resources/services to be considered when calculating accessibility levels.
`travel_cost`	A string. The name of the column in `travel_matrix` with the travel cost between origins and destinations. Defaults to `"travel_time"`.
`demand`	A string. The name of the column in `land_use_data` with the number of people in each origin that will be considered potential competitors. Defaults to `"population"`.

`method` A string. Which floating catchment area measure to use. Current available options are `"2sfca"` and `"bfca"`. More info in the details.

`decay_function` A `fuction` that converts travel cost into an impedance factor used to weight opportunities. This function should take a `numeric` vector and also return a `numeric` vector as output, with the same length as the input. For convenience, the package currently includes the following functions: `decay_binary()`, `decay_exponential()`, `decay_exponential()` and `decay_power()`. See the documentation of each decay function for more details.

`group_by` A `character` vector. When not `character(0)` (the default), indicates the `travel_matrix` columns that should be used to group the accessibility estimates by. For example, if `travel_matrix` includes a departure time column, that specifies the departure time of each entry in the data frame, passing `"departure_time"` to this parameter results in accessibility estimates grouped by origin and by departure time.

`fill_missing_ids` A `logical`. When calculating grouped accessibility estimates (i.e. when `by_col` is not `NULL`), some combinations of groups and origins may be missing. For example, if a single trip can depart from origin `A` at 7:15am and reach destination `B` within 55 minutes, but no trips departing from `A` at 7:30am can be completed at all, this second combination will not be included in the output. When `TRUE` (the default), the function identifies which combinations would be left out and fills their respective accessibility values with 0, which incurs in a performance penalty.

Value

A data frame containing the accessibility estimates for each origin/destination (depending if `active` is `TRUE` or `FALSE`) in the travel matrix.

Details

The package currently includes two built-in FCA measures:

- 2SFCA - the 2-Step Floating Catchment Area measure was the first accessibility metric in the FCA family. It was originally proposed by Luo and Wang (2003).
- BFCA - the Balanced Floating Catchment Area measure calculates accessibility accounting for competition effects while simultaneously correcting for issues of inflation of demand and service levels that are present in other FCA measures. It was originally proposed by Paez et al. (2019) and named in Pereira et al. (2021).

References

Luo W, Wang F (2003). "Measures of Spatial Accessibility to Health Care in a GIS Environment: Synthesis and a Case Study in the Chicago Region." *Environment and Planning B: Planning and Design*, **30**(6), 865–884. ISSN 0265-8135, 1472-3417, doi:10.1068/b29120.

Paez A, Higgins CD, Vivona SF (2019). "Demand and Level of Service Inflation in Floating Catchment Area (FCA) Methods." *PLOS ONE*, **14**(6), e0218773. ISSN 1932-6203, doi:10.1371/journal.pone.0218773.

Pereira RH, Braga CKV, Servo LM, Serra B, Amaral P, Gouveia N, Paez A (2021). "Geographic Access to COVID-19 Healthcare in Brazil Using a Balanced Float Catchment Area Approach." *Social Science & Medicine*, **273**, 113773. ISSN 02779536, doi:10.1016/j.socscimed.2021.113773.

Examples

```
data_dir <- system.file("extdata", package = "accessibility")
travel_matrix <- readRDS(file.path(data_dir, "travel_matrix.rds"))
land_use_data <- readRDS(file.path(data_dir, "land_use_data.rds"))

# 2SFCA with a step decay function
df <- floating_catchment_area(
  travel_matrix,
  land_use_data,
  method = "2sfca",
  decay_function = decay_binary(cutoff = 50),
  opportunity = "jobs",
  travel_cost = "travel_time",
  demand = "population"
)
head(df)


# BFCA with an exponential decay function
df <- floating_catchment_area(
  travel_matrix,
  land_use_data,
  method = "bfca",
  decay_function = decay_exponential(decay_value = 0.5),
  opportunity = "jobs",
  travel_cost = "travel_time",
  demand = "population"
)
head(df)
```

---

gravity *Gravity-based accessibility measures*

Description

Calculates gravity-based accessibility using a decay function specified by the user.

This function is generic over any kind of numeric travel cost, such as distance, time and money.

Usage

```
gravity(
  travel_matrix,
```

```
  land_use_data,
  opportunity,
  travel_cost,
  decay_function,
  group_by = character(0),
  active = TRUE,
  fill_missing_ids = TRUE
)
```

Arguments

travel_matrix	A data frame. The travel matrix describing the costs (i.e. travel time, distance, monetary cost, etc.) between the origins and destinations in the study area. Must contain the columns from_id, to_id and any others specified in travel_cost_col.
land_use_data	A data frame. The distribution of opportunities within the study area cells. Must contain the columns id and any others specified in opportunity_col.
opportunity	A string. The name of the column in land_use_data with the number of opportunities/resources/services to be considered when calculating accessibility levels.
travel_cost	A string. The name of the column in travel_matrix with the travel cost between origins and destinations. Defaults to "travel_time".
decay_function	A fuction that converts travel cost into an impedance factor used to weight opportunities. This function should take a numeric vector and also return a numeric vector as output, with the same length as the input. For convenience, the package currently includes the following functions: decay_binary(), decay_exponential(), decay_exponential() and decay_power(). See the documentation of each decay function for more details.
group_by	A character vector. When not character(0) (the default), indicates the travel_matrix columns that should be used to group the accessibility estimates by. For example, if travel_matrix includes a departure time column, that specifies the departure time of each entry in the data frame, passing "departure_time" to this parameter results in accessibility estimates grouped by origin and by departure time.
active	A logical. Whether to calculate active accessibility (the of opportunities that can be reached from a given origin, the default) or passive accessibility (by how many people each destination can be reached).
fill_missing_ids	A logical. When calculating grouped accessibility estimates (i.e. when by_col is not NULL), some combinations of groups and origins may be missing. For example, if a single trip can depart from origin A at 7:15am and reach destination B within 55 minutes, but no trips departing from A at 7:30am can be completed at all, this second combination will not be included in the output. When TRUE (the default), the function identifies which combinations would be left out and fills their respective accessibility values with 0, which incurs in a performance penalty.

Value

A data frame containing the accessibility estimates for each origin/destination (depending if `active` is `TRUE` or `FALSE`) in the travel matrix.

Examples

```
data_dir <- system.file("extdata", package = "accessibility")
travel_matrix <- readRDS(file.path(data_dir, "travel_matrix.rds"))
land_use_data <- readRDS(file.path(data_dir, "land_use_data.rds"))

df_linear <- gravity(
  travel_matrix,
  land_use_data,
  decay_function = decay_linear(cutoff = 50),
  opportunity = "schools",
  travel_cost = "travel_time"
)
head(df_linear)

df_exp <- gravity(
  travel_matrix,
  land_use_data,
  decay_function = decay_exponential(decay_value = 0.5),
  opportunity = "schools",
  travel_cost = "travel_time"
)
head(df_exp)
```

Index