

# Package ‘egor’

February 1, 2019

**Type** Package

**Title** Import and Analyse Ego-Centered Network Data

**Version** 0.19.1

**Date** 2019-01-31

**Description** Tools for importing, analyzing and visualizing ego-centered network data. Supports several data formats, including the export formats of 'EgoNet', 'EgoWeb 2.0' and 'openeddi'. An interactive (shiny) app for the intuitive visualization of ego-centered networks is provided. Also included are procedures for creating and visualizing Clustered Graphs (Lerner 2008 <DOI:10.1109/PACIFICVIS.2008.4475458>).

**URL** <https://github.com/tilltnet/egor>, <https://tilltnet.github.io/egor/>

**BugReports** <https://github.com/tilltnet/egor/issues>

**License** AGPL-3

**Depends** R (>= 2.10), dplyr, tibble

**Imports** igraph, network, shiny, plyr, survey, tidyr, methods, utils,  
purrr

**Suggests** knitr, testthat, rmarkdown

**VignetteBuilder** knitr

**RoxygenNote** 6.1.1.9000

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Till Krenz [aut, cre],  
Pavel N. Krivitsky [aut],  
Raffaele Vacca [aut],  
Michal Bojanowski [aut],  
Markus Gamper [ctb],  
Andreas Herz [aut],  
Christopher McCarty [ctb]

**Maintainer** Till Krenz <egor@tillt.net>

**Repository** CRAN

**Date/Publication** 2019-02-01 16:43:22 UTC

**R topics documented:**

alters32	2
alter_design	3
alts_diversity_count	4
as_alts_df	5
clustered_graphs	6
composition	7
comp_ei	8
comp_ply	8
convert_egor	9
edges32	10
egor	11
egor-package-doc	12
egor32	13
egor_vis_app	13
egos32	14
ego_density	14
ego_design	15
EI	16
gss2004	17
helper	18
make_egor	19
onefile_to_egor	19
read_egonet	21
rowlist	22
subset.egor	23
summary.egor	25
threefiles_to_egor	25
trim_aaties	27
twofiles_to_egor	28
vert.attr	29
vert.attr.multi	30
vis_clustered_graphs	31
weights.egor	32
<b>Index</b>	<b>33</b>

---

alters32	<i>32 sets of randomly created alters belonging to ego-centered networks</i>
----------	--

---

**Description**

32 sets of randomly created alters belonging to ego-centered networks

**Usage**

alters32

**Format**

A data frame with 32 sets of up to 32 alters per egoID and 7 variables:

**egoID** ego identifier

**ego.sex** ego's gender

**age** ego's age

**netsize** size of ego's network

**alterID** alter identifier

**alter.sex** alter's gender

**alter.age** alter's age

---

alter_design	<i>Set and query the alter nomination design</i>
--------------	--

---

**Description**

Extract, set, or update the alter nomination design associated with an ego-centered dataset.

**Usage**

```
alter_design(x, ...)

## S3 method for class 'egor'
alter_design(x, which, ...)

alter_design(x, ...) <- value

## S3 replacement method for class 'egor'
alter_design(x, which, ...) <- value
```

**Arguments**

x	an <a href="#">egor</a> object.
...	arguments to be passed to methods
which	name of the alter design setting to query or replace
value	if which is specified, the new value of the attribute; if not, a named list of settings that replace their old values.

---

alts\_diversity\_count *Calculate diversity measures on an egor object.*

---

### Description

alts\_diversity\_count() counts the categories of a variable present in the networks of an egor object. alts\_diversity\_entropy() calculates the Shannon entropy as a measurement for diversity of an alter attribute.

### Usage

```
alts_diversity_count(object, alt.attr)
```

```
alts_diversity_entropy(object, alt.attr, base)
```

### Arguments

object	An egor object.
alt.attr	A character naming the variable containing the alter-attribute.
base	Numeric, base value of logarithm for entropy calculation.

### Value

A numeric vector.

### Author(s)

Michał Bojanowski, <m.bojanowski@uw.edu.pl>

Till Krenz, <public@tillt.net>

### Examples

```
data("egor32")
alts_diversity_count(egor32, "age")
alts_diversity_entropy(egor32, "age")
```

---

as_alts_df	<i>Create global alters and alter-alter relations dataframes from an egor object</i>
------------	--

---

## Description

Provided an egor-object, these functions create a 'global' data.frame, containing alter attributes, or alter-alter relations. The resulting dataframes are useful for advanced analysis procedures, e.g. multi-level regressions.

## Usage

```
as_alts_df(object, egoID = "egoID", include.ego.vars = FALSE)

as_aaties_df(object, egoID = "egoID", include.ego.vars = FALSE,
  include.alt.vars = FALSE, aatie_vars = c(".srcID", ".tgtID"))
```

## Arguments

object	An egor object.
egoID	Character, naming the variable used to identify egos. If this variable does not exist, a new variable with the specified name is created.
include.ego.vars	Logical, specifying if ego variables should be included in the result.
include.alt.vars	Logical, specifying if alter variables should be included in the result.
aatie_vars	Character vector, specifying the names of the source and target columns.

## Examples

```
# Load example data
data(egor32)

# Create global alters dataframes
as_alts_df(egor32)

# Create global alter-alter relations dataframes
as_aaties_df(egor32)

# ... adding alter variables
as_aaties_df(egor32, include.alt.vars = TRUE)
```

---

clustered_graphs	<i>Cluster ego-centered networks by a grouping factor</i>
------------------	---

---

### Description

The idea of clustered graphs is to reduce the complexity of an ego-centered network graph by visualizing alters in clusters defined by a categorical variable (Lerner et al. 2008). `clustered_graphs()` calculates group sizes, inter and intra group tie densities and returns these informations in a list of `igraph` objects.

### Usage

```
clustered_graphs(object, ..., clust.groups)

## S3 method for class 'list'
clustered_graphs(object, aaties, clust.groups, ...)

## S3 method for class 'egor'
clustered_graphs(object, clust.groups, ...)

## S3 method for class 'data.frame'
clustered_graphs(object, aaties, clust.groups,
  egoID = "egoID", ...)
```

### Arguments

<code>object</code>	Either an <code>egor</code> object or a <code>data.frame/ list</code> containing alter attributes.
<code>...</code>	arguments to be passed to methods
<code>clust.groups</code>	A character naming the factor variable defining the groups.
<code>aaties</code>	<code>data.frame/ list</code> containing alter-alter relations as a 'global edge list' or as a list of 'edge lists'. (not needed if <code>object</code> is an <code>egor</code> object).
<code>egoID</code>	Character. Name of the variable identifying egos (default: "egoID").

### Value

`clustered_graphs` returns a list of graph objects representing the clustered ego-centered network data;

### References

Brandes, U., Lerner, J., Lubbers, M. J., McCarty, C., & Molina, J. L. (2008). Visual Statistics for Collections of Clustered Graphs. 2008 IEEE Pacific Visualization Symposium, 47-54.

### See Also

[vis\\_clustered\\_graphs](#) for visualizing clustered graphs

## Examples

```
data("egor32")

# Simplify networks to clustered graphs, stored as igraph objects
graphs <- clustered_graphs(egor32, "country")

# Visualise
vis_clustered_graphs(graphs,
                      node.size.multiplier = 5,
                      edge.width.multiplier = 25,
                      labels = TRUE)
```

---

composition

*Calculate the composition of alter attributes in an egor object*

---

## Description

`composition()` calculates the proportional or absolute composition of alters for a given attribute/variable.

## Usage

```
composition(object, alt.attr, absolute = FALSE)
```

## Arguments

<code>object</code>	An egor object.
<code>alt.attr</code>	A character naming the variable containing the alter-attribute.
<code>absolute</code>	Logical indicating if the results should be absolute.

## Value

A tibble with the values per category in the columns.

## Examples

```
data("egor32")
composition(egor32, "sex")
```

---

comp_ei	<i>Calculate the EI-Indices of an egor object as a measurement of ego-alter homophily</i>
---------	---

---

**Description**

comp\_ei() calculates the EI-Index values as a measurement for ego-alter homophily.

**Usage**

```
comp_ei(object, alt.attr, ego.attr)
```

**Arguments**

object	An egor object.
alt.attr	A character naming the variable containing the alter-attribute.
ego.attr	A character naming an ego attribute.

**Value**

A numeric vector.

**Examples**

```
data("egor32")
comp_ei(egor32, "age", "age")
```

---

comp_ply	<i>Calculate third-party compositional measures on an egor object</i>
----------	---

---

**Description**

comp\_ply() applies a function, that uses an alter attribute to calculate a compositional measurement, on all networks in an egor object and returns a numeric vector.

**Usage**

```
comp_ply(object, alt.attr, .f, ..., ego.attr = NULL)
```

**Arguments**

object	An egor object.
alt.attr	A character naming the variable containing the alter-attribute.
.f	A function that returns a numeric.
...	Optional arguments to .f.
ego.attr	Optional character naming an ego attribute.



## Details

When an ego attribute is used the `.f` is called like this: `.f(alt.attr, ego.attr, ...)`. `.f` must return a single numeric value.

## Value

A numeric vector.

## Author(s)

Michał Bojanowski, <m.bojanowski@uw.edu.pl>

Till Krenz, <public@tillt.net>

## Examples

```
df <- make_egor(10, 32)
comp_ply(df, "age.years", sd, na.rm = TRUE)
```

---

convert\_egor

*Convert egor object to network or igraph objects*

---

## Description

These functions convert an egor object into a list of network or igraph objects. By default ego itself is not included in the created objects, there is a parameter (**include.egor**) that allows for including ego.

## Usage

```
as_igraph(x, directed = FALSE, include.ego = FALSE, ego.attrs = NULL,
  ego.alter.weights = NULL)
```

```
## S3 method for class 'egor'
as.igraph(x, directed = FALSE, include.ego = FALSE,
  ego.attrs = NULL, ego.alter.weights = NULL)
```

```
as_network(x, directed = FALSE, include.ego = FALSE,
  ego.attrs = NULL, ego.alter.weights = NULL)
```

```
## S3 method for class 'egor'
as.network(x, directed = FALSE, include.ego = FALSE,
  ego.attrs = NULL, ego.alter.weights = NULL)
```

**Arguments**

<code>x</code>	An egor object.
<code>directed</code>	Logical, indicating if alter-alter relations are directed.
<code>include.ego</code>	Logical. Should ego be included?
<code>ego.attrs</code>	Vector of names (character) or indices (numeric) of ego variables that should be carried over to the network/ igraph objects.
<code>ego.alter.weights</code>	Vector of names (character) or indices (numeric) of alter variables that should be carried over to the the network/ igraph objects, as edge attributes of the ego-alter relations.

**Details**

The names of the variables specified in `ego.attr` and `ego.alter.attr` need to be the same as the names of corresponding alter attributes, in order for those variables to be complete in the resulting network/ igraph object (see example).

**Functions**

- `as_igraph`: Creates a list of igraph objects from an egor object.
- `as_network`: Creates a list of statnet's network objects, from an egor object.

---

edges32

*32 sets of randomly created alter-alter ties belonging to ego-centered networks*


---

**Description**

32 sets of randomly created alter-alter ties belonging to ego-centered networks

**Usage**

```
edges32
```

**Format**

A data frame with 32 sets of alter-alter relations and 4 variables:

**egoID** ego identifier

**Source** source alter ID

**Target** target alter ID

**weight** weight of relation

---

 egor

*egor - a data class for ego-centered network data.*


---

## Description

The function `egor()` is used to create an egor object from ego-centered network data.

## Usage

```
egor(alters.df, egos.df = NULL, aaties.df = NULL, ID.vars = list(ego
  = "egoID", alter = "alterID", source = "Source", target = "Target"),
  ego_design = list(~1), alter_design = list(max = Inf))
```

```
as.egor(x, ...)
```

## Arguments

<code>alters.df</code>	either a <code>data.frame</code> containing the alters (whose nominator is identified by the column specified by <code>egoID</code> or a list of data frames with the same columns, one for each ego, with empty data frames or <code>NULL</code> s corresponding to egos with no nominees.
<code>egos.df</code>	<code>data.frame</code> containing the egos.
<code>aaties.df</code>	<code>data.frame</code> containing the alter-alter relations in the style of an edge list, or a list of data frames similar to <code>alters.df</code> .
<code>ID.vars</code>	A named list containing column names of the relevant input columns: <ul style="list-style-type: none"> <li><code>ego</code> unique identifier associated with each ego, defaulting to <code>"egoID"</code>; has no effect if <code>alters.df</code> and <code>aaties.df</code> are both lists of data frames.</li> <li><code>alter</code> unique-within-ego identifier associated with each alter, defaulting to <code>"alterID"</code>; optional <code>aaties.df</code> are not provided.</li> <li><code>source</code> if <code>aaties.df</code> is provided, the column given the alter identifier of the origin of a relation.</li> <li><code>target</code> if <code>aaties.df</code> is provided, the column given the alter identifier of the destination of a relation.</li> </ul>
<code>ego_design</code>	A <code>list</code> of arguments to <code>survey::svydesign()</code> specifying the sampling design for the egos. If formulas, they can refer to columns of <code>egos.df</code> .
<code>alter_design</code>	A <code>list</code> of arguments specifying nomination information. Currently, the following elements are supported: <ul style="list-style-type: none"> <li><code>"max"</code> Maximum number of alters that an ego can nominate.</li> </ul>
<code>x</code>	an object to be coerced to <code>egor</code> .
<code>...</code>	arguments to be passed to methods

## Details

If parameters `alters.df`, `egos.df`, and `aaties.df` are data frames, they need to share a common ego ID variable, with corresponding values. If `alters.df` and `aaties.df` are lists of data frames, `egoID` is ignored and they are matched positionally with the rows of `egos.df`. Of the three parameters only `alters.df` is necessary to create an `egor` object, and `egos.df` and `aaties.df` are optional.

## Value

Returns an `egor` object. An `egor` object is a `tibble` whose top-level columns store the ego attributes, and which has two special nested columns: `.alts`, containing, for each row (ego) a table of that ego's alter attributes and `.aaties`, a table containing that ego's alter–alter ties, if observed.

If alter–alter ties are observed, `.alts` also has a column `.altID` giving a unique (within each ego) ID of the alter, by which the alter can be identified in the `.aaties` table for that ego. `.aaties`, in turn, has columns `.srcID` and `.tgtID` that contain the source and the target of the alter–alter relation.

In addition, `egor` has two attributes: `ego_design`, containing an object returned by `survey::svydesign()` specifying the sampling design by which the egos were selected and `alter_design`, a `list` containing specification of how the alters were nominated. See the argument above for currently implemented settings.

## Note

Column names `.alts`, `.aaties`, and `.egoRow` are reserved for internal use of `egor` and should not be used to store persistent data. Other `.-led` column names may be reserved in the future.

## Examples

```
data("edges32")
data("egos32")
data("alters32")

egor(alters.df = alters32,
     egos.df = egos32,
     aaties = edges32)
```

---

egor-package-doc	egor
------------------	------

---

## Description

R Package for importing and analyzing ego-centered-network data.

## Details

[Further Information](#) or [GitHub](#)

Thanks to: Martina Morris, Michał Bojanowski

**Author(s)**

Till Krenz, <egor@tillt.net>  
 Pavel Krivitsky, <pavel@uow.edu.au>  
 Raffaele Vacca, <r.vacca@ufl.edu>  
 Andreas Herz, <herzand@uni-hildesheim.de>  
 Christopher McCarty, <ufchris@ufl.edu>  
 Markus Gamper, <m.gamper@uni-koeln.de>

---

 egor32

---

*32 randomly created ego-centered networks stored as an egor object*


---

**Description**

32 randomly created ego-centered networks stored as an egor object

**Usage**

egor32

**Format**

An egor object with 32 ego-centered networks (5 variables):

**egoID** ego identifier  
**sex** ego's gender  
**age** ego's age  
**.alts** nested column/list containing alters  
**.aaties** nested column/list containing alter-alter relations

---

 egor\_vis\_app

---

*egor Network Visualization App*


---

**Description**

Launches an interactive Shiny Web App, that creates a list of igraph objects from an 'egor' object and offers the user several graphical means of interacting with the visualization parameters for all networks in the egor object.

**Usage**

```
egor_vis_app(object, shiny_opts = list(launch.browser = TRUE))
```

**Arguments**

**object** An egor object.  
**shiny\_opts** List of arguments to be passed to shinyApp()'s options argument.

**Examples**

```
if(interactive()){
  data("egor32")
  egor_vis_app(egor32)
}
```

---

egos32 *32 randomly created egos belonging to ego-centered networks*

---

**Description**

32 randomly created egos belonging to ego-centered networks

**Usage**

```
egos32
```

**Format**

A data frame with 32 sets of alter-alter relations and 4 variables:

**egoID** ego identifier

**sex** ego's gender

**age** ego's age

**netsize** size of ego's network

---

ego\_density *Calculate the relationship density in ego-centered networks*

---

**Description**

This function uses an egor object and calculates the density of all the ego-centered networks listed in the 'egor' object. Instead of an egor object, alter and alter-alter data can be provided as lists or data.frames.

**Usage**

```
ego_density(object, ...)

## S3 method for class 'list'
ego_density(object, aaties, weight = NULL,
            max.netsize = NULL, directed = FALSE, ...)

## S3 method for class 'egor'
ego_density(object, weight = NULL, max.netsize = NULL,
            directed = FALSE, ...)

## S3 method for class 'data.frame'
ego_density(object, aaties, egoID = "egoID",
            weight = NULL, max.netsize = NULL, directed = FALSE, ...)
```

**Arguments**

object	Either an egor object or a data.frame/ list containing alter attributes.
...	arguments to be passed to methods
aaties	data.frame/ list containing alter-alter relations as a 'global edge list' or as a list of 'edge lists'. (not needed if object is an egor object).
weight	Character naming a variable containing the weight values of relations. Weights should range from 0 to 1.
max.netsize	Optional parameter. Constant value used if the number of alters whose relations were collected is limited.
directed	logical indicating if the alter-alter relation data/ edges are directed or undirected.
egoID	Name of ego ID variable. Only needs to be specified if alter and alter-alter data is provided in global data.frames.

**Value**

returns a vector of network density values.

**Examples**

```
data("egor32")
ego_density(egor32)
```

---

ego\_design

*Set and query the ego sampling design*

---

**Description**

Extract, set, or update the [svydesign](#) associated with an ego-centered dataset.

**Usage**

```
ego_design(x, ...)

## S3 method for class 'egor'
ego_design(x, ...)

ego_design(x, ...) <- value

## S3 replacement method for class 'egor'
ego_design(x, ...) <- value
```

**Arguments**

x	an <a href="#">egor</a> object.
...	arguments to be passed to methods
value	either <code>survey.design</code> object (like one constructed by <a href="#">svydesign()</a> ) or a <a href="#">list</a> of arguments to <a href="#">svydesign()</a> specifying the sampling design for the egos. If the arguments are formulas, they can refer to columns (ego attributes) of x.

**Note**

This can be useful for adjusting or reinitializing the ego design information after the underlying ego attributes had been modified.

---

 EI

*Calculate the EI-Index for the alter-alter ties of an ego object*

---

**Description**

The EI-Index is the division of the intra-group edge density and the outer-group edge density. It is calculated for the whole network and for subgroups. The whole network EI is a metric indicating the tendency of a network to be clustered by the categories of a given factor variable. The EI value of a groups describes the tendency of a group to be connected or not connected to other groups. Additionally, the EI index can be employed as a measurement for egos tendency to homo-/heterophily - use the `comp_ei()` command for that version of EI-Index.

**Usage**

```
EI(object, alt.attr)
```

**Arguments**

object	An <code>egor</code> object.
alt.attr	Character naming grouping variable.



## References

- Krackhardt, D., Stern, R.N., 1988. Informal networks and organizational crises: an experimental simulation. *Social Psychology Quarterly* 51 (2), 123-140.
- Everett, M. G., & Borgatti, S. P. (2012). Categorical attribute based centrality: E-I and G-F centrality. *Social Networks*, 34(4), 562-569.

## Examples

```
data("egor32")
EI(egor32, "sex")
```

---

gss2004

*A selective subset of GSS 2004 data*

---

## Description

This is a selective subset of General Social Survey 2004 data containing variables from network questions. See Details for description how this particular subset was selected. The data has a near 0 research value, it is provided to illustrate the functions in **egor** package.

## Format

A tibble with 499 rows and the variables listed below. Data was imported from SPSS file (with `haven::read_sav()`) and are labelled. Functions in the **labelled** package can be used to handle them.

Variables:

**id** Case ID

**vpsu, vstrat, wtssall** Design variables and weight

**age** Ego's age in years

**race** Ego's race. 1=white, 2=black, 3=other

**sex** Ego's sex. 1=male, 2=female

**marital** Ego's marital status. 1=married, 2=widowed, 3=divorced, 4=separated, 5=never married

**numgiven** Number of alters mentioned

**age[1-5 ]** Alter's age in years

**race[1-5 ]** Alter's race. 1=asian, 2=black, 3=hispanic, 4=white, 5=other

**sex[1-5 ]** Alter's sex. 1=male, 2=female

**spouse[1-5 ]** Whether alter is a spouse of ego. 1=mentioned, 2=not mentioned

**close[1-4 [2-5]]** How close are the two alters according to ego. 1=especially close, 2=know each other, 3=total strangers

**Details**

This dataset was created from original GSS 2004 data for illustrative purposes such that (1) it is small and (2) contains just enough variation in respondent's personal networks to illustrate various functions in the package. It is essentially a stratified sample from original data (1472 cases). Strata correspond to groups of cases created from unique combinations of values on the following ego variables: age (3 categories), race, sex, marital, numgiven. At most 2 cases were sampled from each stratum via simple random sampling with replacement.

**Source**

General Social Survey data at NORC: <http://gss.norc.org/get-the-data>

---

helper	<i>General helper functions</i>
--------	---------------------------------

---

**Description**

Helper functions for ego centered network analysis

**Usage**

```
dyad.poss(max.alters, directed = FALSE)
```

```
sanitize.wide.edges(max.alters)
```

```
create_edge_names_wide(x)
```

```
dyads_possible_between_groups(x, y)
```

```
din_page_dist(x)
```

**Arguments**

<code>max.alters</code>	A numeric giving the maximum number of alters.
<code>directed</code>	A logical value indicating directedness of alter-alter data.
<code>x</code>	Numeric.
<code>y</code>	Numeric.

**Functions**

- `dyad.poss`: Returns the count of possible edges in an undirected or directed, ego-centered network, based on the number of alters.
- `sanitize.wide.edges`: Generates a `data.frame` marking possible dyads in a wide alter-alter relation `data.frame`. Row names corresponds to the network size. This is useful for sanitizing alter-alter relations in the wide format.

- `create_edge_names_wide`: Creates a vector of names for variables containing data on alter-alter relations/ dyads in ego-centered networks.
- `dyads_possible_between_groups`: Calculates the possible edges between members of different groups in an ego-centered network.
- `din_page_dist`: Calculates the optimal distribution of a number of equally sized objects on a DIN-Norm DIN 476 (i.e. DIN A4) page in landscape view.

---

 make\_ego

*Generate random ego-centered-network data.*


---

### Description

This function generates random ego-centered-network data for a specified number of networks with a maximum network size. The network size of the generated networks is a normal distribution with `sd=5`.

### Usage

```
make_ego(net.count, max.alters, net.size = NULL, plot = FALSE)
```

### Arguments

<code>net.count</code>	Number of networks/ egos to generate.
<code>max.alters</code>	Maximum size of networks.
<code>net.size</code>	Numeric for fixed network sizes.
<code>plot</code>	whether to plot the network size distribution.

---

 onefile\_to\_ego

*Import ego-centered network data from 'one file format'*


---

### Description

This function imports ego-centered network data, stored in a single file, providing ego, alter and edge data. This data format is used by the Allbus 2010 (GESIS) and similar social surveys.

### Usage

```
onefile_to_ego(egos, net.size, ID.vars = list(ego = "egoID"),
  attr.start.col, attr.end.col, max.alters, aa.first.var,
  aa.regex = NULL, ego.vars = NULL, var.wise = FALSE, ...)
```

**Arguments**

<code>egos</code>	Data frame containing ego data (egos as cases)
<code>netsize</code>	Vector containing values of network size per ego.
<code>ID.vars</code>	A named list containing column names of the relevant input columns: <ul style="list-style-type: none"> <li><code>ego</code> unique identifier associated with each ego, defaulting to "egoID"; has no effect if <code>alters.df</code> and <code>aaties.df</code> are both lists of data frames.</li> <li><code>alter</code> unique-within-ego identifier associated with each alter, defaulting to "alterID"; optional <code>aaties.df</code> are not provided.</li> <li><code>source</code> if <code>aaties.df</code> is provided, the column given the alter identifier of the origin of a relation.</li> <li><code>target</code> if <code>aaties.df</code> is provided, the column given the alter identifier of the destination of a relation.</li> </ul>
<code>attr.start.col</code>	Index or name of the first column containing alter attributes.
<code>attr.end.col</code>	Index or name of the last column containing alter attributes.
<code>max.alters</code>	Maximum number of alters.
<code>aa.first.var</code>	First column containing alter-alter relations/ edges.
<code>aa.regex</code>	A Perl regular expression with name capture, intended to be run on column names and capturing via named capture the following regex groups: "attr", "src", and "tgt", representing the edge attribute being captured, the source (or the first alter identified), and the target (or the second alter identified) of the edge, respectively. See <a href="#">regex</a> for more information.
<code>ego.vars</code>	A data frame of alter attributes in the wide format.
<code>var.wise</code>	Logical value indicating if the alter attributes are sorted variable wise (defaults to FALSE).
<code>...</code>	additional arguments to <code>egor()</code> .

**Value**

A list of six objects - the **egoR** object: (1) `egos.df`: dataframe of all egos and their attributes; (2) `alters.df`: dataframe of all alters; (3) `alters.list`: list of dataframes of all alters per ego; (4) `edges`: list of dataframes with edge lists per network; (5) `graphs`: list of igraph objects; (6) `results`: a result dataframe, pre-populated with the network size of each network

**References**

Muller, C., Wellman, B., & Marin, A. (1999). How to Use SPSS to Study Ego-Centered Networks. *Bulletin de Methodologie Sociologique*, 64(1), 83-100.

---

read_egonet	<i>Read ego-centered network data exported with EgoNet software as an egor object</i>
-------------	---

---

### Description

This function imports ego-centered network data from folders with separate files for alters-level and edge data. It will run some basic checks upon the completeness of the data and inform the user of potential problems. This function can be used to import data exported from EgoNet (McCarty 2011).

### Usage

```
read_egonet(egos.file, alter.folder, edge.folder, csv.sep = ",",
  ID.vars = list(ego = "egoID", alter = "alterID", source = "Source",
  target = "Target"), first.col.row.names = FALSE, ...)
```

### Arguments

egos.file	File name of the .csv file containing the ego data.
alter.folder	Folder name of the folder containing the alter data in separate .csv files for each ego/ network.
edge.folder	Folder name of the folder containing the edge/ tie data in separate .csv files for each ego/ network.
csv.sep	Character indicating the separator used in csv files.
ID.vars	A named list containing column names of the relevant input columns: ego unique identifier associated with each ego, defaulting to "egoID"; has no effect if alters.df and aaties.df are both lists of data frames. alter unique-within-ego identifier associated with each alter, defaulting to "alterID"; optional aaties.df are not provided. source if aaties.df is provided, the column given the alter identifier of the origin of a relation. target if aaties.df is provided, the column given the alter identifier of the destination of a relation.
first.col.row.names	Boolean indicating if first column contains row names, that are to be skipped, default is FALSE.
...	additional arguments to <code>egor()</code> .

### Value

A list of six objects - the **egoR** object: (1) egos.df: dataframe of all egos and their attributes; (2) alters.df: dataframe of all alters; (3) alters.list: list of dataframes of all alters per ego; (4) edges: list of dataframes with edge lists per network; (5) graphs: list of igraph objects; (6) results: a result dataframe, pre-populated with the network size of each network

## Examples

```
egos.file <- system.file("extdata", "egos_32.csv", package = "egor")
alters.folder <- system.file("extdata", "alters_32", package = "egor")
edge.folder <- system.file("extdata", "edges_32", package = "egor")

ef <- read_egonet(egos.file = egos.file,
                 alter.folder = alters.folder,
                 edge.folder = edge.folder,
                 csv.sep = ";")
```

---

rowlist

*Convert a table to a list of rows*

---

## Description

A convenience function converting a `data.frame()` or a `tibble()`.

## Usage

```
rowlist(x)
```

## Arguments

`x` a `data.frame()`, a `tibble()`, or some other table data structure backed by a `list()` of columns.

## Value

A `list()` of length `nrow(x)`, with each element itself a named `list()` containing the elements in the corresponding row.

## Examples

```
library(tibble)
(df <- tibble(x=2:1, y=list(list(1:3), list(3:4))))
rowlist(df)
```

**Description**

Functions to index and take subsets of `egor()` objects: manipulate egos, alters, or alter-alter ties.

**Usage**

```
## S3 method for class 'egor'
subset(x, subset, ..., unit = c("ego", "alter", "aatie"))

## S3 method for class 'egor'
x[i, j, unit = c("ego", "alter", "aatie"), ...]
```

**Arguments**

`x` an `egor()` object.

`subset` either an expression evaluated on each of the rows of `egor()` (as in the eponymous argument of `subset()`) or a function whose first argument is a row, specifying which egos, alters, or alter-alter ties to keep; output format depends on `unit`:

- "ego" a single logical value specifying whether the ego should be kept.
- "alter" either an integer vector of indices specifying which alters to select or a logical vector of length `nrow(x)` specifying which alters should be kept.
- "aatie" either an integer vector of indices specifying which alter-alter ties to select or a logical vector of length `nrow(x)` specifying which alter-alter ties should be kept.

the expressions can access variables in the calling environment; columns of `egor()` as variables (which mask the variables in the calling environment), as well as the following "virtual" columns to simplify indexing:

- Ego index** `.egoRow` contains the index (counting from 1) of the row being evaluated. (This can be used to access vector variables in the calling environment.)
- Alter index** `.altRow` contains the index (counting from 1) of the row number in the alter table.
- Alter-alter indices** `.srcRow` **and** `.tgtRow` contain the index (counting from 1) of the row of the alter being refereced by `.srcID` and `.tgtID`. (This can be used to quickly access the attributes of the alters in question.)

`...` extra arguments to `subset` if `subset` is a function; otherwise unused.

`unit` a selector of the unit of analysis being affected: the egos, the alters or the (alter-alter) ties. Note that only one type of unit can be affected at a time.

`i` depends on `unit`:

"ego" either an integer vector of indices specifying which egos to select or a logical vector of length `nrow(x)` specifying which rows should be kept; a logical list of length `nrow(x)` is acceptable as well.

"alter" a ragged array (a `list()`) of length `nrow(x)`, either of integer vectors of indices specifying which alters to select for the corresponding ego or of logical vectors of length `nrow(x$.alts[k, , drop=FALSE])` specifying which alters should be kept.

"aatie" a ragged array (a `list()`) of length `nrow(x)`, either of integer vectors of indices specifying which alter-alter ties to select for the corresponding ego or of logical vectors of length `nrow(x$.aaties[k, , drop=FALSE])` specifying which ties should be kept

In general, constructing selection arrays for alters and ties is complicated and error-prone, so the use of `subset()` is recommended.

`j` either an integer vector specifying which columns of the filtered structure (ego, alters, or ties) to select, or a logical vector specifying which columns to keep.

## Details

Removing or duplicating an ego will also remove or duplicate their alters and ties.

## Value

An `egor()` object.

## Examples

```
# Generate a small sample dataset
(e <- make_egor(5,4))

# First three egos in the dataset
e[1:3,]

# Similarly with subset()
subset(e, .egoRow <= 3)

# Using an external vector
# (though normally, we would use e[.keep,] here)
.keep <- rep(c(TRUE, FALSE), length.out=nrow(e))
subset(e, .keep[.egoRow])
# a more robust version of the above: pass a function of row and
# keep (which is passed as an additional argument to the function):
subset(e, function(r, keep) keep[r$.egoRow], .keep)

# Only keep egos with exactly three alters
subset(e, nrow(.alts)==3)

# Only keep egos with exactly two female alters
subset(e, sum(.alts$sex=="w")==2)
```



```
# Only keep female alters
subset(e, .alts$sex=="w", unit="alter")

# Only keep alters of a different sex form ego
subset(e, sex != .alts$sex, unit="alter")

# Only keep homophilous alter-alter ties
subset(e, .alts$sex[.aaties$.srcRow] ==
        .alts$sex[.aaties$.tgtRow],
        unit="aatie")
```

---

summary.egor

*Methods to print and summarize [egor](#) objects*


---

## Description

Methods to print and summarize [egor](#) objects

## Usage

```
## S3 method for class 'egor'
summary(object, ...)

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

## Arguments

object, x      an [egor](#) object.  
...              additional arguments, either unused or passed to lower-level functions.

---

threefiles\_to\_egor

*Read/ import ego-centered network data from the three files format, EgoWeb2.0 or openeddi.*


---

## Description

These functions read ego-centered network data from the three files format, EgoWeb2.0 or openeddi and transform it to an egoR object. The three files format consists of an ego file, on alters file and one file containing the edge data. EgoWeb2.0 and openeddi use variations of this format.

**Usage**

```
threefiles_to_egor(egos, alters.df, edges, ID.vars = list(ego = "egoID",
  alter = "alterID", source = "Source", target = "Target"),
  ego.vars = NULL, ...)

read_egoweb(alter.file, edges.file, egos.file = NULL,
  ID.vars = list(ego = "EgoID", alter = "Alter.Number", source =
  "Alter.1.Number", target = "Alter.2.Number"), ego.vars = NULL, ...)

read_openeddi(egos.file = NULL, alters.file = NULL,
  edges.file = NULL, ID.vars = list(ego = "puid", alter = "nameid",
  source = "nameid", target = "targetid"), ego.vars = NULL, ...)
```

**Arguments**

egos	Data frame containing ego data (egos as cases)
alters.df	dataframe containing alters data (alters as cases), alters are separated by a variable containing an egoID.
edges	Dataframe. A global edge list, first column is ego ID variable. egos.
ID.vars	A named list containing column names of the relevant input columns: ego unique identifier associated with each ego, defaulting to "egoID"; has no effect if alters.df and aaties.df are both lists of data frames. alter unique-within-ego identifier associated with each alter, defaulting to "alterID"; optional aaties.df are not provided. source if aaties.df is provided, the column given the alter identifier of the origin of a relation. target if aaties.df is provided, the column given the alter identifier of the destination of a relation.
ego.vars	A data.frame of alter attributes in the wide format.
...	additional arguments to <code>egor()</code> .
alter.file	A character specifying the filename of the alters data.
edges.file	A character specifying the filename of the edge data.
egos.file	A character specifying the filename of the ego data.
alters.file	Character name of the alters data file.

**Value**

A list of six objects - the **egoR** object: (1) egos.df: dataframe of all egos and their attributes; (2) alters.df: dataframe of all alters; (3) alters.list: list of dataframes of all alters per ego; (4) edges: list of dataframes with edge lists per network; (5) graphs: list of igraph objects; (6) results: a result dataframe, pre-populated with the network size of each network

## Functions

- `read_egoweb`: This function reads in data from an EgoWeb 2.0 survey and transforms it to an egoR object. If no file name for the egos file is provided ego data is assumed to be merged with alters data and it will be extracted by `read_egoweb`. By default the standard ID variable names of EgoWeb are used, if you need to specify the ID variable names use the `ID.vars` parameter. Further Information: [github.com/qualintitative/egoweb](https://github.com/qualintitative/egoweb)
- `read_openeddi`: This function reads in data created by the openeddi survey software and transforms it to an egoR object. If no parameters are provided `read_openeddi` will try to find the adequate files in the working directory. By default the standard ID variable names of openeddi are used, if you need to specify the ID variable names use the `ID.vars` parameter. Further Information: [www.openeddi.com](http://www.openeddi.com)

## Examples

```
# The data for read.egonet.threefiles() needs to be loaded with read.csv(),
# for it to be converted to an egoR object.
egos.file <- system.file("extdata", "egos_32.csv", package = "egor")
alters.file <- system.file("extdata", "alters_32.csv", package = "egor")
edges.file <- system.file("extdata", "edges_32.csv", package = "egor")

egos <- read.csv2(egos.file)
alters <- read.csv2(alters.file)
edges <- read.csv2(edges.file)

tf <- threefiles_to_egor(egos = egos, alters.df = alters, edges = edges)

# read_egoweb() and read_openeddi() read the files directly from the disk.

#' # Fetch current working directory
wd <- getwd()

#' setwd(system.file("extdata", "openeddi", package = "egor"))
oe <- read_openeddi()

setwd(system.file("extdata", "egoweb", package = "egor"))
ew <- read_egoweb(alter.file = "alters_32.csv", edges.file = "edges_32.csv",
                 egos.file = "egos_32.csv")

# Restore working directory
setwd(wd)
```

---

trim\_aaties

*Trims alter-alter ties of alters that are missing/ deleted from alters data*

---

## Description

Trims alter-alter ties of alters that are missing/ deleted from alters data

**Usage**

```
trim_aaties(object)
```

**Arguments**

object            An egor object.

**Value**

An egor object with trimmed alter-alter ties (.aaties).

---

twofiles_to_egor	<i>Import ego-centered network data from two file format</i>
------------------	--

---

**Description**

This function imports ego-centered network data, stored in two files, where one file contains the ego attributes and the edge information and the other file contains the alters data. This form of data storage for ego-centered network data is proposed by Muller, Wellman and Marin (1999).

**Usage**

```
twofiles_to_egor(egos, alters, netsize = NULL, ID.vars = list(ego =
  "egoID", alter = "alterID", source = "Source", target = "Target"),
  e.max.alters, e.first.var, ego.vars = NULL, selection = NULL, ...)
```

**Arguments**

egos	Data frame containing ego data (egos as cases)
alters	Data frame containing alters data (alters as cases), alters are separated by a variable containing an egoID.
netsize	Vector containing values of network size per ego.
ID.vars	A named list containing column names of the relevant input columns: ego unique identifier associated with each ego, defaulting to "egoID"; has no effect if alters.df and aaties.df are both lists of data frames. alter unique-within-ego identifier associated with each alter, defaulting to "alterID"; optional aaties.df are not provided. source if aaties.df is provided, the column given the alter identifier of the origin of a relation. target if aaties.df is provided, the column given the alter identifier of the destination of a relation.
e.max.alters	Maximum number of alters that are included in edge data.
e.first.var	Index or name of the first column in egos containing edge data.
ego.vars	Character vector naming variables in the egos data, in order to copy them in to the long alters dataframe.

selection	Character naming numeric variable indicating alters selection with zeros and ones.
...	additional arguments to <code>egor()</code> .

### Value

A list of six objects - the **egoR** object: (1) `egos.df`: dataframe of all egos and their attributes; (2) `alters.df`: dataframe of all alters; (3) `alters.list`: list of dataframes of all alters per ego; (4) `edges`: list of dataframes with edge lists per network; (5) `graphs`: list of igraph objects; (6) `results`: a result dataframe, pre-populated with the network size of each network

---

vert.attr	<i>Vertex attribute to data frame into graph.</i>
-----------	---

---

### Description

Extracts a vertex variable from a data.frame and returns it as a vector ready to be set as vertex attribute in a graph. May aggregate the variable by node before returning. This function extracts variables at the author level from 'data' and set them as vertex.attributes in 'graph', a network of authors. The function aggregates the variable by author using the aggregating function FUN. The function needs a 'graph' igraph object of authors in which the UFIDs are recorded as a attribute, and a 'data' data.frame in which the same UFIDs are associated to a variable (attribute). The function takes data and aggregates FUN(attribute) by dataID in 'data'. It returns the result of the aggregation in the order of dataID given by V(graph)\$name. NOTE that if attribute is categorical (e.g. College), the argument FUN is ignored and vert.attr() is just going to take the category of the variable for authorID (if there are more than 1 category, e.g. more than 1 Academic Units, for the same authorID, vert.attr just takes the 1st category associated to authorID in 'data').

### Usage

```
vert.attr(data, attribute, graph, dataID, graphID = "name", FUN = NA,
          unfactor = TRUE)
```

### Arguments

data	Data frame. The data.frame containing the author attribute variable to be imported in the graph (it must also include a authorID variable).
attribute	A variable name in 'data' (as character)
graph	Researchers graph where to import the attribute.
dataID	Character. The variable in data giving the ID of nodes in the graph (authors)
graphID	Character. The vertex attribute of graph that gives node IDs (i.e. IDs of authors) which are the same as dataID.
FUN	A function name given as character. The argument is ignored if "data" has one record for each dataID value, i.e. does not need aggregation by dataID. If NA or missing, attribute is aggregated by just taking its 1st non-NA value for each value of dataID.

unfactor Logical. If TRUE, and attribute is character, keeps it from being turned into factor. Notice that this argument has no effect if the attribute is aggregated first. **TODO:** regulate this behavior when attribute is factor (not character) in the first place; regulate this behavior when attribute is aggregated first.

### Value

A vector that contains the attribute 'attribute' from data, in the order given by the order of authorID in the graph 'graph'.

### See Also

[vert.attr.multi](#)

---

vert.attr.multi	<i>Get multiple variables from a data.frame and set them as vertex attributes in a graph. Uses vert.attr() in a loop.</i>
-----------------	---

---

### Description

This function repeatedly applies vert.attr() to a set of attributes. The problem that this function solves is that authors (nodes) in the graph and authors (records) in the data.frame data are not necessarily in the same sort order.

### Usage

```
vert.attr.multi(data, graph, dataID, graphID = "name", attributes,
  FUN = NA, attr.names = attributes)
```

### Arguments

data	The data.frame containing the vertex attribute variables to be imported in the graph (it must also include a authorID variable).
graph	The graph where to import the attribute.
dataID	Character. The ID variable for nodes in data. Must be the name of a variable in data. Will be used as a merge key with graphID.
graphID	Character. The vertex attribute that contains node IDs in graph. Must be the name of a vertex attribute in graph. Will be used as a merge key with dataID.
attributes	Character. Variable names in data.
FUN	Character. Function names given as character vector (the FUN argument to vert.attr()). If the vector is shorter than "attributes", it's recycled. Defaults to NA for all attributes.
attr.names	Character. Names to use for the attributes in the graph. Defaults to attributes.

### Value

An igraph object.

**Uses**

`vert.attr`

---

vis\_clustered\_graphs *Visualize clustered graphs*

---

**Description**

vis\_clustered\_graphs visualizes clustered\_graphs using a list of clustered graphs created with clustered\_graphs.

**Usage**

```
vis_clustered_graphs(graphs, node.size.multiplier = 1,
  node.min.size = 0, node.max.size = 200, edge.width.multiplier = 30,
  center = 1, label.size = 0.8, labels = FALSE,
  legend.node.size = 45, pdf.name = NULL, ...)
```

**Arguments**

graphs	List of graph objects, representing the clustered graphs.
node.size.multiplier	Numeric used to multiply the node diameter of visualized nodes.
node.min.size	Numeric indicating minimum size of plotted nodes
node.max.size	Numeric indicating maximum size of plotted nodes
edge.width.multiplier	Numeric used to multiply the edge width.
center	Numeric indicating the vertex to be plotted in center.
label.size	Numeric.
labels	Boolean. Plots with turned off labels will be preceded by a 'legend' plot giving the labels of the vertices.
legend.node.size	Numeric used as node diameter of legend graph.
pdf.name	Character giving the name/path of the pdf file to create.
...	Arguments to pass to plot.igraph.

**Value**

vis\_clustered\_graphs plots a list of igraph objects created by the clustered\_graphs function.

clustered\_graphs returns a list of graph objects representing the clustered ego-centered network data;

## References

Brandes, U., Lerner, J., Lubbers, M. J., McCarty, C., & Molina, J. L. (2008). Visual Statistics for Collections of Clustered Graphs. 2008 IEEE Pacific Visualization Symposium, 47-54.

## See Also

[clustered\\_graphs](#) for creating clustered graphs objects

## Examples

```
data("egor32")

# Simplify networks to clustered graphs, stored as igraph objects
graphs <- clustered_graphs(egor32, "country")

# Visualise
vis_clustered_graphs(graphs,
                      node.size.multiplier = 5,
                      edge.width.multiplier = 25,
                      labels = TRUE)
```

---

weights.egor	<a href="#">weights.egor()</a> extracts the (relative) sampling weights of each ego in the dataset.
--------------	---

---

## Description

[weights.egor\(\)](#) extracts the (relative) sampling weights of each ego in the dataset.

## Usage

```
## S3 method for class 'egor'
weights(object, ...)
```

## Arguments

object	an <a href="#">egor</a> object.
...	arguments to be passed to methods

## See Also

[weights.survey.design](#)



# Index

- \*Topic **analysis**
  - alts\_diversity\_count, 4
  - clustered\_graphs, 6
  - comp\_ei, 8
  - comp\_ply, 8
  - composition, 7
  - ego\_density, 14
  - egor, 11
  - egor\_vis\_app, 13
  - vis\_clustered\_graphs, 31
- \*Topic **datasets**
  - alters32, 2
  - edges32, 10
  - egor32, 13
  - egos32, 14
- \*Topic **ego-centered**
  - alts\_diversity\_count, 4
  - clustered\_graphs, 6
  - comp\_ei, 8
  - comp\_ply, 8
  - composition, 7
  - ego\_density, 14
  - egor, 11
  - egor\_vis\_app, 13
  - EI, 16
  - make\_egor, 19
  - read\_egonet, 21
  - vis\_clustered\_graphs, 31
- \*Topic **import**
  - onefile\_to\_egor, 19
  - read\_egonet, 21
  - twofiles\_to\_egor, 28
- \*Topic **network**
  - alts\_diversity\_count, 4
  - clustered\_graphs, 6
  - comp\_ei, 8
  - comp\_ply, 8
  - composition, 7
  - ego\_density, 14
  - egor, 11
  - egor\_vis\_app, 13
  - EI, 16
  - make\_egor, 19
  - vis\_clustered\_graphs, 31
- \*Topic **random**
  - make\_egor, 19
- \*Topic **sna**
  - EI, 16
  - [.egor (subset.egor), 23
- alter\_design, 3
- alter\_design<- (alter\_design), 3
- alters32, 2
- alts\_diversity\_count, 4
- alts\_diversity\_entropy
  - (alts\_diversity\_count), 4
- as.egor (egor), 11
- as.igraph.egor (convert\_egor), 9
- as.network.egor (convert\_egor), 9
- as\_aaties\_df (as\_alts\_df), 5
- as\_alts\_df, 5
- as\_igraph (convert\_egor), 9
- as\_network (convert\_egor), 9
- clustered\_graphs, 6, 31, 32
- comp\_ei, 8
- comp\_ply, 8
- composition, 7
- convert\_egor, 9
- create\_edge\_names\_wide (helper), 18
- data.frame(), 22
- din\_page\_dist (helper), 18
- dyad.poss (helper), 18
- dyads\_possible\_between\_groups (helper), 18
- edges32, 10
- ego\_density, 14

ego\_design, 15  
ego\_design<- (ego\_design), 15  
egor, 3, 11, 11, 12, 16, 25, 32  
egor(), 11, 20, 21, 23, 24, 26, 29  
egor-package-doc, 12  
egor32, 13  
egor\_vis\_app, 13  
egos32, 14  
EI, 16

gss2004, 17

haven::read\_sav(), 17  
helper, 18

list, 11, 12, 16  
list(), 22, 24

make\_egor, 19

onefile\_to\_egor, 19

print.egor (summary.egor), 25

read\_egonet, 21  
read\_egoweb (threefiles\_to\_egor), 25  
read\_openeddi (threefiles\_to\_egor), 25  
regex, 20  
rowlist, 22

sanitize.wide.edges (helper), 18  
subset(), 23, 24  
subset.egor, 23  
summary.egor, 25  
survey::svydesign(), 11, 12  
svydesign, 15  
svydesign(), 16

threefiles\_to\_egor, 25  
tibble, 12  
tibble(), 22  
trim\_aaties, 27  
twofiles\_to\_egor, 28

vert.attr, 29, 31  
vert.attr.multi, 30, 30  
vis\_clustered\_graphs, 6, 31

weights.egor, 32  
weights.egor(), 32  
weights.survey.design, 32