Package 'MaddisonData'

May 7, 2026

Title Maddison Project Data

Version 1.1.0

Date 2026-01-09

Description Relatively easy access is provided to 2023 version of the Maddison project data downloaded 2025-08-28. This project collates all the credible data on population and GDP for 169 countries, with some dating back to the year 1 of the current era. One function makes it easy to find the leaders for each year, allowing users to delete countries like OPEC with narrow economies to focus on technology leaders. Another function makes it easy to plot data for only selected countries or years. Another function makes it relatively easy to obtain references to the original sources, which must be cited per the copyright rules of the Maddison Project for different uses of their data.

License MIT + file LICENSE

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

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

Depends R (>= 4.1)

Language en-US

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 2026-01-10 01:00:02 UTC

Contents

`getMaddisonSources` *Get Maddison sources*

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 data.frame 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
GBR <- 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, horizontal and vertical lines and labels.

Usage

```
ggplotPath(
  x = "year",
  y,
  group,
  data,
  scaley = 1,
  logy = TRUE,
```

```
  ylab,
  legend.position,
  hlines,
  vlines,
  labels,
  fontsize = 10,
  color,
  linetype
)
```

Arguments

x	name of column in data to pass as x in aes(x=.data[[x]], ...); default = year.
y	name of column in data to pass as y in aes(y=.data[[y]], ...); must be supplied.
group	name of grouping variable, i.e., plot a separate line for each level of group using aes(group=.data[[group]], ...), unless group is missing or length(unique(data[, group])) = 1.
data	data.frame or tibble::tibble with columns x, y, and group.
scaley	factor to divide y by for plotting. Default = 1, but for data in monetary terms, e.g., for MaddisonData, y = 'gdppc' is Gross domestic product (GDP) per capita in 2011 dollars at purchasing power parity (PPP), for which we typically want scaley = 1000.
logy	logical: if TRUE, y axis is on a log scale; default = TRUE.‘
ylab	y axis label. Default = if(scaley==1) y else paste(y, '/', scaley)
legend.position	argument passed to ggplot2::theme. If !missing(labels), default is no legend. Otherwise, default depends on nGps <- length(unique(data[, group]): If nGps = 1, there is no legend. If nGps > 10, legend.position = 'right'. In between, legend.position = c(.15, .5) = center left. For alternatives, see ggplot2::theme.
hlines	numeric vector of locations on the y axis for horizontal lines using ggplot2::geom_hline(yintercept = hlines, ...) with color='grey', lty='dotted' unless color or colour and / or lty are available as attr(x, ...).
vlines	numeric vector of locations on the x axis for vertical lines using ggplot2::geom_vline(xintercept = vlines, ...) with color='grey', lty='dotted' unless color or colour and / or lty are available as attr(x, ...).
labels	= data.frame with columns x, y, label, and optionally srt, col, size, where x, y, srt, and size are are numeric, label is character, and col are acceptable values for color in with(labels, annotate('text', x=x, y=y, label = label, srt=srt, color=col, size=size)). Defaults for srt, col, and size are 0, 'black', and 4, respectively.
fontsize	for legend and axes labels in theme(text=element_text(size=fontsize)); default = 10.

color — for lines to pass to `scale_color_manual(values = color)` if present. If present, `length(color)` should equal `length(unique(data[, group]))`.

linetype — optional vector. Default `if(missing(group)) 1 else rep(1:6, length=length(unique(data[, group` Else either

- an integer (0-6), a name (0 = blank, 1 = solid, 2 = dashed, 3 = dotted, 4 = dotdash, 5 = longdash, 6 = twodash), or
- a mapping to a discrete variable, or
- a string of an even number (up to eight) of hexadecimal digits which give the lengths in consecutive positions in the string.

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 <- MaddisonData::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
ISOll <- data.frame(x=c(1500, 1800), y=c(2.5, 1.7), label=c('GBR', 'USA'),
              srt=c(0, 30), col=c('red', 'green'), size=c(2, 9))
GBR_USA2 <- ggplotPath('year', 'gdppc', 'ISO', GBR_USA, 1000,
                    labels=ISOll, fontsize = 20)

# h, vlines, manual legend only
Hlines <- c(1,3, 10, 30)
Vlines = c(1849, 1929, 1933, 1939, 1945)
(GBR_USA3 <- ggplotPath('year', 'gdppc', 'ISO', GBR_USA, 1000,
       ylab='GDP per capita (2011 PPP K$)',
       legend.position = NULL, hlines=Hlines, vlines=Vlines, labels=ISOll))
```

---

logMaddison *Select countries and add logged variables*

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. This object provides easy access to the 2023 version of the Maddison project data downloaded 2025-08-28.

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 *Identify leading 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 object of class c('MaddisonLeaders', 'data.frame'), with columns

- paste0(x, 'Begin'),
- paste0(x, 'End'),
- paste0(y, '0'),
- paste0(y, '1'), and
- {{group}}
- paste0('d', x, '0') = paste0(x, 'End') - paste0(x, 'Begin') + min(dx), where dx = min(diff(sort(unique(data[, x]))))
- paste0('d', x, '1') = c(tail(paste0(x, 'Begin'), -1) - head(paste0(x, 'End'), -1), NA) (defaults: dy0 = yearEnd - yearBegin +1 and dy1 = c(tail(yearBegin, -1) - head(yearEnd, -1), NA))

(defaults:

- yearBegin,
- yearEnd,
- gdppc0,
- gdppc1, and
- ISO, plus
- dyear0 = yearEnd - yearBegin + 1 and
- dyear1 = 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)

# max pop by region within percentiles of gdppc
noGDP <- is.na(MaddisonData$gdppc)
MadDat <-MaddisonData[!noGDP, ]
gdpPcts <- quantile(MadDat$gdppc, seq(0, 1, .01), na.rm=TRUE)
gdpPct <- unique(as.numeric(gdpPcts[-1]))
gdpPc <-c(gdpPct[-100], tail(gdpPct, 1)*(1+sqrt(.Machine$double.eps)))
gdp100 <- MadDat$gdppc
nObs <- nrow(MadDat)
for(i in 1:nObs){gdp100[i] <- min(gdpPc[MadDat$gdppc[i]<gdpPc])}
MadDat$gdp100 <- gdp100
MadDat$region <- MaddisonCountries[MadDat$ISO, 'region', drop=TRUE]
MadPopRgnGDP<-MaddisonLeaders(y='pop',group='region',data=MadDat,x='gdp100')
```

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 `MaddisonSpources` for each `ISO`.

Usage

```
MaddisonSources

MaddisonYears
```

Format

`MaddisonSources`**:**

A named list of `tibble::tibble`s, 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`s 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 *Construct a path to a location within an installed or development package*

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 of folder desired.
package	Name of the package to in which to search. If NULL, search in the working directory. Otherwise search in system.file(package).
nparents	integer indicate the number of parents of the working directory in which to search; default = 1.
subdirs	= c('extdata', paste('inst','extdata', sep=.Platform$file.sep))

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))
```

`summary.MaddisonLeaders`

Summary method for an object of class `MaddisonLeaders`

Description

`summary.MaddisonLeaders` returns a `data.frame` with columns `ISO`, `paste0(x, 'Begin)`, `paste0(x, 'End')`, `n`, and `p`.

Usage

```
## S3 method for class 'MaddisonLeaders'
summary(object, sortBy = "ISO", decreasing = FALSE, ...)
```

Arguments

`object`	= object of class `MaddisonLeaders`.
`sortBy`	= column of output used for sorting; default = `ISO`
`decreasing`	default = `FALSE`
`...`	= optional arguments for `summary` (not used)

Value

a `data.frame` with columns

- `ISO` = One row for each level of `ISO` in `unique(object[, 'ISO'])`
- `paste0(x, 'Begin)` = earliest `object[, paste0(x, 'Begin')]` for `ISO`
- `paste0(x, 'End')`, last `object[, paste0(x, 'End')]` for `ISO`
- `n` = sum of `(paste0(x, 'End') - paste0(x, 'Begin') + 1` for `ISO`.
- `p` = `n/(paste0(x, 'End') - paste0(x, 'Begin') + 1)`.

)

(defaults:

- ISO = One row for each level of ISO in unique(object[, 'ISO'])
- yearBegin = earliest object[, 'yearBegin')] for ISO
- yearEnd = last object[, 'yearEnd')] for ISO
- n = sum of ('yearEnd' - 'yearBegin' + 1) for ISO.
- p = n/(yearEnd - yearBegin + 1).

[, 'yearBegin')]: R:,%20'yearBegin') [, 'yearEnd')]: R:,%20'yearEnd')

Examples

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

---

yr *year with fraction'*

Description

yr converts a Date to a year and fraction. For example, 2025-01-01 becomes 2025.00000, while 2025-01-02 becomes 2025.00234, because (2-1)/365 is 0.00234 to 5 significant digits. However, 2024-01-02 becomes 2024.0233, because (2-1)/366 is only 0.00233 to 5 significant digits.

Usage

```
yr(x, ...)
```

Arguments

x	quantity that can be converted to a Date object using as.Date(x).
...	arguments passed to lubridate::ymd().

Value

a number (numeric vector).

See Also

lubridate::decimal_date(), lubridate::ymd()

Examples

```
Jan2_24_25 <- c('2024-01-02', '2025-01-02')
J2yr <- yr(Jan2_24_25)
J2y <- yr(as.POSIXct(Jan2_24_25))
all.equal(J2yr, J2y)
```

Index