Package 'NetOrigin'

April 20, 2018

Title Origin Estimation for Propagation Processes on Complex Networks

Description Performs network-based source estimation. Different approaches are available: effective distance median, recursive backtracking, and centrality-based source estimation. Additionally, we provide public transportation network data as well as methods for data preparation, source estimation performance analysis and visualization.

Version 1.0-3

Depends R (>= 3.2.2)

Imports igraph, Hmisc, colorspace

License GPL-3

LazyData true

RoxygenNote 5.0.1

Collate '0_helper_net.r' 'NetOrigin.r' 'origin_helper.r' 'origin_methods.r' 'distance.r' 'data.r' 'data_handling.r' 'robustness.r'

NeedsCompilation no

Author Juliane Manitz [aut, cre], Jonas Harbering [ctb]

Maintainer Juliane Manitz <r@manitz.org>

Repository CRAN

Date/Publication 2018-04-20 03:12:11 UTC

R topics documented:

`aggr_data` *convert individual event information to aggregated information per network node*

Description

convert individual event information to aggregated information per network node

Usage

```
aggr_data(dat, from = NULL, cumsum = TRUE)
```

Arguments

`dat`	`data.frame` with variables `'node'`, `'time'`, `'delay'`, events data with single events with count magnitude
`from`	character in `strftime` format, e.g. `"2014-06-12 16:15"`, data is subsetted accordingly before aggregation
`cumsum`	logical indicating whether data is aggregated by cumulative sum, default is `TRUE`

Value

`data.frame` of dimension `(TxK)`, where `T` is the number of observation times and `K` the number of network nodes. Thus, each row represents a snapshot of the spreading process at a specific observation time with the event magnitude observed at the network nodes. Rownames are observation times, colnames are node names.

See Also

Other data_handling: `read_DB_data`

`analyze_ptn` *analyze public transportation network characteristics*

Description

analyze public transportation network characteristics

Usage

```
analyze_ptn(g)
```

Arguments

`g`	`igraph` object, network graph representing the public transportation network, vertices represent stations, which are linked by an edge if there is a direct transfer between them

Value

`'data.frame'`: 1 obs. of 7 variables:

- `vcount` number of nodes,
- `ecount` number of edges,
- `density` network graph density,
- `av_deg` average degree,
- `av_cent` average unit betweenness,
- `av_spl` average shortest path length,
- `diam` diameter, and
- `trans` transitivity.

References

Details to the computation and interpretation can be found in:

- Kolaczyk, E. D. (2009). Statistical analysis of network data: methods and models. Springer series in statistics. Springer. <DOI: 10.1007/978-0-387-88146-1>
- Manitz, J. (2014): Statistical Inference for Propagation Processes on Complex Networks. Ph.D. thesis, Georg-August-University Goettingen. Verlag Dr.~Hut, ISBN 978-3-8439-1668-4. Available online: http://ediss.uni-goettingen.de/handle/11858/00-1735-0000-0022-5F38-B.

See Also

Other network helper: `plot_ptn`

Examples

```
data(ptnAth)
analyze_ptn(ptnAth)

data(ptnGoe)
analyze_ptn(ptnGoe)
```

`delay-data` *Delay propagation data examples simulated by LinTim software*

Description

Delay propagation data examples simulated by LinTim software

`delayAth` Delay propagation data generated on the Athens metro network by LinTim software

`delayGoe` Delay propagation data generated on the Goettingen bus system by LinTim software

Details

`delayAth` Delay data on the Athens metro network. Propagation simulation under consideration of secruity distances and fixed-waiting time delay management. 'data.frame' with 510 observations (10 sequential time pictures for delay spreading pattern from 51 stations) of 53 variables (`k0` true source, `time`, delays at 51 stations).

`delayGoe` Delay data on the directed Goettingen bus system. Progation simulation under consideration of secruity distances and fixed-waiting time delay management. 'data.frame' with 2570 observations (10 sequential time pictures for delay spreading pattern from 257 stations) of 259 variables (`k0` true source, `time`, delays at 257 stations).

Author(s)

Jonas Harbering

Source

Public transportation network datasets are generated by LinTim software (Integrated Optimization in Public Transportation; http://lintim.math.uni-goettingen.de/index.php?go=data&lang=en).

References

Manitz, J., J. Harbering, M. Schmidt, T. Kneib, and A. Schoebel (2016). Source Estimation for Propagation Processes on Complex Networks with an Application to Delays in Public Transportation Systems. Accepted at JRSS-C.

See Also

ptn-data

Examples

```
## Not run:
# compute effective distance
data(ptnAth)
athnet <- igraph::as_adjacency_matrix(ptnAth, sparse=FALSE)
p <- athnet/rowSums(athnet)
eff <- eff_dist(p)
# apply source estimation
if (requireNamespace("aplyr", quietly = TRUE)) {
data(delayAth)
res <- alply(.data=delayAth[,-c(1:2)], .margins=1, .fun=origin_edm, distance=eff,
             silent=TRUE, .progress='text')
perfAth <- ldply(Map(performance, x = res, start = as.list(delayAth$k0),
                     list(graph = ptnAth)))
}

## End(Not run)
## Not run:
# compute effective distance
data(ptnGoe)
goenet <- igraph::as_adjacency_matrix(ptnGoe, sparse=FALSE)
p <- goenet/rowSums(goenet)
eff <- eff_dist(p)
# apply source estimation
if (requireNamespace("aplyr", quietly = TRUE)) {
data(delayGoe)
res <- alply(.data=delayGoe[,-c(1:2)], .margins=1, .fun=origin_edm, distance=eff,
             silent=TRUE, .progress='text')
perfGoe <- ldply(Map(performance, x = res, start = as.list(delayGoe$k0),
                     list(graph = ptnGoe)))
}

## End(Not run)
```

`eff_dist` *Computation of effective path distance*

Description

`eff_dist` computes the effective distance between all nodes in the network

`eff_dijkstra` computes the shortest effective paths using the dijkstra algorithm

`spd_dijkstra` computes the shortest paths using the dijkstra algorithm

Usage

```
eff_dist(p)

eff_dijkstra(p, start)

spd_dijkstra(p, start)
```

Arguments

p	numeric matrix, representing the transition probability matrix for the network graph
start	start of path

Value

A numeric matrix, representing the effective distance between all nodes in the network graph.

References

- Dijkstra, E. W. (1959): A note on two problems in connexion with graphs. Numerische Mathematik, 1, 269-271. <DOI: 10.1007/BF01386390>
- Brockmann, D. and Helbing, D. (2013): The hidden geometry of complex, network-driven contagion phenomena. Science, 342, 1337-1342. <DOI: 10.1126/science.1245200>
- Manitz, J. (2014): Statistical Inference for Propagation Processes on Complex Networks. Ph.D. thesis, Georg-August-University Goettingen. Verlag Dr. Hut, ISBN 978-3-8439-1668-4. Available online: http://ediss.uni-goettingen.de/handle/11858/00-1735-0000-0022-5F38-B.

Examples

```
# compute effective shortest path distance
data(ptnAth)
require(igraph)
net <- igraph::as_adjacency_matrix(ptnAth, sparse=FALSE)
p <- net/rowSums(net)
eff <- eff_dist(p)

# compute shortest path distance
data(ptnAth)
athnet <- as_adj(ptnAth, sparse=FALSE)
spd <- spd_dijkstra(athnet, start=1)

# compare calculations with the one from igraph
spd_igraph <- igraph::distances(ptnAth, v=1, algorithm='dijkstra')
all(spd[[1]] == spd_igraph)
```

NetOrigin *Origin Estimation for Propagation Processes on Complex Networks*

Description

Performs different approaches for network-based source estimation: effective distance median, recursive backtracking, and centrality-based source estimation. Additionally, we provide public transportation network data as well as methods for data preparation, source estimation performance analysis and visualization.

Details

The main function for origin estimation of propagation processes on complex network is `origin`. Different methods are available: effective distance median (`'edm'`), recursive backtracking (`'backtracking'`), and centrality-based source estimation (`'centrality'`). For more details on the methodological background, we refer to the corresponding publications.

Author(s)

Juliane Manitz with contributions by Jonas Harbering

References

- Manitz, J., J. Harbering, M. Schmidt, T. Kneib, and A. Schoebel (2016). Source Estimation for Propagation Processes on Complex Networks with an Application to Delays in Public Transportation Systems. Accepted at JRSS-C.
- Manitz, J., Kneib, T., Schlather, M., Helbing, D. and Brockmann, D. (2014) Origin detection during food-borne disease outbreaks - a case study of the 2011 EHEC/HUS outbreak in Germany. PLoS Currents Outbreaks, 1. <DOI: 10.1371/currents.outbreaks.f3fdeb08c5b9de7c09ed9cbcef5f01f2>
- Comin, C. H. and da Fontoura Costa, L. (2011) Identifying the starting point of a spreading process in complex networks. Physical Review E, 84. <DOI: 10.1103/PhysRevE.84.056105>

`origin` *Origin Estimation for Propagation Processes on Complex Networks*

Description

This is the main function for origin estimation for propagation processes on complex networks. Different methods are available: effective distance median (`'edm'`), recursive backtracking (`'backtracking'`), and centrality-based source estimation (`'centrality'`). For details on the methodological background, we refer to the corresponding publications.

`origin_edm` for effective distance-median origin estimation (Manitz et al., 2016)

`origin_backtracking` for recursive backtracking origin estimation (Manitz et al., 2016)

`origin_centrality` for centrality-based origin estimation (Comin et al., 2011)

Usage

```
origin(events, type = c("edm", "backtracking", "centrality"), ...)

origin_edm(events, distance, silent = TRUE)

origin_backtracking(events, graph, start_with_event_node = TRUE,
  silent = TRUE)

origin_centrality(events, graph, silent = TRUE)
```

Arguments

events	numeric vector of event counts at a specific time point
type	character specifying the method, 'edm', 'backtracking' and 'centrality' are available.
...	parameters to be passed to origin methods origin_edm, origin_backtracking or origin_centrality
distance	numeric matrix specifying the distance matrix (for type='edm')
silent	locigal, should the messages be suppressed?
graph	igraph object specifying the underlying network graph (for type='backtracking' and type='centrality')
start_with_event_node	logical specifying whether backtracking only starts from nodes that experienced events (for type='backtracking')

Value

origin_edm returns an object of class origin, list with

- est origin estimate
- aux data.frame with auxiliary variables
  - id as node identifier,
  - events for event magnitude,
  - wmean for weighted mean,
  - wvar for weighted variance, and
  - mdist mean distance from a node to all other nodes.
- type = 'edm' effective distance median origin estimation

origin_backtracking returns an object of class origin, list with

- est origin estimate
- aux data.frame with auxiliary variables
  - id as node identifier,
  - events for event magnitude, and
  - bcount for backtracking counts, how often backtracking identifies this source node.
- type = 'backtracking' backtracking origin estimation

origin_centrality returns an object of class origin, list with

- est origin estimate
- aux data.frame with auxiliary variables
  - id as node identifier,
  - events for event magnitude, and
  - cent for node centrality (betweenness divided degree).
- type = 'centrality' centrality-based origin estimation

Author(s)

Juliane Manitz with contributions by Jonas Harbering

References

- Comin, C. H. and da Fontoura Costa, L. (2011). Identifying the starting point of a spreading process in complex networks. Physical Review E, 84. <DOI: 10.1103/PhysRevE.84.056105>
- Manitz, J., J. Harbering, M. Schmidt, T. Kneib, and A. Schoebel (2016). Source Estimation for Propagation Processes on Complex Networks with an Application to Delays in Public Transportation Systems. Accepted at JRSS-C.
- Manitz, J. (2014). Statistical Inference for Propagation Processes on Complex Networks. Ph.D. thesis, Georg-August-University Goettingen. Verlag Dr.~Hut, ISBN 978-3-8439-1668-4. Available online: http://ediss.uni-goettingen.de/handle/11858/00-1735-0000-0022-5F38-B.
- Manitz, J., Kneib, T., Schlather, M., Helbing, D. and Brockmann, D. (2014). Origin detection during food-borne disease outbreaks - a case study of the 2011 EHEC/HUS outbreak in Germany. PLoS Currents Outbreaks, 1. <DOI: 10.1371/currents.outbreaks.f3fdeb08c5b9de7c09ed9cbcef5f01f2>

See Also

Other origin-est: origin_multiple

Examples

```
data(delayGoe)

# compute effective distance
data(ptnGoe)
goenet <- igraph::as_adjacency_matrix(ptnGoe, sparse=FALSE)
p <- goenet/rowSums(goenet)
eff <- eff_dist(p)
# apply effective distance median source estimation
om <- origin(events=delayGoe[10,-c(1:2)], type='edm', distance=eff)
summary(om)
plot(om, 'mdist',start=1)
plot(om, 'wvar',start=1)
performance(om, start=1, graph=ptnGoe)

# backtracking origin estimation (Manitz et al., 2016)
ob <- origin(events=delayGoe[10,-c(1:2)], type='backtracking', graph=ptnGoe)
summary(ob)
plot(ob, start=1)
performance(ob, start=1, graph=ptnGoe)

# centrality-based origin estimation (Comin et al., 2011)
oc <- origin(events=delayGoe[10,-c(1:2)], type='centrality', graph=ptnGoe)
summary(oc)
plot(oc, start=1)
performance(oc, start=1, graph=ptnGoe)
```

origin-methods *methods for origin estimation objects of class* origin

Description

print produces an output for objects of class origin.

summary produces an object summary for objects of class origin.

plot generates an illustration of an origin object using the variable to be optimized.

performance evaluates an object of class origin and returns a data.frame identifying correct estimation, and computing rank and distance of correct detection.

Usage

```
## S3 method for class 'origin'
print(x, ...)

## S3 method for class 'origin'
summary(object, x = object, ...)

## S3 method for class 'origin'
plot(x, y = "id", start, ...)

## S3 method for class 'origin'
performance(x, start, graph = NULL, ...)
```

Arguments

x	object of class origin, origin estimation object from function origin_xxx
...	further arguments to be passed to default plot function
object	object of class origin, origin estimation object from function origin_xxx; passed to x
y	character specifying the variable being plotted at the y-axis; options are 'id' for node identifier (default), 'mdist' for mean distance (only available for origin_edm) or 'wvar' for weighted variance (only available for origin_edm)
start	numeric, giving the node of the true origin
graph	igraph object specifying the underlying network graph with attribute 'length' on edges for calculation of distance to the correct origin

Value

performance.origin returns a data.frame with variables

- origin = start representing the true origin,
- est the estimated node of origin,

- hitt logical indicating whether origin estimation is correct or not,
- rank rank of correct detection,
- spj number of segments from estimated origin to true origin (requires an igraph object),
- dist distance along the shortest path from estimated origin to true origin (igraph edge attribute length)

See Also

origin plot_performance

Examples

```
data(ptnGoe)
data(delayGoe)

res <- origin(events=delayGoe[10,-c(1:2)], type='centrality', graph=ptnGoe)
res

summary(res)
plot(res, start=1)
performance(res, start=1, graph=ptnGoe)
```

---

origin_multiple *Multiple origin estimation using community partitioning*

Description

Multiple origin estimation using community partitioning

Usage

```
origin_multiple(events, type = c("edm", "backtracking", "centrality"), graph,
  no = 2, distance, fast = TRUE, ...)
```

Arguments

events	numeric vector of event counts at specific time point
type	character specifying the method, 'edm', 'backtracking' and 'centrality' are available.
graph	igraph object specifying the underlying network graph
no	numeric specifying the number of supposed origins
distance	numeric matrix specifying the distance matrix
fast	logical specifying community partitioning algorithm, default is 'TRUE' that uses fastgreedy.community, 'FALSE' refers to leading.eigenvector.community
...	parameters to be passed to origin methods origin_edm, origin_backtracking or origin_centrality

Value

`origin_multiple` returns an list object with objects of class `origin` of length `no`

References

Zang, W., Zhang, P., Zhou, C. and Guo, L. (2014) Discovering Multiple Diffusion Source Nodes in Social Networks. Procedia Computer Science, 29, 443-452. <DOI: 10.1016/j.procs.2014.05.040>

See Also

Other origin-est: `origin`

Examples

```
data(ptnAth)
# backtracking
origin_multiple(events=delayAth[10,-c(1:2)], type='backtracking', graph=ptnAth, no=2)
# edm
athnet <- igraph::as_adjacency_matrix(ptnAth, sparse=FALSE)
p <- athnet/rowSums(athnet)
eff <- eff_dist(p)
origin_multiple(events=delayAth[10,-c(1:2)], type='edm', graph=ptnAth, no=2, distance=eff)
```

---

performance *generic method for performance evaluation*

Description

generic method for performance evaluation

Usage

```
performance(x, ...)
```

Arguments

`x`	object
`...`	further arguments

See Also

`origin-methods` `plot_performance`

plot *generic method for plots*

Description

generic method for plots

Usage

```
plot(x, y, ...)
```

Arguments

x	object
y	object
...	further arguments

plot_performance *A plot method combining a time series of performance results.*

Description

A plot method combining a time series of performance results.

Usage

```
plot_performance(x, var = "rank", add = FALSE, offset = NULL,
  log = FALSE, col = 1, ylim = NULL, text.padding = 0.9, ...)
```

Arguments

x	data.frame obtained by combined results from performance.origin with variables X1 for time point, start for true origin, est for estimated origin, and performance variables
var	character, variable to be plotted, performance.origin returns rank, spj, and dist, default is 'rank'
add	logical, should be added to another performance plot
offset	POSIXct, starting time of spreading
log	logical, should y-axis be logarithmized?
col	numeric or character, color of lines
ylim	numeric vector, range of y axis
text.padding	a numeric value specifying the factor for the text position relative to the y values
...	further graphical parameters passed to default plot function

Examples

```
## Not run:
### delays on Goettingen bus network
# compute effective distance
data(ptnGoe)
goenet <- igraph::as_adjacency_matrix(ptnGoe, sparse=FALSE)
p <- goenet/rowSums(goenet)
eff <- eff_dist(p)
# apply source estimation
data(delayGoe)
if (requireNamespace("aplyr", quietly = TRUE)) {
   res <- alply(.data=delayGoe[11:20,-c(1:2)], .margins=1, .fun=origin_edm,
               distance=eff, silent=TRUE, .progress='text')
   perfGoe <- ldply(Map(performance, x = res, start = 2, list(graph = ptnGoe)))
   # performance plots
  plot_performance(perfGoe, var='rank', ylab='rank of correct detection', text.padding=0.5)
   plot_performance(perfGoe, var='dist', ylab='distance to correct detection')
}

### delays on Athens metro network
# compute effective distance
data(ptnAth)
athnet <- igraph::as_adjacency_matrix(ptnAth, sparse=FALSE)
p <- athnet/rowSums(athnet)
eff <- eff_dist(p)
# apply source estimation
data(delayAth)
if (requireNamespace("aplyr", quietly = TRUE)) {
   res <- alply(.data=delayAth[11:20,-c(1:2)], .margins=1, .fun=origin_edm,
             distance=eff, silent=TRUE, .progress='text')
   perfAth <- ldply(Map(performance, x = res, start = as.list(delayAth$k0),
                    list(graph = ptnAth)))
   # performance plots
  plot_performance(perfAth, var='rank', ylab='rank of correct detection',text.padding=0.5)
   plot_performance(perfAth, var='dist', ylab='distance to correct detection')
}

## End(Not run)
```

`plot_ptn` *A plot method for public transportation networks (PTNs).*

Description

A plot method for public transportation networks (PTNs).

Usage

```
plot_ptn(g, color.coding = NULL, color.scheme = rev(sequential_hcl(5)),
  legend = FALSE, ...)
```

Arguments

g	igraph object, network graph representing the public transportation network, vetrices represent stations, which are linked by an edge if there is a direct transfer between them
color.coding	numeric vector with length equal to the number of network nodes
color.scheme	character vector of length 5 indicating the vertex.color, default is rev(sequential_hcl(5))
legend	logical indicating whether legend for color-coding should be added or not.
...	further arguments to be passed to plot.igraph

See Also

Other network helper: analyze_ptn

Examples

```
data(ptnAth)
plot_ptn(ptnAth)

data(ptnGoe)
plot_ptn(ptnGoe)
```

---

ptn-data *Public transportation network datasets from LinTim software (Integrated Optimization in Public Transportation)*

Description

Public transportation network datasets from LinTim software (Integrated Optimization in Public Transportation)

ptnAth The data of the Athens Metro, consisting of 51 nodes and 52 edges.

- Vertex attributes: station name, additonal station info.
- Edge attributes: track length (in meter), minimal and maximal time required to pass the track (in minutes).

ptnGoe The data of the Goettingen bus network, consisting of 257 nodes and 548 edges.

- Vertex attributes: station name.
- Edge attributes: track length (in meter), minimal and maximal time required to pass the track (in minutes).

Author(s)

Juliane Manitz and Jonas Harbering

Source

Public transportation network datasets are extracted from LinTim software (Integrated Optimization in Public Transportation; http://lintim.math.uni-goettingen.de/index.php?go=data&lang=en). Special thanks to Anita Schoebel for making the data available.

The Athens Metro data was collected by Konstantinos Gkoumas.

The Goettingen bus network data was collected by Barbara Michalski.

See Also

delay-data

Examples

```
# Athens metro system
data(ptnAth)
analyze_ptn(ptnAth)
plot_ptn(ptnAth)

# Goettingen bus system
data(ptnGoe)
analyze_ptn(ptnGoe)
plot_ptn(ptnGoe)
```

---

read_DB_data *Reads a data file as provided by 'Deutsche Bahn' (for internal use).*

Description

Reads a data file as provided by 'Deutsche Bahn' (for internal use).

Usage

```
read_DB_data(file)
```

Arguments

file	character with path and file name containing the variables for 'stationID', 'date', 'hour', 'minutes', and 'delay'

Value

data.frame with variables 'node', 'time', 'delay'

See Also

Other data_handling: aggr_data

`robustness` *run robustness analysis for a source estimate by subsampling individual events.*

Description

run robustness analysis for a source estimate by subsampling individual events.

Usage

```
robustness(x, type = c("edm", "backtracking", "centrality"), prop, n = 100,
  ...)
```

Arguments

`x`	`data.frame`, dataset with individual events and their magnitude, to be passed to `aggr_data`
`type`	character, specifying the method, `'edm'`, `'backtracking'` and `'centrality'` are available.
`prop`	numeric, value between zero and one, proportion of events to be sampled
`n`	numeric, number of resamplings
`...`	parameters to be passed to origin methods `origin_edm`, `origin_backtracking` or `origin_centrality`

Details

We create subsamples of individual events and their magnitude using a sampling proportion p in [0, 1]. After aggregating the data, we apply the source estimation approach. Using this result, we deduce the relative frequency of how often the source estimate obtained with the complete data set can be recovered by source estimation based on the subsample. Thus, the estimate robustness is assessed by the proportion of estimate recovery.

Value

`data.frame` with columns

- `est` origin estimated when all data is evaluated
- `rob` estimate uncertainty, computed as the proportion of resamplings when origin estimate was recovered

See Also

`robustness-methods`

Examples

```
# generate random delay data
data(ptnAth)
require(igraph)
dat <- data.frame(node  = sample(size = 500, make.names(V(ptnAth)$name), replace = TRUE),
                  time  = sample(size = 500, 1:10, replace = TRUE),
                  delay = rexp(500, rate=10))
# compute effective distance
net <- igraph::as_adjacency_matrix(ptnAth, sparse=FALSE)
p <- net/rowSums(net)
eff <- eff_dist(p)
colnames(eff) <- paste('x.',colnames(eff),sep='')

# run robustness analysis
r5 <- robustness(x=dat, type='edm', prop=0.5, n=10, distance=eff)
summary(r5)
plot(r5)

# compare results
r9 <- robustness(x=dat, type='edm', prop=0.9, n=10, distance=eff)
plot(r9, add=TRUE, col='gray')
```

---

robustness-methods *methods for robustness estimation objects of class* robustness

---

Description

`print` produces an output for objects of class `robustness`

`summary` produces an object summary for objects of class `robustness`

`plot` produces a time series plot of the `robustness` estimate object

Usage

```
## S3 method for class 'robustness'
print(x, ...)

## S3 method for class 'robustness'
summary(object, x = object, ...)

## S3 method for class 'robustness'
plot(x, y = NULL, add = FALSE, ...)
```

Arguments

`x`	`data.frame` obtained by `robustness`, robustness estimation object for source estimation from function `robustness`

...	further arguments passed to the default print method
object	object of class origin, origin estimation object from function origin_xxx; passed to x
y	not used; default NULL
add	logical specifying whether this should be added to another robustness plot

See Also

robustness

---

var_wtd_mean_cochran *Computes the variance of a weighted mean following the definition by Cochran (1977; see Gatz and Smith, 1995)*

Description

This is a helper method for weighted variance computation in origin_edm, which is the closest to the bootstrap.

Usage

```
var_wtd_mean_cochran(x, w)
```

Arguments

x	numeric vector of values
w	numeric vector of weights

Value

numeric value of weighted variance

References

- Gatz, D. F., and Smith, L. (1995). The standard error of a weighted mean concentration-I. Bootstrapping vs other methods. Atmospheric Environment, 29(11), 1185-1193. <DOI: 10.1016/1352-2310(94)00210-C>
- Gatz, D. F., and Smith, L. (1995). The standard error of a weighted mean concentration-II. Estimating confidence intervals. Atmospheric Environment, 29(11), 1195-1200. <DOI: 10.1016/1352-2310(94)00209-4>
- http://r.789695.n4.nabble.com/Problem-with-Weighted-Variance-in-Hmisc-td826437.html

Index