Package 'AppliedPredictiveModeling'

May 6, 2026

Type Package

Title Functions and Data Sets for 'Applied Predictive Modeling'

Version 1.1-7

Date 2018-05-22

Author Max Kuhn, Kjell Johnson

Maintainer Max Kuhn <mxkuhn@gmail.com>

Description A few functions and several data set for the Springer book 'Applied Predictive Modeling'.

URL http://appliedpredictivemodeling.com/

Depends R (>= 2.10)

Imports CORElearn, MASS, plyr, reshape2, lattice, ellipse

Suggests caret (>= 6.0-22)

License GPL-2

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-22 19:14:21 UTC

Contents

`AppliedPredictiveModeling-package`
Data, Functions and Scripts for 'scriptLocation'

Description

This package can be used to reproduce the analyses in the text. Scripts for each chapter are located in the "chapters" directory. Use `scriptLocation()` to find their exact location.

Details

Package:	AppliedPredictiveModeling
Type:	Package
Version:	1.1-1
Date:	2013-05-29
License:	GPL

Author(s)

Max Kuhn

Maintainer: Max Kuhn <mkuhn@gmail.com>

References

Kuhn M and Johnson K (2013) Applied Predictive Modeling, Springer, NY

abalone *Abalone Data*

Description

The Abalone data consist of data from 4177 abalones. The data consist of measurements of the type (male, female and infant), the longest shell measurement, the diameter, height and several weights (whole, shucked, viscera and shell). The outcome is the number of rings. The age of the abalone is the number of rings plus 1.5.

The data are taken from the UCI database (http://archive.ics.uci.edu/ml/datasets/Abalone).

Usage

```
data(abalone)
```

Value

abalone	a data frame with 4177 rows and 9 columns

Examples

```
data(abalone)
```

AlzheimerDisease *Alzheimer's Disease CSF Data*

Description

Washington University conducted a clinical study to determine if biological measurements made from cerebrospinal fluid (CSF) can be used to diagnose or predict Alzheimer's disease (Craig-Schapiro et al. 2011). These data are a modified version of the values used for the publication.

The R factor vector `diagnosis` contains the outcome data for 333 of the subjects. The demographic and laboratory results are collected in the data frame `predictors`.

One important indicator of Alzheimer's disease is the genetic background of a subject. In particular, what versions of the Apolipoprotein E gene inherited from one's parents has an association with the disease. There are three variants of the gene: E2, E3 and E4. Since a child inherits a version of the gene from each parent, there are six possible combinations (e.g. E2/E2, E2/E3, and so on). This data is contained in the predictor column named `Genotype`.

Usage

```
data(AlzheimerDisease)
```

Value

diagnosis	labels for the patients, either "Impaired" or "Control".
predictors	predictors for demographic data (eg. age, gender), genotype and assay results.

Source

Craig-Schapiro, R., Kuhn, M., Xiong, C., Pickering, E. H., Liu, J., Misko, T. P., Perrin, R. J., et al. (2011). Multiplexed Immunoassay Panel Identifies Novel CSF Biomarkers for Alzheimer's Disease Diagnosis and Prognosis. PLoS ONE, 6(4), e18850.

Examples

```
data(AlzheimerDisease)
```

---

bookTheme *Lattice Themes*

Description

Two **lattice** themes used throughout the book.

Usage

```
bookTheme(set = TRUE)

transparentTheme(set = TRUE, pchSize = 1, trans = 0.2)
```

Arguments

set	a logical: should these settings be applied to the current device?
pchSize	the size of the plot symbols
trans	the amount of transparency (via the alpha channel). Note that transparency is not supported by all graphics devices.

Details

When using these functions to save a plot, make sure to invoke them after the device has been opened (e.g. after calls such as `pdf()`.

Value

Each function returns a list of theme parameters. See Sarkar (2008) or `trellis.par.get` for specific details.

Author(s)

Max Kuhn

References

Some of the colors are based on values from ColorBrewer http://www.colorbrewer.org.

Sarkar, D. (2008). Lattice: Multivariate Data Visualization with R. UseR! (1st ed. p. 286). Springer.

Examples

```
library(lattice)

example <- quadBoundaryFunc(100)

bookTheme(set = TRUE)
xyplot(X2 ~ X1, data = example, groups = class, auto.key = TRUE)

transparentTheme(set = TRUE, trans = .6)
xyplot(X2 ~ X1, data = example, groups = class, auto.key = TRUE)
```

ChemicalManufacturingProcess

Chemical Manufacturing Process Data

Description

This data set contains information about a chemical manufacturing process, in which the goal is to understand the relationship between the process and the resulting final product yield. Raw material in this process is put through a sequence of 27 steps to generate the final pharmaceutical product. The starting material is generated from a biological unit and has a range of quality and characteristics. The objective in this project was to develop a model to predict percent yield of the manufacturing process. The data set consisted of 177 samples of biological material for which 57 characteristics were measured. Of the 57 characteristics, there were 12 measurements of the biological starting material, and 45 measurements of the manufacturing process. The process variables included measurements such as temperature, drying time, washing time, and concentrations of by–products at various steps. Some of the process measurements can be controlled, while others are observed. Predictors are continuous, count, categorical; some are correlated, and some contain missing values. Samples are not independent because sets of samples come from the same batch of biological starting material.

Usage

```
data(ChemicalManufacturingProcess)
```

Value

`ChemicalManufacturingProcess`: a data frame with columns for the outcome (`Yield`) and the predictors (`BiologicalMaterial01` though `BiologicalMaterial12` and `ManufacturingProcess01` though `ManufacturingProcess45`

Examples

```
data(ChemicalManufacturingProcess)
```

---

concrete *Compressive Strength of Concrete from Yeh (1998)*

Description

Yeh (1998) describes a collection of data sets from different sources that can be used for modeling the compressive strength of concrete formulations as a functions of their ingredients and age.

Usage

```
data(concrete)
```

Details

The data are from Yeh (1998) and taken from the UCI ML website http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength.

There are 1030 data points from the UCI website, but the paper states that approximately 1,000 samples were made, but only 727 were analyzed in the source material. It is unclear which samples were excluded.

Value

`concrete`	data frame of data with predictor columns `Cement`, `BlastFurnaceSlag`, `FlyAsh`, `Water`, `Superplasticizer`, `CoarseAggregate`, `FineAggregate` and `Age` with response column `CompressiveStrength`. These are the amounts.
`mixtures`	The same data where all the ingredients have been converted to proportions of the total amounts.

Source

Yeh, I. C. (1998). Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12), 1797-1808. Elsevier.

Examples

```
data(concrete)

library(caret)

### Split used in the book:
set.seed(975)
inTrain <- createDataPartition(mixtures$CompressiveStrength, p = 3/4)[[1]]
training <- mixtures[ inTrain,]
testing  <- mixtures[-inTrain,]
```

FuelEconomy *Fuel Economy Data*

Description

The http://fueleconomy.gov website, run by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy and the U.S. Environmental Protection Agency, lists different estimates of fuel economy for passenger cars and trucks. For each vehicle, various characteristics are recorded such as the engine displacement or number of cylinders. Along with these values, laboratory measurements are made for the city and highway miles per gallon (MPG) of the car.

Predictors extracted from the website include: EngDispl, NumCyl, Transmission, AirAspirationMethod, NumGears, TransLockup, TransCreeperGear, DriveDesc, IntakeValvePerCyl, ExhaustValvesPerCyl, CarlineClassDesc, VarValveTiming and VarValveLift. The outcome used in the book is in column FE and is the unadjusted highway data.

Usage

```
data(FuelEconomy)
```

Value

cars2010	data in cars from model year 2010.
cars2011	cars introduced in 2011 that were not in the model year 2010 data.
cars2012	cars introduced in 2012 that were not in the model year 2010 or 2011 data

Examples

```
data(FuelEconomy)

library(lattice)

### Plot shown in the text:

cars2010 <- cars2010[order(cars2010$EngDispl),]
cars2011 <- cars2011[order(cars2011$EngDispl),]

cars2010a <- cars2010
cars2010a$Year <- "2010 Model Year"
cars2011a <- cars2011
cars2011a$Year <- "2011 Model Year"

plotData <- rbind(cars2010a, cars2011a)

plotTheme <- bookTheme(FALSE)
plotTheme$plot.symbol$col <- rgb(.2, .2, .2, .5)
plotTheme$plot.symbol$cex <- 0.7
trellis.par.set(plotTheme)
```

```
xyplot(FE ~ EngDispl|Year, plotData,
       xlab = "Engine Displacement",
       ylab = "Fuel Efficiency (MPG)",
       between = list(x = 1.2))
```

`getPackages` *Install Packages for Each Chapter*

Description

This function identifies the physical location on the user's computer where the chapter R scripts are located.

Usage

```
getPackages(chapter, ...)
```

Arguments

`chapter`	an integer vector (or character versions of the integer) for the chapter number. See Details below:
`...`	options to pass to `install.packages`

Details

Chapter names and packages. about dependencies.

Author(s)

Max Kuhn

Examples

```
## Not run:
getPackages(2)
getPackages(2:3)
getPackages("4")

## End(Not run)
```

hepatic *Hepatic Injury Data*

Description

This data set was used to develop a model for predicting compounds' probability of causing hepatic injury (i.e. liver damage). This data set consisted of 281 unique compounds; 376 predictors were measured or computed for each. The response was categorical (either "None", "Mild" or "Severe"),and was highly unbalanced.

This kind of response often occurs in pharmaceutical data because companies steer away from creating molecules that have undesirable characteristics. Therefore, well-behaved molecules often greatly outnumber undesirable molecules. The predictors consisted of measurements from 184 biological screens and 192 chemical feature predictors. The biological predictors represent activity for each screen and take values between 0 and 10 with a mode of 4. The chemical feature predictors represent counts of important sub-structures as well as measures of physical properties that are thought to be associated with hepatic injury.

Usage

```
data(hepatic)
```

Value

`bio`	Biological screen results.
`chem`	Chemical fingerprints for sub-structures.
`injury`	A factor vector of outcomes.

Examples

```
data(hepatic)
```

logisticCreditPredictions *Logistic Regression Predictions for the Credit Data*

Description

add some notes

Usage

```
data(solubility)
```

Value

A data frame with columns

Bad	The predicted class probability for bad credit.
Good	The predicted class probability for good credit.
pred	The predicted class.
obs	The observed class

Examples

```
## show code to make the predictions
```

`permeability` *Permeability Data*

Description

This pharmaceutical data set was used to develop a model for predicting compounds' permeability. In short, permeability is the measure of a molecule's ability to cross a membrane. The body, for example, has notable membranes between the body and brain, known as the blood-brain barrier, and between the gut and body in the intestines. These membranes help the body guard critical regions from receiving undesirable or detrimental substances. For an orally taken drug to be effective in the brain, it first must pass through the intestinal wall and then must pass through the blood-brain barrier in order to be present for the desired neurological target. Therefore, a compound's ability to permeate relevant biological membranes is critically important to understand early in the drug discovery process. Compounds that appear to be effective for a particular disease in research screening experiments, but appear to be poorly permeable may need to be altered in order improve permeability, and thus the compound's ability to reach the desired target. Identifying permeability problems can help guide chemists towards better molecules.

Permeability assays such as PAMPA and Caco-2 have been developed to help measure compounds' permeability (Kansy et al, 1998). These screens are effective at quantifying a compound's permeability, but the assay is expensive labor intensive. Given a sufficient number of compounds that have been screened, we could develop a predictive model for permeability in an attempt to potentially reduce the need for the assay. In this project there were 165 unique compounds; 1107 molecular fingerprints were determined for each. A molecular fingerprint is a binary sequence of numbers that represents the presence or absence of a specific molecular sub-structure. The response is highly skewed, the predictors are sparse (15.5 percent are present), and many predictors are strongly associated.

Usage

```
data(permeability)
```

Value

permeability	permeability values for each compound.
fingerprints	a matrix of binary fingerprint indicator variables.

Source

Kansy, M., Senner, F., and Gubernator, K. (1998). Physicochemical High Throughput Screening: Parallel Artificial Membrane Permeation Assay in the Description of Passive Absorption Processes. J. Med. Chem, 41(7), 1007-1010.

Examples

```
data(permeability)

hist(permeability)

summary(apply(fingerprints, 2, mean))
```

`permuteRelief` *Permutation Statistics for the Relief Algorithm*

Description

This function uses a permutation approach to determining the relative magnitude of Relief scores (Kira and Rendell, 1992 and Kononenko, 1994).

Usage

```
permuteRelief(x, y, nperm = 100, ...)
```

Arguments

`x`	a data frame of predictor data
`y`	a vector of outcomes
`nperm`	the number of random permutations of the data
`...`	options to pass to `attrEval`, such as the exact Relief algorithm, to use

Details

The scores for each predictor are computed using the original data and after outcome data are randomly scrambled (`nprem` times). The mean and standard deviation of the permuted values are determined and a standardized version of the observed scores are determined by subtracting the permuted means from the original values, then dividing each by the corresponding standard deviation.

Value

a list with elements

`standardized`	a vector of standardized predictor scores
`permutations`	the values of the permuted scores, for plotting to assess the permutation distribution
`observed`	the observed scores
`options`	a list of options passed using ...

Author(s)

Max Kuhn

References

Kira, K., & Rendell, L. (1992). The feature selection problem: Traditional methods and a new algorithm. *Proceedings of the Eleventh International Conference on Machine Learning*, 129-129.

Kononenko, I. (1994). Estimating attributes: analysis and extensions of RELIEF. Machine Learning: ECML-94, 171-182.

See Also

attrEval

Examples

```
set.seed(874)
reliefEx3 <- easyBoundaryFunc(500)
reliefEx3$X1 <- scale(reliefEx3$X1)
reliefEx3$X2 <- scale(reliefEx3$X2)
reliefEx3$prob <- NULL

standardized <- permuteRelief(reliefEx3[, 1:2], reliefEx3$class,
                              ## For efficiency, a small number of
                              ## permutations are used here.
                              nperm = 50,
                              estimator="ReliefFequalK",
                              ReliefIterations= 50)
```

---

quadBoundaryFunc *Functions for Simulating Data*

Description

These functions simulate data that are used in the text.

Usage

```
quadBoundaryFunc(n)

easyBoundaryFunc(n, intercept = 0, interaction = 2)
```

Arguments

n	the sample size
intercept	the coefficient for the logistic regression intercept term
interaction	the coefficient for the logistic regression interaction term

Details

The quadBoundaryFunc function creates a class boundary that is a function of both predictors. The probability values are based on a logistic regression model with model equation: $-1 - 2X_1 - 0.2X_1^2 + 2X_2^2$. The predictors here are multivariate normal with mean (1, 0) and a moderate degree of positive correlation.

Similarly, the easyBoundaryFunc uses a logistic regression model with model equation: $intercept - 4X_1 + 4X_2 + interaction \times X_1 \times X_2$. The predictors here are multivariate normal with mean (1, 0) and a strong positive correlation.

Value

Both functions return data frames with columns

X1	numeric predictor value
X2	numeric predictor value
prob	numeric value reflecting the true probability of the first class
class	a factor variable with levels 'Class1' and 'Class2'

Author(s)

Max Kuhn

Examples

```
## in Chapter 11, 'Measuring Performance in Classification Model'
set.seed(975)
training <- quadBoundaryFunc(500)
testing <- quadBoundaryFunc(1000)


## in Chapter 20, 'Factors That Can Affect Model Performance'
set.seed(615)
dat <- easyBoundaryFunc(200, interaction = 3, intercept = 3)
dat$X1 <- scale(dat$X1)
dat$X2 <- scale(dat$X2)
dat$Data <- "Original"
dat$prob <- NULL

## in Chapter X, 'An Introduction to Feature Selection'

set.seed(874)
reliefEx3 <- easyBoundaryFunc(500)
reliefEx3$X1 <- scale(reliefEx3$X1)
reliefEx3$X2 <- scale(reliefEx3$X2)
reliefEx3$prob <- NULL
```

schedulingData *HPC Job Scheduling Data*

Description

These data consist of information on 4331 jobs in a high performance computing environment. Seven attributes were recorded for each job along with a discrete class describing the execution time.

The predictors are: `Protocol` (the type of computation), `Compounds` (the number of data points for each jobs), `InputFields` (the number of characteristic being estimated), `Iterations` (maximum number of iterations for the computations), `NumPending` (the number of other jobs pending at the time of launch), `Hour` (decimal hour of day for launch time) and `Day` (of launch time).

The classes are: `VF` (very fast), `F` (fast), `M` (moderate) and `L` (long).

Usage

```
data(schedulingData)
```

Value

`schedulingData` a data frame with 4331 rows and 8 columns

Examples

```
data(schedulingData)

library(caret)

set.seed(1104)
inTrain <- createDataPartition(schedulingData$Class, p = .8, list = FALSE)

schedulingData$NumPending <- schedulingData$NumPending + 1

trainData <- schedulingData[ inTrain,]
testData  <- schedulingData[-inTrain,]

modForm <- as.formula(Class ~ Protocol + log10(Compounds) +
                      log10(InputFields)+ log10(Iterations) +
                      log10(NumPending) + Hour + Day)
```

scriptLocation *Find Chapter Script Files*

Description

This function identifies the physical location on the user's computer where the chapter R scripts are located.

Usage

```
scriptLocation()
```

Author(s)

Max Kuhn

Examples

```
scriptLocation()
```

segmentationOriginal *Cell Body Segmentation*

Description

Hill, LaPan, Li and Haney (2007) develop models to predict which cells in a high content screen were well segmented. The data consists of 119 imaging measurements on 2019. The original analysis used 1009 for training and 1010 as a test set (see the column called `Case`).

The outcome class is contained in a factor variable called `Class` with levels "PS" for poorly segmented and "WS" for well segmented.

A pre-processed version of these data can be found in the **caret** package.

Usage

```
data(segmentationOriginal)
```

Value

`segmentationOriginal`
: data frame of cells

Source

Hill, LaPan, Li and Haney (2007). Impact of image segmentation on high-content screening data quality for SK-BR-3 cells, *BMC Bioinformatics*, Vol. 8, pg. 340, http://www.biomedcentral.com/1471-2105/8/340.

solubility *Solubility Data*

Description

Tetko et al. (2001) and Huuskonen (2000) investigated a set of compounds with corresponding experimental solubility values using complex sets of descriptors. They used linear regression and neural network models to estimate the relationship between chemical structure and solubility. For our analyses, we will use 1267 compounds and a set of more understandable descriptors that fall into one of three groups: 208 binary "fingerprints" that indicate the presence or absence of a particular chemical sub-structure, 16 count descriptors (such as the number of bonds or the number of Bromine atoms) and 4 continuous descriptors (such as molecular weight or surface area).

Usage

```
data(solubility)
```

Value

`solTrainX`	training set predictors in their natural units.
`solTrainXtrans`	training set predictors after transformations for skewness and centering/scaling.
`solTrainY`	a vector of log10 solubility values for the training set.
`solTestX`	test set predictors in their natural units.
`solTestXtrans`	test set predictors after the same transformations used on the training set are applied.
`solTestY`	a vector of log10 solubility values for the training set.

Source

Tetko, I., Tanchuk, V., Kasheva, T., and Villa, A. (2001). Estimation of aqueous solubility of chemical compounds using E-state indices. *Journal of Chemical Information and Computer Sciences*, 41(6), 1488-1493.

Huuskonen, J. (2000). Estimation of aqueous solubility for a diverse set of organic compounds based on molecular topology. *Journal of Chemical Information and Computer Sciences*, 40(3), 773-777.

Examples

```
data(solubility)

library(caret)

### Cross-validation splits used in the book:
set.seed(100)
indx <- createFolds(solTrainY, returnTrain = TRUE)
```

```
### To re-create the transformed version of the data:
## Not run:
## Find the predictors that are not fingerprints
contVars <- names(solTrainX)[!grepl("FP", names(solTrainX))]
## Some have zero values, so we need to add one to them so that
## we can use the Box-Cox transformation. Alternatively, we could
## use the Yeo-Johnson transformation without altering the data.
contPredTrain <- solTrainX[,contVars] + 1
contPredTest  <-  solTestX[,contVars] + 1

pp <- preProcess(contPredTrain, method = "BoxCox")
contPredTrain <- predict(pp, contPredTrain)
contPredTest  <- predict(pp, contPredTest)

## Reassemble the fingerprint data with the transformed values.
trainXtrans <- cbind(solTrainX[,grep("FP", names(solTrainX))], contPredTrain)
testXtrans  <- cbind( solTestX[,grep("FP", names(solTestX))],  contPredTest)

all.equal(trainXtrans, solTrainXtrans)
all.equal(testXtrans, solTestXtrans)

## End(Not run)
```

`twoClassData` *Two Class Example Data*

Description

These data contain two predictors measured for 208 samples. Of these, 111 samples are labeled as `Class1` and the remaining 97 are `Class2`.

Usage

```
data(twoClassData)
```

Value

`predictors`	data frame of two predictors
`classes`	a factor vector of class labeled

Examples

```
data(twoClassData)

library(lattice)
xyplot(PredictorB ~ PredictorA,
       data = predictors,
       groups = classes,
       auto.key = TRUE)
```

Index