# R Workshop Module 3: Plotting Data

Katherine Thompson (katherine.thompson@uky.edu)
Department of Statistics, University of Kentucky

October 15, 2013

## Reading in Data

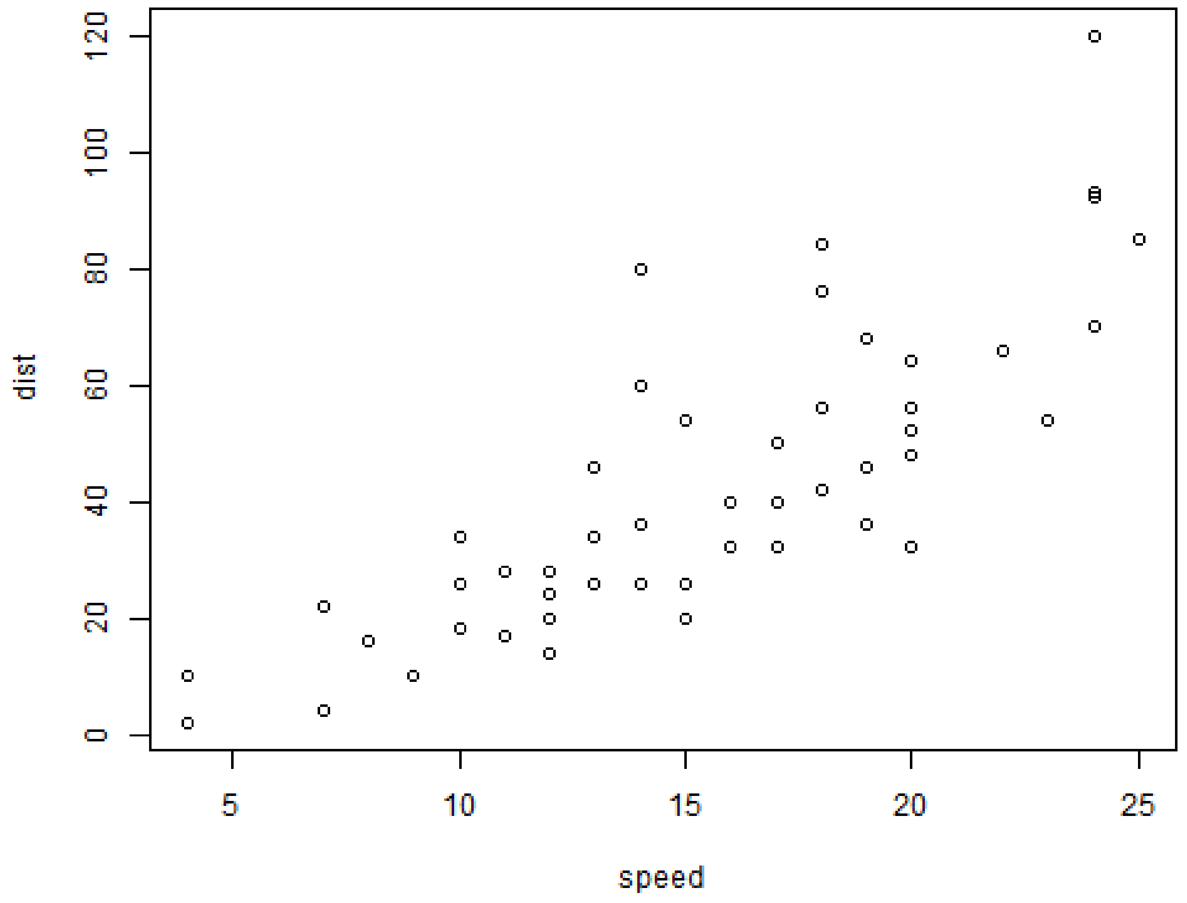
Start by reading the dataset ’practicedata.txt’ into R. (Having trouble? See the instructions in Module 2.)

## Plotting One Quantitative Variable

For this example, we will plot the variable `respvar` from the data set, `practicedata`. Remember than by using a ’$’, we can refer to the variable as `practicedata$respvar` in the following code.

**Histograms:** One way to plot quantitative data is using a histogram.

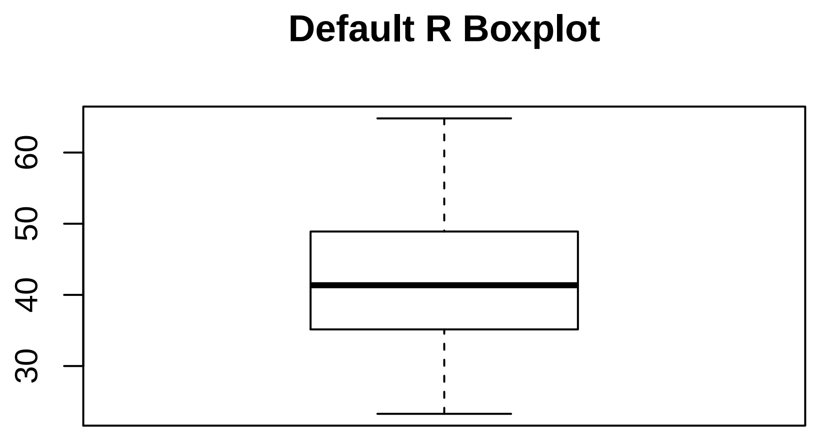
```
hist(practicedata$respvar, # what the histogram is plotting
     main='Histogram of Response Variable', # change the main axis title
     xlab='Response Variable', # change the x-axis label
     # freq=TRUE # histogram is of counts
     # breaks="Sturges", # this can be changed to specify a series of points
          # for the breaks in the histogram
     # xlim=c(20,70), # sets the limits of the x-axis
     # ylim=c(0,25), # sets the limits of the y-axis
     )
```



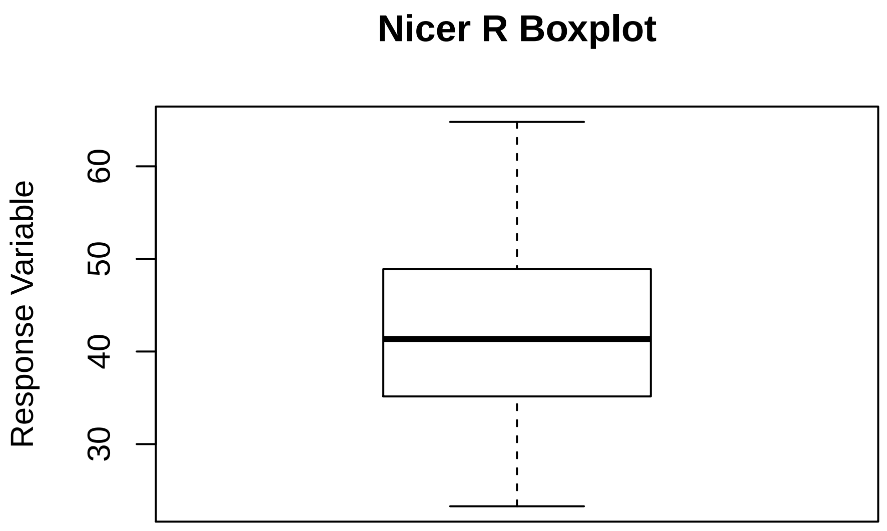
dist
0
20
40
60
80
100
120
5
10
15
20
25
speed

**Boxplots:** Another way to plot quantitative data is using a boxplot.

```
boxplot(practicedata$respvar, # what the boxplot is plotting
     main='Default R Boxplot' # change the main title
     )
```



```
boxplot(practicedata$respvar, # what the boxplot is plotting
     main='Nicer R Boxplot', # change the main title
     ylab='Response Variable', # change the y-axis label
     names='All Data', # change the name under the boxplot
     outline=TRUE # Draw outliers if there are any in the data
     )
```



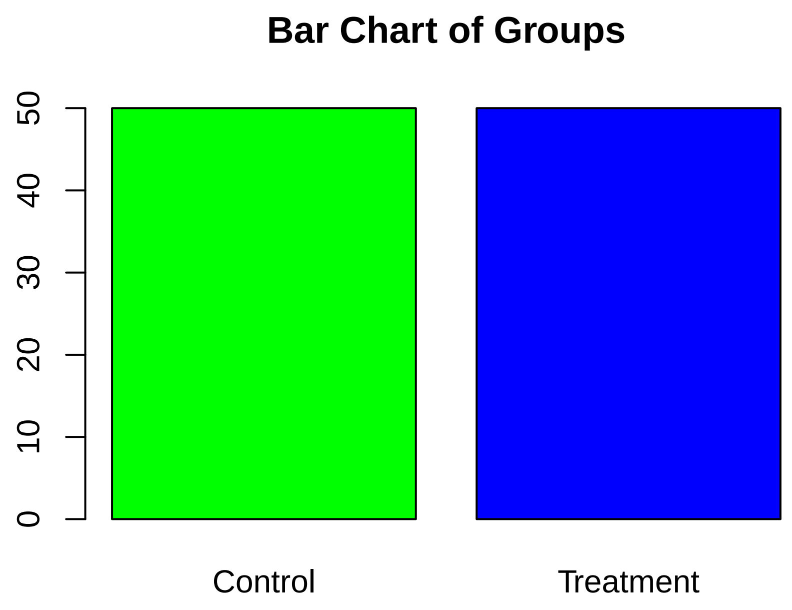
## Plotting One Categorical Variable

To practice plotting one variable at a time, we will plot `groupvar` from `practicedata`. We'll see what values this variable takes and then plot the data. (You'll notice that the variable has equal numbers of controls and treatments, so it's not very interesting, but here are the code to make a bar chart and a pie chart for you to use in other situations.)

```
practicedata$groupvar
```

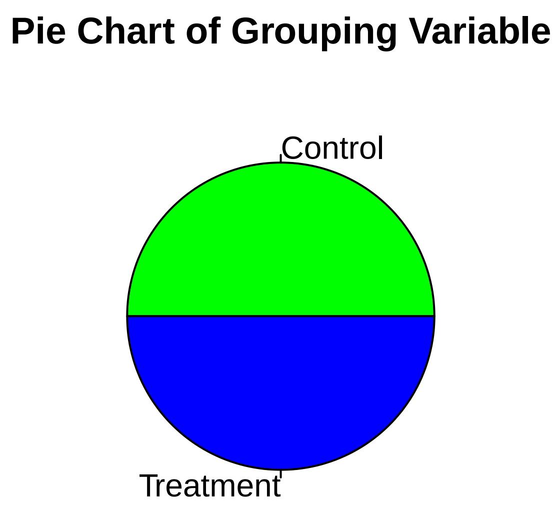
**Bar Chart:** If you have a categorical variable in one column in your data set (as we do here with `groupvar`), R requires a table of the counts for that variable to create a bar chart.

```
plot.group<-table(practicedata$groupvar) # Create table of counts
plot.group  # Look at the table

##
##   Control Treatment
##        50        50

barplot(plot.group, # Bar chart of the variable
        main='Bar Chart of Groups', # change main title
        col=c('green','blue') # change color of each bar
        )
```



**Pie Chart:** As in the case of bar charts, if you have a categorical variable in one column in your data set (as we do here with `groupvar`), R can use a table of the counts for that variable to create a pie chart.

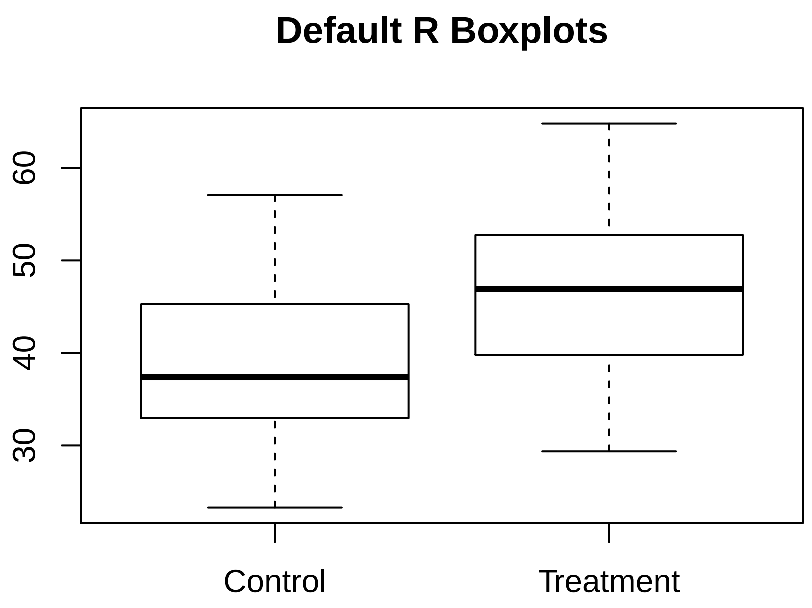
```
plot.group<-table(practicedata$groupvar) # Create table of counts
plot.group  # Look at the table

##
##   Control Treatment
##        50        50

pie(plot.group, # Pie chart of the variable
    main='Pie Chart of Grouping Variable', # change main title
    col=c('green','blue')) # change the color of each slice of the pie
```



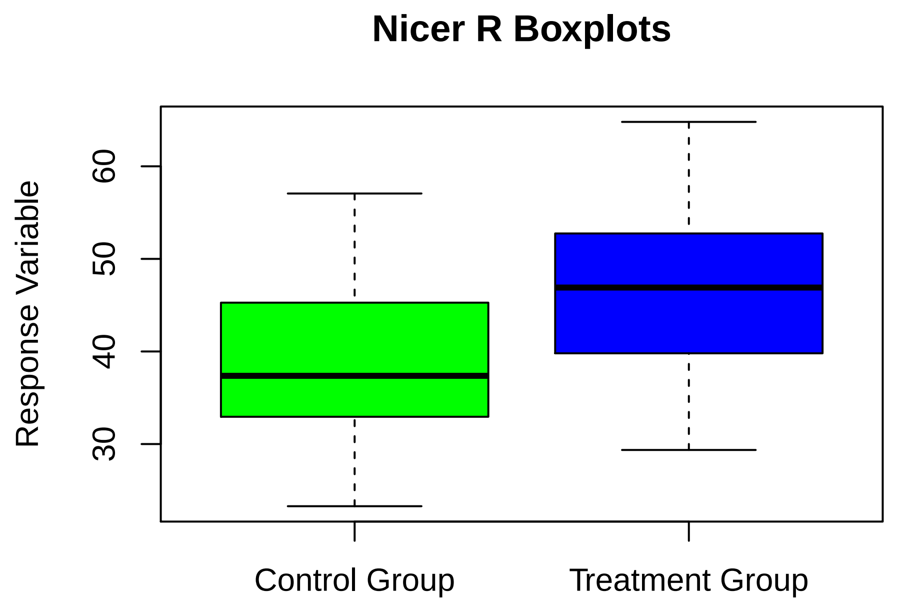
## Plotting Two Variables

**Plotting One Quantitative and One Categorical Variable:** If you have one quantitative variable that you want to compare across two (or many) groups, you can use two boxplots or two histograms.

```r
boxplot(practicedata$respvar ~ practicedata$groupvar, main = "Default R Boxplots")
```

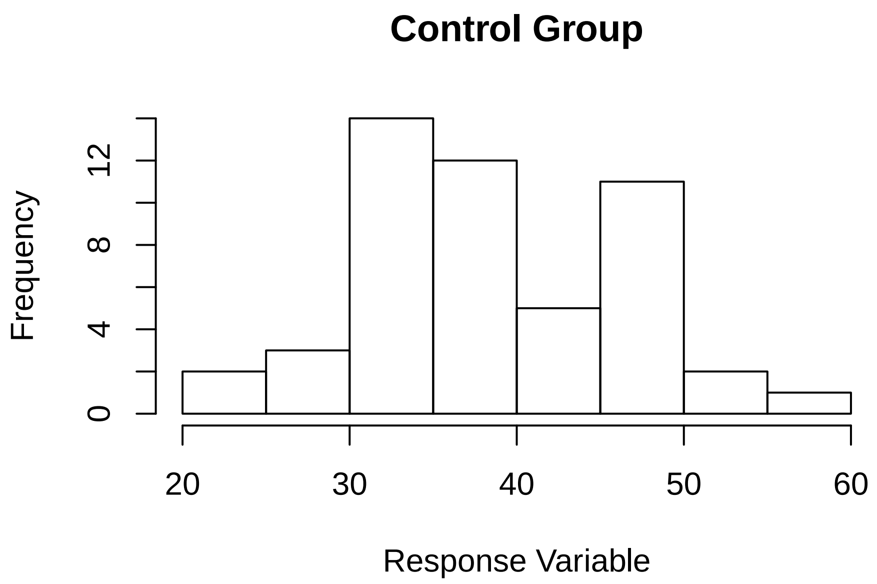


```r
boxplot(practicedata$respvar~practicedata$groupvar,
        main='Nicer R Boxplots',
        names=c('Control Group', 'Treatment Group'),
        col=c('green','blue'), # change color of boxes
        ylab='Response Variable' # change y-axis label
        # horizontal=TRUE # change boxplots to horizontal
)
```

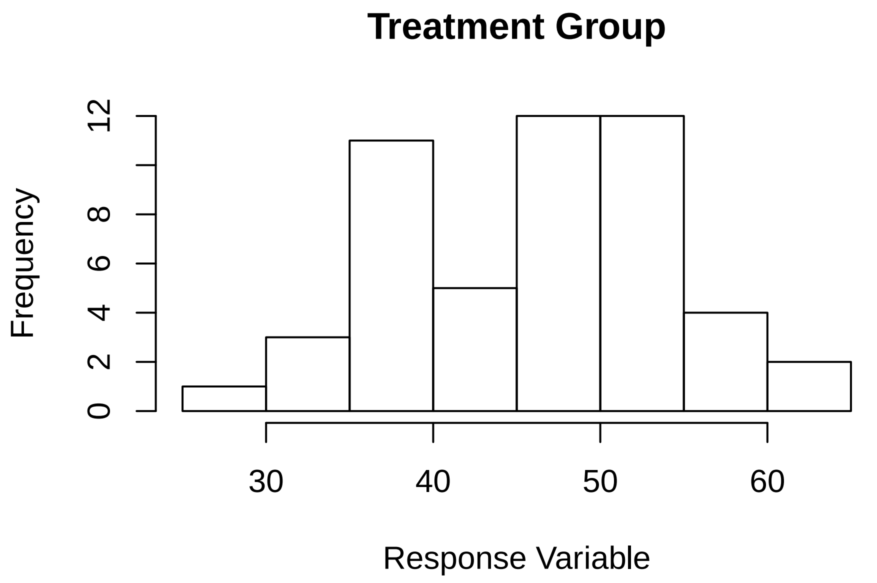


To create two histograms, we need to separate the observations for control and treatment individuals. These two variables are sometimes called "dummy variables." We are lucky that the control individuals are the first 50 in the data set, but there are other ways to separate data that will keep you from sorting data in another program.

```
#### Create a two data subsets for control and treatment individuals
n.controls=50
n.treatments=50
# Extract the first n.controls rows from the data
controldata=practicedata[1:n.controls,]
# Extract everything except the first n.controls rows from the data
treatmentdata=practicedata[-(1:n.controls),]
#### Histograms for each group
hist(controldata$respvar,
    main='Control Group', # change the main title
    xlab='Response Variable'
    )
```

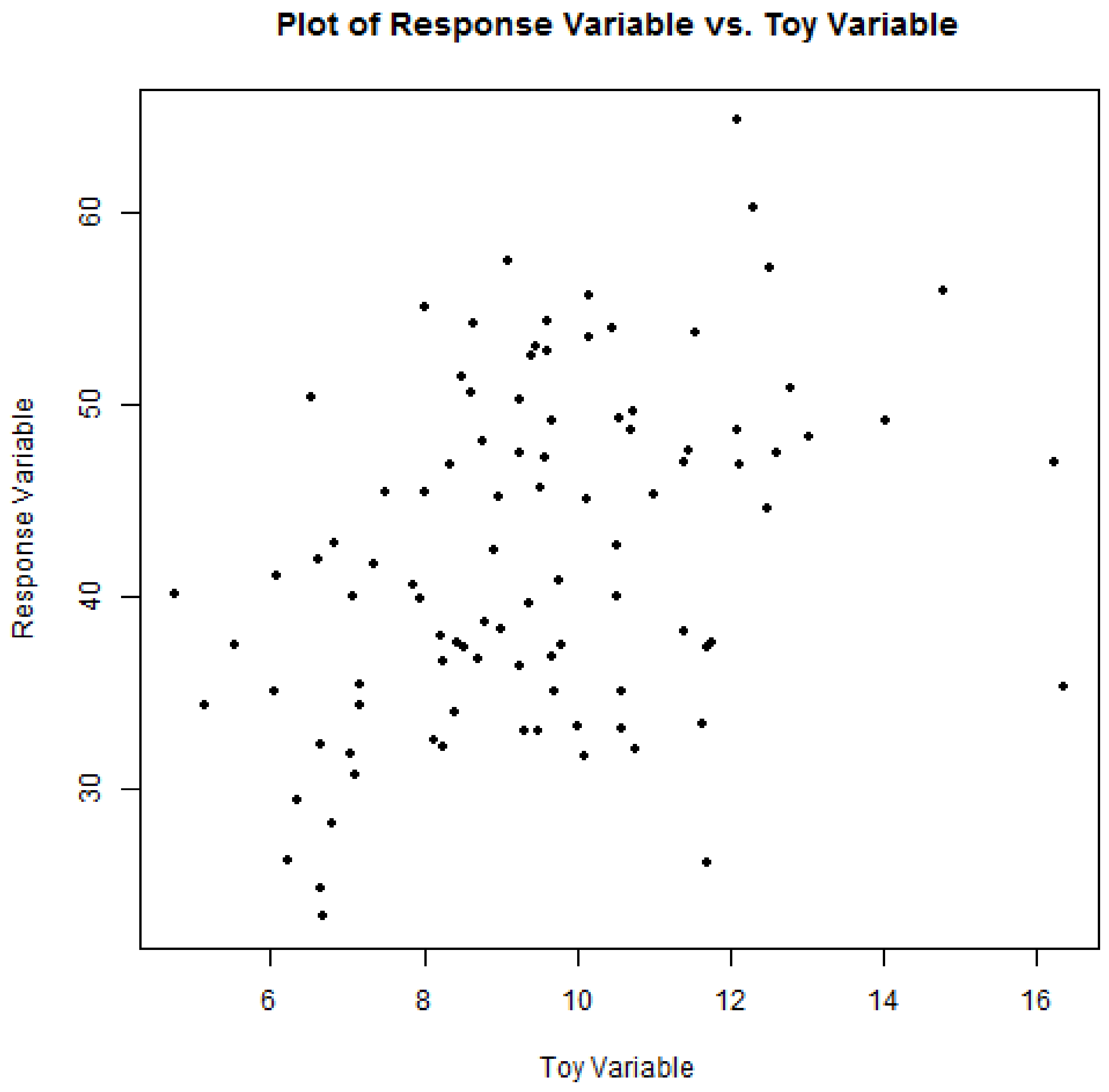


```r
hist(treatmentdata$respvar,
    main='Treatment Group', # change the main title
    xlab='Response Variable'
    )
```

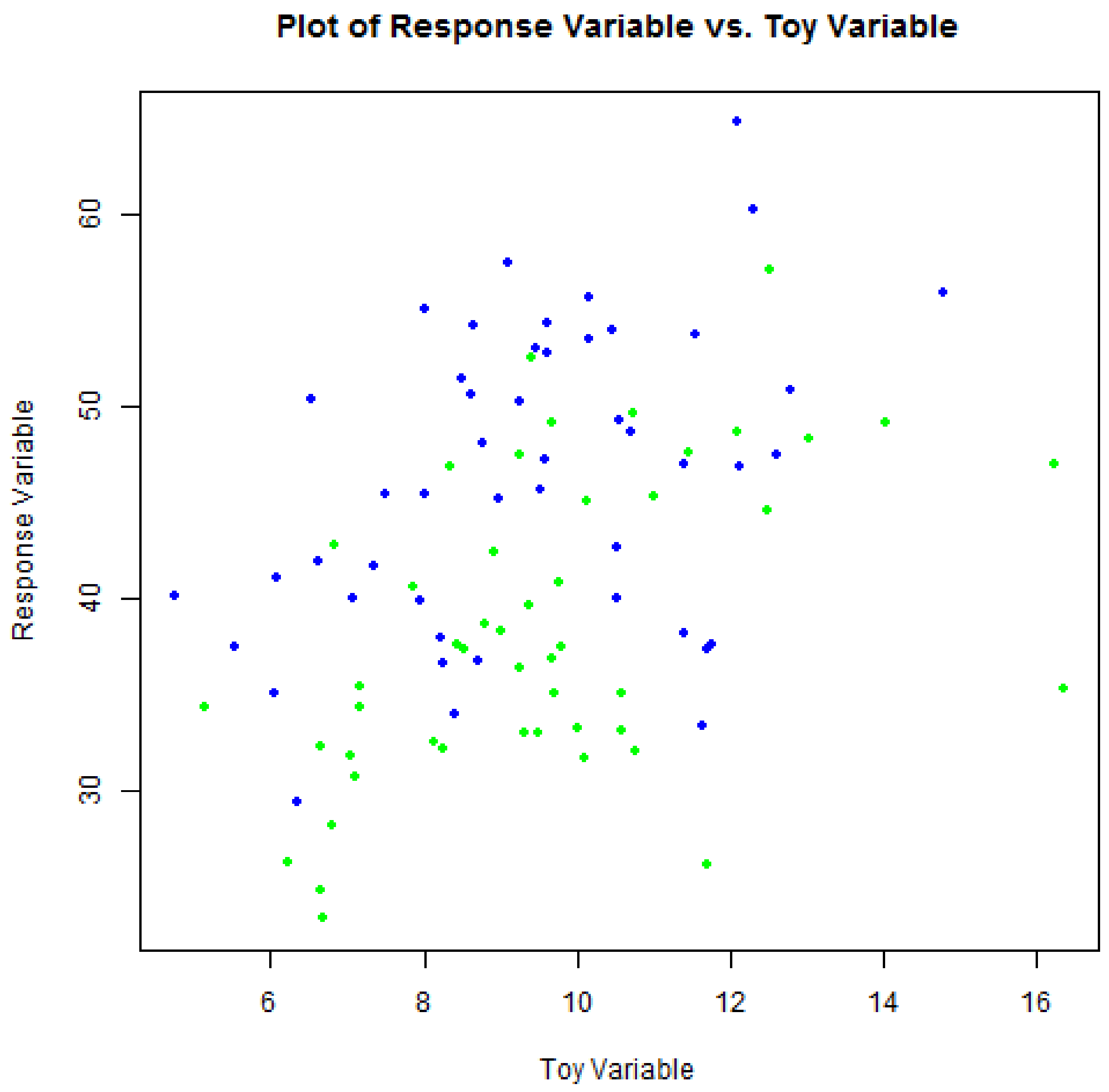


**Scatterplots:** If you have two quantitative variables that you want to compare, a scatterplot is a good way to explore the relationship. Here, we will investigate the relationship between `practicedata$respvar` and `practicedata$toyvar`.

```r
plot(practicedata$toyvar,practicedata$respvar, # x variable, y variable
    main="Default R Scatterplot")
```



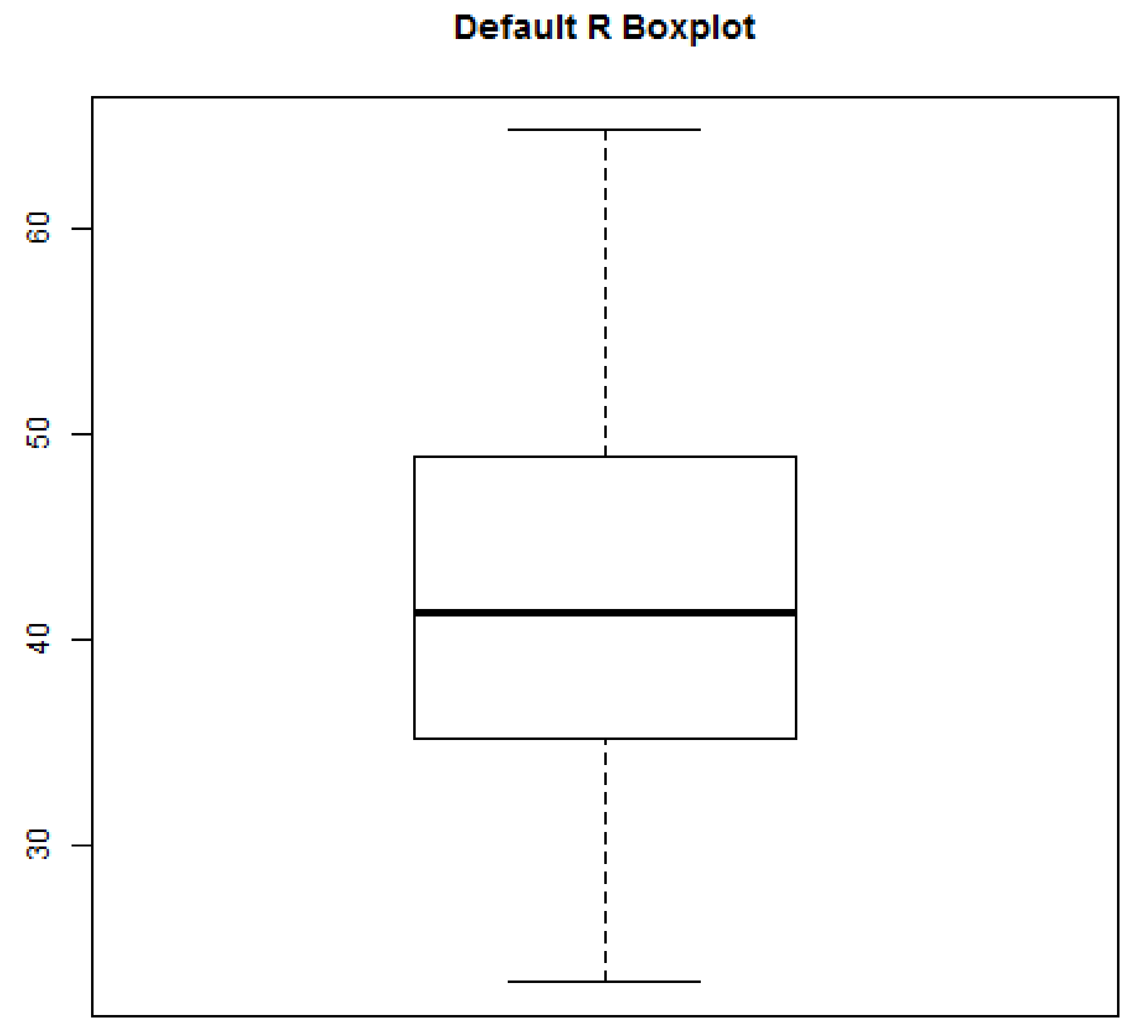
```r
plot(practicedata$toyvar,practicedata$respvar, # x variable, y variable
    main="Plot of Response Variable vs. Toy Variable", # change main label
    ylab='Response Variable', # change y-axis label
    xlab='Toy Variable', # change x-axis label
    pch=20, # change the plotting symbol
    # type='l', # instead of pch, you can create a line plot
                # (make sure your x's are ordered if you do this.)
    # col='black' # change color of plotting symbol
     )
```



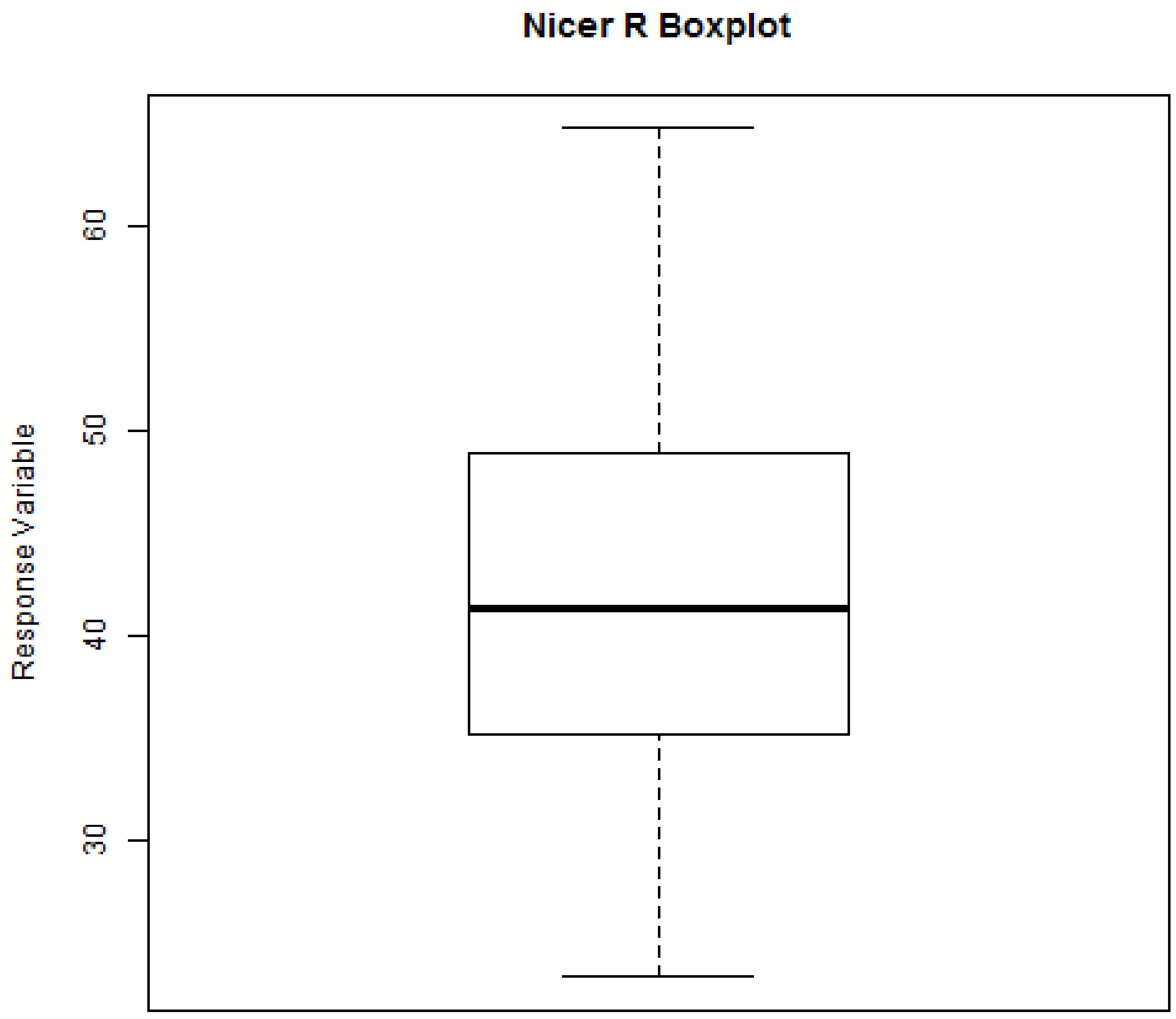
## Plots Showing More than Two Variables

What if we are interested in the relationship between the response variable and both the grouping variable and toy variable? We can add this information into our plot using different plotting symbols and/or colors, along with a legend.

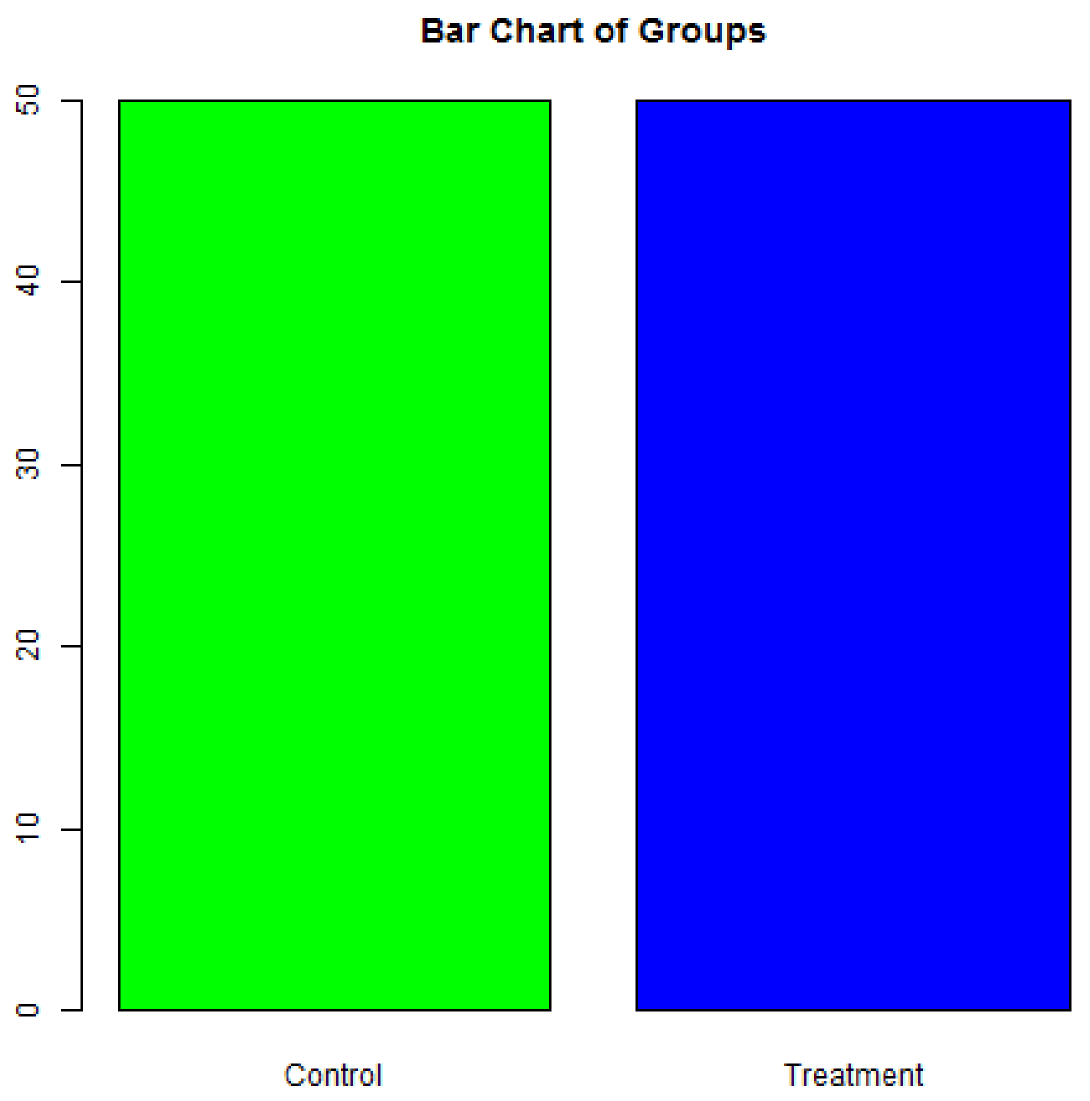
We can start with the same plot as before, but only plotting the data from the control group. To make sure the plot shows all of our data, we will find the range of the variables we are plotting in the original data set using the `range()` function.



Now, we can add the observations from the treatment group onto the plot using the `points()` function.



It would be good to add a legend to this plot. We can do so using the `legend()` function as follows.



```r
## Find the range of the variables we are plotting for BOTH groups
ydatarange=range(practicedata$respvar,na.rm=TRUE)
xdatarange=range(practicedata$toyvar,na.rm=TRUE)
## Creating the plot and adding points for the control group
plot(controldata$toyvar,controldata$respvar, # x variable, y variable
    main="Plot of Response Variable vs. Toy Variable", # change main label
    ylab='Response Variable', # change y-axis label
    xlab='Toy Variable', # change x-axis label
    pch=20, # change the plotting symbol
    col='green', # change color of plotting symbol
    xlim=xdatarange, # change the x-axis to cover the range
```

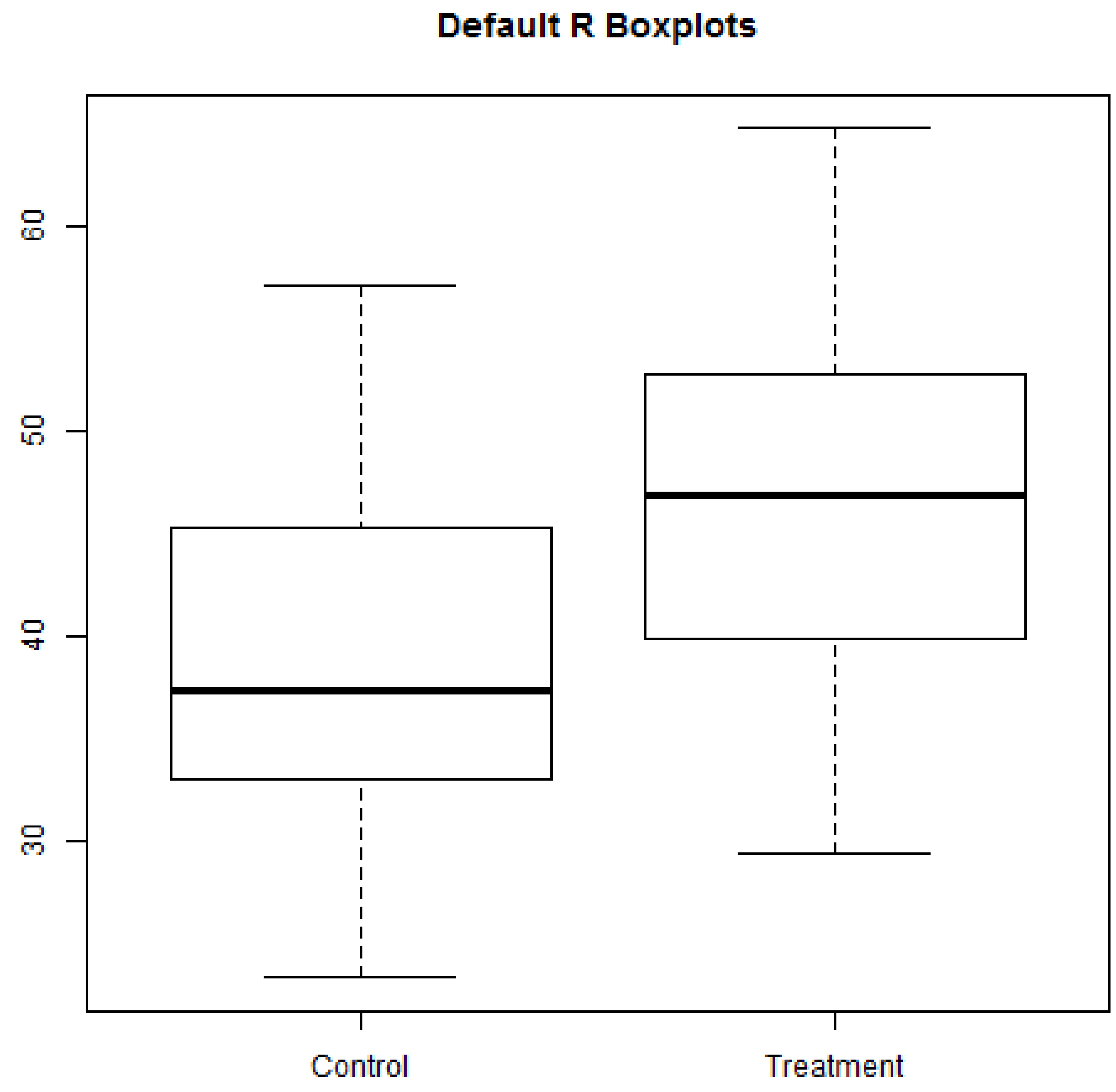
```
            # of both the treatment and control groups
    ylim=ydatarange # change the y-axis to cover the range
            # of both the treatment and control groups
     )
## Add observations for the treatment data to the plot
points(treatmentdata$toyvar,treatmentdata$respvar, # x variable, y variable
    pch=20, # change the plotting symbol
    col='blue', # change color of plotting symbol
  )
## Adding a Legend to the Plot
legend('topleft', # location of legend
       legend=c('Control Group','Treatment Group'), # lines of text in the legend
       pch=20, # symbol used in the legend
       col=c('green','blue') # colors of the symbol in the same
              # order as the lines of text in 'legend'
       # lty=1, lwd=1 # change the line type and width if lines are on the plot
       )
```

## Using `par()`

In addition to changing parameters within each plotting function (for instance, within `boxplot()`, `hist()`, `barplot()`, `pie()`, or `plot()`), you can also change other features of the plotting area. We will look at two such features here, and many, many more are described at this website: `http://www.`

One thing that is often helpful is creating more than one plot simultaneously. The option, `mfrow()`, inside `par()` does this. Take the histograms we made earlier. We can plot these simultaneously so that they are easier to compare. Once we set a value in `par()`, we either need to reset it or close the plotting window to return it to its default.

```
#Using the par function
par(mfrow=c(1,2))
  #mfrow=c(1,2) creates a plotting window with 2 rows and 1 column of plots
  #mfrow=c(2,1) creates a plotting window with 1 row and 2 columns of plots
#### Histograms for each group
hist(controldata$respvar,
    main='Control Group', # change the main title
    xlab='Response Variable'
    )
hist(treatmentdata$respvar,
    main='Treatment Group', # change the main title
    xlab='Response Variable'
    )
```


Default R Boxplots
60
50
40
30
Control
Treatment