Numerical Descriptives in R

Jim Bentley

1 Sample Data

The following code creates a couple of sample data frames that we will use in our examples.

```
> sex = c(rep("Female",12),rep("Male",7))
> mass = c(36.1, 54.6, 48.5, 42.0, 50.6, 42.0, 40.3, 33.1, 42.4, 34.5,
+          51.1, 41.2, 51.9, 46.9, 62, 62.9, 47.4, 48.7, 51.9)
> rate = c(995, 1425, 1396, 1418, 1502, 1256, 1189, 913, 1124, 1052,
+          1347, 1204, 1867, 1439, 1792, 1666, 1362, 1614, 1460)
> bps5.4.9 = data.frame(sex, mass, rate)
> htwt = read.csv(
+ "http://newton.uor.edu/facultyfolder/jim_bentley/downloads/math111/htwt.csv")
> htwt$Group = factor(htwt$Group,levels=c(1,2),labels=c("Male","Female"))
> hospitals = read.csv(
+ "http://newton.uor.edu/facultyfolder/jim_bentley/downloads/math111/hospitals.csv")
```

Note that the plus signs (+) at the begining of the lines are there to indicate that R is reading from a new line. They should not be entered as part of the code.

We can now check to see if the data frames have been created by entering

```
> ls()

[1] "bps5.4.9"  "hospitals" "htwt"      "mass"      "rate"      "sex"
```

Note that the listing also shows the individual variables that were used to create the data frame. These can be deleted by using rm().

```
> rm("sex","mass","rate")
> ls()

[1] "bps5.4.9"  "hospitals" "htwt"
```

R contains a number of predefined data frames. Some of these will be used in the examples that are presented below.

2 Loading R Packages

```
> ## load a few required packages
> #install.packages("xtable")
> library(Hmisc)
> library(xtable)
> library(ggplot2)
> library(survival)
> library(Rcmdr)
```

3 Simple Univariate Descriptives

Summary statistics for the `htwt` data can be obtained via the `summary` function.

```
> summary(htwt)
```

```
     Height          Weight          Group
 Min.   :51.0   Min.   : 82.0   Male  : 9
 1st Qu.:56.0   1st Qu.:108.2   Female:11
 Median :59.5   Median :123.5
 Mean   :62.1   Mean   :139.6
 3rd Qu.:68.0   3rd Qu.:166.8
 Max.   :79.0   Max.   :228.0
```

```
> summary(subset(htwt,Group=="Male"))
```

```
     Height       Weight       Group
 Min.   :52   Min.   : 87   Male  :9
 1st Qu.:59   1st Qu.:119   Female:0
 Median :64   Median :159
 Mean   :65   Mean   :155
 3rd Qu.:71   3rd Qu.:191
 Max.   :79   Max.   :228
```

```
> summary(subset(htwt,Group=="Female"))
```

```
     Height          Weight          Group
 Min.   :51.00   Min.   : 82.0   Male  : 0
 1st Qu.:54.00   1st Qu.:106.5   Female:11
 Median :58.00   Median :119.0
 Mean   :59.73   Mean   :127.0
 3rd Qu.:64.00   3rd Qu.:153.0
 Max.   :76.00   Max.   :199.0
```

Note that subsets of the data can be summarized using the `Group` option.

Specific values may be obtained by using specialized functions. The sample mean is computed using the `mean` function. The same value can be found through the use of the `sum` function.

```
> mean(htwt$Weight)

[1] 139.6

> sum(htwt$Weight)

[1] 2792

> length(htwt$Weight)

[1] 20

> sum(htwt$Weight)/length(htwt$Weight)

[1] 139.6

> colMeans(htwt[,1:2])

Height Weight
  62.1  139.6
```

We now compute the variance by summing the squared deviations from the mean and dividing by $n-1$. Computing the mean once and assigning it to `xbar` and then calling `xbar` is more efficient than using `mean(htwt$Weight)` in the sum.

```
> xbar = mean(htwt$Weight)
> sum((htwt$Weight-xbar)^2)

[1] 35330.8

> sum((htwt$Weight-xbar)^2)/(length(htwt$Weight)-1)

[1] 1859.516
```

Or, we can use the `var` fucntion to compute the variance.

```
> var(htwt$Weight)

[1] 1859.516

> apply(htwt[,1:2],2,var)

    Height     Weight
  71.25263 1859.51579
```

The standard deviation is the square root of the variance. Thus, it is simple to compute the standard deviation for the `Weight` data.

```
> sqrt(var(htwt$Weight))
```

```
[1] 43.1221
```

```
> sqrt(apply(htwt[,1:2],2,var))
```

```
   Height    Weight
 8.441127 43.122103
```

When outliers or skewness are present, the above measures of centrality and spread become suspect. At these times we often turn to the median and the IQR. R makes it easy to compute these values.

We can compute the median and quartiles by sorting and then counting. The `sort` function makes this easy.

```
> sort(htwt$Weight)
```

```
 [1]  82  87  87 101 103 110 112 119 119 122 125 151 155 157 159 190 191 195 199
[20] 228
```

However, for large data sets this may be problematic. Using the R functions `median` and `quantile` are more efficient.

```
> median(htwt$Weight)
```

```
[1] 123.5
```

```
> quantile(htwt$Weight)
```

```
    0%    25%    50%    75%   100%
 82.00 108.25 123.50 166.75 228.00
```

```
> apply(htwt[,1:2],2,median)
```

```
Height Weight
  59.5  123.5
```

`Rcmdr` has the function `numSummary` which can be called from the `Rcmdr` menu –**Statistics** – **Summaries** – **Numerical Summaries**. It can also be called from the command prompt. `numSummary` computes all of the above statistics with a single call.

```
> numSummary(htwt["Weight"], statistics=c("mean", "sd", "quantiles"))
```

```
  mean       sd 0%    25%   50%    75% 100%  n
 139.6 43.1221 82 108.25 123.5 166.75  228 20
```

While it is possible to use mean, var, etc. and get results by group, using `numSummary` with `groups=` is easier.

```
> numSummary(htwt[,c("Height","Weight")], groups=htwt$Group,
+ statistics=c("mean", "sd", "quantiles"))
```

```
Variable: Height
           mean       sd 0% 25% 50% 75% 100%  n
Male   65.00000 8.972179 52  59  64  71   79  9
Female 59.72727 7.564270 51  54  58  64   76 11

Variable: Weight
       mean       sd 0%   25% 50% 75% 100%  n
Male    155 48.99235 87 119.0 159 191  228  9
Female  127 34.99714 82 106.5 119 153  199 11
```

4 Tables

Tables can be created both from the command line and from `Rcmdr`. We will take a look at the `hospitals` data set.

4.1 Manual Tables

The `hospitals` data frame contains three variables and 2900 observations.

```
> names(hospitals)

[1] "hospital"  "condition" "survival"

> hospitals[c(1:3,2900),]

     hospital condition survival
1           A      Good Survived
2           A      Good Survived
3           A      Good Survived
2900        B      Poor     Died
```

To get simple frequencies of each of the variables we can enter

```
> table(hospitals[,"hospital"])

   A    B
2100  800

> table(hospitals[,"condition"])

Good Poor
1200 1700

> table(hospitals[,"survival"])

    Died Survived
      79     2821
```

Two way tables are created by providing two columns of data. Examples might be `survival` by `hospital` or `survival` by `condition`.

```
> table(hospitals[,c("hospital","survival")])

        survival
hospital Died Survived
       A   63     2037
       B   16      784

> table(hospitals[,c("survival","condition")])

          condition
survival   Good Poor
  Died       14   65
  Survived 1186 1635
```

Notice that the order of the columns determines the rows and columns respectively.

The `table` function will also generate three-way tables.

```
> table(hospitals[,c("survival","hospital","condition")])

, , condition = Good

          hospital
survival      A    B
  Died        6    8
  Survived  594  592

, , condition = Poor

          hospital
survival      A    B
  Died       57    8
  Survived 1443  192
```

The `table` function assumes that the columns are entered as rows, columns, and tables respectively.

While the `table` function is good for getting counts, it does not generate row, column, or table percentages. `Rcmdr` does this through the use of the `xtabs`, `colPercents`, and `rowPercents` functions which are accessible through its menus—**Statistics – Contingency Tables**. These functions can also be called from the command line.

We first generate a counts table using `xtabs`.

```
> .Table = xtabs(~survival+hospital+condition, data=hospitals)
> .Table
```

```
, , condition = Good

          hospital
survival      A    B
  Died        6    8
  Survived  594  592

, , condition = Poor

          hospital
survival      A    B
  Died       57    8
  Survived 1443  192
```

To get column percents we us `colPercents` on the saved table.

```
> colPercents(.Table)

, , condition = Good

          hospital
survival     A     B
  Died       1   1.3
  Survived  99  98.7
  Total    100 100.0
  Count    600 600.0

, , condition = Poor

          hospital
survival        A   B
  Died        3.8   4
  Survived   96.2  96
  Total     100.0 100
  Count    1500.0 200
```

Row percents can be generated in a similar manner by using `rowPercents`.

```
> rowPercents(.Table)

, , condition = Good

          hospital
survival      A    B Total Count
  Died     42.9 57.1   100    14
  Survived 50.1 49.9   100  1186
```

```
, , condition = Poor

          hospital
survival      A    B Total Count
  Died     87.7 12.3   100    65
  Survived 88.3 11.7   100  1635
```

Finally, we can compute table percentages for two-way tables by using `totPercents`. We clean up by removing the table with `rm`.

```
> .Table = xtabs(~survival+hospital, data=hospitals)
> totPercents(.Table)

            A    B Total
Died      2.2  0.6   2.7
Survived 70.2 27.0  97.3
Total    72.4 27.6 100.0

> rm(.Table)
```