

1 Playing with R

Before you can play with R, you need to install it: <http://www.r-project.org/> Once you have it installed, you'll want to read Chapter 1 of Dalgaard. There are also several excellent beginning tutorials online linked to from cTools.

2 Age guessing

Let's play with the age guessing data that you all generated last time in class. We may first want to change the working directory, using the `File>Change dir...` option. This is where R will look for data files by default, output images, and save your work (`File>Change dir...`). Otherwise we need to specify the full path to the file we are trying to load, e.g. `"c:/dir1/dir2/ageguessing.dat"`.

```
> ages <- read.table("ageguessing.dat", head=TRUE, row.names=1)}
```

Here we loaded the file `ageguessing.dat` using the `read.table` command. The file happens to have tab separated columns. But if they had been separated by commas, we would have added an extra `sep=","` argument. As it is, we give the command two arguments, the first lets it know that the first row contains the column headers (`head=TRUE`), in our case the numbers assigned to the different photographs (1..12), the second that the first column is a list of names for the rows (`row.names=1`), in our case the names of the groups doing the guessing (A..I).

For more information on loading data from text files, please refer to <http://pbil.univ-lyon1.fr/library/base/html/read.table.html>.

Let's have a peek at what we have loaded

```
> ages
```

```
      X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12
A      35 60 42 38 64 32 25 55 32  23  40  48
B      33 62 44 40 54 23 28 64 26  30  47  48
C      37 64 36 39 57 31 27 59 23  25  42  42
D      37 68 42 40 65 28 34 60 24  27  45  47
E      33 65 42 48 60 28 28 56 26  30  51  40
F      37 60 35 42 57 25 27 60 28  29  50  45
G      35 63 45 43 65 31 35 60 36  28  55  42
H      31 62 40 37 57 25 30 60 25  30  45  45
I      32 63 36 46 61 37 29 59 26  25  49  49
truth 27 54 51 55 69 24 37 62 34  33  47  40
```

Aha, it all made it nicely into our 'ages' data frame. We would now like to plot the guesses of the people's ages and compare with the ages that they themselves reported. Currently, our data is stored in a `dataframe` <http://www.maths.lth.se/help/R/.R/library/base/html/data.frame.html>, a matrix with some other attributes attached. Some of the plotting functions will have nothing to do with a dataframe, so we need to convert the data to a `matrix` first. We will be plotting just the group guesses first, so that means that we will be selecting the first 9 rows and all of the columns:

```
> as.matrix(ages[1:9,])
```

```

  X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12
A 35 60 42 38 64 32 25 55 32 23 40 48
B 33 62 44 40 54 23 28 64 26 30 47 48
C 37 64 36 39 57 31 27 59 23 25 42 42
D 37 68 42 40 65 28 34 60 24 27 45 47
E 33 65 42 48 60 28 28 56 26 30 51 40
F 37 60 35 42 57 25 27 60 28 29 50 45
G 35 63 45 43 65 31 35 60 36 28 55 42
H 31 62 40 37 57 25 30 60 25 30 45 45
I 32 63 36 46 61 37 29 59 26 25 49 49

```

The plotting function that we will be passing our data to `matplot` <http://www.maths.lth.se/help/R/.R/library/graphics/html/matplot.html>, plots the columns of matrices, but we actually want to plot the rows, showing the guesses for each photo by group. So we need to flip it around using the transpose function `t()` like this:

```

> justgessesestranspose = t(as.matrix(ages[1:9,]))
> justgessesestranspose

```

```

  A B C D E F G H I
X1 35 33 37 37 33 37 35 31 32
X2 60 62 64 68 65 60 63 62 63
X3 42 44 36 42 42 35 45 40 36
X4 38 40 39 40 48 42 43 37 46
X5 64 54 57 65 60 57 65 57 61
X6 32 23 31 28 28 25 31 25 37
X7 25 28 27 34 28 27 35 30 29
X8 55 64 59 60 56 60 60 60 59
X9 32 26 23 24 26 28 36 25 26
X10 23 30 25 27 30 29 28 30 25
X11 40 47 42 45 51 50 55 45 49
X12 48 48 42 47 40 45 42 45 49

```

Now we are ready for `matplot()`.

```

> matplot(justgessesestranspose,xlab="photo #",ylab="age",
+ main="age guesses vs. truth",verbose=TRUE,pch=row.names(ages)[1:9])

```

```

matplot: doing 9 plots with col= ("1" "2" "3" "4" "5" "6")
pch= ("A" "B" "C" "D" "E" "F" "G" "H" "I") ...

```

Here is what we told R to do. First, we gave it our data, now in matrix format and transposed. Next we gave it an x label, y label and a title, and told it to be verbose, that is, give us some feedback about what it's doing. Finally, we had it label each point with the name of the row (which corresponds to the group name) using the `row.names()` function. It said that it was generating nine plots (for our nine groups), using the colors 1...6, which are presumably the colors we see on the plot below, and assigning different markers to each group, in this case the row names of the original data frame. The complexity of doing this plot may seem horrifying, and frankly, it is. Let's just hope we get used to it. Because we're not done yet.

Next we want to add the “true” ages of the people depicted in the photographs. We use the `points()` function, which *adds* points to an existing plot.

We create just a vector of 12 ordered points that we can plot against (since we have a *y* vector but not an *x* vector).

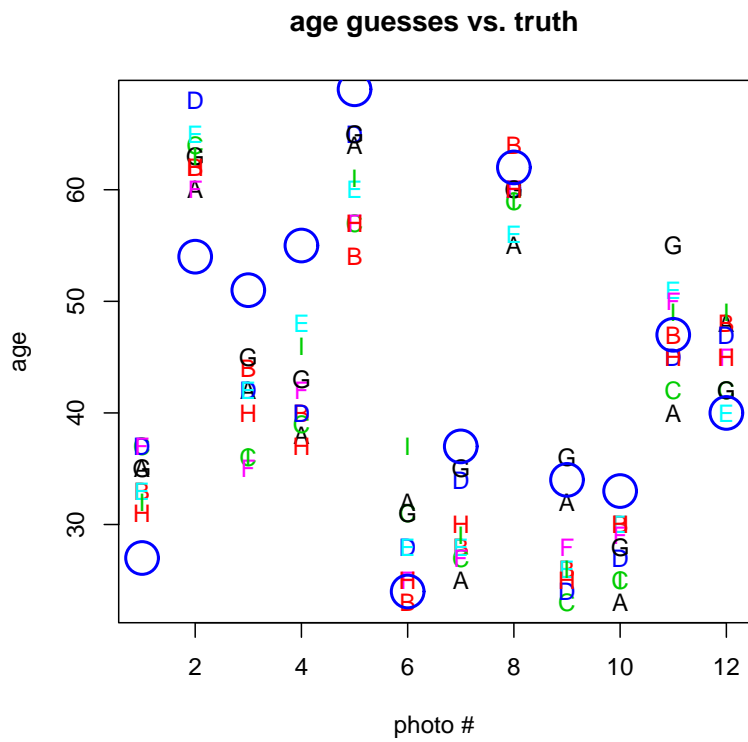


Figure 1: Figure showing the group guesses vs. the true age.

```
> xpoints = 1:12
> xpoints
```

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12
```

OK, now we plot the true ages against this dummy x vector:

```
> truth = ages[10,]
> points(xpoints,truth,pch=21,col='blue',cex=3,lwd=2)
```

We told R to plot empty circles with the `pch=21` argument, to make them blue with the `col='blue'` argument, to make them bigger with the `cex=3` argument, and to make the line thicker with the `lwd=2` (line width = 2) argument. We could have skipped all the arguments and gotten small black inconspicuous dots. This way we get nice big blue circles that everyone would notice. And voila, we've got a nice plot (Figure 1) we can now interpret (pause to interpret...).

We can sort of see that some groups were better guessers than others, so let's calculate the errors that each group made. We subtract the true value from each guessed value

```
> guesserrors = sweep(as.matrix(ages[1:9,]),2,as.matrix(truth),"-")
> guesserrors
```

```
  X1 X2  X3  X4  X5 X6  X7 X8  X9 X10 X11 X12
A  8  6  -9 -17  -5  8 -12 -7  -2 -10  -7  8
B  6  8  -7 -15 -15 -1 -9  2  -8  -3  0  8
C 10 10 -15 -16 -12  7 -10 -3 -11  -8  -5  2
D 10 14  -9 -15  -4  4  -3 -2 -10  -6  -2  7
E  6 11  -9  -7  -9  4  -9 -6  -8  -3  4  0
F 10  6 -16 -13 -12  1 -10 -2  -6  -4  3  5
G  8  9  -6 -12  -4  7  -2 -2  2  -5  8  2
H  4  8 -11 -18 -12  1  -7 -2  -9  -3  -2  5
I  5  9 -15  -9  -8 13  -8 -3  -8  -8  2  9
```

Note that we had to use `as.matrix` to make `sweep()` happy. What is `sweep`? And why do dataframes make it unhappy? Who knows. It's R. OK, `sweep` appears to be something built on top of `apply()` which we'll get into later. Normally, you would use it to subtract the mean from each value, or some similar thing. We could also have used a for loop as follows. First we set the `agesdiff` variable to be the guessed age, and then from each group's guesses we subtract the truth.

```
> agesdiff=ages[1:9,]
> for (i in 1:9) {
+ agesdiff[i,] = agesdiff[i,]-truth + }
> agesdiff
```

Well, moving on, we can now get some summary statistics. For example, we can run the `summary` command. For each photograph it tells us things like the median, the upper and lower quartiles, and the minimum and maximum data values. Running it on our error matrix we get the following:

```
> summary(guesserrors)
```

```
      X1      X2      X3      X4      X5
Min.   : 4.000  Min.   : 6  Min.   : -16.00  Min.   : -18.00  Min.   : -15
1st Qu.: 6.000  1st Qu.: 8  1st Qu.: -15.00  1st Qu.: -16.00  1st Qu.: -12
Median : 8.000  Median : 9  Median :  -9.00  Median : -15.00  Median :  -9
Mean   : 7.444  Mean   : 9  Mean   : -10.78  Mean   : -13.56  Mean   :  -9
3rd Qu.:10.000  3rd Qu.:10  3rd Qu.:  -9.00  3rd Qu.: -12.00  3rd Qu.:  -5
```

Max. :10.000	Max. :14	Max. : -6.00	Max. : -7.00	Max. : -4
X6	X7	X8	X9	
Min. :-1.000	Min. :-12.000	Min. :-7.000	Min. :-11.000	
1st Qu.: 1.000	1st Qu.: -10.000	1st Qu.: -3.000	1st Qu.: -9.000	
Median : 4.000	Median : -9.000	Median : -2.000	Median : -8.000	
Mean : 4.889	Mean : -7.778	Mean : -2.778	Mean : -6.667	
3rd Qu.: 7.000	3rd Qu.: -7.000	3rd Qu.: -2.000	3rd Qu.: -6.000	
Max. :13.000	Max. : -2.000	Max. : 2.000	Max. : 2.000	
X10	X11	X12		
Min. :-10.000	Min. :-7.0000	Min. :0.000		
1st Qu.: -8.000	1st Qu.: -2.0000	1st Qu.:2.000		
Median : -5.000	Median : 0.0000	Median :5.000		
Mean : -5.556	Mean : 0.1111	Mean :5.111		
3rd Qu.: -3.000	3rd Qu.: 3.0000	3rd Qu.:8.000		
Max. : -3.000	Max. : 8.0000	Max. :9.000		

`summary` obviously works on columns, which in this case each represent a photograph. We can also plot a boxplot, which is much easier to look at, and will show us the median, the two middle quartiles inside the box, the upper and lower quartiles as the 'whiskers'. If outliers are present (defined as extending more than 1.5 the inter-quartile range), they are shown as individual points. Boxplot seems to want a data frame in order to draw a separate box and whisker plot for each photo. So we create a data frame, attach it to `guesserrors`, and ask for a boxplot. Just as in the previous figure, we see that some people looked younger than their self-reported age, and some were older. But we can also draw other conclusions...

```
> guesserrors.df = data.frame(guesserrors)
> boxplot(guesserrors.df,xlab="photo #",ylab="guess",main="boxplot of the errors in age guesses")
> dev.copy2eps(file="guessesbyphotoboxplot.eps")
```

```
windows
2
```

What can we observe about the difficulty of guessing an age for a particular photograph? Which is more often closer to the true age, the mean or the median?

But what we'd really like to know right now is what the average error is per group, and so confirm the winner. First, let's get a boxplot of the guesses by group, which means that we need to use the transpose function `t()`.

```
> boxplot(data.frame(t(abs(guesserrors))),
+ main="boxplot of the errors in age guesses")
> dev.copy2eps(file="guessesbygroupboxplot.eps")
> summary(t(abs(guesserrors)))
```

A	B	C	D
Min. : 2.00	Min. : 0.000	Min. : 2.000	Min. : 2.000
1st Qu.: 6.75	1st Qu.: 2.750	1st Qu.: 6.500	1st Qu.: 3.750
Median : 8.00	Median : 7.500	Median :10.000	Median : 6.500
Mean : 8.25	Mean : 6.833	Mean : 9.083	Mean : 7.167
3rd Qu.: 9.25	3rd Qu.: 8.250	3rd Qu.:11.250	3rd Qu.:10.000
Max. :17.00	Max. :15.000	Max. :16.000	Max. :15.000
E	F	G	H
Min. : 0.000	Min. : 1.000	Min. : 2.000	Min. : 1.000
1st Qu.: 4.000	1st Qu.: 3.750	1st Qu.: 2.000	1st Qu.: 2.750
Median : 6.500	Median : 6.000	Median : 5.500	Median : 6.000
Mean : 6.333	Mean : 7.333	Mean : 5.583	Mean : 6.833



Figure 2: Figure showing the boxplots of group guesses vs. the true age.

```

3rd Qu.: 9.000   3rd Qu.:10.500   3rd Qu.: 8.000   3rd Qu.: 9.500
Max.    :11.000   Max.    :16.000   Max.    :12.000   Max.    :18.000
I
Min.    : 2.000
1st Qu.: 7.250
Median  : 8.000
Mean    : 8.083
3rd Qu.: 9.000
Max.    :15.000

```

Note that we used the `abs()` function to take the absolute value of the error. We see that group G did indeed have the lowest mean absolute error of around 5, which was how confident the class was to begin with in their guessing ability.

boxplot of the errors in age guesses

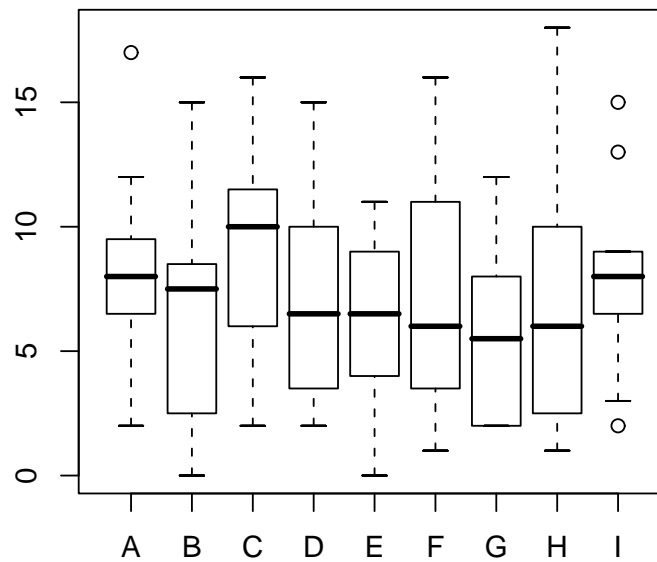


Figure 3: Figure showing the boxplots of group guesses vs. the true age.

3 Handedness

In class you took a handedness survey. What do you think the histogram will look like? We read in the data and check to see that what we see that we did so correctly.

```

> hand <- read.table("handedness.txt",head=TRUE)
> hand

```

```

leftcount rightcount

```

```
1         2         12
2         9         7
3         5         16
4         3         12
. . .
. . .
```

Now we compute the handedness ratio: $(r - l)/(r + l)$, and plot a histogram.

```
> hand.handedness = (hand$right-hand$left)/(hand$right+hand$left)
> hist(hand.handedness)
> hist(hand.handedness,breaks=20)
```

The second command tells the histogram function that we would like 20 bins. The rest awaits in the problem set...