Package 'sdam'

October 14, 2022

Type Package

Title Social Dynamics and Complexity in the Ancient Mediterranean

Description Provides digital tools for performing analyses within Social Dynamics and complexity in the Ancient Mediterranean (SDAM), which is a research group based at the Department of History and Classical Studies at Aarhus University.

Version 1.1.4

Date 2022-09-02

Depends R (>= 4.1.0)

Imports grImport2, multiplex

Suggests httr, rjson, grid, knitr, rmarkdown, multigraph

Author Antonio Rivero Ostoic [aut, cre],
Adela Sobotkova [ctb],
Vojtech Kase [ctb],
Petra Hermankova [ctb]

Maintainer Antonio Rivero Ostoic <jaro@cas.au.dk>

URL https://github.com/sdam-au/sdam/

BugReports https://github.com/sdam-au/sdam/issues/

Repository CRAN

NeedsCompilation no

VignetteBuilder knitr

Date/Publication 2022-09-05 07:30:02 UTC

R topics documented:

`sdam-package` *Social Dynamics and Complexity in the Ancient Mediterranean*

Description

Provides digital tools for performing analyses within Social Dynamics and complexity in the Ancient Mediterranean (SDAM), which is a research group based at the Department of History and Classical Studies at Aarhus University.

Details

Package:	sdam
Type:	Package
Version:	1.1.4
Date:	2 September 2022
License:	CC BY-SA 4.0

The `"sdam"` package is a suite of tools for performing analyses in the history of the Mediterranean world and its neighbouring regions in the antiquity period before the Middle Ages. Currently, it is possible to access data of writing material from the Epigraphic Database Heidelberg API with function `get.edh` and the wrapper function `get.edhw`. Most of the epigraphic data, at least until 10-11-2020, is available in the `EDH` dataset, which can be transformed in diverse ways by using function `edhw` and function `cln` for re-encoding and cleaning portions of text in inscriptions in the `EDH` dataset.

With function `prex`, there is also the possibility to compute probabilities of existence of inscriptions and other artefacts with either the aoristic sum or count matching for observations for different periodization options. Function `plot.dates` allow visualizing interval time events that can be adjusted by the internal function `dts` as illustrated in a vignette.

request function allows, with user authentication, performing different types of HTTP requests aimed to DEiC's servers in https://sciencedata.dk or another cloud repository with a customized URL address.

A plotting function is found in plot.map that allows visualizing cartographical maps of ancient Roman provinces that are part of the EDH dataset. It relies other datasets from the package that are related to the Roman world in rp, rpd, rpmp, rpmcd, rpcp, and retn for a transport network with terrestrial and maritime main routes.

Similarity by simple matching among column vectors in a table or data frame is achieved by function simil for making analyses of relations between e.g. assemblages and artefacts. Function edhwpd is to organize the EDH dataset per province and dates based on a similarity measure, and it is the basis for function rmids to compute values of missing dates with a restricted imputation on data subsets of artefacts.

Author(s)

Author: Antonio Rivero Ostoic [aut, cre], Adela Sobotkova [ctb], Vojtech Kase [ctb], Petra Hermankova [ctb]

Maintainer: Antonio Rivero Ostoic <jaro@cas.au.dk>

References

Epigraphic Database Heidelberg – Data Reuse Options, (Online; retrieved on 16 June 2019). URL https://edh-www.adw.uni-heidelberg.de/data

See Also

Datasets in "sdam" package

Re-encoding people in the EDH dataset

Dates and missing dating data

Cartographical maps and networks

Articles using "sdam"

Release candidate version

Examples

```
data(package="sdam")
```

---

cln *Clean and re-encode a vector, list or dataframe*

Description

A function to re-encode Greek (and other) characters and to remove symbols.

Usage

```
cln(x, level, what, na.rm, case, repl, unlist)
```

Arguments

x	a vector, list or dataframe
level	optional clean level, either 0 for no-clean, default 1 to most strict 9 (see details)
what	additional characters to clean (optional)
na.rm	remove entries with NA data? (optional and logical)
case	case for text 1 for 1st uppercase, code2 for lowercase, code3 for uppercase (optional)
repl	data frame with text to replace (optional)
unlist	return a vector? (optional and logical, for vector input)

Details

This function is meant to re-encode Greek (and other) characters in the EDH set given either as list format, vector, or a dataframe produced with function edhw for example.

By default, the symbols "?" "*" "+" placed at the end of each record are removed after the re-encoding. However, when level is 0 only re-encoding is performed, and level 2 is either to force an extra iteration in the re-encoding, to remove extra spaces, or what is in what at the end of a record when clean what is invoked. With level 9 all content after an opening parenthesis is removed with all the consequences for the input text.

With repl, is possible to replace a list of text in two columns, for 'text to replace' and for 'text that replaces'.

Disabling option unlist returns a vector in case that x is also a vector; otherwise, it returns a list with the two versions of the input.

Value

Depending on the input, a vector, list or dataframe.

Warning

Encoding more than once the same input requires re-starting the console; otherwise, the re-encoding is not complete.

Author(s)

Antonio Rivero Ostoic

See Also

edhw, get.edh, edhwpd, cs

Examples

```
# clean Greek characters
cln("Caesar?*+")
```

---

cs *Change case in a text*

Description

A function to change letters to uppercase and lowercase in a given text.

Usage

```
cs(xz, level = 1, case = 1, flgdf = FALSE, na.rm = FALSE)
```

Arguments

xz	input text
level	optional clean level, either 0 for no-clean, default 1 to most strict 9 (see cln)
case	change case: 1 first letter uppercase and rest lowercase, 2 all letters lowercase, 3 all letters uppercase
flgdf	is xz a data frame?
na.rm	remove entries with NA data? (optional and logical)

Details

This is a convenient function to change letters to uppercase and lowercase in a text for argument 'case' in function cln, and it inherits values from this function for 'level', 'flgdf', and 'na.rm'.

Value

Input text with defined case.

Author(s)

Antonio Rivero Ostoic

See Also

cln

Examples

```
cs("Caesar?*+", level=2, case=3)
```

dts *Converting dates into a numerical format*

Description

A function for converting different types of dates into a numerical format.

Usage

```
dts(x, cent, sep, last)
```

Arguments

x	scalar, vector or list with dates to format
cent	use centuries? (optional and logical)
sep	separator, default " to " (only for cent)
last	take last input value in x? (optional and logical)

Details

When dating data has a character format like when involving AD, BC, BCE, C.E., etc., or even centuries like Cent., it is many times convenient to convert these type of dating data into a numerical format for a further computation.

In case that the input data has two or more plausible dates, then the outcome by default takes the first value of the input; otherwise the last date with option last.

dts also accepts dates involving centuries with the cent option, and in this case, it is possible to specify a separator of the century endpoints in sep or use " to " as the default separator.

For dates having character format, then hyphens are regarded as separators of the plausible dates in x.

Value

Dating data with a numerical format.

Note

Dating data with unknown year notations produce NA as output value.

Author(s)

Antonio Rivero Ostoic

See Also

plot.dates, prex.

Examples

```
# negative first value
dts("58 BC - 30 AD")

# positive second value
dts("58 BC - 30 AD", last=TRUE)

# use century notation
dts("15th Cent. AD", cent=TRUE)
```

---

edhw *Wrapper function for manipulation of the EDH dataset*

Description

A function to obtain variable data and perform transformations on the EDH dataset.

Usage

```
edhw(x = "EDH", vars, as = c("df", "list"), type = c("long", "wide", "narrow"),
     split, select, addID, limit, id, na.rm, ldf, province, gender, rp, ...)
```

Arguments

x	a list object name with fragments of the EDH dataset (optional)
vars	vector of variables of interest from x; if x=NULL, the entire EDH dataset is taken (optional)
as	format to return the output; either as a "list" or a data frame "df" object.
type	type format of data frame; either "long" or "wide" ("narrow" not yet implemented)
split	divide the data into groups by id? (optional and logical)
select	vector with "people" variables (optional)
addID	add identification to the output? (optional and logical)
limit	integer or vector to limit the returned output. Ignored if id is specified (optional)
id	select only hd_nr records (optional, integer or character)
na.rm	remove entries with NA data? (optional and logical)
ldf	is x list of data frames? (optional and logical)
province	name or abbreviation of Roman province in EDH as in rp dataset
gender	gender of people in EDH: male or female
rp	customized list of Roman provinces as in rp dataset
...	optional arguments if needed.

Details

This is an interface to extract attribute variables from the EDH dataset attached to this package either as a built-in dataset or as external data. EDH dataset is a built-in data set of Latin epigraphy retrieved from the Epigraphic Database Heidelberg API repository where epigraphs or inscriptions in this dataset are recorded in a list object of 84701 items (until 10-11-2020) with at least one of the following 47 (or more) attribute names in the list:

"ID", "commentary", "fotos", "country", "depth", "diplomatic_text", "edh_geography_uri", "findspot", "findspot_ancient", "findspot_modern", "geography", "height", "id", "language", "last_update", "letter_size", "literature", "material", "military", "modern_region", "not_after", "not_before", "people" (which is a list with: "person_id", "nomen", "cognomen", "praenomen", "name", "gender", "status", "tribus", "origo", "occupation", "age: years", "age: months", "age: days"), "present_location", "province_label", "religion", "responsible_individual", "social_economic_legal_history", "transcription", "trismegistos_uri", "type_of_inscription", "type_of_monument", "uri", "width", "work_status", and "year_of_find".

The input in x, however, can be fragments of the EDH dataset or from the Epigraphic Database Heidelberg API obtained by functions get.edh or get.edhw with the "rjson" format, or transformed data organized, for example, by provinces. When x is explicit, it must be at least a list object with a comparable structure to the EDH dataset. Argument ldf is a flag when the input in x is a created list of data frames that are organised by variables rather than by records as in the EDH dataset. The return of the output is either as a list with list or by default as a data frame with option df.

The extraction from EDH is typically through argument vars in the function, and in case that vars is missing, then it takes all entries in x. Ad hoc arguments are the EDH entries province and gender for entering a Roman province and people's gender in x as a data frame; otherwise, these arguments are ignored. When province is used, it is possible to refer to a customized list of provinces with argument "rp"; otherwise, dataset rp is the default where names and abbreviations are accepted.

By default, this wrapper returns a list object with or without a numerical 'ID' identification provided by the addID argument. When the output is a data frame, the ordering of the variables is alphabetically and, if desired, it is possible to remove missing data from the output by activating na.rm and work with complete cases.

Arguments id and limit serve to reduce the returned output either to some Epigraphic Database number or to numbers, which are specified by hd_nr, or else by limiting the amount of the returned output. limit here is like the limit argument of function get.edh, but in this case the offset can be specified as a sequence. While limit is a faster way to get to entries in the EDH dataset, argument id is for referring to precisely one or more hd_nrs in the Epigraphic Database Heidelberg API.

Component "people" is a separated list in the EDH dataset, and it should be considered as a separate case from the rest of the variables. In the case that the output is a data frame, the default output is a 'long' type table; that is records can appear in different rows and each variable is assigned into a single column, and with this option is possible to select "people" variables like gender and origin. When choosing people variables with select and a data frame output, then "people" attribute must be in vars.

By setting "wide" in type, it is possible to place the different people from a single entry column by column in the data frame and each record has a single row. Finally, argument split allows for dividing the data in the data frame into groups by 'id', which corresponds to the HD number of inscription in the EDH dataset.

Value

A list or a data frame with a long or wide format, depending on the input arguments.

Argument `province` with no `vars` returns a list of lists.

Warning

`EDH` is a built-in dataset in the development and legacy version of the package but, because of its size, re not part of the CRAN distribution. Functions `edhw` and `edhwpd` download `EDH` from another repository in References.

Note

`Warning` messages are given for the `EDH` dataset as the input, and when choosing the `province` argument alone.

Author(s)

Antonio Rivero Ostoic

References

Epigraphic Database Heidelberg – Data Reuse Options, (Online; retrieved on 16 June 2019). URL https://edh-www.adw.uni-heidelberg.de/data

https://edh-www.adw.uni-heidelberg.de/data/api (database retrieved on November 2020)

https://github.com/sdam-au/sdam/tree/master/data

https://github.com/mplex/cedhar/tree/master/pkg/sdam/data

See Also

`get.edh`, `get.edhw`, `rp`, `edhwpd`, `prex`, `plot.dates`, `cln`, `rjson`

Examples

```
## Not run:
# load dataset
data(EDH)

# make a list for three variables in 'EDH' for first 4 entries
edhw(vars=c("type_of_inscription", "not_after", "not_before"), limit=4 )

# as before, but also select 'gender' from 'people'
edhw(vars=c("people", "not_after", "not_before"), select="gender", limit=4 )
## End(Not run)
```

edhwpd *Organize EDH dataset province and dates by similarity*

Description

Wrapper function to organize EDH dataset province and dates by simple match similarity.

Usage

```
edhwpd(x = "EDH", vars, province, dates, clean, ...)
```

Arguments

x	EDH dataset, or fragments of, or database via API (optional, list)
vars	vector with variables or attributes chosen from x
province	Roman province abbreviation as in rp
dates	vector with TAQ and TPQ (optional)
clean	whether to remove special characters in text (optional and logical)
...	additional arguments if needed

Details

This wrapper function aims to organize data per Roman province and date by simple match similarities among inscriptions in the EDH dataset. As with function edhw, it is an interface to extract attribute variables in vars from the EDH or similar dataset if x is not specified.

The Roman Empire province is the abbreviation used in the value given by function get.edh and which is in rp dataset.

Argument dates is optional to specify the variables for time intervals (TAQ and TPQ) that in EDH are not_after and not_before, but other datasets may have different names for the endpoints of the timespan. Another dependence with this function is from package "multiplex" to find clusters of items having similar characteristics as co-occurrence relations and for removing isolated items from the system of relations.

Argument clean applies function cln to the province data frame with the chosen variables to remove special characters such as ?*+ and, if needed, re-encode the text.

The output is a list of data frames with similar arguments by descending matches. The records with one or less similarity matches (or having NA attribute values) are placed in the last data frame of the list.

Value

A EDH class object with the province and the number of records with a list of data frames organised by components where the first one has records having most common attribute variables, whereas the last component is a dataframe with records having least common attribute variables.

Note

This function depends on EDH that is a built-in dataset in the development and legacy version of the package but, because of its size, for the CRAN distribution it downloads from another repository in References.

Author(s)

Antonio Rivero Ostoic

References

https://edh-www.adw.uni-heidelberg.de/data/api (database retrieved on November 2020)

https://github.com/sdam-au/sdam/tree/master/data

https://github.com/mplex/cedhar/tree/master/pkg/sdam/data

See Also

edhw, rmids, rp, get.edh, cln

Examples

```
## Not run:
# load dataset
data(EDH)

# extract province & dates with a single variable attribute from EDH
edhwpd(vars="type_of_inscription", province="Rom", dates=c("not_after", "not_before"))
## End(Not run)
```

---

get.edh *Get data from the Epigraphic Database Heidelberg API*

Description

A function to obtain data from the Epigraphic Database Heidelberg REST like API repository.

Usage

```
get.edh(search = c("inscriptions", "geography"), url =
                "https://edh.ub.uni-heidelberg.de/data/api", hd_nr,
                province, country, findspot_modern, findspot_ancient,
                year_not_before, year_not_after, tm_nr, transcription,
                type, bbox, findspot, pleiades_id, geonames_id,
                offset, limit, maxlimit = 4000, addID, printQ)
```

Arguments

search	whether the search is on inscriptions *or* on geography.
url	open data repository API
hd_nr	HD number of inscription
province	ancient Roman province name
country	actual country name
findspot_modern	actual location name findspot
findspot_ancient	ancient location name findspot
year_not_before	year, not before (integer, BC years are negative)
year_not_after	year, not after (integer, BC years are negative)
tm_nr	trismegistos ID (integer)
transcription	automatic leading and trailing truncation (brackets are ignored)
type	type of inscription (case insensitive)
bbox	bounding box with character format bbox = "minLong, minLat, maxLong, maxLat"
findspot	level of village, street etc. (add leading and/or trailing)
pleiades_id	Pleiades identifier of a place (integer)
geonames_id	Geonames identifier of a place (integer)
offset	clause to specify which row to start from retrieving data (optional and integer)
limit	clause to limit the number of results (optional and integer)
maxlimit	maximum limit of the query (integer, default 4000)
addID	add identification to the output? (optional and logical)
printQ	print also query? (optional and logical)

Details

Since with the inscriptions option the id "component" of the output list is not with a numeric format, then the function adds an ID at the beginning of the list with the identifier with a numerical format. hd_nr has not the same value as ID nor id. In case you want to grab several items from the Epigraphic Database Heidelberg API use function get.edhw.

A list with the of valid values from the EDH API for the ancient Roman provinces that are also available in dataset rp are

"Ach"	Achaia	"Cor"	Corsica	"Mes"	Mesopotamia
"Aeg"	Aegyptus	"Cre"	Creta	"MoI"	Moesia inferior
"Aem"	Aemilia (Regio VIII)	"Cyp"	Cyprus	"MoS"	Moesia superior
"Afr"	Africa Proconsularis	"Cyr"	Cyrene	"Nar"	Narbonensis
"AlC"	Alpes Cottiae	"Dac"	Dacia	"Nor"	Noricum
"AlG"	Alpes Graiae	"Dal"	Dalmatia	"Num"	Numidia
"AlM"	Alpes Maritimae	"Epi"	Epirus	"PaI"	Pannonia inferior

"AlP"	Alpes Poeninae	"Etr"	Etruria (Regio VII)	"PaS"	Pannonia superior
"ApC"	Apulia et Calabria (Regio II)	"Gal"	Galatia	"Pic"	Picenum (Regio V)
"Aqu"	Aquitania	"GeI"	Germania inferior	"Rae"	Raetia
"Ara"	Arabia	"GeS"	Germania superior	"ReB"	Regnum Bospori
"Arm"	Armenia	"HiC"	Hispania citerior	"Rom"	Roma
"Asi"	Asia	"Inc"	Provincia incerta	"Sam"	Samnium (Regio IV)
"Ass"	Assyria	"Iud"	Iudaea	"Sar"	Sardinia
"Bae"	Baetica	"LaC"	Latium et Campania (Regio I)	"Sic"	Sicilia, Melita
"Bar"	Barbaricum	"Lig"	Liguria (Regio IX)	"Syr"	Syria
"Bel"	Belgica	"Lug"	Lugdunensis	"Thr"	Thracia
"BiP"	Bithynia et Pontus	"Lus"	Lusitania	"Tra"	Transpadana (Regio XI)
"BrL"	Bruttium et Lucania (Regio III)	"LyP"	Lycia et Pamphylia	"Tri"	Tripolitania
"Bri"	Britannia	"MaC"	Mauretania Caesariensis	"Umb"	Umbria (Regio VI)
"Cap"	Cappadocia	"MaT"	Mauretania Tingitana	"Val"	Valeria
"Cil"	Cilicia	"Mak"	Macedonia	"VeH"	Venetia et Histria (Regio X)

And the valid values for `country` entries are abbreviated country names where the inscription was located.

"ad"	Andorra	"gr"	Greece	"pl"	Poland
"al"	Albania	"hr"	Croatia	"pt"	Portugal
"am"	Armenia	"hu"	Hungary	"rks"	Kosovo
"at"	Austria	"il"	Israel	"ro"	Romania
"az"	Azerbaijan	"iq"	Iraq	"rs"	Serbia
"ba"	Bosnia and Herzegovina	"it"	Italy	"ru"	Russia
"be"	Belgium	"jo"	Jordan	"sa"	Saudi Arabia
"bg"	Bulgaria	"kg"	Kyrgyzstan	"sd"	Sudan
"ch"	Switzerland	"kz"	Kazakhstan	"se"	Sweden
"cy"	Cyprus	"lb"	Lebanon	"si"	Slovenia
"cz"	Czech Republic	"li"	Liechtenstein	"sk"	Slovakia
"de"	Germany	"lu"	Luxembourg	"sm"	San Marino
"dk"	Denmark	"ly"	Libyan Arab Jamahiriya	"sy"	Syrian Arab Republic
"dz"	Algeria	"ma"	Morocco	"tj"	Tajikistan
"eg"	Egypt	"mc"	Monaco	"tn"	Tunisia
"es"	Spain	"md"	Moldova	"tr"	Turkey
"fr"	France	"me"	Montenegro	"ua"	Ukraine
"gb"	United Kingdom	"mk"	Macedonia	"uz"	Uzbekistan
"ge"	Georgia	"mt"	Malta	"va"	Vatican City State
"gi"	Gibraltar	"nl"	Netherlands	"ye"	Yemen

Value

A list object with at least one the following items:

"commentary"

"fotos"

"country"

"depth"

"diplomatic_text"

"edh_geography_uri"

"findspot"

"findspot_ancient"

"findspot_modern"

"geography"

"height"

"id"

"language"

"last_update"

"letter_size"

"literature"

"material"

"military"

"modern_region"

"not_after"

"not_before"

"people" This item is another list with at least one the following items:

"person_id"
"nomen"
"cognomen"
"praenomen"
"name"
"gender"
"status"
"tribus"
"origo"
"occupation"
"age: years"
"age: months"
"age: days"

"present_location"

"religion"

"province_label"

"responsible_individual"

"social_economic_legal_history"

"transcription"

"trismegistos_uri"

"type_of_inscription"

"type_of_monument"

"uri"

"width"

"work_status"

"year_of_find"

"ID" (Optional), only if addID is set to TRUE.

The query is also printed if specified by printQ.

Warning

For queries having more than 4000 records, the server can produce a timeout break to be handled by offset.

Note

This function requires "rjson", and is for the [EDH] database [API] at the URL in references starting in year January 2022, and changes in the URL should be updated with the url option.

Search options "photos" and "bibliography" are not supported.

Author(s)

Antonio Rivero Ostoic

References

https://edh.ub.uni-heidelberg.de/data/api

See Also

get.edhw, edhw, edhwpd, rp, plot.map, simil, rjson

Examples

```
## Not run:
# get inscriptions from EDH database API
get.edh(findspot_modern="madrid")
## End(Not run)
```

`get.edhw` *Wrapper to get data from the Epigraphic Database Heidelberg API*

Description

A wrapper function to obtain data from the Epigraphic Database Heidelberg REST like API repository.

Usage

```
get.edhw(file = NULL, hd_nr, ...)
```

Arguments

`file`	JSON file with EDH data (optional)
`hd_nr`	HD number of inscriptions
`...`	additional arguments

Details

This wrapper function aims to obtain sample data from the Epigraphic Database Heidelberg API repository by their HD numbers or a file with a valid format JSON can be specified in `file`.

In any case, the JSON output becomes a list object with the `rjson` package.

Value

A list of lists object with the items described in `get.edh`.

Note

Large samples can take a lot of time.

Author(s)

Antonio Rivero Ostoic

References

https://edh.ub.uni-heidelberg.de/data/api

See Also

get.edh, simil, rjson

Examples

```
## Not run:
# get 10 records from EDH API data
get.edhw(hd_nr=1:10)
## End(Not run)
```

---

plot.dates *Plot interval dates*

Description

A function to plot interval dates with different forms.

Usage

```
## S3 method for class 'dates'
plot(x, y, type = c("ts", "mp", "rg"), taq, tpq, id, out,
    col, cex, lwd, lty, pch, main = NULL, xlab = NULL, ylab = NULL,
    xlim = NULL, axes = TRUE, alpha, file = NULL, ...)
```

Arguments

x	dataset as a data frame object of variables and observations.
y	vector of identifiers (optional)
type	Type of date format to plot: ts timespans with endpoints mp mid points and range rg range only
taq	timespan endpoint *terminus ante quem* (TAQ)
tpq	timespan endpoint *terminus post quem* (TPQ)
id	IDs as variable or rownames in dataset x
out	integer or vector with number of outliers to omit (first entry id for latest date)
col	color of pch
cex	size of pch
lwd	width of time interval segments
lty	shape of time interval segments

pch	symbol for taq and tpq
main	plot's main tile
xlab	plot's x label
ylab	plot's y label
xlim	plot's x limits
axes	plot's axes (logical)
alpha	alpha transparency for time interval segments
file	path to produce a file with a PDF format (optional)
...	additional optional parameters

Details

This plot function is for time interval segments given in the dataset x, which is given as a dataframe or as a "tibble" class object.

Value

A graphical plot.

Note

If x is NULL, then EDH dataset is taken by default.

Author(s)

Antonio Rivero Ostoic

See Also

dts, get.edh, edhw, prex, tibble

Examples

```
## Not run:
# first 100 entries in the EDH dataset (default)
EDHdates <- edhw(vars=c("not_after", "not_before"), as="df", limit=100)

# timespans
plot.dates(EDHdates, taq="not_before", tpq="not_after")
## End(Not run)
```

plot.map *Plot cartographical maps*

Description

A function to plot cartographical maps of the Roman world and Mediterranean region.

Usage

```
## S3 method for class 'map'
plot(x = NULL, type = c("plain", "rp", "si", "tetra", "med"), settl, roads, shipr,
         main, cap, date, name, fsize, fcol, fsize2, fcol2, xd, yd, new, ...)
```

Arguments

x	acronym of ancient Roman province or Italian region (see "rp")
type	Type of cartographical map: plain most of Europe and land around the Mediterranean rp ancient Roman provinces si Senatorial-Imperial provinces tetra First Tetrarchy med Mediterranean region
settl	display settlements? (optional and logical, for cartographical map)
roads	display terrestrial routes? (optional and logical, for cartographical map)
shipr	display shipping routes? (optional and logical, for cartographical map)
main	plot's main title (optional)
cap	display caption? (optional and logical, for provinces)
date	display date? (optional and logical, for provinces)
name	display map title name? (optional and logical, for provinces)
fsize	font size in main title (optional)
fcol	font color in main title (optional)
fsize2	font size in date (optional)
fcol2	font color in date (optional)
xd	*x* positioning for the date (optional)
yd	*y* positioning for the date (optional)
new	whether the plotted map has superimposed graphics (optional)
...	additional optional parameters

Details

This plot function is for creating cartographical maps of ancient provinces and Italian regions of the Roman Empire around the year AD 117. The input data `x` can be a character vector, but this is intended for a recording output. By default, the argument's `name` and `cap` are set to `TRUE` while the `date` is set to `FALSE`; however, the argument `main` prevails over `name`.

The `type` argument allows plotting cartographical maps related to the Roman Empire and the Mediterranean basin as specified in `rpmp` and `rpmcd` datasets. In the cartographical maps, settlements are displayed as circles while squares are for military forts, while terrestrial and maritime routes are given as solid paths with different colours. Shapes of places and routes are specified in the `retn` dataset.

Dataset `retn` is a list of lists with specifications to plot different cartographical maps of the Roman Empire and the Mediterranean with transport network including settlements, roads, and shipping routes. This list of lists object has the shape data in different slots for 4 cartographical maps of the Roman Empire with names `rcoast` for a plain map, `rpcoast` for a map with provinces, `rpsi` for a map with senatorial and imperial provinces, and `rptetra` for a tetrarchy map. These options for cartographical maps in the Mediterranean are for both the classical and the late antiquity periods. Three components in `retn` dataset have coordinates for settlements `nds`, roads `rds`, and shipping routes `srs` for these maps. In addition, the dataset has a cartographical map of the Mediterranean in `med` where settlements and transport network is yet to complete.

Dataset `rpmp` is a list with specifications to plot cartographical maps of ancient Roman provinces and Italian regions. This list of lists object has 59 Roman provinces and Italian regions in year 117AD, and where `names(rpmp)` gives the province acronyms according to `rp` dataset. Each province in `rpmp` has a two-length list with the province name and the shape data for a cartographical map in different slots.

Value

A plot of a cartographical map for the Roman world with a title name, and a caption with an approximate province establishment date.

Warning

`rpmp` and `retn` are built-in datasets in the development and legacy version of the package but, because of its size, are not part of the CRAN distribution, which means that they are downloaded from another repository.

Note

Positions for caption and date are for a PDF output and the rendering may vary for browser displays.

Author(s)

Antonio Rivero Ostoic

References

https://github.com/sdam-au/sdam/tree/master/data

https://github.com/mplex/cedhar/tree/master/pkg/sdam/data

See Also

rpmp, rpmcd, retn, get.edh

Examples

```
# Roman Empire transport network
plot.map(roads=TRUE, shipr=TRUE)


# Roman province of Aegyptus
plot.map(x="Aeg")
```

---

prex *Compute probabilities of existence*

Description

A function to compute probabilities of existence of artefacts related events.

Usage

```
prex(x, type = c("aoristic", "mp", "other"), taq, tpq, vars,
     bins = NULL, cp, weight = 1, DF, out, plot = FALSE, main = NULL,
     ylim, keep, ...)
```

Arguments

x	list or data frame object of variables and observations.
type	Type of date format to compute or plot: aoristic aoristic sum mp mid points and range other
taq	timespan endpoint *terminus ante quem* (TAQ)
tpq	timespan endpoint *terminus post quem* (TPQ)
vars	boundaries of existence of x (vector for timespan endpoints)
bins	length of the break (integer and optional)
cp	Chronological phase: "bin5" five-bins from the antiquity period "bin8" eight-bins from the antiquity period list with a customized chronological phase
weight	value to observations (optional)
DF	return also data frame with observations? (optional and logical)
out	number of outliers to omit (integer or vector where first entry id for latest date)

plot	plot the results?
main	plot's main title (optional)
ylim	limit in y-axis (optional, for plot)
keep	for mp, keep variables in output? (optional and logical)
...	additional optional parameters

Details

Currently, the probability of existence of the observations is the *aoristic sum* computed across events for portions of time delimited by a TAQ in taq and TPQ in tpq, which are endpoints from the stance of the timespan. Alternatively, these two boundaries of existence of x are specified in vars.

In case the bins are set to NULL, then the time breaks take the chronological periods in cp, which by default is "bin5" or five-periods for the EDH dataset, which are Arch (Archaic), Class (Classical), Hell (Hellenistic), Rom, (Roman), and Byz (Byzantine). Another built-in option is "bin8" for eight chronological periods where the Roman period is divided into ERom (Early Roman), MRom (Middle Roman), and LRom (Late Roman) while the Byzantine period is divided into EByz (Early Byzantine) and MByz (Middle Byzantine). However, option cp is open for other periodization models as long as the categories of time blocks are components of a list object.

Value

A data frame with values according to the chosen either bins or cp. If plot is specified, a bar plot with bars of outcomes.

Note

When aoristic is set to FALSE, then a simple matching of only TAQ and TPQ is computed from x.

Author(s)

Antonio Rivero Ostoic

References

Crema, E. "Modelling temporal uncertainty in archaeological analysis," *J Archaeol Method Theory*, 19:440–461. (2012). (for aoristic sum)

Bevan, *et al*. "Measuring chronological uncertainty in intensive survey finds: A case study from Antikythera, Greece," *Archaeometry*, 55, 2, 312–328. (2013). (default chronological periods)

See Also

edhw, plot.dates, dts.

Examples

```
## Not run:
# first 100 entries in the EDH dataset (default)
EDHdates <- edhw(vars=c("not_after", "not_before"), as="df", limit=100)

# compute aoristic sum with five-periods
prex(x=EDHdates, taq="not_before", tpq="not_after", cp="bin5")

# compute aoristic sum with 75 year span
prex(x=EDHdates, taq="not_before", tpq="not_after", bins=75, plot=TRUE)
## End(Not run)
```

`request` *Perform an HTTP request*

Description

A function to perform an HTTP request to https://sciencedata.dk or other server.

Usage

```
request(file, URL = "https://sciencedata.dk", method = c("GET", "POST", "PUT", "DELETE"),
     anonymous = FALSE, cred = NULL, path = "/files", subdomain = NULL, force = FALSE,
        rm.file, ...)
```

Arguments

`file`	the requested file
`URL`	protocol and domain of the url
`method`	the http verb for the object
`anonymous`	unauthenticated user?
`cred`	vector with username and password as authentication credentials
`path`	path to add to the url (optional)
`subdomain`	subdomain to add to the url (optional)
`force`	force remote file overwriting? (optional and logical)
`rm.file`	remove file in local machine? (optional and logical)
`...`	further parameters if required

Details

`request` is an HTTP request, first aimed to interact with DEiC's (Danish e-Infrastructure Cooperation) RESTful APIs at https://sciencedata.dk; however, it is possible to specify the URL path and subdomain if necessary.

DEiC's https://sciencedata.dk servers have different types of folders and both *personal* and *shared* folders require authentication with credentials.

The *path* to the shared folders where the files are located must be specified with the `path` argument. However, for personal folders the `file` argument that includes the path information. Many times, DEiC's `https://sciencedata.dk` places the data on a *subdomain*, and for some methods like `PUT` it is required to specify the subdomain as well.

When a file already exists on the remote server, there is a prompt question for overwriting the file when the `PUT` method is invoked, and by activating argument `force` we can prevent confirmation and replace the file. Method `POST` is not yet supported.

In case that accessing the server requires basic authentication, then package "`tcltk`" is required to input the credentials with a widget prompt. However, there is a `cred` argument for performing a basic authentication without a prompt, and public folders can be accessed without credentials with the option of `anonymous` user.

Value

Depends on the method, an action on the server site.

A *Response* message is returned when the method is `PUT` with the URL and items `Date`, `Status`, `Content-Type`.

Note

This function requires "`httr`", and aliases for `request` are `sddk()` and `SDDK()`.

Author(s)

Antonio Rivero Ostoic

References

https://sciencedata.dk/sites/developer/ (retrieved on January 2020)

`https://sciencedata.dk`

`https://www.deic.dk/`

See Also

`httr`, `tcltk`

Examples

```
## get a public file from remote server as anonymous user
## Not run:
request("filename.extension", method="GET", anonymous=TRUE)
## End(Not run)

## put a file in remote server
## Not run:
request("filename.extension", method="PUT")
## End(Not run)

## put an existing file in remote server and force overwriting
```

```
## Not run:
request("filename.extension", method="PUT", force=TRUE)
## End(Not run)

## put an existing file in remote server and remove file from local machine
## Not run:
request("filename.extension", method="PUT", rm.file=TRUE)
## End(Not run)


## remove a file in remote server
## Not run:
request("filename.extension", method="DELETE")
## End(Not run)
```

`retn` *Roman Empire transport network and Mediterranean region*

Description

This is a list of lists with specifications to plot different cartographical maps of the Roman Empire and the Mediterranean with transport network including settlements, roads, and shipping routes.

Usage

```
data("retn")
```

Format

A list of lists object with the shape data in different slots for 4 cartographical maps of the Roman Empire with names `rcoast` for a plain map, `rpcoast` for a map with provinces, `rpsi` for a map with senatorial and imperial provinces, and `rptetra` for a tetrarchy map. These options for cartographical maps in the Mediterranean are for both the classical and the late antiquity periods.

Three components in `retn` dataset have coordinates for settlements `nds`, roads `rds`, and shipping routes `srs` for these maps. In addition, the dataset has a cartographical map of the Mediterranean in `med` where settlements and transport network is yet to complete.

Source

DARMC, Center for Geographic Analysis, Harvard University

Rodrigue, Comtois, Slack. *The geography of transport systems*. Routledge (2013)

https://commons.wikimedia.org/wiki/File:RomanEmpire_117.svg

https://commons.wikimedia.org/wiki/File:Roman_provinces_trajan.svg

https://commons.wikimedia.org/wiki/File:Regioni_dell'Italia_Augustea.svg

See Also

plot.map, rp

---

rmids *Restricted multiply-imputed data subsets*

Description

A function to perform multiple imputation of missing dating data in the EDH dataset.

Usage

```
rmids(x, vars, collapse, pool, type = c("1", "2"))
```

Arguments

x	dataframe or list of dataframes with a data set to impute
vars	vector of attribute variables in x, typically dating data (optional)
collapse	collapse list of dataframes? (optional and logical, default FALSE)
pool	pool the results? (optional and logical)
type	type of pooling: "1" for min TAQ and max TPQ. "2" for conditional pooling

Details

Imputation refers to the replacement process of missing data, and this is the case of entries in the Epigraphic Database Heidelberg and related datasets. In this context, the missing data for imputation are the endpoints of the timespan of existence of epigraphs or inscriptions represented by variables TAQ and TPQ (cf. prex) as "not_before" and "not_after" in the EDH dataset with cases of censoring with one limit of the timespan known.

To perform imputation on subsets of missing dating data in the EDH dataset, function edhwpd serves to organize records per Roman province and dates by simple match similarity of different attribute variables specified in vars. Such organisation is in the form of a dataframe or a list of dataframes depending on the province characteristics, and a restricted multiply-imputed data subsets takes place on this outcome, and where collapse is for collapsing lists of dataframes.

When dating data is complete missing, rpd provides the average date, min TAQ, max TPQ, and the average length timespan for each Roman province that applies for a multiple imputation.

Value

A list of dataframes with imputed data where imputed dating data is not preceeded by a zero as with the recorded values. Component cases and names are:

NA-NA	all missing
taq-NA	censored
NA-tpq	censored
complete	complete data

Note

Rownames of complete dating data belonging to a component having imputed data gets replaced in the collapsed dataframe produced from a list of dataframes.

Author(s)

Antonio Rivero Ostoic

References

Ostoic, A and Letina, S. "Network imputation for missing dating data in archaeological artefacts," *The Connected Past: Artefactual Intelligence* conference, Aarhus (2021).

See Also

edhwpd, rpd, edhw, get.edh, cln

Examples

```
## Not run:
# extract from EDH dataset province, dates, and single variable attribute
arm <- edhwpd(vars="type_of_inscription", province="Arm", dates=c("not_after", "not_before"))

# perform restricted imputation
rmids(arm, vars=c("not_after", "not_before"))
## End(Not run)
```

---

rp *Roman province names and acronyms as in EDH dataset*

Description

This is a list with Roman province names and acronyms as in the Epigraphic Database Heidelberg recorded in EDH dataset.

Usage

```
data("rp")
```

Format

A list object of 66 Roman provinces names and acronyms as in "province_label" in EDH dataset.

Source

https://edh-www.adw.uni-heidelberg.de/data/api/terms/province

See Also

get.edh, EDH, edhw, retn, rpmp,

rpcp *Roman provinces and chronological periods*

Description

This dataset is a list of two data frames with 45 Roman provinces and regions with chronological periods of early and late Roman influence dates as time intervals.

Usage

```
data("rpcp")
```

Format

A list with two data frames named `Early` and `Late` of size 45×3 with ancient Roman provinces as `Province` where each data frame represent an historical period. The row names in each data frame records the acronyms of the Roman province.

Time intervals in the first data frame that corresponds to the `Early` period of influence in provinces and regions are `EarInf` and `OffPrv`, while time intervals in the second data frame for the `Late` period of influence are `LateInf` and `Fall` with the year of fall from the Roman Empire.

Source

https://www.unrv.com/provinces/provincetable.php (Retrieved on 2 July 2021)

See Also

`rp`, `plot.dates`

rpd *Roman provinces dates from EDH dataset*

Description

Dataset with a list with Roman province dates from the Epigraphic Database Heidelberg as in `EDH` dataset.

Usage

```
data("rpd")
```

Format

A list object of 66 Roman provinces with dates for inscriptions. Each list component has a vector for the province containing following dating data: average date, min TAQ, max TPQ, and the average length timespan. Components in the list have also an attribute class with the `HD_nr` entries of the province in `EDH` dataset where timespans, TAQ and TPQ are from entries `not_before` and `not_after`, respectively.

Source

https://edh-www.adw.uni-heidelberg.de/data/api/

See Also

`rmids`, `edhwpd`, `EDH`, `get.edh`, `rp`

`rpmcd` *Caption maps and affiliation dates of Roman provinces*

Description

This is a list with specifications to plot caption maps of 59 Roman provinces (year 117 AD) and Italian regions under Emperor Augustus (year 27 BC).

Usage

```
data("rpmcd")
```

Format

`rpmcd` is a list of lists for each province or region with two main components. One component is a list with shape data for a cartographical map caption in different slots, and the second component has for each Roman province an affiliation date when the territory became Roman province. `names(rpmcd)` have the acronyms according to `EDH` dataset.

Source

https://commons.wikimedia.org/wiki/File:RomanEmpire_117.svg

https://commons.wikimedia.org/wiki/File:Roman_provinces_trajan.svg

https://commons.wikimedia.org/wiki/File:Regioni_dell'Italia_Augustea.svg

See Also

`retn`, `rpmp`, `plot.map`, `rp`, `EDH`

simil *Similarity between vectors in columns*

Description

A function to compute similarity between vectors from columns in a data frame based on common attribute characteristics.

Usage

```
simil(x, vars, uniq, diag.incl, dichot, rm.isol, k)
```

Arguments

x	a list or a data frame object
vars	vector with column(s) in x representing variable attributes
uniq	unique entries? (optional and logical)
diag.incl	include also entries in matrix diagonal? (optional and logical)
dichot	dichotomize output? (optional and logical)
rm.isol	remove isolates in output? (optional and logical)
k	cut-off for dichotomization (if not specified, max of output)

Details

This is a function to compute the similarity between two or more vectors, which can arise from columns in a data frame and from list entries. The similarity of artefacts or other units having common variable attributes specified in vars is by simple matching, and this represents a measure of proximity among the items to compare. Comparison takes an id column from x; otherwise, the first column is taken provided that there are no duplicated entry names.

Both the dichotomization of the output and the removing isolated items from the system of co-occurrence relations depends on functions from package "multiplex".

Value

A valued matrix of similarities among items in x.

Author(s)

Antonio Rivero Ostoic

See Also

edhw, get.edh, dichot, rm.isol, multigraph.

Examples

```
## Not run:
# get inscriptions from a Roman province
arm <- edhw(province="Armenia")

# choose variables to a data frame
armv <- edhw(x=arm, as="df",
       vars=c("findspot_ancient", "type_of_inscription", "type_of_monument", "language"))

# matrix of similarities of two variables
simil(armv, vars=c("findspot_ancient", "language"))
## End(Not run)
```

Index