

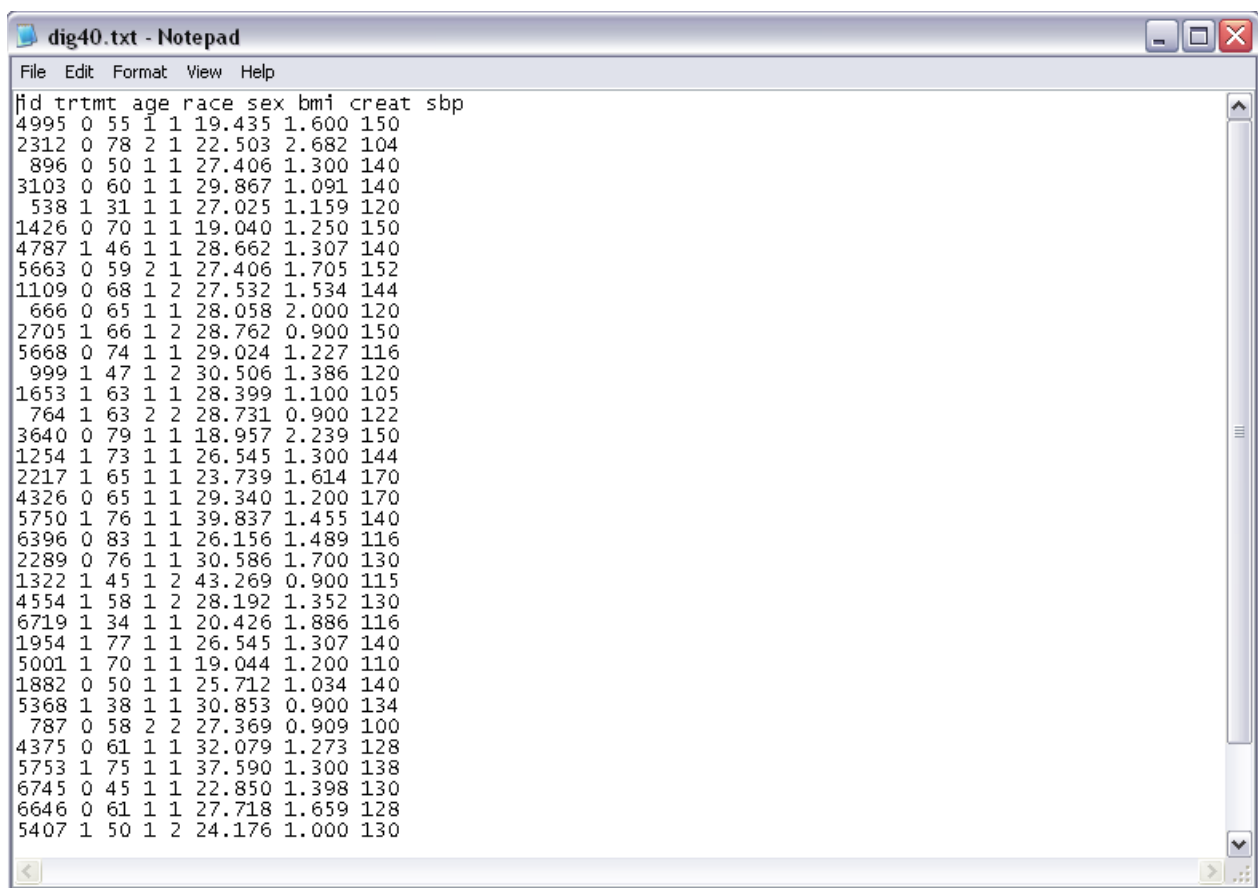
R Program Notes

Biostatistics: A Guide to Design, Analysis, and Discovery Second Edition

by Ronald N. Fortofer, Eun Sul Lee, Mike Hernandez

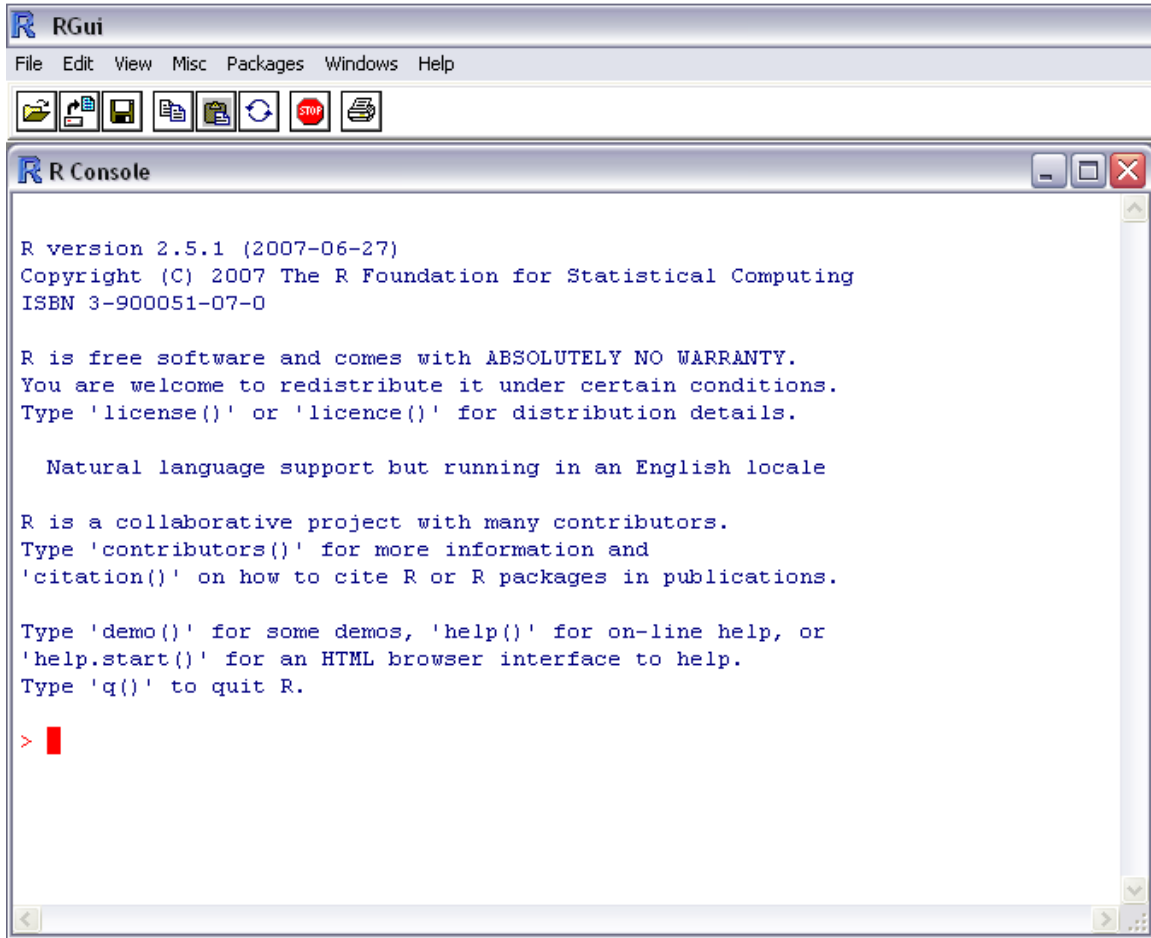
Chapter 3: Descriptive Statistics

Before discussing how to use **R** to summarize data and make graphs, we begin by looking at how data can be read directly into **R**. Start by accessing the **dig40.txt** file from the companion website and use an ASCII editor like notepad to view the contents of the file as shown below. The data set is described in **Table 3.1 Digoxin clinical trial data for 40 participants** in the textbook.

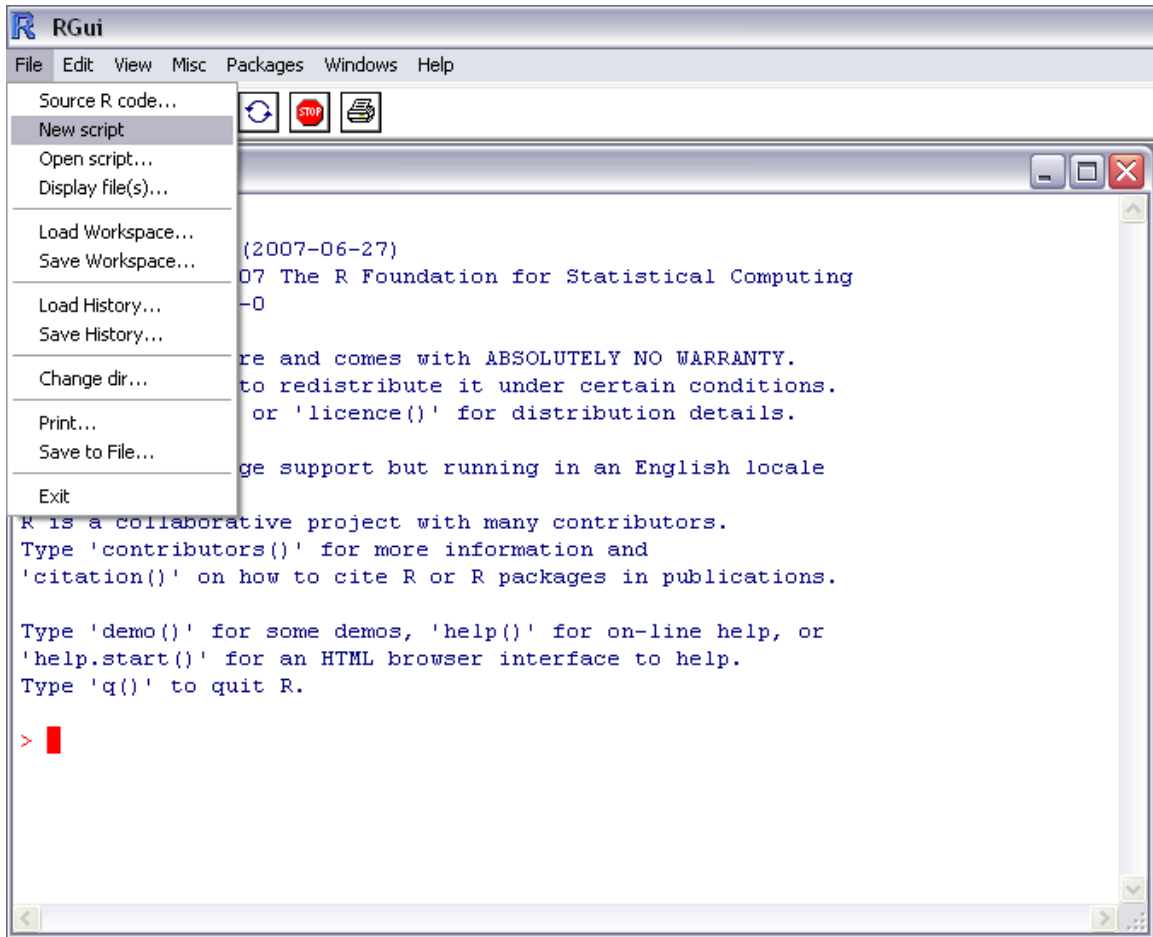



```
id trtmt age race sex bmi creat sbp
4995 0 55 1 1 19.435 1.600 150
2312 0 78 2 1 22.503 2.682 104
896 0 50 1 1 27.406 1.300 140
3103 0 60 1 1 29.867 1.091 140
538 1 31 1 1 27.025 1.159 120
1426 0 70 1 1 19.040 1.250 150
4787 1 46 1 1 28.662 1.307 140
5663 0 59 2 1 27.406 1.705 152
1109 0 68 1 2 27.532 1.534 144
666 0 65 1 1 28.058 2.000 120
2705 1 66 1 2 28.762 0.900 150
5668 0 74 1 1 29.024 1.227 116
999 1 47 1 2 30.506 1.386 120
1653 1 63 1 1 28.399 1.100 105
764 1 63 2 2 28.731 0.900 122
3640 0 79 1 1 18.957 2.239 150
1254 1 73 1 1 26.545 1.300 144
2217 1 65 1 1 23.739 1.614 170
4326 0 65 1 1 29.340 1.200 170
5750 1 76 1 1 39.837 1.455 140
6396 0 83 1 1 26.156 1.489 116
2289 0 76 1 1 30.586 1.700 130
1322 1 45 1 2 43.269 0.900 115
4554 1 58 1 2 28.192 1.352 130
6719 1 34 1 1 20.426 1.886 116
1954 1 77 1 1 26.545 1.307 140
5001 1 70 1 1 19.044 1.200 110
1882 0 50 1 1 25.712 1.034 140
5368 1 38 1 1 30.853 0.900 134
787 0 58 2 2 27.369 0.909 100
4375 0 61 1 1 32.079 1.273 128
5753 1 75 1 1 37.590 1.300 138
6745 0 45 1 1 22.850 1.398 130
6646 0 61 1 1 27.718 1.659 128
5407 1 50 1 2 24.176 1.000 130
```

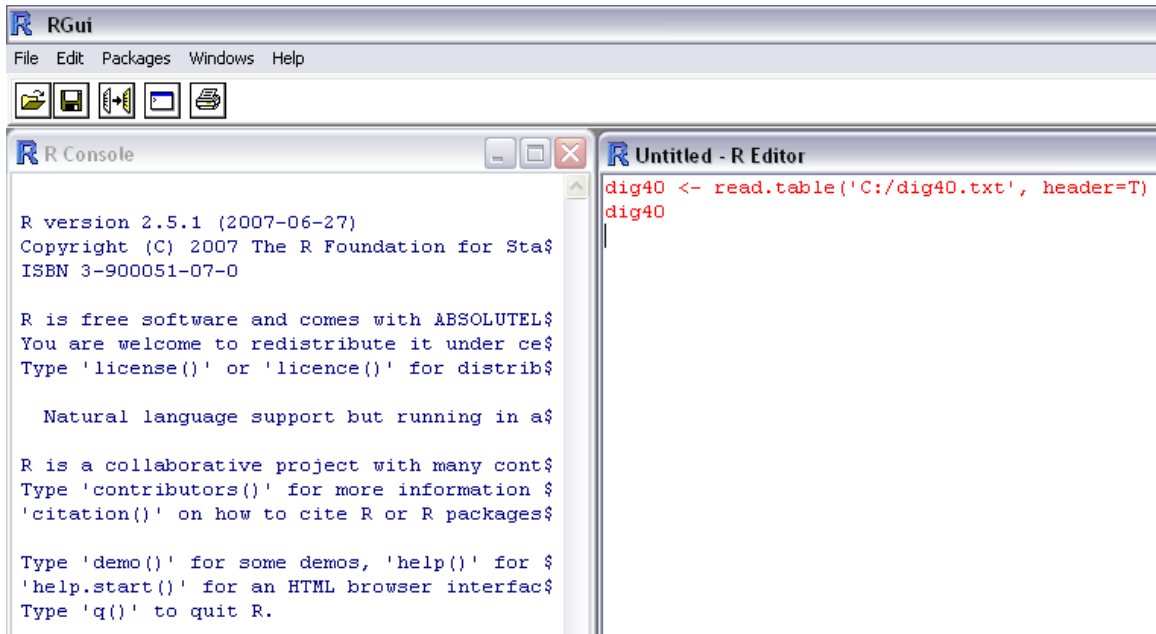
Next, launch the **R** program. Notice that **R version 2.5.1** is displayed below; however, newer versions become available periodically so you may be using a more recent version of the software.



Select **File** -> **New** script to launch the **R Editor**.



Commands are written in the **R Editor** as shown below. To run the commands, highlight them first then click the run button. The run button looks like this:  .



As the commands in the **R Editor** above indicate, to create an **R** data frame we used the **read.table** function in **R**. To do this, start by saving the data set **dig40.txt** to some auxiliary drive, like the C drive. The text file should be opened and inspected. Check the data in the text file to see if a header line is provided indicating the names of each of the variables in the data set. In the case of the **dig40.txt** data set, the first line contains variable names therefore we need to specify that the data contain a header. This is easily done within the **read.table** command by including the option **header = T**. The command **help(read.table)** can be used to access the **R** help documentation when needed.

R commands:

```
#Access the data directly from the book's companion website  
  
dig40 <- read.table('http://www.biostat-edu.com/files/dig40.txt', header=T)  
dig40
```

R output:

	id	trtmt	age	race	sex	bmi	creat	sbp
1	2289	0	76	1	1	30.586	1.700	130
2	6745	0	45	1	1	22.850	1.398	130
3	1322	1	45	1	2	43.269	0.900	115
4	538	1	31	1	1	27.025	1.159	120
5	999	1	47	1	2	30.506	1.386	120
6	3103	0	60	1	1	29.867	1.091	140
7	1954	1	77	1	1	26.545	1.307	140
8	5750	1	76	1	1	39.837	1.455	140
9	1109	0	68	1	2	27.532	1.534	144
10	4787	1	46	1	1	28.662	1.307	140
11	666	0	65	1	1	28.058	2.000	120
12	6396	0	83	1	1	26.156	1.489	116
13	5753	1	75	1	1	37.590	1.300	138
14	1882	0	50	1	1	25.712	1.034	140
15	5663	0	59	2	1	27.406	1.705	152
16	6719	1	34	1	1	20.426	1.886	116
17	4995	0	55	1	1	19.435	1.600	150
18	4055	0	71	1	1	22.229	1.261	100
19	4554	1	58	1	2	28.192	1.352	130
20	2217	1	65	1	1	23.739	1.614	170
21	896	0	50	1	1	27.406	1.300	140
22	5368	1	38	1	1	30.853	0.900	134
23	3403	0	55	1	2	21.790	1.170	130
24	1426	0	70	1	1	19.040	1.250	150
25	764	1	63	2	2	28.731	0.900	122
26	5668	0	74	1	1	29.024	1.227	116
27	1653	1	63	1	1	28.399	1.100	105
28	1254	1	73	1	1	26.545	1.300	144
29	2312	0	78	2	1	22.503	2.682	104
30	2705	1	66	1	2	28.762	0.900	150
31	4181	0	44	2	2	26.370	1.148	124
32	3641	0	64	1	1	21.228	0.900	130
33	2439	1	49	1	1	15.204	1.307	140
34	3640	0	79	1	1	18.957	2.239	150
35	6646	0	61	1	1	27.718	1.659	128
36	787	0	58	2	2	27.369	0.909	100
37	5407	1	50	1	2	24.176	1.000	130
38	5001	1	70	1	1	19.044	1.200	110
39	4375	0	61	1	1	32.079	1.273	128
40	4326	0	65	1	1	29.340	1.200	170

To see a list of variable names in the dataset use the command `names(dig40)` as shown below.

R commands:

```
names(dig40)
```

R output:

```
[1] "id"      "trtmt"  "age"    "race"   "sex"    "bmi"    "creat"  "sbp"
```

Suppose we are interested in seeing only the first 10 observations. This is easily accomplished using the command `dig40[1:10,]` as shown below.

R commands:

```
dig40[1:10, ]
```

R output:

	id	trtmt	age	race	sex	bmi	creat	sbp
1	4995	0	55	1	1	19.435	1.600	150
2	2312	0	78	2	1	22.503	2.682	104
3	896	0	50	1	1	27.406	1.300	140
4	3103	0	60	1	1	29.867	1.091	140
5	538	1	31	1	1	27.025	1.159	120
6	1426	0	70	1	1	19.040	1.250	150
7	4787	1	46	1	1	28.662	1.307	140
8	5663	0	59	2	1	27.406	1.705	152
9	1109	0	68	1	2	27.532	1.534	144
10	666	0	65	1	1	28.058	2.000	120

Program Note 3.1 - Tabulating Data

By displaying the entire **DIG40** data set, we are able to see the variables: **treatment**, **age**, **race**, **sex** and some other characteristics for the forty participants in the data set. However, there is also a need to summarize the information contained in the data set. For example, we may want to know how many males and females are in the **DIG40** data set. This would simply require the creation of a table displaying the frequency of males and females which is referred to as a one-way frequency table.

Below, we use the **R** function `table()` to show the frequency distribution for the variable **sex** from the **DIG40** data set.

```
R commands:
table(dig40$sex)

R output:
 1  2
30 10
```

The results from a two-way frequency table showing the cross-tabulation of **sex** and **race** are shown below.

```
R commands:
table(dig40$sex, dig40$race)

R output:
      1  2
1 28  2
2  7  3
```

We can use the following **R** commands to assign the label “Male” to the value “1” and the label “Female” to the value “2”.

```
R commands:
sex<- factor(dig40$sex, levels=c(1,2), labels=c("Male", "Female"))
table(sex)

R output:
sex
  Male Female
   30     10
```

We can use the following **R** commands to assign the label “White” to the value “1” and the label “Nonwhite” to the value “2”. See **Table 3.2 Frequencies of sex and race for 40 patients in DIG40** in the textbook.

```
R commands:

race<- factor(dig40$race, levels=c(1,2), labels=c("White","Nonwhite"))
table(race)

R output:

race
  White Nonwhite
    35         5
```

To create categories for the continuous variable `age`, we start by creating a new variable called `age.cat` as shown below. After assigning the values: 0, 1, 2, 3, and 4 to each of the age categories, we assign labels to the categorical values using the **R** function: `factor` as shown below. See **Table 3.3 Frequency of age groups for 40 patients in DIG40** in the textbook.

```
R commands:

age.cat<-NA
age.cat[dig40$age<40]<-0
age.cat[dig40$age>=40 & dig40$age<50]<-1
age.cat[dig40$age>=50 & dig40$age<60]<-2
age.cat[dig40$age>=60 & dig40$age<70]<-3
age.cat[dig40$age>=70]<-4

age.cat<- factor(age.cat, levels=c(0,1,2,3,4), labels=c("Under 40", "40-49", "50-59", "60-69", "70-79"))

table(age.cat)

R output:

age.cat
Under 40  40-49  50-59  60-69  70-79
      3      6      8      11     12
```

A cross-tabulation of `bmi.cat`, the categorical variable for body mass index, and the variable `sex` can be accomplished as shown below. See **Table 3.4 Cross-tabulation of body mass index and sex for 40 patients in DIG40 with column percentages in parentheses** in the textbook.

R commands:

```
bmi.cat<-NA
bmi.cat[dig40$bmi<18.5]<-0
bmi.cat[dig40$bmi>=18.5 & dig40$bmi<24.9]<-1
bmi.cat[dig40$bmi>=25.0 & dig40$bmi<29.9]<-2
bmi.cat[dig40$bmi>=30]<-3

bmi.cat<- factor(bmi.cat, levels=c(0,1,2,3), labels=c("Underweight",
"Normal","Overweight", "Obese"))

sex<- factor(dig40$sex, levels=c(1,2), labels=c("Male", "Female"))

table(bmi.cat, sex)
```

R output:

bmi.cat	sex	
	Male	Female
Underweight	1	0
Normal	10	2
Overweight	14	6
Obese	5	2

Program Note 3.2 - Creating Line graphs and Bar charts

1. Line graphs

In **Table 3.6**, we present health expenditures data as a percentage of GDP by year for Canada, the United Kingdom, and the United States. The data are entered as shown below. Use the **type** option to specify the desired plot. Options for plot types include following:

(Note: This information can be found by using the **R** command: **help(plot)**)

"p" for points

"l" for lines

"b" for both

"c" for lines part alone of "b"

"o" for both "over-plotted"

"h" for "histogram" like vertical lines

"s" for steps

"n" for no plotting

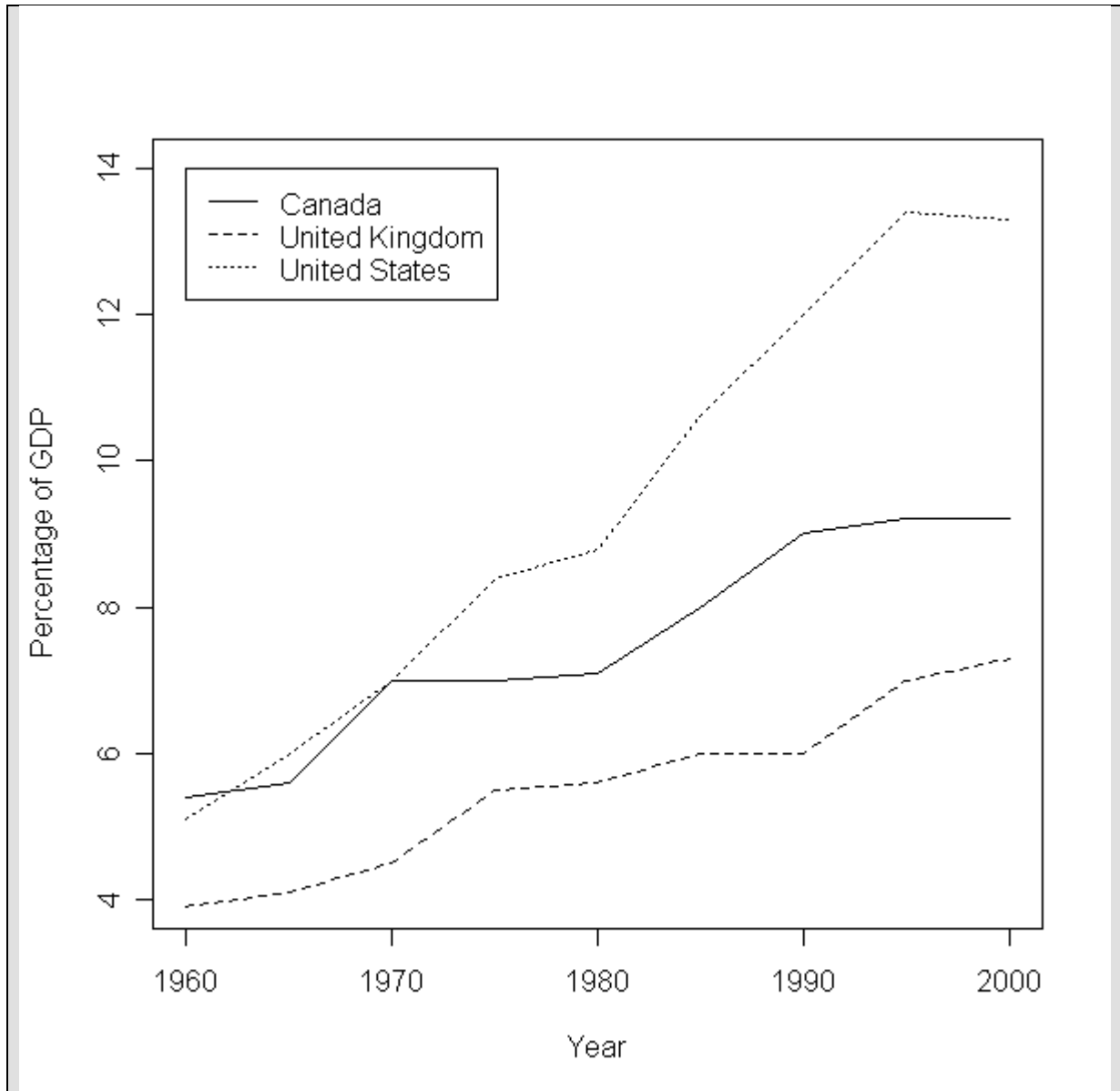
The option **lty** allows you to specify a particular line pattern. For example, the option **lty=1** will produce a solid line while the option **lty=2** produces a dashed line. The options **xlab** and **ylab** are used to provide titles for the x- and y-axes, and the options **xlim** and **ylim** are used to control the limits of the x- and y-axes. See **Figure 3.1 Line graph: Health expenditures as percentage of GDP for Canada, United Kingdom, and United States** in the textbook.

R commands:

```
Year<- c(1960, 1965, 1970, 1975, 1980, 1985, 1990, 1995, 2000)
Canada<- c(5.4, 5.6, 7.0, 7.0, 7.1, 8.0, 9.0, 9.2, 9.2)
United.Kingdom<- c(3.9, 4.1, 4.5, 5.5, 5.6, 6.0, 6.0, 7.0, 7.3)
United.States<- c(5.1, 6.0, 7.0, 8.4, 8.8, 10.6, 12.0, 13.4, 13.3)

plot(Canada ~ Year, type="l", lty=1, ylim=c(4,14), ylab="Percentage of
GDP"))
points(United.Kingdom ~ Year, type="l", lty=2)
points(United.States ~ Year, type="l", lty=3)
legend(1960,14, c("Canada", "United Kingdom", "United States"),
lty=c(1,2,3))
```

R output:



2. Bar charts

For example, the horizontal bar chart in **Figure 3.5** displays the proportion of patients with diabetes in different age groups from the **DIG200** data set. We use the **R** function `barplot` to create bar charts. The option `horiz` when set equal to `T` transforms a vertical bar chart into a horizontal bar chart. The option `names.arg` is used to specify names for each bar. Although unnecessary, the `col` option allows the user to specify colors for each bar. (**Note:** More information on creating bar charts can be obtained by running the **R** command: `help(barplot)`)

Before we get to the commands that create a bar chart, we begin by reading the data from the **DIG200** data set using the following commands:

R commands:

```
#Access the data directly from the book's companion website

dig200 <- read.table('http://www.biostat-edu.com/files/DIG200.txt', header=T)
dig200

#Or from a directory on your computer

dig200 <- read.table('C:/DIG200.txt', header=T)

#-----#

#Next, use the commands below to create the appropriate age categories:

age.cat<-NA
age.cat[dig200$age< 40]<-0
age.cat[dig200$age>=40 & dig200$age<50]<-1
age.cat[dig200$age>=50 & dig200$age<60]<-2
age.cat[dig200$age>=60 & dig200$age<70]<-3
age.cat[dig200$age>=70]<-4

#Finally, use the commands below to obtain the number of patients with
#diabetes in each age category:

table(dig200$diabetes, age.cat)

#The output should look as follows:
```

R output:

```
  age.cat
  0  1  2  3  4
0  5 17 39 50 35
1  2  6 13 16 17
```

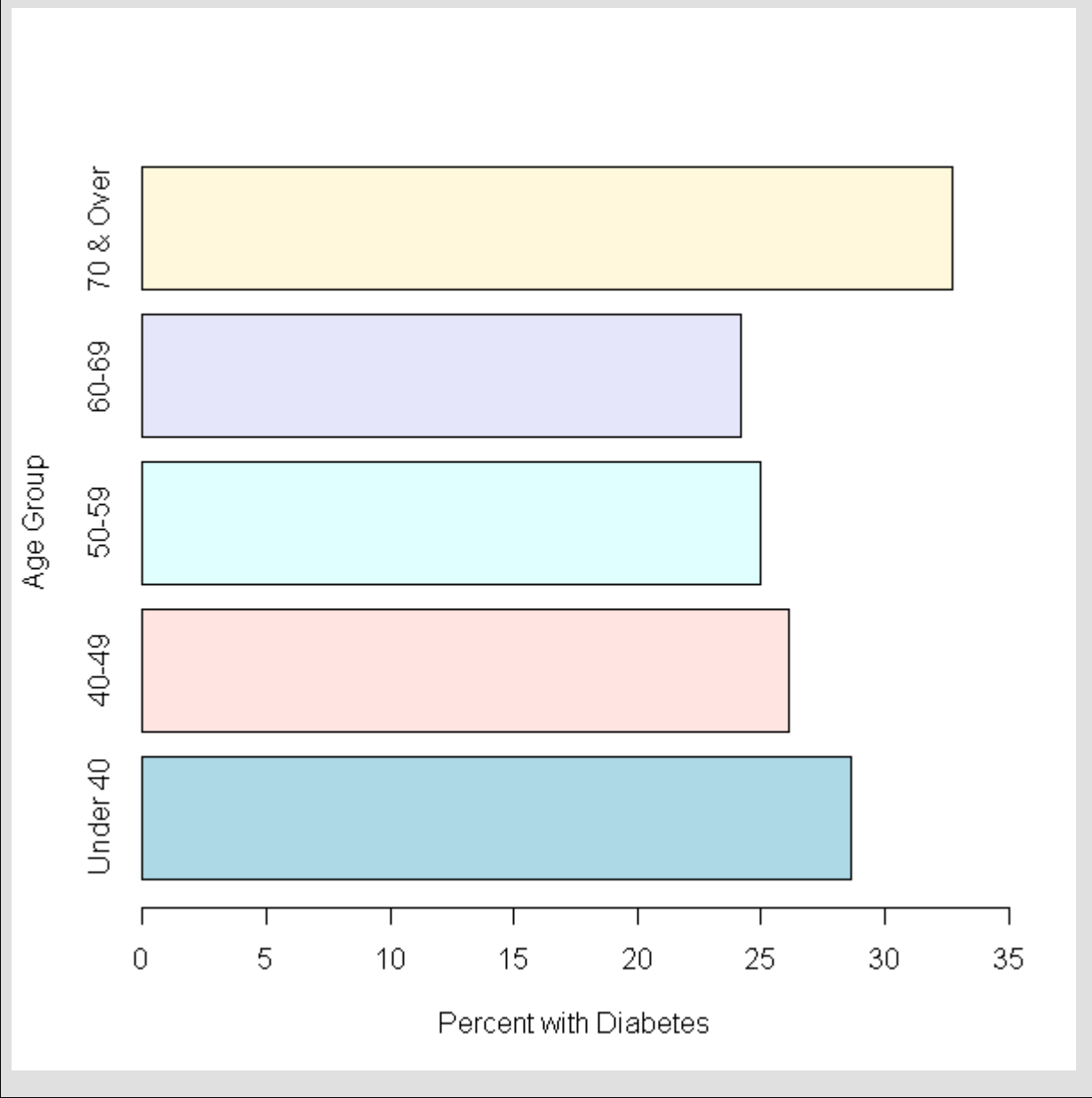
Using the output, we calculated percentages by evaluating each of the following values: $2/7$, $6/23$, $13/52$, $16/66$, and $17/52$. With this information, we are now able to create the variable **percent** that contains the proportion of people with diabetes in each age category. See **Figure 3.5 Bar chart showing proportion of people in each age group with diabetes, DIG200** in the textbook.

R commands:

```
percent<-c(28.6, 26.1, 25.0, 24.2, 32.7)

barplot(percent, horiz=T,
names.arg=c("Under 40", "40-49", "50-59", "60-69", "70 & Over"),
xlab="Percent with Diabetes", xlim=c(0, 35), ylab="Age Group")
col=c("lightblue", "mistyrose", "lightcyan", "lavender", "cornsilk"))
```

R output:



Program Note 3.3 – Creating histograms

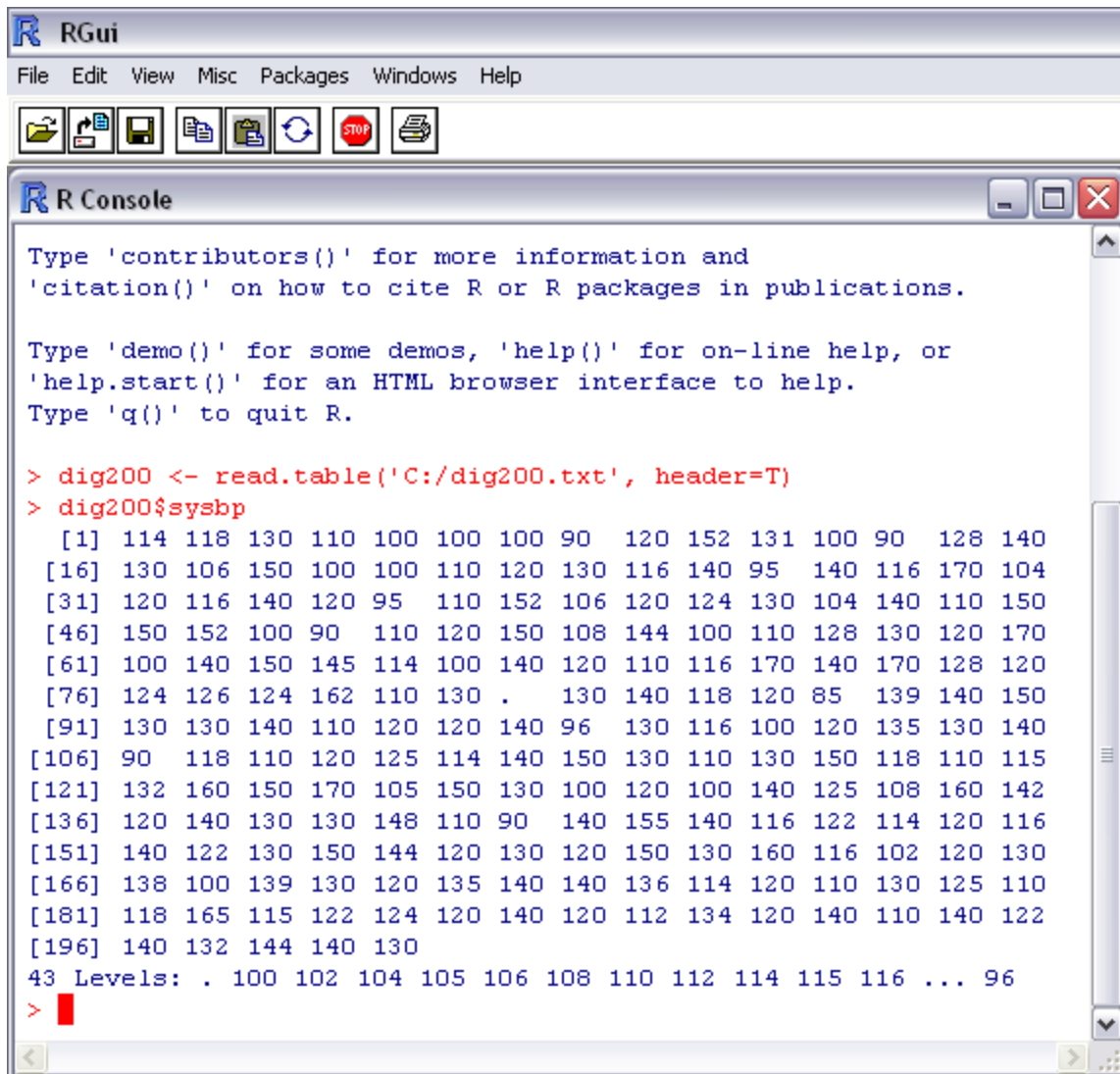
Before we begin creating histograms, we would like to comment on our experience with **R** in executing what would be considered simple data management in other statistical software packages. For example, when we attempted to create the histogram for the systolic blood pressure readings of participants in the **DIG200** data set, the procedure was not straightforward. To begin with, the variable **sysbp** in the **DIG200** data set contains a missing observation denoted by a period. You can see this by running the **R** commands below:

```
R commands:
#Access the data directly from the book's companion website
dig200 <- read.table('http://www.biostat-edu.com/files/DIG200.txt', header=T)
dig200

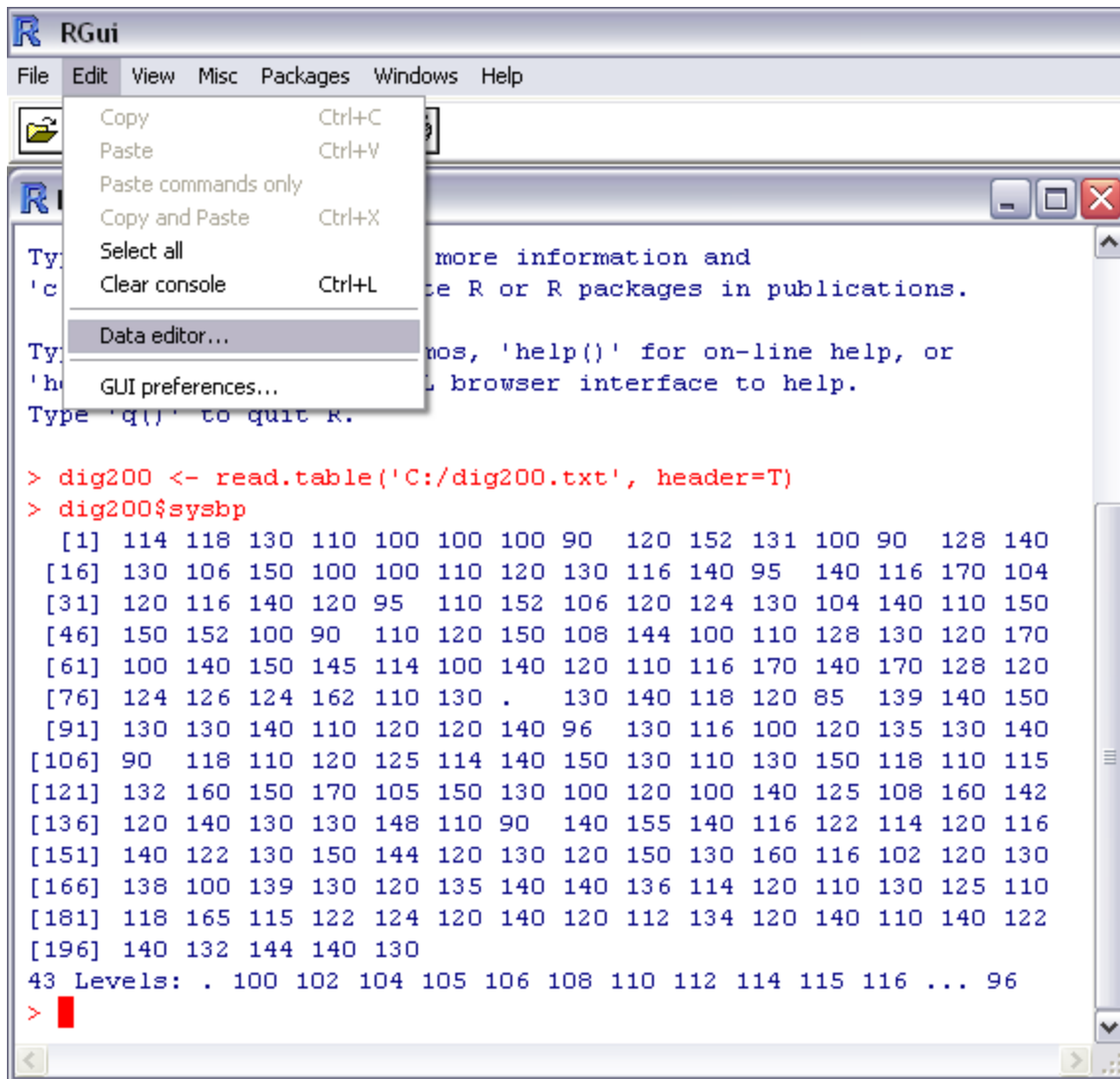
#Or from a directory on your computer
dig200 <- read.table('C:/DIG200.txt', header=T)
dig200$sysbp

#The commands above produce the following output:
R output:
 [1] 114 118 130 110 100 100 100 90 120 152 131 100 90 128 140 130 106 150
[19] 100 100 110 120 130 116 140 95 140 116 170 104 120 116 140 120 95 110
[37] 152 106 120 124 130 104 140 110 150 150 152 100 90 110 120 150 108 144
[55] 100 110 128 130 120 170 100 140 150 145 114 100 140 120 110 116 170 140
[73] 170 128 120 124 126 124 162 110 130 . 130 140 118 120 85 139 140 150
[91] 130 130 140 110 120 120 140 96 130 116 100 120 135 130 140 90 118 110
[109] 120 125 114 140 150 130 110 130 150 118 110 115 132 160 150 170 105 150
[127] 130 100 120 100 140 125 108 160 142 120 140 130 130 148 110 90 140 155
[145] 140 116 122 114 120 116 140 122 130 150 144 120 130 120 150 130 160 116
[163] 102 120 130 138 100 139 130 120 135 140 140 136 114 120 110 130 125 110
[181] 118 165 115 122 124 120 140 120 112 134 120 140 110 140 122 140 132 144
[199] 140 130
43 Levels: . 100 102 104 105 106 108 110 112 114 115 116 118 120 122 ... 96
```

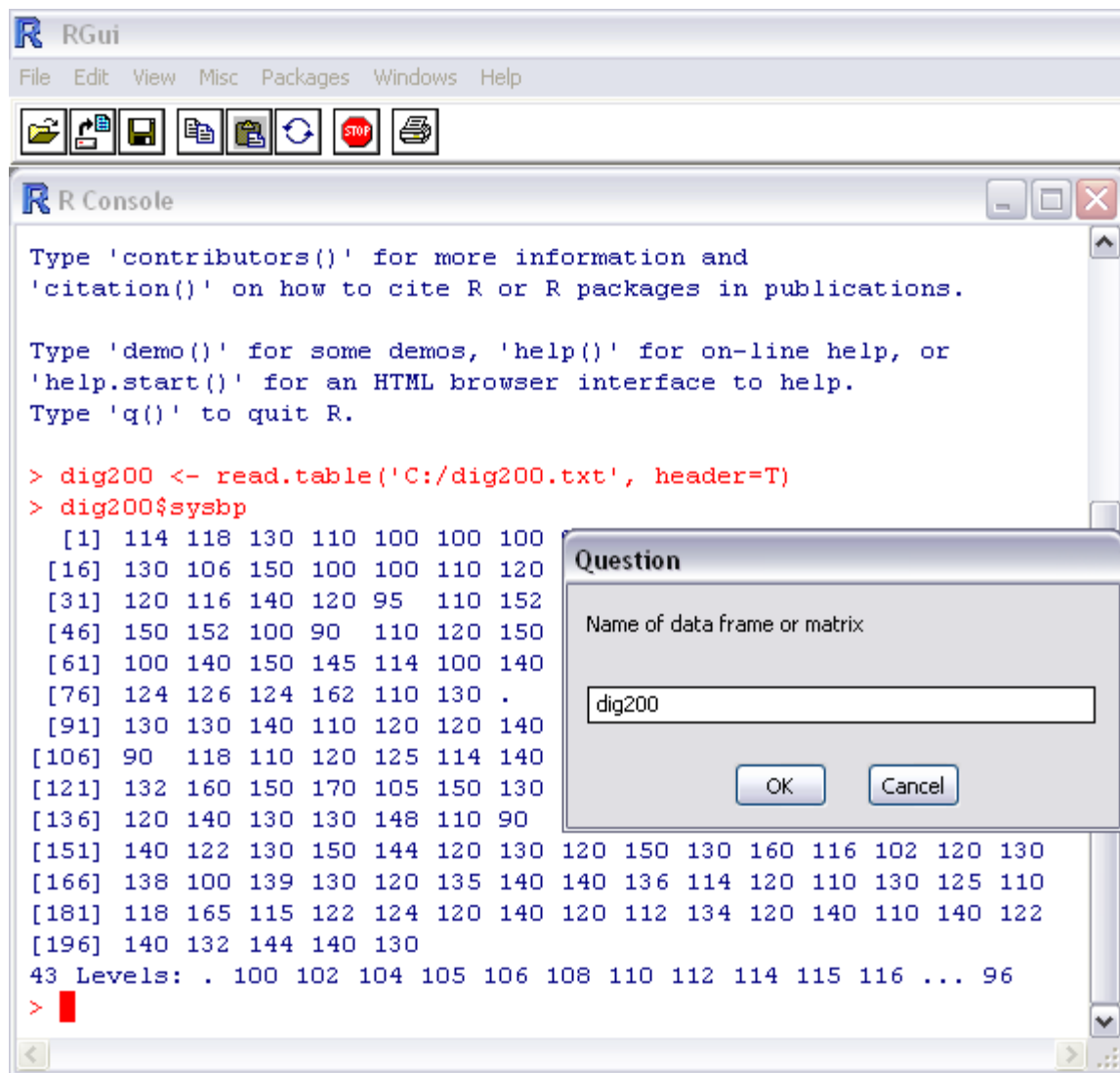
Notice that observation 82 is a “.” indicating that the value for observation 82 is missing. Below are the same commands we have already shown you; however, we have used screenshots to show how the output should appear in **R**.



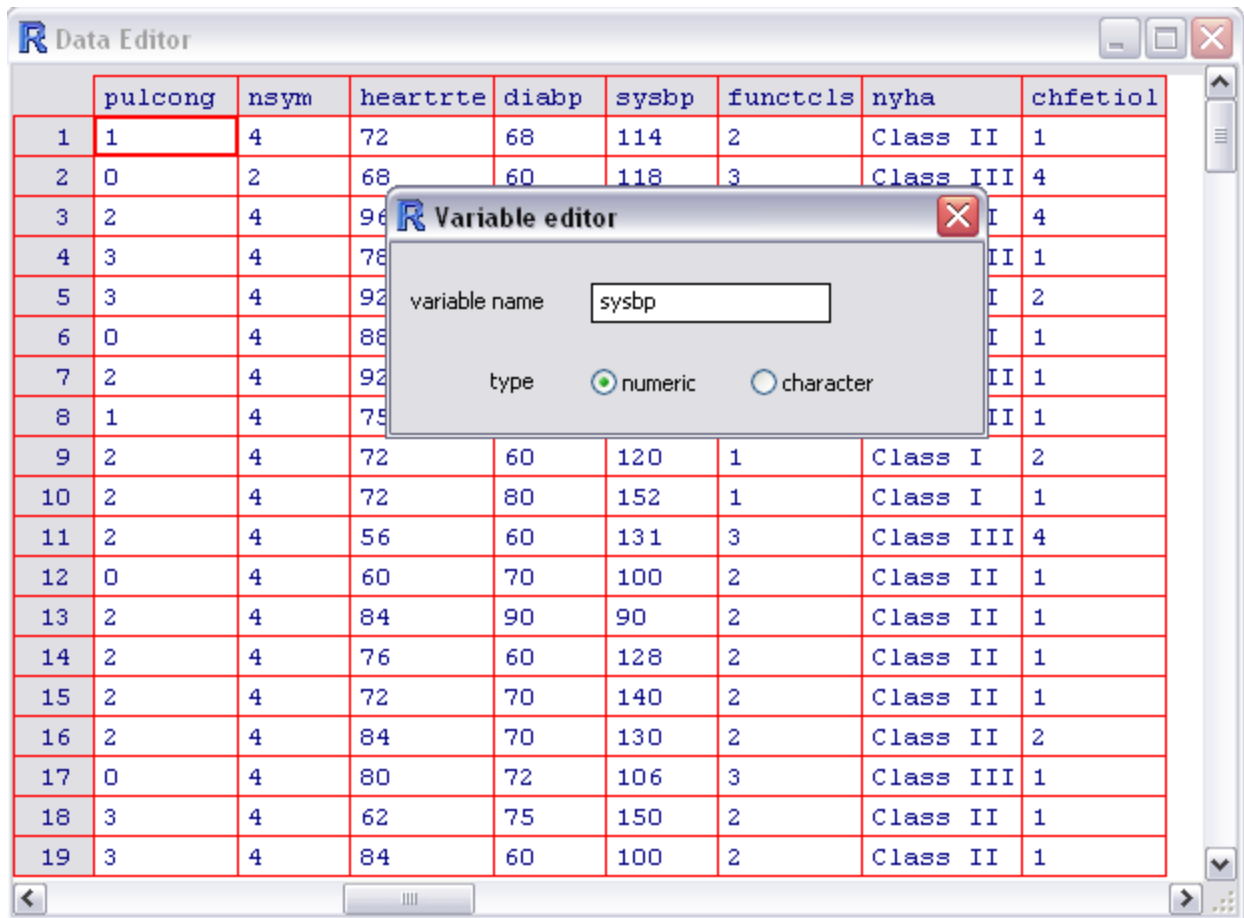
The next few screenshots are used to show how you can remedy the missing value problem by using the **Data editor...** under the **Edit** option in R's menu bar.



At this point, the **Question** box will appear asking for the name of the data set you would like to access as shown below.



After clicking **OK**, the **R Data Editor** will launch. Search for the variable `sysbp` and double-click on the column name in the gray area to launch the **R Variable editor**. Now, you can change the type from character to numeric as shown below.

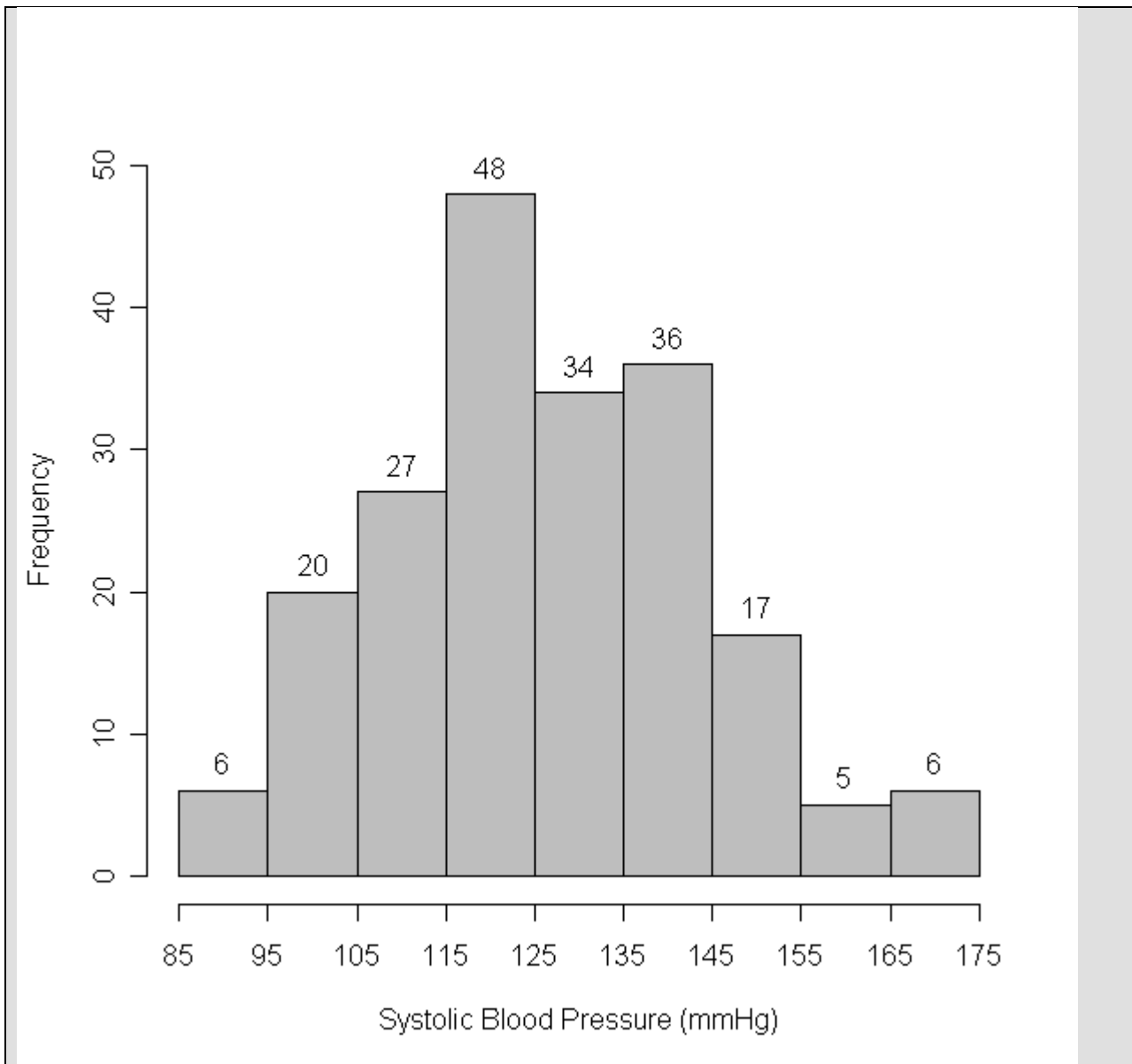


Finally, proceed with the following R commands to create a frequency histogram. See **Figure 3.8 Histogram of 199 systolic blood pressure values using 9 intervals of size 10 starting at 85** in the textbook.

R commands:

```
syshist<-hist(dig200$sysbp, breaks=c(85,95,105,115,125,135,145,155,165,175),
  ylim=c(0,50), xlab="Systolic Blood Pressure (mmHg)", main=" ", axes=F,
  right=F, col="gray")
axis(1, at=c(85,95,105,115,125,135,145,155,165,175))
axis(2, at=c(0,10,20,30,40,50))
text(syshist$mids, syshist$count+2, label=c(syshist$count))
```

R output:



Program Note 3.4 – Creating stem and leaf plots and scatter plots

1. Stem and Leaf plots

Using the **DIG40** data set, a stem and leaf plot for systolic blood pressure readings can be created with **R** as shown below.

R commands:

```
stem(dig40$sbp)
```

R output:

```
The decimal point is 1 digit(s) to the right of the |
```

```
10 | 0045  
11 | 05666  
12 | 0002488  
13 | 00000048  
14 | 000000044  
15 | 00002  
16 |  
17 | 00
```

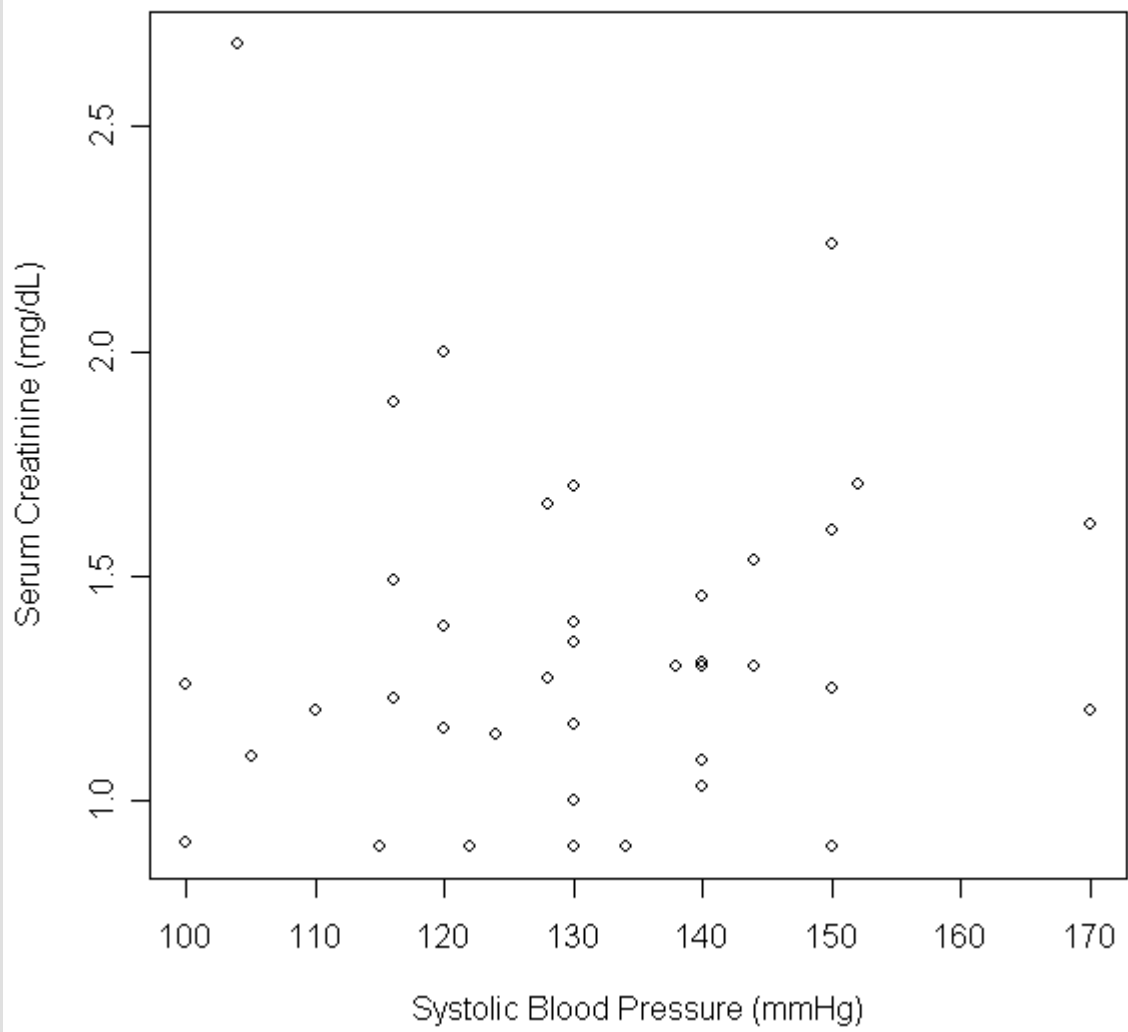
2. Scatter plots

In **Figure 3.12**, we use a scatter plot to examine the relationship between serum creatinine and systolic blood pressure using the **DIG40** data set.

R commands:

```
plot(dig40$creat~dig40$sbp,xlab="Systolic Blood Pressure (mmHg)", ylab="Serum  
Creatinine (mg/dL)")
```

R output:



Program Note 3.5 – Descriptive statistics and creating box plots

1. Descriptive Statistics

R can be used to get the mean, standard deviation, median, and range for systolic blood pressure for patients from the **DIG40** data set.

R commands:

```
mean(dig40$sbp)
sd(dig40$sbp)
median(dig40$sbp)
min(dig40$sbp)
max(dig40$sbp)
```

R output:

```
> mean(dig40$sbp)
[1] 131.4
> sd(dig40$sbp)
[1] 16.87024
> median(dig40$sbp)
[1] 130
> min(dig40$sbp)
[1] 100
> max(dig40$sbp)
[1] 170
```

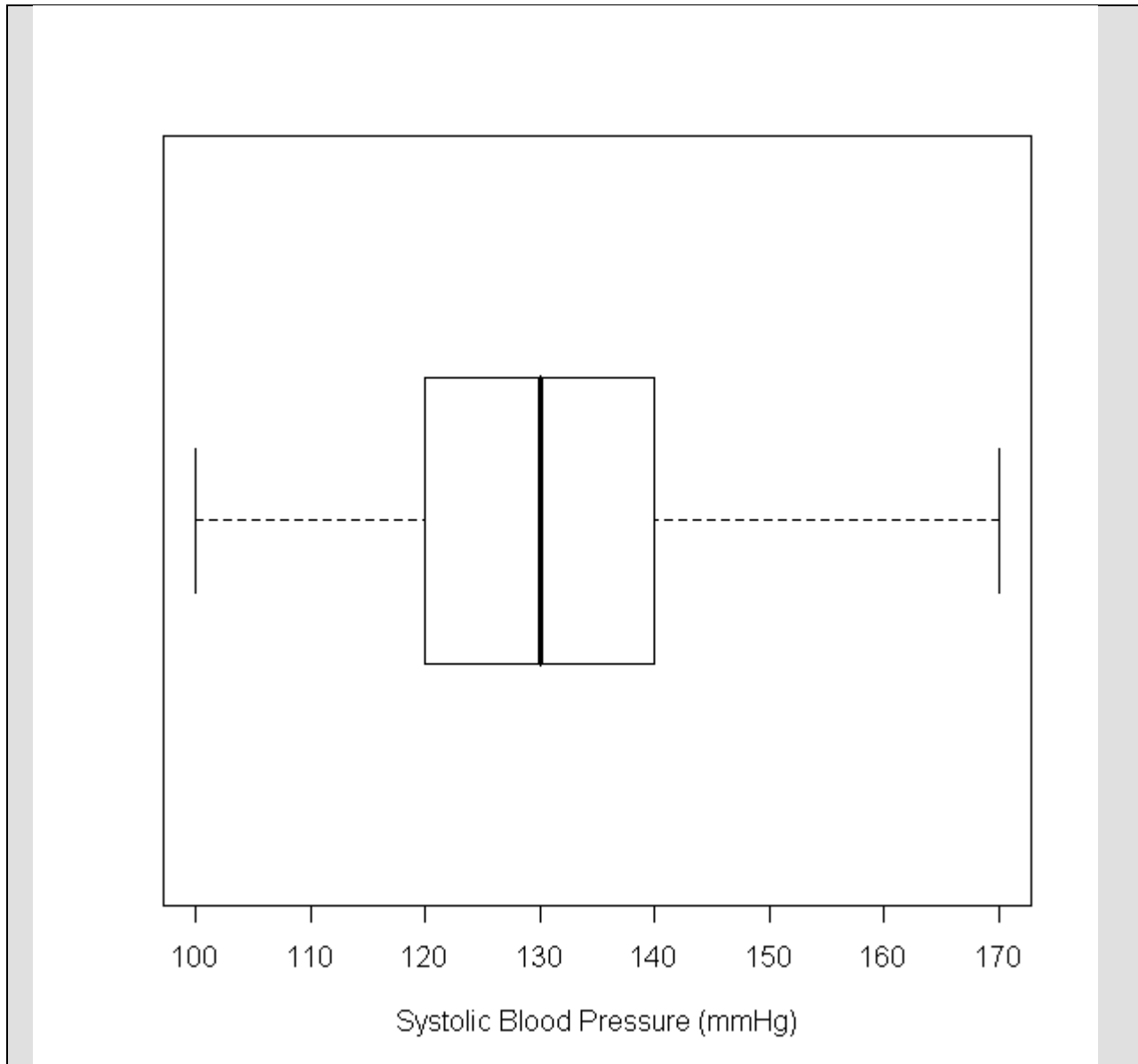
2. Box plots

Below we show the R commands used to create the box plot like the one shown in **Figure 3.14** in the textbook using the **DIG40** data set.

R commands:

```
dig40 <- read.table('C:/dig40.txt', header=T)
boxplot(dig40$sbp, ylab="Systolic Blood Pressure (mmHg)")
```

R output:



One way to create the box plots displayed in **Example 3.13** in the textbook is shown below.

R commands:

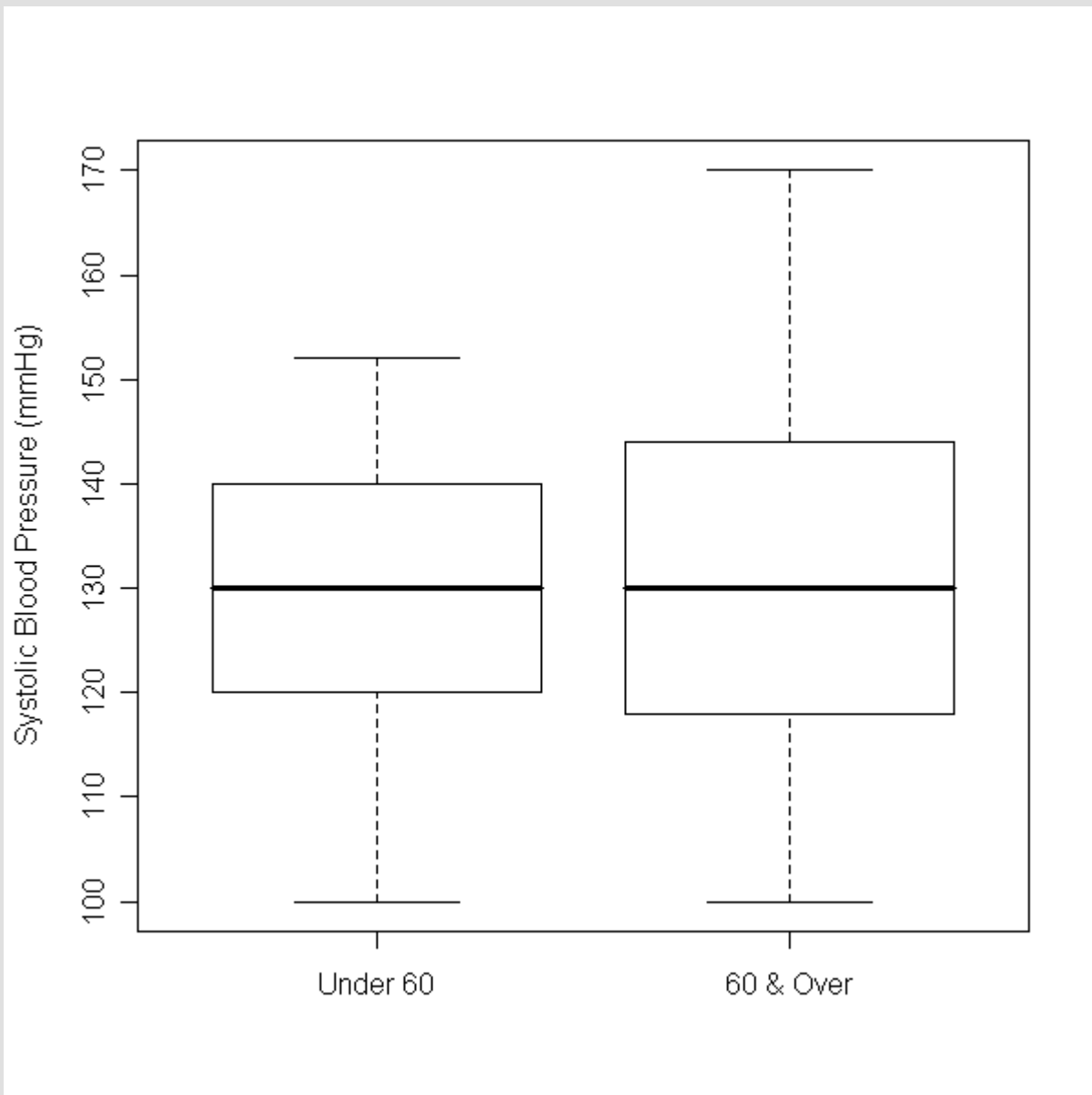
```
#Access the data directly from the book's companion website
dig40 <- read.table('http://www.biostat-edu.com/files/dig40.txt', header=T)

age.cat<-NA
age.cat[dig40$age<60]<-0
age.cat[dig40$age>=60]<-1

age.cat<- factor(age.cat, levels=c(0,1), labels=c("Under 60", "60 & Over"))

boxplot(dig40$sbp~age.cat,ylab="Systolic Blood Pressure (mmHg)")
```

R Output:



Program Note 3.6 – Calculating Pearson and Spearman correlation coefficients

Below are the data used in **Example 3.18** in the textbook:

```
Systolic blood pressure: 120 118 130 140 140 128 140 140 120 128 124 135
Diastolic blood pressure: 60 60 68 90 80 75 94 80 60 80 70 85
```

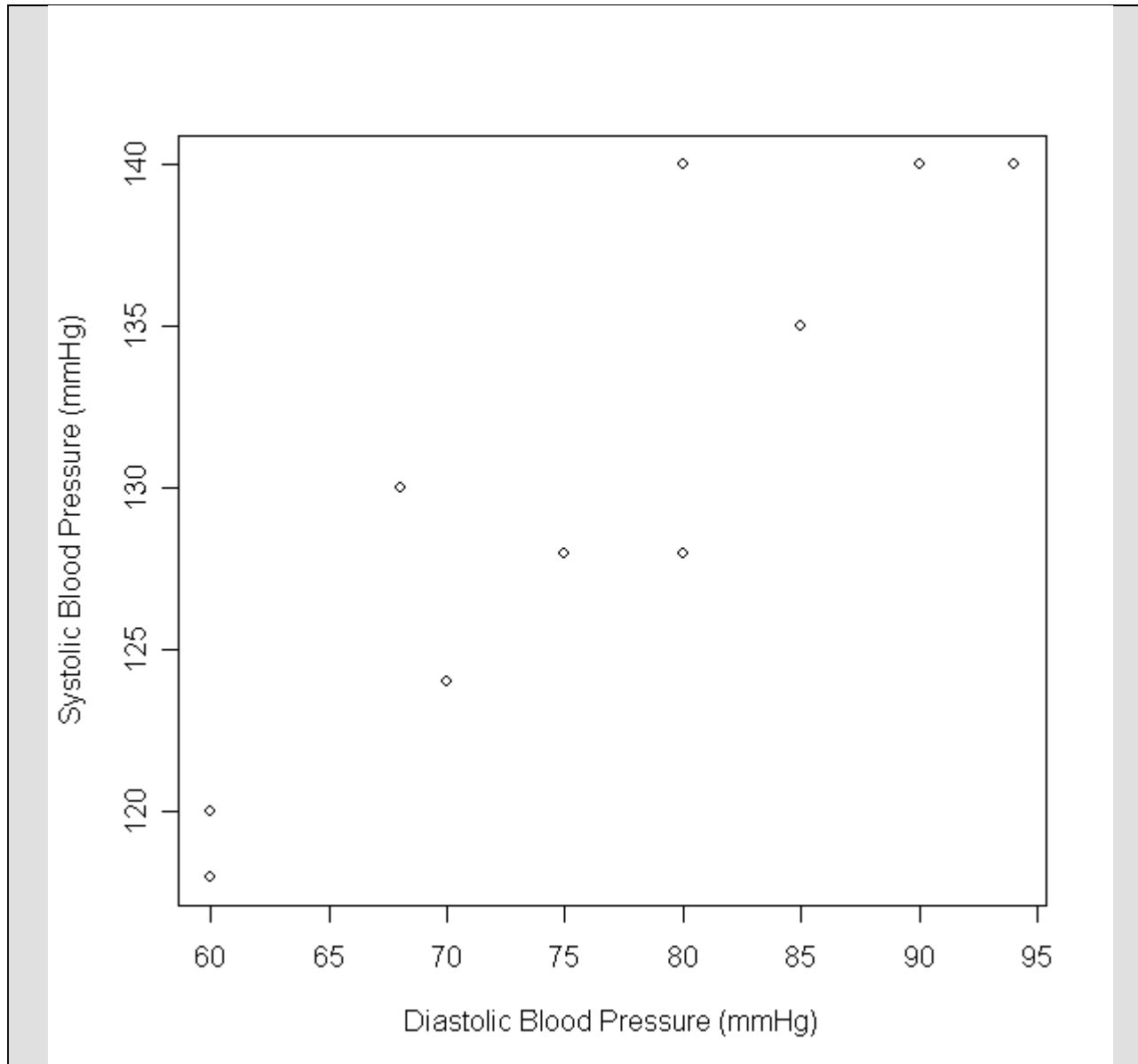
The **R** commands below show how the blood pressure data can be read directly into **R** along with the commands to create a scatter plot like the one presented in **Example 3.18** in textbook.

R commands:

```
sysbp<- c(120, 118, 130, 140, 140, 128, 140, 140, 120, 128, 124, 135)
diabp<- c(60, 60, 68, 90, 80, 75, 94, 80, 60, 80, 70, 85)

# Scatter plot of systolic blood pressure versus diastolic blood pressure
plot(diabp,sysbp,xlab="Diastolic Blood Pressure (mmHg)",
     ylab="Systolic Blood Pressure (mmHg)")
```

R output:



The **R** commands below are used to calculate both Pearson's correlation coefficient and Spearman's correlation coefficient. See **Example 3.20** in textbook.

R commands :

```
sysbp<- c(120, 118, 130, 140, 140, 128, 140, 140, 120, 128, 124, 135)
diabp<- c(60, 60, 68, 90, 80, 75, 94, 80, 60, 80, 70, 85)

# Pearson correlation coefficient
cor.test(diabp, sysbp)

# Spearman correlation coefficient
cor.test(diabp, sysbp, method="spearman")
```

R output:

Pearson's product-moment correlation

```
data: diabp and sysbp
t = 6.2946, df = 10, p-value = 8.971e-05
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.6561730 0.9700242
sample estimates:
      cor
0.8935738
```

Spearman's rank correlation rho

```
data: diabp and sysbp
S = 38.3341, p-value = 0.0002709
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.8659648
```