# Package 'TDLM'

July 22, 2026

**Type** Package

**Title** Systematic Comparison of Trip Distribution Laws and Models

**Version** 1.1.4

**Description** The main purpose of this package is to propose a rigorous framework to fairly compare trip distribution laws and models as described in Lenormand et al. (2016) <doi:10.1016/j.jtrangeo.2015.12.008>.

**Depends** R (>= 4.0.0)

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**LazyDataCompression** xz

**Imports** Ecume, mathjaxr, Rdpack(>= 1.0.0), readr (>= 2.0.0), rmarkdown (>= 2.0.0), sf (>= 1.0.0)

**RdMacros** mathjaxr, Rdpack

**Suggests** knitr, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**BugReports** https://github.com/RTDLM/TDLM/issues

**URL** https://rtdlm.github.io/TDLM/

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Maxime Lenormand [aut, cre] (ORCID: <https://orcid.org/0000-0001-6362-3473>)

**Maintainer** Maxime Lenormand <maxime.lenormand@inrae.fr>

**Repository** CRAN

**Date/Publication** 2026-07-22 15:10:02 UTC

# Contents

calib_param . . . 2
check_format_names . . . 3
coords . . . 4
coords_xy . . . 5
county . . . 5
distance . . . 6
extract_distances . . . 6
extract_opportunities . . . 8
extract_spatial_information . . . 9
gof . . . 10
mass . . . 13
od . . . 14
run_law . . . 14
run_law_model . . . 17
run_model . . . 21

**Index** **24**

---

`calib_param` *Automatic calibration of trip distribution laws' parameter*

---

## Description

This function returns an estimate of the optimal parameter value based on the average surface area of the locations (in square kilometers) according to the law. This estimation has only been tested on commuting data (in kilometers).

## Usage

```
calib_param(av_surf, law = "NGravExp")
```

## Arguments

| | |
|---|---|
| `av_surf` | A positive `numeric` value indicating the average surface area of the locations (in square kilometers). |
| `law` | A `character` string indicating which law to use (see Details). |

## Details

The estimation is based on Figure 8 in Lenormand *et al.* (2016) for four types of laws: the normalized gravity law with an exponential distance decay function (`law = "NGravExp"`), the normalized gravity law with a power distance decay function (`law = "NGravPow"`), Schneider's intervening opportunities law (`law = "Schneider"`), and the extended radiation law (`law = "RadExt"`).

## Value

An estimate of the optimal parameter value based on the average surface area of the locations.

## Author(s)

Maxime Lenormand (<maxime.lenormand@inrae.fr>)

## References

Lenormand M, Bassolas A, Ramasco JJ (2016) Systematic comparison of trip distribution laws and models. *Journal of Transport Geography* 51, 158-169.

## See Also

Associated functions: `extract_opportunities()` `extract_spatial_information()` `check_format_names()`

## Examples

```
data(county)

res <- extract_spatial_information(county, id = "ID")
av_surf <- mean(res$surface)

calib_param(av_surf = av_surf, law = "NGravExp")
calib_param(av_surf = av_surf, law = "NGravPow")
calib_param(av_surf = av_surf, law = "Schneider")
calib_param(av_surf = av_surf, law = "RadExt")
```

---

# check_format_names *Check format of TDLM's inputs*

---

## Description

This function checks that the TDLM's inputs have the required format (and names).

## Usage

```
check_format_names(vectors, matrices = NULL, check = "format_and_names")
```

## Arguments

| | |
|---|---|
| `vectors` | A `list` of vectors. The list can contain one vector. It is recommended to name each element of the list. If `vectors` = `NULL`, only the matrices will be considered. |
| `matrices` | A `list` of matrices. The list can contain one matrix. It is recommended to name each element of the list. If `matrices` = `NULL`, only the vectors will be considered (by default). |
| `check` | A `character` indicating which types of check (`"format"` or `"format_and_names"`) should be used (see Details). |

## Details

The TDLM's inputs should be based on the same number of locations sorted in the same order. check = "format" will run basic checks to ensure that the structure of the inputs (dimensions, class, type...) is correct.

It is recommended to use the location ID as vector names, matrix rownames, and matrix colnames. Set check = "format_and_names" to check the inputs' names. The checks are run successively, so run the function as many times as needed to get the message indicating that the inputs passed the check successfully.

## Value

A message indicating if the check has passed or failed.

## Author(s)

Maxime Lenormand (<maxime.lenormand@inrae.fr>)

## Examples

```
data(mass)
data(distance)

mi <- as.numeric(mass[, 1])
names(mi) <- rownames(mass)
mj <- mi

check_format_names(
  vectors = list(mi = mi, mj = mj),
  matrices = list(distance = distance),
  check = "format_and_names"
)
```

---

# coords *Geographical coordinates of Kansas counties' centroids (Longitude/Latitude, 2000)*

---

## Description

A dataset containing the geographical coordinates of Kansas counties' centroids.

## Usage

```
coords
```

## Format

**Longitude** Longitude coordinate of the county centroid.

**Latitude** Latitude coordinate of the county centroid.

### Source

https://www.census.gov/geographies/mapping-files/time-series/geo/carto-boundary-file.html

---

`coords_xy` *Geographical coordinates of Kansas counties' centroids (X/Y, Web Mercator, 2000)*

---

### Description

A dataset containing the geographical coordinates of Kansas counties' centroids in Web Mercator projection.

### Usage

```
coords_xy
```

### Format

**X** X coordinate of the county centroid.

**Y** Y coordinate of the county centroid.

### Source

https://www.census.gov/geographies/mapping-files/time-series/geo/carto-boundary-file.html

---

`county` *Spatial distribution of Kansas counties, USA (2000)*

---

### Description

A dataset containing the geometry of 105 Kansas counties.

### Usage

```
county
```

### Format

**ID** County ID.

**Longitude** Longitude coordinate of the county centroid.

**Latitude** Latitude coordinate of the county centroid.

**Area** Surface area of the county (in square kilometers).

**geometry** Geometry of the county.

## Source

https://www.census.gov/geographies/mapping-files/time-series/geo/carto-boundary-file.html

---

`distance` *Great-circle distances between Kansas counties, USA*

---

## Description

A dataset containing the great-circle distance (in kilometers) between 105 Kansas counties.

## Usage

```
distance
```

## Format

A `matrix` with 105 rows and 105 columns. Each element represents the distance between two counties. County IDs are used as row and column names.

## Source

https://www.census.gov/geographies/mapping-files/time-series/geo/carto-boundary-file.html

---

`extract_distances` *Compute the distance between pairs of locations*

---

## Description

This function computes the distance between pairs of locations based on geographical coordinates.

## Usage

```
extract_distances(
  coords,
  method = "Haversine",
  id = NULL,
  show_progress = FALSE
)
```

## Arguments

| | |
|---|---|
| coords | A two-column matrix or data.frame where each row represents the coordinates of a location (see Details). |
| method | A character string indicating which method to choose to compute the distances (see Details). Available options are "Haversine" or "Euclidean". |
| id | A vector with length equal to the number of locations, used as row names and column names for the output distance matrix (optional, NULL by default). |
| show_progress | A boolean indicating whether a progress bar should be displayed. |

## Details

coords must contain two columns: the first one for the longitude or "X" coordinates, and the second one for the latitude or "Y" coordinates. The "Haversine" method is used to compute great-circle distances from longitude/latitude, while the "Euclidean" method should be used for "X/Y" coordinates.

## Value

A square matrix representing the distance (in kilometers) between locations.

## Note

The outputs are based on the locations contained in coords, sorted in the same order. An optional id can also be provided to be used as names for the outputs.

## Author(s)

Maxime Lenormand (<maxime.lenormand@inrae.fr>)

## See Also

Associated functions: extract_opportunities()

## Examples

```
data(coords)

distance <- extract_distances(coords = coords,
                              method = "Haversine",
                              id = rownames(coords))
```

`extract_opportunities` *Compute the number of opportunities between pairs of locations*

## Description

This function computes the number of opportunities between pairs of locations as defined in Lenormand *et al.* (2016). For a given pair of locations, the number of opportunities between the origin location and the destination location is based on the number of opportunities within a circle of radius equal to the distance between the origin and the destination, with the origin location as the center. The number of opportunities at the origin and destination locations are not included.

## Usage

```
extract_opportunities(opportunity, distance, check_names = FALSE)
```

## Arguments

| | |
|---|---|
| `opportunity` | A `numeric` vector representing the number of opportunities per location. The value should be positive. |
| `distance` | A squared `matrix` representing the distances between locations. |
| `check_names` | A `boolean` indicating whether the location IDs are used as `vector` names, `matrix` row names, and `matrix` column names, and whether they should be checked (see Note). |

## Value

A squared `matrix` in which each element represents the number of opportunities between a pair of locations.

## Note

`opportunity` and `distance` should be based on the same number of locations sorted in the same order. It is recommended to use the location ID as `matrix` `rownames` and `matrix` `colnames` and to set `check_names = TRUE` to verify that everything is consistent before running this function (`check_names = FALSE` by default). Note that the function `check_format_names()` can be used to validate all inputs before running the main package's functions.

## Author(s)

Maxime Lenormand (<maxime.lenormand@inrae.fr>)

## References

Lenormand M, Bassolas A, Ramasco JJ (2016) Systematic comparison of trip distribution laws and models. *Journal of Transport Geography* 51, 158-169.

## See Also

Associated functions: `extract_distances()`, `extract_spatial_information()`.

## Examples

```
data(mass)
data(distance)

opportunity <- mass[, 1]

sij <- extract_opportunities(opportunity = opportunity,
                             distance = distance,
                             check_names = FALSE)
```

---

# extract_spatial_information

*Extract distances and surface areas from a spatial object*

---

## Description

This function returns a `matrix` of distances between locations (in kilometers) along with a vector of surface areas for the locations (in square kilometers).

## Usage

```
extract_spatial_information(
  geometry,
  id = NULL,
  great_circle = FALSE,
  show_progress = FALSE
)
```

## Arguments

| | |
|---|---|
| `geometry` | A spatial object that can be handled by the `sf` package. |
| `id` | The name or number of the column to use as `rownames` and `colnames` for the output distance `matrix` (optional, `NULL` by default). A `vector` with a length equal to the number of locations can also be used. |
| `great_circle` | A `boolean` indicating whether distances and surface areas should be computed using longitude/latitude coordinates (see Details). |
| `show_progress` | A `boolean` indicating whether a progress bar should be displayed. |

## Details

The geometry must be projected in a valid coordinate reference system (CRS). By default, if great_circle = TRUE, the coordinates will be reprojected in degrees longitude/latitude to compute great-circle distances between centroids using an internal function, and surface areas will be calculated using sf::st_area(). If great_circle = FALSE, the coordinates are assumed to be planar (e.g., in meters) and Euclidean distances will be used.

## Value

A list composed of two elements. The first element is a square matrix representing the great-circle distances (in kilometers) between locations. The second element is a vector containing the surface area of each location (in square kilometers).

## Note

The outputs are based on the locations contained in geometry and sorted in the same order. An optional id can also be provided to be used as names for the outputs.

## Author(s)

Maxime Lenormand (<maxime.lenormand@inrae.fr>)

## See Also

Associated functions: extract_distances() extract_opportunities()

## Examples

```
data(county)

res <- extract_spatial_information(county, id = "ID")

dim(res$distance)

length(res$surface)
```

---

# gof *Compute goodness-of-fit measures between observed and simulated OD matrices*

---

## Description

This function returns a data.frame where each row provides one or several goodness-of-fit measures between a simulated and an observed Origin-Destination (OD) matrix.

## Usage

```
gof(
  sim,
  obs,
  measures = "all",
  distance = NULL,
  bin_size = 2,
  use_proba = FALSE,
  check_names = FALSE
)
```

## Arguments

| | |
|---|---|
| sim | An object of class TDLM (output of run_law_model(), run_law(), or run_model()). A matrix or a list of matrices can also be used (see Note). |
| obs | A square matrix representing the observed mobility flows. |
| measures | A character vector or a single character string indicating which goodness-of-fit measure(s) to compute (see Details). Available options are "CPC", "NRMSE", "KL", "CPL", "CPC_d" and "KS". If "all" is specified, all measures will be calculated. |
| distance | A square matrix representing the distances between locations. This is only necessary for distance-based measures. |
| bin_size | A numeric value indicating the size of bins used to discretize the distance distribution when computing CPC_d (default is 2 kilometers). |
| use_proba | A boolean indicating whether the proba matrix should be used instead of the simulated OD matrix to compute the measure(s). This is only valid for output from run_law_model() with the argument write_proba = TRUE (see Note). |
| check_names | A boolean indicating whether the location IDs used as matrix rownames and colnames should be checked for consistency (see Note). |

## Details

Several goodness-of-fit measures are considered, such as the Common Part of Commuters (CPC), the Common Part of Links (CPL), and the Common Part of Commuters based on the distance (CPC_d), as described in Lenormand *et al.* (2016). It also includes classical metrics such as the Normalized Root Mean Square Error (NRMSE), the Kullback–Leibler divergence (KL), and the Kolmogorov-Smirnov statistic and p-value (KS). These measures are based on the observed and simulated flow distance distributions and are computed using the ks_test function from the Ecume package.

## Value

A data.frame providing one or several goodness-of-fit measures between simulated OD(s) and an observed OD. Each row corresponds to a matrix sorted according to the list (or list of lists) elements (names are used if provided).

## Note

By default, if sim is an output of run_law_model(), the measure(s) are computed only for the simulated OD matrices and not for the proba matrix (included in the output when write_proba = TRUE). The argument use_proba can be used to compute the measure(s) based on the proba matrix instead of the simulated OD matrix. In this case, the argument obs should also be a proba matrix.

All inputs should be based on the same number of locations, sorted in the same order. It is recommended to use the location ID as matrix rownames and matrix colnames and to set check_names = TRUE to verify that everything is consistent before running this function (check_names = FALSE by default). Note that the function check_format_names() can be used to validate all inputs before running the main package's functions.

## Author(s)

Maxime Lenormand (<maxime.lenormand@inrae.fr>)

## References

Lenormand M, Bassolas A, Ramasco JJ (2016) Systematic comparison of trip distribution laws and models. *Journal of Transport Geography* 51, 158-169.

## See Also

For more details illustrated with a practical example, see the vignette: https://rtdlm.github.io/TDLM/articles/TDLM.html#goodness-of-fit-measures.

Associated functions: run_law(), run_model(), run_law_model().

## Examples

```
data(mass)
data(distance)
data(od)

mi <- as.numeric(mass[, 1])
mj <- mi
Oi <- as.numeric(mass[, 2])
Dj <- as.numeric(mass[, 3])

res <- run_law_model(law = "GravExp",
                     mass_origin = mi,
                     mass_destination = mj,
                     distance = distance,
                     opportunity = NULL,
                     param = 0.01,
                     model = "DCM",
                     nb_trips = NULL,
                     out_trips = Oi,
                     in_trips = Dj,
                     average = FALSE,
                     nbrep = 1,
                     maxiter = 50,
```

```
                      mindiff = 0.01,
                      write_proba = FALSE,
                      check_names = FALSE)

gof(sim = res,
    obs = od,
    measures = "CPC",
    distance = NULL,
    bin_size = 2,
    use_proba = FALSE,
    check_names = FALSE)
```

---

`mass` *Population and number of out- and in-commuters in Kansas counties, USA (2000)*

---

## Description

A dataset containing the number of inhabitants, in-commuters, and out-commuters for 105 Kansas counties in 2000.

## Usage

```
mass
```

## Format

A `data.frame` with 105 rows and 4 columns:

**rownames** County ID.

**Population** Number of inhabitants.

**Out_Commuters** Number of out-commuters.

**In_Commuters** Number of in-commuters.

## Source

https://www2.census.gov/programs-surveys/decennial/tables/2000/county-to-county-worker-flow-files/

## od — *Origin-Destination commuting matrix for Kansas counties, USA (2000)*

### Description

A dataset containing the number of commuters between 105 Kansas counties in 2000.

### Usage

```
od
```

### Format

A `matrix` with 105 rows and 105 columns. Each element represents the number of commuters between two counties. County IDs are used as row and column names.

### Source

<https://www2.census.gov/programs-surveys/decennial/tables/2000/ county-to-county-worker-flow-files/>

## run_law — *Estimate mobility flows based on different trip distribution laws*

### Description

This function estimates mobility flows using different distribution laws and models. As described in Lenormand *et al.* (2016), the function uses a two-step approach to generate mobility flows by separating the trip distribution law (gravity or intervening opportunities) from the modeling approach used to generate the flows based on this law. This function only uses the first step to generate a probability distribution based on the different laws.

### Usage

```
run_law(
  law = "Unif",
  mass_origin,
  mass_destination = mass_origin,
  distance = NULL,
  opportunity = NULL,
  param = NULL,
  check_names = FALSE
)
```

## Arguments

| | |
|---|---|
| law | A character indicating which law to use (see Details). |
| mass_origin | A numeric vector representing the mass at the origin (i.e. demand). |
| mass_destination | A numeric vector representing the mass at the destination (i.e. attractiveness). |
| distance | A squared matrix representing the distance between locations (see Details). |
| opportunity | A squared matrix representing the number of opportunities between locations (see Details). Can be easily computed with extract_opportunities(). |
| param | A numeric vector or a single numeric value used to adjust the importance of distance or opportunity associated with the chosen law. Not necessary for the original radiation law or the uniform law (see Details). |
| check_names | A boolean indicating whether the location IDs used as matrix rownames and colnames should be checked for consistency (see Note). |

## Details

We compute the matrix proba estimating the probability to observe a trip from one location to another. This probability is based on the demand (argument mass_origin) and the attractiveness (argument mass_destination). Note that the population is typically used as a surrogate for both quantities (this is why mass_destination = mass_origin by default). It also depends on the distance between locations (argument distance) OR the number of opportunities between locations (argument opportunity) depending on the chosen law. Both the effect of the distance and the number of opportunities can be adjusted with a parameter (argument param) except for the original radiation law and the uniform law.

In this package we consider eight probabilistic laws described in details in Lenormand *et al.* (2016). Four gravity laws (Barthelemy, 2011), three intervening opportunity laws (Schneider, 1959; Simini *et al.*, 2012; Yang *et al.*, 2014) and a uniform law.

1. Gravity law with an exponential distance decay function (law = "GravExp"). The arguments mass_origin, mass_destination (optional), distance and param will be used.
2. Normalized gravity law with an exponential distance decay function (law = "NGravExp"). The arguments mass_origin, mass_destination (optional), distance and param will be used.
3. Gravity law with a power distance decay function (law = "GravPow"). The arguments mass_origin, mass_destination (optional), distance and param will be used.
4. Normalized gravity law with a power distance decay function (law = "NGravPow"). The arguments mass_origin, mass_destination (optional), distance and param will be used.
5. Schneider's intervening opportunities law (law = "Schneider"). The arguments mass_origin, mass_destination (optional), opportunity and param will be used.
6. Radiation law (law = "Rad"). The arguments mass_origin, mass_destination (optional) and opportunity will be used.
7. Extended radiation law (law = "RadExt"). The arguments mass_origin, mass_destination (optional), opportunity and param will be used.
8. Uniform law (law = "Unif"). The argument mass_origin will be used to extract the number of locations.

## Value

An object of class TDLM. An object of class TDLM. A list of list of matrice containing for each parameter value the matrix of probabilities (called proba). If length(param) = 1 or law = "Rad" or law = "Unif" only a list of matrices will be returned.

## Note

All inputs should be based on the same number of locations, sorted in the same order. It is recommended to use the location ID as matrix rownames and matrix colnames and to set check_names = TRUE to verify that everything is consistent before running this function (check_names = FALSE by default). Note that the function check_format_names() can be used to validate all inputs before running the main package's functions.

## Author(s)

Maxime Lenormand (<maxime.lenormand@inrae.fr>)

## References

Barthelemy M (2011). Spatial Networks. *Physics Reports* 499, 1-101.

Lenormand M, Bassolas A, Ramasco JJ (2016) Systematic comparison of trip distribution laws and models. *Journal of Transport Geography* 51, 158-169.

Schneider M (1959) Gravity models and trip distribution theory. *Papers of the regional science association* 5, 51-58.

Simini F, González MC, Maritan A & Barabási A (2012) A universal model for mobility and migration patterns. *Nature* 484, 96-100.

Yang Y, Herrera C, Eagle N & González MC (2014) Limits of Predictability in Commuting Flows in the Absence of Data for Calibration. *Scientific Reports* 4, 5662.

## See Also

For more details illustrated with a practical example, see the vignette: https://rtdlm.github.io/TDLM/articles/TDLM.html#run-functions.

Associated functions: run_law_model(), run_model(), gof().

## Examples

```
data(mass)
data(distance)

mi <- as.numeric(mass[, 1])
mj <- mi

res <- run_law(
  law = "GravExp", mass_origin = mi, mass_destination = mj,
  distance = distance, opportunity = NULL, param = 0.01,
  check_names = FALSE
)
```

```
# print(res)
```

---

`run_law_model` *Estimate mobility flows based on different trip distribution laws and models*

---

## Description

This function estimates mobility flows using different distribution laws and models. As described in Lenormand *et al.* (2016), the function uses a two-step approach to generate mobility flows by separating the trip distribution law (gravity or intervening opportunities) from the modeling approach used to generate the flows based on this law.

## Usage

```
run_law_model(
  law = "Unif",
  mass_origin,
  mass_destination = mass_origin,
  distance = NULL,
  opportunity = NULL,
  param = NULL,
  model = "UM",
  nb_trips = 1000,
  out_trips = NULL,
  in_trips = out_trips,
  average = FALSE,
  nbrep = 3,
  maxiter = 50,
  mindiff = 0.01,
  eps = 1e-06,
  write_proba = FALSE,
  check_names = FALSE
)
```

## Arguments

| | |
|---|---|
| `law` | A `character` indicating which law to use (see Details). |
| `mass_origin` | A `numeric` vector representing the mass at the origin (i.e. demand). |
| `mass_destination` | A `numeric` vector representing the mass at the destination (i.e. attractiveness). |
| `distance` | A squared `matrix` representing the distance between locations (see Details). |
| `opportunity` | A squared `matrix` representing the number of opportunities between locations (see Details). Can be easily computed with `extract_opportunities()`. |

| | |
|---|---|
| param | A numeric vector or a single numeric value used to adjust the importance of distance or opportunity associated with the chosen law. Not necessary for the original radiation law or the uniform law (see Details). |
| model | A character indicating which model to use. |
| nb_trips | A numeric value indicating the total number of trips. Must be an integer if average = FALSE (see Details). |
| out_trips | A numeric vector representing the number of outgoing trips per location. Must be a vector of integers if average = FALSE (see Details). |
| in_trips | A numeric vector representing the number of incoming trips per location. Must be a vector of integers if average = FALSE (see Details). |
| average | A boolean indicating if the average mobility flow matrix should be generated instead of the nbrep matrices based on random draws (see Details). |
| nbrep | An integer indicating the number of replications associated with the model run. Note that nbrep = 1 if average = TRUE (see Details). |
| maxiter | An integer indicating the maximal number of iterations for adjusting the Doubly Constrained Model (see Details). |
| mindiff | A numeric strictly positive value indicating the stopping criterion for adjusting the Doubly Constrained Model (see Details). |
| eps | A strictly positive numeric value used to replace any zero values in the marginals of the Doubly Constrained Model, in order to avoid numerical issues during the IPF procedure (see Details). |
| write_proba | A boolean indicating if the estimation of the probability to move from one location to another obtained with the distribution law should be returned along with the flow estimations. |
| check_names | A boolean indicating whether the location IDs used as matrix rownames and colnames should be checked for consistency (see Note). |

## Details

First, we compute the matrix proba estimating the probability to observe a trip from one location to another. This probability is based on the demand (argument mass_origin) and the attractiveness (argument mass_destination). Note that the population is typically used as a surrogate for both quantities (this is why mass_destination = mass_origin by default). It also depends on the distance between locations (argument distance) OR the number of opportunities between locations (argument opportunity) depending on the chosen law. Both the effect of the distance and the number of opportunities can be adjusted with a parameter (argument param) except for the original radiation law and the uniform law.

In this package we consider eight probabilistic laws described in details in Lenormand *et al.* (2016). Four gravity laws (Barthelemy, 2011), three intervening opportunity laws (Schneider, 1959; Simini *et al.*, 2012; Yang *et al.*, 2014) and a uniform law.

1. Gravity law with an exponential distance decay function (law = "GravExp"). The arguments mass_origin, mass_destination (optional), distance and param will be used.
2. Normalized gravity law with an exponential distance decay function (law = "NGravExp"). The arguments mass_origin, mass_destination (optional), distance and param will be used.

3. Gravity law with a power distance decay function (`law = "GravPow"`). The arguments `mass_origin`, `mass_destination` (optional), `distance` and `param` will be used.
4. Normalized gravity law with a power distance decay function (`law = "NGravPow"`). The arguments `mass_origin`, `mass_destination` (optional), `distance` and `param` will be used.
5. Schneider's intervening opportunities law (`law = "Schneider"`). The arguments `mass_origin`, `mass_destination` (optional), `opportunity` and `param` will be used.
6. Radiation law (`law = "Rad"`). The arguments `mass_origin`, `mass_destination` (optional) and `opportunity` will be used.
7. Extended radiation law (`law = "RadExt"`). The arguments `mass_origin`, `mass_destination` (optional), `opportunity` and `param` will be used.
8. Uniform law (`law = "Unif"`). The argument `mass_origin` will be used to extract the number of locations.

Second, we propose four constrained models to generate the flows from these distribution of probability as described in Lenromand *et al.* (2016). These models respect different level of constraints. These constraints can preserve the total number of trips (argument `nb_trips`) OR the number of out-going trips (argument `out_trips`) AND/OR the number of in-coming (argument `in_trips`) according to the model. The sum of out-going trips should be equal to the sum of in-coming trips.

1. Unconstrained model (`model = "UM"`). Only `nb_trips` will be preserved (arguments `out_trips` and `in_trips` will not be used).
2. Production constrained model (`model = "PCM"`). Only `out_trips` will be preserved (arguments `nb_trips` and `in_trips` will not be used).
3. Attraction constrained model (`model = "ACM"`). Only `in_trips` will be preserved (arguments `nb_trips` and `out_trips` will not be used).
4. Doubly constrained model (`model = "DCM"`). Both `out_trips` and `in_trips` will be preserved (arguments `nb_trips`will not be used). The doubly constrained model is based on an Iterative Proportional Fitting process (Deming & Stephan, 1940). The arguments `maxiter` (50 by default) and `mindiff` (0.01 by default) can be used to tune the model. `mindiff` is the minimal tolerated relative error between the simulated and observed marginals. `maxiter` ensures that the algorithm stops even if it has not converged toward the `mindiff` wanted value. The argument `eps` (1e-6 by default) can be used to replace zero values in the marginals in order to avoid numerical issues during the IPF procedure.

By default, when `average = FALSE`, `nbrep` matrices are generated from `proba` with multinomial random draws that will take different forms according to the model used. In this case, the models will deal with positive integers as inputs and outputs. Nevertheless, it is also possible to generate an average matrix based on a multinomial distribution (based on an infinite number of drawings). In this case, the models' inputs can be either positive integer or real numbers and the output (`nbrep = 1` in this case) will be a matrix of positive real numbers.

## Value

An object of class `TDLM`. A `list` of `list` of matrices containing for each parameter value the `nbrep` simulated matrices and the matrix of probabilities (called `proba`) if `write_proba = TRUE`. If `length(param) = 1` or `law = "Rad"` or `law = "Unif"` only a list of matrices will be returned.

## Note

All inputs should be based on the same number of locations, sorted in the same order. It is recommended to use the location ID as `matrix rownames` and `matrix colnames` and to set `check_names = TRUE` to verify that everything is consistent before running this function (`check_names = FALSE` by default). Note that the function `check_format_names()` can be used to validate all inputs before running the main package's functions.

## Author(s)

Maxime Lenormand (<maxime.lenormand@inrae.fr>)

## References

Barthelemy M (2011). Spatial Networks. *Physics Reports* 499, 1-101.

Deming WE & Stephan FF (1940) On a Least Squares Adjustment of a Sample Frequency Table When the Expected Marginal Totals Are Known. *Annals of Mathematical Statistics* 11, 427-444.

Lenormand M, Bassolas A, Ramasco JJ (2016) Systematic comparison of trip distribution laws and models. *Journal of Transport Geography* 51, 158-169.

Schneider M (1959) Gravity models and trip distribution theory. *Papers of the regional science association* 5, 51-58.

Simini F, González MC, Maritan A & Barabási A (2012) A universal model for mobility and migration patterns. *Nature* 484, 96-100.

Yang Y, Herrera C, Eagle N & González MC (2014) Limits of Predictability in Commuting Flows in the Absence of Data for Calibration. *Scientific Reports* 4, 5662.

## See Also

For more details illustrated with a practical example, see the vignette: https://rtdlm.github.io/TDLM/articles/TDLM.html#run-functions.

Associated functions: `run_law()`, `run_model()`, `gof()`.

## Examples

```
data(mass)
data(distance)

mi <- as.numeric(mass[, 1])
mj <- mi
N <- 1000

res <- run_law_model(law = "GravExp",
                     mass_origin = mi,
                     mass_destination = mj,
                     distance = distance,
                     opportunity = NULL,
                     param = 0.01,
                     model = "UM",
                     nb_trips = N,
```

```
                    out_trips = NULL,
                    in_trips = NULL,
                    average = TRUE,
                    nbrep = 2,
                    maxiter = 50,
                    mindiff = 0.01,
                    eps = 0.000001,
                    write_proba = FALSE,
                    check_names = FALSE)
```

---

`run_model` *Estimate mobility flows based on different trip distribution models*

---

## Description

This function estimates mobility flows using different distribution laws and models. As described in Lenormand *et al.* (2016), the function uses a two-step approach to generate mobility flows by separating the trip distribution law (gravity or intervening opportunities) from the modeling approach used to generate the flows based on this law. This function only uses the second step to generate mobility flow based on a matrix of probabilities using different models.

## Usage

```
run_model(
  proba,
  model = "UM",
  nb_trips = 1000,
  out_trips = NULL,
  in_trips = out_trips,
  average = FALSE,
  nbrep = 3,
  maxiter = 50,
  mindiff = 0.01,
  eps = 1e-06,
  check_names = FALSE
)
```

## Arguments

| | |
|---|---|
| `proba` | A squared `matrix` of probability. The sum of the matrix element must be equal to 1. It will be normalized automatically if it is not the case. |
| `model` | A `character` indicating which model to use. |
| `nb_trips` | A `numeric` value indicating the total number of trips. Must be an `integer` if `average = FALSE` (see Details). |
| `out_trips` | A `numeric` vector representing the number of outgoing trips per location. Must be a vector of integers if `average = FALSE` (see Details). |

| | |
|---|---|
| in_trips | A numeric vector representing the number of incoming trips per location. Must be a vector of integers if average = FALSE (see Details). |
| average | A boolean indicating if the average mobility flow matrix should be generated instead of the nbrep matrices based on random draws (see Details). |
| nbrep | An integer indicating the number of replications associated with the model run. Note that nbrep = 1 if average = TRUE (see Details). |
| maxiter | An integer indicating the maximal number of iterations for adjusting the Doubly Constrained Model (see Details). |
| mindiff | A numeric strictly positive value indicating the stopping criterion for adjusting the Doubly Constrained Model (see Details). |
| eps | A strictly positive numeric value used to replace any zero values in the marginals of the Doubly Constrained Model, in order to avoid numerical issues during the IPF procedure (see Details). |
| check_names | A boolean indicating whether the location IDs used as matrix rownames and colnames should be checked for consistency (see Note). |

## Details

We propose four constrained models to generate the flows from these distribution of probability as described in Lenromand *et al.* (2016). These models respect different level of constraints. These constraints can preserve the total number of trips (argument nb_trips) OR the number of out-going trips (argument out_trips) AND/OR the number of in-coming (argument in_trips) according to the model. The sum of out-going trips should be equal to the sum of in-coming trips.

1. Unconstrained model (model = "UM"). Only nb_trips will be preserved (arguments out_trips and in_trips will not be used).
2. Production constrained model (model = "PCM"). Only out_trips will be preserved (arguments nb_trips and in_trips will not be used).
3. Attraction constrained model (model = "ACM"). Only in_trips will be preserved (arguments nb_trips and out_trips will not be used).
4. Doubly constrained model (model = "DCM"). Both out_trips and in_trips will be preserved (arguments nb_tripswill not be used). The doubly constrained model is based on an Iterative Proportional Fitting process (Deming & Stephan, 1940). The arguments maxiter (50 by default) and mindiff (0.01 by default) can be used to tune the model. mindiff is the minimal tolerated relative error between the simulated and observed marginals. maxiter ensures that the algorithm stops even if it has not converged toward the mindiff wanted value. The argument eps (1e-6 by default) can be used to replace zero values in the marginals in order to avoid numerical issues during the IPF procedure.

By default, when average = FALSE, nbrep matrices are generated from proba with multinomial random draws that will take different forms according to the model used. In this case, the models will deal with positive integers as inputs and outputs. Nevertheless, it is also possible to generate an average matrix based on a multinomial distribution (based on an infinite number of drawings). In this case, the models' inputs can be either positive integer or real numbers and the output (nbrep = 1 in this case) will be a matrix of positive real numbers.

## Value

An object of class TDLM. A list of matrices containing the nbrep simulated matrices.

## Note

All inputs should be based on the same number of locations, sorted in the same order. It is recommended to use the location ID as matrix rownames and matrix colnames and to set check_names = TRUE to verify that everything is consistent before running this function (check_names = FALSE by default). Note that the function check_format_names() can be used to validate all inputs before running the main package's functions.

## Author(s)

Maxime Lenormand (<maxime.lenormand@inrae.fr>)

## References

Deming WE & Stephan FF (1940) On a Least Squares Adjustment of a Sample Frequency Table When the Expected Marginal Totals Are Known. *Annals of Mathematical Statistics* 11, 427-444.

Lenormand M, Bassolas A, Ramasco JJ (2016) Systematic comparison of trip distribution laws and models. *Journal of Transport Geography* 51, 158-169.

## See Also

For more details illustrated with a practical example, see the vignette: https://rtdlm.github.io/TDLM/articles/TDLM.html#run-functions.

Associated functions: run_law_model(), run_law(), gof().

## Examples

```
data(mass)
data(od)

proba <- od / sum(od)

Oi <- as.numeric(mass[, 2])
Dj <- as.numeric(mass[, 3])

res <- run_model(proba = proba,
                 model = "DCM",
                 nb_trips = NULL,
                 out_trips = Oi,
                 in_trips = Dj,
                 average = FALSE,
                 nbrep = 3,
                 maxiter = 50,
                 mindiff = 0.01,
                 eps = 0.000001,
                 check_names = FALSE)
```

# Index


∗ **datasets**
coords, 4
coords_xy, 5
county, 5
distance, 6
mass, 13
od, 14

calib_param, 2
check_format_names, 3
check_format_names(), *3*, *8*, *12*, *16*, *20*, *23*
coords, 4
coords_xy, 5
county, 5

distance, 6

extract_distances, 6
extract_distances(), *9, 10*
extract_opportunities, 8
extract_opportunities(), *3*, *7*, *10*, *15*, *17*
extract_spatial_information, 9
extract_spatial_information(), *3*, *9*

gof, 10
gof(), *16*, *20*, *23*

ks_test, *11*

mass, 13

od, 14

run_law, 14
run_law(), *11, 12*, *20*, *23*
run_law_model, 17
run_law_model(), *11, 12*, *16*, *23*
run_model, 21
run_model(), *11, 12*, *16*, *20*

sf::st_area(), *10*