

Package ‘MaddisonData’

November 24, 2025

Title Maddison Project Data

Version 1.0.2

Date 2025-11-20

Description Relatively easy access is provided to Maddison project data, which collates all the credible data on population and GDP for 169 countries, with some dating back to the year 1. 'MaddisonLeaders' makes it easy to find the leaders for each year, allowing users to delete countries like OPEC with narrow economies to focus on the technology leaders. 'ggplotPath' makes it easy to plot data for only selected countries or years.

License MIT + file LICENSE

URL <https://github.com/sbgraves237/MaddisonData>

BugReports <https://github.com/sbgraves237/MaddisonData/issues>

Depends R (>= 4.1)

Suggests ggplot2, ipumsr, KFAS, knitr, lubridate, readxl, rmarkdown, testthat (>= 3.0.0), tibble, usethis

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author Spencer Graves [aut, cre] (ORCID:
<<https://orcid.org/0009-0005-5387-729X>>)

Maintainer Spencer Graves <spencer.graves@effectivedefense.org>

Repository CRAN

Date/Publication 2025-11-24 18:10:08 UTC

Contents

getMaddisonSources	2
ggplotPath	3
logMaddison	5
MadDateRanges	6
MaddisonCountries	6
MaddisonData	7
MaddisonLeaders	8
MaddisonSources	9
path_package2	11

Index	13
--------------	-----------

getMaddisonSources	<i>Get Maddison sources</i>
--------------------	-----------------------------

Description

The **Maddison project** collates historical economic statistics from many sources.

They have a citation policy: **CONDITIONS UNDER WHICH ALL ORIGINAL PAPERS MUST BE CITED:**

a) If the data is shown in any graphical form b) If subsets of the full dataset that include less than a dozen (12) countries are used for statistical analysis or any other purposes

When neither a) or b) apply, then the MDP as a whole can be cited.

getMaddisonSources returns a [data.frame](#) of relevant sources for a particular application.

Usage

```
getMaddisonSources(
  ISO = NULL,
  plot = TRUE,
  sources = MaddisonData::MaddisonSources,
  years = MaddisonData::MaddisonYears
)
```

Arguments

ISO	either NULL to return all sources or a character vector of ISO codes for the countries included in the analysis or a data.frame with the first column being the ISO codes followed by yearBegin and optionally yearEnd.
plot	logical indicating whether the use does nor does not include plotting data. The Maddison project requires citing all relevant MaddisonSources if they are plotted, denoted here by plot = TRUE. If no data are plotted, then the Maddison project requires citing all sources only if less than a dozen are used, denoted here by plot = FALSE, in which case, the Maddison project requires a specific project-level citation. Default = TRUE.

sources list of sources in the format of [MaddisonSources](#); default is `MaddisonSources`.
years `data.frame` in the format of [MaddisonYears](#); default is `MaddisonYears`.

Value

a `tibble::tibble` with 3 columns:

ISO 3-letter ISO code for country.

years character vector of years or year ranges for which source applies.

source character vector of sources.

in the format of [MaddisonSources](#).

Examples

```
getMaddisonSources() # all
getMaddisonSources(plot=FALSE) # only MDP
getMaddisonSources('GBR') # GBR
getMaddisonSources(names(MaddisonSources)[1:12], FALSE) # only MDP
getMaddisonSources(data.frame(ISO=c('GBR', 'USA'),
                               yearBegin=rep(1500, 2)) ) #GBR, USA since 1500
getMaddisonSources('AUS') # AUS: no special sources for AUS.
```

ggplotPath

ggplot paths

Description

`ggplotPath` plots `y` vs. `x` (typically year) with a separate line for each group with options for legend placement, vertical lines and labels.

Usage

```
ggplotPath(
  x = "year",
  y,
  group,
  data,
  scaley = 1,
  logy = TRUE,
  legend.position,
  vlines,
  labels
)
```

Arguments

<code>x</code>	name of column in data to pass as x in <code>aes(x=.data[[x]], ...)</code> ; default = year.
<code>y</code>	name of column in data to pass as y in <code>aes(y=.data[[y]], ...)</code> ; must be supplied.
<code>group</code>	name of grouping variable, i.e., plot a separate line for each level of group using <code>aes(group=.data[[group]], ...)</code> , unless group is missing or <code>length(unique(data[, group])) = 1</code> .
<code>data</code>	<code>data.frame</code> or <code>tibble::tibble</code> with columns x, y, and group.
<code>scaley</code>	factor to divide y by for plotting. Default = 1, but for data in monetary terms, e.g., for <code>MaddisonData</code> , y = 'gdppc' is Gross domestic product (GDP) per capita in 2011 dollars at purchasing power parity (PPP), for which we typically want <code>scaley = 1000</code> .
<code>logy</code>	logical: if TRUE, y axis is on a log scale; default = TRUE.
<code>legend.position</code>	argument passed to <code>ggplot2::theme</code> . Default depends on <code>nGps <- length(unique(data[, group])</code> : If <code>nGps = 1</code> , there is no legend. If <code>nGps > 10</code> , <code>legend.position = 'right'</code> . In between, <code>legend.position = c(.15, .5) = center left</code> . For alternatives, see <code>ggplot2::theme</code> .
<code>vlines</code>	= locations on the x axis for vertical lines using <code>ggplot2::geom_vline(aes(xintercept = .data[[x]]), data=vlines, ...)</code> with <code>color='grey'</code> , <code>lty='dotted'</code> unless <code>color</code> or <code>colour</code> and / or <code>lty</code> are available as <code>attr(x, ...)</code> .
<code>labels</code>	= <code>data.frame</code> with columns x, y, label, srt, col, where x, y, and srt are numeric, label is character, and col are acceptable values for color in <code>with(labels, annotate('text', x=x, y=y, label = label, srt=srt, color=col))</code> .

Value

an object of class `ggplot2::ggplot`, which can be subsequently edited, and whose `print` method produces the desired plot.

Examples

```
str(GBR_USA <- subset(MaddisonData::MaddisonData, ISO %in% c('GBR', 'USA')))
GBR_USA1 <- ggplotPath('year', 'gdppc', 'ISO', GBR_USA, 1000)

GBR_USA1+ggplot2::coord_cartesian(xlim=c(1500, 1850)) # for only 1500-1850
GBR_USA1+ggplot2::coord_cartesian(xlim=c(1600, 1700), ylim=c(7, 17))

# label the lines
IS011 <- data.frame(x=c(1500, 1750), y=c(1.4, .7), label=c('GBR', 'USA') )
(GBR_USA2 <- ggplotPath('year', 'gdppc', 'ISO', GBR_USA, 1000,
  labels=IS011) )

# vlines
Vlines = c(1849, 1929, 1933, 1939, 1945)
(GBR_USA3 <- ggplotPath('year', 'gdppc', 'ISO', GBR_USA, 1000,
```

```
vlines=Vlines, labels=ISO11) )
```

logMaddison	<i>Select countries and add logged variables</i>
-------------	--

Description

logMaddison returns a `tibble::tibble` of data on selected countries extracted from MaddisonData, appending columns `lnGDPpc` and `lnPop` = natural logarithms of `gdppc` and `pop`.

Usage

```
logMaddison(ISO = NULL)
```

Arguments

ISO either `NULL` to select all the data in `MaddisonData` or a character vector of ISO codes used in the Maddison project.

Value

a `tibble::tibble` with 6 columns:

ISO 3-letter ISO code for countries selected

year numeric year in the current era.

gdppc Gross domestic product per capita adjusted for inflation to 2011 dollars at purchasing power parity.

pop Population, mid-year (thousands)

lnGDPpc `log(gdppc)`

lnPop `log(pop)`

Examples

```
logMaddison() # all
logMaddison(c('GBR', 'USA')) # GBR, USA
```

MadDateRanges *Convert a vector of date ranges into a data.frame*

Description

MadDateRanges returns a `data.frame` with 3 numeric columns: `yearBegin`, `yearEnd`, and `sourceNum` from the vector of `dateRanges` associated with different sources in [MaddisonSources](#).

Usage

```
MadDateRanges(dateRanges)
```

Arguments

`dateRanges` character vector of date ranges, each associated with a different source.

Value

a `data.frame` with 3 columns

yearBegin, **yearEnd** numeric years

sourceNum 1, 2, 3, ... for the location in `dateRanges`

Examples

```
MadDateRanges(c('1', '700 - 1500', '1252-1700 (England)',
                '1915-1919 & 1949', '1820, 1870, 1913, 1950'))
# equal
data.frame(
  yearBegin=c(1, 700, 1252, 1820, 1870, 1913, 1950),
  yearEnd =c(1, 1500, 1700, 1820, 1870, 1913, 1950),
  sourceNum=c(1, 2, 3, rep(4, 4)))
```

MaddisonCountries *Maddison Project data*

Description

The **Maddison project** collates historical economic statistics from many sources. `MaddisonCountries` is a `data.frame` of all (countrycode, country, region) combinations in those data.

Usage

```
MaddisonCountries
```

Format

MaddisonCountries:
 A data frame with 3 columns:
ISO 3-letter ISO country code
country Country name used by the Maddison project
region Geographic region including country
 Its rownames = ISO.

Source

<https://www.rug.nl/ggdc/historicaldevelopment/maddison/releases/maddison-project-database-2020?lang=en>"Groningen Growth and Development Centre"

Examples

```
# Get the country for a countrycode (IS)
subset(MaddisonCountries, ISO=='GBR', country)
# Or
MaddisonCountries['GBR', 'country']
# Find Yugoslavia
subset(MaddisonCountries, grepl('Yugo', country), 1:3)
# number of countries by region
table(MaddisonCountries$region)
# What are "Western Offshoots"?
subset(MaddisonCountries, grepl('Of', region), c(country, ISO))
```

MaddisonData

Maddison Project data

Description

The **Maddison project** collates historical economic statistics from many sources. MaddisonCountries is a [data.frame](#) of all (countrycode, country, region) combinations in those data.

Usage

```
MaddisonData
```

Format

MaddisonData:
 A data frame with 4 columns:
ISO 3-letter ISO country code
year numeric year starting with year 1 CE
gdppc Gross domestic product (GDP) per capita in 2011 dollars at purchasing power parity (PPP)
pop Population, mid-year (thousands)

Source

<https://www.rug.nl/ggdc/historicaldevelopment/maddison/releases/maddison-project-database-2020?lang=en>"Groningen Growth and Development Centre"

Examples

```
# Get the countrycode for a country
subset(MaddisonCountries, country=='United Kingdom', ISO)
# Select
str(GBR <- MaddisonData[MaddisonData$ISO=='GBR', ])
```

MaddisonLeaders *Plot selected countries*

Description

MaddisonLeaders computes the countries with the highest gdppc for each year.

Usage

```
MaddisonLeaders(
  except = character(0),
  y = "gdppc",
  group = "ISO",
  data = MaddisonData::MaddisonData,
  x = "year"
)
```

Arguments

except	either NULL to select all the data in MaddisonData or a character vector of group codes to EXCLUDE, e.g., so the result reflects apparent technology leaders, excluding countries whose high gdppc may be due to a dominant position in a single commodity.
y	name of column in data to consider. Default = gdppc.
group	name of column in data as the grouping variable. Default = ISO.
data	data.frame or tibble::tibble with first two columns being ISO and year and y being the name of another column.
x	time variable. Default = year.

Value

an [data.frame](#) with columns

- `paste0(x, 'Begin')`,
- `paste0(x, 'End')`,

- `paste0(y, '0')`,
- `paste0(y, '1')`, and
- `{{group}}`

(defaults:

- `yearBegin`,
- `yearEnd`,
- `gdppc0`,
- `gdppc1`, and
- ISO, plus
- `dy0 = paste0(x, 'End') - paste0(x, 'Begin')` and
- `dy1 = c(tail(paste0(x, 'Begin'), -1) - head(paste0(x, 'End'), -1), NA)` (defaults: `dy0 = yearEnd - yearBegin` and `dy1 = c(tail(yearBegin, -1) - head(yearEnd, -1), NA)`)

with an attribute `LeaderByYear = a data.frame` with columns, `{{x}}`, `paste0('max', y)`, and `{{group}}` (defaults: `year`, `maxgdppc`, `ISO`).

Examples

```
Leaders0 <- MaddisonLeaders() # max GDPpc for each year.

# Presumed technology leaders without commodity leaders with narrow
# economies
Leaders1 <- MaddisonLeaders(c('ARE', 'KWT', 'QAT'))
# since 1600
MadDat1600 <- subset(MaddisonData, year>1600)
Leaders1600 <- MaddisonLeaders(c('ARE', 'KWT', 'QAT'), data=MadDat1600)
```

MaddisonSources

Maddison Project data

Description

The **Maddison project** collates historical economic statistics from many sources. `MaddisonSources` is a [list](#) of `tibble::tibble`s with ISO names giving the sources of GDP per capita for different years for the said country.

`MaddisonYears` is a [data.frame](#) giving `yearBegin` and `yearEnd` and the number of each source in `MaddisonSources` for each ISO.

Usage

`MaddisonSources`

`MaddisonYears`

Format

MaddisonSources:

A named list of `tibble::tibbles`, one for each country, named with the ISO country codes. Each tibble has one row for each source for the indicated ISO and two columns:

years character variable of year(s) for this source starting with year 1 CE.

source character variable giving the source for the years described.

In addition, MaddisonSources has an attribute `since2008`, which says, "gdppc since 2008: Total Economy Database (TED) from the Conference Board for all countries included in TED and UN national accounts statistics for all others."

MaddisonYears:

A `data.frame` with 4 columns:

ISO 3-letter country code.

yearBegin, yearEnd Integer year begin and end for each source.

sourceNum Integer of the source within `MaddisonSources[[ISO]]`.

An object of class `data.frame` with 133 rows and 4 columns.

Source

<https://www.rug.nl/ggdc/historicaldevelopment/maddison/releases/maddison-project-database-2020?lang=en>"Groningen Growth and Development Centre"

Examples

```
MaddisonSources[['GBR']]
MaddisonSources[['GBR']][, 1, drop=TRUE]
# = c('1', '1252-1700 (England)', '1700-1870')
# for data from the year 1
# and for England only between 1252 and 1700, etc.
```

```
MaddisonSources[['IRN']][, 1, drop=TRUE]
# = '1820, 1870, 1913, 1950'
# for those 4 years only.
```

```
MaddisonSources[c('GBR', 'USA')]
```

```
MaddisonSources[['GBR']][, 1, drop=TRUE]
# = c('1', '1252-1700 (England)', '1700-1870')
```

```
MaddisonYears[MaddisonYears$ISO=='GBR', ] =
data.frame(
  ISO=rep('GBR', 3),
  yearBegin=c(1, 1252, 1700),
  yearEnd =c(1, 1700, 1870),
  sourceNum=1:3
)
```

```
MaddisonSources[['EGY']][, 1, drop=TRUE]
```

```
# = c('1', '700 - 1500', '1820, 1870, 1913, 1950')

MaddisonYears[MaddisonYears$ISO=='EGY', ] =
data.frame(
ISO=rep('EGY', 6),
yearBegin=c(1, 700, 1820, 1870, 1913, 1950),
yearEnd =c(1, 1500, 1820, 1870, 1913, 1950),
sourceNum=c(1, 2, rep(3, 4))
)
```

path_package2	<i>Construct a path to a location within an installed or development package</i>
---------------	--

Description

path_package2 returns a character vector of matches to target. It differs from `system.file()` in that it supports searching for a target file or folder possibly in subdirs of the working directory or in nparents of its parents.

Usage

```
path_package2(
  target,
  package = NULL,
  nparents = 1,
  subdirs = c("extdata", paste("inst", "extdata", sep = .Platform$file.sep))
)
```

Arguments

target	A regular expression describing the file or folder desired.
package	Name of the package to in which to search. If NULL, search in the working directory. Otherwise search in <code>system.file(package)</code> .
nparents	integer indicate the number of parents of the working directory in which to search; default = 1.
subdirs	= <code>c('extdata', paste('inst', 'extdata', sep=.Platform\$file.sep))</code>

Details

This works in a vignette searching for a target that could be in the vignettes directory of its parent package or in the package directory or in, e.g., one of `subdirs = c('extdata', paste('inst', 'extdata', sep=.Platform$file.sep))`.

Returns the full path to match(s) if found and a character vector of length 0 if no matches are found. The returned object also has a searched attribute being a character vector of the directories searched.

This was inspired by a desire to share with others a vignette describing how to create data objects from a file that could not itself be shared on CRAN. This is not easy, because the working director available to code in a vignette changes depending on how that code is run.

`path_package2` allows the user to store the target locally, e.g., in `inst/extdata` but include it in `.gitignore` to prevent it from leaving the local computer. The vignette then decides what to do after calling `path_package2()` based on the length of the the object returned.

Value

a character vector with an attribute `searched` giving the full paths of all directories searched for target.

Examples

```
# search for a file matching a regular expression
path_package2('^mpd.*xlsx$')
# search only in the working directory
path_package2('^mpd.*xlsx$', nparents=0, subdirs=character(0))
```

Index

- * **datasets**
 - MaddisonCountries, 6
 - MaddisonData, 7
 - MaddisonSources, 9
- * **file**
 - path_package2, 11
- * **manip**
 - getMaddisonSources, 2
 - logMaddison, 5
 - MadDateRanges, 6
 - MaddisonLeaders, 8
- * **plot**
 - ggplotPath, 3

data.frame, 2, 4, 6–10

getMaddisonSources, 2

ggplot2::ggplot, 4

ggplot2::theme, 4

ggplotPath, 3

list, 9

logMaddison, 5

MadDateRanges, 6

MaddisonCountries, 6

MaddisonData, 7

MaddisonLeaders, 8

MaddisonSources, 3, 6, 9

MaddisonYears, 3

MaddisonYears (MaddisonSources), 9

path_package2, 11

print, 4

system.file(), 11

tibble::tibble, 3–5, 8–10