

Chapter 3

Describing Data Numerically and Graphically**3.1 Obtaining Numerical Summaries**

Follow along with me in this section by importing the SPSS file `LarsonHall.Forbidden.sav` into R. Make sure that it is the active data set in R Commander. The data set should have 44 rows and 12 columns. I have named it `forget`.

In R Commander, choose `STATISTICS > SUMMARIES > NUMERICAL SUMMARIES`. This will let you make a summary of one variable with all cases lumped together, or you can also summarize the variable by groups.

For this particular data set, I want to see summaries of my data divided into the groups that I created, and I will look at summaries for the listening test variable only (`RLWTEST`).

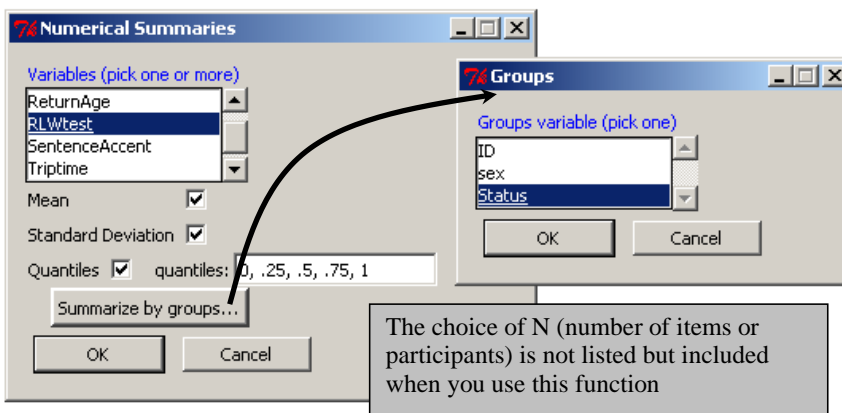


Figure 3.1 Inside the “Numerical Summaries” box in R.

R Commander uses the `numSummary` command to produce the summary.

```
numSummary(forget["RLWtest"], groups= Status, statistics=c("mean", "sd", "quantiles"))
```

	mean	sd	0%	25%	50%	75%	100%	n
Non	67.73333	13.76054	47	56.5	67	79.00	91	15
Late	69.20000	15.37716	36	60.5	66	80.50	90	15
Early	80.85714	14.27670	47	71.5	88	90.75	95	14

First we see the mean, then the standard deviation, then the scores corresponding to the various percentiles, and finally the number (N) of items evaluated. If there are missing values they will be counted in a column at the end of the output as well.

To get other types of statistical summaries using R Commander, choose STATISTICS > SUMMARIES > TABLE OF STATISTICS and enter the function in the “Other(specify)” box (a list of other summary statistics can be found in the table below). This particular command runs only when you have factors to split the data into groups.

To obtain the same results as the numerical summary in R, albeit one test at a time, use the `tapply()` command as shown below:

<code>attach(forget)</code>	
<code>tapply(RLWtest, list(Status=forget\$Status), mean, na.rm=T)</code>	
<code>attach(forget)</code>	The <code>attach</code> command means the current data set is attached high in the search path, and thus the names of the variables in the set can be used alone instead of specifying them with their data set (such as <code>forget\$RLWTEST</code>).
<code>tapply (data, indices, function, . . .)</code>	<code>tapply</code> is a command that creates a table for the function defined inside of it.
<code>RLWtest</code>	The data in this case is the <code>RLWtest</code> column of the data frame <code>forget</code> ; since <code>forget</code> is already attached, we can just type the variable name (otherwise, type <code>forget\$RLWtest</code>).
<code>list(Status=forget\$Status)</code>	This is the index that splits the <code>RLWtest</code> variable; more categorical factors could be listed here if one wanted to split the scores even further, for example: <code>list(Status=forget\$Status, sex=forget\$sex)</code>
<code>mean</code> <code>sd</code> <code>var</code> <code>median</code> <code>quantile</code> <code>range</code> <code>min</code> <code>max</code> <code>mean, tr=.2</code>	This is the function applied: standard deviation variance median quantile points range minimum value maximum value trimmed means (here 20%).
<code>na.rm=T</code>	Removes any values which are not defined.

Here is the output for the `tapply()` command using the function to obtain the average score:

```

      Non      Late      Early
67.73333 69.20000 80.85714

```

You will also want to obtain a count of the number of observations of a variable or, if you have groups, the number of participants in each group. The following command will

summarize the entire number of observations in a variable and remove any NAs for that variable:

```
sum(!is.na(RLWtest)) #overall N for the variable
[1] 44
```

To get a count of the number of participants in each group for a certain variable, making sure to exclude NAs, use the following command:

```
table(Status[!is.na(RLWtest)]) #N count for each group
```

```
Non Late Early
15 15 14
```

This command counts all of the entries which are *not* NAs divided up by the categorizing factor, in this case **Status**. To split by more than one variable, just add a comma, as in this example from the .csv file writing:

```
table(writing$L1, writing$condition[!is.na(writing$score)])
      correctAll correctTarget noCorrect
Arabic          59           76         85
Japanese        30           40         50
Russian          30           40         48
Spanish          30           40         50
```

Tip: Another way to obtain the entire number of participants in a study is to use the `length(forget$RLWtest)` command, but the `na.rm=T` argument cannot be used with this command.

Deleted: ¶

A quick step to get the mean, median, minimum, maximum, and Q_1 and Q_3 for all numeric variables in the set (not split by groups, however) and counts of participants in each level of a factor variable is in R Commander. Choose **STATISTICS > SUMMARIES > ACTIVE DATA SET**. The R syntax for this command is:

```
summary(writing)
```

```
      score          L1          condition
Min.   : 35.0   Arabic :220   correctAll   :149
1st Qu.: 63.0   Japanese:120   correctTarget:196
Median :103.0   Russian :118   noCorrect    :233
Mean   :121.8   Spanish :120
3rd Qu.:163.8
Max.   :373.0
```

Another method of obtaining the number of observations, minimum and maximum scores, mean, median, variance, standard deviation, and skewness and kurtosis numbers is the `basicStats()` function from the `fBasics` library.

```
basicStats(forget$RLWtest[1:15]) #chooses just rows 1–15, which are the “non” group
```

Obtaining the Mean, Standard Deviation, and Quantiles with R

1. Import your data set and make sure it is active in R Commander, or `attach()` it in R Console.
2. Choose STATISTICS > SUMMARIES > NUMERICAL SUMMARIES.
3. Open the “Summarize by groups . . .” button if you have categorical groups. Press OK, and then OK again.

The basic R code for this command is:

```
numSummary(forget[,"RLWtest"], groups= Status, statistics=c("mean", "sd",
"quantiles"))
```

4. Another method that produces a large number of summary items is to use the `basicStats()` command from the `fBasics` library.

Formatted: Font: Not Bold, Italic

Deleted: *Quantiles*

Formatted: Font: Not Bold, Italic

Formatted: Font: Not Bold, Italic

Deleted:

Formatted: Not Small caps

Deleted:

Formatted: Indent: Left: 0"

Deleted:

Deleted:

Deleted: ¶

¶

3.1.1 Skewness, Kurtosis, and Normality Tests with R

To get a numerical measure of skewness and kurtosis using R, use the `fBasics` library. If you want to calculate the skewness and kurtosis levels in data that has groups, you must first subset the data into those groups. The following commands show how this was accomplished with the `RLWtest` variable and only the Non-immersionists group (LarsonHall.forgotten.sav SPSS data set).

```
library(fBasics)
non=forget$RLWtest[1:15]
#chooses rows 1–15 of the variable to subset and names them “non”
skewness(non, na.rm=T)
kurtosis(non, na.rm=T)
```

Note that I basically manually subsetted my file above by choosing rows 1–15, but you could also use the `subset()` command (through either DATA > ACTIVE DATA SET > SUBSET ACTIVE DATA SET in R Commander or the R code).

The `skewness` help file states that the default method of computing skewness is the “moment” method. Another possible method, that of “fisher” (`method=c("fisher")`) should be used when resampling with a bootstrap.

A variety of numerical tests of normality are available, including the Shapiro–Wilk and Kolmogorov–Smirnov tests. The Shapiro–Wilk test needs essentially only one argument. This test for normality is appropriately used with group sizes under 50.

```
shapiro.test(non)
```

```
Shapiro-Wilk normality test

data: non
W = 0.9613, p-value = 0.7142
```

If the p -value is above $p=.05$, the data is considered to be normally distributed. The Kolmogorov–Smirnov test needs two arguments. To use a one-sided test to assess whether

the data comes from a normal distribution, the second argument needs to be specified as the `pnorm` distribution (the normal distribution).

```
ks.test(non, "pnorm")
```

```
Warning in ks.test(non, "pnorm") :
  cannot compute correct p-values with ties

      One-sample Kolmogorov-Smirnov test

data:  non
D = 1, p-value = 1.872e-13
alternative hypothesis: two-sided
```

The print-out warns that there are ties in the data. It gives a very small *p*-value (1.872e-13 means you would move the decimal point 12 places to the left of the number), indicating that the non-immersionists' data is not normally distributed.

The `nortest` library provides five more tests of normality. Ricci (2005) states that the Lilliefors test is especially useful with small group sizes, and is an adjustment of the Kolmogorov–Smirnov test.

```
library(nortest)
lillie.test(non)
```

```
      Lilliefors (Kolmogorov-Smirnov) normality test

data:  non
D = 0.1156, p-value = 0.8512
```

I want to warn my readers here that just because a numerical normality test does not find a *p*-value less than .05 it does not necessarily mean your distribution is normal. If your data set is small, these kinds of tests do not have enough power to find deviations from the normal distribution. Graphical tools are just as important as these numerical tests.

To Obtain Skewness, Kurtosis, and Normality Tests to Assess Normality in R,

1. If needed, first subset data into groups by creating a new object that specifies the rows you want. One way of doing so is to specify the column you want with the rows that pertain to the group you want, like this:

```
non= forget$RLWTEST[1:15]
```

2. Use these commands from the `fBasics` library:

```
skewness(non)
kurtosis(non)
```

Alternately, the `basicStats()` command will give a number of descriptive statistics including the skewness and kurtosis numbers.

```
shapiro.test(non)
```

- Deleted: &
- Deleted: .
- Formatted: Font: Not Bold, Italic
- Formatted: Font: Not Bold, Italic
- Formatted: Font: Not Bold, Italic
- Formatted: Font: Not Bold, Italic
- Deleted:
- Deleted:
- Deleted:
- Formatted: Indent: Left: 0"
- Deleted: *
- Deleted:
- Deleted: ¶
- ¶

3.2 Application Activities with Numerical Summaries

1. Import the SPSS file DeKeyser2000.sav and name it `dekeyser`. Summarize scores for the GJT grouped by the `Status` variable. Make sure you have data for the number of participants, the mean, and the standard deviation. By just eyeballing the statistics, does it look as though the groups have similar mean scores (maximum possible score was 200)? What about the standard deviation?

2. Import the SPSS file Obarow.sav and call it `obarow` (it has 81 rows and 35 columns). Summarize scores for the gain score in the immediate post-test (`gnsc1.1`) grouped according to the four experimental groups (`trtmnt1`). Each group was tested on 20 vocabulary words, but most knew at least 15 of the words in the pre-test. Obtain summaries for number of participants, mean scores, and standard deviations. Do the groups appear to have similar mean scores and standard deviations?

3.3 Generating Histograms, Stem and Leaf Plots, and Q-Q Plots

3.3.1 Creating Histograms with R

If you want a histogram of an entire variable (not split by groups), it is easy to use R Commander to do this. Go to `GRAPHS > HISTOGRAM`.

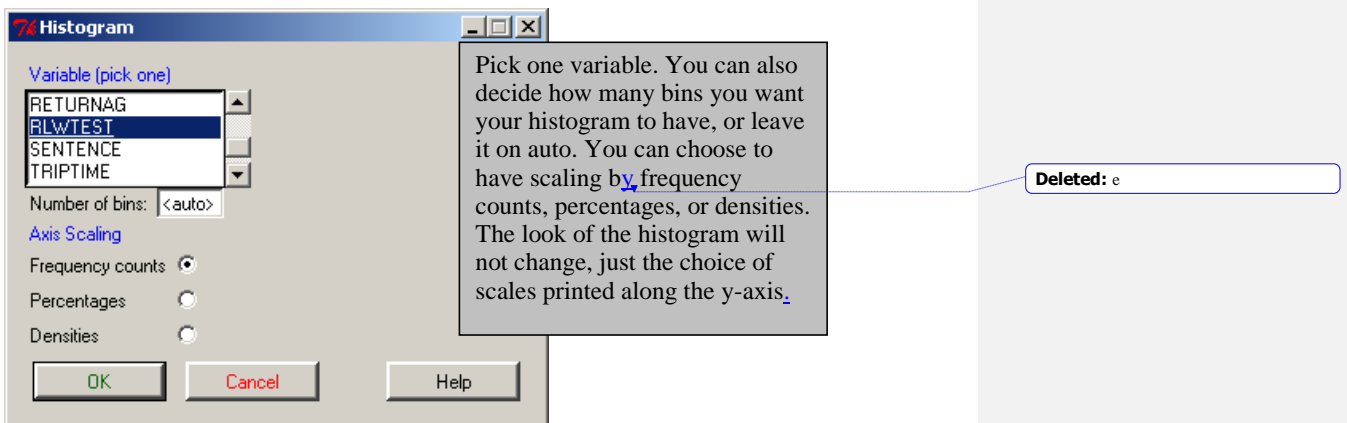


Figure 3.2 Obtaining a histogram in R Commander.

There are many ways you might want to change the histogram that R Commander generates, especially as to the labels it automatically prints out. You can use R Commander to see the code that generates a histogram, and then take that code and play with it in R Console until you get the look you like. The commands used to generate the three histograms with the Larson-Hall and Connell (2005) data in Figure 3.14 of the SPSS book (*A Guide to Doing Statistics in Second Language Research Using SPSS*, p. 81) are given below. Note that the first histogram leaves the y axis label in, but the second and third histograms get rid of the y label by putting nothing between the parentheses.

```
par(mfrow=c(1,3)) #sets the graphics display to 1 row, 3 columns
Hist(forget$RLWtest[1:15], col="gray", border="darkgray", xlab="", main="Non-immersionists")
```

```
Hist(forget$RLWtest[16:30], col="gray", border="darkgray", xlab="", ylab="",
main="Late immersionists")
Hist(forget$RLWtest[31:44], col="gray", border="darkgray", xlab="", ylab="",
main="Early immersionists")
```

The code below generates a histogram of 50 samples of the normal distribution (as seen in multiples in Figure 3.11 of the SPSS book (*A Guide to Doing Statistics in Second Language Research Using SPSS*, p. 79) and singly in Figure 3.3).

<code>hist(rnorm(50,0,1),xlab="", main="", col="lightgray", border="darkgray",prob=T)</code>	
<code>hist(x, . . .)</code>	The command for a histogram; many other general graphics parameters can be added.
<code>rnorm(50,0,1)</code>	Generates 50 samples of a normal distribution with mean=0 and s=1
<code>xlab="", main=""</code>	Deletes the generic x label and main title that R would automatically print.
<code>col="lightgray"</code>	Colors the inside of the histogram bins.
<code>border="darkgray"</code>	Changes the color of the bin borders.
<code>prob=T</code>	Changes the scaling to density; can also use <code>scale="density"</code> (also "percent" or "frequency")

To overlay the histogram with a density plot of the normal distribution (Figure 3.3) I used the following code:

```
norm.x=rnorm(50,0,1)
x=seq(-3.5, 3.5, .1)
dn=dnorm(x)
hist(norm.x, xlab="", main="50 samples", col="lightgray", border="darkgray", prob=T)
lines(x, dn, col="red", lwd=2)
```

Note that in R you could use the command `Hist` or `hist`. `Hist` is specific to R Commander and is set up to call the `hist` command, but adds a couple of default arguments that `hist` doesn't have.

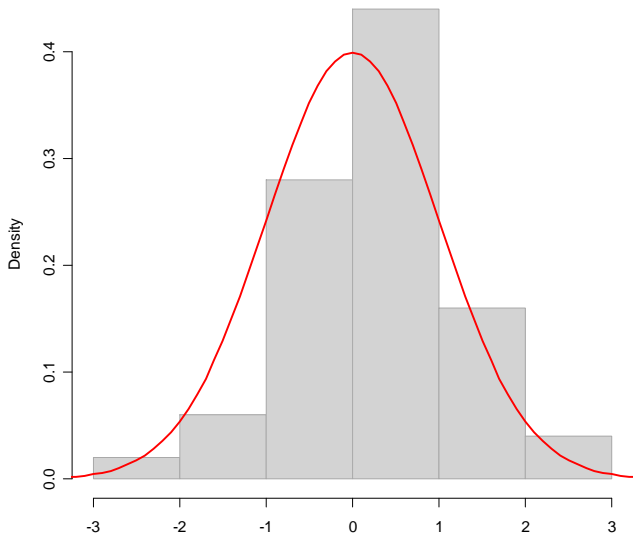


Figure 3.3 Histogram with overlaid normal distribution.

To create histograms of separate groups, you must subset the data into groups or specify certain rows in one vector of data, as was shown above in this section.

To Obtain Histograms in R

1. If needed, first subset data into groups by creating a new object that specifies the rows you want, like this:

```
non= forget$RLWTEST[1:15]
```

2. Use the hist command:

```
Hist(non) OR hist(non)
```

Formatted: Font: Not Bold, Italic

Formatted: Font: Not Bold, Italic

Deleted: .

Deleted: ¶
¶

3.3.2 Creating Stem and Leaf Plots with R

Creating a Stem and Leaf Plot with R Commander

1. If needed, first subset data into groups.
2. From the menu, choose **GRAPHS > STEM AND LEAF DISPLAY**. Pick the variable you want to graph. If using R Console, the command `stem.leaf()` creates the plot.
3. There are options to specify how to divide up stems if you have large data files (Leaf Digit, Parts Per Stem, Style of Divided Stems). Use the Help button if you'd like more information about these options.
4. The option of "Trim outliers" is self-explanatory, but I do not recommend this option for most data sets, as it is not a diagnostic. Uncheck this box, but leave the others checked. The option "Show depths" prints counts of each line. The option "Reverse negative leaves" concerns when numbers are negative.

Formatted: Font: Not Bold, Italic
 Formatted: Font: Not Bold, Italic
 Formatted: Font: Not Bold, Italic

Deleted: ()

Deleted: s
 Formatted: Font: Not Bold

3.3.3. Creating Q-Q Plots with R

To create a Q-Q plot for an entire variable, use R Commander. Go to **GRAPHS > QUANTILE-COMPARISON PLOT**.

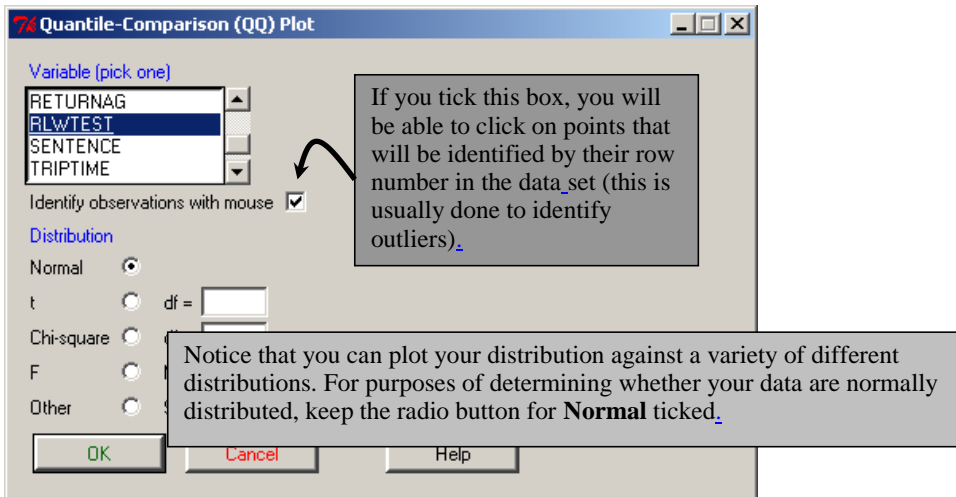


Figure 3.4 Obtaining a Q-Q plot in R Commander.

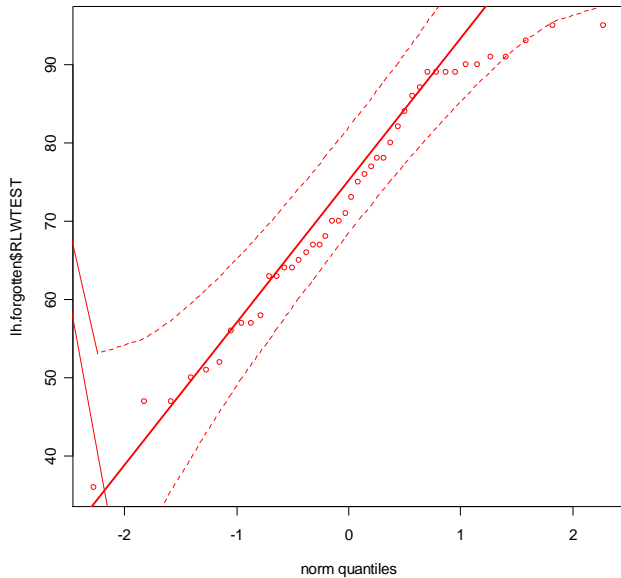


Figure 3.5 A Q-Q plot with 95% confidence intervals.

In the Q-Q plot in Figure 3.5, the dotted lines contain a simulated 95% confidence interval envelope. Maronna, Martin, and Yohai (2006, p. 10) say that if no points fall outside the confidence interval we may be “moderately sure that the data are normally distributed.” For this data set, however, we do have one point which falls outside the envelope (top right-hand side) and several points which are right on the border. Thus it would be hard to conclude that these data are exactly normally distributed.

To analyze a Q-Q plot for different groups, first split the variable into different subsets. The command detailed below produces a Q-Q plot for the RLWtest data for just the non-immersionist group (non).

<code>qq.plot(non, dist="norm", labels=F)</code>	
<code>qq.plot(x, . . .)</code>	Performs quantile-comparisons plots; the command <code>qqnorm</code> also performs a Q-Q plot against the normal distribution.
<code>dist="norm"</code>	<code>qq.plot</code> can plot data against several different distributions: “t,” “f,” and “chisq.”
<code>labels=F</code>	If set to TRUE, will allow identification of outliers with the mouse.

To produce a set of Q-Q plots for a variable split into groups without manually subsetting the data yourself, you can use the Lattice graphics library. The following commands produced Figure 3.6.

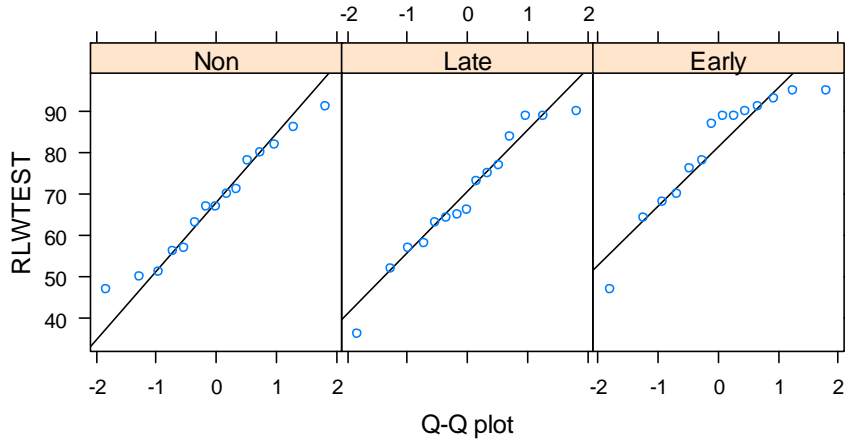


Figure 3.6 Q-Q plots for the LarsonHall.forgotten data.

```
library(lattice)
qqmath(~RLWtest|Status, aspect="xy", data=forget,layout=c(3,1), xlab="Q-Q
plot",
prepanel=prepanel.qqmathline,
panel=function(x, ...){
panel.qqmathline(x,...)
panel.qqmath(x,...)
})
```

The commands following the bolded command are used to insert the reference line into the graphic and were taken from the example found in the `help(qqmath)` page. If the bolded command is used by itself, it will produce the three Q-Q plots without the reference line. This bolded command is analyzed below:

```
qqmath(~RLWtest|Status, aspect="xy", data=forget, layout=c(3,1), xlab="Q-Q
plot")
```

<code>qqmath (~RLWtest Status, . . .)</code>	Performs a Q-Q plot on the data split by the second argument (Status).
<code>aspect="xy"</code>	This argument controls the physical aspect ratio of the panels; "xy" keeps a certain aspect ratio; alternatively, <code>aspect="fill"</code> makes the panels as large as possible.
<code>data=forget</code>	Specifies the data set.
<code>layout=c(3,1)</code>	Specifies there should be 3 columns and 1 row; if not specified, the layout will be automatically plotted.
<code>xlab="Q-Q plot"</code>	Gives x label.

To learn more about both traditional graphics and lattice graphics in R, see the indispensable book by Paul Murrell (2006).

3.4 Application Activities for Exploring Assumptions

Note: Keep your output from the activities in Checking Normality, as you will use some of the same output for the activities in Checking Homogeneity of Variance as well.

3.4.1 Checking Normality

1. Chapter 6 on correlation will feature data from Flege, Yeni-Komshian, and Liu (1999). Import the SPSS file *FlegeYKLiu.sav* file and name it *FYL* (it has 264 rows and 3 columns). Examine the data for pronunciation divided according to groups by using Q-Q plots. From the available evidence, does the data appear to be normally distributed?
2. Create a histogram with the *GJT* variable for each of the two groups in the SPSS file *DeKeyser2000.sav* file (imported as *dekeyser*). To make the groups, specify the row numbers. The “Under 15 group” is found in rows 1–15 and the “Over 15” group is found in rows 16–57. What is the shape of the histograms? Describe them in terms of skewness and normality. Look at stem and leaf plots for the same groups. Do you see the same trends?
3. Chapter 12 on repeated-measures ANOVA will feature data from Lyster (2004). Import the SPSS file *Lyster.written.sav* and name it *lyster* (it has 180 rows and 12 columns). Examine the gain scores for the comprehension task (*CompGain1, CompGain2*) divided according to groups (*Cond* for condition). From the available evidence, does the data appear to be normally distributed? Look at Q-Q plots only.

3.4.2 Checking Homogeneity of Variance

1. Check the homogeneity of variance assumption for pronunciation for the groups in the Flege, Yeni-Komshian, and Liu (1999) study (see activity 1 under “Checking Normality” above) by looking at standard deviations for each group. The maximum number of points on this test was 9. What do you conclude about the equality of variances?
2. Check the homogeneity of variance assumption for the *GJTScore* variable for the two groups in the DeKeyser (2000) study (use the *DeKeyser2000.sav* file) by looking at standard deviations for each group. The maximum number of points on this test was 200. What do you conclude about the equality of variances?
3. Check the homogeneity of variance assumption for the *CompGain2* variable for the four groups in the Lyster (2004) study (use the *Lyster.Written.sav* file) by looking at standard deviations for each group. The maximum number of points any person gained was 21. What do you conclude about the equality of variances?

3.5 Imputing Missing Data

It is a very common occurrence, especially in research designs where data is collected at different time periods, to be missing data for some participants. The question of what to do with missing data is a rich one and cannot be fully explored here, but see Tabachnick and Fidell (2001) and Little and Rubin (1987) for more detailed information. I will merely give a short summary of what can be done when the missing data points do not follow any kind of pattern and make up less than 5% of the data, a situation that Tabachnick and Fidell (2001) assert can be handled equally well with almost any method for handling missing data. An example of a pattern of missing data would be that in administering a questionnaire you found that all respondents from a particular first language background would not answer a certain question. If this whole group is left out of a summary of the data, this distorts the findings. In this case, it may be best to delete the entire question or variable.

It is up to the user to explicitly specify what to do with missing values. I would like to point out, however, that Wilkinson and the Task Force on Statistical Inference (1999) assert that the worst way to deal with missing data is listwise and pairwise deletion. If you delete an entire case because a value in one variable is missing (listwise deletion), this of course means that all the work you went to in obtaining data from that participant is wasted, and of course with a smaller sample size your power to find results is reduced. Pairwise deletion is when you omit data for those cases where the variable you are examining doesn't have data, but you don't omit the whole case from your data set. Doing this may result in different Ns for different comparisons in the data set.

Two methods that Howell (n.d.) recommends are maximum likelihood and multiple imputation. The details of how these methods work are beyond the scope of this book, but Howell provides a comprehensible summary at http://www.uvm.edu/~dhowell/StatPages/More_Stuff/Missing_Data/Missing.html#Return1. A free program called NORM, available at www.stat.psu.edu/~jls/misoftwa.html, will perform both maximum likelihood and multiple imputation, and it also available as an R library, also called norm.

However, for imputation I prefer the `mice` library. It is quite easy to impute values using this library and the function of the same name. The default method of imputation depends on the measurement level of the column, according to the `mice` help file. For an interval-level variable, the default will be predictive mean matching. Below is an example of how I imputed missing variables for the Lafrance and Gottardo (2005) data (imported as `lafrance`).

```
library(mice)
imp<-mice(lafrance)
complete(imp) #shows the completed data matrix
implafrance<-complete(imp) #name the new file to work with
```

A very cool function in R will create a graph of the observations of the data, leaving white lines where data is missing, as well as list the number of missing values and the percentage of your data that represents. It is a quick way to get a look at what is missing in your data set.

```
library(dprep)
imagmiss(lafrance, name="Lafrance")
```

```
Report on missing values for Lafrance :

Number of missing values overall: 3
Percent of missing values overall: 1.5
Features with missing values (percent):
NamingSpeed
    7.5

Percent of features with missing values: 20
Number of instances with missing values: 3
Percent of instances with missing values: 7.5
```

With the imputed data file (the right-hand graph in Figure 3.7) we no longer have the missing values.

```
imagmiss(implafrance, name="Lafrance imputed")
```

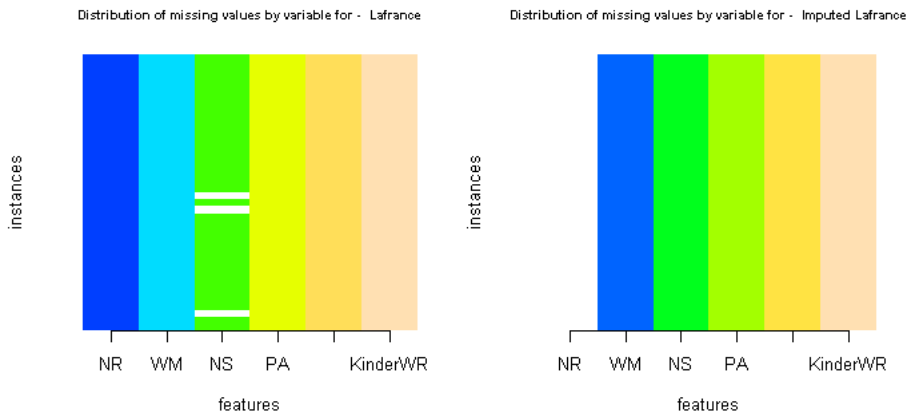


Figure 3.7 Original Lafrance and Gottardo (2005) data and with missing values imputed.

3.6 Transformations

Tabachnick and Fidell (2001) have a very useful table which gives recommendations for what kinds of transformations to use when data appears in various shapes. These transformations are simple to do. If you do decide to transform your data, you should check all of the indicators of normality and homogeneity of variances (as well as other assumptions that apply only in certain tests, such as the linearity assumption in correlation) with the transformed data to see whether any improvement has been achieved. Tabachnick and Fidell's recommendations are gathered into Table 3.1.

Table 3.1 Recommended transformations for data problems

<i>Distributional Shape</i>	<i>Recommended Transformation</i> (<i>X=Column You Want to Transform</i>)	<i>Transformation if Any Zeros Found in Data</i>
Moderate positive skewness	\sqrt{x}	
Substantial positive skewness	$\log_{10}(x)$	$\log_{10}(x + C^*)$
Severe positive skewness	$1/x$	$1/(x + C)$
Moderate negative skewness	$\sqrt{z^{**}-x}$	
Substantial negative skewness	$\log_{10}(z^{**}-x)$	
Severe negative skewness	$\log_{10}(z^{**}(-x))$	

*C is a constant that is added so that the smallest value will equal 1.

**Z is a constant that is added such that $(z-x)$ does not equal 0.

To perform transformations of the data in R Commander choose DATA > MANAGE ACTIVE VARIABLES IN DATA SET > COMPUTE NEW VARIABLE. The commands for transformations are not listed in the dialogue box itself. You will also have to add your own parentheses to the variable to show that the function, whether it be the logarithm or square root, is applying to the data. This procedure will produce a new column in your data set that you can then work with.

Throughout the exercises in this chapter we have seen that the GJTSCORE variable in DeKeyser (2000) is negatively skewed. I will walk the reader through an example of

transforming this data using R. Tabachnick and Fidell (2001) recommend using a reflect and square root transformation for moderately negatively skewed data. To determine the Z constant, we first find that the largest value in the data set is 199. We will add 1, so that $Z=200$. In R, I used R Commander's "Compute new variable" command and it filled it in like this:

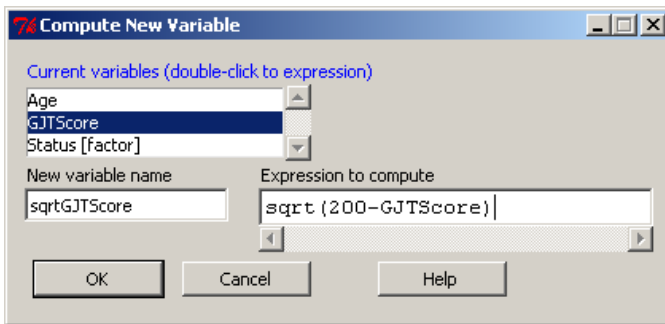


Figure 3.8 Computing a square root transformation in R.

The expression I write in the box is `sqrt(200-GJTScore)`. After obtaining this new column, I need to check on the distribution of this new variable, which I will do by looking at a histogram.

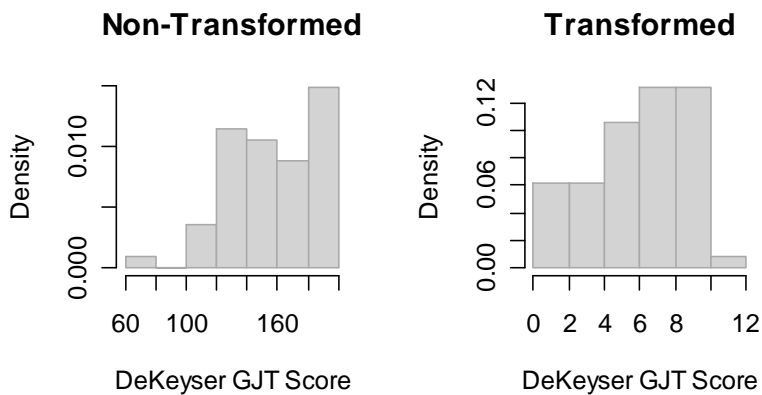


Figure 3.9 Checking the GJTScore variable with and without transformation for skewness.

Although the histogram in Figure 3.9 seems to show less skewness in the transformed variable, the skewness value of the non-transformed variable is -0.3 while the skewness of the transformed variable is -0.4 , a little worse but really not much of a difference. It's probably not worthwhile to transform this variable, given that interpreting the scale of the transformed variable is difficult.

3.7 Application Activity for Transformations

In previous exercises (from the online document “Describing data.Application activity_Exploring assumptions,” items 1 and 4) we have seen that the distribution of data for group 11 in the Flege, Yeni-Komshian, and Liu (1999) data is positively skewed. Perform the appropriate transformation on the data and look at the new histogram for this group. What is your conclusion about using a transformation for this variable?