|  |  |
| --- | --- |
| Benefits {Ecdat} | R Documentation |

Unemployment of Blue Collar Workers

**Description**

a cross-section from 1972

*number of observations* : 4877

*observation* : individuals

*country* : United States

**Usage**

data(Benefits)

**Format**

A time serie containing :

stateur

state unemployment rate (in %)

statemb

state maximum benefit level

state

state of residence code

age

age in years

tenure

years of tenure in job lost

joblost

a factor with levels (slack\\_work,position\\_abolished,seasonal\\_job\\_ended,other)

nwhite

non-white ?

school12

more than 12 years of school?

sex

a factor with levels (male, female)

bluecol

blue collar worker ?

smsa

lives is smsa ?

married

married ?

dkids

has kids ?

dykids

has young kids (0-5 yrs) ?

yrdispl

year of job displacement (1982=1,..., 1991=10)

rr

replacement rate

head

is head of household ?

ui

applied for (and received) UI benefits ?

**Source**

McCall, B.P. (1995) “The impact of unemployment insurance benefit levels on recipiency”, *Journal of Business and Economic Statistics*, **13**, 189–198.

**References**

Verbeek, Marno (2004) *A Guide to Modern Econometrics*, John Wiley and Sons, chapter 7.

Journal of Business Economics and Statistics web site :<http://amstat.tandfonline.com/loi/ubes20>.

**See Also**

[Index.Source](http://127.0.0.1:30641/help/library/Ecdat/help/Index.Source), [Index.Economics](http://127.0.0.1:30641/help/library/Ecdat/help/Index.Economics), [Index.Econometrics](http://127.0.0.1:30641/help/library/Ecdat/help/Index.Econometrics),[Index.Observations](http://127.0.0.1:30641/help/library/Ecdat/help/Index.Observations),

[Index.Time.Series](http://127.0.0.1:30641/help/library/Ecdat/help/Index.Time.Series)

**Scripts:**

#Library used for this program to run.

library(Ecdat)

#data used

data(Benefits)

mean(Benefits$age)

median(Benefits$age)

mode(Benefits$age)

#pre processing

summary(Benefits)

view(Benefits)

str(Benefits)

attributes(Benefits)

boxplot(Benefits$age)

sd(Benefits$age)

head(Benefits)

tail(Benefits)

levels(Benefits$joblost)

par(mfrow=c(1,1))

hist(Benefits$stateur, xlim=c(2,18), main = "State Unemployment Rates in 1972", xlab = "State Unemployment Rate (in %)")

par(mfrow=c(1,1))

boxplot(Benefits$stateur~Benefits$joblost, main = "Boxplot of State Unemployment Rates in 1972 among varying Reasons for Job Loss ", ylim = c(2,18), xlab = "Reasons for Job Loss", ylab = "State Unemployment Rate (in %)")

model\_job\_loss <- aov(stateur~joblost,Benefits)

anova(model\_job\_loss)

Tukey\_job\_loss = TukeyHSD(model\_job\_loss, ordered = FALSE, conf.level = 0.95)

Tukey\_job\_loss

par(mfrow=c(1,1))

plot(Tukey\_job\_loss)

with(Benefits,tapply(stateur,joblost,mean))

with(Benefits,tapply(stateur,joblost,var))

with(Benefits,tapply(stateur,joblost,length))

summary(aov(stateur~joblost,data=Benefits))

meanstar = mean(Benefits$stateur)

sdstar = sqrt(6.23)

simjoblost = Benefits$joblost

R = 10000

Fstar = numeric(R)

for (i in 1:R) {

#Residual should be normally distributed with known pooled-variance.

groupA = rnorm(1976, mean=meanstar, sd=sdstar)

groupB = rnorm(402, mean=meanstar, sd=sdstar)

groupC = rnorm(177, mean=meanstar, sd=sdstar)

groupD = rnorm(2322, mean=meanstar, sd=sdstar)

simstateur = c(groupA,groupB,groupC,groupD)

simdata = data.frame(simstateur,simjoblost)

Fstar[i] = oneway.test(simstateur~simjoblost, var.equal=T, data=simdata)$statistic

}

par(mfrow=c(1,1))

hist(Fstar, ylim=c(0,1), xlim=c(0, 8), prob=T, main = "Historgram of Empirical F-distribution")

x=seq(.25,6,.25)

points(x,y=df(x,3,4873),type="b",col="red")

print(realFstar<-oneway.test(stateur~joblost, var.equal=T, data=Benefits)$statistic)

mean(Fstar>=realFstar)

qf(.95,5,90)

quantile(Fstar,.95)

summary(Benefits$stateur)

sd(Benefits$stateur, na.rm = FALSE)

summary(Benefits$joblost)

sd(Benefits$joblost, na.rm = FALSE)

qqnorm(Benefits[,"stateur"], main = "Normal Q-Q Plot of the State Unemployment Rate")

qqline(Benefits[,"stateur"])

qqnorm(residuals(model\_job\_loss), main = "Normal Q-Q Plot of Residuals of 'model\_job\_loss'")

qqline(residuals(model\_job\_loss))

shapiro.test(Benefits[,"stateur"])

plot(fitted(model\_job\_loss),residuals(model\_job\_loss))

#Method applied

library(nutshell)

library(lattice)

library(MASS)

tbl=table(Benefits$joblost,Benefits$stateur)

tbl

chisq.test(tbl)

plot(tbl)

# Let's now see whether changes in two numerical variables (maternal age and estimated gestation) are related.

# We will use covariance for this

# The result indicates a positive linear relationship between the two variables.

as.numeric(as.character(Benefits$joblost))

as.nuemric(Benefits$joblost)

joblost = Benefits$joblost

gest = Benefits$stateur

cov(joblost,gest)

# Producing a Pearson correlation on the entire raw dataset.

# This will generate an error because the dataset has missing values

# Note that -1 <= R == negative correlation, +1 >= R == positive correlation

# anything around 0 == no correlation

na.rm=TRUE

cor(Benefits$stateur,Benefits$joblost,use="all.obs",method=c("pearson"))

cor(Benefits$stateur,Benefits$age,use="all.obs",method=c("pearson"))

cor(Benefits$stateur,Benefits$statemb,use="complete.obs",method=c("peasrson"))

# The correlation formula requires a numeric attribute, so we need to transform

Benefits$joblost=as.numeric(Benefits$joblost)

is.numeric(Benefits$joblost)

summary(Benefits$joblost)

Benefits$bluecol=as.numeric(Benefits$bluecol)

is.numeric(Benefits$bluecol)

summary(Benefits$bluecol)

aggdata<-aggregate(Benefits, by =list(Benefits$stateur,Benefits$joblost), FUN=mean, na.rm=TRUE)

warnings()

plot(Benefits$stateur,Benefits$joblost,main="Scatterplot Stateur vs. joblost",xlab="stateur",ylab="joblost")

plot(Benefits$stateur,Benefits$bluecol,main="Scatterplot stateur vs. bluecol", xlab="stateur",ylab="bluecol")

# Regression function

reg1<- lm(Benefits$stateur~Benefits$joblost)

plot(Benefits$stateur~Benefits$joblost)

abline(reg1)

plot(Benefits$stateur,Benefits$bluecol,main="Scatterplot stateur vs. bluecol", xlab="stateur",ylab="bluecol")

reg2<- lm(Benefits$stateur~Benefits$bluecol)

abline(reg2)

install.packages("tree")

install.packages("party")

install.packages("rpart")

install.packages("car")

install.packages("mlbench")

install.packages("mboost")

install.packages("textir")

install.packages("class")

install.packages("e1071")

install.packages("randomForest")

library(tree)

Ben <- sample(2, nrow(Benefits), replace=TRUE, prob=c(0.7, 0.3))

trainDataTree <- Benefits[Ben==1,]

testDataTree <- Benefits[Ben==2,]

myFormula <- stateur ~ joblost + bluecol + age + state

Benefits\_tree <- tree(myFormula, data=trainDataTree)

summary(Benefits\_tree)

str(Benefits\_tree)

print(Benefits\_tree)

plot(Benefits\_tree)

text(Benefits\_tree)

testPred <- predict(Benefits\_tree, newdata = testDataTree)

show(testPred)

library(textir)

library(MASS)

par(mfrow=c(3,3), mai=c(.3,.6,.1,.1))

plot(joblost ~ type, data=Benefits, col=c(grey(.2),2:6))

n=length(Benefits$stateur)

nt=200

set.seed(1) ## to make the calculations reproducible in repeated runs

train <- sample(1:n,nt)

x=Benefits[,c(4,1)]

x[,1]=(x[,1]-mean(x[,1]))/sd(x[,1])

x[,2]=(x[,2]-mean(x[,2]))/sd(x[,2])

x[1:3,]

library(class)

nearest1 <- knn(train=x[train,],test=x[-train,],cl=Benefits$stateur[train],k=1)

nearest5 <- knn(train=x[train,],test=x[-train,],cl=Benefits$stateur[train],k=5)

data.frame(Benefits$stateur[-train],nearest1,nearest5)

par(mfrow=c(1,2))

plot(x[train,],col=Benefits$stateur[train],cex=.8,main="1-nearest neighbor")

points(x[-train,],bg=nearest1,pch=21,col=grey(.9),cex=1.25)

## plot for k=5 nearest neighbors

plot(x[train,],col=Benefits$stateur[train],cex=.8,main="5-nearest neighbors")

points(x[-train,],bg=nearest5,pch=21,col=grey(.9),cex=1.25)

legend("topright",legend=levels(Benefits$stateur),fill=1:6,bty="n",cex=.75)

boxplot(train)

plot(train)

## calculate the proportion of correct classifications on this one

## training set

pcorrn1=100\*sum(Benefits$stateur[-train]==nearest1)/(n-nt)

pcorrn5=100\*sum(Benefits$stateur[-train]==nearest5)/(n-nt)

pcorrn1

pcorrn5

#Association

Benefits[1:59,]

length(Benefits$stateur)

Benefits$stateur <- factor(Benefits$stateur)

levels(Benefits$stateur)

levels(Benefits$joblost)

library(arules)

playlist <- split(x=Benefits[,"stateur"],f=Benefits$stateur)

playlist[1:2]

playlist <- lapply(playlist,unique)

playlist <- as(playlist,"transactions")

playlist[1:2]

itemFrequency(playlist)

musicrules <- apriori(playlist,parameter=list(support=.01,confidence=.5))

inspect(musicrules)

## Too much unordered data. We need to sort this mess!

## Sorting by support, i.e. frequency

inspect(sort(subset(musicrules), by="support"))

## Now let's see what this looks like by confidence, i.e. rule strength

inspect(sort(subset(musicrules), by="confidence"))

## That's a lot of Coldplay and Radiohead. Ugh.

## Let's see how support and confidence work together in the lift metric.

inspect(sort(subset(musicrules), by="lift"))

## Ha! We can remove Coldplay and Radiohead from our output if we set the

## cutoff for lift > 5.

## Remember that lift gives us the best quality rules

inspect(subset(musicrules, subset=lift > 5))

## Ha! No more Coldplay! No more Radiohead!

## Lastly, let's sort by confidence to make it easier to understand

inspect(sort(subset(musicrules, subset=lift > 5), by="confidence"))

#classification

library(mlbench)

## barplots for specific issue

plot(as.factor(Benefits[,2]))

title(main="Benefts of joblost", xlab="Benefits", ylab="Joblost")

Benefits[,"train"] <- ifelse(runif(nrow(Benefits))<0.80,1,0)

## Get col number of train / test indicator column (needed later)

trainColNum <- grep('train', names(Benefits))

## separate training and test sets and remove training column before modeling

trainB <- Benefits[Benefits$stateur==1,-trainColNum]

testB <- Benefits[Benefits$joblost==0,-trainColNum]

## Now we can build the Naive Bayes model

## Load e1071 library and invoke naiveBayes method

library(e1071)

nb\_model <- naiveBayes(stateur~.,data = Benefits)

nb\_model

summary(nb\_model)

str(nb\_model)

## Now that we have a model, we can do some predicting. We do this by feeding

## our test data into our model and comparing the predicted party affiliations

## with the known ones. The latter is done via the wonderfully named confusion

## matrix - a table in which true and predicted values for each of the predicted

## classes are displayed in a matrix format.

## ... and the moment of reckoning

nb\_test\_predict <- predict(nb\_model,Benefits[,-1])

## Building the confusion matrix

table(pred=nb\_test\_predict,true=Benefits$stateur)

## Remember that in the confusion matrix (as defined above), the true values

## are in columns and the predicted values in rows.

## The output doesn't look too bad, does it?

## However, we need to keep in mind that this could well be quirk of the choice of dataset.

## To address this, we should get a numerical measure of the efficacy of the algorithm

## and for different training and testing datasets. A simple measure of efficacy would be

## the fraction of predictions that the algorithm gets right.

## fraction of correct predictions

mean(nb\_test\_predict==Benefits$stateur)

## But how good is this prediction? This question cannot be answered with only a single

## run of the model; we need to do many runs and look at the spread of the results. To do

## this, we'll create a function which takes the number of times the model should be run

## and the training fraction as inputs and spits out a vector containing the proportion

## of correct predictions for each run.

## Function to create, run and record model results

nb\_multiple\_runs <- function(train\_fraction,n){

fraction\_correct <- rep(NA,n)

for (i in 1:n){

Benefits[,"train"] <- ifelse(runif(nrow(Benefits))<train\_fraction,1,0)

trainColNum <- grep('train',names(Benefits))

trainB <- Benefits[Benefits$train==1,-trainColNum]

testB <- Benefits[Benefits$train==0,-trainColNum]

nb\_model <- naiveBayes(Class~.,data = Benefits)

nb\_test\_predict <- predict(nb\_model,testB[,-1])

fraction\_correct[i] <- mean(nb\_test\_predict==Benefits$stateur)

}

return(fraction\_correct)

}

## Let's do 20 runs, 80% of data randomly selected for training set in each run

fraction\_correct\_predictions <- nb\_multiple\_runs(0.8,20)

fraction\_correct\_predictions

## Summary of results

summary(fraction\_correct\_predictions)

## Standard deviation

sd(fraction\_correct\_predictions)