sdam: Social Dynamics and Complexity in the Ancient Mediterranean

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
Depends:	R (≥ 4.1.0)
Imports:	grImport2, multiplex
Suggests:	httr, rjson, grid, knitr, rmarkdown, multigraph
Published:	2022-09-05
Author:	Antonio Rivero Ostoic [aut, cre], Adela Sobotkova [ctb], Vojtech Kase [ctb], Petra Hermankova [ctb]
Maintainer:	Antonio Rivero Ostoic <jaro at cas.au.dk>
BugReports:	https://github.com/sdam-au/sdam/issues/
License:	CC BY 4.0
URL:	https://github.com/sdam-au/sdam/
NeedsCompilation:	no
CRAN checks:	sdam results

Documentation:

Reference manual:	sdam.pdf
Vignettes:	Dates and missing dating data in '"sdam"' Re-encoding 'people' in the 'EDH' dataset Datasets in '"sdam"' package Cartographical maps and networks

Downloads:

Package source:	sdam_1.1.4.tar.gz
Windows binaries:	r-devel: sdam_1.1.4.zip, r-release: sdam_1.1.4.zip, r-oldrel: sdam_1.1.4.zip
macOS binaries:	r-release (arm64): not available, r-oldrel (arm64): not available, r-release (x86_64): sdam_1.1.4.tgz, r-oldrel (x86_64): sdam_1.1.4.tgz

Linking:

Please use the canonical form `https://CRAN.R-project.org/package=sdam` to link to this page.

Datasets in "sdam" package

Antonio Rivero Ostoic

September 2022

Preliminaries

Install and load one version of "sdam" package.

```
install.packages("sdam") # from CRAN
devtools::install_github("sdam-au/sdam") # development version
devtools::install_github("mplex/cedhar", subdir="pkg/sdam") # a legacy version R 3.6.x
```

```
# load and check version
library(sdam)
packageVersion("sdam")
```

```
[1] '1.0.0'
```

Built-in datasets

Package "sdam" comes with a suite of datasets and external data to execute different functions available in the package and to perform analysis.

For a list of built-in datasets in "sdam" use the "utils" function data() or utils::data() with the 'package' argument.

The CRAN distribution has four built-in datasets, while the development and legacy distributions add three more built-in datasets.

```
# pop-up a new window
data(package="sdam")
```

```
# Data sets in package 'sdam':
#
# retn      Roman Empire transport network and Mediterranean sea
# rp        Roman province names and acronyms as in EDH
# rpcp      Roman provinces chronological periods
# rpd       Roman provinces dates from EDH
# rpmcd     Caption maps and affiliation dates of Roman provinces
```

```
# Additional built-in datasets in 'sdam':
#
# EDH       Epigraphic Database Heidelberg Dataset
# rpmp      Maps of ancient Roman provinces and Italian regions
```

A description of each dataset is available in the manual that from the R console is accessible as e.g. the EDH dataset in a non-CRAN distribution.

```
# Epigraphic Database Heidelberg Dataset help
?EDH
```

Ancient Mediterranean built-in datasets

The `EDH` dataset in `"sdam"` has information about Latin epigraphy retrieved from the Epigraphic Database Heidelberg API repository from the Roman world during the antiquity period.

A list of Roman provinces and regions in this dataset is available in dataset `"rp"`, and use again function `data()` to load this built-in dataset to look at its internal structure with `utils::str()` function.

- Dataset `"rp"` is a named list with Roman provinces and regions with acronyms according to the Epigraphic Database Heidelberg.

```
# load dataset
data("rp")

# obtain object structure
str(rp)
```

```
List of 66
 $ Ach: chr "Achaia"
 $ Aeg: chr "Aegyptus"
 $ Aem: chr "Aemilia (Regio VIII)"
 $ Afr: chr "Africa Proconsularis"
 $ AlC: chr "Alpes Cottiae"
 $ AlG: chr "Alpes Graiae"
 $ AlM: chr "Alpes Maritimae"
 $ AlP: chr "Alpes Poeninae"
 $ ApC: chr "Apulia et Calabria (Regio II)"
 $ Aqu: chr "Aquitania"
 $ Ara: chr "Arabia"
 $ Arm: chr "Armenia"
 $ Asi: chr "Asia"
 $ Ass: chr "Assyria"
 $ Bae: chr "Baetica"
 $ Bar: chr "Barbaricum"
 $ Bel: chr "Belgica"
 $ BiP: chr "Bithynia et Pontus"
 $ BrL: chr "Bruttium et Lucania (Regio III)"
 $ Bri: chr "Britannia"
 $ Cap: chr "Cappadocia"
 $ Cil: chr "Cilicia"
 $ Cor: chr "Corsica"
 $ Cre: chr "Creta"
 $ Cyp: chr "Cyprus"
 $ Cyr: chr "Cyrene"
 $ Dac: chr "Dacia"
 $ Dal: chr "Dalmatia"
 $ Epi: chr "Epirus"
 $ Etr: chr "Etruria (Regio VII)"
 $ Gal: chr "Galatia"
 $ GeI: chr "Germania inferior"
```

```
 $ GeS: chr "Germania superior"
 $ HiC: chr "Hispania citerior"
 $ Inc: chr "Provincia incerta"
 $ Iud: chr "Iudaea"
 $ LaC: chr "Latium et Campania (Regio I)"
 $ Lig: chr "Liguria (Regio IX)"
 $ Lug: chr "Lugdunensis"
 $ Lus: chr "Lusitania"
 $ LyP: chr "Lycia et Pamphylia"
 $ MaC: chr "Mauretania Caesariensis"
 $ MaT: chr "Mauretania Tingitana"
 $ Mak: chr "Macedonia"
 $ Mes: chr "Mesopotamia"
 $ MoI: chr "Moesia inferior"
 $ MoS: chr "Moesia superior"
 $ Nar: chr "Narbonensis"
 $ Nor: chr "Noricum"
 $ Num: chr "Numidia"
 $ PaI: chr "Pannonia inferior"
 $ PaS: chr "Pannonia superior"
 $ Pic: chr "Picenum (Regio V)"
 $ Rae: chr "Raetia"
 $ ReB: chr "Regnum Bospori"
 $ Rom: chr "Roma"
 $ Sam: chr "Samnium (Regio IV)"
 $ Sar: chr "Sardinia"
 $ Sic: chr "Sicilia, Melita"
 $ Syr: chr "Syria"
 $ Thr: chr "Thracia"
 $ Tra: chr "Transpadana (Regio XI)"
 $ Tri: chr "Tripolitania"
 $ Umb: chr "Umbria (Regio VI)"
 $ Val: chr "Valeria"
 $ VeH: chr "Venetia et Histria (Regio X)"
```

edhw() interface with "rp" dataset

- Function `edhw()` is a wrapper to extract and transform the records in the `EDH` dataset that invokes `"rp"` dataset to retrieve the records from a specific Roman province or region in `EDH`.

```
# Armenian records in 'EDH'
edhw(province="Arm")[1]
```

```
Warning in edhw(province = "Arm"): "x" is for dataset "EDH".

Warning in edhw(province = "Arm"): "province" with no "vars" returns lists.

[[1]]
[[1]]$ID
[1] "015521"

[[1]]$commentary
[1] " Mehrere, teils aneinanderpassende Fragmente erhalten. Die Inschrift lief über mehrere Ta
```

```
[[1]]$country
[1] "Armenia"

[[1]]$depth
[1] "21 cm"

[[1]]$diplomatic_text
[1] "IMP CAESAR DIV[ ] NERVAE F[ ]ERVA TRAIANVS / OPTIMVS A[ ]G G[ ]RM DACI[ ]THICVS PONT MAX

[[1]]$edh_geography_uri
[1] "https://edh-www.adw.uni-heidelberg.de/edh/geographie/3407"

[[1]]$findspot_ancient
[1] "Artaxata, bei"

[[1]]$findspot_modern
[1] "Pokr Vedi"

[[1]]$height
[1] "80 cm"

[[1]]$id
[1] "HD015521"

[[1]]$language
[1] "Latin"

[[1]]$last_update
[1] "2015-10-22"

[[1]]$letter_size
[1] "20-16 cm"

[[1]]$literature
[1] "AE 1968, 0510.; B. Arakelean, RevPaz 126, 1968, 135-136; Foto u. Zeichnung. - AE 1968 "

[[1]]$material
[1] "limestone: rocks - clastic sediments"

[[1]]$military
[1] "data available"

[[1]]$not_before
[1] "0116"

[[1]]$people
[[1]]$people[[1]]
[[1]]$people[[1]]$person_id
[1] "1"

[[1]]$people[[1]]$name
[1] "div[i] Nervae f[il. N]erva Traianus"
```

```
[[1]]$people[[1]]$gender
[1] "male"

[[1]]$people[[1]]$cognomen
[1] "Nerva+ Traianus"



[[1]]$province_label
[1] "Armenia"

[[1]]$responsible_individual
[1] "Gräf"

[[1]]$transcription
[1] "Imp(erator) Caesar div[i] Nervae f[il(ius) N]erva Traianus / optimus A[u]g(ustus) G[e]rm(

[[1]]$trismegistos_uri
[1] "https://www.trismegistos.org/text/217430"

[[1]]$type_of_inscription
[1] "building/dedicatory inscription"

[[1]]$type_of_monument
[1] "tabula"

[[1]]$uri
[1] "https://edh-www.adw.uni-heidelberg.de/edh/inschrift/HD015521"

[[1]]$width
[1] "(205) cm"

[[1]]$work_status
[1] "provisional"

[[1]]$year_of_find
[1] "1967"
```

The `Warning` messages from `edhw()` are first because there is not an explicit input in `x`, it is assumed that the input data is from the `EDH` dataset. The second warning message just tells the type object to return is always a list for argument `province` alone.

`EDH` in data frames

All records in the `EDH` dataset have a list format and it is possible to transform this information into a dataframe format with the wrapper function `edhw()`. For instance, displaying the first record from `Arm` as a data frame in argument `'as'` is made by the record `'id'` number.

```
# record HD015521
edhw(id="15521", as="df")
```

However, it is easier to visualise in the screen only the variables related to people.

```
# record HD015521 with explicit variables
edhw(id="15521", vars="people", as="df")
```

```
        id        cognomen gender                                  name person_id
1 HD015521 Nerva+ Traianus   male div[i] Nervae f[il. N]erva Traianus         1
```

```
# record HD015521 with more explicit variables
edhw(id="15521", vars=c("people", "province_label"), as="df")
```

```
        id        cognomen gender                                  name person_id
1 HD015521 Nerva+ Traianus   male div[i] Nervae f[il. N]erva Traianus         1
  province_label
1        Armenia
```

Obtaining all `people` variables

Start by looking at the `people` variables in the `EDH` dataset for the Roman province of **Armenia**.

Armenia

Fig. 1: Roman province of Armenia (ca 117 AD).

Transformation of the entire province from the `EDH` dataset requires extracting first a list with the province content. Function `edhw()` is to obtain available inscriptions per province from `EDH` and all data attributes from `people` variable. The default outputs are a list and a dataframe for the first and the second instance of the function.

```
# people in Armenia
edhw(province="Arm") |>
  edhw(vars="people")
```

```
        id         age: years        cognomen gender
1 HD015521               <NA> Nerva+ Traianus   male
2 HD015524 data not available        Cre(---)   male
3 HD015524               <NA>           [---]   male
                                name     nomen person_id praenomen             status
1 div[i] Nervae f[il. N]erva Traianus      <NA>         1      <NA>               <NA>
2                       C. Val. Cre. Valerius*         1        C. military personnel
3                              [---]     [---]         2       [-] military personnel
```

People attribute variables in inscriptions for `Armenia` are `age: years`, `cognomen`, `gender`, `name`, `nomen`, `person_id`, `praenomen`, and, `status`, but any inscription with `tribus` or `origo` as in the case of other provinces.

For `Armenia`, two inscriptions have people variables and all people scripted are `male`, where record `HD015524` spans two rows because there are two persons where one have `nomen`, `cognomen`, and `name` ineligible.

Datasets for cartographical maps

The plotting of the Roman province in the previous section requires other datasets. Apart from "rp". In "sdam", there are other three datasets invoked for plotting cartographical maps related to the Roman Empire and the Mediterranean basin, which are "rpmp", "rpmcd", and "retn".

Function plot.map() calls dataset "rpmp" for the shapes and colours in the plotting of the cartographical maps of different regions of the Roman Empire. For the caption and province dates with this function shapes and colours are in dataset "rpmcd".

- Dataset "retn" bears the shapes of places and routes of an ancient transportation system in the Mediterranean region and political divisions of the Roman Empire. It also has it contours and parts of the European continent.

```
# land contour around Mediterranean
plot.map(type="plain")
```



```
# display settlements and shipping routes
plot.map(type="plain", settl=TRUE, shipr=TRUE)
```

Vignette Cartographical maps and networks has more about transportation networks in the ancient Mediterranean.

Datasets with dates

There are built-in datasets in "sdam" related to dates as well that are either displayed in a cartographical map or used for other computations.

- Dataset "rpd" that has dates for provinces from the EDH dataset. It serves for performing a restricted imputation on data subsets in EDH or in another dataset.

```
# dates from EDH
data("rpd")

# three provinces in object structure
str(rpd[1:3])
```

```
List of 3
 $ Ach: 'HD001917' num [1:4] 105 -190 571 761
 $ Aeg: 'HD000741' num [1:4] 144 -71 500 571
 $ Aem: 'HD000010' num [1:4] 121 -201 771 972
```

From this set of three Roman provinces in the EDH, the longest timespan is for Aem, and on average Ach has the oldest incriptions, while Aeg has incriptions with the newest dates.

- Dataset "rpcp" with chronological periods for regions with early and later Roman influence per province.

```
# periods for Roman provinces
data("rpcp")

# object structure
str(rpcp)
```

```
List of 2
 $ Early:'data.frame':  45 obs. of  3 variables:
  ..$ Province: chr [1:45] "Italia (Final Consolidation)" "Sicilia" "Sardinia & Corsica" "Hisp
  ..$ EarInf  : num [1:45] -509 -241 -238 -206 -206 -206 -202 -202 -188 -188 ...
  ..$ OffPrv  : num [1:45] -272 -241 -238 -197 -197 -197 -146 -81 43 -133 ...
 $ Late :'data.frame':  45 obs. of  3 variables:
  ..$ Province: Factor w/ 45 levels "Achaea","Aegyptus",..: 30 43 42 27 28 26 3 23 32 9 ...
  ..$ LateInf : num [1:45] 476 436 436 409 409 ...
  ..$ Fall    : num [1:45] 476 436 436 409 409 409 409 418 1400 1500 ...
```

The early and later Roman influence in the 45 ancient provinces and regions are timespans with a *terminus ante quem* and a *terminus post quem*.

Vignette Dates and missing dating data has the visualisation of these and other dates.

External data

Apart from the built-in datasets, it is attached as external data the semi-colon separated file StraussShipwrecks.csv with the Shipwrecks dataset for performing analyses: Reference and documentation in

Strauss, J. (2013). *Shipwrecks Database*. Version 1.0. Accessed (07-12-2021) from oxrep.classics.ox.ac.uk/databa

Built from Parker, A.J. *Ancient Shipwrecks of the Mediterranean and the Roman Provinces* (Oxford: BAR International Series 580, 1992)

Details about the access to the database are in:

- Shipwrecks network in the Mediterranean Basin (23-June-2022)
- Vignettes Dates and missing dating data and Cartographical maps and networks also use the Shipwrecks dataset.

Re-encoding `people` in the EDH dataset

Antonio Rivero Ostoic

September 2022

```
# load and check versions
library(sdam)
packageVersion("sdam")
```

```
[1] '1.0.0'
```

EDH people

- EDH is a dataset in `"sdam"` that contains the texts of Latin and Latin-Greek inscriptions of the Roman Empire, which have been retrieved from the Epigraphic Database Heidelberg API repository through routines `get.edh()` and `get.edhw()`.

Since the year 2022 and still today, the API repository does not support people variables, and the EDH dataset serves as an alternative for the analysis of people-related inscriptions.

One challenge with people variables in EDH is that some records contain characters in Greek and Latin extended that need re-encoding for a proper rendering and display.

Re-encoding `people` in EDH

Ancient inscriptions in some Roman provinces have Greek characters written and, due to encoding and decoding steps in the process of extraction, loading, and transformation of the data (perhaps Treating UTF-8 Bytes as Windows-1252?), Greek and other Latin characters are not displayed properly with the actual version of the EDH dataset. Most of the encoding issues are in variables related to people, and some examples with inscriptions in Roman provinces are next.

Achaia

The Roman province of **Achaia** in the EDH dataset has inscriptions related to people.

Figure 2: Roman province of Achaia (ca 117 AD).

Function `edhw()` is to obtain the available inscriptions per province in the EDH dataset, which is a list that is the input for the same function to extract `people` variables *cognomen* and *nomen*. In this case, the `'province'` argument is `Ach` that stands for `Achaia`.

```
# select two people variables from Achaia
Ach <- edhw(province="Ach") |>
  edhw(vars="people", select=c("cognomen","nomen"))
```

There are 1539 records with people in `Ach` that corresponds to the number of rows in this data frame.

```
# number of people entries in Achaia
nrow(Ach)
```

```
[1] 1539
```

However, some records have either missing data or are inscriptions where *cognomen* and *nomen* are not available.

```
# also remove NAs
Ach <- edhw(province="Ach") |>
  edhw(vars="people", select=c("cognomen","nomen"), na.rm=TRUE)

nrow(Ach)
```

```
[1] 1465
```

Clean function for re-encoding

Treating with `people` attribute variables requires many times re-encoding that is one option in function `cln()`. For instance, values in *cognomen* in the first entries of `Ach` are likely in Greek.

```
# some people entries in Achaia
head(Ach)
```

```
        id                                                cognomen                    nomen
1 HD001917                                                   Rufus Ponponius (= Pomponius)
2 HD001917                                                     Eia   Ponponia (= Pomponia)
3 HD001917                                 Î<U+0094>á½¹Î¾Î± Î\235á½·Î°Î·                    <NA>
4 HD002097 Î<U+0092>Î±Î»Î»ÎµÎ½Î<U+0084>Î¹Î½Î¹Î±Î½á½¹Ï<U+0082>+                    <NA>
5 HD002097                                  Î<U+0092>á½±Î»Î·Ï<U+0082>                    <NA>
6 HD002097                                               Arcadius+                    <NA>
```

Function `cln()` serves to re-encode Greek and Latin characters to render Greek, Greek extended, and Latin extended glyphs.

```
# re-encode in Ach cognomen
Ach$cognomen |>
  head() |>
  cln()
```

```
cognomen
```

Rufus
Eia
ΔόξαΝίκη
Βαλλεντινιανός+
Βάλης
Arcadius+

For *cognomen* in the last people entries in `Achaia`.

```r
# last entries
tail(Ach)
```

```
             id                                                                                            cognomen
1534 HD068263                                                                           Î<U+009A>á½±Î»Î»Ï<U+0085>Ï<U+0082>
1535 HD068315 Î¦Ï\201Î¿Î½Ï<U+0084>Îµá¿<U+0096>Î½Î¿Ï<U+0082> Î\235ÎµÎ¹Îºá½µÏ\201Î±Ï<U+0084>Î¿Ï<
1536 HD068319 Î¦Ï\201Î¿Î½Ï<U+0084>Îµá¿<U+0096>Î½Î¿Ï<U+0082> Î\235ÎµÎ¹Îºá½µÏ\201Î±Ï<U+0084>Î¿Ï<
1537 HD072342                                                                     Î<U+0091>á¼°Î¼Î¹Î»Î¹Î±Î½á½¹Ï<U+0082>+
1538 HD072342                                                                         Î<U+009A>Î±Î¹Î»Î¹Î±Î½á½¹Ï<U+0082>+
1539 HD078079                                                                                                 Eburo
                                    nomen
1534                                 <NA>
1535 Î<U+009A>Î»Î±á½»Î´Î¹Î¿Ï<U+0082>
1536 Î<U+009A>Î»Î±á½»Î´Î¹Î¿Ï<U+0082>
1537 Î<U+009F>á½\220á½±Ï\201Î¹Î¿Ï<U+0082>+
1538                                 <NA>
1539                                 <NA>
```

After re-encoding the last records in `Ach` with `cln()`, it is easier to see, for example, that some have identical *cognomen* where entries having `<NA>` in the input become `NA`.

```r
# clean last entries of cognomen
Ach$cognomen |>
  tail() |>
  cln()
```

cognomen

Κάλλυς
ΦροντεῖνοςΝεικήρατος
ΦροντεῖνοςΝεικήρατος
Αἰμιλιανός+
Καιλιανός+
Eburo

```r
# clean last entries of nomen
Ach$nomen |>
  tail() |>
  cln()
```

nomen

NA
Κλαύδιος
Κλαύδιος
Οὐάριος+
NA
NA

Re-encode Greek and Latin within data frames

Aegyptus

In the case of the province of **Aegyptus**, three people variables have a mixing og Greek and Latin characters scripted that need *re-codification* as well.

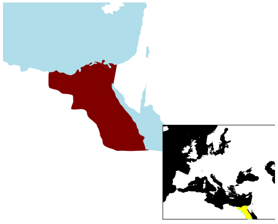


Figure 3: Roman province of Aegyptus (ca 117 AD).

```r
# Aegyptus people
Aeg <- edhw(province="Aeg") |>
  edhw(vars="people")
```

```r
# three variables of the last eight records
Aeg[ , c(3,5:6)] |>
  tail(8)
```

```
                                                                      cognomen
81                              Augustus+ / Î£ÎµÎ²Î±Ï<U+0083>Ï<U+0084>á½¹Ï<U+0082>
82                                                   Aquila / á¼<U+0088>Îºá½»Î»Î±
83 Traianus Hadrianus / Î¤Ï\201Î±Î¹Î±Î½á½¸Ï<U+0082> á¼<U+0089>Î´Ï\201Î¹Î±Î½á½¹Ï<U+0082>
84                                                Serenus / Î£ÎµÏ\201Î·Î½á½¹Ï<U+0082>
85                        Domitianus+ / Î<U+0094>Î¿Î¼Î¹Ï<U+0084>Î¹Î±Î½á½¹Ï<U+0082>++
86                                Vegetus / Î<U+009F>á½\220ÎÎ³ÎµÏ<U+0084>Î¿Ï<U+0082>
87                                 Î<U+009B>Ï<U+0085>Ï<U+0083>á¾¶Ï<U+0082> / Lysas
88                                            Î Î»á½¹ÎºÎ±Î¼Î¿Ï<U+0082> / Plocamus

81 Imp. Caesar divi f. August. / Î<U+0091>á½\220Ï<U+0084>Î¿ÎºÏ\201á½±Ï<U+0084>Ï<U+0089>Ï\201 Î
82                                                                                       C.
83
84                       Sulpic. Serenus / Î£Î¿Ï<U+0085>Î»Ï<U+0080>á½·ÎºÎ¹Î¿Ï<U+0082> Ï<U+0085>
85
86                                                       G. Septimio Vegeto / Î<U+0093>Î±á¿<U
87                                            Î<U+009B>Ï<U+0085>Ï<U+0083>á¾¶Ï<U+0082> Î Î¿Ï<U+0
88                                                                                       Î Î
                                       nomen
81            Caesar / Î<U+009A>Î±á¿<U+0096>Ï<U+0083>Î±Ï\201
82                        Iulius / á¼¸Î¿á½»Î»Î¹Î¿Ï<U+0082>
83                                                  <NA>
84 Sulpicius* / Î£Î¿Ï<U+0085>Î»Ï<U+0080>á½·ÎºÎ¹Î¿Ï<U+0082>
85                                                  <NA>
86   Septimius / Î£ÎµÏ<U+0080>Ï<U+0084>á½·Î¼Î¹Î¿Ï<U+0082>
87                                                  <NA>
88                       á¼<U+008C>Î½Î½Î¹Î¿Ï<U+0082> / Annius
```

For people in `Aegyptus`, columns three, and five to six correspond to *cognomen*, *name*, and *nomen*, where the output from `cln()` in the console is a dataframe.

```
# re-encode three variables from last entries
Aeg[ ,c(3,5:6)] |>
  tail() |>
  cln()
```

```
cognomen
```

Augustus+ / Σεβαστός
Aquila / Ἀκύλα
Traianus Hadrianus / ΤραιανὸςἌδριανός
Serenus / Σερηνός
Domitianus+ / Δομιτιανός++
Vegetus / Οὐέγετος
Λυσᾶς / Lysas
Πλόκαμος / Plocamus

```
name
```

Imp. Caesar divi f. August. / ΑὐτοκράτωρΚαῖσαρθεοῦυἱὸςΣεβαστὸς
C. Iulio Aquila / ΓαΐουἸουλίουἈκύλα
Traiani Hadriani / ΤραιανοῦἌδριανοῦ
Sulpic. Serenus / ΣουλπίκιοςυἱὸςΓναίουΚουιρίναΣερηνὸς
[Domitiani] / [[Δομιτια
G. Septimio Vegeto / ΓαΐουΣεπτιμίουΟὐεγέτου
ΛυσᾶςΠοπλίουἈννίουΠλοκάμου / Lysas P. Anni Plocami
ΠοπλίουἈννίουΠλοκάμου / P. Anni Plocami

```
nomen
```

Caesar / Καῖσαρ
Iulius / Ἰούλιος
NA
Sulpicius* / Σουλπίκιος
NA
Septimius / Σεπτίμιος
NA
Ἄννιος / Annius

Some entries in `Aeg` have Greek extended characters, and one entry in Latin has a special character at the end (`Sulpicius*`), which can be omitted for further computations by raising the cleaning level to 2.

nomen in Aegyptus

Benefits from re-encoding and cleaning text from the `EDH` dataset are evident like when counting occurrences in the different attribute variables as with `nomen` in `Aeg`.

```
# default cleaning level 1
Aeg$nomen |>
  cln() |>
  table() |>
  sort(decreasing=TRUE)
```

Sempronius+

```
[1] 4
```

Κούρτιος

```
[1] 2
```

Μέμμιος

```
[1] 2
```

Ἰούλιος

```
[1] 2
```

etc.

```
...
```

By raising the cleaning level to 2, all special characters are removed from the end, and it is possible to see that, in the Roman province of Aegyptus, `Sempronius`, `Sentius`, `Valerius` are the three most common *nomen* in inscriptions with four occurrences each.

```
# raise cleaning level and remove NAs
Aeg$nomen |>
  cln(level=2, na.rm=TRUE) |>
  table() |>
  sort(decreasing=TRUE)
```

Sempronius

```
[1] 4
```

Sentius

```
[1] 4
```

Valerius

```
[1] 4
```

Κούρτιος

```
[1] 2
```

etc.

```
...
```

Caveats

See `Warnings` section in manual.

Dates and missing dating data in "sdam"

Antonio Rivero Ostoic

September 2022

```
# load and check versions
library(sdam)
packageVersion("sdam")
```

```
[1] '1.0.0'
```

Dating data

Temporal data is significant when it comes to analysing the history of archaeological artefacts like written markers from the Ancient Mediterranean. In the `EDH` dataset, for example, dates for inscriptions are plausible timespans of existence with the endpoints in variables `not_before` and `not_after` that, from the perspective of the timespan, are the *terminus ante quem* (TAQ) and *terminus post quem* (TPQ) of the time segment. However, not all inscriptions have these two variables filled by domain experts and replacing missing dating data constitutes a challenge.

Besides `EDH`, other datasets with `"sdam"` the package and related functions involve dating data in the ancient Mediterranean like displaying dates and time segments in a plot, by organising dates within Roman provinces, and by performed imputation techniques for missing dating data.

Plotting temporal data

Shipwrecks dataset dating data

An example of plotting dates is with the Shipwrecks external dataset, which is a semicolon separated file of different variables.

References for shipwrecks data are in

- Vignette Datasets in `"sdam"` package

When reading the shipwrecks external dataset with `read.csv` make sure to use the right separator in `sep` and leave untouched the names of the variables.

```
# load shipwrecks external dataset
sw <- system.file("extdata","StraussShipwrecks.csv",package="sdam") |>
  read.csv(sep=";", check.names=FALSE)
```

```
# variables in shipwrecks dataset
colnames(sw)
```

```
 [1] "Wreck ID"                "Strauss ID"              "Name"
 [4] "Parker Number"           "Sea area"                "Country"
 [7] "Region"                  "Latitude"                "Longitude"
[10] "Min depth"               "Max depth"               "Depth"
[13] "Period"                  "Dating"                  "Earliest date"
[16] "Latest date"             "Date range"              "Mid point of date range"
[19] "Probability"             "Place of origin"         "Place of destination"
[22] "Reference"               "Comments"                "Amphorae"
[25] "Marble"                  "Columns etc"             "Sarcophagi"
[28] "Blocks"                  "Marble type"             "Other cargo"
[31] "Hull remains"            "Shipboard paraphernalia" "Ship equipment"
[34] "Estimated tonnage"       "Amphora type"
```

Plot the time segments with function `plot.dates()` and a customized `'id'` where variables 15 to 16 in `sw` have timespans of existence as `'taq'` and `'tpq'`.

```
# shipwrecks dates with Wreck ID
plot.dates(sw, id="Wreck ID", type="rg", taq="Earliest date", tpq="Latest date", col=4)
```

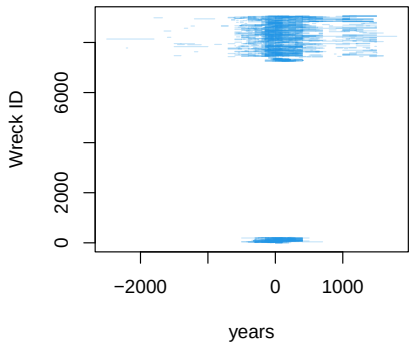


Fig. 4: Range of timespans in Shipwrecks dataset

Mid points and range of timespan

The mid points and range of shipwrecks data are explicitly computed by function `prex()` with the `mp` option in the `'type'` argument. `'vars'` stands for the variables that in this case are TAQ and TPQ, and the `'keep'` option allows maintaining the rest of the variables in the output that for `prex()` with mid points is a data frame.

```
# add mid points and range to shipwrecks data
prex(sw[c(1,7,15:16)], type="mp", vars=c("Earliest date", "Latest date"), keep=TRUE) |>
  tail()
```

```
     Wreck ID   Region Earliest date Latest date Mid point Range
1779     9057 Sardinia            50         200     125.0   150
1780     9058 Sardinia           400         500     450.0   100
1781     9059 Sardinia          1000        1500    1250.0   500
1782     9060  Liguria          -100          -1     -50.5    99
1783     9061   Sicily          1100        1200    1150.0   100
1784     9063 Calabria           300         500     400.0   200
```

The default `'type'` option and chronological phase in `prex()` are the aoristic sum with a five periods bin or `bin5`.

```
# aoristic sum shipwrecks
prex(sw[c(1,7,15:16)], vars=c("Earliest date", "Latest date"))
```

```
      Arch      Class       Hell        Rom        Byz
  202.5187   312.0645  4460.9831 13235.0372   622.2608
```

For an eight chronological periods bin in the shipwrecks dataset

```
# aoristic sum shipwrecks 8 bin
prex(sw[c(1,7,15:16)], vars=c("Earliest date", "Latest date"), cp="bin8")
```

```
     Arch     Class      Hell      ERom      MRom      LRom      EByz      MByz
 202.5187  312.0645 4460.9831 2431.3934  881.8685 1197.9617  101.5077  226.2947
```

For aoristic sum algorithm, cf. Temporal Uncertainty.

Dating data in the Roman world

Many functions and datasets in `"sdam"` are related to temporal information of the Roman world, particularly from the Roman Empire during the classical ancient period.

Function `plot.map()` is to depict cartographical maps per Roman province or region, and it has a `'date'` argument to display dates within the caption. Dates in this case are one or two years either for the consolidation of the Italian peninsula or the affiliation of the region to the Roman Empire.

```
# silhouette of Italian peninsula
plot.map(x="Ita", date=TRUE)
## not run
```

- The built-in dataset `rpmcd` has the shapes and colours used in the cartographical maps with `plot.map()`, and some dates related to provinces as well.

```
# 59 provinces dates, colors, and shapes
data("rpmcd")

# province acronyms as in EDH
names(rpmcd)
```

```
 [1] "Ach" "Aeg" "Afr" "AlC" "AlM" "AlP" "Aqu" "Ara" "Arm" "Asi" "Ass" "Bae" "Bel" "BiP" "Bri"
[16] "Cap" "Cil" "Cor" "Cre" "Cyp" "Cyr" "Dac" "Dal" "Epi" "Gal" "GeI" "GeS" "HiC" "Ita" "Iud"
[31] "Lug" "Lus" "LyP" "MaC" "Mak" "MaT" "Mes" "MoI" "MoS" "Nar" "Nor" "PaI" "PaS" "Rae" "Sar"
[46] "Sic" "Syr" "Thr" "Aem" "ApC" "BrL" "Etr" "LaC" "Lig" "Pic" "Sam" "Tra" "Umb" "VeH"
```

Roman provinces establishment dates

The establishment dates of Roman provinces used in the cartographical map captions are in the second component of `rpmcd`.

```
# pipe dataset for dates in second component
rpmcd |>
  lapply(function (x) x[[2]]) |>
  head()
```

```
$Ach
[1] "27 BC"

$Aeg
[1] "30 BC"

$Afr
[1] "146 BC"

$AlC
[1] "63AD or 58AD"

$AlM
[1] "63AD or 14BC"

$AlP
[1] "63AD or 14BC"
```

A vector of establishment dates in years from the `"rpmcd"` dataset is recorded in object `est` that allow making a chronology of the Roman provinces.

```
# second component in dataset
est <- rpmcd |>
  lapply(function (x) x[[2]]) |>
  unlist(use.names=FALSE)
est
```

```
 [1] "27 BC"              "30 BC"              "146 BC"             "63AD or 58AD"
 [5] "63AD or 14BC"       "63AD or 14BC"       "51 BC"              "105 AD"
 [9] "114 AD"             "133 BC"             "116 AD"             "197 BC"
[13] "51 BC"              "74BC or 64BC"       "43 AD"              "17 AD"
[17] "64 BC"              "238 BC"             "66 BC?"             "58 BC -30 BC"
[21] "74 BC"              "106 AD"             "32BC or 10AD"       "148 BC"
[25] "25 BC"              "27 BC"              "27 BC"              "197 BC"
[29] "272 BC"             "6 AD"               "51 BC"              "197 BC"
[33] "43 AD"              "42AD or 44AD"       "148 BC?"            "42 AD or 44 AD"
[37] "116 AD"             "6 AD"               "6 AD"               "121 BC"
[41] "16BC or 15BC"       "9AD or 10AD"        "9AD or 10AD"        "16BC or 15BC"
[45] "238 BC"             "241 BC"             "64 BC"              "46 AD"
[49] "272 BC (Ita cons.)" "272 BC (Ita cons.)" "272 BC (Ita cons.)" "272 BC (Ita cons.)"
[53] "272 BC (Ita cons.)" "272 BC (Ita cons.)" "272 BC (Ita cons.)" "272 BC (Ita cons.)"
[57] "272 BC (Ita cons.)" "272 BC (Ita cons.)" "272 BC (Ita cons.)"
```

Formatting dates

The establishment dates of Roman provinces and regions are in vector `est`, and these dates can become more standard with the function `cln()` for further processing. This is a cleaning function where, for instance, level `9` removes all content after the first parenthesis in the input while the other levels are for specific needs.

```
# clean levels are 0-9
cln(est, level=9)
```

```
 [1] "27 BC"          "30 BC"          "146 BC"         "63AD or 58AD"   "63AD or 14BC"
```

```
 [6] "63AD or 14BC"   "51 BC"          "105 AD"         "114 AD"         "133 BC"
[11] "116 AD"         "197 BC"         "51 BC"          "74BC or 64BC"   "43 AD"
[16] "17 AD"          "64 BC"          "238 BC"         "66 BC"          "58 BC-30 BC"
[21] "74 BC"          "106 AD"         "32BC or 10AD"   "148 BC"         "25 BC"
[26] "27 BC"          "27 BC"          "197 BC"         "272 BC"         "6 AD"
[31] "51 BC"          "197 BC"         "43 AD"          "42AD or 44AD"   "148 BC"
[36] "42 AD or 44 AD" "116 AD"         "6 AD"           "6 AD"           "121 BC"
[41] "16BC or 15BC"   "9AD or 10AD"    "9AD or 10AD"    "16BC or 15BC"   "238 BC"
[46] "241 BC"         "64 BC"          "46 AD"          "272 BC"         "272 BC"
[51] "272 BC"         "272 BC"         "272 BC"         "272 BC"         "272 BC"
[56] "272 BC"         "272 BC"         "272 BC"         "272 BC"
```

After this transformation of the data in `est`, is possible to format dates as numerical data with function `dts()`, which takes the first value when there are two competing dates in the input; unless the opposite is specified in the `'last'` argument.

```
# update object with establishment dates
est <- est |>
  cln(level=9) |>
  dts()
```

```
est
```

```
        27 BC          30 BC        146 BC   63AD or 58AD   63AD or 14BC   63AD or 14BC
          -27            -30          -146             63             63             63
        51 BC        105 AD         114 AD        133 BC         116 AD         197 BC
          -51            105           114          -133            116          -197
        51 BC   74BC or 64BC         43 AD         17 AD          64 BC         238 BC
          -51           -74            43            17           -64          -238
        66 BC    58 BC-30 BC         74 BC        106 AD   32BC or 10AD         148 BC
          -66           -58           -74           106           -32          -148
        25 BC          27 BC         27 BC        197 BC         272 BC          6 AD
          -25            -27           -27          -197          -272             6
        51 BC        197 BC         43 AD   42AD or 44AD         148 BC 42 AD or 44 AD
          -51          -197            43            42          -148            42
      116 AD          6 AD          6 AD        121 BC   16BC or 15BC    9AD or 10AD
         116             6             6          -121          -16             9
  9AD or 10AD   16BC or 15BC        238 BC        241 BC         64 BC         46 AD
           9           -16          -238          -241          -64            46
      272 BC        272 BC        272 BC        272 BC         272 BC         272 BC
        -272          -272          -272          -272          -272          -272
      272 BC        272 BC        272 BC        272 BC         272 BC
        -272          -272          -272          -272          -272
```

Chronology of Roman provinces

Object `est` has a chronology for the establishment dates of Mediterranean regions and territories as Roman provinces that corresponds to the provinces in `"rpmcd"` dataset. The union of the names of provinces and dates of establishment as a Roman province is a data frame object `rpde` that better displays without the row names.

```
# Roman province dates of establishement (strings still strings)
rpde <- cbind(names(rpmcd),dts(est)) |>
  as.data.frame(stringsAsFactors=FALSE)
```

```
rownames(rpde) <- NULL
head(rpde)
```

```
   V1   V2
1 Ach  -27
2 Aeg  -30
3 Afr -146
4 AlC   63
5 AlM   63
6 AlP   63
```

Because the dates have a numerical format from function `dts()`, the data frame allows producing a chronology of affiliation dates for the provinces and regions to the Roman Empire by ordering the second variable in `rpde`.

```
# order of affiliation of provinces
rpde[order(as.numeric(rpde$V2)),1]
```

```
 [1] "Ita" "Aem" "ApC" "BrL" "Etr" "LaC" "Lig" "Pic" "Sam" "Tra" "Umb" "VeH" "Sic" "Cor" "Sar"
[16] "Bae" "HiC" "Lus" "Epi" "Mak" "Afr" "Asi" "Nar" "BiP" "Cyr" "Cre" "Cil" "Syr" "Cyp" "Aqu"
[31] "Bel" "Lug" "Dal" "Aeg" "Ach" "GeI" "GeS" "Gal" "Nor" "Rae" "Iud" "MoI" "MoS" "PaI" "PaS"
[46] "Cap" "MaC" "MaT" "Bri" "LyP" "Thr" "AlC" "AlM" "AlP" "Ara" "Dac" "Arm" "Ass" "Mes"
```

The regions in the Italian peninsula have the earliest affiliation dates, and Mesopotamia has the latest affiliation date to the Roman Empire.

Roman influence periods

- Dataset "`rpcp`" has influence periods of the Roman Empire.

```
# list with 45 early and late influence dates provinces
data("rpcp")
```

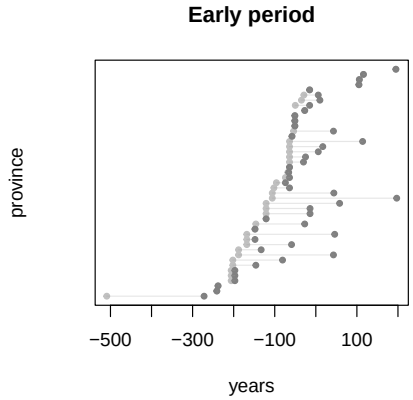
```
# look at data internal structure
str(rpcp)
```

```
List of 2
 $ Early:'data.frame':  45 obs. of  3 variables:
  ..$ Province: chr [1:45] "Italia (Final Consolidation)" "Sicilia" "Sardinia & Corsica" "Hisp
  ..$ EarInf  : num [1:45] -509 -241 -238 -206 -206 -206 -202 -202 -188 -188 ...
  ..$ OffPrv  : num [1:45] -272 -241 -238 -197 -197 -197 -146 -81 43 -133 ...
 $ Late :'data.frame':  45 obs. of  3 variables:
  ..$ Province: Factor w/ 45 levels "Achaea","Aegyptus",..: 30 43 42 27 28 26 3 23 32 9 ...
  ..$ LateInf : num [1:45] 476 436 436 409 409 ...
  ..$ Fall    : num [1:45] 476 436 436 409 409 409 409 418 1400 1500 ...
```

Early period of Roman influence

Visualize time intervals of early Roman influence in provinces and regions.

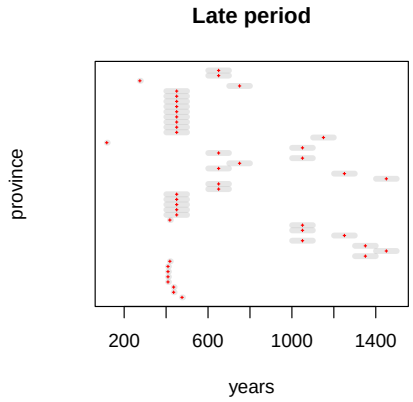
```
# early influence dates are in first list of 'rpcp'
plot.dates(x=rpcp[[1]], taq="EarInf", tpq="OffPrv", main="Early period", ylab="province")
```



Late period and fall from the Roman Empire

Time intervals of late Roman influence in provinces and regions depicted with mid points and range interval if longer than one.

```
# late influence dates are in second list of 'rpcp'
plot.dates(x=rpcp[[2]], type="mp", taq="LateInf", tpq="Fall", lwd=5, col="red",
           main="Late period", ylab="province")
```



Restricted imputation of missing dating data

- Dataset `rpd` has time intervals for `"not_before"` and `"not_after"` that corresponds to the dating data in the `EDH` dataset.

```
# Roman provinces dates from EDH
data("rpd")
```

```
# Rome
summary(rpd$Rom)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -301.0    50.0   372.0   330.2   652.2   878.0
```

```
# Aegyptus
summary(rpd$Aeg)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -71.00   90.25  322.00  286.00  517.75  571.00
```

These intervals are the basis for a restricted imputation of missing dating data in `EDH`

Imputation of dates by province

Function `edhwpd()` constructs, for a chosen province, a list of data frames with the components made of its inscriptions related by attribute co-occurrences. The replacement of missing dates occurs in this setting with function `rmids()` that stand for *restricted multiple imputation on data subsets*.

An example of restricted multiple imputations is the province of **Armenia** which has the fewest inscriptions in the `EDH` dataset. Dataset `rpd` is a list where each component corresponds to a province and where the component class provides the `HD ids` of inscriptions.

```
# Armenia
rpd$Arm
```

```
[1] 116 114 116   2
attr(,"class")
[1] "HD015521" "HD015524" "HD029916"
```

Imputation of inscriptions by similarity

Imputation from similarities of attribute variables per province and dates is organised with wrapper function `edhwpd()` having different argument options.

```
# list with arguments
formals(edhwpd)
```

```
$x
[1] "EDH"

$vars


$province


$dates


$clean


$...
```

By default, the input data for this function is the `EDH` dataset and the organisation is based on characteristics of the artefacts in `vars`.

```
# characteristics of inscriptions
vars = c("findspot_ancient", "type_of_inscription", "type_of_monument", "language")
```

Function `rmids()` performs the multiple imputation of missing dating data in `EDH` by default or in another dataset as input. In the case of `Arm`, record `HD015521` has censored data in dates while the other two records have complete missing dating data.

```
# Armenia: restricted imputation of dates
edhwpd(vars=vars, province="Arm") |>
  rmids()
```

```
Warning in edhwpd(vars = vars, province = "Arm"): "x" is for dataset "EDH".

Warning in rmids(edhwpd(vars = vars, province = "Arm")): max TPQ taken from province.

Warning in rmids(edhwpd(vars = vars, province = "Arm")): avg len TS taken from province.

Warning in rmids(edhwpd(vars = vars, province = "Arm")): avg taken from province.

Warning in rmids(edhwpd(vars = vars, province = "Arm")): min TAQ taken from province.

Warning in rmids(edhwpd(vars = vars, province = "Arm")): max TPQ taken from province.

Warning in rmids(edhwpd(vars = vars, province = "Arm")): avg len TS taken from province.

[[1]]
[[1]][[1]]
[[1]][[1]]$`taq-NA`
             id type_of_monument            type_of_inscription not_before not_after languag
15521.1 HD015521           tabula building/dedicatory inscription       0116       116    Lati
15521.2 HD015521           tabula building/dedicatory inscription       0116       118    Lati
        findspot_ancient
15521.1    Artaxata, bei
15521.2    Artaxata, bei

[[1]][[1]]$`NA-NA`
             id type_of_monument type_of_inscription not_before not_after language
15524.1 HD015524            stele             epitaph        116       116    Latin
15524.2 HD015524            stele             epitaph        116       118    Latin
15524.3 HD015524            stele             epitaph        114       116    Latin
15524.4 HD015524            stele             epitaph        116       116    Latin
        findspot_ancient
15524.1    Artaxata, bei
15524.2    Artaxata, bei
15524.3    Artaxata, bei
15524.4    Artaxata, bei



[[2]]
[[2]]$`NA-NA`
             id type_of_monument type_of_inscription not_before not_after language
32270.1 HD029916             <NA>                <NA>        114       116    Latin
32270.2 HD029916             <NA>                <NA>        114       116    Latin
32270.3 HD029916             <NA>                <NA>        114       116    Latin
32270.4 HD029916             <NA>                <NA>        116       116    Latin
        findspot_ancient
32270.1             <NA>
32270.2             <NA>
32270.3             <NA>
32270.4             <NA>
```

```
attr(,"class")
[1] EDH Arm 3   9
```

The warnings tell us that the imputation values are taken from the respective province in the `rpd` dataset where `avg len TS` stands for *average length of timespan*, `min TAQ` is the minimum value of `not_before`, and `max TPQ` is the maximum value of `not_after`.

Pooling results

Since there are multiple imputations of missing dating data, one next step is to combine the data by pooling rules of the m results from function `rmids()` into final point estimates plus standard error.

Pooling options for time intervals are take:

- average time-span with `avg len TS`
- `min TAQ` and `max TPQ`
- `max TAQ` and `min TPQ`

With these options, there is a single imputed value per variable with implied consequences.

Cartographical maps and networks

Antonio Rivero Ostoic

September 2022

```
# load and check versions
library(sdam)
packageVersion("sdam")
```

```
[1] '1.0.0'
```

Cartographical maps

Cartographical maps of Roman provinces under Emperors Trajan and Hadrian (year 117AD), and Italian regions under Emperor Augustus (year 27 BC) are part of the suite of datasets of the `"sdam"` package. For instance, four cartographical maps related to the Roman Empire in the antiquity period are available in dataset `"retn"` that is loaded by function `plot.map()` to depict cartographical maps and transportation systems. `plot.map()` is an interface that also invokes datasets `"rpmp"` for names and vector shapes of Roman provinces and regions, and dataset `"rpmcd"` for including map captions and dates in the plot.

Political division

Plotting political divisions of the Roman Empire needs the `'type'` argument of function `plot.map()` as well.

```
# Roman provinces
plot.map(type="rp")
```

```
# senatorial/imperial division
plot.map(type="si", main="Roman Empire (AD 117)")
```

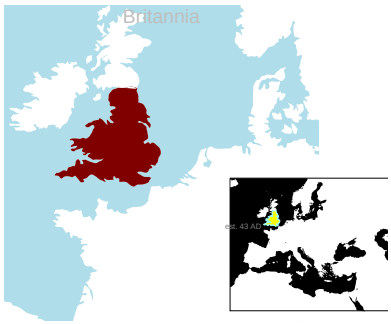

Roman Empire (AD 117)

Roman provinces and regions

- Roman provinces and regions shapes and colours for `plot.map()` are in dataset `"rpmp"`, while the acronyms in `x` are as in dataset `"rp"`.

```
# Italian peninsula silhouette
plot.map(x="Ita")
```


```
# Roman province with establishment date
plot.map(x="Bri", date=TRUE)
```



```
# Italian region
plot.map(x="VeH", cap=FALSE, fsize=12)
```

Transportation system

A `plain` cartographical map with a Roman transportation system made of settlements and terrestrial and maritime routes is possible with the `type` argument in `plot.map()`, or else by specifying with arguments `settl`, `roads`, and `shipr` for the settlements and this routes in the transportation system.

```
# settlements and main roads
plot.map(type="plain", settl=TRUE, roads=TRUE)
```

```
# settlements and main shipping routes
plot.map(type="plain", settl=TRUE, shipr=TRUE)
```

Graphs

Shipwrecks network

Graph representations of the transport network is possible with packages "multigraph" and "multiplex".

Access to the shipwrecks dataset.

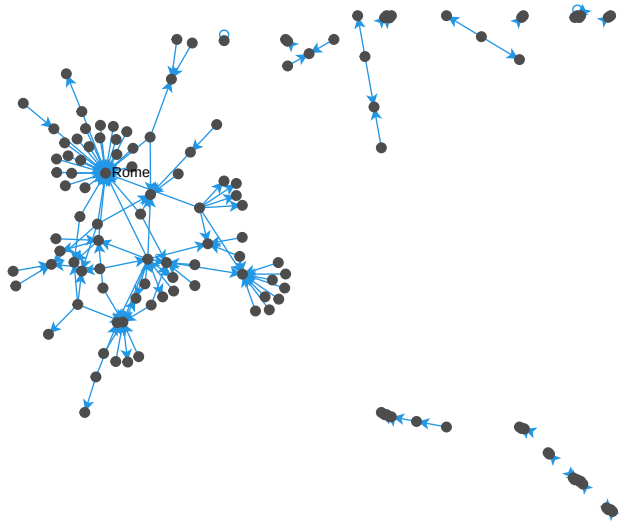
```
# shipwrecks dataset
sw <- system.file("extdata","StraussShipwrecks.csv",package="sdam") |>
  read.csv(sep=";")
```

```
# variables are checked
colnames(sw)
```

```
 [1] "Wreck.ID"              "Strauss.ID"              "Name"
 [4] "Parker.Number"         "Sea.area"                "Country"
 [7] "Region"                "Latitude"                "Longitude"
[10] "Min.depth"             "Max.depth"               "Depth"
[13] "Period"                "Dating"                  "Earliest.date"
[16] "Latest.date"           "Date.range"              "Mid.point.of.date.range"
[19] "Probability"           "Place.of.origin"         "Place.of.destination"
[22] "Reference"             "Comments"                "Amphorae"
[25] "Marble"                "Columns.etc"             "Sarcophagi"
[28] "Blocks"                "Marble.type"             "Other.cargo"
[31] "Hull.remains"          "Shipboard.paraphernalia" "Ship.equipment"
[34] "Estimated.tonnage"     "Amphora.type"
```

A system of maritime routes is found in variables `Place.of.origin` and `Place.of.destination`, which correspond to columns 20 to 21 in `sw`. This edge list needs some transformations with functions from packages "sdam" and "multiplex" to be able to plot the shipwrecks network as a directed graph with loops with a force directed layout.

```
# graph of shipwrecks network
sw[, 20:21] |>
  sdam::cln(level=2, what=c("(",")"), case=1, na.rm=TRUE) |>
  multiplex::transf(type="toarray") |>
  multigraph::multigraph(layout="force", seed=123, loops=TRUE, ecol=4, sel="Rome")
```



In this case, a single node is labeled in the graph that represents the shipwrecks network without missing information. To keep records with `NA` in the data, function `cln()` has the `'na.rm'` argument to be set to `FALSE`.

References for shipwrecks data are in

- Vignette: Datasets in `"sdam"` package

See also

- `"sdam"` manual
- `sdam`: Social Dynamics and Complexity in the Ancient Mediterranean

Project

- Release candidate version
- Code snippets using `"sdam"`
- sdam: Digital Tools for the SDAM Project at Aarhus University