Package 'statisticalModeling'

November 12, 2016

Type Package

Title Functions for Teaching Statistical Modeling

Version 0.3.0

Author Daniel Kaplan

Maintainer Daniel Kaplan <kaplan@macalester.edu>

Description Provides graphics and other functions that evaluate and display models across many different kinds of model architecture. For instance, you can evaluate the effect size of a model input in the same way, regardless of architecture, interaction terms, etc.

License MIT + file LICENSE

LazyData TRUE

Depends R (>= 3.1), ggplot2

Imports lazyeval, rpart, mosaic, magrittr

RoxygenNote 5.0.1

Suggests dplyr, testthat, mosaicData, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2016-11-12 15:47:15

R topics documented:

AARP 2
Alder 3
Birth_weight 4
College_grades 5
collinearity 6
Crime 7
cv_pred_error 8
data_from_model 9
effect_size 9
ensemble 10

evaluate_model . . . 11
fmodel . . . 12
gf_point . . . 13
gmodel . . . 15
HDD_Minneapolis . . . 17
Houses_for_sale . . . 18
Mussels . . . 18
NCI60_snippet . . . 20
Oil_history . . . 20
reference_values . . . 21
Runners . . . 21
School_data . . . 22
Tadpoles . . . 23
train . . . 24
Trucking_jobs . . . 25
typical_levels . . . 25
Used_Fords . . . 26

Index **27**

AARP *Prices for life insurance*

Description

The AARP (original named the "American Association of Retired People" but now just AARP) offers life insurance to it's members. The data come from a full-page advertisement (circa 2012) in the "AARP Bulletin", which has the second largest circulation in the world of any magazine, with upward of 40 million subscribers. (Only the "AARP Magazine" has a larger circulation.)

Usage

```
data(AARP)
```

Format

A data frame with 36 observations on the following variables.

- `Age` The age of the person covered by the insurance policy.
- `Sex` The sex of the person covered by the insurance policy.
- `Coverage` The "death benefit" in 1000 USD.
- `Cost` Monthly cost in USD.

Details

Life insurance provides a "death benefit", money paid out to the insured person's survivors upon death of the insured. There is a cost for the insurance. Among other factors, the cost depends on both age and sex. (For this type of insurance, called "term insurance", the cost changes as the insured person ages.)

Source

The "AARP Bulletin". A copy of the ad is available at this link.

Examples

```
mod_1 <- lm(Cost ~ Age + Coverage, data = AARP)
effect_size(mod_1, ~ Coverage)
```

`Alder` *Nitrogen fixing by alder plants*

Description

These data were collected by biologist Mike Anderson in a study of nitrogen fixation by bacteria growing on the root nodules of alder bushes.

Usage

```
data(Alder)
```

Format

A data frame `Alder` with 196 rows and 24 variables:

- LAND. landscape, floodplain vs. upland.
- SAMPPER. sampling period: early, mid, late
- SPECIES. host species, Alnus tenuifolia (AT) vs. A. crispa (AC)
- STAGE. successional stage, early vs. late in floodplain and upland landscapes. This is not equivalent across landscapes.
- JULDAY. Julian day
- PERNODN. nodule percent nitrogen by mass
- RF. bacterial genotype
- SNF. nitrogen fixation rate of nodule tissue, umol N2/gram of nodule dry weight/hr
- SLA. specific leaf weight, grams of leaf weight/square-meter, dry
- ONECM. soil temperature at 1 cm depth
- FIVECM. soil temperature at 5 cm depth
- PERH2O. soil moisture, percent H2O by mass
- DEL. del15N of leaf tissue
- DELNOD. del15N of nodule tissue
- NPERAREA. leaf nitrogen content per unit leaf area
- NDiff. nitrogen content difference between leaf and nodule of the same plant
- delDiff. del15N difference between leaf and nodule of the same plant

- SITE. Site designations: 1A,B,C for replicate early succession floodplain sites, 4A,B,C for late succession floodplain, UP1A,B,C for early succession upland and UP3A,B,C for late succession upland
- HABSPEC. habitat+species, concatenated LAND, STAGE, SPECIES
- SITESPEC. concatenated SITE, SPECIES
- REP. replicate site within a given level of HABSPEC
- PLNO. plant number, unique for individuals of each species (AT1-180, AC1-270)

Details

Two questions that Anderson wanted to answer are: (1) Can any variation in nitrogen fixation (variable SNF) be attributed to genotype (variable RF)? (2) What are the major sources of variation in SNF and PERLEAFN? Variables of biological interest are seasonality (SAMPPER or JULDAY), soil temperature and moisture, and habitat differences (STAGE for host species AT andSTAGE and LAND for host species AC).

Three replicate sites were sampled for each landscape/stage combination in three sampling periods across the growing season. Site sampling was arranged in a Latin Square design in order to systematize any effects of seasonality on N2-fixation rates.

Source

Michael Anderson

References

Anderson MD, Ruess RW, Myrold DD, Taylor DL. "Host species and habitat affect modulation by specific Frankia genotypes in interior Alaska" Oecologia (2009) 160:619-630.

Examples

```
mod <- lm(logSNF ~ RF + SITESPEC, data = Alder)
```

Birth_weight *Birth weights and maternal data*

Description

Birth weight, date, and gestational period collected as part of the Child Health and Development Studies in 1961 and 1962. Information about the baby's parents — age, education, height, weight, and whether the mother smoked is also recorded. The data were present by Nolan and Speed to address the question of whether there is a link between maternal smoking and the baby's health.

Usage

```
data(Birth_weight)
```

Format

A data frame with 886 observations on the following variables.

- `baby_wt` Birth weight of baby, in ounces.
- `mother_wt` in pounds
- `gestation` Length of the pregnancy, in days.
- `smoker` Whether the mother smoked during the pregnancy.
- `income`Family yearly income in 2500USD increments 0 = under 2500, 1=2500-4999, ..., 8= 12,500-14,999, 9=15000+

Source

D. Nolan and T.P. Speed (2009) "Stat Labs: Mathematical Statistics Through Applications"

Examples

```
mod_1 <- lm(baby_wt ~ gestation + mother_wt, data = Birth_weight)
effect_size(mod_1, ~ gestation)
```

College_grades *Grades at a small college*

Description

These are the actual grades for 400+ individual students in the courses they took at a small, liberal-arts college in the midwest US. All the students graduated in 2006. Each row corresponds to a single student in a single course. The data have been de-identified by translating the student ID, the instructor ID, and the name of the department. Typically a graduating student has taken about 32 courses. As another form of de-identification, only half of the courses each student, selected randomly, are included. Only courses with 10 or more students enrolled were included.

Usage

```
data(College_grades)
```

Format

A data frame with 6146 Grades for 443 students.

- `grade` The letter grade for the student in this course: A is the highest.
- `sessionID` An identifier for the course taken. Courses offered multiple times in one semester or across semesters have individual IDs.
- `sid` The student ID
- `dept` The department in which the course was offered. 100 is entry-level, 200 sophomore-level, 300 junior-level, 400 senior-level.

- `enroll` Student enrollment in the course. This includes students who are not part of this sample.
- `iid` Instructor ID
- `gradepoint` A translation of the letter grade into a numerical scale. 4 is high. Some letter grades are not counted in a student's gradepoint average. These have `NA` for the gradepoint.

Source

The data were helpfully provided by the registrar of the college with the proviso that the de-identification steps outlined above be performed.

Examples

```
## Not run:
GPA <- lm(gradepoint ~ sid - 1, data = College_grades)

## End(Not run)
```

`collinearity` *Calculate measures of collinearity*

Description

Calculate measures of collinearity

Usage

```
collinearity(formula, data, format = c("SeIF", "degrees", "radians", "VIF"))
```

Arguments

`formula`	a formula giving, on the right-hand side, the explanatory variables to be considered. Iteractions, etc. may also be specified.
`data`	a data frame from which to draw the variables in the formula
`format`	choice of `"SeIF"` for inflation of standard errors, `"degrees"` or `"radians"` for collinearity described as an angle or `"VIF"` for the variance inflation factor (which is the square of SeIF).

Examples

```
collinearity( ~ cyl * disp * hp, data = mtcars)
collinearity( ~ cyl * disp * hp, data = mtcars, format = "degrees")
```

`Crime` *Data from the US FBI Uniform Crime Report, 1960*

Description

A report of the number of offenses reported to police per million population, and many other social and demographic variables. Each case corresponds to a state in the US.

Usage

```
data(Crime)
```

Format

A data frame with 47 cases, each of which is a US state, with observations on the following variables.

- `R` Crime rate: number of offenses reported to police per million population.
- `Age` Number of males aged 14-24 per 1000 population
- `N` State population (in 100,000s)
- `W` State-wise median value of transferable goods and assets or family income in tens of dollars.
- `X` Number of families per 1000 earning below half the median income.
- `ExDiff` Change in per capita expenditure on police by state and local government from 1950 to 1960
- `Ex0` 1960 per capita expenditures on police.

Source

FBI Uniform Crime Report via DASL: Data and Story Library

Examples

```
mod_1 <- lm(R ~ W, data = Crime)
mod_2 <- lm(R ~ X, data = Crime)
mod_3 <- lm(R ~ W + X, data = Crime)
effect_size(mod_1, ~ W)
effect_size(mod_3, ~ W)
effect_size(mod_2, ~ X)
effect_size(mod_3, ~ X)
```

`cv_pred_error` *Compare models with k-fold cross-validation*

Description

Compare models with k-fold cross-validation

Usage

```
cv_pred_error(..., k = 10, ntrials = 5, output = c("mse", "likelihood",
  "error_rate", "class"))
```

Arguments

`...`	one or more models on which to perform the cross-validation
`k`	the k in k-fold. cross-validation will use k-1/k of the data for training.
`ntrials`	how many random partitions to make. Each partition will be one case in the output of the function
`output`	The kind of output to produce from each cross-validation. See details.

Details

The purpose of cross-validation is to provide "new" data on which to test a model's performance. In k-fold cross-validation, the data set used to train the model is broken into new training and testing data. This is accomplished simply by using most of the data for training while reserving the remaining data for evaluating the model: testing. Rather than training a single model, k models are trained, each with its own particular testing set. The testing sets in the k models are arranged to cover the whole of the data set. On each of the k testing sets, a performance output is calculated. Which output is most appropriate depends on the kind of model: regression model or classifier. The most basic measure is the mean square error: the difference between the actual response variable in the testing data and the output of the model when presented with inputs from the testing data. This is appropriate in many regression models.

For classification models, two different outputs are appropriate. The first is the error rate: the frequency with which the classifier produces an incorrect output when presented with inputs from the testing data. This is a rather course measure. A more graded measure is the likelihood: the probability of the response values from the test data given the model. (The "class" method is exactly the same as "error rate", but provided for compatibility purposes with other software under development.)

data_from_model *Extract component parts of models*

Description

Functions for extracting data and names of explanatory and response variables from a model object

Usage

```
data_from_model(object, ...)

explanatory_vars(object, ...)

response_var(object, ...)
```

Arguments

object	the model you are extracting features from.
...	additional arguments (not used)

effect_size *Calculate effect sizes in a model*

Description

Like a derivative or finite-difference

Usage

```
effect_size(model, formula, step = NULL, bootstrap = FALSE, to = step,
  data = NULL, at = NULL, ...)
```

Arguments

model	the model from which the effect size is to be calculated
formula	a formula whose right-hand side is the variable with respect to which the effect size is to be calculated.
step	the numerical stepsize for the change var, or a comparison category for a categorical change var. This will be either a character string or a number, depending on the type of variable specified in the formula.
bootstrap	If TRUE, calculate a standard error using bootstrapping. Alternatively, you can specify the number of bootstrap replications (default:100).
to	a synonym for step. (In English, "to" is more appropriate for a categorical input, "step" for a quantitative. But you can use either.)

data	Specifies exactly the cases at which you want to calculate the effect size. Unlike ... or at, no new combinations will be created.
at	similar to ... but expects a list or dataframe of the values you want to set. Like ..., all combinations of the values specified will be used as inputs.
...	additional arguments for evaluation levels of explanatory variables or to be passed to predict(). For instance, for a glm, perhaps you want type = "response".

Details

When you want to force or restrict the effect size calculation to specific values for explanatory variables, list those variables and levels as a vector in ... For example, educ = c(10, 12, 16) will cause the effect size to be calculated at each of those three levels of education. Any variables whose levels are not specified in ... will have values selected automatically.

Examples

```
mod1 <- lm(wage ~ age * sex * educ + sector, data = mosaicData::CPS85)
effect_size(mod1, ~ sex)
effect_size(mod1, ~ sector)
effect_size(mod1, ~ age, sex = "M", educ = c(10, 12, 16), age = c(30, 40))
effect_size(mod1, ~ age, sex = "F", age = 34, step = 1)
effect_size(mod1, ~ sex, age = 35, sex = "M", to = "F" )
```

---

ensemble *Create bootstrapped ensembles of a model*

Description

Create bootstrapped ensembles of a model

Usage

```
ensemble(mod, nreps = 2, data = NULL)
```

Arguments

mod	a model whose data will be used for resampling
nreps	how many resampling trials should be created
data	a data table to use for resampling. This is not needed for many common model types, such as lm, glm, etc. See details.

Details

The approach to bootstrapping implemented by this function is to create a set of bootstrap trials all in one go. Then, other functions such as `effect_size` and `evaluate_model()` will be used to extract the information from each of the bootstrap replicates. Many model types in R carry the data used to train the model as part of the model object produced. For these types of models, e.g. `lm` and `glm`, there is no need to provide a value for the `data` argument. But there are some types of models for which the training data cannot be extracted from the model object. In such situations, you use `data =` to provide the data set to use for resampling.

`evaluate_model` *Evaluate a model for specified inputs*

Description

Find the model outputs for specified inputs. This is equivalent to the generic `predict()` function, except it will choose sensible values by default. This simplifies getting a quick look at model values.

Usage

```
evaluate_model(model = NULL, data = NULL, on_training = FALSE,
  nlevels = 3, at = NULL, ...)
```

Arguments

`model`	the model to display graphically
`data`	optional set of cases from which to extract levels for explanatory variables
`on_training`	flag whether to use the training data for evaluation. Only needed when there are random terms, e.g. from `rand()`, `shuffle()`, ... See details.
`nlevels`	how many levels to construct for input variables. For quantitative variables, this is a suggestion. `pretty()` will refine your choice. (default: 3)
`at`	named list giving specific values at which to hold the variables. Use this to override the automatic generation of levels for any or all explanatory variables. Unlike `data =` the variables given in `at =` or `...` will be crossed, so that the evaluation will occur at all combinations of the various levels.
`...`	arguments about or values at which to evaluate the model or the kind of output to be passed along to predict().

Details

This function is set up to let you look easily at typical outputs. The function will choose "typical" levels of the explanatory variables at which to evaluate the model. (See the examples.) The `nlevels` controls how many levels of these levels to use. If you wish to choose your own levels for one or more explanatory variables, give those variables as named arguments assigned to the levels you want. If you have a data frame with the desired inputs for some or all of the explanatory variables, use the `data` argument to pass those values.

There are two ways to evaluate the model on the training data. The first is to set the `data` argument to the same data frame used to train the model. The second is to use the `on_training = TRUE` argument. These are equivalent unless there is some random component among the explanatory terms, as with 'mosaic::rand()', 'mosaic::shuffle()' and so on. When you want to restrict/force the evaluation at specified values of an explanatory variable, include a vector of those variables in ... for instance `sex = "F"` will restrict the evaluation to females.

Value

A dataframe containing both the explanatory variable inputs and the resulting model output (in the 'model_value' field). This differs from the output of `predict()`, which for many model classes/architectures may be a vector or matrix.

A data frame containing both the inputs to the model and the corresponding outputs.

Examples

```
## Not run: mod1 <- lm(wage ~ age * sex + sector, data = mosaicData::CPS85)
evaluate_model(mod1)
mod3 <- glm(married == "Married" ~ age + sex * sector,
            data = mosaicData::CPS85, family = "binomial")
evaluate_model(mod3, nlevels = 2, type = "response")
evaluate_model(mod3, nlevels = 2, type = "response", at = list(sex = "F"))

## End(Not run)
```

`fmodel` *Plot out model values*

Description

Plot out model values

Usage

```
fmodel(model = NULL, formula = NULL, data = NULL, nlevels = 3,
  at = list(), prob_of = NULL, intervals = c("none", "confidence",
  "prediction"), post_transform = NULL, ...)
```

Arguments

`model`	the model to display graphically
`formula`	setting the y ~ x + color variables
`data`	optional data set from which to extract levels for explanatory variables
`nlevels`	how many levels to display for those variables shown at discrete levels
`at`	named list giving specific values at which to hold the variables. You can accomplish this without forming a list by using `....` See examples.

prob_of	if to show probability of a given level of the output, name the class here as a character string.
intervals	show confidence or prediction intervals: values "none", "confidence", "prediction"
post_transform	a scalar transformation and new name for the response variable, e.g. post_transform = c(price = exp) to undo a log transformation of price.
...	specific values for explantory variables and/or arguments to predict()

Details

Often you will want to show some data along with the model functions. You can do this with 'ggplot2::geom_point()' making sure to set the data argument to be a data frame with the cases you want to plot.

Examples

```
## Not run:
mod1 <- lm(wage ~ age * sex + sector, data = mosaicData::CPS85)
fmodel(mod1)
fmodel(mod1, ~ sector + sex + age) # not necessarily a good ordering
# show the data used for fitting along with the model
fmodel(mod1, ~ age + sex + sector, nlevels = 8) +
  ggplot2::geom_point(data = mosaicData::CPS85, alpha = 0.1)
require(ggplot2)
fmodel(mod1, ~ age + sex + sector, nlevels = 8) +
  geom_point(data = mosaicData::CPS85, alpha = 0.1) +
  ylim(0, 20)
mod2 <- lm(log(wage) ~ age + sex + sector, data = mosaicData::CPS85)
fmodel(mod2, post_transform = c(wage = exp)) # undo the log in the display
mod3 <- glm(married == "Married" ~ age + sex * sector,
            data = mosaicData::CPS85, family = "binomial")
fmodel(mod3, type = "response")
# Adding the raw data requires an as.numeric() trick when it's TRUE/FALSE
fmodel(mod3, ~ age + sex + sector, nlevels = 10, type = "response") +
  geom_point(data = mosaicData::CPS85,
  aes(x = age, y = as.numeric(married == "Married")), alpha = .1)

## End(Not run)
```

---

gf_point *gf_ plotting functions*

Description

These functions provide a formula interface to ggplot2 and various geoms. For plots with just one layer, the formula interface is more compact and is consistent with modeling and mosaic notation. The functions generate a ggplot command string which can be displayed by setting verbose = TRUE as an argument.

Usage

```
gf_point(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)

gf_jitter(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)

gf_line(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)

gf_path(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)

gf_density(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)

gf_density_2d(placeholder = NULL, formula = NULL, data = NULL,
  geom = type, verbose = FALSE, add = FALSE, ...)

gf_hex(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)

gf_hline(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)

gf_abline(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)

gf_hex(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)

gf_boxplot(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)

gf_freqpoly(placeholder = NULL, formula = NULL, data = NULL,
  geom = type, verbose = FALSE, add = FALSE, ...)

gf_histogram(placeholder = NULL, formula = NULL, data = NULL,
  geom = type, verbose = FALSE, add = FALSE, ...)

gf_text(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)

gf_counts(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)

gf_bar(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)
```

```
gf_frame(placeholder = NULL, formula = NULL, data = NULL, geom = type,
  verbose = FALSE, add = FALSE, ...)
```

Arguments

placeholder	Ignore this argument. See details.
formula	A formula describing the x and y (if any) variables and other aesthetics in a form like `y ~ x + color:red + shape:sex + alpha:0.5`
data	A data frame with the variables to be plotted
geom	A way to specify ggplot geoms that are not aliased to gf functions.
verbose	If `TRUE` print the ggplot2 command in the console.
add	If `TRUE` then construct just the layer with no frame. The result can be added to an existing frame.
...	Other arguments such as `position="dodge"`,

Details

These `gf_` functions are written to interact with ggplot objects. The `placeholder` argument is part of this interaction system; the end user can ignore it.

Examples

```
gf_point(mpg ~ hp + color:cyl + size:wt, data = mtcars)
gf_line(mpg ~ hp + group:cyl, data = mtcars) + facet_grid(~ am)
gf_histogram(~ Sepal.Length + fill:Species, data = iris)
gf_text(Sepal.Length ~ Sepal.Width + label:Species + color:Species , data = iris)
```

---

gmodel *Graph the function implicit in a model*

Description

Currently, this is the same as `fmodel`. I think the name `gmodel` makes more sense, and I want to be able to add additional functionality (such as including data points on the plot) without breaking fmodel.

Usage

```
gmodel(model = NULL, formula = NULL, data = NULL, nlevels = 3,
  at = list(), prob_of = NULL, intervals = c("none", "confidence",
  "prediction"), post_transform = NULL, ...)
```

Arguments

model	the model to display graphically
formula	setting the y ~ x + color variables
data	optional data set from which to extract levels for explanatory variables
nlevels	how many levels to display for those variables shown at discrete levels
at	named list giving specific values at which to hold the variables. You can accomplish this without forming a list by using .... See examples.
prob_of	if to show probability of a given level of the output, name the class here as a character string.
intervals	show confidence or prediction intervals: values "none", "confidence", "prediction"
post_transform	a scalar transformation and new name for the response variable, e.g. post_transform = c(price = exp) to undo a log transformation of price.
...	specific values for explantory variables and/or arguments to predict()

Details

#' Plot out model values

Often you will want to show some data along with the model functions. You can do this with 'ggplot2::geom_point()' making sure to set the data argument to be a data frame with the cases you want to plot.

Examples

```
## Not run:
mod1 <- lm(wage ~ age * sex + sector, data = mosaicData::CPS85)
fmodel(mod1)
fmodel(mod1, ~ sector + sex + age) # not necessarily a good ordering
# show the data used for fitting along with the model
fmodel(mod1, ~ age + sex + sector, nlevels = 8) +
  ggplot2::geom_point(data = mosaicData::CPS85, alpha = 0.1)
require(ggplot2)
fmodel(mod1, ~ age + sex + sector, nlevels = 8) +
  geom_point(data = mosaicData::CPS85, alpha = 0.1) +
  ylim(0, 20)
mod2 <- lm(log(wage) ~ age + sex + sector, data = mosaicData::CPS85)
fmodel(mod2, post_transform = c(wage = exp)) # undo the log in the display
mod3 <- glm(married == "Married" ~ age + sex * sector,
            data = mosaicData::CPS85, family = "binomial")
fmodel(mod3, type = "response")
# Adding the raw data requires an as.numeric() trick when it's TRUE/FALSE
fmodel(mod3, ~ age + sex + sector, nlevels = 10, type = "response") +
  geom_point(data = mosaicData::CPS85,
  aes(x = age, y = as.numeric(married == "Married")), alpha = .1)

## End(Not run)
```

HDD_Minneapolis *Heating degree days in Minneapolis, Minnesota, USA*

Description

A "heating degree day" is a measure of weather coldness. It's defined to be the difference between the outdoor ambient temperature and 65 degrees F, but has a value of zero when the ambient temperature is above 65 degrees. This difference is averaged over time and multiplied by the number of days in the time period covered. The heating degree day is often used as a measure of the demand for domestic heating in a locale.

Usage

```
data(HDD_Minneapolis)
```

Format

A data frame HDD_Minneapolis with 1412 rows and 4 variables:

- year the year
- month the month
- hdd the number of heating degree days for that period.
- loc the location at which the temperature was measured. In the early years, this was downtown Minneapolis. Later, the site was moved to the Minneapolis/Saint-Paul International Airport.

Details

These data report monthly heating degree days. For teaching purposes, the data give an extreme example of how a relationship (hdd vs year) can be revealed by including a covariate (month). Although interest focusses on the change in temperature over the century the data cover, there is such regular seasonal variation that no systematic trend over the years is evident unless month is taken into account.

Examples

```
mod_1 <- lm(hdd ~ year, data = HDD_Minneapolis)
mod_2 <- lm(hdd ~ year + month, data = HDD_Minneapolis)
```

Houses_for_sale *Houses for sale*

Description

A random sample of 1,728 homes taken from public records from the Saratoga County (http://www.saratogacountyny.gov/dep property-tax-service-agency/). Collected by Candice Corvetti (Williams College '07) for her senior thesis.

Usage

```
data(Houses_for_sale)
```

Format

A dataframe with 1728 cases, each of which is a house for sale.

Details

These data are part of a case study developed by Prof. Dick de Veaux at Williams. They are available from the American Statistical Association's Stat 101 collection of case studies and included in this package for convenience.

References

Dick De Veaux (2015) "How much is a fireplace worth?" Stats 101: A resource for teaching introductory statistics, American Statistical Association

Examples

```
mod_1 <- lm(price ~ fireplaces, data = Houses_for_sale)
mod_2 <- lm(price ~ fireplaces + living_area, data = Houses_for_sale)
effect_size(mod_1, ~ fireplaces)
effect_size(mod_2, ~ fireplaces)
fmodel(mod_2, ~ living_area + fireplaces)
```

Mussels *Metabolism of zebra mussels*

Description

Zebra mussels are a small, fast reproducing species of freshwater mussel native to the lakes of southeast Russia. They have accidentally been introduced in other areas, competing with native species and creating problems for people as they cover the undersides of docks and boats, clog water intakes and other underwater structures. Zebra mussels even attach themselves to other mussels, sometimes starving those mussels.

Usage

```
data(Mussels)
```

Format

A data frame `Mussels` with 30 rows and 11 variables.

- `GroupID` ID for the cluster of mussels growing on a substrate.
- `dry.mass` The mass of the mussels (as a group) after dehydration.
- `count` How many mussels were in the cluster.
- `attachment` The substrate to which the mussels were attached.
- `lipid` Percentage of dry mass that is lipid.
- `protein` Percentage of dry mass that is protein.
- `carbo` Percentage of dry mass that is carbohydrate.
- `ash` Percentage of dry mass that is ash.
- `ammonia` Nitrogen excretion measured as ammonia in mg per hour for the group.
- `Kcal` Total calorific value of the tissue in kilo-calories per gram.
- `O2` Oxygen uptake in mg per hour for the group.

Details

Ecologists Shirley Baker and Daniel Hornbach examined whether zebra mussels gain an advantage by attaching to other mussels rather than to rocks.(baker-hornbach-2008) The ecologists collected samples of small rocks and Amblema plicata mussels, each of which had a collection of zebra mussels attached. The samples were transported to a laboratory where the group of mussels from each individual rock or Amblema were removed and placed in an aquarium equipped to measure oxygen uptake and ammonia excretion. After these physiological measurements were made, the biochemical composition of the mussel tissue was determined: the percentage of protein, lipid, carbohydrate, and ash. Baker and Hornbach found that zebra mussels attached to Amblema had greater physiological activity than those attached to rocks as measured by oxygen uptake and ammonia excretion. But this appears to be a sign of extra effort for the Amblema-attached zebra mussels, since they had lower carbohydrate and lipid levels. In other words, attaching to Amblema appears to be disadvantageous to the zebra mussels compared to attaching to a rock.

Examples

```
Mussels$ind.mass <- with(Mussels, dry.mass/count)
mod_1 <- lm(O2/count ~ attachment, data = Mussels)
mod_2 <- lm(ammonia/count ~ attachment, data = Mussels)
mod_3 <- lm(O2/count ~ ind.mass + attachment, data = Mussels)
mod_4 <- lm(ammonia/count ~ ind.mass + attachment, data = Mussels)
```

NCI60_snippet *Gene expression in cancer cells.*

Description

The data come from a National Cancer Institute study of gene expression in cell lines drawn from various sorts of cancer. Each row corresponds to a different cell line. The type of cancer is identified by the first two or three letters of the row names, e.g. CO is colon, ME is melanoma, RE is kidney.

Usage

```
data(NCI60_snippet)
```

Format

A data frame `NCI60_snippet` with 60 rows and 6000 variables.

Details

For each row, there are 6000 measurements of the gene expression as identified by activity on a microarray probe. The variable names are the names of the probes.

Examples

```
data(NCI60_snippet)
```

Oil_history *Historical production of crude oil, worldwide 1880-2014*

Description

Annual production of crude oil, in millions of barrels (mbbl).

Usage

```
data(Oil_history)
```

Format

A data frame with 47 cases, each of which is a US state, with observations on the following variables.

- `year` the year for which production is reported
- `mbbl` oil production in millions of barrels

Source

Assembled from older information from RH Romer (1976) "Energy: An Introduction to Physics" and more recent data from data.oecd.org.

Examples

```
model <- lm(log(mbbl) ~ year, data = Oil_history)
fmodel(model)
```

---

reference_values *Compute sensible values from a data set for use as a baseline*

Description

Compute sensible values from a data set for use as a baseline

Usage

```
reference_values(data, n = 1, at = list())
```

Arguments

data	a data frame
n	number of values for specified variables: could be a single number or a list assigning a number of levels for individual variables
at	optional values at which to set values: a list whose names are the variables whose values are to be set.

Details

Variables not listed in at will be assigned levels using these principles: Categorical variables: the most populated levels. Quantitative variables: central quantiles, e.g. median for n=1, 0.33 and 0.67 for n=2, and so on.

Runners *Performance of runners in a ten-mile race as they age*

Description

These data are assembled from the records of the "Cherry Blossom Ten Miler" held in Washington, DC each April. The records span the years 1999 to 2008.

Usage

```
data(Runners)
```

Format

A data frame `Runners` with 24,334 rows and 8 variables.

- `age` The runners age at the time of the race.
- `net` The time (in min.) elapsed from when the runner crossed the start line until the finish.
- `gun` The time (in min.) from when the starter's gun was fired to when the runner finished the race. With so many participants, some runners do not reach the start line until several minutes after the gun.
- `sex` The runner's sex.
- `previous` How many times the runner participated before this year's race.
- `nruns` How many of the 10 years' runs the runner eventually participated in.
- `start_position` A made-up categorical description of where the runner was situated in the line up for the race..

Details

There are about 10,000 participants each year, of whom roughly half have run the race previously. During the ten years of these records, some 25 runners ran in ech of the years, 73 ran in nine of the years, and so on. The data allow you to track the performace of runners as they age. This simplified version of the data does not include personal identifying data other than sex, age, and number of previous runs in any given year. (Runs before 1999 are not considered.)

Examples

```
mod_1 = lm(net ~ age + sex, data = Runners)
```

School_data *Simulated data bearing on school vouchers*

Description

In the US, there have been controversial proposals to provide vouchers to students in failing public schools. The vouchers would allow the students to attend private schools. There are arguments pro and con that are often rooted in political philosophy (free choice!) and politics. The presumption behind the *pro* arguments is that attending private schools would create better outcomes for students.

Usage

```
data(School_data)
```

Format

A data frame with 500 cases, each of which is a simulated student, with observations on the following variables.

- `test_score` a simulated test score for the student
- `school` whether the student attended public or private school
- `lottery` whether the student was entered into a lottery for a private-school voucher
- `group` the racial/ethnic group of the student
- `acad_motivation` the overall level of involvement and concern of the student's parents for the student's academic performance
- `relig_motivation` the overall level of interest motivated by religion. This is potentially an issue because a large majority of urban private schools are Catholic.

Details

A reasonable way to test this presumption is to compare test scores for students in public and private schools. One famous analysis (Howell and Peterson, 2003, "The Education Gap: Vouchers and Urban Schools") found that voucher schools are most helpful for African-American students, and not so much for white or Hispanic students.

The `School_data` data frame comes from a simulation designed by the package author to replicate the overall results but supporting a very different policy recommendation. WARNING: This is just a simulation, reflecting one hypothesis about how the world might work. Don't be tempted to draw conclusions about the actual factors involved in school performance from such simulated data.

Examples

```
lm(test_score ~ school, data = School_data)
# the simulation mechanism itself:
nstudents <- 500
acad_motivation <- rnorm(nstudents)
group <- sample(c("black", "hispanic", "white"), replace = TRUE, size = nstudents)
relig_motivation <- ifelse( group == "black", -1, ifelse(group == "white", 0, 1))
relig_motivation <- rnorm(nstudents, mean = relig_motivation)
lottery <- (acad_motivation + relig_motivation) > 0
school <- ifelse( (runif(nstudents) + .8* lottery ) > 1, "private", "public")
test_score <- rnorm(nstudents, mean = 100 - 5 * (school == "private") + 20 * acad_motivation)
School_data <- data.frame(test_score, acad_motivation, group, relig_motivation, lottery, school)
```

Tadpoles *Swimming speed of tadpoles.*

Description

Tim Watkins examined the swimming speed of tadpoles as a function of the water temperature, and the water temperature at which the tadpoles had been raised. Since size is a major determinant of speed, the tadpole's length was measured as well.

Usage

```
data(Tadpoles)
```

Format

A data frame `Trucking_jobs` with 129 rows and 11 variables.

- `group` Whether the tadpole was raised in cold water ("c") or warm ("w").
- `rtemp` The temperature (C) of the water in which the swimming speed was measured. (The swimming channel is called a "race".)
- `length` The tadpole's length, in mm.
- `vmax` The maximum swimming speed attained in one trial, in mm/sec.

Details

It was hypothesized that tadpoles would swim faster in the temperature of water close to that in which they had been raised.

Examples

```
mod_1 = lm(vmax ~ poly(rtemp, 2) * group + length, data = Tadpoles)
```

`train` *Training statistical models*

Description

This function coordinates four components in building statistical models: 1) the data to use for training 2) the response variable and 3) the explanatory variables, specified as a model formula 4) the architecture of the model.

Usage

```
train(data, formula = formula(data), architecture = "lm")
```

Arguments

`data`	the data to use for training
`formula`	the formula describing the structure of the relationship among variables
`architecture`	the model architecture, given as a function name, quoted or not, e.g. lm or "lm"

Details

This is a light wrapper on the architecture-specific function fitting programs

Trucking_jobs *Earnings of workers at a trucking company.*

Description

A dataset from a mid-western US trucking company on annual earnings of its employees in 2007. Datasets like this are used in audits by the Federal Government to look for signs of discrimination.

Usage

```
data(Trucking_jobs)
```

Format

A data frame `Trucking_jobs` with 129 rows and 11 variables.

- `sex` The employee's sex: M or F
- `earnings` Annual earnings, in dollars. Hourly wages have been converted to a full-time basis.
- `age` The employee's age, in years.
- `title` The employee's job title.
- `hiredyears` How long the employee has been working for the company.

Examples

```
mod_1 = lm(earnings ~ age + hiredyears + sex, data = Trucking_jobs)
```

typical_levels *Find typical levels of explanatory variables in a model/dataset.*

Description

This function tries to choose sensible values of the explanatory variables from the data used to build a model or any other specified data. (or from data specified with the `data =` argument.)

Usage

```
typical_levels(model = NULL, data = NULL, nlevels = 3, at = list(), ...)
```

Arguments

`model`	the model to display graphically
`data`	optional data frame from which to extract levels for explanatory variables
`nlevels`	how many levels to construct for input variables. For quantitative variables, this is a suggestion. `pretty()` will determine
`at`	named list giving specific values at which to hold the variables. Use this to override the automatic generation of levels for any or all explanatory variables.
`...`	a more concise mechanism to passing desired values for variables

Details

For categorical variables, the most populated levels are used. For quantitative variables, a sequence of `pretty()` values is generated.

Value

A dataframe containing all combinations of the selected values for the explanatory variables. If there are p explanatory variables, there will be about `nlevels^p` cases.

Examples

```
## Not run: mod1 <- lm(wage ~ age * sex + sector, data = mosaicData::CPS85)
typical_levels(mod1)
mod3 <- glm(married == "Married" ~ age + sex * sector,
            data = mosaicData::CPS85, family = "binomial")
typical_levels(mod3, nlevels = 2)

## End(Not run)
```

Used_Fords *Prices of used Ford automobiles in 2009*

Description

These data were compiled by Macalester College students for a class project. Then-undergraduates Elise delMas, Emiliano Urbina, and Candace Groth collected the data from `cars.com`

Usage

```
data(Used_Fords)
```

Format

A data frame `Used_Fords` with 635 rows and 7 variables.

- `Price` The asking price for the car in USD.
- `Year` The model year of the car.
- `Mileage` The reported odometer reading.
- `Location` Which of the several regions the car was marketed in.
- `Color` The car's body color.
- `Age` The age of the car at the time the data were collected in 2009. This is directly related to `Year`

Examples

```
mod_1 <- lm(Price ~ Mileage, data = Used_Fords)
mod_2 <- lm(Price ~ Mileage + Age, data = Used_Fords)
```

Index

∗Topic **datasets**
AARP, 2
Alder, 3
Birth_weight, 4
College_grades, 5
Crime, 7
HDD_Minneapolis, 17
Houses_for_sale, 18
Mussels, 18
NCI60_snippet, 20
Oil_history, 20
Runners, 21
School_data, 22
Tadpoles, 23
Trucking_jobs, 25
Used_Fords, 26

AARP, 2
Alder, 3

Birth_weight, 4

College_grades, 5
collinearity, 6
Crime, 7
cv_pred_error, 8

data_from_model, 9

effect_size, 9
ensemble, 10
evaluate_model, 11
explanatory_vars *(*data_from_model*)*, 9

fmodel, 12

gf_abline *(*gf_point*)*, 13
gf_bar *(*gf_point*)*, 13
gf_boxplot *(*gf_point*)*, 13
gf_counts *(*gf_point*)*, 13
gf_density *(*gf_point*)*, 13
gf_density_2d *(*gf_point*)*, 13
gf_frame *(*gf_point*)*, 13
gf_freqpoly *(*gf_point*)*, 13
gf_hex *(*gf_point*)*, 13
gf_histogram *(*gf_point*)*, 13
gf_hline *(*gf_point*)*, 13
gf_jitter *(*gf_point*)*, 13
gf_line *(*gf_point*)*, 13
gf_path *(*gf_point*)*, 13
gf_point, 13
gf_text *(*gf_point*)*, 13
gmodel, 15

HDD_Minneapolis, 17
Houses_for_sale, 18

Mussels, 18

NCI60_snippet, 20

Oil_history, 20

reference_values, 21
response_var *(*data_from_model*)*, 9
Runners, 21

School_data, 22

Tadpoles, 23
train, 24
Trucking_jobs, 25
typical_levels, 25

Used_Fords, 26