Package ‘ipsRdbs’

April 10, 2024

Type Package

Title Introduction to Probability, Statistics and R for Data-Based Sciences

Version 1.0.0

Date 2024-04-10

Description Contains data sets, programmes and illustrations discussed in the book, ``Introduction to Probability, Statistics and R: Foundations for Data-Based Sciences.'' Sahu (2024, isbn:9783031378645) describes the methods in detail.

License GPL-3

Maintainer Sujit K. Sahu <`S.K.Sahu@soton.ac.uk`>

URL `https://www.sujitsahu.com`

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports methods, Rdpack, utils, ggplot2, extraDistr

Depends R (>= 4.1.0)

Suggests xtable, GGally, magick, MASS, knitr, rmarkdown, testthat, tidyr, huxtable, RColorBrewer, markdown, BiocStyle

VignetteBuilder knitr

LinkingTo

RdMacros Rdpack

BugReports `https://github.com/sujit-sahu/ipsRdbs/issues`

NeedsCompilation no

Author Sujit K. Sahu [aut, cre] (<`https://orcid.org/0000-0003-2315-3598`>)

Repository CRAN

Date/Publication 2024-04-10 17:20:03 UTC

R topics documented:

`beanie` *Age and value of 50 beanie baby toys*

Description

Age and value of 50 beanie baby toys

Usage

```
beanie
```

Format

A data frame with 50 rows and 3 columns:

name Name of the toy

age Age of the toy in months

value Market value of the toy in US dollars

Source

Beanie world magazine

Examples

```
head(beanie)
summary(beanie)
plot(beanie$age, beanie$value, xlab="Age", ylab="Value", pch="*", col="red")
```

`bill` *Wealth, age and region of 225 billionaires in 1992 as reported in the Fortune magazine*

Description

Wealth, age and region of 225 billionaires in 1992 as reported in the Fortune magazine

Usage

```
bill
```

Format

A data frame with 225 rows and three columns:

wealth Wealth in billions of US dollars

age Age of the billionaire

region five regions:Asia, Europe, Middle East, United States, and Other

Source

Fortune magazine 1992.

Examples

```
head(bill)
summary(bill)
table(bill$region)
levels(bill$region)
levels(bill$region) <- c("Asia", "Europe", "Mid-East", "Other", "USA")
bill.wealth.ge5 <- bill[bill$wealth>5, ]
bill.wealth.ge5
bill.region.A <-  bill[ bill$region == "A", ]
bill.region.A
a <-  seq(1, 10, by =2)
oddrows <- bill[a, ]
barplot(table(bill$region), col=2:6)
hist(bill$wealth) # produces a dull looking plot
hist(bill$wealth, nclass=20)  # produces a more detailed plot.
hist(bill$wealth, nclass=20, xlab="Wealth",
main="Histogram of wealth of billionaires")
# produces a more informative plot.
```

```
plot(bill$age, bill$wealth)  # A very dull plot.
plot(bill$age, bill$wealth, xlab="Age", ylab="Wealth", pch="*")  # better
plot(bill$age, bill$wealth, xlab="Age", ylab="Wealth", type="n")
# Lays the plot area but does not plot.
text(bill$age, bill$wealth, labels=bill$region, cex=0.7, col=2:6)
# Adds the points to the empty plot.
# Provides a better looking plot with more information.
boxplot(data=bill, wealth ~ region, col=2:6)
tapply(X=bill$wealth, INDEX=bill$region, FUN=mean)
tapply(X=bill$wealth, INDEX=bill$region, FUN=sd)
round(tapply(X=bill$wealth, INDEX=bill$region, FUN=mean), 2)
library(ggplot2)
gg <- ggplot2::ggplot(data=bill, aes(x=age, y=wealth)) +
geom_point(aes(col=region, size=wealth)) +
geom_smooth(method="loess", se=FALSE) +
xlim(c(7, 102)) +
ylim(c(1, 37)) +
labs(subtitle="Wealth vs Age of Billionaires",
y="Wealth (Billion US $)", x="Age",
title="Scatterplot", caption = "Source: Fortune Magazine, 1992.")
plot(gg)
```

`bodyfat` *Body fat percentage data for 102 elite male athletes training at the Australian Institute of Sport.*

Description

Body fat percentage data for 102 elite male athletes training at the Australian Institute of Sport.

Usage

`bodyfat`

Format

A data frame with 102 rows and two columns:

Skinfold Skin-fold thicknesses measured using calipers

Bodyfat Percentage of fat content in the body

Source

Data collected by Dr R. Telford who was working for the Australian Institute of Sport (AIS)

Examples

```
 summary(bodyfat)
plot(bodyfat$Skinfold,  bodyfat$Bodyfat, xlab="Skin", ylab="Fat")
plot(bodyfat$Skinfold,  log(bodyfat$Bodyfat), xlab="Skin", ylab="log Fat")
plot(log(bodyfat$Skinfold),  log(bodyfat$Bodyfat), xlab="log Skin", ylab="log Fat")
old.par <- par(no.readonly = TRUE)
# par(mfrow=c(2,2)) # draws four plots in a graph
plot(bodyfat$Skinfold,  bodyfat$Bodyfat, xlab="Skin", ylab="Fat")
plot(bodyfat$Skinfold,  log(bodyfat$Bodyfat), xlab="Skin", ylab="log Fat")
plot(log(bodyfat$Skinfold),  log(bodyfat$Bodyfat), xlab="log Skin", ylab="log Fat")
plot(1:5, 1:5, axes=FALSE, xlab="", ylab="",  type="n")
text(2, 2, "Log both X and Y")
text(2, 1, "To have the best plot")
# Keep the transformed variables in the data set
bodyfat$logskin <- log(bodyfat$Skinfold)
bodyfat$logbfat <- log(bodyfat$Bodyfat)
bodyfat$logskin <- log(bodyfat$Skinfold)
par(old.par)
 # Create a grouped variable
bodyfat$cutskin <- cut(log(bodyfat$Skinfold), breaks=6)
boxplot(data=bodyfat, Bodyfat~cutskin, col=2:7)
head(bodyfat)
require(ggplot2)
p2 <- ggplot(data=bodyfat, aes(x=cutskin, y=logbfat)) +
geom_boxplot(col=2:7) +
stat_summary(fun=mean, geom="line", aes(group=1), col="blue", linewidth=1) +
labs(x="Skinfold", y="Percentage of log bodyfat",
title="Boxplot of log-bodyfat percentage vs grouped log-skinfold")
plot(p2)

n <- nrow(bodyfat)
x <- bodyfat$logskin
y <- bodyfat$logbfat
xbar <- mean(x)
ybar <- mean(y)
sx2 <- var(x)
sy2 <- var(y)
sxy <- cov(x, y)
r <- cor(x, y)
print(list(n=n, xbar=xbar, ybar=ybar, sx2=sx2, sy2=sy2, sxy=sxy, r=r))
hatbeta1 <- r * sqrt(sy2/sx2) # calculates estimate of the slope
hatbeta0 <- ybar - hatbeta1 * xbar # calculates estimate of the intercept
rs <-  y - hatbeta0 - hatbeta1 * x  # calculates residuals
s2 <- sum(rs^2)/(n-2)  # calculates estimate of sigma2
s2
bfat.lm <- lm(logbfat ~ logskin, data=bodyfat)
### Check the diagnostics
plot(bfat.lm$fit, bfat.lm$res, xlab="Fitted values", ylab = "Residuals")
abline(h=0)
### Should be a random scatter
qqnorm(bfat.lm$res, col=2)
qqline(bfat.lm$res, col="blue")
```

```
# All Points should be on the straight line
summary(bfat.lm)
anova(bfat.lm)
plot(bodyfat$logskin,  bodyfat$logbfat, xlab="log Skin", ylab="log Fat")
abline(bfat.lm, col=7)
title("Scatter plot with the fitted Linear Regression line")
# 95% CI for beta(1)
# 0.88225 + c(-1, 1) * qt(0.975, df=100) *  0.02479
# round(0.88225 + c(-1, 1) * qt(0.975, df=100) *  0.02479, 2)
# To test H0: beta1 = 1.
tstat <- (0.88225 -1)/0.02479
pval <- 2 * (1- pt(abs(tstat), df=100))
x <- seq(from=-5, to=5, length=500)
y <- dt(x, df=100)
plot(x, y,  xlab="", ylab="", type="l")
title("T-density with df=100")
abline(v=abs(tstat))
abline(h=0)
x1 <- seq(from=abs(tstat), to=10, length=100)
y1 <- rep(0, length=100)
x2 <- x1
y2 <- dt(x1, df=100)
segments(x1, y1, x2, y2)
abline(h=0)
# Predict at a new value of Skinfold=70
# Create a new data set called new
newx <- data.frame(logskin=log(70))
a <- predict(bfat.lm, newdata=newx, se.fit=TRUE)
# Confidence interval for the mean of log bodyfat  at skinfold=70
a <- predict(bfat.lm, newdata=newx, interval="confidence")
# a
#          fit      lwr     upr
# [1,] 2.498339 2.474198 2.52248
# Prediction interval for a future log bodyfat  at skinfold=70
a <- predict(bfat.lm, newdata=newx, interval="prediction")
a
#          fit      lwr      upr
# [1,] 2.498339 2.333783 2.662895
#prediction intervals for the mean
pred.bfat.clim <- predict(bfat.lm, data=bodyfat, interval="confidence")
#prediction intervals for future observation
pred.bfat.plim <- suppressWarnings(predict(bfat.lm, data=bodyfat, interval="prediction"))
plot(bodyfat$logskin,  bodyfat$logbfat, xlab="log Skin", ylab="log Fat")
abline(bfat.lm, col=5)
lines(log(bodyfat$Skinfold), pred.bfat.clim[,2], lty=2, col=2)
lines(log(bodyfat$Skinfold), pred.bfat.clim[,3], lty=2, col=2)
lines(log(bodyfat$Skinfold), pred.bfat.plim[,2], lty=4, col=3)
lines(log(bodyfat$Skinfold), pred.bfat.plim[,3], lty=4, col=3)
title("Scatter plot with the fitted line and prediction intervals")
symb <- c("Fitted line", "95% CI for mean", "95% CI for observation")
## legend(locator(1), legend = symb, lty = c(1, 2, 4), col = c(5, 2, 3))
# Shows where we predicted earlier
```

```
abline(v=log(70))
summary(bfat.lm)
anova(bfat.lm)
```

`bombhits` *Number of bomb hits in London during World War II*

Description

Number of bomb hits in London during World War II

Usage

```
bombhits
```

Format

A data frame with two columns and six rows:

numberhit The number of bomb hits during World War II in each of the 576 areas in London.

freq Frequency of the number of hits

Source

Shaw and Shaw (2019).

Examples

```
summary(bombhits)
# Create a vector of data
x <- c(rep(0, 229), rep(1, 211), rep(2, 93), rep(3, 35), rep(4, 7), 5)
y <- c(229, 211, 93, 35, 7, 1) # Frequencies
rel_freq <- y/576
xbar <- mean(x)
pois_prob <- dpois(x=0:5, lambda=xbar)
fit_freq <- pois_prob * 576
 #Check
 cbind(x=0:5, obs_freq=y, rel_freq =round(rel_freq, 4),
 Poisson_prob=round(pois_prob, 4), fit_freq=round(fit_freq, 1))
 obs_freq <- y
 xuniques <- 0:5
 a <- data.frame(xuniques=0:5, obs_freq =y, fit_freq=fit_freq)
 barplot(rbind(obs_freq, fit_freq),
 args.legend = list(x = "topright"),
 xlab="No of bomb hits",
 names.arg = xuniques,  beside=TRUE,
 col=c("darkblue","red"),
 legend =c("Observed", "Fitted"),
 main="Observed and Poisson distribution fitted frequencies
 for the number of bomb hits in London")
```

`butterfly` *Draws a butterfly as on the front cover of the book*

Description

Draws a butterfly as on the front cover of the book

Usage

```
butterfly(color = 2, a = 2, b = 4)
```

Arguments

`color`	This is the color to use in the plot. It can take any value that R can use for color, e.g. 1, 2, "blue" etc.
`a`	Parameter controlling the shape of the butterfly
`b`	Second parameter controlling the shape of the butterfly

Value

No return value, called for side effects. It generates a plot whose colour and shape are determined by the supplied parameters.

Examples

```
butterfly(color = 6)
old.par <- par(no.readonly = TRUE)
par(mfrow=c(2, 2))
butterfly(color = 6)
butterfly(a=5, b=5, color=2)
butterfly(a=10, b=1.5, color = "seagreen")
butterfly(a=20, b=4, color = "blue")
par(old.par) # par(mfrow=c(1, 1))
```

`cement` *Breaking strength of cement data*

Description

Breaking strength of cement data

Usage

```
cement
```

Format

A data frame with 36 rows and 3 columns:

strength	Breaking strength in pounds per square inch
gauger	Three different gauger machines which mixes cement with water
breaker	Three different breakers breaking the cement cubes

Examples

```
summary(cement)
```

`cfail` *Weekly number of failures of a university computer system over a period of two years. This is a data vector containing 104 values.*

Description

Weekly number of failures of a university computer system over a period of two years. This is a data vector containing 104 values.

Usage

```
cfail
```

Format

An object of class `numeric` of length 104.

Examples

```
summary(cfail)
# 95% Confidence interval
c(3.75-1.96 * 3.381/sqrt(104), 3.75+1.96*3.381/sqrt(104)) # =(3.10,4.40).
x <- cfail
n <- length(x)
h <- qnorm(0.975)
# 95% Confidence interval Using quadratic inversion
mean(x) + (h*h)/(2*n) + c(-1, 1) * h/sqrt(n) * sqrt(h*h/(4*n) + mean(x))
# Modelling
# Observed frequencies
obs_freq <- as.vector(table(x))
# Obtain unique x values
xuniques <- sort(unique(x))
lam_hat <- mean(x)
fit_freq <- n * dpois(xuniques, lambda=lam_hat)
fit_freq <- round(fit_freq, 1)
# Create a data frame for plotting
a <- data.frame(xuniques=xuniques, obs_freq = obs_freq, fit_freq=fit_freq)
```

```
barplot(rbind(obs_freq, fit_freq), args.legend = list(x = "topright"),
xlab="No of weekly computer failures",
names.arg = xuniques,  beside=TRUE, col=c("darkblue","red"),
legend =c("Observed", "Fitted"),
main="Observed and Poisson distribution fitted frequencies
for the computer failure data: cfail")
```

`cheese` *Testing of cheese data set*

Description

Testing of cheese data set

Usage

```
cheese
```

Format

A data frame with 30 rows and 5 columns

Taste A measure of taste quality of cheese

AceticAcid Concentration of Acetic acid

H2S Concentration of hydrogen sulphide

LacticAcid Concentration lactic acid

logH2S Logarithm of H2S

Examples

```
data(cheese)
summary(cheese)
pairs(cheese)
cheese.lm <- lm(Taste ~ AceticAcid +  LacticAcid + logH2S, data=cheese, subset=2:30)
# Check the diagnostics
plot(cheese.lm$fit, cheese.lm$res, xlab="Fitted values", ylab = "Residuals")
abline(h=0)
# Should be a random scatter
qqnorm(cheese.lm$res, col=2)
qqline(cheese.lm$res, col="blue")
summary(cheese.lm)
cheese.lm2 <- lm(Taste ~ LacticAcid + logH2S, data=cheese)
# Check the diagnostics
plot(cheese.lm2$fit, cheese.lm2$res, xlab="Fitted values", ylab = "Residuals")
abline(h=0)
qqnorm(cheese.lm2$res, col=2)
qqline(cheese.lm2$res, col="blue")
summary(cheese.lm2)
```

```
# How can we predict?
newcheese <- data.frame(AceticAcid = 300, LacticAcid = 1.5, logH2S=4)
cheese.pred <- predict(cheese.lm2, newdata=newcheese, se.fit=TRUE)
cheese.pred
# Obtain confidence interval
cheese.pred$fit + c(-1, 1) * qt(0.975, df=27) * cheese.pred$se.fit
# Using R to predict
cheese.pred.conf.limits <- predict(cheese.lm2, newdata=newcheese, interval="confidence")
cheese.pred.conf.limits
# How to find prediction interval
cheese.pred.pred.limits <- predict(cheese.lm2, newdata=newcheese, interval="prediction")
cheese.pred.pred.limits
```

`emissions` *Nitrous oxide emission data*

Description

Nitrous oxide emission data

Usage

`emissions`

Format

An object of class `data.frame` with 54 rows and 13 columns.

Source

Australian Traffic Accident Research Bureau @format A data frame with thirteen columns and 54 rows.

Make Make of the car

Odometer Odometer reading of the car

Capacity Engine capacity of the car

CS505 A measurement taken while running the engine from a cold start for 505 seconds

T867 A measurement taken while running the engine in transition from cold to hot for 867 seconds

H505 A measurement taken while running the hot engine for 505 seconds

ADR27 A previously used measurement standard

ADR37 Result of the Australian standard ADR37test

logCS505 Logarithm of CS505

logT867 Logarithm of T867

logH505 Logarithm of H505

logADR27 Logarithm of ADR27

logADR37 Logarithm of ADR37

Examples

```
 summary(emissions)

 rawdata <- emissions[, c(8, 4:7)]
 pairs(rawdata)
# Fit the model on the raw scale
raw.lm <- lm(ADR37 ~ ADR27 + CS505  + T867 + H505, data=rawdata)
old.par <- par(no.readonly = TRUE)
par(mfrow=c(2,1))
plot(raw.lm$fit, raw.lm$res,xlab=”Fitted values”,ylab=”Residuals”, main=”Anscombe plot”)
abline(h=0)
qqnorm(raw.lm$res,main=”Normal probability plot”, col=2)
qqline(raw.lm$res, col=”blue”)
# summary(raw.lm)
logdata <- log(rawdata)
# This only logs the values but not the column names!
# We can use the following command to change the column names or you can use
# fix(logdata) to do it.
dimnames(logdata)[[2]] <- c(”logADR37”, ”logCS505”, ”logT867”, ”logH505”, ”logADR27”)
pairs(logdata)
log.lm <- lm(logADR37 ~ logADR27 + logCS505  + logT867 + logH505, data=logdata)
plot(log.lm$fit, log.lm$res,xlab=”Fitted values”,ylab=”Residuals”, main=”Anscombe plot”)
abline(h=0)
qqnorm(log.lm$res,main=”Normal probability plot”, col=2)
qqline(log.lm$res, col=”blue”)
summary(log.lm)
log.lm2 <- lm(logADR37 ~ logADR27 + logT867 + logH505, data=logdata)
summary(log.lm2)
plot(log.lm2$fit, log.lm2$res,xlab=”Fitted values”,ylab=”Residuals”, main=”Anscombe plot”)
abline(h=0)
qqnorm(log.lm2$res,main=”Normal probability plot”, col=2)
qqline(log.lm2$res, col=”blue”)
par(old.par)
#####################################
# Multicollinearity Analysis
######################################
mod.adr27 <-  lm(logADR27 ~ logT867 + logCS505 + logH505, data=logdata)
summary(mod.adr27) # Multiple R^2 = 0.9936,
mod.t867 <-  lm(logT867 ~ logADR27 + logH505 + logCS505, data=logdata)
summary(mod.t867) # Multiple R^2 = 0.977,
mod.cs505 <-  lm(logCS505 ~ logADR27 + logH505 + logT867, data=logdata)
summary(mod.cs505) # Multiple R^2 = 0.9837,
mod.h505 <-  lm(logH505 ~ logADR27 + logCS505 + logT867, data=logdata)
summary(mod.h505) # Multiple R^2 = 0.5784,
# Variance inflation factors
vifs <- c(0.9936, 0.977, 0.9837, 0.5784)
vifs <- 1/(1-vifs)
#Condition numbers
X <- logdata
# X is a copy of logdata
X[,1] <- 1
# the first column of X is 1
```

```
# this is for the intercept
X <- as.matrix(X)
# Coerces X to be a matrix
xtx <- t(X) %*% X # Gives X^T X
eigenvalues <- eigen(xtx)$values
kappa <- max(eigenvalues)/min(eigenvalues)
kappa <- sqrt(kappa)
# kappa = 244 is much LARGER than 30!

### Validation statistic
# Fit the log.lm2 model with the first 45 observations
# use the fitted model to predict the remaining 9 observations
# Calculate the mean square error validation statistic
log.lmsub <- lm(logADR37 ~ logADR27 + logT867 + logH505, data=logdata, subset=1:45)
# Now predict all 54 observations using the fitted model
mod.pred <- predict(log.lmsub, logdata, se.fit=TRUE)
mod.pred$fit # provides all the 54 predicted values
logdata$pred <- mod.pred$fit
# Get only last 9
a <- logdata[46:54, ]
validation.residuals <- a$logADR37 - a$pred
validation.stat <- mean(validation.residuals^2)
validation.stat
```

`err_age` *Errors in guessing ages of Southampton mathematicians*

Description

Errors in guessing ages of Southampton mathematicians

Usage

```
err_age
```

Format

A data frame with 550 rows and 10 columns

group Group number of the students guessing the ages
size Number of students in the group
females How many female guessers were in the group
photo Photograph number guessed, can take value 1 to 10.
sex Gender of the photographed person.
race Race of the photographed person.
est_age Estimated age of the photographed person.
tru_age True age of the photographed person.
error The value of error, estimated age minus true age
abs_error Absolute value of the error

Examples

```
summary(err_age)
```

`ffood` *Service (waiting) times (in seconds) of customers at a fast-food restaurant.*

Description

Service (waiting) times (in seconds) of customers at a fast-food restaurant.

Usage

`ffood`

Format

A data frame with 10 rows and 2 columns:

AM Waiting times for customers served during 9-10AM

PM Waiting times for customers served during 2-3PM

Examples

```
summary(ffood)
# 95% Confidence interval for the mean waiting time usig t-distribution
a <- c(ffood$AM, ffood$PM)
mean(a) + c(-1, 1) * qt(0.975, df=19) * sqrt(var(a))/sqrt(20)
# Two sample t-test for the difference between morning and afternoon times
t.test(ffood$AM, ffood$PM)
```

`gasmileage` *Gas mileage of four models of car*

Description

Gas mileage of four models of car

Usage

`gasmileage`

Format

A data frame with two columns and eleven rows:

mileage Mileage obtained

model Four types of car models

Examples

```
summary(gasmileage)
y <- c(22, 26,  28, 24, 29,   29, 32, 28,  23, 24)
xx <- c(1,1,2,2,2,3,3,3,4,4)
# Plot the observations
plot(xx, y, col="red", pch="*", xlab="Model", ylab="Mileage")
# Method1: Hand calculation
ni <- c(2, 3, 3, 2)
means <- tapply(y, xx, mean)
vars <- tapply(y, xx, var)
round(rbind(means, vars), 2)
sum(y^2) # gives 7115
totalSS <- sum(y^2) - 10 * (mean(y))^2 # gives 92.5
RSSf <- sum(vars*(ni-1)) # gives 31.17
groupSS <- totalSS - RSSf # gives 61.3331.17/6
meangroupSS <- groupSS/3 # gives 20.44
meanErrorSS <- RSSf/6 # gives 5.194
Fvalue <- meangroupSS/meanErrorSS # gives 3.936
pvalue <- 1-pf(Fvalue, df1=3, df2=6)

#### Method 2: Illustrate using dummy variables
#################################
#Create the design matrix X for the full regression model
g <- 4
n1 <- 2
n2 <- 3
n3 <- 3
n4 <- 2
n <- n1+n2+n3+n4
X <- matrix(0, ncol=g, nrow=n)       #Set X as a zero matrix initially
X[1:n1,1] <- 1    #Determine the first column of X
X[(n1+1):(n1+n2),2] <- 1   #the 2nd column
X[(n1+n2+1):(n1+n2+n3),3] <- 1    #the 3rd
X[(n1+n2+n3+1):(n1+n2+n3+n4),4] <- 1    #the 4th
#################################
####Fitting the  full model####
#################################
#Estimation
XtXinv <- solve(t(X)%*%X)
betahat <- XtXinv %*%t(X)%*%y   #Estimation of the coefficients
Yhat <- X%*%betahat   #Fitted Y values
Resids <- y - Yhat   #Residuals
SSE <- sum(Resids^2)   #Error sum of squares
S2hat <- SSE/(n-g)   #Estimation of sigma-square; mean square for error
Sigmahat <- sqrt(S2hat)

##############################################################
####Fitting the reduced model -- the 4 means are equal #####
##############################################################
Xr <- matrix(1, ncol=1, nrow=n)
kr <- dim(Xr)[2]
#Estimation
```

```
Varr <- solve(t(Xr)%*%Xr)
hbetar <- solve(t(Xr)%*%Xr)%*%t(Xr)%*% y   #Estimation of the coefficients
hYr = Xr%*%hbetar   #Fitted Y values
Resir <- y - hYr   #Residuals
SSEr <- sum(Resir^2)   #Total sum of squares
###################################################################
####F-test for comparing the reduced model with the full model ####
###################################################################
FStat <- ((SSEr-SSE)/(g-kr))/(SSE/(n-g))  #The test statistic of the F-test
alpha <- 0.05
Critical_value_F <- qf(1-alpha, g-kr,n-g)  #The critical constant of F-test
pvalue_F <- 1-pf(FStat,g-kr, n-g)   #p-value of F-test

modelA <- c(22, 26)
modelB <- c(28, 24, 29)
modelC <- c(29, 32, 28)
modelD <- c(23, 24)

SSerror = sum( (modelA-mean(modelA))^2 ) + sum( (modelB-mean(modelB))^2 )
+ sum( (modelC-mean(modelC))^2 ) + sum( (modelD-mean(modelD))^2 )
SStotal <-  sum( (y-mean(y))^2 )
SSgroup <- SStotal-SSerror

####
#### Method 3: Use the  built-in function lm directly

#####################################
aa <- "modelA"
bb <- "modelB"
cc <- "modelC"
dd <- "modelD"
Expl <- c(aa,aa,bb,bb,bb,cc,cc,cc,dd,dd)
is.factor(Expl)
Expl <- factor(Expl)
model1 <- lm(y~Expl)
summary(model1)
anova(model1)
###Alternatively ###

xxf <- factor(xx)
is.factor(xxf)
model2 <- lm(y~xxf)
summary(model2)
anova(model2)
```

`monty` *Simulation of the Monty Hall Problem. This demonstrates that switching is always better than staying with the initial guess. Programme written by Corey Chivers, 2012*

Description

Simulation of the Monty Hall Problem. This demonstrates that switching is always better than staying with the initial guess. Programme written by Corey Chivers, 2012

Usage

```
monty(strat = "stay", N = 1000, print_games = TRUE)
```

Arguments

`strat`	Strategy to use; possibilities are: • "stay": Do not change the initial door chosen. • "swap": Swap the door chosen initially. • "random": Randomly decide to stay or swap.
`N`	How many games to play, defaults to 1000.
`print_games`	Logical; whether to print the results of each game.

Value

No return value, called for side effects. If the supplied parameter `print_games` is TRUE, then it prints out the result (Win or Loss) of each of the N simulated games. Finally it reports the overall percentage of winning.

###

Source

Corey Chivers (2012) Chivers (2012)

Examples

```
# example code
monty("stay")
monty("switch")
monty("random")
```

`possum` *Body weight and length of possums (tree living furry animals who are mostly nocturnal (marsupial) caught in 7 different regions of Australia.*

Description

Body weight and length of possums (tree living furry animals who are mostly nocturnal (marsupial) caught in 7 different regions of Australia.

Usage

```
possum
```

Format

A data frame with 101 rows and 3 columns:

Body_Weight Body weight in kilogram

Length Body length of the possum

Location 7 different regions of Australia: (1) Western Austrailia (W.A), (2) South Australia (S.A), (3) Northern Territory (N.T), (4) Queensland (QuL), (5) New South Wales (NSW), (6) Victoria (Vic) and (7) Tasmania (Tas).

Source

Lindenmayer and Donnelly (1995).

References

Lindenmayer DBVKLCRB, Donnelly CF (1995). "Morphological variation among columns of the mountain brushtail possum, Trichosurus caninus Ogilby (Phalangeridae: Marsupiala)." *Australian Journal of Zoology*, **43**, 449-458.

Examples

```
head(possum)
 dim(possum)
 summary(possum)
 ## Code lines used in the book
 ## Create a new data set
 poss <- possum
 poss$region <- factor(poss$Location)
 levels(poss$region) <- c("W.A", "S.A", "N.T", "QuL", "NSW", "Vic", "Tas")
 colnames(poss)<-c("y","z","Location", "x")
 head(poss)
 # Draw side by side boxplots
 boxplot(y~x, data=poss, col=2:8, xlab="region", ylab="Weight")
 # Obtain scatter plot
 # Start with a skeleton plot
 plot(poss$z, poss$y, type="n", xlab="Length", ylab="Weight")
 # Add points for the seven regions
 for (i in 1:7) {
   points(poss$z[poss$Location==i],poss$y[poss$Location==i],type="p", pch=as.character(i), col=i)
    }
## Add legends
 legend(x=76, y=4.2, legend=paste(as.character(1:7), levels(poss$x)),  lty=1:7, col=1:7)
 # Start  modelling
 #Fit the model with interaction.
 poss.lm1<-lm(y~z+x+z:x,data=poss)
 summary(poss.lm1)
 plot(poss$z, poss$y,type="n", xlab="Length", ylab="Weight")
 for (i in 1:7) {
 lines(poss$z[poss$Location==i],poss.lm1$fit[poss$Location==i],type="l",
 lty=i, col=i, lwd=1.8)
 points(poss$z[poss$Location==i],poss$y[poss$Location==i],type="p",
```

```
pch=as.character(i), col=i)
}
poss.lm0 <- lm(y~z,data=poss)
abline(poss.lm0, lwd=3, col=9)
# Has drawn the seven parallel regression lines
legend(x=76, y=4.2, legend=paste(as.character(1:7), levels(poss$x)),
lty=1:7, col=1:7)

n <- length(possum$Body_Weight)
# Wrong model since Location is not a numeric covariate
wrong.lm <- lm(Body_Weight~Location, data=possum)
summary(wrong.lm)

nis <- table(possum$Location)
meanwts <- tapply(possum$Body_Weight, possum$Location, mean)
varwts <- tapply(possum$Body_Weight, possum$Location, var)
datasums <- data.frame(nis=nis, mean=meanwts, var=varwts)
datasums <- data.frame(nis=nis, mean=meanwts, var=varwts)
modelss <- sum(datasums[,2] * (meanwts - mean(meanwts))^2)
residss <- sum( (datasums[,2] - 1) * varwts)

fvalue <- (modelss/6) / (residss/94)
fcritical <- qf(0.95, df1= 6, df2=94)
x <- seq(from=0, to=12, length=200)
y <- df(x, df1=6, df2=94)
plot(x, y, type="l", xlab="", ylab="Density of F(6, 94)", col=4)
abline(v=fcritical, lty=3, col=3)
abline(v=fvalue, lty=2, col=2)
pvalue <- 1-pf(fvalue, df1=6, df2=94)

### Doing the above in R
# Convert  the Location column to a factor
possum$Location <- as.factor(possum$Location)
summary(possum)  # Now Location is a factor

# Put the identifiability constraint:
options(contrasts=c("contr.treatment", "contr.poly"))
colnames(possum) <- c("y", "z", "x")
# Fit model M1
possum.lm1 <- lm(y~x, data=possum)
summary(possum.lm1)
anova(possum.lm1)
possum.lm2 <- lm(y~z, data=poss)
summary(possum.lm2)
anova(possum.lm2)
# Include both location and length but no interaction
possum.lm3 <-  lm(y~x+z, data=poss)
summary(possum.lm3)
anova(possum.lm3)
# Include interaction effect
possum.lm4 <-  lm(y~x+z+x:z, data=poss)
summary(possum.lm4)
anova(possum.lm4)
```

```
anova(possum.lm2, possum.lm3)
#Check the diagnostics for M3
plot(possum.lm3$fit, possum.lm3$res,xlab="Fitted values",ylab="Residuals",
main="Anscombe plot")
abline(h=0)
qqnorm(possum.lm3$res,main="Normal probability plot", col=2)
qqline(possum.lm3$res, col="blue")
```

`puffin` *Puffin nesting data set. It contains data regarding nesting habits of common puffin*

Description

Puffin nesting data set. It contains data regarding nesting habits of common puffin

Usage

```
puffin
```

Format

A data frame with 38 rows and 5 columns:

Nesting_Frequency Number of nests

Grass_Cover Percentage of area covered by grass

Mean_Soil_Depth Mean soil depth in centimeter

Slope_Angle Slope angle in degrees

Distance_from_Edge Distance of the plot from the cliff edge in meter

Source

Nettleship (1972).

References

Nettleship DN (1972). “Breeding Success of the Common Puffin (Fratercula arctica L.) on Different Habitats at Great Island, Newfoundland.” *Ecological Monographs*, **42**, 239-268.

Examples

```
head(puffin)
dim(puffin)
summary(puffin)
pairs(puffin)
puffin$sqrtfreq <- sqrt(puffin$Nesting_Frequency)
puff.sqlm <- lm(sqrtfreq~ Grass_Cover + Mean_Soil_Depth + Slope_Angle
```

```
+Distance_from_Edge, data=puffin)
old.par <- par(no.readonly = TRUE)
par(mfrow=c(2,1))
qqnorm(puff.sqlm$res,main="Normal probability plot", col=2)
qqline(puff.sqlm$res, col="blue")
plot(puff.sqlm$fit, puff.sqlm$res,xlab="Fitted values",ylab="Residuals",
main="Anscombe plot")
abline(h=0)
summary(puff.sqlm)
par(old.par)
#####################################
# F test for two betas at the  same time:
######################################
puff.sqlm2 <- lm(sqrtfreq~ Mean_Soil_Depth + Distance_from_Edge, data=puffin)
anova(puff.sqlm)
anova(puff.sqlm2)
fval <-  1/2*(14.245-12.756)/0.387 # 1.924
qf(0.95, 2, 33) # 3.28
1-pf(fval, 2, 33) # 0.162
anova(puff.sqlm2, puff.sqlm)
```

`rice` *Riece yield data*

Description

Riece yield data

Usage

`rice`

Format

A data frame with three columns and 68 rows:

Yield Yield of rice in kilograms

Days Number of days after flowering before harvesting

Source

Bal and Ojha (1975).

Examples

```
summary(rice)
plot(rice$Days, rice$Yield, pch="*", xlab="Days", ylab="Yield")
rice$daymin31 <- rice$Days-31
rice.lm <- lm(Yield ~ daymin31, data=rice)
summary(rice.lm)
```

```
# Check the diagnostics
plot(rice.lm$fit, rice.lm$res, xlab="Fitted values", ylab = "Residuals")
abline(h=0)
# Should be a random scatter
# Needs a quadratic term

qqnorm(rice.lm$res, col=2)
qqline(rice.lm$res, col="blue")
rice.lm2 <- lm(Yield ~ daymin31 + I(daymin31^2) , data=rice)
old.par <- par(no.readonly = TRUE)
par(mfrow=c(1, 2))
plot(rice.lm2$fit, rice.lm2$res, xlab="Fitted values", ylab = "Residuals")
abline(h=0)
# Should be a random scatter
# Much better plot!
qqnorm(rice.lm2$res, col=2)
qqline(rice.lm2$res, col="blue")
summary(rice.lm2)
par(old.par) # par(mfrow=c(1,1))
plot(rice$Days,  rice$Yield, xlab="Days", ylab="Yield")
lines(rice$Days, rice.lm2$fit, lty=1, col=3)
rice.lm3 <- lm(Yield ~ daymin31 + I(daymin31^2)+I(daymin31^3) , data=rice)
#check the diagnostics
summary(rice.lm3) # Will print the summary of the fitted model
#### Predict at a new value of Days=31.1465

# Create a new data set called new
new <- data.frame(daymin31=32.1465-31)

a <- predict(rice.lm2, newdata=new, se.fit=TRUE)
# Confidence interval for the mean of rice yield  at day=31.1465
a <- predict(rice.lm2, newdata=new, interval="confidence")
a
#          fit      lwr      upr
# [1,] 3676.766 3511.904 3841.628
# Prediction interval for a future yield at day=31.1465
b <- predict(rice.lm2, newdata=new, interval="prediction")
b
# fit      lwr      upr
#[1,] 3676.766 3206.461 4147.071
```

`see_the_clt_for_Bernoulli`

Illustration of the CLT for samples from the Bernoulli distribution

---

Description

Illustration of the CLT for samples from the Bernoulli distribution

Usage

```
see_the_clt_for_Bernoulli(nsize = 10, nrep = 10000, prob = 0.8)
```

Arguments

`nsize`	Sample size, n. Its default value is 10.
`nrep`	Number of replications. How many samples of size `nsize` should be taken, default value is 10000.
`prob`	True probability of success for the Bernoulli trials

Value

A vector of means of the replicated samples. It also has the side effect of drawing a histogram of the standardized sample means and a superimposed density function of the standard normal distribution. The better the CLT approximation, the closer are the superimposed density and the histogram.

Examples

```
a <- see_the_clt_for_Bernoulli()
old.par <- par(no.readonly = TRUE)
par(mfrow=c(2, 3))
a30 <- see_the_clt_for_Bernoulli(nsize=30)
a50 <- see_the_clt_for_Bernoulli(nsize=50)
a100 <- see_the_clt_for_Bernoulli(nsize=100)
a500 <- see_the_clt_for_Bernoulli(nsize=500)
a1000 <- see_the_clt_for_Bernoulli(nsize=1000)
a5000 <- see_the_clt_for_Bernoulli(nsize=5000)
par(old.par)
```

`see_the_clt_for_uniform` *Illustration of the central limit theorem for sampling from the uniform distribution*

Description

Illustration of the central limit theorem for sampling from the uniform distribution

Usage

```
see_the_clt_for_uniform(nsize = 10, nrep = 10000)
```

Arguments

`nsize`	Sample size, n. Its default value is 10.
`nrep`	Number of replications. How many samples of size `nsize` should be taken, default value is 10000.

Value

A vector of means of the replicated samples. The function also has the side effect of drawing a histogram of the sample means and two superimposed density functions: one estimated from the data using the `density` function and the other is the density of the CLT approximated normal distribution. The better the CLT approximation, the closer are the two superimposed densities.

Examples

```
a <- see_the_clt_for_uniform()
old.par <- par(no.readonly = TRUE)
par(mfrow=c(2, 3))
a1 <- see_the_clt_for_uniform(nsize=1)
a2 <- see_the_clt_for_uniform(nsize=2)
a3 <- see_the_clt_for_uniform(nsize=5)
a4 <- see_the_clt_for_uniform(nsize=10)
a5 <- see_the_clt_for_uniform(nsize=20)
a6 <- see_the_clt_for_uniform(nsize=50)
par(old.par)
ybars <- see_the_clt_for_uniform(nsize=12)
zbars <- (ybars - mean(ybars))/sd(ybars)
k <- 100
u <- seq(from=min(zbars), to= max(zbars), length=k)
ecdf <-  rep(NA, k)
for(i in 1:k) ecdf[i] <- length(zbars[zbars<u[i]])/length(zbars)
tcdf <- pnorm(u)
plot(u, tcdf, type="l", col="red", lwd=4, xlab="", ylab="cdf")
lines(u, ecdf, lty=2, col="darkgreen", lwd=4)
symb <- c("cdf of sample means", "cdf of N(0, 1)")
legend(x=-3.5, y=0.4, legend = symb, lty = c(2, 1),
col = c("darkgreen","red"), bty="n")
```

`see_the_wlln_for_uniform` *Illustration of the weak law of large numbers for sampling from the uniform distribution*

Description

Illustration of the weak law of large numbers for sampling from the uniform distribution

Usage

```
see_the_wlln_for_uniform(nsize = 10, nrep = 1000)
```

Arguments

`nsize`	Sample size, n. Its default value is 10.
`nrep`	Number of replications. How many samples of size `nsize` should be taken, default value is 10000.

Value

A list giving the x values and the density estimates y, from the generated random samples. The function also draws the empirical density on the current graphics device.

Examples

```
a1 <- see_the_wlln_for_uniform(nsize=1, nrep=50000)
a2 <- see_the_wlln_for_uniform(nsize=10, nrep=50000)
a3 <- see_the_wlln_for_uniform(nsize=50, nrep=50000)
a4 <- see_the_wlln_for_uniform(nsize=100, nrep=50000)
plot(a4, type="l", lwd=2, ylim=range(c(a1$y, a2$y, a3$y, a4$y)), col=1,
lty=1, xlab="mean", ylab="density estimates")
lines(a3, type="l", lwd=2, col=2, lty=2)
lines(a2, type="l", lwd=2, col=3, lty=3)
lines(a1, type="l", lwd=2, col=4, lty=4)
symb <- c("n=1", "n=10", "n=50", "n=100")
legend(x=0.37, y=11.5, legend = symb, lty =4:1, col = 4:1)
```

`wgain` *Weight gain data from 68 first year students during their first 12 weeks in college*

Description

Weight gain data from 68 first year students during their first 12 weeks in college

Usage

```
wgain
```

Format

A data frame with three columns and 68 rows:

student Student number, 1 to 68.

initial Initial weight in kilogram

final Final weight in kilogram

Examples

```
summary(wgain)
plot(wgain$initial, wgain$final)
abline(0, 1, col="red")
plot(wgain$initial, wgain$final, xlab="Wt in Week 1",
ylab="Wt in Week 12", pch="*", las=1)
abline(0, 1, col="red")
title("A scatter plot of the weights in Week 12 against
the  weights in Week 1")
```

```
# 95% Confidence interval for mean weight gain
x <- wgain$final - wgain$initial
mean(x) + c(-1, 1) * qt(0.975, df=67) * sqrt(var(x)/68)
# t-test to test the mean difference equals 0
t.test(x)
```

Index