Package 'audited'

September 20, 2014

Type Package

Title Automatic Tracing and Display of Changes to Row Count

Version 1.9

Date 2014-09-18

Author Tim Bergsma

Maintainer Tim Bergsma <timb@metrumrg.com>

Imports igraph, metrumrg, methods

Suggests xlsx, rJava

Description Classifies a data.frame such that row deletions and additions are tracked. A mechanism exists to give formal names to the row subsets that are coming or going. These names are used to populate a directed graph giving an account of all the transactions contributing to the state of the data.frame. The generic as.audited() has a method for keyed data.frames that creates an audited data.frame. Methods exist that track row count changes for the generics: Ops, !, ^, |,[, subset, head, tail, unique, cast, melt, aggregate, and merge. audit() extracts the transaction table from the audited object, while write.audit() and read.audit() control exchange with the file system. An audit method for as.igraph() creates a graph object that can be displayed with the corresponding plot method. Use options(audit=) to provide an extra level of classification. Use options(artifact=TRUE) and as.xlsx() to save dropped records to file.

License GPL

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-20 22:03:15

R topics documented:

`audited-package` *Automatic Tracing and Display of Changes to Row Count*

Description

audited classifies a data.frame such that row deletions and additions are tracked. A mechanism exists to give formal names to the row subsets added or deleted. These names are used to populate a directed graph giving an account of all the transactions contributing to the state of the object.

Details

Package:	audited
Type:	Package
Version:	1.9
Date:	2014-09-18
License:	GPL3

Audited data frames keep track of what happens to them, so later you can create a graph showing row deletions and additions.

`as.audited` creates an `audited` data frame (which is also `keyed`). You can supply a name (id) or accept the default (same as object name). When you add or delete rows using supported methods, a transaction will be stored in the audit table (bound to the object) indicating the number of changed rows and the resulting total. If the data frame has a column listed in the vector `options('audit')` (set this using `options(audit=)`), unique values of that column (first match) will be tracked also. If `options('artifact')` is `TRUE` (default `NULL`) then dropped record sets will be stored in a list as the `artifact` attribute of the audited data frame. (Alternatively, this can be a vector with any of `drop`, `add`, `create`, `modify`, `transform` or `merge`: see `artifact` for details.) `as.xlsx` writes artifacts to a workbook.

Author(s)

Tim Bergsma

Maintainer: Tim Bergsma <timb@metrumrg.com>

References

http://github.com/bergsmat/audited, http://metrumrg.googlecode.com

See Also

as.audited plot.audited as.xlsx.audited metrumrg

Examples

```
options(audit='Subject')
options(artifact='drop')
rstudiogd()
pc <- data.frame(Theoph)
pc$Subject <- as.numeric(as.character(pc$Subject))
pc <- as.audited(pc,key=c('Subject','Time'))
ex <- pc[pc$Time == 0,,id='ex']
ex$EVID <- 1
pc$EVID <- 0
dat <- alias(id='dat', merge(pc, ex, all=TRUE))
dat <- dat[dat$Subject > 1,]
dat <- dat[dat$Wt > 70,,id='heavier subjects',od='lighter subjects']
audit(dat)
artifact(dat)

# default igraph aesthetics
plot(dat, list())

# unscaled audit aesthetics
plot(dat,scale=FALSE)

# default audit aesthetics
plot(dat)

# adjusting the label presentation
plot(dat, format='%a sub\n%l\n%r row')

# adjusting vertex inflation
plot(dat,inflation=1.2)

# adjusting vertex proportion
plot(dat,proportion=1.2)

# progressively longer audit trails
dat2 <- dat[dat$Subject != 2,,od='subj 2']
dat3 <- dat2[dat2$Subject != 3,,od='subj 3']
dat4 <- dat3[dat3$Subject != 4,,od='subj 4']
```

```
dat6 <- dat4[dat4$Subject != 6,,od='subj 6']
dat8 <- dat6[dat6$Subject != 8,,od='subj 8']
dat9 <- dat8[dat8$Subject != 9,,od='subj 9']

# vertex and edge proportions nearly constant across scale
# alternatively, use scale=FALSE and change canvas (image) proportionately
plot(dat)
plot(dat2)
plot(dat3)
plot(dat4)
plot(dat6)
plot(dat8)
plot(dat9)

# a more neutral aesthetic
plot(
  dat3,
  vertex.shape='rectangle',
  vertex.color=NA, # or maybe 'white'
  create.vertex.color=NA,
  drop.vertex.color=NA,
  merge.vertex.color=NA,
  vertex.frame='black',
  edge.color='black',
  vertex.label.color='black',
  merge.edge.color='black',
  drop.edge.color='black'
)

## Not run:
aud <- audit(dat)
write.audit(aud,'dat.audit')
aud <- read.audit('dat.audit')
as.xlsx(dat,'dat.xlsx')

## End(Not run)
```

`as.audited` *Create Audited Objects*

Description

These methods coerce their arguments to class `audited`. An audited object keeps track of changes in row count caused by supported methods. The record of a change is called a transaction, and is stored in an audit table. The table is retrievable using `audit` and can be plotted as a directed graph using `plot.igraph`.

Usage

```
## S3 method for class 'data.frame'
```

```
as.audited(x, key = character(0), id, ...)
## S3 method for class 'keyed'
as.audited(x, key = match.fun('key')(x), id, ...)
## S3 method for class 'audited'
as.audited(x, key = match.fun('key')(x), id, ...)
## S3 method for class 'nm'
as.audited(x, key = match.fun('key')(x), id, ...)
## S3 method for class 'audited'
alias(object, id, ...)
```

Arguments

x	object to be coerced
key	character; see as.keyed
id	optional name for this object for use in the audit table
...	ignored or passed
object	object to be aliased

Details

The argument id will be guessed from context if missing.

as.audited.audited provides a chance to change the key or id of the object. An additional record will be added if counts differ from most recent, even if no other changes are made.

alias.audited updates the id not just on the object, but also in last result identifier in the audit table. It supports the notion that the last transaction resulted in a new entity rather than merely a modification to an existing entity.

as.audited.nm promotes ('nm','keyed','data.frame') to ('nm','audited','keyed','data.frame'). There is no going back: used on ('nm','audited','keyed','data.frame') it gives simply ('audited','keyed','data.frame'). A passed key is respected. id is respected if passed; otherwise id is same as existing or is inferred from the argument.

Value

audited

Author(s)

Tim Bergsma

References

http://metrumrg.googlecode.com

See Also

- as.keyed
- as.keyed.audited
- as.igraph
- audit
- audited-package
- Ops.audited
- write.audit
- subset.audited
- melt.audited
- cast,audited-method
- cast,keyed-method

Examples

```
example(audited)
```

---

as.igraph *Visualize an Audit Transaction Table*

Description

Objects of class audited have embedded transaction tables that can be retrieved as an object of class audit using audit(x). The plot method for class audited extracts the audit table, converts it to igraph using as.igraph, and plots it with the plot method for class igraph. For fine control of plotting characteristics, you can load **igraph** and reproduce the sequence manually.

Usage

```
## S3 method for class 'audited'
plot(x,attrs=auditAttrs(x,...),...)
auditAttrs(x=NULL,...)
## S3 method for class 'audited'
auditAttrs(x,...)
## S3 method for class 'audit'
auditAttrs(
x,
inflation = 1,
proportion = 1,
canvas.nominal = 7,
dimension.nominal = 4,
chars.nominal = 6,
vertex.nominal = 30 * inflation,
vertex.aspect = 1.618,
```

```
canvas = min(dev.size()),
chars = max(chars.nominal,maxchar(x)),
dimension = max(nrow(x),2),
dimension.scale = dimension.nominal / dimension,
deconvolution = deconvolute(canvas,canvas.nominal),
modulation = modulate(dev.name()),
projection = project(canvas.nominal)/project(canvas),
distention = distend(max(dimension,2),  1.5 / dimension.nominal),
canvas.scale = canvas / canvas.nominal,
chars.scale = chars.nominal / chars,
edge.scale = dimension.scale * canvas.scale * projection,
vertex.scale =       dimension.scale * deconvolution
* distention * projection * proportion * modulation,
vertex.label.scale = dimension.scale * canvas.scale
* distention * projection * proportion * chars.scale,
vertex.size = vertex.nominal * vertex.scale ^ scale,
vertex.size2 = vertex.size / vertex.aspect ^ scale,
vertex.label.cex = vertex.label.scale ^ scale,
edge.width =      edge.scale ^ scale,
edge.arrow.size = edge.scale ^ scale,
edge.arrow.width =  1,
scale=TRUE,
rescale=!scale,
...
)
## Default S3 method:
auditAttrs(
x,
color = TRUE,
vertex.shape = 'circle',
vertex.size = 15,
vertex.size2 = vertex.size,
vertex.color =         if (color) 'lightgreen' else NA,
vertex.frame.color =   if (color) NA           else 'black',
vertex.label.color =   if (color) 'darkblue'   else 'black',
vertex.label.cex = 1,
vertex.label.family = 'mono',
edge.color =           if (color) 'darkgreen'  else 'black',
edge.width = 1,
edge.arrow.size = 1,
edge.arrow.width = 1,
edge.label.color = vertex.label.color,
edge.label.cex = vertex.label.cex,
create.vertex.color =  if (color) 'gold'       else NA,
drop.vertex.color =    if (color) 'pink'       else NA,
drop.edge.color =      if (color) 'red'        else 'black',
drop.edge.label =      if (color) NA           else 'drop',
add.vertex.color =     if (color) 'lightblue'  else NA,
```

```
add.edge.color =       if (color) 'blue'       else 'black',
add.edge.label =       if (color) NA           else 'add',
merge.vertex.color =   if (color) 'orchid'     else NA,
merge.edge.color =     if (color) 'orchid4'    else 'black',
merge.edge.label =     if (color) NA           else 'merge',
transform.edge.label = if (color) NA           else 'transform',
modify.edge.label =    if (color) NA           else 'modify',
rescale = FALSE,
...
)
## S3 method for class 'state'
format(x,format='\n%l\n%r',...)
```

Arguments

x	object of method dispatch
inflation	adjust vertex size relative to text size; default 1
proportion	adjust vertex and text size relative to vertex spacing; default 1
format	a format string for text of vertices; default: label over record count
color	logical: whether to use colorful defaults
vertex.shape	vertex shape; circle (default) and rectangle are well-supported
vertex.color	fill color
vertex.frame.color	frame color (default NA gives none)
vertex.label.color	vertex label color
vertex.label.cex	vertex label character expansion
vertex.label.family	vertex label family; default mono
edge.label.color	edge label default color
edge.label.cex	edge label cex
edge.color	default edge color
create.vertex.color	edge color for create events
drop.vertex.color	vertex color for drop events
drop.edge.color	edge color for drop events
drop.edge.label	edge label for drop events
add.vertex.color	vertex color for add events

`add.edge.color` edge color for add events

`add.edge.label` edge label for add events

`merge.vertex.color`
: vertex color for merge events

`merge.edge.color`
: edge color for merge events

`merge.edge.label`
: edge label for merge events

`transform.edge.label`
: edge label for transform events

`modify.edge.label`
: edge label for modify events

`canvas.nominal` reference canvas size

`dimension.nominal`
: reference graph dimension

`chars.nominal` reference label width in monospaced characters

`vertex.nominal` reference vertex size

`vertex.aspect` ratio of vertex size and vertex size2

`canvas` effective canvas size

`chars` effective label width in monospaced characters

`dimension` effective graph dimension

`dimension.scale`
: dimension scale factor

`deconvolution` size-specific adjustment of vertex size (for strict proportionality)

`modulation` device-specific adjustment of vertex size

`distention` dimension-specific adjustment of vertex size

`projection` canvas-specific adjustment of vertex size

`canvas.scale` canvas scale factor

`chars.scale` label width scale factor

`edge.scale` edge scale factor

`vertex.scale` vertex scale factor

`vertex.label.scale`
: vertex label scale factor

`vertex.size` vertex size

`vertex.size2` vertex secondary size

`edge.width` edge width

`edge.arrow.size`
: edge arrow size

`edge.arrow.width`
: edge arrow width

`scale` logical: whether to apply scale factors (default TRUE)

`rescale` logical: whether to use igraph-rescaling (default FALSE)

`attrs` list; see details

`...` passed to called functions

Details

An audited data frame x has a default visualization by means of plot(x). Labels and record counts are plotted in circles with colors that represent transaction types; these are joined by arrows that represent transaction sequence. The option color = FALSE gives a black-and-white aesthetic with transaction types as edge labels.

Default sizes are chosen to maintain a constant aesthetic regardless of transaction count, label length, device, and (particularly) device size. Some effort is made to contain the label text within the vertex, and to make the vertex size about .35 of the inter-vertex distance. These can be adjusted with arguments inflation and proportion.

The format argument controls how labels are displayed. It supports tokens %l, %r, and %a, representing the vertex label (id), the integer number of records, and the integer number of aggregates, respectively. By default, record number is printed under the label, with a blank line above (which centers the label portion in the vertex).

You should not usually need to be concerned with arguments other than inflation, proportion, format, and perhaps shape. Since the plot is implemented with the directed graph from **igraph**, additional arguments can be specified in the form vertex.parameter or edge.parameter ('parameter' not literal). See igraph.plotting for a full list. You can generate transaction-specific parameters by prepending create, add, drop, modify, transform, merge; for example merge.vertex.parameter.

Set any parameter to NULL to defeat the defaults given above. Set attrs to list() to defeat all defaults, giving igraph-native aesthetics.

Value

plot: an invisible list of the effective (supplied) parameters; auditAttrs: a list

Note

The algorithms here try to emulate the aesthetics of a four-record transaction table plotted with 6-character labels on a 7-inch pdf device. You are welcome to experiment with dimension.nominal, chars.nominal, canvas.nominal, and vertex.nominal.

Arrows and vertex frames (when printed) do not scale well for device sizes much less than 7 inches.

You can supply a device-specific adjustment by defining a method modulation.device ('device' not literal). The function rstudiogd() is an experimental convenience that defines modulation.RStudioGD for rapid prototyping that transfers well to pdf.

Dots arguments to plot.audited are passed to as.igraph but not to plot because plot.igraph does not support unrecognized arguments.

For x with class path ('nm','audited','keyed','data.frame'), consider plot(as.audited(x)) which will demote to audited, and so give the audited rather than nm plotting behavior.

Author(s)

Tim Bergsma

References

http://metrumrg.googlecode.com, http://en.wikipedia.org/wiki/Golden_rectangle (vertex.aspect)

See Also

- plot.igraph
- igraph.plotting
- as.keyed
- as.audited
- as.keyed.audited
- audit
- audited-package
- Ops.audited
- write.audit
- subset.audited
- melt.audited

Examples

```
example(audited)
```

---

as.keyed.audited *Coerce Audited to Related Class*

Description

as.keyed.audited demotes audited to just keyed. as.nm.audited executes as.nm.keyed, preserving audited class attributes. digest and as.digest call the keyed method, preserving class attributes. The combinations method demotes to keyed and calls the next method.

Usage

```
## S3 method for class 'audited'
as.keyed(x, key = match.fun('key')(x), ...)
## S3 method for class 'audited'
as.nm(x, key = match.fun('key')(x), id, ...)
## S3 method for class 'audited'
as.digest(x, ...)
## S3 method for class 'audited'
digest(x, ...)
## S3 method for class 'audited'
combinations(x, ...)
```

Arguments

x	object to be coerced
key	optional replacement key, see as.keyed
id	optional replacement id, see as.audited
...	passed to other methods or ignored

Value

as.keyed and combinations: keyed; as.digest (digest): digest; as.nm: nm

Author(s)

Tim Bergsma

References

http://metrumrg.googlecode.com

See Also

- as.keyed
- as.digest
- as.nm
- as.audited
- as.igraph
- audit
- audited-package
- Ops.audited
- write.audit
- subset.audited
- melt.audited

Examples

```
example(audited)
```

---

as.xlsx.artifact *Convert Audit Artifacts to Excel Workbook*

Description

These functions save a collection of artifacts as an Excel 2007 workbook, optionally as a file, using package **xlsx**.

Usage

```
## S3 method for class 'audited'
as.xlsx(x, file = NULL, ...)
## S3 method for class 'artifact'
as.xlsx(x, names, file = NULL, simplify = TRUE, gc = FALSE, ...)
```

Arguments

x	an audited object, or its artifact
names	names for the workbook sheets
file	a filename for the workbook
simplify	whether to drop artifacts with no rows
gc	whether to attempt garbage collection
...	passed to called functions

Details

All the hard work is done by the artifact method, but typically one wants to call as.xlsx for an audited object so that names can be supplied automatically. In that case, they will be the vector arg.id from the audit table. Care is taken to make them unique.

If simplify is true, zero-row artifacts are ignored, and their names dropped if necessary. (Names vector can have the length either of the artifact list, or of the subset having 1 or more rows.)

If file is specified, the workbook is written to file and returned invisibly. The workbook can be assigned in an environment for further manipulation.

Value

a java pointer for a workbook, as for createWorkbook(); invisible if file is specified

Note

For particularly large work sheets, you may need to run, e.g., options(java.parameters = "-Xmx1000m") (see this) or options(java.parameters = "-Xmx4g") (see this) before the first call to as.xlsx. Also, you could pass gc = TRUE in the call, possibly with performance and memory tradeoffs.

Author(s)

Tim Bergsma

References

http://metrumrg.googlecode.com

See Also

- audit
- artifact

Examples

```
example(audited)
```

audit *Retrieve or Assign the Audit and Artifact Attributes*

Description

Use `audit(x)` to retrieve the transaction table for an audited object.

If possible, operations whose names appear in the vector `options('artifact')` store the changed records in the artifact list, which can be retrieved using `artifact(x)`. Length of the artifact list is always equal to number of records in the audit table, and elements correspond to audit records. However, artifacts will have zero rows by default. For example if you specify `options(artifact='drop')` or `options(artifact=TRUE)` then any dropped records are archived; `add` operations etc. generate header only. Other possible character elements are `add`, `create`, `transform`, `merge`, and `modify`. For `merge`, the artifact, if any, is simply `y`.

Normally one should not need to use the assignment forms.

Usage

```
## S3 method for class 'audited'
audit(x, ...)
## S3 method for class 'audited'
artifact(x, ...)
audit(x) <- value
artifact(x) <- value
```

Arguments

`x`	an audited object
`...`	ignored
`value`	an audit object

Value

for assignment: the audited object; for retrieval: the audit or artifact attribute

Author(s)

Tim Bergsma

References

http://metrumrg.googlecode.com

See Also

- as.keyed
- as.audited
- as.keyed.audited
- as.igraph
- audited-package
- Ops.audited
- write.audit
- subset.audited
- melt.audited

Examples

```
example(audited)
```

---

audited-class *Class* "audited"

Description

This class is a keyed data frame with an audit table of transactions and an id attribute. The class is implemented as S3, and is promoted to S4 to take advantage of S4 generic mechanisms.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

.Data: Object of class "list"

names: Object of class "character"

row.names: Object of class "data.frameRowLabels"

.S3Class: Object of class "character"

Extends

Class "keyed", directly. Class "data.frame", by class "keyed", distance 1. Class "list", by class "data.frame", distance 2. Class "oldClass", by class "data.frame", distance 2. Class "vector", by class "data.frame", distance 3.

Author(s)

Tim Bergsma

References

http://metrumrg.googlecode.com

See Also

- as.audited
- keyed-class

Examples

```
showClass("keyed")
```

---

cast-methods *Methods for Function* cast *in Package* **reshape**

Description

Methods for function cast in package **reshape**. **metrumrg** converts reshape::cast to the default method for an S4 generic; it also promotes its own S3 class keyed to an S4 class, and defines a cast method for it. Provided here is the analogous cast method for class audited.

Methods

signature(data = "audited") The method for audited is mostly a wrapper for cast,keyed-method that maintains attributes. It adds a 'transform' transaction to the audit table.

---

melt.audited *Transform Methods for Class Audited*

Description

These methods alter the audited object and return it with an updated transaction table. They are based on the corresponding methods for class keyed, and generate 'transform' transactions (aggregate, melt) or 'merge' transactions (merge).

Usage

```
## S3 method for class 'audited'
aggregate(
x,
by = x[, setdiff(key(x), across),
drop = FALSE],
FUN,
across = character(0),
...
```

```
)
## S3 method for class 'audited'
melt(
data,
id.vars = key(data),
measure.vars,
variable_name = "variable",
na.rm = FALSE,
...
)
## S3 method for class 'audited'
merge(
x,
y,
strict = TRUE,
...
)
```

Arguments

x	audited object
...	passed to other methods
by	passed to aggregate.keyed
FUN	passed to aggregate.keyed
across	passed to aggregate.keyed
data	passed to melt.keyed
id.vars	passed to melt.keyed
measure.vars	passed to melt.keyed
variable_name	passed to melt.keyed
na.rm	passed to melt.keyed
y	right element of a join
strict	prevent dropping records from x

Value

audited

Note

`merge` can drop or add records, or both. It is an error if `strict` is TRUE and any records in x have no representation (match on common columns) in the result. I.e., merge cannot drop records without your expressed permission. Beware that due to the rules for `base::merge`, record counts may not be strictly additive when merging.

Author(s)

Tim Bergsma

References

http://metrumrg.googlecode.com

See Also

- cast,audited-method
- as.keyed
- as.audited
- as.keyed.audited
- as.igraph
- audit
- audited-package
- Ops.audited
- subset.audited
- write.audit

Examples

```
example(audited)
```

Ops.audited *Binary and Unary Operations with Audited*

Description

These are binary and unary operators like those for keyed data frames. + and * generate add transactions; - & and / generate drop transactions. ! inherits the behavior of the subset operator, where i is naKeys() or dupKeys(). ^ and | convert their arguments to class keyed, perform the usual operation, and preserve attributes; they are not expected to add or delete records, so the audit table is not updated. The value of these functions inherits the attributes of e1 by default.

Usage

```
## S3 method for class 'audited'
Ops(e1, e2)
## S3 method for class 'audited'
e1 | e2
## S3 method for class 'audited'
e1 ^ e2
## S3 method for class 'audited'
e1
```

Arguments

e1	left argument to binary operator, or only unary argument
e2	right argument to binary operator

Value

audited

Author(s)

Tim Bergsma

References

http://metrumrg.googlecode.com

See Also

- Ops.keyed
- as.keyed
- as.audited
- as.keyed.audited
- as.igraph
- audit
- audited-package
- write.audit
- subset.audited
- melt.audited

Examples

```
example(audited)
```

---

subset.audited *Drop Methods for Class Audited*

Description

These methods alter the audited object and return it with an updated transaction table. They generate drop transactions.

Usage

```
## S3 method for class 'audited'
x[i, j, drop, id, od]
## S3 method for class 'audited'
subset(x, subset, select, drop = FALSE, id, od = "! subset", ...)
## S3 method for class 'audited'
unique(x, incomparables = FALSE, fromLast = FALSE, id, od = "! unique", ...)
## S3 method for class 'audited'
head(x, n = 6L, ..., id, od = "! head")
## S3 method for class 'audited'
tail(x, n = 6L, ..., id, od = "! tail")
## S3 method for class 'audited'
rbind(..., deparse.level = 1)
```

Arguments

id	character (scalar); see details
od	character (scalar); see details
x	audited
i	passed to subset operator
j	passed to subset operator
drop	passed to other functions
subset	passed to subset.data.frame
select	passed to subset.data.frame
...	passed to other functions
incomparables	passed to unique
fromLast	passed to unique
n	passed to head or tail
deparse.level	as for rbind.data.frame

Details

The most important arguments here are id and od. All the others are passed through to related functions.

id and od will always have informative defaults. However, you will often want to supply customized values. id is a label for the set of rows that is returned. od is a label for the set of rows that is dropped; it will be used for plotting.

When supplying id or od, remember to maintain the proper number of dimensions. For example,

`Theoph[ Theoph$WT > 70, , id = 'heavier', od = 'lighter' ]` not

`Theoph[ Theoph$WT > 70,  id = 'heavier', od = 'lighter' ]`

The subset operator, called by all the other functions listed above, deserves special consideration. Argument i controls the rows that are returned. For a data.frame, i may be negative, positive, zero, logical, character, and NA, with varying effects. In particular, use of repeated positive

creates duplicates of rows, while use of NA, numeric greater than `nrow(x)`, or character not in `row.names(x)` creates NA rows. `[.audited` rejects indices that create NA rows. It allows duplication of rows, but it is an error to both add and drop rows simultaneously. For example, `x[c(2,2),]` should fail, because record 1 (probably others) is being dropped while a copy of record 2 is being added.

Value

audited

Author(s)

Tim Bergsma

References

http://metrumrg.googlecode.com

See Also

- as.keyed
- as.audited
- as.keyed.audited
- as.igraph
- audit
- artifact
- audited-package
- Ops.audited
- melt.audited
- write.audit

Examples

```
example(audited)
```

`write.audit` *Write and Read Audits*

Description

Functions to write and read objects of class `audit`.

Usage

```
write.audit(x, file, quote = FALSE, row.names = FALSE, na = ".", ...)
read.audit(x, ...)
```

Arguments

x	an audit object, or a filename for read
file	passed to write.table
quote	passed to write.table
row.names	passed to write.table
na	passed to write.table
...	passed to write.table

Value

for read, an object of class audit

Author(s)

Tim Bergsma

References

http://metrumrg.googlecode.com

See Also

- as.keyed
- as.audited
- as.keyed.audited
- as.igraph
- audit
- audited-package
- Ops.audited
- subset.audited
- melt.audited
- write.table

Examples

```
example(audited)
```

Index