Package 'cppSim'

September 3, 2025

Title Fast and Memory Efficient Spatial Interaction Models

Version 0.2

Author Ivann Schlosser [aut, cre] (ORCID:
<https://orcid.org/0009-0004-4099-3198>)

Maintainer Ivann Schlosser <ivann.schlosser.19@ucl.ac.uk>

Description Building on top of the 'RcppArmadillo' linear
algebra functionalities to do fast spatial interaction models in the
context of urban analytics, geography, transport modelling. It uses
the Newton root search algorithm to determine the optimal cost
exponent and can run country level models with thousands of origins
and destinations. It aims at implementing an easy approach based on
matrices, that can originate from various routing and processing steps earlier in an workflow.
Currently, the simplest form of production, destination and
doubly constrained models are implemented.
Schlosser et al. (2023) <doi:10.48550/arXiv.2309.02112>.

Date 2025-08-29

License MIT + file LICENSE

URL https://ischlo.github.io/cppSim/, https://github.com/ischlo/cppSim

Depends R (>= 3.6)

Imports Rcpp

Suggests data.table, foreach, knitr, rlist, rmarkdown, cli, sf,
testthat (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

LazyDataCompression bzip2

RoxygenNote 7.3.1

SystemRequirements Quarto command line tools
(https://github.com/quarto-dev/quarto-cli).

NeedsCompilation yes

Repository CRAN

Date/Publication 2025-09-03 21:00:14 UTC

Contents

`calibration_cpp` *Calibrating the balancing factors*

Description

Function to calibrate the A and B coefficients of the gravity model through

---

`cpp_found_openmp` *Function to check availability of OPENMP to run in parallel. If openmp is found, this function returns TRUE*

Description

Function to check availability of OPENMP to run in parallel. If openmp is found, this function returns TRUE

Usage

```
cpp_found_openmp()
```

distance_test *distance_test*

Description

distance_test

Usage

```
distance_test
```

Format

`distance_test`:
A 983x983 matrix of distances between MSOAs in London. Computed using the London road network from OpenStreetMap and the cppRouting package.

Source

Ivann Schlosser, 2022

flows_london *flows_london*

Description

flows_london

Usage

```
flows_london
```

Format

`flows_london`:
A data.table with flows information

Source

ONS, Office for National Statistics, 2011

flows_test *flows_test*

Description

flows_test

Usage

```
flows_test
```

Format

`flows_test`:
A matrix of size 983x983 containing flows of users using walking or cycling as the main method of commute.

Source

UK Census, 2011

london_msoa *london_msoa*

Description

london_msoa

Usage

```
london_msoa
```

Format

`london_msoa`:
A data.table with London MSOA, their centroids and geometries

Source

ONS, Office for National Statistics, 2011

run_model *Running doubly constrained model*

Description

This function is the C++ implementation of run_model, it will run a doubly constrained model

Usage

```
run_model(flows, distance, beta = 0.25)
```

Arguments

flows	A integer matrix of Origin-Destination flows.
distance	a distance matrix between origins and destinations, provide distance in km.
beta	Exponent to use when calculating the cost function.

Value

A list containing an integer matrix with predicted values.

Examples

```
data(flows_test)
data(distance_test)

model_test <- run_model(flows_test,distance_test)
```

run_model_cpp *Run model*

Description

This function is the C++ implementation of run_model, it will run a model

Arguments

flows	A integer matrix of Origin-Destination flows.
distance	a distance matrix between origins and destinations.
beta	Exponent to use when calculating the cost function.

run_model_single *Running a singly constrained model*

Description

This function is the C++ implementation of run_model, it will run a singly constrained model there must be a match in the dimensions, when running a production constrained model, any(dim(distance) == length(flows)) must be TRUE if no values for weight are provided, a vector with ones is used

Usage

```
run_model_single(flows, distance, weight = NULL, beta = 0.25)
```

Arguments

flows	A vector of either origin (production constrained) or destination (attraction constrained) flows.
distance	a distance matrix between origins and destinations, provide distance in km.
weight	a vector of weights for the unconstrained part of the model.
beta	Exponent to use when calculating the cost function, default .25.

Value

A list containing a matrix with predicted values.

Examples

```
data(flows_test)
data(distance_test)

flows_test <- apply(flows_test,MARGIN = 1,FUN = sum)

model_test <- run_model_single(flows_test,distance_test)
```

simulation *Running a whole simulation of a doubly constrained gravity model*

Description

this script takes flows data, distance matrix, and a reference beta parameter and finds the optimal beta value for the model, runs it, and returns the result and the beta of best fit.

currently only the exp value is accepted for the cost_fun parameter.

Usage

```
simulation(flows_matrix, dist_matrix, beta_offset = 0.25)
```

Arguments

flows_matrix	a integer matrix of flows
dist_matrix	a distance matrix containing numeric values in kilometers
beta_offset	an offset from 0 from which to start looking for the best fit value.

Value

creates a folder based on the run_name parameter to which images and files are written. The file run_name_best_fit.rds contain the matrices with values from the model , and the quality of fit values for the beta values.

Examples

```
data(flows_test)
data(distance_test)

model <- simulation(flows_test,distance_test)
```

Index