

# Package ‘egor’

October 7, 2021

**Type** Package

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

**Version** 1.21.10

**Date** 2021-10-07

**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](https://doi.org/10.1109/PACIFICVIS.2008.4475458)>).

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

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

**License** AGPL-3

**Depends** R (>= 3.5.0), dplyr, tibble

**Imports** tidygraph, srvyr, tidyr, methods, utils, purrr, rlang

**Suggests** knitr, testthat (>= 2.1.0), rmarkdown, survey, shiny, igraph, network, haven

**VignetteBuilder** knitr

**RoxygenNote** 7.1.1

**LazyData** true

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Till Krenz [aut, cre],  
Pavel N. Krivitsky [aut],  
Raffaele Vacca [aut],  
Michal Bojanowski [aut] (<<https://orcid.org/0000-0001-7503-852X>>),  
Markus Gamper [ctb],  
Andreas Herz [aut],  
Christopher McCarty [ctb]

**Maintainer** Till Krenz <[egor@tillt.net](mailto:egor@tillt.net)>

Repository CRAN

Date/Publication 2021-10-07 17:00:05 UTC

## R topics documented:

aaties32 . . . . .	3
activate.egor . . . . .	3
allbus_2010_simulated . . . . .	4
alters32 . . . . .	4
alter_design . . . . .	5
alts_diversity_count . . . . .	6
append_egor . . . . .	7
as.egor . . . . .	8
as_igraph . . . . .	10
as_tibble.egor . . . . .	12
clustered_graphs . . . . .	14
composition . . . . .	15
comp_ei . . . . .	16
comp_ply . . . . .	16
count_dyads . . . . .	17
egor-package-doc . . . . .	18
egor32 . . . . .	19
egor_options . . . . .	19
egor_vis_app . . . . .	20
egos32 . . . . .	20
ego_constraint . . . . .	21
ego_density . . . . .	22
ego_design . . . . .	22
EI . . . . .	23
extract_egos_and_return . . . . .	25
gss2004 . . . . .	25
helper . . . . .	26
layout_egogram . . . . .	28
make_egor . . . . .	28
onefile_to_egor . . . . .	29
plot_egograms . . . . .	30
read_egonet . . . . .	34
return_results . . . . .	35
rowlist . . . . .	35
subset.egor . . . . .	36
summary.egor . . . . .	38
threefiles_to_egor . . . . .	38
transnat . . . . .	41
trim_aaties . . . . .	42
trim_alters . . . . .	43
twofiles_to_egor . . . . .	43
vis_clustered_graphs . . . . .	45

<i>aaties32</i>	3
<i>weights.egor</i> . . . . .	46
<b>Index</b>	<b>48</b>

---

<i>aaties32</i>	<i>32 sets of randomly created alter-alter ties belonging to ego-centered networks</i>
-----------------	--

---

**Description**

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

**Usage**

*aaties32*

**Format**

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

- .EGOID** ego identifier
- .SRCID** source alter ID
- .TGTID** target alter ID
- weight** weight of relation

---

<i>activate.egor</i>	<i>Activate ego, alter or alter-alter tie data level of an egor dataset</i>
----------------------	---

---

**Description**

This function activates one of the data levels of an egor dataset, so that the dplyr verbs know which level to execute on.

**Usage**

```
## S3 method for class 'egor'
activate(.data, what)
```

**Arguments**

- .data* The egor dataset.
- what* Character naming the level to activate, this can be "ego", "alter" or "aatie".

**Examples**

```
e <- make_egor(5,50)
e %>%
  activate("aatie") %>%
  mutate(weight2 = 2 + weight) %>%
  activate("alter") %>%
  mutate(age.years = age.years^3)
```

---

allbus\_2010\_simulated *Simulated Allbus 2010 Data*

---

**Description**

A dataset simulated based on the the original Allbus 2010 SPSS data. The dataset simulates 100 respondents and does not resemble any actual Allbus respondents. Each variable is randomly generated based on the range of the original variables, co-varianaces between variables are disregarded. The data's purpose is purely to demonstrate how to technically work with the Allbus data using egor and R - no analytical assumptions should be made based on this data!

**Usage**

```
allbus_2010_simulated
```

**Format**

A tibble/ data.frame of 100 simulated respondents/ rows and 981 variables/ columns. Each variable is a labelled dbl.

**Details**

The dataset contains (simulated!) answers to two ego-centered name generators.

---

alters32 *32 sets of randomly created alters belonging to ego-centered networks*

---

**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:

**.ALTID** alter identifier

**.EGOID** ego identifier

**age** age in categories

**age.years** age in years

**country** country

**income** income

**sex** gender

---

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 = 2)
```

### 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 tibble with the ego ID and a numeric result 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")
```

---

append_egor	<i>Append rows/columns to ego, alter or aatie data</i>
-------------	--

---

### Description

These work like dplyr's `bind_cols()` and `bind_rows()`. The first argument has to be an egor object. Additional rows/columns are added bottom/RHS of the active data level (ego, alter, aatie).

### Usage

```
append_rows(.egor, ..., .id = NULL)

append_cols(.egor, ...)
```

### Arguments

<code>.egor</code>	An egor object.
<code>...</code>	Data frames to combine.
<code>.id</code>	Data frame identifier.

### Value

egor object containing the additional rows/ columns on the active level.

### Examples

```
e <- make_egor(12, 15)

# Adding a column to the ego level
additional_ego_columns <-
  tibble(x = sample(1:3, 12, replace = TRUE))

append_cols(e, additional_ego_columns)

# Adding rows to the ego and alter level
additional_ego_rows <-
  list(
    .egoID = 13,
    sex = "w",
    age = factor("56 - 65"),
    age.years = 60,
    country = "Australia"
  ) %>%
  as_tibble()

additional_alter_rows <-
  list(
    .altID = 1:5,
    .egoID = rep(13, 5),
```

```

    sex = sample(c("f", "m"), 5, replace = TRUE)
  ) %>%
  as_tibble()

append_rows(e, additional_ego_rows) %>%
  activate(alter) %>%
  append_rows(additional_alter_rows)

```

---

as.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. `as.egor()` converts a list of `igraph/network` objects or a `nested_egor` objects to an `egor` object.

### Usage

```

as.egor(x, ...)

## S3 method for class 'nested_egor'
as.egor(
  x,
  ID.vars = list(ego = ".egoID", alter = ".alterID", source = ".Source", target =
    ".Target"),
  ...
)

## S3 method for class 'list'
as.egor(x, ego_name = NULL, ...)

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

```

### Arguments

<code>x</code>	list of <code>igraph/network</code> objects representing ego networks.
<code>...</code>	arguments to be passed to methods
<code>ID.vars</code>	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 <code>alters.df</code> and <code>aaties.df</code> are both lists of data frames.
	alter	unique-within-ego identifier associated with each alter, defaulting to "alterID"; optional <code>aaties.df</code> are not provided.
	source	if <code>aaties.df</code> is provided, the column given the alter identifier of the origin of a relation.
	target	if <code>aaties.df</code> is provided, the column given the alter identifier of the destination of a relation.
ego_name		character or numeric of length one or same length as there are networks. If the <code>igraph/network</code> objects don't include egos as a node, set to NULL (default).
alters		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 NULLs corresponding to egos with no nominees.
egos		<code>data.frame</code> containing the egos.
aaties		<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> .
ego_design		A <a href="#">list</a> of arguments to <code>srvyr::as_survey_design()</code> specifying the sampling design for the egos. If formulas, they can refer to columns of <code>egos.df</code> . NULL means that no design is set.
alter_design		A <a href="#">list</a> of arguments specifying nomination information. Currently, the following elements are supported: "max" Maximum number of alters that an ego can nominate.

### 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, which is a named list with three tibble `data.frames`: `ego`, `alter` and `aatie` (alter-alter ties). Each data set has an `.egoID` column, that groups the data belonging to one ego. Additionally the alter data has an `.alterID` column, that links to the columns `.srcID` and `.tgtID` in the alter-alter tie data.

In addition, `egor` has two attributes: `ego_design`, containing an object returned by `srvyr::as_survey_design()` 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.

### Methods (by generic)

- `as.egor`: Can convert (legacy) `nested_egor` object to `egor` object.

**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.

**See Also**

`as_tibble()` for extracting ego, alter, and alter–alter tables, as `tibbles` or as `srvyr`'s `tbl_svy` surveys.

**Examples**

```
data("egos32")
data("alters32")
data("aaties32")

egor(alters32,
     egos32,
     aaties32,
     ID.vars = list(ego = ".EGOID",
                   alter = ".ALTID",
                   source = ".SRCID",
                   target = ".TGTID"))
```

---

as\_igraph

---

*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,
  graph.attrs = ".egoID"
)

## S3 method for class 'nested_egor'
as_igraph(
  x,
  directed = FALSE,
  include.ego = FALSE,
```

```

    ego.attrs = NULL,
    ego.alter.weights = NULL,
    graph.attrs = ".egoID"
  )

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

as_network(
  x,
  directed = FALSE,
  include.ego = FALSE,
  ego.attrs = NULL,
  ego.alter.weights = NULL,
  graph.attrs = ".egoID"
)

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

```

### 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. This is ignored, when <code>include.ego = FALSE</code> (default).
<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. This is ignored, when <code>'include.ego = FALSE'</code> (default).
<code>graph.attrs</code>	Vector of names (character) or indices (numeric) of ego variables that are supposed to be carried over to the igraph object as graph attributes or the network

object as network attributes. By default .egoID is carried over.

### 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 merged successfully in the resulting network/ igraph object (see example).

### Examples

```
e <- make_egor(3, 22)
as_igraph(e)
```

---

as_tibble.egor	<i>Extract ego, alter, and alter-alter tables from an egor object.</i>
----------------	--

---

### Description

Provided an egor object, these functions create a "global" tibble or srvyr's `tbl_svy` object containing egos, alter attributes, or alter-alter relations. The resulting tables are useful for advanced analysis procedures, e.g. multi-level regressions.

`as_tibble()` method for egor extracts the currently active component (ego, alter, or aaties) table, optionally joining it with the others, dropping any survey design information.

`as_survey()` method for egor instead returns a srvyr `tbl_svy` survey, taking into account any replication due to multiple alters or alter-alter ties incident on each ego. If no design is specified for the egos, then the default design (simple random sample with replacement) is assumed as the starting point.

`as_egos_df()`, `as_alters_df()`, `as_aaties_df()`, `as_egos_survey()`, `as_alters_survey()`, and `as_aaties_survey()` are convenience functions for the `as_tibble()` and `as_survey()` methods, activating the corresponding component of the egor object.

### Usage

```
## S3 method for class 'egor'
as_tibble(x, ..., include.ego.vars = FALSE, include.alter.vars = FALSE)

## S3 method for class 'egor'
as_survey(.data, ..., include.ego.vars = FALSE, include.alter.vars = FALSE)

as_egos_df(object)

as_alters_df(object, include.ego.vars = FALSE)

as_aaties_df(object, include.ego.vars = FALSE, include.alter.vars = FALSE)

as_egos_survey(object, include.ego.vars = FALSE)
```

```
as_alters_survey(object, include.ego.vars = FALSE)
```

```
as_aaties_survey(object, include.ego.vars = FALSE, include.alter.vars = FALSE)
```

### Arguments

x, object, .data

An egor object.

... Additional arguments, currently unused.

include.ego.vars

Logical, specifying if ego variables should be included in the result.

include.alter.vars

Logical, specifying if alter variables should be included in the result.

### Value

A tibble for the as\_tibble and \*\_df functions and a tbl\_svy for as\_survey and the \*\_survey functions.

### Examples

```
# Load example data
data(egor32)

as_tibble(egor32) # Ego table.

egor32 %>%
  activate("alter") %>%
  as_tibble(include.ego.vars=TRUE) # Alter table, but also with ego variables.

library(srvyr)
as_survey(egor32) # Ego table with survey design.

# Despite alter table being active, obtain the ego table.
(egor32 <- activate(egor32, "alter"))
as_egos_df(egor32)

# Create global alter table
as_alters_df(egor32)

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

# ... adding alter variables
as_aaties_df(egor32, include.alter.vars = TRUE)
as_egos_survey(egor32)
as_alters_survey(egor32) # Notice the resulting cluster design.
```

---

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>	An egor object.
<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> containg alter-alter relations as a 'global edge list' or as a list of 'edge lists'. (not needed if <code>object</code> is an egor 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
par(mfrow = c(2,3))
vis_clustered_graphs(
  graphs[1:5]
)
par(mfrow = c(1,1))
```

---

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**

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

**Value**

A tibble with the ego ID and values per category of alt.attr as numeric 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 homo-/heterophily.

**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 tibble with the ego ID and a numeric result vector.

**Examples**

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

---

comp_ply	<i>Calculate custom 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, result.name = "result")
```

**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.
result.name	Optional character naming the result column.



**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 tibble with the ego ID and a numeric result vector.

**Author(s)**

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

Till Krenz, <public@tillt.net>

**Examples**

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

---

count\_dyads

*Count attribute combinations of dyads in ego-centered networks*

---

**Description**

`count_dyads()` counts the attribute combinations of alter-alter ties/ dyads in ego-centered networks. The results can be returned as a wide or long tibble/ data.frame.

**Usage**

```
count_dyads(
  object,
  alter_var_name,
  return_as = c("wide", "long"),
  prefix = NULL
)
```

**Arguments**

`object` An ego object.

`alter_var_name` Character, naming the alter variable to use as attribute.

`return_as` Character, either "wide" (default) or "long".

`prefix` Character, added in front of variables. Only used if `return_as` is "wide". If NULL (default) prefix is automatically generated.

**Value**

Wide or long tibble/ data.frame.

## Examples

```
data(egor32)
count_dyads(object = egor32,
             alter_var_name = "country")

# Return result as long tibble.
count_dyads(object = egor32,
             alter_var_name = "country",
             return_as = "long")
```

---

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

---

## Description

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

## Details

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

When analyzing ego-centered network data it is common to either include or exclude data on ego. By default egor excludes ego for most analytic and visual functions and offers to include ego with the argument `include.ego`. In order for egor to automatically detect which ego-level variables correspond to the alter-variables, the variables on the different levels need to be named exactly the same. Alternatively the functions that have an `include.ego` argument have additional arguments that allow to specify the name of the ego-level variable in question.

## Author(s)

Till Krenz, <egor@tillt.net>

Pavel Krivitsky, <pavel@uow.edu.au>

Michał Bojanowski <m.bojanowski at icm.edu.pl>

Andreas Herz, <herzand@uni-hildesheim.de>

Raffaele Vacca, <r.vacca@ufl.edu>

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\_options *Display names and values of global egor options.*

**Description**

Display names and values of global egor options.

**Usage**

egor\_options()

**Details**

Currently egor makes use of the following global options. Use options() to change values.

- egor.print.rows.active.level:**
  - Numeric. Amount of rows to display when printing the active level of an egor object.
- egor.print.rows.inactive.level:**
  - Numeric. Amount of rows to display when printing the inactive levels of an egor object.
- egor.print.switch.active.level.to.top:**
  - Logical. When printing an egor object, should the active data-level always be printed first?
- egor.return.results.with.design:**
  - Logical. egor functions that return ego-level results (e.g. one value per ego) return a svy\_tbl object containing the ego\_design(), when this is set to TRUE.

---

`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 = NULL, shiny_opts = list(launch.browser = TRUE))
```

**Arguments**

`object` An egor object.

`shiny_opts` A [list](#) of arguments to be passed to `shiny::shinyApp()`'s options argument.

**Note**

This function requires [shiny](#) to be installed.

**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  
**age** age in categories  
**age.years** age in years  
**country** country  
**income** income  
**sex** gender

---

ego_constraint	<i>Calculate Burt constraint for the egos of ego-centered networks</i>
----------------	--

---

**Description**

This calculates Burt's *network constraint* for all egos in an egor object. It iterates over each network and applies `igraph::constraint`. A weight variable can be specified.

**Usage**

```
ego_constraint(object, weights = NULL, ego.alter.weights = weights)
```

**Arguments**

object	An egor object.
weights	Character, naming the alter-alter tie weight variable.
ego.alter.weights	Character, naming the ego-alter weight tie weight variable. This defaults to the same value as weights, only specify if the name of the ego.alter.weights is different from weights.

**Details**

The calculation of weighted network constraint only works, if the alter-alter tie weights are complemented by a alter level variable specifying the same weight for the ego-alter ties.

**Value**

Numeric vector with a constraint value for each ego.

**References**

Burt, R. (2004). Structural holes and good ideas. *American Journal of Sociology*, (110), 349–399.

**Examples**

```
data(egor32)
ego_constraint(egor32)
```

---

ego_density	<i>Calculate the relationship density in ego-centered networks</i>
-------------	--

---

### 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 'egor'
ego_density(object, weight = NULL, max.netsize = NULL, directed = FALSE, ...)
```

### Arguments

object	An egor object.
...	arguments to be passed to methods
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.

### Value

returns a vector of network density values.

### Examples

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

---

ego_design	<i>Set and query the ego sampling design</i>
------------	--

---

### Description

Extract, set, remove, or update the survey design associated with an ego-centered dataset.

**Usage**

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

**Arguments**

x	an <a href="#">egor</a> object.
...	arguments to be passed to methods
value	a <a href="#">list</a> of arguments to <a href="#">srvyr::as_survey_design()</a> specifying the sampling design for the egos. If the arguments are formulas, they can refer to columns (ego attributes) of x. NULL clears design information.

**Note**

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

## Description

The EI-Index is the division of the surplus count intra-group edges over inter-group edges, divided by total count of all edges. This implementation uses the intra-group and inter-group density instead of edge counts, when `rescale` is set to `TRUE` (default). The EI-Index is calculated for the whole network and for subgroups. Alternatively, the EI index can be employed as a measurement for egos tendency to homo-/heterophily - use `comp_ei()`. for that variant of the EI-Index.

## Usage

```
EI(object, alt.attr, include.ego = FALSE, ego.attr = alt.attr, rescale = TRUE)
```

## Arguments

<code>object</code>	An egor object.
<code>alt.attr</code>	Character naming grouping variable.
<code>include.ego</code>	Logical. Include or exclude ego from EI calculation.
<code>ego.attr</code>	Character, naming the ego variable corresponding to <code>ego.attr</code> . Defaults to <code>ego.attr</code> .
<code>rescale</code>	Logical. If <code>TRUE</code> , the EI index calculation is rescaled, so that the EI is not distorted by differing group sizes.

## Details

The whole network EI is a metric indicating the tendency of a network to be clustered by the categories of a given factor variable (`alt.attr`). The EI value of a group describes the tendency of that group within a network to be connected (if between 0 and 1) or not connected (if between -1 and 0) to other groups. Differing group sizes can lead to a distortion of EI values i.e. the ability of a big group A to form relationships to much smaller group B is limited by the size of B. Even when all possible edges between A and B exist, the EI value for group A might still be negative, classifying it as *homophile*. The rescaled EI-Index values provided by this implementation substitutes absolute edge counts by inter- and intra-group edge densities in order to avoid the distortion of the EI-Index values. These values express the extend of homo- or heterophily of the network and its subgroups, *as made possible by subgroup sizes*.

## Value

Returns tibble with the following columns:

- ego ID ("`egoID`")
- network EI-Index ("`ei`")
- subgroup EI-Index values (named by value levels of `alt.attr/ego.attr`)

## 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.



**See Also**

[comp\\_ei\(\)](#), for an ego level homophily measure.

**Examples**

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

---

```
extract_egos_and_return
```

*This extracts egos from igraph/network data if they are named in ego\_name and returns an egor object*

---

**Description**

This extracts egos from igraph/network data if they are named in ego\_name and returns an egor object

**Usage**

```
extract_egos_and_return(graph_attrs, alters, edges, ego_name = NULL)
```

**Arguments**

graph_attrs	List of graph attributes
alters	alters
edges	edges
ego_name	ego_name

---

```
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.

**Usage**

```
gss2004
```

**Format**

A tibble with 499 rows and the variables listed below. Data was imported from SPSS file 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

*General helper functions*

---

**Description**

Helper functions for ego centered network analysis

**Usage**

```

as_nested_ego(x)

alters_by_ego(x)

## S3 method for class 'ego'
alters_by_ego(x)

## S3 method for class 'nested_ego'
alters_by_ego(x)

aaties_by_ego(x)

## S3 method for class 'ego'
aaties_by_ego(x)

## S3 method for class 'nested_ego'
aaties_by_ego(x)

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

sanitize.wide.edges(max.alters)

create_edge_names_wide(x)

dyads_possible_between_groups(x, y, geometric = TRUE)

din_page_dist(x)

```

**Arguments**

x	Numeric.
max.alters	A numeric giving the maximum number of alters.
directed	A logical value indicating directedness of alter-alter data.
y	Numeric.
geometric	Logical. Calculate possible dyads for geometric mean?

**Functions**

- `as_nested_ego`: Converts an ego object to a "legacy" ego object with nested `.alts` and `.aaties` columns.
- `alters_by_ego`: Splits the alter table into a list of tables (possibly 0-row) of alters associated with each ego, in the same order as the ego table.
- `aaties_by_ego`: Splits the alter–alter ties table into a list of tables (possibly 0-row) of alter–alter associated with each ego, in the same order as the ego table.
- `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.

---

<code>layout_egogram</code>	<i>Create layout for an egogram</i>
-----------------------------	-------------------------------------

---

### Description

This creates pairs of x and y coordinates for a egogram, accompanied by alter IDs for each coordinate pair.

### Usage

```
layout_egogram(altID, venn_var, pie_var)
```

### Arguments

<code>altID</code>	Vector of alter IDs.
<code>venn_var</code>	Vector of values representing alter groups corresponding with venns in an egogram.
<code>pie_var</code>	Vector of values representing alter groups corresponding with pieces of pie in an egogram.

### Value

A dataframe with three columns: x, y and altID.

---

<code>make_egor</code>	<i>Generate random ego-centered-network data.</i>
------------------------	---

---

### 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_egor(net.count, max.alters, netsize_fixed = FALSE, plot = FALSE)
```

**Arguments**

<code>net.count</code>	Number of networks/ egos to generate.
<code>max.alters</code>	Maximum size of networks.
<code>netsize_fixed</code>	Logical, if TRUE all networks will have max.alters as network size.
<code>plot</code>	whether to plot the network size distribution.

---

<code>onefile_to_egor</code>	<i>Import ego-centered network data from 'one file format'</i>
------------------------------	--

---

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

**Arguments**

<code>egos</code>	Data frame containing ego data (egos as cases)
<code>netsize</code>	Numeric, network size values are used to filter out empty alter entries. If the alter data is not structured in a way, where valid alters are stored before the invalid alters, pass NULL here and filter out invalid alters afterwards.
<code>ID.vars</code>	Character. For <code>onefile_to_egor</code> only the name of the ego ID needs to be provided.
<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.

var.wise            Logical value indicating if the alter attributes are sorted variable wise (defaults to FALSE).

...                additional arguments to `egor()`.

### Value

An **egor** object is returned. It is a list of three data frames: (1) `ego`: dataframe of all egos and their attributes; (2) `alter`: dataframe of all alters; (3) `aatie`: dataframe of alter alter ties/ edges

### 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.

### Examples

```
path_to_one_file_8 <- system.file("extdata", "one_file_8.csv", package = "egor")
egos_8 <- read.csv2(path_to_one_file_8, row.names = 1)

attr.start.col <- which(names(egos_8) == "alter.sex.1")
attr.end.col <- which(names(egos_8) == "alter.age.8")
dy.first.var <- which(names(egos_8) == "X1.to.2")

onefile_to_egor(
  egos = egos_8, netsize = egos_8$netsize,
  attr.start.col = attr.start.col,
  attr.end.col = attr.end.col,
  aa.first.var = dy.first.var,
  max.alter = 8)
```

---

plot\_egograms

*Plotting egor objects*

---

### Description

*egor* objects can be plotted as *egographs* or *egograms*. By default networks of the four first egos are plotted.

### Usage

```
plot_egograms(
  x,
  ego_no = 1,
  x_dim = 1,
  y_dim = 1,
  venn_var = NULL,
  pie_var = NULL,
  ascending_inwards = TRUE,
```

```

    vertex_size_var = NULL,
    vertex_color_var = NULL,
    vertex_color_palette = "Heat Colors",
    vertex_color_legend_label = vertex_color_var,
    vertex_label_var = "name",
    edge_width_var = NULL,
    edge_color_var = NULL,
    edge_color_palette = "Heat Colors",
    highlight_box_col_var = NULL,
    highlight_box_col_palette = "Heat Colors",
    res_disp_vars = NULL,
    vertex_zoom = 1,
    edge_zoom = 2,
    font_size = 1,
    pie_colors = NULL,
    venn_gradient_reverse = FALSE,
    show_venn_labels = TRUE,
    include_ego = FALSE,
    ...
)

plot_ego_graphs(
  x,
  ego_no = 1,
  x_dim = 1,
  y_dim = 1,
  vertex_size_var = NULL,
  vertex_color_var = NULL,
  vertex_color_palette = "Heat Colors",
  vertex_color_legend_label = vertex_color_var,
  vertex_label_var = "name",
  edge_width_var = NULL,
  ego_alter_edge_width_var = if (!is.null(edge_width_var) & include_ego) edge_width_var,
  edge_color_var = NULL,
  ego_alter_edge_color_var = if (!is.null(edge_color_var) & include_ego) edge_color_var,
  edge_color_palette = "Heat Colors",
  highlight_box_col_var = NULL,
  highlight_box_col_palette = "Heat Colors",
  res_disp_vars = NULL,
  vertex_zoom = 1,
  edge_zoom = 3,
  font_size = 1,
  include_ego = FALSE,
  ...
)

plot_egor(
  x,

```

```

    ego_no = 1,
    x_dim = 2,
    y_dim = 2,
    ...,
    type = c("egograph", "egogram")
)

```

```

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

```

### Arguments

<code>x</code>	An <i>egor</i> object.
<code>ego_no</code>	Ego row number.
<code>x_dim</code>	Number of ego networks to plot horizontally.
<code>y_dim</code>	Number of ego networks to plot vertically
<code>venn_var</code>	Name (character) of alter column.
<code>pie_var</code>	Name (character) of alter column.
<code>ascending_inwards</code>	Logical determining the venn circle order. If TRUE (default) values ascend from the outside to the inside, if FALSE the reverse.
<code>vertex_size_var</code>	Name (character) of alter column.
<code>vertex_color_var</code>	Name (character) of alter column.
<code>vertex_color_palette</code>	Name (character) of color palette, see details for available color palettes.
<code>vertex_color_legend_label</code>	Character.
<code>vertex_label_var</code>	Name (character) of alter column. Set this to NULL to suppress labels.
<code>edge_width_var</code>	Name (character) of aatie column.
<code>edge_color_var</code>	Name (character) of aatie column.
<code>edge_color_palette</code>	Name (character) of color palette, see details for available color palettes.
<code>highlight_box_col_var</code>	Name (character) of ego column.
<code>highlight_box_col_palette</code>	Name (character) of color palette, see details for available color palettes.
<code>res_disp_vars</code>	Name (character) of ego column.
<code>vertex_zoom</code>	Numeric.
<code>edge_zoom</code>	Numeric.
<code>font_size</code>	Numeric.
<code>pie_colors</code>	Character vector of colors to be used for coloring the subsections of the circle.



venn_gradient_reverse	Logical, set to TRUE in order to have the color intensity of venns increase going from the inner circles to the outer circles.
show_venn_labels	Logical.
include_ego	Logical.
...	Additional arguments forwarded to plot.igraph.
ego_alter_edge_width_var	Name (character) of alter column.
ego_alter_edge_color_var	Name (character) of alter column.
type	Character. Either "egograph" or "egogram".

### Details

For type equals "egograph" ego networks are plotted with igraph's plotting engine. "egogram" uses a special layout that places the nodes on a map of (1) concentric circles with (2) subsections, that can be mapped to alter variables.

Available color palettes are:

- Heat Colors
- Yellow-Green
- Red-Yellow
- Blue-Red
- Black-White
- Greys
- Rainbow
- Topo Colors

### Functions

- plot\_egograms: Plots an ego-socio-gram.
- plot\_ego\_graphs: Plots an ego graph.

### Examples

```
e <- make_ego(net.count = 5, max.alters = 12)
plot_egograms(x = e,
              ego_no = 2,
              venn_var = "sex",
              pie_var = "country",
              vertex_size_var = "age")
plot(e)
```

---

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**

An **egor** object is returned. It is a list of three data frames: (1) ego: dataframe of all egos and their attributes; (2) alter: dataframe of all alters; (3) aatie: dataframe of alter alter ties/ edges

**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 = ";")
```

---

return_results	<i>Returns results inheriting srvyr design if the input egor object has a an ego_design and global option "egor.return.results.with.design" is TRUE or 'NULL'.</i>
----------------	--

---

**Description**

Returns results inheriting srvyr design if the input egor object has a an ego\_design and global option "egor.return.results.with.design" is TRUE or 'NULL'.

**Usage**

```
return_results(x, results)
```

**Arguments**

x	Original egor object, as submitted in call to parent function.
results	data.frame with .egoID column and a column that hold the ego-level results.

---

rowlist	<i>Convert a table to a list of rows</i>
---------	--

---

**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)
```

---

subset.egor

---

*Filter and Subset Ego-centered Datasets*


---

**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 = attr(x, "active"))

## S3 method for class 'egor'
x[i, j, unit = attr(x, "active"), ...]
```

**Arguments**

`x` an `egor()` object.

`subset` either an expression evaluated on each of the rows of the selected unit (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. The expressions can access variables in the calling environment; columns of the active unit, columns of other units with which the active unit shares an ego via `ego$`, `alter$`, and `aatie$` 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. Defaults to the current active unit selected by `activate.egor()`.

`i` numeric or logical vector indexing the appropriate unit.

`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. Note that the special columns `.egoID`, `.altID`, `.srcID`, `.tgtID` are not indexed by `j`.

## 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,]

# Using an external vector
# (though normally, we would use e[.keep,] here)
.keep <- rep(c(TRUE, FALSE), length.out=nrow(e$ego))
subset(e, .keep)

# Filter egos
subset(x = egor32, subset = egor32$ego$variables$sex == "m", unit="ego")
subset(x = egor32, sex == "m")

# Filter alters
subset(x = egor32, sex == "m", unit = "alter")

# Filter aaties
subset(x = egor32, weight != 0, unit = "aatie")

# Filter egos by alter variables (keep only egos that have more than 13 alters)
subset(x = egor32, ego$country == "Poland", unit = "ego")

# Filter alters by ego variables (keep only alters that have egos from Poland)
subset(x = egor32, nrow(alter) > 13, unit = "alter")
```

```
# Filter edges by alter variables (keep only edges between alters where `sex == "m"`)
subset(x = egor32, all(alter$sex == "m"), unit = "aatie")
```

---

summary.ego
*Methods to print and summarize [ego](#) objects*


---

## Description

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

## Usage

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

## S3 method for class 'ego'
print(
  x,
  ...,
  n.active = getOption("ego.rows_active_level"),
  n.inactive = getOption("ego.rows_inactive_level")
)
```

## Arguments

object, x	an <a href="#">ego</a> object.
...	additional arguments, either unused or passed to lower-level functions.
n.active	Numeric. Number of rows to print for active data level.
n.inactive	Numeric. Number of rows to print for inactive data levels.
n	Number of rows to print.

---

threefiles\_to\_ego
*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, an 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: <ul style="list-style-type: none"> <li>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.</li> <li>alter unique-within-ego identifier associated with each alter, defaulting to "alterID"; optional aaties.df are not provided.</li> <li>source if aaties.df is provided, the column given the alter identifier of the origin of a relation.</li> <li>target if aaties.df is provided, the column given the alter identifier of the destination of a relation.</li> </ul>

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

An **egor** object is returned. It is a list of three data frames: (1) ego: dataframe of all egos and their attributes; (2) alter: dataframe of all alters; (3) aatie: dataframe of alter alter ties/ edges

### 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)
```

---

transnat	<i>Transnational personal communities of social support of German migrants in Great Britain</i>
----------	---

---

## Description

This is an egor object derived from a subset of the data of a personal network study on support relationships German migrants living in the UK maintain. The data was collected in 2010 using respondent driven sampling (snowball sampling). While the number of alters the respondents were allowed to enter was not limited, only a random subsample of up to eight alters were selected for the alter name interpreter and alter-alter tie questions. This data set contains the data for 50 of the originally 234 egos.

## Usage

```
transnat

alter_df

ego_df
```

## Format

```
transnat: an egor object of 50 egos.
alter_df: alter data.frame of the transnat dataset.
ego_df: ego data.frame of the transnat dataset.
```

## Details

The questionnaire used seven name generators:

1. From time to time, people rely on other people's advice and opinions to help them find their way in life better. In the last 12 months, who have you sought advice from when it came to important decisions, for example about your family or work? (emotional)
2. In the last 12 months who has done little jobs and favors for you or helped you, for example in filling in forms or moving home? (instrumental)
3. In the past year, who have you turned to when you felt down and wanted someone to talk to? (emotional)
4. In the last 12 months, who have you borrowed money from? (instrumental)

5. In the past year, who have you spent your free time with or shared a hobby? ( social companionship)
6. In the past year who have you had disagreements or arguments with (e.g. about everyday affairs, money or property)? (conflict)
7. Who has let you know that you can rely on them (e.g. that they will always be there for you if you need help)? (emotional).

## References

- Herz, A. (2015). Relational constitution of social support in migrants' transnational personal communities. *Social Networks*, 40 (1), S. 64-74.
- Herz, A. (2012). *Strukturen transnationaler sozialer Unterstützung*. Springer Fachmedien Wiesbaden.

---

trim\_aaties

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

---

## Description

This is used in the background by dplyr methods, to maintain the alter-alter ties according to changes made to the ego and alter data levels.

## Usage

```
trim_aaties(object)
```

## Arguments

object            An egor object.

## Value

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

---

trim_alters	<i>Trims alters that are missing/ deleted from ego data.</i>
-------------	--

---

### Description

This is used in the background by dplyr methods, to maintain the alter ties according to changes made to the ego data level.

### Usage

```
trim_alters(object)
```

### Arguments

object            An egor object.

### Value

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

---

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

**Arguments**

<code>egos</code>	Data frame containing ego data (egos as cases)
<code>alters</code>	Data frame containing alters data (alters as cases), alters are separated by a variable containing an egoID.
<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>e.max.alters</code>	Maximum number of alters that are included in edge data.
<code>e.first.var</code>	Index or name of the first column in egos containing edge data.
<code>selection</code>	Character naming numeric variable indicating alters selection with zeros and ones.
<code>...</code>	additional arguments to <code>egor()</code> .

**Value**

An **egor** object is returned. It is a list of three data frames: (1) `ego`: dataframe of all egos and their attributes; (2) `alter`: dataframe of all alters; (3) `aatie`: dataframe of alter alter ties/ edges

**Examples**

```
path_to_alters_8.csv <- system.file("extdata", "alters_8.csv", package = "egor")
path_to_one_file_8 <- system.file("extdata", "one_file_8.csv", package = "egor")

# read data from disk
egos_8 <- read.csv2(path_to_one_file_8, row.names = 1)
alters_8 <- read.csv2(path_to_alters_8.csv, row.names = 1)

dy.first.var <- which(names(egos_8) == "X1.to.2")

# convert to egor object
twofiles_to_egor(
  egos = egos_8,
  alters = alters_8,
  e.max.alters = 8,
  e.first.var = dy.first.var)
```

---

 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,  
  normalise.node.sizes = TRUE,  
  edge.width.multiplier = 1,  
  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
normalise.node.sizes	Logical. If TRUE (default) node sizes are plotted using per network proportions rather than counts.
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 preceeded 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
par(mfrow = c(2,3))
vis_clustered_graphs(
  graphs[1:5]
)
par(mfrow = c(1,1))
```

---

weights.egor

[weights.egor\(\)](#) 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 [egor](#) object.  
`...` arguments to be passed to methods

*weights.egor*

47

**See Also**

[weights.survey.design](#)

# Index

- \* **analysis**
  - alts\_diversity\_count, 6
  - as.egor, 8
  - clustered\_graphs, 14
  - comp\_ei, 16
  - comp\_ply, 16
  - composition, 15
  - ego\_density, 22
  - egor\_vis\_app, 20
  - vis\_clustered\_graphs, 45
- \* **datasets**
  - aaties32, 3
  - allbus\_2010\_simulated, 4
  - alters32, 4
  - egor32, 19
  - egos32, 20
  - gss2004, 25
  - transnat, 41
- \* **ego-centered**
  - activate.egor, 3
  - alts\_diversity\_count, 6
  - as.egor, 8
  - clustered\_graphs, 14
  - comp\_ei, 16
  - comp\_ply, 16
  - composition, 15
  - ego\_density, 22
  - egor\_vis\_app, 20
  - EI, 23
  - make\_egor, 28
  - read\_egonet, 34
  - vis\_clustered\_graphs, 45
- \* **import**
  - onefile\_to\_egor, 29
  - read\_egonet, 34
  - twofiles\_to\_egor, 43
- \* **network**
  - activate.egor, 3
  - alts\_diversity\_count, 6
  - as.egor, 8
  - clustered\_graphs, 14
  - comp\_ei, 16
  - comp\_ply, 16
  - composition, 15
  - ego\_density, 22
  - egor\_vis\_app, 20
  - EI, 23
  - make\_egor, 28
  - vis\_clustered\_graphs, 45
- \* **random**
  - make\_egor, 28
- \* **sna**
  - EI, 23
  - [.egor (subset.egor), 36
  - aaties32, 3
  - aaties\_by\_ego (helper), 26
  - activate.egor, 3
  - activate.egor(), 37
  - allbus\_2010\_simulated, 4
  - alter\_design, 5
  - alter\_design<- (alter\_design), 5
  - alter\_df (transnat), 41
  - alters32, 4
  - alters\_by\_ego (helper), 26
  - alts\_diversity\_count, 6
  - alts\_diversity\_entropy (alts\_diversity\_count), 6
  - append\_cols (append\_egor), 7
  - append\_egor, 7
  - append\_rows (append\_egor), 7
  - as.egor, 8
  - as.igraph.egor (as\_igraph), 10
  - as.network.egor (as\_igraph), 10
  - as\_aaties\_df (as\_tibble.egor), 12
  - as\_aaties\_survey (as\_tibble.egor), 12
  - as\_alters\_df (as\_tibble.egor), 12
  - as\_alters\_survey (as\_tibble.egor), 12
  - as\_egos\_df (as\_tibble.egor), 12



`as_egos_survey` (`as_tibble.egor`), 12  
`as_igraph`, 10  
`as_nested_egor` (`helper`), 26  
`as_network` (`as_igraph`), 10  
`as_survey()`, 12  
`as_survey.egor` (`as_tibble.egor`), 12  
`as_tibble()`, 10, 12  
`as_tibble.egor`, 12

`clustered_graphs`, 14, 45, 46  
`comp_ei`, 16  
`comp_ei()`, 24, 25  
`comp_ply`, 16  
`composition`, 15  
`count_dyads`, 17  
`create_edge_names_wide` (`helper`), 26

`data.frame()`, 35, 36  
`din_page_dist` (`helper`), 26  
`dyad.poss` (`helper`), 26  
`dyads_possible_between_groups` (`helper`), 26

`ego_constraint`, 21  
`ego_density`, 22  
`ego_design`, 22  
`ego_design<-` (`ego_design`), 22  
`ego_df` (`transnat`), 41  
`egor`, 5, 9, 23, 38, 46  
`egor` (`as.egor`), 8  
`egor()`, 30, 34, 36, 37, 40, 44  
`egor-package-doc`, 18  
`egor32`, 19  
`egor_options`, 19  
`egor_vis_app`, 20  
`egos32`, 20  
`EI`, 23  
`extract_egos_and_return`, 25

`gss2004`, 25

`has_ego_design` (`ego_design`), 22  
`helper`, 26

`igraph::constraint`, 21

`layout_egogram`, 28  
`list`, 9, 20, 23  
`list()`, 36

`make_egor`, 28  
`onefile_to_egor`, 29

`plot.egor` (`plot_egograms`), 30  
`plot_ego_graphs` (`plot_egograms`), 30  
`plot_egograms`, 30  
`plot_egor` (`plot_egograms`), 30  
`print.egor` (`summary.egor`), 38

`read_egonet`, 34  
`read_egoweb` (`threefiles_to_egor`), 38  
`read_openeddi` (`threefiles_to_egor`), 38  
`regex`, 29  
`return_results`, 35  
`rowlist`, 35

`sanitize.wide.edges` (`helper`), 26  
`shiny`, 20  
`shiny::shinyApp()`, 20  
`srvyr::as_survey_design()`, 9, 23  
`strip_ego_design` (`ego_design`), 22  
`subset()`, 36  
`subset.egor`, 36  
`summary.egor`, 38

`tbl_svy`, 10, 12  
`threefiles_to_egor`, 38  
`tibble`, 10  
`tibble()`, 35, 36  
`transnat`, 41  
`trim_aaties`, 42  
`trim_alters`, 43  
`twofiles_to_egor`, 43

`vis_clustered_graphs`, 14, 45

`weights.egor`, 46  
`weights.egor()`, 46  
`weights.survey.design`, 47