Working with two variables

Teaching note:

This tutorial samples some of the ways we can compare and relate pairs of variables in R. We begin by showing some of the ways to compare the distributions of a number of similar variables, particularly box-and-whiskery plots. After that we examine the relationship between a pair of variables using scatterplots and fitted lines, eventually developing an interaction plot and a scatterplot matrix.

Getting the data:

For this assignment, we'll be using the same survey data about a supplier called "HBAT" that we used in the previous tutorial. (See the document HBAT-description for full details.)

Either of these lines of code from Tutorial 1 can be used again load the data:

```
hbat <- read.csv("HBAT.csv")  # if you have set the working directory
hbat <- read.csv("C:\\datasets\\HBAT.csv")  # to use the file's full address
```

For this assignment, use the following variables:

- Website_User_Friendliness: Customer perception of HBAT's website.
- Support: Extent to which technical support is offered to help solve product/service issues.
- SF_Image: Overall image of HBAT's sales force.
- Satisfaction: Customer satisfaction with past purchases from HBAT.
- Region: Customer location. 0=USA/North America, 1=Outside North America.

Examine the data set:

In the previous tutorial we looked at the dataset as a set of individual variables. Now let's try to view them in comparison to one another.

1. Recall that the str() function lets us quickly see the type of data in each variable side by side:

   ```
   str(hbat)
   ```

   Gives us something like:

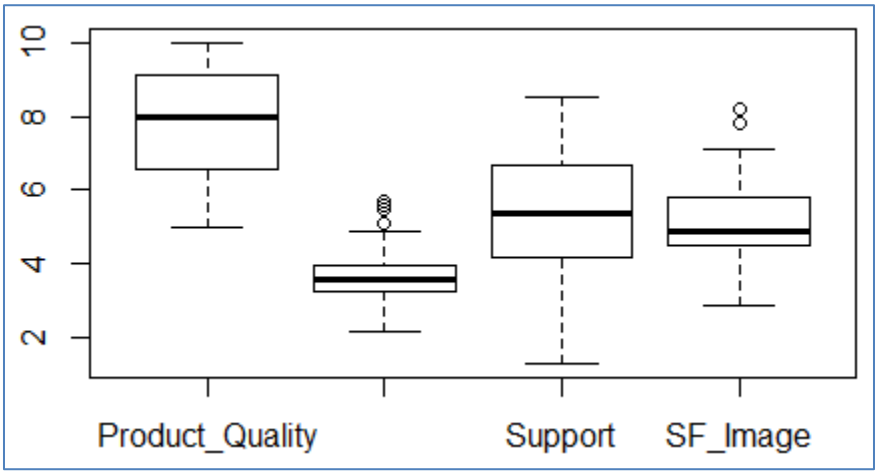
   ```
   'data.frame':     100 obs. of  24 variables:
    $ ID                      : int  1 2 3 4 5 6 7 8 9 10 ...
    $ Cust_Type               : int  2 3 3 1 2 1 1 2 2 1 ...
    $ Ind_Type                : int  0 1 0 1 0 1 1 0 1 0 ...
    $ Firm_Size               : int  1 0 1 1 1 0 1 1 1 1 ...
    $ Region                  : int  1 0 1 1 0 1 1 1 1 1 ...
   ```

```
$ Dist_System              : int  1 0 1 0 1 0 0 0 0 0 ...
$ Product_Quality          : num  8.5 8.2 9.2 6.4 6.9 6.2 5.8 6.4 ...
$ Website_User_Friendliness: num  3.9 2.7 3.4 3.3 3.4 3.7 3.3 4.5 ...
$ Support                  : num  2.5 5.1 5.6 7 5.2 3.1 5 5.1 5.1 ...
$ Complaint_resolution     : num  5.9 7.2 5.6 3.7 4.6 4.1 4.8 6.1 ...
$ Advertising              : num  4.8 3.4 5.4 4.7 2.2 4 2.1 4.7 ...
$ Product_Line             : num  4.9 7.9 7.4 4.7 2.3 3.6 5.9 5.7 ...
$ SF_Image                 : num  6 3.1 4.5 3.7 5.4 5.1 5.8 5.7 ...
$ Competitive_Pricing      : num  6.8 5.3 4.5 8.8 6.8 8.5 8.9 ...
$ Warranty                 : num  4.7 5.5 6.2 5.1 4.8 5.4 5.9 5.4 ...
$ New_Products             : num  4.3 4 4.6 9.5 2.5 4.8 4.4 5.3 ...
$ Ordering_Billing         : num  5 3.9 4.5 3.6 2.1 4.3 4.4 4.1 ...
$ Price_Flexibility        : num  5.1 4.3 3.5 4.7 4.2 6.3 6.1 5.8 ...
$ Delivery_Spped           : num  3.7 4.9 4.5 3 3.3 2 3.7 4.6 4.4 ...
$ Satisfaction             : num  8.2 5.7 8.9 4.8 7.1 6.3 7 5.5 ...
$ Recommendation           : num  8 6.5 8.4 6 6.6 6.3 7.8 5.8 5.9 ...
$ Future_Purchase          : num  8.4 7.5 9 7.2 9 6.1 7.2 7.7 8.2 ...
$ Usage_Level              : num  65.1 67.1 72.1 40.1 57.1 50.1 ...
$ Future_Relations         : int  1 0 1 0 0 0 0 0 1 0 ...
```

2. We can see that many of the variables (from Product_Quality through Usage_Level) are scored on similar scales, probably based on a questionnaire instrument that asked for ratings from 0 to 10. A box-and-whisker plot is an intuitive visualization that allows us to compare the distributions of our first four variables of interest:

   ```
   boxplot(hbat[c(7,8,9,13)])
   ```

   The `boxplot()` function takes a vector of observations or a list of vectors. Our data frame hbat is such a list, so the command `boxplot(hbat)` would plot all the variables together despite their very different scales. The code `hbat[c(7,8,9,13)]` indicates a narrower selection: only the 7th, 8th, 9th, and 13th variables in `hbat` will be sent to the `boxplot()` function. The resulting plot:

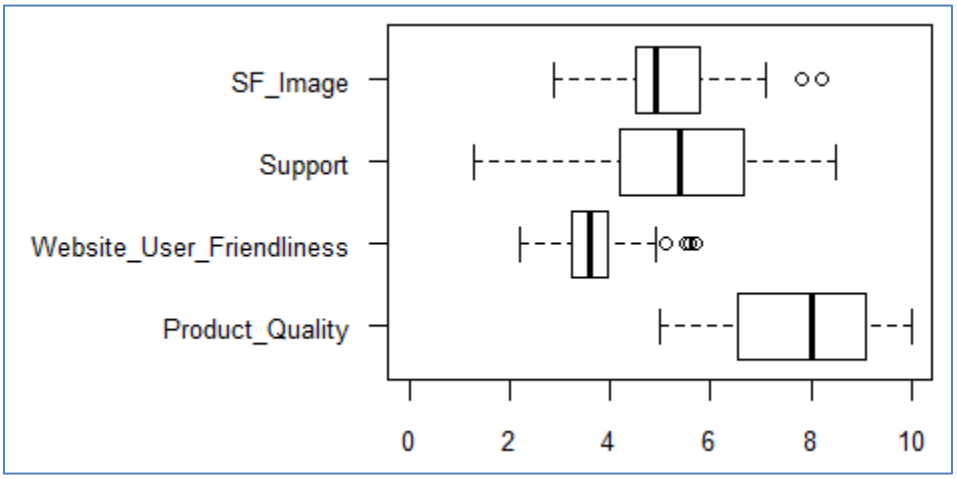


   We can see the median of each variable, the interquartile range, and some indication of the "ordinary" range of the variable as well as some outliers. Although the precise meanings of these may vary (consult a textbook for what they mean and why), this visual allows us to very easily see how the variables are similar and how they are different, without requiring us to consult statistics and p-values.

Two critiques of this plot are that not all of the variables are named (because there isn't space for the name) and that the bars are rather squashed due to the wide aspect of the plot. One solution is to make it a horizontal boxplot. Use these two lines of code:

```
par(mar=c(5,10,4,2))  # sets an extra-large left margin
boxplot(hbat[c(7,8,9,13)],horizontal=TRUE,ylim=c(0,10),las=1,cex.axis=0
.8)
```

The first line sets the `mar` parameter for margins to give extra space for the labels on the left. In the `boxplot()` code, the horizontal parameter turns the plot on its side. The `ylim` parameter forces the display to show the whole range from zero to ten points, `las=1` means the labels are to be printed horizontally, and `cex.axis=0.8` reduces the font size of the labels slightly. The result is a more readable box-and-whisker plot:

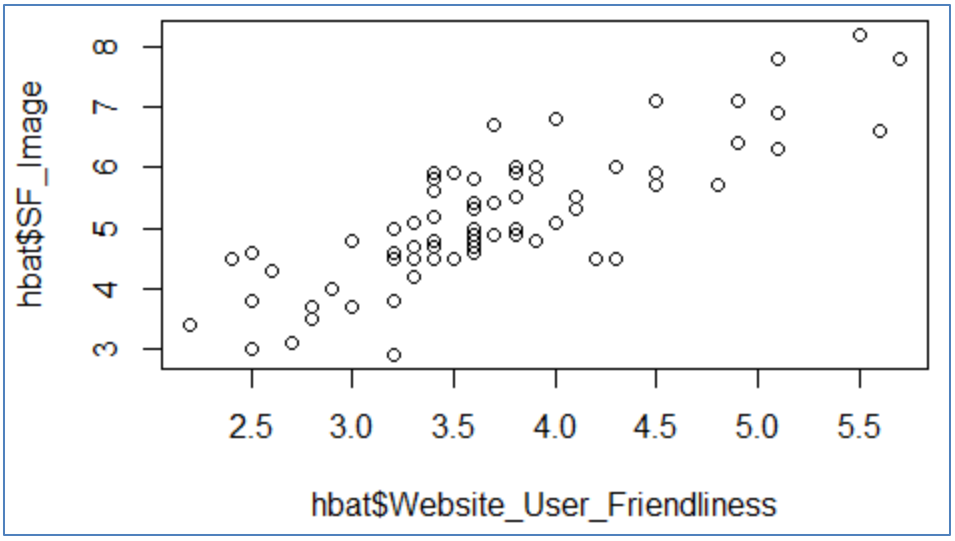


Compare and relate two variables:

Knowing a bit about how the variables compare to one another, we can look more closely at the relationships between pairs of variables.

1. A scatterplot or XY plot is a very simple data visualization which plots each observation (i.e. each survey respondent) with one variable as the x coordinate and the other variable as the y coordinate. Try a scatterplot relating Website_Use_Friendliness to SF_Image:

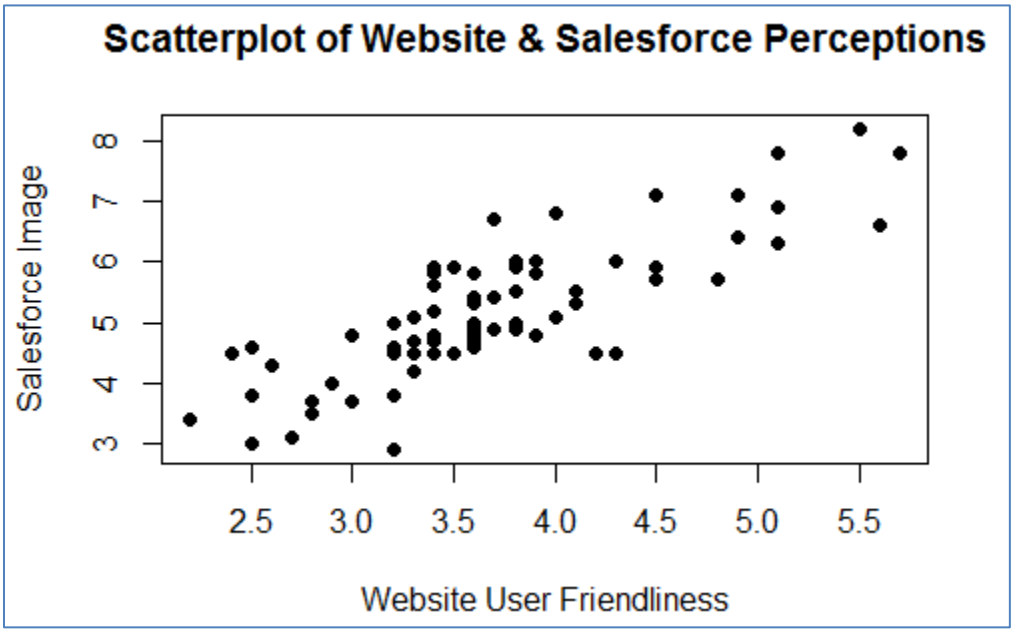
   ```
   plot(hbat$Website_User_Friendliness,hbat$SF_Image)
   ```



This is a good time to introduce some parameters for making your plots easier to understand. The parameter `main` sets the plot's title, `xlab` and `ylab` set the axis labels, and `pch` sets the type of "dot". Try `pch=19` for a solid round dot, or `pch=4` for an "x". If a `plot` function becomes too long, you can type it on multiple lines of a script, as long as no line ends with a ")" which could be mistaken for the end of the function.

```
plot(hbat$Website_User_Friendliness,hbat$SF_Image,
     main="Scatterplot of Website & Salesforce Perceptions",
     ylab="Salesforce Image", xlab="Website User Friendliness",
     pch=19)
```

You can learn about more graphical parameters by typing `?par` at the interactive prompt for help. The above code gives us:



A plot that makes it easier for our readers to see that, in general, survey respondents who rated the website highly also had a high image of HBAT's sales force, and the same was true of those who gave poor ratings to both.

2. We like to visualize relationships as a function (i.e. a line on the graph) and try to understand their shape. A common first step is to plot a regression line. Use the following code to add a linear regression line to the scatterplot. Let's make it red.

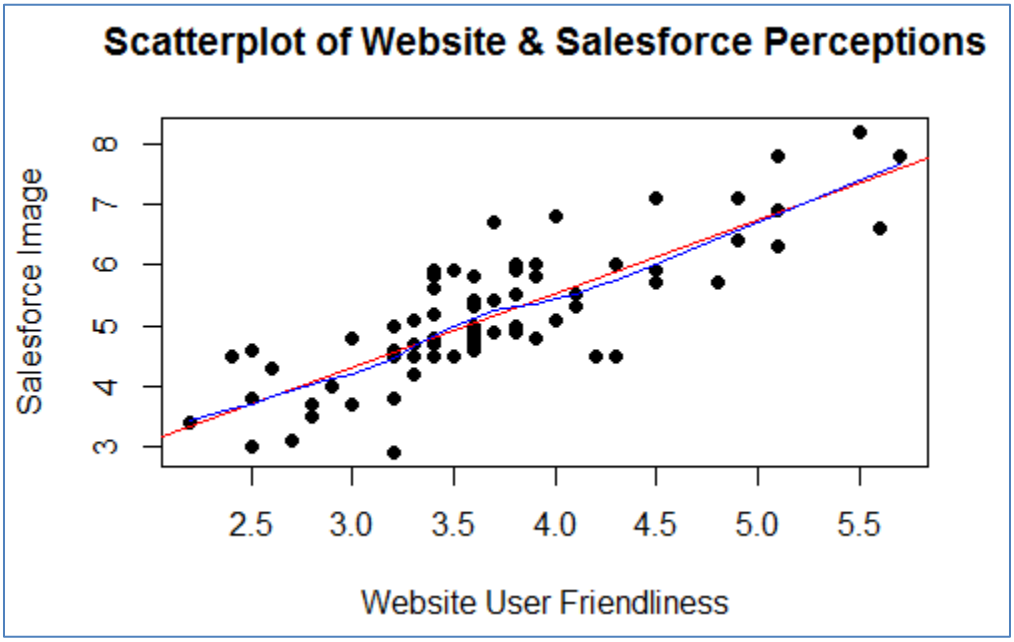
```
model1 <- lm(hbat$SF_Image~hbat$Website_User_Friendliness)
abline(model1,col="red")
```

The `lm()` function is R code to generate a linear model by regression of the first variable onto the variable(s) to the right of the "~". The `abline()` function draws a straight line.

3. The LOWESS method is one of many methods for generating a smooth curve to approximate a nonlinear relationship between the data. Let's add a LOWESS line in blue:

```
lines(lowess(hbat$Website_User_Friendliness,hbat$SF_Image),col="blue")
```

In this case the two lines are pretty close together. The relationship is pretty close to linear.



4. We may sometimes want to show the interaction with a third variable. One way to do this is to split the data according to the third variable and essentially draw two scatterplots in one. We will use the Region variable to see if the correspondence between website and sales force scores is different outside of North America.

The expression `hbat$Region==0` returns an array of TRUE and FALSE values corresponding to data points in North America (TRUE) and elsewhere (FALSE). Thus the expression `hbat$Website_User_Friendliness[hbat$Region==0]` gives us the website scores only for North American survey responses. Let's start a new plot with just the North American values as black dots as above. This time we use `xlim` and `ylim` parameters to make sure the graph's range is wide enough to accommodate the points we haven't plotted yet:

```
plot(hbat$Website_User_Friendliness[hbat$Region==0],
     hbat$SF_Image[hbat$Region==0],
     main="Scatterplot of Website & Salesforce Perceptions",
     ylab="Salesforce Image", xlab="Website User Friendliness",
     ylim=c(2,8),xlim=c(2,6),
     pch=19)
```

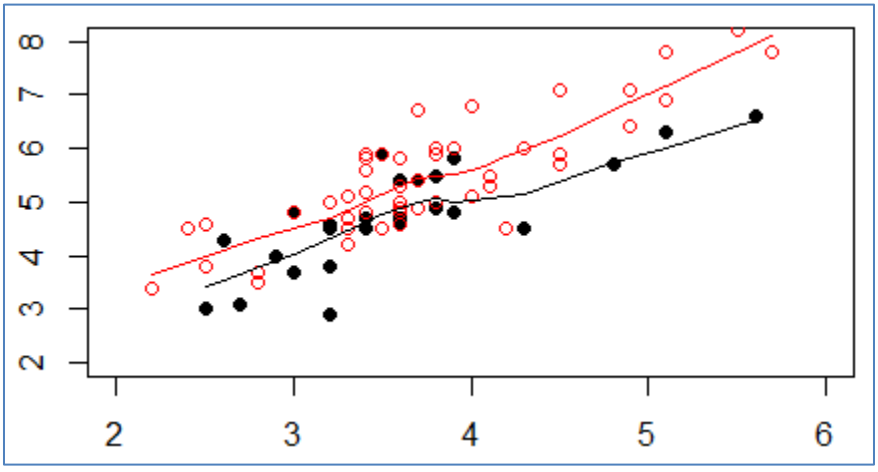
Now we use the `points()` function which adds a scatterplot of a second dataset without generating a new graph. Most of the same parameters that go into a `plot()` function can be used here. These points will be red, unfilled circles.

```
points(hbat$Website_User_Friendliness[hbat$Region==1],
     hbat$SF_Image[hbat$Region==1],
     pch=1,col="red")
```

Plotting LOWESS lines allows us to compare the relationships between the two variables in the different regions:

```
lines(lowess(hbat$Website_User_Friendliness[hbat$Region==0],
             hbat$SF_Image[hbat$Region==0]), col="black")
lines(lowess(hbat$Website_User_Friendliness[hbat$Region==1],
             hbat$SF_Image[hbat$Region==1]), col="red")
```

The final plot:



We might conclude from this plot that the more-or-less linear relationship between perceptions of the website and the sales force is consistent worldwide, but that overseas respondents tended to rate the sales force relatively higher than North Americans did.

5. One more variation on the topic of scatterplots is the "scatterplot matrix" provided in R by the pairs() function. A little bit like a statistical correlation matrix but with richer visuals, a scatterplot matrix allows you to see, at a glance, what the relationships between several variables in your dataset look like. Focusing again on our four variables of interest, we code:

```
pairs(hbat[c(7,8,9,13)],cex.labels=1)
```