The Basics of R for Windows

We will use the data set `timetrial.repeated.dat` to learn some basic code in R for Windows. Commands will be shown in a different font, e.g., `read.table`, after the command line prompt, shown here as `>`. After you open R, type your commands after the prompt in what is called the Console window. Do not type the prompt, just the command. R stores the variables you create in a working directory. From the File menu, you need to first change the working directory to the directory where your data are stored.

1. Reading in data files

Download the data file from `www.biostat.umn.edu/~lynn/correlated.html` to your local disk directory, go to File -> Change Directory to select that directory, and then read the data in:

```
> contact =  read.table(file="timetrial.repeated.dat", header=T)
```

Note: if the first row of the data file had data instead of variable names, you would need to use `header=F` in the `read.table` command.

Now you have named the data set `contact`, which you can then see in R just by typing its name:

```
> contact
```

or you can look at, for example, just the first 10 rows of the data set:

```
> contact[1:10,]

   time sex age shape trial subj
1  2.68   M  31   Box     1    1
2  4.14   M  31   Box     2    1
3  7.22   M  31   Box     3    1
4  8.00   M  31   Box     4    1
5  7.09   M  30   Box     1    2
6  8.55   M  30   Box     2    2
7  8.79   M  30   Box     3    2
8  9.68   M  30   Box     4    2
9  6.05   M  30   Box     1    3
10 6.25   M  30   Box     2    3
```

This data set has 6 variables, one each in a column, where `sex` and `shape` are character valued. Sometimes it is easier to deal with numeric variables. We will create a new variable called `female` using the `ifelse` command. This command will test each element of `sex` to see whether or not it is equal to `F`. If it is, the variable `female` will take the value 1; otherwise, the variable `female` will take the value 0. R uses a double equal sign (`==`) as a logical operator to test whether things are "equal." R uses a dollar sign (`$`) to refer to specific variables within a data set. We will create two new variables called `female` and `box` within the `contact` data set.

```
> contact$female = ifelse(contact$sex=='F',1,0)
> contact$box = ifelse(contact$shape=='Box',1,0)
```

Now `contact` has 8 variables:

```
> contact[1:5,]

  time sex age shape trial subj female box
1 2.68   M  31   Box     1    1      0   1
2 4.14   M  31   Box     2    1      0   1
3 7.22   M  31   Box     3    1      0   1
4 8.00   M  31   Box     4    1      0   1
5 7.09   M  30   Box     1    2      0   1
```

In order to avoid calling variables by their full name, such as `contact$time, contact$sex,` etc., we will use the `attach` command. This allows us to refer to the variables, such as `time`, `sex`, etc., without referring to the data set to which they are attached.

```
> attach(contact)
```

Notes:
1. R is case sensitive. `time`, `Time`, and `TIME` are different for R.
2. If you are going to repeat commands that you have typed in already, you can use the ↑ and ↓ on your keyboard to avoid typing the commands again.
3. Use `?` to get quick help for a command in R. For example, type `?attach` in the R Console window. Then a new window will pop up in which the usage of the command `attach` will be explained in detail.

2. Making a scatterplot

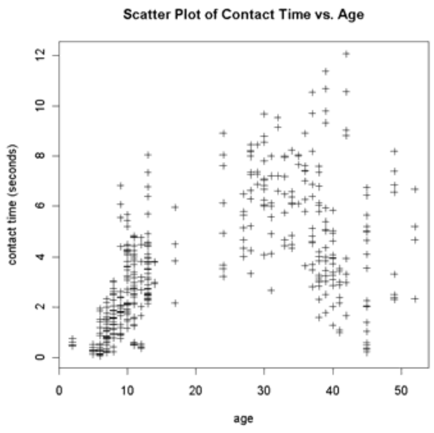
In R, you can plot interactively or in batch mode. Batch mode means that you create a plot and save it directly to a figure file before looking at it; interactive mode means you make the plot while you are looking at it, and then save it to a file when you are done.

By typing in the following command, a scatterplot of contact time (y-axis) vs. age (x-axis) will appear in a new graphics window. Each data point is a cross; different values of the option `pch` represent different symbols for the data points.

```
> plot(age,time,xlab="age",ylab="contact time (seconds)",pch=3)
```

The following command will add a title to the plot:

```
> title("Scatter Plot of Contact Time vs. Age")
```



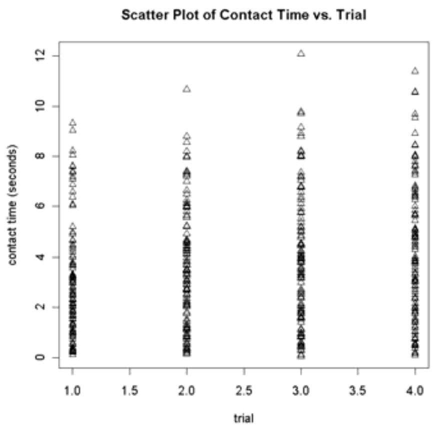
Once the plot is done, activate the graphics window by moving the mouse to it. Then go to the pull-down menu File -> Save as to save the plot as a separate file. You can choose the type of file that you want to use for the plot. If you want to insert the plot directly into a document file instead, go to the pull-down menu File -> Copy to the clipboard and then paste the plot into your file. The plots in this tutorial have been inserted this way.

If you want to plot in batch mode, you will need a command to open the figure file (such as `postscript`) and another command to close the figure file once the plotting commands have been given (such as `graphics.off`).

```
> ps.options(paper="letter")
> postscript(file="c:/coursework/contact.scatter.ps")
> plot(age,time,xlab="age",ylab="contact time (seconds)",pch=3)
> title("Scatter Plot of Contact Time vs. Age")
> graphics.off()
```

Using the `postscript` command, your plot will be saved as a `.ps` file, and the plot will not appear in the graphics window. Here is another example to plot interactively contact time vs. trial. Each data point is now represented by a triangle.

```
> plot(trial,time,xlab="trial",ylab="contact time (seconds)",pch=2)
> title("Scatter Plot of Contact Time vs. Trial")
```



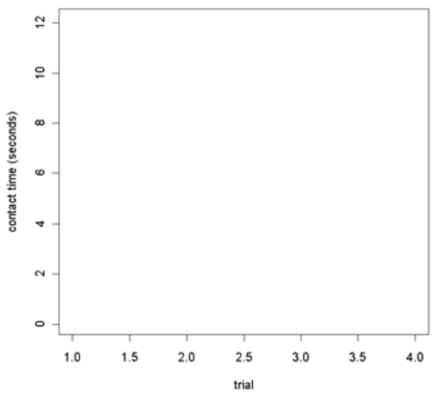
Notes: If you use interactive plotting, you should save the first plot before creating the second. Otherwise, the second plot will replace the first one in the graphics window. If you use # in front of a line, R will skip the line; # means to comment out that line.

```
#postscript(file="c:/coursework/contact.scatter.ps")
#graphics.off()
```

3. Making a scatterplot for subsets of the data

The following command is similar to what we have seen already except for `type="n",` which will result in a plot without data points.

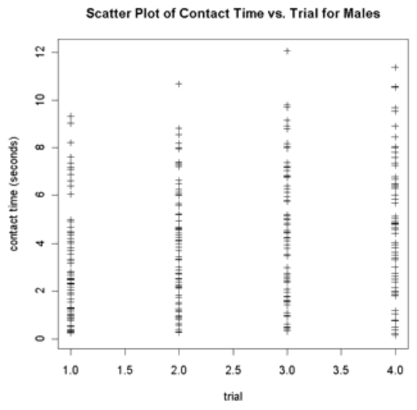
```
> plot(trial,time,type="n",xlab="trial",ylab="contact time (seconds)")
```



When making multiple plots of different subsets of the same data, this is an easy way to make sure the axes are always scaled in exactly the same way. The following commands will then add data points that correspond to males only and add a title to the plot.

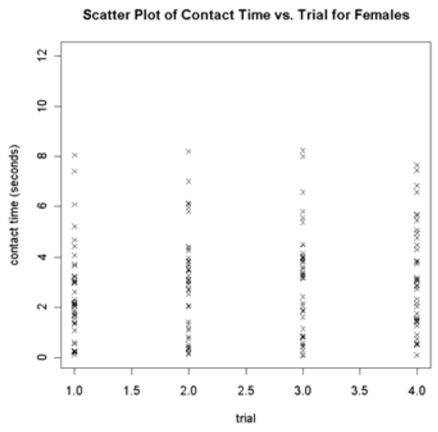
```
> points(trial[female==0],time[female==0],type="p", pch=3)
> title("Scatter Plot of Contact Time vs. Trial for Males")
```

Note: `trial[female==0]` is sub-setting syntax. It will pick out the values of trial which also have `female` equal to 0. `time[female==0]` will pick out the values of time which have `female` equal to 0. `type="p"` means points (rather than lines) will be added to the plot.



The following commands will give a scatterplot of contact time vs. trial for females:

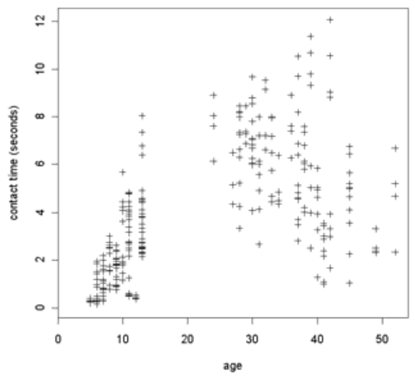
```
> plot(trial,time,type="n",xlab="trial",ylab="contact time (seconds)")
> points(trial[female==1],time[female==1],type="p", pch=4)
> title("Scatter Plot of Contact Time vs. Trial for Females")
```



4. Adding data summaries (Loess smooth line)

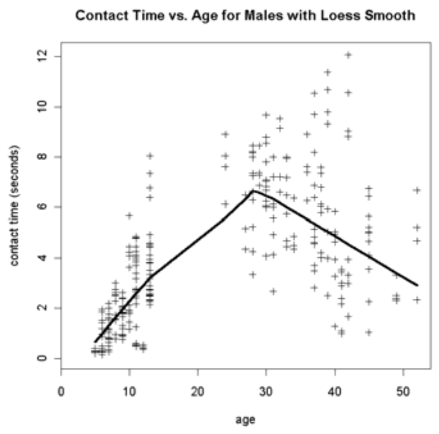
A loess curve is a type of “smoother,” which is a type of line-fitting technique that does NOT require that the line follow a particular structure, such as linear, quadratic, etc. We will learn about how a loess smooth is calculated in class; here we will show you how to use R to calculate and graph one. First, the following commands return a scatter plot of contact time vs age for males.

```
> plot(age,time,type="n",xlab="age",ylab="contact time (seconds)")
> points(age[female==0],time[female==0],type="p",pch=3)
```



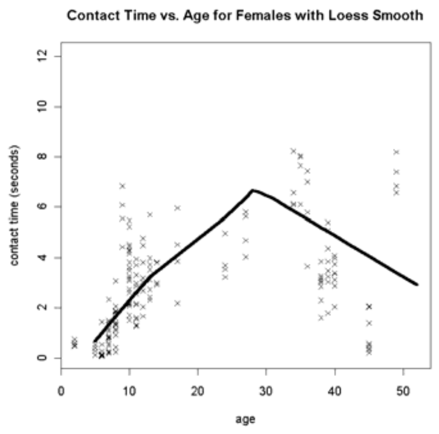
Next, the following commands will add a loess smooth line and a title to the scatter plot.

```
> lines(lowess(age[female==0],time[female==0]),lwd=3)
> title("Contact Time vs. Age for Males with Loess Smooth")
```



The following commands do the same for females.

```
> plot(age,time,type="n",xlab="age",ylab="contact time (seconds)")
> points(age[female==1],time[female==1],type="p",pch=4)
> lines(lowess(age[female==1],time[female==1]),lwd=4)
> title("Contact Time vs. Age for Females with Loess Smooth")
```



5. Line plots for each person

For longitudinal repeated measures data, we often want to see how an individual's data change across time. First, we create a scatterplot of contact time vs. trial for all subjects.

```
> plot(trial,time,type="n",xlab="trial",ylab="contact time (seconds)")
> points(trial,time,type="p",pch=3)
```

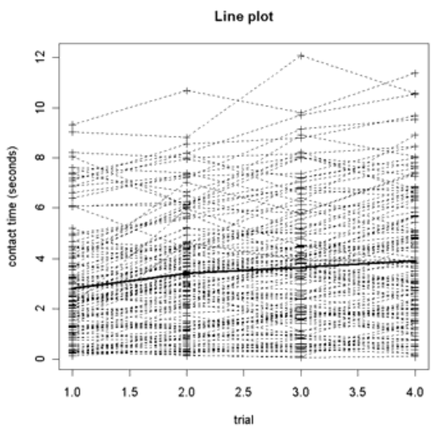
Then we want to connect the points for each person separately. This requires writing a loop, where each iteration in the loop connects the points for one person. In order to do this, we need to know which observations in the data belong to which person. This information is in the subject ID variable called `subj`.

```
> for(i in unique(subj))
> {
> lines(trial[subj==i],time[subj==i],type="l",lty=2)
> }
```

This data set involves 108 subjects, so the variable `subj` has 432 elements containing integer values from 1, 2, 3, etc. up to 108. Each value is repeated 4 times (one value for each person, repeated for each of the four trials). The command `unique(subj)` will pick out the unique values of `subj` and return them as a new variable. Thus `for(i in unique(subj))`says `i` should go from 1 to 108 and increase by 1 each time. In the i^th iteration, a line will be plotted to connect the four points `(trial[subj==i],time[subj==i])` that correspond to the i^th person.

The following two lines will then add the Loess smooth line and title to the plot.

```
> lines(lowess(trial, time),lwd=3)
> title("Line plot")
```



Now let's create the line plot for males only. First, we'll need to create a new variable that tells the number of trials for each person.

```
> ntimes=4
```

Then, we create a scatter plot of contact time vs. trial for males.

```
> plot(trial,time,type="n",xlab="trial",ylab="contact time (seconds)")
> points(trial[female==0],time[female==0],type="p",pch=3)
```

Now we plot the lines for all males.

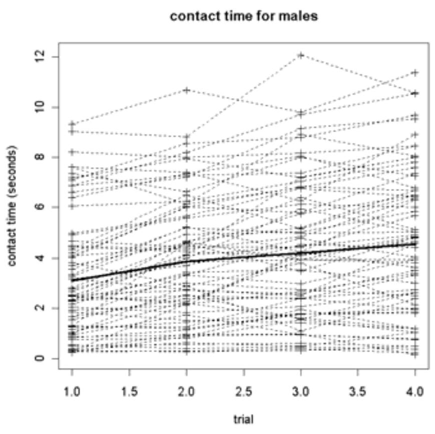
```
> for(i in unique(subj))
> {
> if(female[ntimes*i]==0) {lines(trial[subj==i],time[subj==i],type="l",lty=2)}
> }
```

Let's go through the first two iterations. Note that `ntimes=4.` In the 1st iteration, `i=1`, so `ntimes*i=4`. If the 4th element of `female` is 0, i.e. if the 1st subject is a male, we'll plot the lines for that person. If the 4th element of `female` is 1, i.e. if the 1st subject is a female, we'll skip the `lines` command and move to the next iteration.

In the 2nd iteration, `i=2`, so `ntimes*i=8`. If the 8th element of female is 0, i.e. if the 2nd subject is a male, we'll plot the lines for that person. Otherwise, we'll move on.

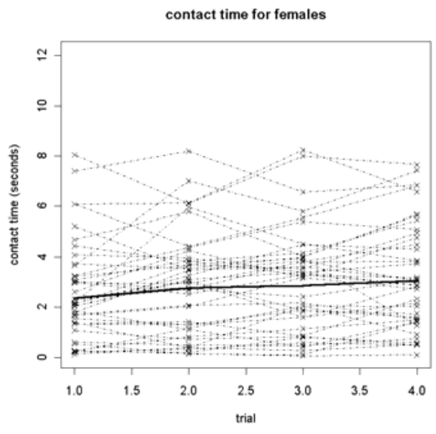
Thus, we'll repeat the steps for each one of the 108 subjects but plot the lines only for the males. Lastly, the following two lines will add the Loess smooth line and title to the plot.

```
> lines(lowess(trial[female==0],time[female==0]),lwd=3)
> title("contact time for males")
```



The following commands give the line plot for females.

```
> plot(trial,time,type="n",xlab="trial",ylab="contact time (seconds)")
> points(trial[female==1],time[female==1],type="p",pch=4)
> for(i in unique(subj))
> {
> if(female[ntimes*i]==1) {lines(trial[subj==i],time[subj==i],type="l",lty=4)}
> }
> lines(lowess(trial[female==1],time[female==1]),lwd=3)
> title("contact time for females")
```

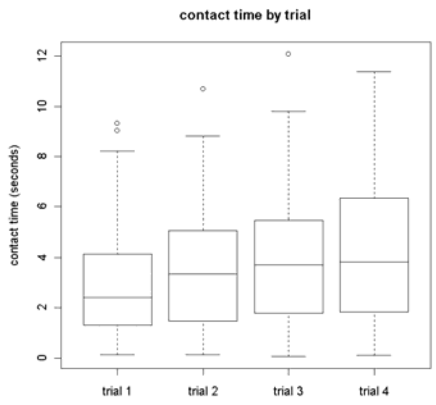


Notice how starting the plot with the `plot` command including `type="n"` in both the male and female plots has forced the axes to be identical for the male and female plots.

6. Box plots

Box plots are a good way to see both the center and the spread of data. The following command gives the box plots of contact time separately by trial. `time[trial==1]` will pick out all contact times for trial 1.

```
> boxplot(time[trial==1],time[trial==2],time[trial==3],time[trial==4],
   names=c("trial 1","trial 2","trial 3","trial 4"),
   ylab="contact time (seconds)")
> title("contact time by trial",cex=0.8)
```



7. Getting ordinary least squares (OLS) residuals

Since the age trend looks approximately quadratic, before we fit any models we need to create a variable for age squared. We will also center the age variable around its mean.

```
> cage = age - mean(age)
> cage2 = cage*cage
```

The following line regress contact time on all possible covariates. Here `trial` has been treated as a categorical variable using the `factor` command and the regression is fit with the linear model (`lm`) command.

```
> ols.fit = lm( time ~ female*cage*box*factor(trial) +
   female*cage2*box*factor(trial) )
```

The following lines get the OLS residuals and create four new variables that correspond to the OLS residuals from the 4 trials.

```
> res = ols.fit$residuals
> res1 = res[trial==1]
> res2 = res[trial==2]
> res3 = res[trial==3]
> res4 = res[trial==4]
```

In longitudinal data, it is important to understand how the repeated measures are correlated with each other across time. The following command will return the correlation matrix of the residuals from the 4 trials.

```
> cor(cbind(res1,res2,res3,res4))
```

If you want to save the result to a file, you could use the `sink` command. Results of all commands given in between two `sink` commands will be dumped to the specified file.

```
> sink("correlation.out.txt")
> cor(cbind(res1,res2,res3,res4))
> sink()
```

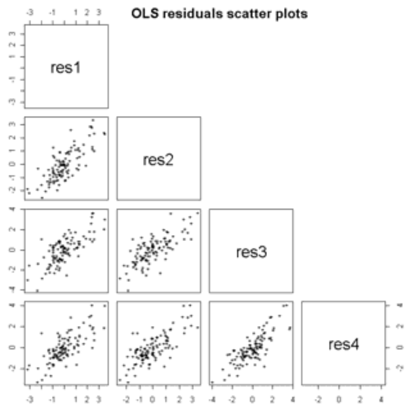
If you open the file `correlation.out.txt`, you'll see the following matrix.

```
          res1      res2      res3      res4
res1 1.0000000 0.7838540 0.7340572 0.7359904
res2 0.7838540 1.0000000 0.7424664 0.7456868
res3 0.7340572 0.7424664 1.0000000 0.8000878
res4 0.7359904 0.7456868 0.8000878 1.0000000
```

8. Multivariate scatter plots

The following command creates what is called a scatter plot matrix, that is, a matrix of small sub-plots of residuals from each trial plotted against each other.

```
> pairs(cbind(res1,res2,res3,res4),upper.panel=NULL,pch=42)
> title("OLS residuals scatter plots")
```



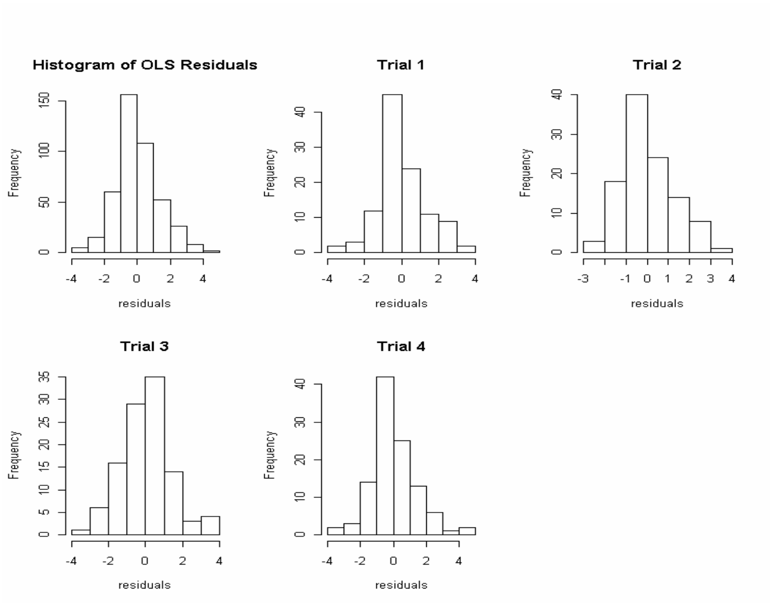
9. Multiple plots in one figure

You can display multiple plots in one figure using the command `par` with the option `mfrow`. The following command will divide the page into `2x3` small pieces (2 rows and 3 columns).

```
> par(mfrow=c(2,3))
```

Then you can put plots in each of these small areas sequentially. The next commands create the histograms of all OLS residuals, overall and separately by trial. The 5 histograms will be placed in one figure.

```
> hist(res,xlab="residuals",main="Histogram of OLS Residuals")
> hist(res1,xlab="residuals",main="Trial 1")
> hist(res2,xlab="residuals",main="Trial 2")
> hist(res3,xlab="residuals",main="Trial 3")
> hist(res4,xlab="residuals",main="Trial 4")
```



10. Adding extra text to plots

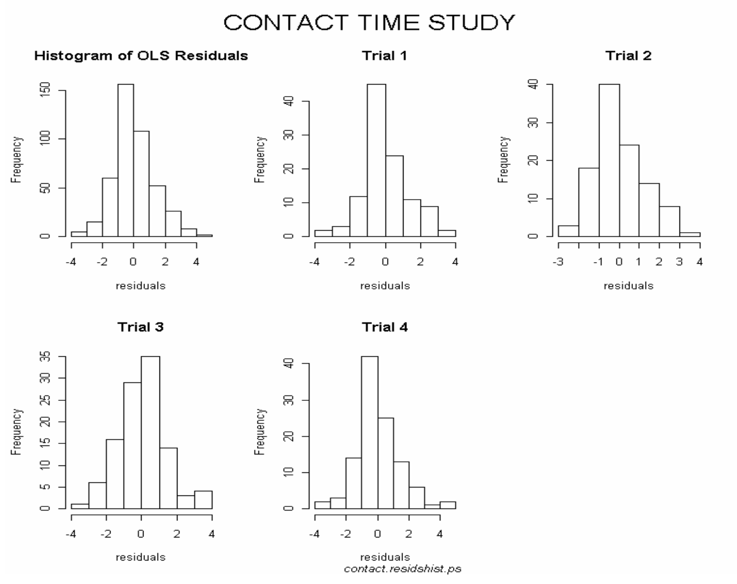
The mtext command can be used to add a title across all the smaller plots at the top and a title command with the sub option can be use to put words across all the smaller plots at the bottom. The following commands create a figure of 5 histograms with a figure title and a filename printed at the right lower corner.

```
> par(mfrow=c(2,3),oma=c(0,0,3,0),pch=42,font.sub=3,adj=0.5)
> hist(res,xlab="residuals",main="Histogram of OLS Residuals")
> hist(res1,xlab="residuals",main="Trial 1",)
> hist(res2,xlab="residuals",main="Trial 2")
> hist(res3,xlab="residuals",main="Trial 3")
> hist(res4,xlab="residuals",main="Trial 4")
> mtext(side=3,outer=T,line=0,"CONTACT TIME STUDY",cex=1.3)
> par(adj=1)
> title(sub="contact.residshist.ps")
```

When we use mtext, we need to specify values in the par option oma as well. This option changes the margins of the figure to accommodate the figure title. oma is a vector with four elements of the form c(bottom,left,top,right) where each element gives the size of the outer margins for that side of the figure. For example, oma=c(0,0,3,0) lowers the top margin by 3 lines to make room for the figure title "CONTACT TIME STUDY". The font.sub option tells R to make the sub-title in a font of type 3, which corresponds to italics. The adj option, when set to 0.5, tells R that text should be centered on the page.

In the mtext command, the option side=3 means that the title will be placed at the top of the figure. The option outer=T tells R that the title should go in the outer margin of the figure (that is, outside of the plotting space) and the line=0 options tells R that the title should go in the 0th line of the outer margin. Lastly, the cex=1.3 option means the font size should be scaled up by 30% (multiplied by 1.3).

The second par command, just before the title command, tells R that now the adj option should be set equal to 1, which corresponds to right-justification (the text slides to the right-hand side, not centered and not on the left).



Here is another example to explain the usage of `mtext` and `oma`. The following codes give the first plot shown below.

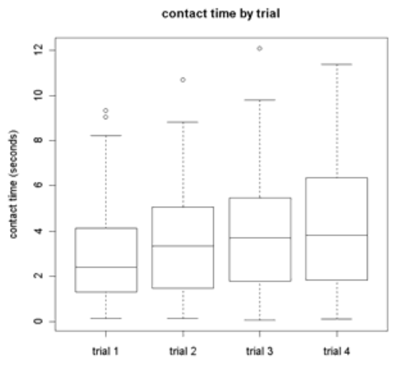
```
> boxplot(time[trial==1],time[trial==2],time[trial==3],time[trial==4],
   names=c("trial 1","trial 2","trial 3","trial 4"),
   ylab="contact time (seconds)")
> title("contact time by trial",cex=0.8)
```

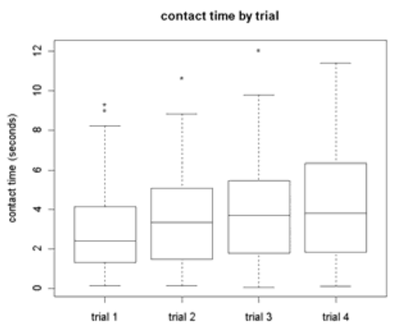
Adding the following line to the beginning of the previous commands results in the second plot.

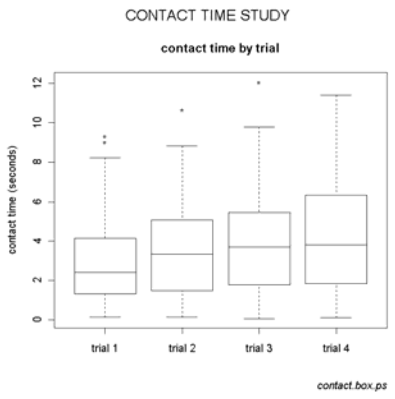
```
> par(mfrow=c(1,1),oma=c(0,0,3,0),pch=42,font.sub=3,adj=0.5)
```

Now add a few more lines in the end and we get the third plot.

```
> mtext(side=3,outer=T,line=0,"CONTACT TIME STUDY",cex=1.3)
> par(adj=1)
> title(sub="contact.box.ps")
```







11. Quitting R

You can quit out of R by typing `q()` or by exiting from the menus. R will ask you if the workspace should be saved. If you say no, then any variables you created will be deleted. If you say yes, then all variables you created will be saved for when you next open R. In general, you should not need to save all your variables.

Appendix

The original data set has all observations for the same person in one row.

```
   Sex Age Shape Trial1 Trial2 Trial3 Trial4
1    M  31   Box   2.68   4.14   7.22   8.00
2    M  30   Box   7.09   8.55   8.79   9.68
3    M  30   Box   6.05   6.25   7.04   7.80
4    M  27   Box   4.35   6.50   5.17   6.50
5    M  30   Box   4.08   6.00   6.82   6.68
```

```
6    M  28   Box   8.22   8.01   8.18   8.45
7    M  34   Box   4.51   4.35   4.87   6.38
8    M  28   Box   7.36   6.63   7.99   7.26
9    M  28   Box   3.34   5.24   4.27   6.29
10   M  33   Box   7.19   7.96   7.18   7.99
… …
```

Following is the SAS code to rearrange the data set so that each row has only one observation for each person. The code also creates the new data file named `timetrial.repeated.dat` and saves it to the current directory. This is the data set that we use in this tutorial.

```
data contact;
    infile '/home/muskie/correlated.data/tracking.dat' firstobs=2;
    input Sex $ Age Shape $ Trial1-Trial4;
run;

data repdat;
    set contact;
    time=trial1;trial=1;subj=_N_;output;
    time=trial2;trial=2;subj=_N_;output;
    time=trial3;trial=3;subj=_N_;output;
    time=trial4;trial=4;subj=_N_;output;
run;

data _NULL_;
    set repdat;
    file 'timetrial.repeated.dat';
    if _N_=1 then put "time sex age shape trial subj";
    put time sex age shape trial subj;
run;
```

Last modified: 27 Jul 2004. Thanks to Haijun Ma, PhD Student, for her help in preparing the tutorial.