Factor:

factors are variables in R which take on a limited number of different values; such variables are often refered to as categorical variables.

Factors in R are stored as a vector of integer values with a corresponding set of character values to use when the factor is displayed. The factor function is used to create a factor. The only required argument to factor is a vector of values which will be returned as a vector of factor values. Both numeric and character variables can be made into factors.

Factors represent a very efficient way to store character values, because each unique character value is stored only once, and the data itself is stored as a vector of integers. Because of this, read.table will automatically convert character variables to factors unless the as.is= argument is specified.

Example for changing levels:

> data = c(1,2,2,3,1,2,3,3,1,2,3,3,1)

> fdata = factor(data)

> fdata

 [1] 1 2 2 3 1 2 3 3 1 2 3 3 1

Levels: 1 2 3

> rdata = factor(data,labels=c("I","II","III"))

> rdata

 [1] I   II  II  III I   II  III III I   II  III III I

Levels: I II III

To convert the default factor fdata to roman numerals, we use the assignment form of the levels function:

> levels(fdata) = c('I','II','III')

> fdata

 [1] I   II  II  III I   II  III III I   II  III III I

Levels: I II III

Commands executed in practical 2:

data("iris")

iris

str(iris)

data = c(1,2,2,3,1,2,3,3,1,2,3,3,1)

fdata = factor(data)

fdata

data = c(56,57,59,62,63,64,65,72,71,73,75,80,91)

fdata = factor(data)

fdata

data = c("january","february","march","april")

fdata = factor(data)

fdata

mons = c("March","April","January","November","January",

"September","October","September","November","August",

"January","November","November","February","May","August",

"July","December","August","August","September","November",

"February","April")

mons = factor(mons)

table(mons)

#ordered factor

mons = factor(mons,levels=c("January","February","March",

"April","May","June","July","August","September",

"October","November","December"),ordered=TRUE)

table(mons)

day\_vector <- c('evening', 'morning', 'afternoon', 'midday', 'midnight', 'evening')

factor\_day <- factor(day\_vector, order = TRUE, levels =c('morning', 'midday', 'afternoon', 'evening', 'midnight'))

factor\_day

days <- factor(factor\_day,levels=c('morning', 'midday', 'afternoon', 'evening', 'midnight'),ordered=TRUE)

days

table(days)

summary(days)

data("mtcars")

mtcars

dataset <- mtcars

class(dataset$mpg)

class(iris$mpg)

data("ToothGrowth")

ToothGrowth

nrow(ToothGrowth)

ncol(ToothGrowth)

head(ToothGrowth)

head(mtcars)

ToothGrowth[6,2]

ToothGrowth[5,]

ToothGrowth[,2]

ToothGrowth[1:5,]

ToothGrowth[1:5,2]

attach(ToothGrowth)

mean(len)

min(len)

x<- ToothGrowth[1:5,]

x

boxplot(x)

barplot(len)

y <- ToothGrowth[1:5,3]

pie(y,labels)

t <- ToothGrowth[,"len"]

t

summary(t)

s <- ToothGrowth[,3]

summary(s)

hist(iris$Sepal.Length, col ="blue" , main = "Sepal Length Histogram" , xlab = "Length", ylab="Frequency")

pdf("petallengthhisto.pdf")

hist(iris$Sepal.Length, col ="blue" , main = "Sepal Length Histogram" , xlab = "Length", ylab="Frequency")

dev.off()

plot(iris$Sepal.Width, iris$Sepal.Length, col ="blue", main= "Scatterplot", xlab ="Sepal Width", ylab="Sepal Length")

png("sepalwidth.png")

plot(iris$Sepal.Width, iris$Sepal.Length, col ="blue", main= "Scatterplot", xlab ="Sepal Width", ylab="Sepal Length")

dev.off()

boxplot(Sepal.Length~Species, data=iris, main = "sepal length by species", xlab ="Species", ylab = "Sepal Length")

# chart of other data

x <- c(30,60,20)

labels <- c("math", "stat", "physics")

pie(x,labels)

pie(x, labels, main = "subject", col = rainbow(length(x)))

pie(x, labels, main = "subject", col = rainbow(length(x)),clockwise = T)

H <- c(7000,10000,6000,8000)

M <- c("Jan" , "Feb", "Mar" , "Apr")

barplot(H,names.arg=M,xlab="Month", ylab ="Revenue",col="green",main ="Revenue chart",border ="black")

H <- c(7,12,28,3,41)

barplot(H)

v <- c(7,12,28,3,41)

plot(v,type = "o")

input <- mtcars[,c('wt','mpg')]

print(head(input))

plot(x = input$wt,y = input$mpg,

xlab = "Weight",

ylab = "Milage",

xlim = c(2.5,5),

ylim = c(15,30),

main = "Weight vs Milage")

# all plots in a data set together

pairs(~wt+mpg+disp+cyl,data = mtcars, main = "Scatterplot Matrix")

getwd()

data <- read.csv("input.csv")

print(data)

data1<-read.csv("copper1.csv")

data1

boxplot(data1$Cu ~ data1$Source, xlab = "Source",ylab = "Cu", main ="Boxplot to show

atmospheric copper concentration by sites", col = "green",

border = "black")

Practical 3:

Creating samples:

print(sample(1:3))

print(sample(1:3, size=3, replace=FALSE)) # same as previous line

print(sample(c(2,5,3), size=4, replace=TRUE)

print(sample(1:2, size=10, prob=c(1,3), replace=TRUE))

1] 3 1 2

[1] 2 1 3

[1] 2 5 2 2

[1] 2 2 2 1 1 2 2 2 1 2

**Overview**

By default [sample()](http://www.stat.psu.edu/~dhunter/R/html/base/html/sample.html) randomly reorders the elements passed as the first argument. This means that the default size is the size of the passed array. replace=TRUE makes sure that no element occurs twice.

The last line uses a weighed random distribution instead of a uniform one. One out of four numbers are 1, the out of four are 3.

**Arguments**

**size**

This is the size of the returned list. If replace is disabled size must be no bigger than the length of the first argument.

**replace**

If this is true a sample may contain an element several times while another element might not occur at all.

print(sample(c(2,5,3), size=3, replace=FALSE))

print(sample(c(2,5,3), size=3, replace=TRUE))

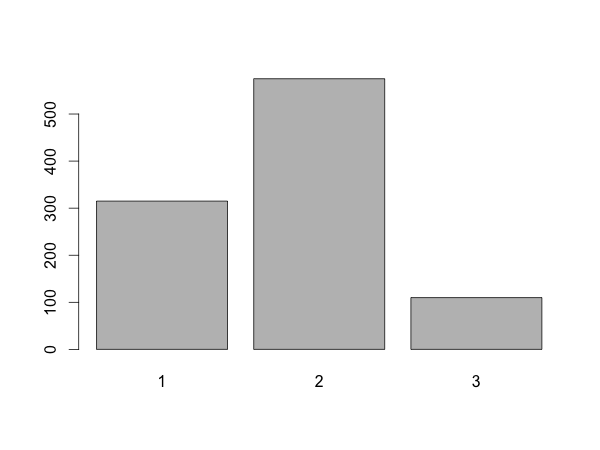
[1] 2 3 5

[1] 2 3 3

Allowing some elements to occur more than once lets you get a sample longer than the first argument.

**prob**

barplot(table(sample(1:3, size=1000, replace=TRUE, prob=c(.30,.60,.10))))



The prob=c(.30,.60,.10) cause 30% ones, 60% twos and 10% threes. The numbers don't have to add up to 1 - they don't in the example at the top of the page.

Table sums up the individual items in the 1000-element list.