Package 'repijson'

August 29, 2016

Title Tools for Handling EpiJSON (Epidemiology Data) Files

Version 0.1.0

Description Supplies classes and routines to convert data to and from EpiJSON files. This package provides conversion functions for data.frame, sp and obkData.

Depends R (>= 3.1.0)

Imports OutbreakTools, jsonlite, plyr, sp, ggplot2, geojsonio

Suggests HistData, knitr, testthat, rgdal, base64enc

VignetteBuilder knitr

License GPL (>= 2)

LazyData true

NeedsCompilation no

Author Andy South [aut, cre],
Thomas Finnie [aut],
Ellie Sherrard-Smith [aut],
Ana Bento [aut],
Thibaut Jombart [aut]

Maintainer Andy South <southandy@gmail.com>

Repository CRAN

Date/Publication 2015-06-16 01:19:04

R topics documented:

`as.data.frame.ejAttribute` *Convert an ej attribute to a dataframe*

Description

Convert an ej attribute to a dataframe

Usage

```
as.data.frame.ejAttribute(x)
```

Arguments

x	An ejAttribute

as.data.frame.ejObject *convert an ejObject to a dataframe with one row per record*

Description

convert an ejObject to a dataframe with one row per record

Usage

```
## S3 method for class 'ejObject'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	an ejObject
row.names	NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
optional	not used.
...	other parameters passed to data.frame.

Value

dataframe

as.ejObject *generic as function*

Description

generic as function

Usage

```
as.ejObject(x, ...)
```

Arguments

x	an object to convert to repijson
...	other parameters to pass to the converter

`as.ejObject.data.frame` *Convert a dataframe to an ejObject*

Description

Convert a dataframe to an ejObject

Usage

```
## S3 method for class 'data.frame'
as.ejObject(x, recordID = NA, recordAttributes,
  eventDefinitions, metadata = list(), ...)
```

Arguments

`x`	The dataframe to convert
`recordID`	an ID for the record, if NA one is created
`recordAttributes`	A character vector containing the names of the columns in the dataframe that are attributes of the record
`eventDefinitions`	A list of event definitions
`metadata`	A list of metadata ejAttribute objects describing the dataset
`...`	other parameters (to maintain consistency with the generic)

Note

We assume one row per record.

Examples

```
#Here we use the toyll (toy line-list) data provided inside the package
data(toyll)
toyll
#Here we some data clean-up (make the date columns POSIX objects)
#Date-time conversion could be made automatic but introduces
#a big risk of mis-conversion so we leave this to the user
#to ensure that we don't silently corrupt their data

toyll$date.of.onset <- as.POSIXct(toyll$date.of.onset)
toyll$date.of.admission <- as.POSIXct(toyll$date.of.admission)
toyll$date.of.discharge <- as.POSIXct(toyll$date.of.discharge)
toyll$contact1.date <- as.POSIXct(toyll$contact1.date)
toyll$contact2.date <- as.POSIXct(toyll$contact2.date)
toyll$contact3.date <- as.POSIXct(toyll$contact3.date)
```

```
ind.fields <- c(names(toyll)[1:5], "hospital", "fever", "sleepy")
x <- as.ejObject(toyll,
                 recordAttributes=ind.fields,
                 eventDefinitions=list(
                 define_ejEvent(name="admission", date="date.of.admission"),
                 define_ejEvent(name="discharge", date="date.of.discharge"),
              define_ejEvent(name="contact1", date="contact1.date", attributes="contact1.id"),
              define_ejEvent(name="contact2", date="contact2.date", attributes="contact2.id"),
              define_ejEvent(name="contact3", date="contact3.date", attributes="contact3.id")
                 ))
print(x)
```

`as.ejObject.default` *By default we don't know how to convert objects*

Description

By default we don't know how to convert objects

Usage

```
## Default S3 method:
as.ejObject(x, ...)
```

Arguments

`x`	an object
`...`	other parameters passed to the call

`as.ejObject.obkData` *This function processes objects from the obkClass data and converts to the epiJSON format*

Description

This function processes objects from the obkClass data and converts to the epiJSON format

Usage

```
## S3 method for class 'obkData'
as.ejObject(x, metadata = list(), ...)
```

Arguments

`x`	An record from the obkData
`metadata`	The list of the components in the metadata
`...`	other parameters (to maintain consistency with the generic)

Value

an ejObject

Note

There is a slight mismatch in symantics here obkData individuals are equivelent to EpiJSON records and obkData records are EpiJSON events. This is beause in EpiJSON the unit of record is not necessarily an individual (it could, for example, be a region or hospital, etc).

Examples

```
require('OutbreakTools')
data(ToyOutbreak)
x=subset(ToyOutbreak,2)
as.ejObject(x, metadata=list())
```

---

as.SpatialPointsDataFrame.ejEvent

convert an ejEvent to a SpatialPointsDataFrame

Description

convert an ejEvent to a SpatialPointsDataFrame

Usage

```
as.SpatialPointsDataFrame.ejEvent(x)
```

Arguments

x	An ejEvent object

---

as.SpatialPointsDataFrame.ejObject

Create a SpatialPointsDataFrame from an ejObject

Description

Create a SpatialPointsDataFrame from an ejObject

Usage

```
as.SpatialPointsDataFrame.ejObject(x)
```

Arguments

x	An ejObject

as.SpatialPointsDataFrame.ejRecord *convert an ejRecord to a SpatialPointsDataFrame*

Description

convert an ejRecord to a SpatialPointsDataFrame

Usage

```
as.SpatialPointsDataFrame.ejRecord(x)
```

Arguments

x	an ejRecord object

attributeAsJSON *Convert an attribute into JSON*

Description

Convert an attribute into JSON

Usage

```
attributeAsJSON(attribute)
```

Arguments

attribute	The attribute object to be converted to JSON

box2 *generic labelled box function*

Description

generic labelled box function

Usage

```
box2(xmin = 0.01, xmax = 0.99, ymin = 0.01, ymax = 0.99,
  label = "box2", colour = "gray60", fill = "white", gg = NULL,
  addSheets = 0, size = 4, print = FALSE)
```

Arguments

xmin	0 to 1
xmax	0 to 1
ymin	0 to 1
ymax	0 to 1
label	to print at top of box
colour	border of box
fill	fill colour for box NA=none
gg	ggplot object, if passed box is added if NULL new ggplot object created
addSheets	how many sheets to add behind box to indicate array
size	font size for label
print	TRUE/FALSE whether to print the ggplot object

Value

a ggplot object

---

create_ejAttribute *Create an attribute This package outlines the aspects of the data for EpiJSON*

Description

This function defines attributes output `ejAttribute`

Usage

```
create_ejAttribute(name, type, value, units = NA)
```

Arguments

name	name of the attribute
type	type of data 'string', 'number', 'integer', 'boolean', 'date' or 'base64'
value	value of this attribute
units	The units for value. May be omitted.

Value

an ejAttribute object

Examples

```
characterAttribute <- create_ejAttribute(name="Format name", type="string",
 value="EpiJSON!")
numericAttribute <- create_ejAttribute(name="Width of building", type="number",
 value=5.2,"metres")
integerAttribute <- create_ejAttribute(name="Days since last accident", type="integer",
 value=as.integer(2))
logicalAttribute <- create_ejAttribute(name="Customer satisfied", type="boolean",
 value=TRUE)
dateAttributeOne <- create_ejAttribute(name="Birthdate", type="date",
 value=as.Date("1998-08-21"))
dateAttributeTwo <- create_ejAttribute(name="Lunch", type="date",
 value=as.POSIXct("2015-05-06 12:30"))
if (require(base64enc, quietly=TRUE)){
binaryAttribute <- create_ejAttribute(name="Lunch", type="base64",
   value=base64encode(as.raw(0:255)))
}
```

`create_ejEvent` *Create an event*

Description

This function defines events output `ejEvent`

Usage

```
create_ejEvent(id = NA, name, date = NULL, location = NULL,
  attributes = list())
```

Arguments

`id`	identifier for the event
`name`	name of the event, usually a column name
`date`	date (or timestamp) for event
`location`	location for event
`attributes`	list of attributes associated with this event

Value

an ejEvent object

Examples

```
#' #generate a polygon
library(sp)
polyPoints <- matrix(c(526870,181390,526817,181447,526880,181467,
526885,181447,526909,181425,526870,181390),ncol=2,byrow=TRUE)
demoPolygon <- sp::SpatialPolygons(list(sp::Polygons(list(sp::Polygon(polyPoints)),"1")),
proj4string=sp::CRS("+init=epsg:27700"))


#Create an attribute
integerAttribute <- create_ejAttribute(name="Days since last accident", type="integer",
  value=integer(2))
logicalAttribute <- create_ejAttribute(name="Customer satisfied", type="boolean",
  value=TRUE)

#create an event
event <- create_ejEvent(id=1, name="A test Event", date=Sys.time(),
location=demoPolygon, attributes=list(integerAttribute, logicalAttribute))
```

`create_ejMetadata` *Create metadata*

Description

This function defines epiJSON Metadata output `ejMetadata`

Usage

```
create_ejMetadata(attributes)
```

Arguments

`attributes`	list of attributes of the metadata

Value

an ejMetadata object

create_ejObject *Create an object*

Description

This function defines epiJSON objects output `ejObject`

Usage

```
create_ejObject(metadata, records)
```

Arguments

`metadata`	metadata for the whole ejObject
`records`	list of records

Value

an ejObject object

create_ejRecord *Create a record*

Description

This function defines records output `ejRecord`

Usage

```
create_ejRecord(id, attributes, events)
```

Arguments

`id`	This is the unique identifier of the record, usually a column name and the essential information for any data
`attributes`	list of attributes associated with this record
`events`	list of events associated with this record

Value

an ejEvent object

Examples

```
#somewhere on South Bank
demoPoints <- sp::SpatialPoints(data.frame(lat=51.4982778, long=-0.0975535),
proj4string=sp::CRS("+init=epsg:4326"))
```

dataFrameToAttributes *The general functions that are used in multiple sections of the epiJSON package*

Description

The general functions that are used in multiple sections of the epiJSON package

Usage

```
dataFrameToAttributes(x)
```

Arguments

x	A dataframe

Value

result A list of attributes that comprise the dataframe Convert a dataframe to attributes

Examples

```
dF<- data.frame(id=c("A","B","3D"),name=c("tom","andy","ellie"),
                dob=c("1984-03-14","1985-11-13","1987-06-16"),
                gender=as.factor(c("male","male","female")),
                rec1contact=c(2,1,5),
                rec1date=as.POSIXct(c("2014-12-28","2014-12-29","2015-01-03")),
                rec1risk=c("high","high","low"),
                rec1temp=c(39.5,41.3,41.8),
                rec2contact=as.integer(c(4,1,1)),
                rec2date=as.POSIXct(c("2015-01-02","2015-01-12","2015-01-09")),
                rec2risk=c("high","low","high"),
                logical=c(FALSE,TRUE,TRUE),
   stringsAsFactors=FALSE)

repijson:::dataFrameToAttributes(dF)
```

define_ejEvent *Creates a event definition*

Description

Simplifies the definition of events from columns within a dataframe

Usage

```
define_ejEvent(id = NA, name = NULL, date = NULL, location = NULL,
  attributes = NULL)
```

Arguments

id	A character string naming the column that defines the id for a event. May be NA, and if so will be automatically generated.
name	Either a character string with the event name. Or the name of the column that contains names for events as a character string. If the column name is found in the input data.frame to as.ejObject.data.frame then the data in the column is used otherwise the string itself is used. Event names might be things such as infection, swab, hospital admission,etc.
date	A character string naming the column that defines the date an event occured. This should be in POSIXct format. May be NA.
location	A list with entities x, y and proj4string. x and y should be character strings naming the columns where the x and y of the location are defined. crs may be "" or a proj4string.
attributes	A character vector naming the columns for attributes of the event. The attributes will be named after the columns, with type taken from column type.

---

ejAttributes *Extract the attributes from an EpiJson Object*

Description

Get the attributes as a list from an EpiJSON Object

Usage

```
ejAttributes(x)
```

Arguments

x	An EpiJSON object (class ejObject, ejRecord or ejEvent)

Value

A list of attributes

ejRecords *Extract the records from an EpiJson Object*

Description

Get the records as a list from an EpiJSON Object

Usage

```
ejRecords(x)
```

Arguments

x	An EpiJSON object (class ejObject)

Value

A list of records

Note

This is functionally equivelent to calling x$records (which may be quicker as therer is no object type checking) but open to change in later versions.

epiJSON2r *converting epiJSON into a list of lists*

Description

takes an epiJSON string or file and converts to a list of lists later this needs to be made more formal

Usage

```
epiJSON2r(file)
```

Arguments

file	an epiJSON filename or string to convert to R

Value

a list of lists of the epijson content

Examples

```
listJSON <- epiJSON2r( system.file("extdata//example.JSON", package="repijson"))
str(listJSON)
#from within the package would do this
#listJSON <- epiJSON2r("extdata//example.JSON")
```

`epijsonObjectVis` *to view the structure of epijson objects and/or schema allowing multiple attribute boxes with different labels*

Description

to view the structure of epijson objects and/or schema allowing multiple attribute boxes with different labels

Usage

```
epijsonObjectVis(attribMeta = "attributes [name, type, value, units]",
  attribRecord = "attributes [name, type, value, units]",
  attribEvent = "attributes [name, type, value, units]",
  labelObject = "Diagram of EpiJSON file structure", labelMeta = "metadata",
  labelRecord = "records [id]",
  labelEvent = "events [id, name, date, location]", colAll = NULL,
  colObject = "gray60", colMeta = "gray60", colRecord = "blue",
  colEvent = "red", colAttrib = "purple", textSize = 4)
```

Arguments

`attribMeta`	label(s) for Metadata attributes
`attribRecord`	label(s) for Record attributes
`attribEvent`	label(s) for Event attributes
`labelObject`	label for the overall object
`labelMeta`	label for Metadata attributes
`labelRecord`	label for Record attributes
`labelEvent`	label for Event attributes
`colAll`	optional single box colour to overide all other col args, e.g. 'grey'
`colObject`	object box colour
`colMeta`	metadata box colour
`colRecord`	record box colour
`colEvent`	event box colour
`colAttrib`	attribute boxes colour
`textSize`	size of labels default=4

Value

a ggplot object

Examples

```
#this gives the base schema
epijsonObjectVis()
#settin single box colour and increasing text size
epijsonObjectVis(colAll ='grey', textSize=7)
#this gives a diagram with named attributes
epijsonObjectVis( attribMeta = c("attribute: disease","attribute: data provider"),
                   attribRecord = c("attribute: gender","attribute: date of birth"),
                   attribEvent = c("attribute: recorder","attribute: test used") )
epijsonObjectVis( attribMeta = c("a"),
                   attribRecord = c("b","c"),
                   attribEvent = c("d","e","f") )
#repijson objects
epijsonObjectVis( attribMeta = 'ejAttribute',
                  attribRecord = 'ejAttribute',
                  attribEvent = 'ejAttribute',
                  labelObject = 'ejObject',
                  labelMeta = 'ejMetadata',
                  labelRecord = 'ejRecord',
                  labelEvent = 'ejEvent')
#repijson objects and constructors
epijsonObjectVis( attribMeta = 'ejAttribute create_ejAttribute()',
                   attribRecord = 'ejAttribute create_ejAttribute()',
                   attribEvent = 'ejAttribute create_ejAttribute()',
                labelObject = 'repijson R objects and constructors : ejObject create_ejObject()',
                   labelMeta = 'ejMetadata create_ejMetadata()',
                   labelRecord = 'ejRecord create_ejRecord()',
                   labelEvent = 'ejEvent create_ejEvent()')
```

`notNA` *Return a value only if another is not NA*

Description

Return a value only if another is not NA

Usage

```
notNA(x, trueValue, defaultValue = NA)
```

Arguments

`x`	The value to test for NA
`trueValue`	The value to return if x is not NA
`defaultValue`	The value to return if x is NA

Value

defaultValue if x is NA or trueValue if x is not NA

Note

Not terribly different from an ifelse but is more graceful with NULL values

---

`objectAsJSON` *Convert an ejObject into JSON*

Description

Convert an ejObject into JSON

Usage

```
objectAsJSON(object)
```

Arguments

`object`	The object to be converted to JSON

Examples

```
library(sp)
library(rgdal)

#create an attribute
attribute <- create_ejAttribute("A test attribute","number",5.2,"metres")

#generate a polygon
polyPoints <- matrix(c(526870,181390,526817,181447,526880,181467,
526885,181447,526909,181425,526870,181390),ncol=2,byrow=TRUE)
demoPolygon <- SpatialPolygons(list(Polygons(list(Polygon(polyPoints)),"1")),
proj4string=CRS("+init=epsg:27700"))

#create an event
event <- create_ejEvent(id=1, name="A test Event", date=Sys.time(),
location=demoPolygon, attributes=list(attribute, attribute))

#create a record
record <- create_ejRecord(id=1, attributes=list(attribute,attribute),
events=list(event,event))

#generate some metadata
metadata <- create_ejMetadata(list(attribute, attribute))

#create an EpiJSON object
object <- create_ejObject(metadata=metadata, records=list(record,record))

#print it as JSON
objectAsJSON(object)
```

print.ejAttribute *print an ejAttribute object*

Description

print an ejAttribute object

Usage

```
## S3 method for class 'ejAttribute'
print(x, ...)
```

Arguments

x	An ejAttribute object
...	Other arguments to print (not used)

print.ejEvent *print an ejEvent object*

Description

print an ejEvent object

Usage

```
## S3 method for class 'ejEvent'
print(x, ...)
```

Arguments

x	An ejEvent object
...	Other arguments to print (not used)

print.ejMetadata *print an ejMetadata object*

Description

print an ejMetadata object

Usage

```
## S3 method for class 'ejMetadata'
print(x, ...)
```

Arguments

x	An ejMetadata object
...	Other arguments to print (not used)

print.ejObject *print an ejObject object*

Description

print an ejObject object

Usage

```
## S3 method for class 'ejObject'
print(x, ...)
```

Arguments

x	An ejObject object
...	Other arguments to print (not used)

print.ejRecord *print an ejRecord object*

Description

print an ejRecord object

Usage

```
## S3 method for class 'ejRecord'
print(x, ...)
```

Arguments

x	An ejRecord object
...	Other arguments to print (not used)

processRecord *This function takes a single record from the obkClass data and converts to the epiJSON format*

Description

This function takes a single record from the obkClass data and converts to the epiJSON format

Usage

```
processRecord(x)
```

Arguments

x	An record from the obkData

Value

an ejRecord

Examples

```
## Not run:
#because this function is not exported this example won't work outside the package
require('OutbreakTools')
data(ToyOutbreak)
x <- subset(ToyOutbreak,1)
processRecord(x)

## End(Not run)
```

`processRecordFrame` *This function processes events from the obkClass data and converts to the epiJSON format*

Description

This function processes events from the obkClass data and converts to the epiJSON format

Usage

```
processRecordFrame(x, recordFrameName, eventID)
```

Arguments

`x`	An record from the obkData
`recordFrameName`	The event of interest
`eventID`	The id of the event

Value

an ejEvent

Examples

```
## Not run:
#because this function is not exported this example won't work outside the package
require('OutbreakTools')

data(ToyOutbreak)
x=subset(ToyOutbreak,2)@records[[1]]
processeventFrame(x,"Fever")

## End(Not run)
```

`repijson` *Classes and functions to handle EpiJSON files*

Description

EpiJSON is a universal JSON format for epidemiological data. More information on this format can be found at: http://github.com/Hackout2/EpiJSON

Details

repijson is a package implementing classes and functions for importing, exporting and converting EpiJSON files.

`toyll` *Toy line list dataset*

Description

This dataset replicates the structure of disease outbreak line lists, where the unit of observation is a case. It contains a mixture of patient meta-data (e.g. gender, name) and time-stamped records (e.g. hospital admission).

Usage

```
toyll
```

Format

A data.frame with 5 rows and 16 columns containing the following variables:

- id ID of the patient
- name name of the patient
- dob date of birth of the patient (format: yyyy-mm-dd)
- gender gender of the patient
- date.of.onset date of symptom onsets
- date.of.admission date of hospital admission
- date.of.discharge date of hospital discharge
- hospital hospital of admission
- fever has fever been observed? (yes/no)
- sleepy has the patient been sleepy? (yes/no)
- contact1.id ID of contact 1
- contact1.date date of contact 1
- contact2.id ID of contact 2
- contact2.date date of contact 2
- contact3.id ID of contact 3
- contact3.date date of contact 3

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>

Examples

```
data(toyll)
toyll

toyll$date.of.onset <- as.POSIXct(toyll$date.of.onset)
toyll$date.of.admission <- as.POSIXct(toyll$date.of.admission)
toyll$date.of.discharge <- as.POSIXct(toyll$date.of.discharge)
toyll$contact1.date <- as.POSIXct(toyll$contact1.date)
toyll$contact2.date <- as.POSIXct(toyll$contact2.date)
toyll$contact3.date <- as.POSIXct(toyll$contact3.date)

ind.fields <- c(names(toyll)[1:5], "hospital", "fever", "sleepy")
x <- as.ejObject(toyll,
                 recordAttributes=ind.fields,
                 eventDefinitions=list(
                 define_ejEvent(name="admission", date="date.of.admission"),
                 define_ejEvent(name="discharge", date="date.of.discharge"),
              define_ejEvent(name="contact1", date="contact1.date", attributes="contact1.id"),
              define_ejEvent(name="contact2", date="contact2.date", attributes="contact2.id"),
              define_ejEvent(name="contact3", date="contact3.date", attributes="contact3.id")
                 ))

x
```

Index