Histograms and bar plots

Roulette is a fascinating example of a betting game using random outcomes. In order to explore some properties of roulette spins, let's visualize some randomly drawn numbers in the range of those in an European roulette game (0 to 36). Histograms allow the graphic representation of the distribution of variables. Let's have a look at it! Type in the following code:

```
1  set.seed(1)
2  drawn = sample(0:36, 100, replace = T)
3  hist(drawn, main = "Frequency of numbers drawn",
4     xlab = "Numbers drawn", breaks=37)
```

Here we first set the seed number to 1 (see line 1). For reproducibility reasons use attribute ylim to define the range of the plotting area (from 0 to 1), and attribute main to print the title of the plots.

```
par(mfrow = c(2,3))
barplot(mean(subset(Data, isCol1 == 1)$isRed), ylim=(c(0,1)),
   main = "Prop. of red in Col. 1")
barplot(mean(subset(Data, isCol2 == 1)$isRed), ylim=(c(0,1)),
   main = "Prop. of red in Col. 2")
barplot(mean(subset(Data, isCol3 == 1)$isRed), ylim=(c(0,1)),
   main = "Prop. of red in Col. 3")
```

```
barplot(mean(subset(Data, isCol1 == 1)$isEven), ylim=(c(0,1)),
    main = "Prop. of even numbers in Col. 1")
barplot(mean(subset(Data, isCol2 == 1)$isEven), ylim=(c(0,1)),
    main = "Prop. of even numbers in Col. 2")
barplot(mean(subset(Data, isCol3 == 1)$isEven), ylim=(c(0,1)),
    main = "Prop. of even numbers in Col. 3")
```

You might have noticed that we have lost important information in the process; the total of numbers drawn from each column, and the number of zeros; and we needed to produce one bar plot per column, which is a bit tricky. Let's solve these problems by first adding a single attribute which indicates the membership of the drawn numbers to `Column 1`, `Column 2` and `Column 3`.

```
1   for (i in 1:nrow(Data)){
2     if(Data$isCol1[i]== 1){ Data$Column[i]=1 }
3       else if (Data$isCol2[i] == 1 ) { Data$Column[i] = 2 }
4       else if (Data$isCol3[i] == 1 ) { Data$Column[i] = 3 }
5       else {Data$Column[i] = 0 }
6   }
```

On line 1, we start a `for` loop that will iterate from i = 1 to i = the number of rows in data frame `Data`. We use nested condition in lines 2 to 5
number (1 if attri

The body of the function computes and returns the transpose of the means of each nominal attributes:

```
1   proportions = function(n = 100) {
2       DF=attributes(n)
3       return(data.frame(t(colMeans(DF[3:ncol(DF)]))))
4   }
```

The body of this function calls our attributes() function and passes the number of roulette draws to it (line 2). It then returns a data frame which contains the transpose means of all the columns, except columns 1 and 2 (which are not of interest here).

Our next function multisample() will return a data frame containing the proportions of each attribute of each of k samples (one sample per row) of n numbers drawn. It will by default draw 100 samples of 100 numbers. After starting the function declaration on line 1, we set the seed to the provided value, or the default value on line 2. We then create a vector containing the values returned by a first call to the proportions() function. In the following loop, we append iteratively values returned by function proportions() (lines 4 to 7). Finally, we return the resulting data frame (line 8), and close the function code block (line 9).

```
1   multisample = function(n=100,k=100, Seed=3){
2       set.seed(Seed)
3       ColMeans.df=proportions(n)
4       for (i in 1:k-1){
5           ColMeans.df=rbind(ColMeans.df,
6               proportions(n))
7       }
8       return(ColMeans.df)
9   }
```

The function here uses the coefficients of a linear model as argument. I will discuss the `lm()` function which provides such coefficients in the chapter about regression:

```
samples = multisample()
par(mfrow=c(1,1))
plot(samples$isOdd,samples$isRed,
    main = "Relationship between attributes Red and Even ",
    xlab = "Proportion of Even numbers",
    ylab = "Proportion of Red numbers")
abline(lm(samples$isOdd~samples$isRed))
```

The output is provided below: