Package 'AMModels'

October 30, 2018

Type Package

Title Adaptive Management Model Manager

Version 0.1.4

Date 2018-10-26

Author Jon Katz <jonkatz4@gmail.com>, Therese Donovan <tdonovan@uvm.edu>

Maintainer Jon Katz <jonkatz4@gmail.com>

Description Helps enable adaptive management by codifying knowledge in the form of models generated from numerous analyses and data sets. Facilitates this process by storing all models and data sets in a single object that can be updated and saved, thus tracking changes in knowledge through time. A shiny application called AM Model Manager (modelMgr()) enables the use of these functions via a GUI.

License GPL-3

Copyright This software is in the public domain because it contains materials that originally came from the United States Geological Survey, an agency of the United States Department of Interior. For more information, see the official USGS copyright policy at http://www.usgs.gov/visual-id/credit_usgs.html#copyright. This software is in the public domain because it contains materials that originally came from the U.S. Geological Survey, an agency of the United States Department of Interior. Although this software program has been used by the U.S. Geological Survey (USGS), no warranty, expressed or implied, is made by the USGS or the U.S. Government as to the accuracy and functioning of the program and related program material nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the USGS in connection therewith. This software is provided ``AS IS."

Encoding UTF-8

VignetteBuilder R.rsp

Imports methods, unmarked

Suggests R.rsp, shiny, shinyBS, knitr, AICcmodavg

RoxygenNote 6.0.1

Collate 'AMModels-package.R' 'amData.R' 'amModel.R' 'amModelLib.R' 'catwrap.R' 'classes.R' 'getters.R' 'grepAMModelLib.R' 'insertAMModelLib.R' 'ls.R' 'metaSummary.R' 'methods-amModel.R' 'methods-amModelLib.R' 'modelMgr.R' 'rmModelsData.R' 'simCovariate.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2018-10-30 17:30:02 UTC

R topics documented:

`amData` *Create An* `amData` *Object That Pairs Datasets With Associated Metadata*

Description

Creates an object of class `amData`, which is typically a data frame of covariate data or model fitting data, with mandatory metadata. It is worth noting that some models include the data as part of the fitted model object, e.g. `lm` and `glm`, which access embedded data with `model.frame`. For large datasets that are embedded in the model, it may be worth documenting metadata a placeholder object such as the character string `"embedded data"` rather than a redundant data object.

An S4 class to store data with descriptive metadata.

Usage

```
amData(data, ...)
```

Arguments

data	A dataset, typically data frame but may be in any structure.
...	Named metadata elements, either supplied individually and coerced to list or supplied as a named list.

Value

An object of class amData suitable for inclusion in an amModelLib object.

Slots

data A single object containing data relevant to a model. There are no restrictions on how the data are used in the model; for example, they may be covariate data (for use on the right side of the equation) or observed data (for use on the left side of the equation).

metadata A named list of length 1 character vectors that form name:value pairs, e.g the source, the collection method, etc. When embedded in an amModelLib, data metadata may be retrieved or set with dataMeta.

See Also

Other amModelLib: AMModels, amModelLib, amModel, getters, grepAMModelLib, insertAMModelLib, lsModels, rmModel

Examples

```
# create dataset from lm helpfile
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)

# create a dataset that is of class data.frame
plant.data <- data.frame(weight, group)

# create an amData data object
dat1 <- amData(data = plant.data, comment='Dataset from lm helpfile.', taxa = 'plants')

# the class of dat1 is amData
class(dat1)

# the summary function will invoke the summary method for the dataset's original class
summary(dat1)

# use the amModelLib function to create a new amModelLib called mymodels that
# includes dat1; data must be supplied in a named list
mymodels <- amModelLib(
```

```
    data=list(dat1 = dat1),
    description = "An example amModelLib called mymodels."
)

# use the lsData function to list the amData objects in an amModelLib
lsData(mymodels)

# the dataMeta function can be used to retrieve an amData object's metadata
dataMeta(amml = mymodels, 'dat1')

#  the dataMeta function can alse be used to set metadata
dataMeta(mymodels, 'dat1') <- list(
    url = "https://stat.ethz.ch/R-manual/R-devel/library/stats/html/lm.html"
)
dataMeta(amml = mymodels, 'dat1')

# use the getAMData function to extract the dataset back to its orginal form
getAMData(amml = mymodels, 'dat1', as.list = FALSE)

# the retrieved datset is in its original class
class(getAMData(amml = mymodels, 'dat1', as.list = FALSE))

# use the amModelLib function to create an empty amModelLib
mymodels2 <- amModelLib(description = "An example amModelLib called mymodels2.")

# use the insertAMModelLib function to insert the amData object to an
# existing amModelLib
mymodels2 <- insertAMModelLib(data = list(dat1 = dat1))

# use rmData to remove an amData object from an amModelLib
rmData('dat1', amml = mymodels2)
```

`amModel` *Create An* `amModel` *Object That Pairs Models With Their Metadata*

Description

Creates an object of class `amModel`, which is a fitted model object (or similar) with mandatory metadata.

An S4 class to store models with descriptive metadata.

Usage

```
amModel(model, ...)
```

Arguments

`model`	A model fit object or similar.
`...`	Named metadata elements, either supplied individually and coerced to list or supplied as a named list.

Value

An object of class amModel suitable for inclusion in an amModelLib.

Slots

model A model fit object, e.g. from lm or any single object containing the means to predict values from data.

metadata A named list of length 1 character vectors that form name:value pairs, e.g. the analyst, the project, the data used, etc. The metadata name data can be used to link the model with the name of a data object, and when manipulated in the amModelLib an effort will be made to keep the model and data together. When embedded in an amModelLib, model metadata may be retrieved or set with modelMeta.

See Also

Other amModelLib: AMModels, amData, amModelLib, getters, grepAMModelLib, insertAMModelLib, lsModels, rmModel

Examples

```
# run models from from lm helpfile
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept

# note the original class of models is 'lm'
class(lm.D9)

# wrap lm.D9 in an amModel object
mod1 <- amModel(model = lm.D9, comment='example from lm')

# summarize mod1; summary for the original object class, plus the metadata
summary(mod1)

# convert lm.D90 to an amModel object
mod2 <- amModel(model = lm.D90, comment='second example from lm')

# use the function, amModelLib, to create a new amModelLib that includes both models
# the models must be supplied as a named list
mymodels <- amModelLib(
    models = list(
        mod1 = mod1,
        mod2 = mod2
    ),
```

```
    description = "This amModelLib stores linear models from the lm helpfile."
)

# list the models of the amModelLib
lsModels(mymodels)

# the modelMeta function can be used to retrieve model metadata
modelMeta(mymodels, 'mod1')

# the modelMeta function can also be used to set model metadata
modelMeta(mymodels, 'mod1') <- list(
    url = "https://stat.ethz.ch/R-manual/R-devel/library/stats/html/lm.html"
)
modelMeta(amml = mymodels, 'mod1')

# use getAMModel to extract the model by  name
getAMModel(amml = mymodels, 'mod2', as.list = FALSE)

# notice the retrieved model is returned in its original class
class(getAMModel(amml = mymodels, 'mod2', as.list = FALSE))

# create a new empty amModelLib called mymodels2
mymodels2 <- amModelLib(description = "A second amModelLib called mymodels2.")

# use insertAMModelLib to insert amModel objects to an existing amModelLib
mymodels2 <- insertAMModelLib(models = list(mod1 = mod1, mod2 = mod2))

# use rmModel to remove an amModel object from an amModelLib
rmModel('mod2', amml = mymodels2)
```

`amModelLib` *Create An* `AMModelLib` *Object That Stores Lists Of* `amModel` *And* `amData` *Objects*

Description

Creates an object of class `amModelLib`, either empty, or containing `amModel` and/or `amData` objects.

An S4 class to hold model and data, each with descriptive metadata.

Usage

```
amModelLib(models = list(), data = list(), info = list(),
  description = "")
```

Arguments

`models`	A list of objects of class amModel.
`data`	A list of objects of class amData.
`info`	Named list with descriptive metadata about the `amModelLib` object.
`description`	Text field describing the `amModelLib` object.

Details

amModelLib objects are useful for storing multiple models and datasets of a related theme, along with relevant metadata. Most extraction and manipulation functions will attempt to keep any relevant data with any model for which it is used. Multiple models may refer to a single dataset.

Value

An object of class amModelLib.

Slots

models A named list of amModel objects. Models are added with insertAMModelLib and removed with rmModel.

data A named list of amData objects. Data are added with insertAMModelLib and removed with rmData.

info A named list of length 1 character strings that forms name:value metadata pairs describing the amModelLib, e.g. the project, the owner, etc. Name:value pairs may be retrieved or set using ammlInfo.

description Length 1 character vector describing the amModelLib, intended to offer a summary of the amModelLib and quickly refresh the user to its contents and purpose without poring over the detailed metadata. The description may be retrieved or set with ammlDesc.

See Also

Other amModelLib: AMModels, amData, amModel, getters, grepAMModelLib, insertAMModelLib, lsModels, rmModel

Examples

```
# code from the lm helpfile
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept

# create a data.frame of the plant data
plant.data <- data.frame(group = group, weight = weight)

# create an amData object that includes the data.frame and metadata
plant.data <- amData(
    data = plant.data,
    comment = 'Dataset from lm helpfile.'
)
```

```
# create an amModel model object that includes model lm.D9 and metadata
# use the metadata keyword 'data' to link the model with the amData object
# that produced it
plant.model1 <- amModel(
    model = lm.D9,
    comment = 'Example model produced from from lm helpfile.',
    data = 'plant.data'
)

# create a second amModel model object that includes model lm.D90 and metadata
# use the metadata keyword 'data' to soft link the model with its data
plant.model2 <- amModel(
    model = lm.D90,
    comment = 'Second model produced from from lm helpfile.',
    data = 'plant.data'
)


# use the amModelLib function to create a new amModelLib containing the two
# amModel objects and one amData object
mymodels <- amModelLib(
     models = list(
        plant.model1 = plant.model1,
        plant.model2 = plant.model2
    ),
    data = list(
        plant.data = plant.data
    ),
    description = "This amModelLib stores models and data from the lm helpfile.",
    info = list(
        owner = "Me"
    )
)

# use the amModelLib function amModelLib to create an empty amModelLib called mymodels2
mymodels2 <- amModelLib(
     description = "A second amModelLib called mymodels2.",
     info = list(
         owner = "Me2"
     )
)

# use the insertAMModelLib function to insert the two amModel objects and one
# amData oject to the existing amModelLib
mymodels2 <- insertAMModelLib(
     models = list(
         plant.model1 = plant.model1,
         plant.model2 = plant.model2),
     data = list(plant.data = plant.data)
)
```

AMModels *Search for a model in a model list using* grep

Description

Enables adaptive management by codifying knowledge in the form of models generated from numerous analyses and datasets. AMModels facilitates this process by storing all models and datasets in a single object that can be seemlessly updated, tracking changes in knowledge through time. A shiny application called AM Model Manager enables the use of these functions via a GUI. To launch the Model Manager, use `modelMgr()`.

Copyright

This software is in the public domain because it contains materials that originally came from the United States Geological Survey, an agency of the United States Department of Interior. For more information, see the official USGS copyright policy at http://www.usgs.gov/visual-id/credit_usgs.html#copyright

Disclaimer

This software is in the public domain because it contains materials that originally came from the U.S. Geological Survey, an agency of the United States Department of Interior. For more information, see the official USGS copyright policy at http://www.usgs.gov/visual-id/credit_usgs.html#copyright. Although this software program has been used by the U.S. Geological Survey (USGS), no warranty, expressed or implied, is made by the USGS or the U.S. Government as to the accuracy and functioning of the program and related program material nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the USGS in connection therewith. This software is provided "AS IS."

See Also

Other amModelLib: `amData`, `amModelLib`, `amModel`, `getters`, `grepAMModelLib`, `insertAMModelLib`, `lsModels`, `rmModel`

Examples

```
# create data and linear models from from lm helpfile
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept

# create an amData object that includes metadata
```

```
plant.data <- data.frame(group = group, weight = weight)
plant.data <- amData(
    data = plant.data,
    comment = 'Dataset from lm helpfile.'
)

# create a second amData object that includes metadata
log.plant.data <- data.frame(group, log.weight = log(weight))
log.plant.data <- amData(
    data = log.plant.data,
    comment = 'data with log weight',
    source = 'lm helpfile (R).'
)

# create two amModel objects with metadata and the metadata keyword 'data' to soft
# link the data used to fit the models
full.model <- amModel(
    model = lm.D9,
    comment = 'full model',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)

no.int.model <- amModel(
    model = lm.D90,
    comment = 'model without intercept',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)

# create an amModelLib that contains the two amModel objects and two amData objects
# the models and data must be supplied as named lists
mymodels <- amModelLib(
    models = list(
        full.model = full.model,
        no.int.model = no.int.model
    ),
    data = list(
        plant.data = plant.data,
        log.plant.data = log.plant.data
    )
)

# show the amModelLib
mymodels

# search the entire amModelLib for the word 'intercept'
# the dataset associated with the model will be returned
grepAMModelLib(pattern = "intercept", amml = mymodels)

# search for data containing the word 'log'
```

```
grepAMModelLib(pattern = "log", amml = mymodels, search = "data")

# search for models containing the word 'full';
# because 'full.model' is soft-linked to a dataset,
# the dataset information will be returned.
grepAMModelLib(pattern = "full", amml = mymodels, search = "model")


# list names of models in an amModelLib
lsModels(mymodels)

# list names of data in an amModelLib
lsData(mymodels)

# extract the dataset by name
getAMData(amml = mymodels, 'plant.data', as.list = FALSE)

# notice the data are returned in their original class
class(getAMData(amml = mymodels, 'plant.data', as.list = FALSE))

# you can also extract by index
getAMData(amml = mymodels, 1, as.list = FALSE)

# extract the model by name
getAMModel(amml = mymodels, 'full.model', as.list = FALSE)

# notice the models are returned in their original class
class(getAMModel(amml = mymodels, 'full.model', as.list = FALSE))

# you can also extract by index
getAMModel(amml = mymodels, 1, as.list = FALSE)

# remove just the second model
rmModel(mymodels, 'no.int.model')

# remove the first plant data
# notice a warning is produced because some amModels are softly linked
# to the dataset via the metadata keyword, 'data'
rmData(mymodels, 'plant.data')

## Not run:
# The shiny app
modelMgr()

## End(Not run)
```

---

getters *Extract A Single Model or Data Object, Get Or Set Info, Description, Or Metadata.*

Description

The function getAMData will extract an amData object from an amModelLib; the function getAMModel will extract an amModel object from an amModelLib. The function ammlDesc can be used to retrieve or set a description of an amModelLib object. The function ammlInfo can be used to retrieve or set information about an amModelLib object. modelMeta and dataMeta retrieve and set metadata within amModelLib objects.

Usage

```
getAMModel(amml, x, as.list = FALSE, ...)

getAMData(amml, x, as.list = FALSE, ...)

ammlDesc(amml)

ammlDesc(amml) <- value

ammlInfo(amml, x = NULL)

ammlInfo(amml) <- value

modelMeta(amml, x = NULL)

modelMeta(amml, x) <- value

dataMeta(amml, x = NULL)

dataMeta(amml, x) <- value
```

Arguments

amml	An amModelLib object.
x	A name or length 1 integer index to extract from or set within the amModelLib object.
as.list	Logical; FALSE to return just the object, TRUE to return a list with both object and metadata.
...	Additional arguments (not used).
value	A named list of metadata to set within x.

Details

The objects created by getAMData and getAMModel are returned as their original class unless the argument as.list is set to TRUE. If as.list, a list is returned with the original object in the first element and metadata in the second.
The setters for ammlInfo, modelMeta, and dataMeta replace individual elements in their respective lists with each call. To remove elements set their value to NULL in the named replacement list.

Value

The model or data object in its original form, or if `as.list` a list with

`models` The original model

(or

`data` The original data

) and

`metadata` metadata

.

See Also

Other amModelLib: `AMModels`, `amData`, `amModelLib`, `amModel`, `grepAMModelLib`, `insertAMModelLib`, `lsModels`, `rmModel`

Examples

```
# create dataset from lm helpfile
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
# notice the models lm.D9 and lm.D90 are of class 'lm'
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept


# create two amModel objects with metadata and a soft link to the data
full.model <- amModel(
    model = lm.D9,
    comment = 'full model',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)

no.int.model <- amModel(
    model = lm.D90,
    comment = 'model without intercept',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)

# create an amData object that includes metadata
```

```
# the plant.data is of class data.frame
plant.data <- data.frame(group = group, weight = weight)
plant.data <- amData(
    data = plant.data,
    comment = 'Dataset from lm helpfile.'
)

# create a second amData object that includes metadata
log.plant.data <- data.frame(group, log.weight=log(weight))
log.plant.data <- amData(
    data = log.plant.data,
    comment = 'data to fit log model',
    source = 'lm helpfile (R).'
)

# create an amModelLib that contains the two amModel objects and two amData objects
# the models and data must be supplied as named lists
mymodels <- amModelLib(
    models = list(
        full.model = full.model,
        no.int.model = no.int.model
    ),
    data = list(
        plant.data = plant.data,
        log.plant.data = log.plant.data
    )
)

# extract the dataset
getAMData(amml = mymodels, 'plant.data', as.list = FALSE)

# you can also extract by index
getAMData(amml = mymodels, 1, as.list = FALSE)

# extract the model
getAMModel(amml = mymodels, 'full.model', as.list = FALSE)

# you can also extract by index
getAMModel(amml = mymodels, 1, as.list = FALSE)

# extraction with '[' and '[[', which are identical here, focus on models
mymodels[c(1,2)]
mymodels[[1]]

# Add a description to the amModelLib
ammlDesc(mymodels) <- "This library demonstrates how to store models
and data in a format that allows for descriptive metadata and easy
retrieval for future reference."

# Extract the description
ammlDesc(mymodels)

# Add some metadata 'info' to the amModelLib
```

```
ammlInfo(mymodels) <- list(owner = 'me', organization = 'My Organization')

# Extract all info for an amModelLib
ammlInfo(mymodels)

# Extract targeted info
ammlInfo(amml = mymodels, 'owner')

# Delete metadata by setting to NULL
ammlInfo(mymodels) <- list(organization = NULL)

# Extract all model metadata
modelMeta(mymodels)

# Extract metadata from specific model
modelMeta(amml = mymodels, 'full.model')

# Add metadata to 'full.model'
modelMeta(amml = mymodels, 'full.model') <- list(
    url = "https://stat.ethz.ch/R-manual/R-devel/library/stats/html/lm.html"
)

# remove metadata by setting value to NULL
modelMeta(amml = mymodels, 'full.model') <- list(url = NULL)

# Extract all data metadata
dataMeta(mymodels)

# Extract metadata from specific data
dataMeta(amml = mymodels, 'plant.data')

# Add metadata to 'plant.data'
dataMeta(mymodels, 'plant.data') <- list(
    url = "https://stat.ethz.ch/R-manual/R-devel/library/stats/html/lm.html"
)

# remove metadata by setting value to NULL
dataMeta(mymodels, 'plant.data') <- list(url = NULL)
```

`grepAMModelLib` *Search For A Model In A Model List Using* `grep`

Description

Returns an abbreviated amModelLib object that contains models and data that meet search terms.

Usage

```
grepAMModelLib(pattern, amml, search = c("all", "model", "data"), ...)
```

Arguments

pattern	Search string or value, typically a model or data name
amml	An amModelLib object
search	Length 1 character vector indicating whether to search and return models or data that meet the search criteria.
...	Additional arguments to grep.

Details

grep is used to search both names, values (models/data), and metadata. An attempt is made to keep data with models if searching for models, or to keep models with data if searching for data, or to keep prior/posterior models together. The relational link between models their data relies on a case-agnostic 'data' element in the model metadata that names the linked data, and the same is true for models that use 'prior' to link to other models.

Value

An object of class amModelLib.

See Also

Other amModelLib: AMModels, amData, amModelLib, amModel, getters, insertAMModelLib, lsModels, rmModel

Examples

```
# create dataset from lm helpfile
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept


# create two amModel objects with metadata and a soft link to the data
full.model <- amModel(
    lm.D9,
    comment = 'full model',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)

no.int.model <- amModel(
    lm.D90,
    comment = 'model without intercept',
```

```
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)


# create an amData object that includes metadata
plant.data <- data.frame(group = group, weight = weight)
plant.data <- amData(
    plant.data,
    comment = 'Dataset from lm helpfile.'
)

log.plant.data <- data.frame(group, log.weight=log(weight))
log.plant.data <- amData(
    log.plant.data,
    comment = 'data to fit log model',
    source = 'lm helpfile (R).'
)

# create an amModelLib that contains the two amModel objects and two amData objects
# the models and data must be supplied as named lists
mymodels <- amModelLib(
    models = list(
        full.model = full.model,
        no.int.model = no.int.model
    ),
    data=list(
        plant.data = plant.data,
        log.plant.data = log.plant.data
    )
)


# search the entire amModelLib for the word 'intercept'
# the dataset associated with the model will be returned
grepAMModelLib("intercept", amml = mymodels)

# the class of returned search is an amModelLib object
class(grepAMModelLib("intercept", amml = mymodels))

# search for data containing the word 'log'
grepAMModelLib("log", amml = mymodels, search = "data")

# search for models containing the word 'full'
# Because 'full.model' is soft-linked to a dataset,
# the dataset information will be returned.
grepAMModelLib("full", amml = mymodels, search = "model")
```

`insertAMModelLib` *Insert Model Of Class* `amModel` *Or Dataset Of Class* `amData` *Into An* `amModelLib` *Object*

Description

Inserts a model into the model slot of an `amModelLib` object, or inserts a dataset into the data slot of an `amModelLib` object. If the `amModelLib` is not specified, the function will create an info-less and description-less lib – or specify these components in the ... argument.

Usage

```
insertAMModelLib(models, data, amml, ...)
```

Arguments

`models`	A named list of `amModel` objects, each composed of a fitted model object and metadata.
`data`	An object of class `amData`.
`amml`	The name of the object that of class `amModelLib`.
`...`	Additional arguments to be passed to the function `amModelLib` to be included in the `amModelLib` if one is created.

Details

If the argument amml is NULL, the function will call `amModelLib` to create the object. The argument info can be passed to this function via the ... argument.

Value

An object of class `amModelLib`

See Also

Other amModelLib: `AMModels`, `amData`, `amModelLib`, `amModel`, `getters`, `grepAMModelLib`, `lsModels`, `rmModel`

Examples

```
# create dataset from lm helpfile
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
```

```
lm.D90 <- lm(weight ~ group - 1) # omitting intercept

# create an amData object that includes metadata
plant.data <- data.frame(group = group, weight = weight)
plant.data <- amData(data = plant.data, comment = 'Dataset from lm helpfile.')

# create an amModel object with metadata and a soft link to the data
full.model <- amModel(
    model = lm.D9,
    comment = 'full model',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)


# create an amModelLib that contains the amModel object and the amData object
# the model and data must be supplied as named lists
mymodels <- amModelLib(
    description = "An example amModelLib.",
    models = list(full.model = full.model),
    data = list(plant.data = plant.data)
)

# create second amModel object with metadata and a soft link to the same data
no.int.model <- amModel(
    model = lm.D90,
    comment = 'model without intercept',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)

# create a second amData object
log.plant.data <- data.frame(group, log.weight=log(weight))
log.plant.data <- amData(
    data = log.plant.data,
    comment = 'data to fit log model',
    source = 'lm helpfile (R).'
)

# insert the second model and second dataset to the amModelLib
mymodels <- insertAMModelLib(
    mymodels,
    models = list(no.int.model = no.int.model),
    data = list(log.plant.data = log.plant.data)
)
```

---

lsModels *List Names Of* amModel *Or* amData *Objects In An* amModelLib *Object.*

Description

List names of all objects of class amModel (a fitted model with mandatory metadata) or amData (a dataset with mandatory metadata) in an amModelLib object.

Usage

```
lsModels(x)

lsData(x)
```

Arguments

x	An object of class amModelLib.

Value

A vector of names.

See Also

Other amModelLib: AMModels, amData, amModelLib, amModel, getters, grepAMModelLib, insertAMModelLib, rmModel

Examples

```
# create dataset from lm helpfile
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept

# create an amData object that includes metadata
plant.data <- data.frame(group = group, weight = weight)
plant.data <- amData(
    data = plant.data,
    comment = 'Dataset from lm helpfile.'
)

# create a second amData object with log data
log.plant.data <- data.frame(group, log.weight=log(weight))
log.plant.data <- amData(
    data = log.plant.data,
    comment = 'Dataset that includes the log of plant weight',
    source = 'lm helpfile (R).'
)
```

```
# create two amModel objects with metadata and a soft link to the data
full.model <- amModel(
    model = lm.D9,
    comment = 'full model',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)

no.int.model <- amModel(
    model = lm.D90,
    comment = 'model without intercept',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)



# create an amModelLib that contains the two amModel objects and two amData objects
# the models and data must be supplied as named lists
mymodels <- amModelLib(
    models = list(
        full.model = full.model,
        no.int.model = no.int.model
    ),
    data = list(
        plant.data = plant.data,
        log.plant.data = log.plant.data
    )
)


# list names of models
lsModels(mymodels)

# list names of data
lsData(mymodels)
```

`methods-amModel` *Methods For Displaying, Summarizing, And Manipulating* `amModel` *And* `amData` *Objects*

Description

Getters and setters for models and data.

Usage

```
## S4 method for signature 'amModel'
summary(object, ...)

## S4 method for signature 'amModel,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'amModel,ANY,ANY'
x[[i]]

## S4 replacement method for signature 'amModel,ANY,ANY'
x[i, j, ...] <- value

## S4 replacement method for signature 'amModel,ANY,ANY'
x[[i]] <- value

## S4 method for signature 'amData'
summary(object, ...)

## S4 method for signature 'amData,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'amData,ANY,ANY'
x[[i]]

## S4 replacement method for signature 'amData,ANY,ANY'
x[i, j, ...] <- value

## S4 replacement method for signature 'amData,ANY,ANY'
x[[i]] <- value
```

Arguments

object, x	An amModel or amData object.
...	Additional arguments passed to other functions or methods.
i, j	indices specifying elements to extract or replace. Indices are numeric or character vectors or empty (missing) or NULL.
drop	Not used.
value	Replacement value.

Details

Summary assumes some meaningful summary method exists for each object in its home package.

Examples

```
# create dataset from lm helpfile
```

```
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept


# create am amModel object
full.model <- amModel(
    model = lm.D9,
    comment = 'full model',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)


# create an amData object
plant.data <- data.frame(group = group, weight = weight)
plant.data <- amData(
    data = plant.data,
    source = 'lm helpfile (R).',
    comment = 'Dataset from lm helpfile.'
)

summary(full.model)

# [ and [[ index from metadata
full.model[c(2,1)]
full.model[[1]]
full.model[['taxa']]

plant.data[c(2,1)]
plant.data[[1]]
plant.data[['comment']]
```

`methods-amModelLib` *Methods For Displaying, Summarizing, And Manipulating* `amModelLib` *Objects*

Description

Getters and setters for AMModelLib objects.

Usage

```
## S4 method for signature 'amModelLib'
summary(object, name, ...)

## S4 method for signature 'amModelLib'
show(object)

## S4 method for signature 'amModelLib'
c(x, ..., recursive = FALSE)

## S4 method for signature 'amModelLib,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'amModelLib,ANY,ANY'
x[[i]]

## S4 method for signature 'amModelLib'
x$name

## S4 replacement method for signature 'amModelLib,ANY,ANY'
x[[i, j, ...]] <- value
```

Arguments

object	An amModelLib object.
name	For $ A literal character string or a name (possibly backtick quoted); for summary an amModel or amData name as character string.
...	Additional arguments passed to other functions or methods.
x	An amModelLib object.
recursive	Iterate recursively through lists (ignored).
i, j	indices specifying elements to extract or replace. Indices are numeric or character vectors or empty (missing) or NULL.
drop	Not used.
value	Replacement value.

Details

Summary adds the metadata to the default show method. If name is supplied the call is passed on to the amModel or amData object with the specified name.

Value

summary returns a list with the same elements displayed during the call. Others return an amModelLib object.

Examples

```
# create dataset from lm helpfile
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept

#' # create an amData object that includes metadata
plant.data <- data.frame(group = group, weight = weight)
plant.data <- amData(
    data = plant.data,
    comment = 'Dataset from lm helpfile.'
)

log.plant.data <- data.frame(group, log.weight=log(weight))
log.plant.data <- amData(
    data = log.plant.data,
    comment = 'data to fit log model',
    source = 'lm helpfile (R).'
)

# create two amModel objects with metadata and a soft link to the data
full.model <- amModel(
    model = lm.D9,
    comment = 'full model',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)

no.int.model <- amModel(
    model = lm.D90,
    comment = 'model without intercept',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)



# create an amModelLib that contains the two amModel objects and two amData objects
# the models and data must be supplied as named lists
mymodels <- amModelLib(
    models = list(
        full.model = full.model,
        no.int.model = no.int.model
    ),
```

```
    data=list(
        plant.data = plant.data,
        log.plant.data = log.plant.data
    )
)

summary(mymodels)
mymodels <- c(mymodels, mymodels)
mymodels[c(2,1)]
mymodels[[1]]
mymodels[['full.model']]
mymodels$full.model
```

`modelMgr` *Graphical UI For The AMModels Package*

Description

The model manager is a graphical user interface supplied to accomplish nearly all functions within the AMModels package. Models and data can be tagged with metadata and organized within model libraries. Model libraries can be imported, searched, subset, edited, and exported through the GUI. The modelMgr GUI allows users to import models and data from either a .RData or .rda file or from the user's `.GlobalEnv`.

Usage

```
modelMgr(...)
```

Arguments

`...`	Additional arguments to `shiny::runApp`.

Details

modelMgr

Author(s)

Jon Katz

Examples

```
## Not run:
# The shiny app
modelMgr()

# Accepts args to shiny::runApp
modelMgr(quiet = TRUE)

## End(Not run)
```

mymodels *Sample* amModelLib *Object Containing Models and Data.*

Description

Four models and four data objects.

Format

An amModelLib object with

Examples

```
data(mymodels)
mymodels
summary(mymodels)
getAMModel(mymodels, 'plant.model')
mymodels[[1]]
str(mymodels)
```

rmModel *Remove An* amModel *Or* amData *Object From An* amModelLib *Object*

Description

Remove an object of class amModel or amData (a fitted model object or data to fit a model or use as covariate data, with mandatory metadata) from an amModelLib object.

Usage

```
rmModel(amml, x)

rmData(amml, x)
```

Arguments

amml	An amModelLib object.
x	A character vector, numeric vector, or logical vector identifying model(s) or data to remove.

Value

An object of class amModelLib.

See Also

Other amModelLib: AMModels, amData, amModelLib, amModel, getters, grepAMModelLib, insertAMModelLib, lsModels

Examples

```
# create dataset from lm helpfile
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept

# create an amData object that includes metadata
plant.data <- data.frame(group = group, weight = weight)
plant.data <- amData(
    data = plant.data,
    comment = 'Dataset from lm helpfile.'
)

log.plant.data <- data.frame(group, log.weight=log(weight))
log.plant.data <- amData(
    data = log.plant.data,
    comment = 'data to fit log model',
    source = 'lm helpfile (R).'
)

# create two amModel objects with metadata and a soft link to the data
full.model <- amModel(
    model = lm.D9,
    comment = 'full model',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)

no.int.model <- amModel(
    model = lm.D90,
    comment = 'model without intercept',
    source = 'lm helpfile (R).',
    taxa = 'plants',
    data = 'plant.data'
)

# create an amModelLib that contains the two amModel objects and two amData objects
# the models and data must be supplied as named lists
mymodels <- amModelLib(
    models = list(
```

```
        full.model = full.model,
        no.int.model = no.int.model
    ),
    data=list(
        plant.data = plant.data,
        log.plant.data = log.plant.data
    )
)

# show the library
mymodels


# remove just the second model
rmModel(mymodels, 'no.int.model')

# remove the first plant data, has a soft-link from a model, throws warning.
rmData(mymodels, 'plant.data')

# show the library
mymodels
```

`simCovariate` *Simulate A Dataframe Of Uncorrelated Covariate(s)*

Description

Quickly create a dataframe of uncorrelated random variables which can be used as covariates. Values are drawn from the normal, uniform, beta, binomial, poisson or bernoulli distributions.

Usage

```
simCovariate(cov.list = NULL, ..., n, add.yr = TRUE)
```

Arguments

`cov.list`	A named list of covariates to be simulated and their required arguments.
`...`	additional arguments to be passed to `distr`. Argument in which the simulating distribution its corresponding arguments are specified.Minimally contains `n`, for number of samples to draw, `shape1, shape2` for `rbeta`, and `size, prob` for `rbinom`. May also contain optional arguments such as `min, max` for `runif`, `mean, sd` for `rnorm`, and `ncp` for `rbeta`. Accepts single values, or vectors that will be applied to multiple columns. Vectors should be used with care as lengths are not checked.
`n`	The number of samples to generate from each covariate.
`add.yr`	Logical, if `TRUE` a field named `yr` is added with indices from `1:n`.

Details

simCovariate will create a vector(s) of random variables from a specified R probability distribution. The distribution can be specified by entering the name or the name of the R function; partial matching is performed. For example, specifying a distribution as runif, 'runif', uniform, or u can be be used to generate random samples from a uniform distribution, in which case R's runif function is called. Additional arguments to the runif function are separated by commas. The function can be parameterized so that multiple covariates can be simulated from either the same distribution or from different distributions.

Value

A data frame of random numbers from the specified distribution, with number of columns equal to the the number of cov.names (ncol=length(cov.names)).

See Also

amData

Examples

```
# We can specify the distribution using a function, function name,
# or distribution name. Partial matching is performed. The examples
# below generate data for a single covariate; random seeds are not
# provided.

# All four examples provide same results and generate 10 random numbers
# from a uniform distribution.  In some examples the results are rounded;
# in other examples add.yr is set to TRUE to add a covariate called yr (year);
# in other examples a random seed is provided to ensure reproducibility.

simCovariate(u1 =list(dist= runif), n=10, add.yr=FALSE)
simCovariate(u2=list(dist = 'runif', round=2), n = 10, add.yr=TRUE)
simCovariate(u3=list(dist ='uniform', seed=302), n=10, add.yr=TRUE)
simCovariate(u4 = list(dist ='u', seed=302, round=3, min=0, max=10), n=10, add.yr=TRUE)

# If multiple covariates are to be simulated, create a list of covariates
# and then pass this covariate list as the argument, cov.list.  Here, create
# a dataframe with seven covariates from five distributions, and
# add a covariate called yr.
cov.list <- list(
    unif1 = list(dist = 'runif', min=0, max=10, seed=334, round=0),
    unif2 = list(dist = 'runif', min=0, max=10, seed=668, round=0),
    norm1=list(dist = 'normal', mean = 10,sd = 2, seed=10, round=1),
    norm2=list(dist = 'normal',  mean = 50, sd = 10, seed=15, round=2),
    beta1=list(dist = rbeta, shape1=2, shape2=1, seed=1002),
    binom1=list(dist = 'bin', size=20, prob=0.5, seed=561),
    bern1=list(dist='bernoulli', size = 1, prob = 0.5, seed = 6)
)

simCovariate(cov.list, n = 10, add.yr = TRUE)
```

Index