Package 'CategoricalDataAnalysis'

November 30, 2017

Type Package

Title Categorical Data Analysis

Version 1.0

Date 2017-11-30

Author Maham Niaz, Saqib Ali

Maintainer Who to complain to <yourfault@somewhere.net>

Description More about what it does (maybe more than one line)

License GPL (>= 2)

R topics documented:

`CategoricalDataAnalysis-package`
Categorical Data Analysis

Description

More about what it does (maybe more than one line)

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.
~~ An overview of how to use the package, including the most important functions ~~

Author(s)

Maham Niaz, Saqib Ali

Maintainer: Who to complain to <yourfault@somewhere.net>

References

~~ Literature or other references for background information ~~

See Also

~~ Optional links to other man pages, e.g. ~~ ~~ <pkg> ~~

---

catbarchart *Plot Barchart for Categorical Data*

Usage

```
catbarchart(x)
```

Arguments

x

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x)
{
    xcolumnnames <- colnames(x)
    responsecol <- ncol(x)
    plot_hist <- function(column, data, response) ggplot(data,
        aes(x = get(column), ..count..)) + geom_bar(aes(fill = get(response)),
        position = "dodge") + xlab(column) + scale_fill_discrete(name = response)
    myplots <- lapply(colnames(x), plot_hist, data = x, response = xcolumnnames[responsecol]
    myplots <- myplots[-length(myplots)]
    grid.arrange(grobs = myplots, ncol = 1)
  }
```

`chisq.indep` *testing for independence between two categorical variable*

Usage

```
chisq.indep(m, level = 0.05, digits = 4, print = TRUE)
```

Arguments

m
level
digits
print

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (m, level = 0.05, digits = 4, print = TRUE)
{
    r.sum <- rowSums(m)
    c.sum <- colSums(m)
    n <- sum(m)
    exp.ct <- outer(r.sum, c.sum, "*")/n
    res <- m - exp.ct
    p.res <- res/sqrt(exp.ct)
    X.sq <- sum(p.res^2)
    G.sq <- 2 * sum(m * (log(m) - log(exp.ct)))
    df <- (nrow(m) - 1) * (ncol(m) - 1)
    c.val <- qchisq(level, df = df, lower.tail = FALSE)
    est.se <- sqrt(exp.ct * outer((1 - r.sum/n), (1 - c.sum/n),
        "*"))
    s.res <- res/est.se
    if (print) {
        cat("Chi-squared test of independence\n")
        cat("  Level = ", level, ", df = ", df, ", critical value = ",
            round(c.val, digits), "\n", sep = "")
        cat("  X-squared = ", round(X.sq, digits), "\n", sep = "")
        cat("  G-squared = ", round(G.sq, digits), sep = "")
    }
    invisible(list(X.sq = X.sq, df = df, expected = exp.ct, pearson.res = p.res,
        std.res = s.res))
  }
```

continous2categorical *continous2categorical function.*

Description

continous2categorical function. This function takes a data frame of continous variables and converts to a data frame of categorical variables. The last variable is the response variable.

Usage

```
continous2categorical(x)
```

Arguments

```
x
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (x)
{
    numberoffactors <- ncol(x) - 1
    out <- data.frame(0, matrix(nrow = nrow(x), ncol = 1))
    for (i in 1:numberoffactors) {
        labs <- c("low", "low-medium", "medium", "medium-high",
            "high")
        vartemp <- cut(x[, i], breaks = 5, labels = labs)
        out[i] <- vartemp
    }
    i <- i + 1
    out[i] <- x[i]
    colnames(out) <- colnames(x)
    return(data.frame(out))
  }
```

count_mat *creating contingency matrix for categorical data analysis*

Usage

```
count_mat(df)
```

Arguments

```
df
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (df)
{
    df_dim <- dim(df)
    if (df_dim[2] == 2 && length(df_dim) == 2) {
        factor_df1 <- as.factor(df[, 1])
        factor_df2 <- as.factor(df[, 2])
        lev_col1 = levels(factor_df1)
        lev_col2 = levels(factor_df2)
        len_col1 = length(lev_col1)
        len_col2 = length(lev_col2)
        val = 1
        for (i in lev_col1) {
            for (j in lev_col2) {
                val = c(val, length(which(df[, 1] == i & df[,
                  2] == j)))
            }
        }
        out = matrix(val[-1], byrow = TRUE, nrow = length(lev_col1),
            dimnames = list(lev_col1, lev_col2))
    }
    else (out = "check dimension")
    return(out)
  }
```

`crabs` *contains the data analyzed by Brockmann (1996) and is discussed extensively in Agresti (2002). This is a space-delimited text file in which the variable names appear in the first row. Background*

Usage

```
data("crabs")
```

Format

A data frame with 174 observations on the following 5 variables.

`V1` a factor with levels `2` `3` `4` `5` `color`

`V2` a factor with levels `1` `2` `3` `spine`

V3 a factor with levels 21.0 22.0 22.5 22.9 23.0 23.1 23.2 23.4 23.5 23.7 23.8 23.9 24.0 24.1 24.2 24.3 24.5 24.7 24.8 24.9 25.0 25.1 25.2 25.3 25.4 25.5 25.6 25.7 25.8 25.9 26.0 26.1 26.2 26.3 26.5 26.7 26.8 27.0 27.1 27.2 27.3 27.4 27.5 27.6 27.7 27.8 27.9 28.0 28.2 28.3 28.4 28.5 28.7 28.9 29.0 29.3 29.5 29.7 29.8 30.0 30.2 30.3 30.5 31.7 31.9 33.5 width

V4 a factor with levels 0 1 10 11 12 14 15 2 3 4 5 6 7 8 9 num.satellites

V5 a factor with levels 1200 1300 1400 1475 1550 1600 1650 1700 1800 1850 1900 1950 1967 2000 2025 2050 2100 2150 2175 2200 2225 2250 2275 2300 2350 2400 2450 2500 2550 2600 2625 2650 2700 2750 2800 2850 2867 2900 2925 2950 3000 3025 3050 3100 3150 3200 3225 3250 3275 3300 3325 3500 3600 3725 3850 5200 weight

Examples

```
data(crabs)
## maybe str(crabs) ; plot(crabs) ...
```

---

crabs2 *contains the data analyzed by Brockmann (1996) and is discussed extensively in Agresti (2002). This is a space-delimited text file in which the variable names appear in the first row. Background*

Usage

```
data("crabs2")
```

Format

A data frame with 173 observations on the following 5 variables.

color a numeric vector

spine a numeric vector

width a numeric vector

weight a numeric vector

satellite a logical vector

Examples

```
data(crabs2)
## maybe str(crabs2) ; plot(crabs2) ...
```

odds.ratios *creating a table with local or global odds ratios*

Usage

```
odds.ratios(m, type = "local")
```

Arguments

m

type

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (m, type = "local")
{
    nr <- nrow(m)
    if (nr < 2)
        stop("number of rows is less than two")
    nc <- ncol(m)
    if (nc < 2)
        stop("number of columns is less than two")
    if (length(type) > 1)
        stop("only one type is allowed")
    opts <- c("local", "global")
    type <- pmatch(type, opts)
    if (is.na(type))
        stop("only \"local\" or \"global\" allowed for type")
    result <- matrix(NA, nrow = nr - 1, ncol = nc - 1)
    if (type == 1)
        for (i in 1:(nr - 1)) for (j in 1:(nc - 1)) result[i,
            j] <- m[i, j] * m[i + 1, j + 1]/(m[i, j + 1] * m[i +
            1, j])
    if (type == 2)
        for (i in 1:(nr - 1)) for (j in 1:(nc - 1)) {
            num <- as.numeric(sum(m[1:i, 1:j])) * as.numeric(sum(m[(i +
                1):nr, (j + 1):nc]))
            den <- as.numeric(sum(m[1:i, (j + 1):nc])) * as.numeric(sum(m[(i +
                1):nr, 1:j]))
            result[i, j] <- num/den
        }
    result
  }
```

plot.local.or *plotting fourfold plots for odds ratios*

Usage

```
plot.local.or(m, col = c("azure4", "aquamarine4"))
```

Arguments

m	
col	

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (m, col = c("azure4", "aquamarine4"))
{
    nr <- nrow(m)
    if (nr < 2)
        stop("number of rows is less than two")
    nc <- ncol(m)
    if (nc < 2)
        stop("number of columns is less than two")
    par(mfrow = c(nr - 1, nc - 1))
    for (i in 1:(nr - 1)) for (j in 1:(nc - 1)) {
        fourfoldplot(m[i:(i + 1), j:(j + 1)], color = col)
    }
  }
```