Data frames

If you have a set of measurements on the same set of subjects, it's useful to organize them into a single structure called a data frame. A data frame is like a matrix, meaning it has rows and columns. Each column is a variableand each row is a subject.

```
> sex = c('m','m','f','f','f')
> age = c(19,23,4,35,34)
> height = c(67,63,49,74,57)
> data = data.frame(sex,age,height)
> data
sex age height
1 m 19 67
2 m 23 63
3 f 4  49
4 f 35 74
5 f 34 57
```

You can retrieve the names of your variables. Notice this is a vector of strings.

```
> names(data)
[1] "sex" "age" "height"
```

You can change the variable names:

```
> names(data) = c("gender","ageInYears",”heightInInches”)
```

Another useful function is `summary()`, which gives several descriptive statistics at once. You can apply `summary()` to a data frame to get summaries of all variables at once. This is usually the first command you'll type when you open a dataset, to get an initial impression of the data.

```
> summary(data)
sex          age         height
f:3     Min.     : 4 Min.     :49
m:2     1st Qu.  :19 1st Qu.  :57
        Median   :23 Median   :63
        Mean     :23 Mean     :62
        3rd Qu.  :34 3rd Qu.  :67
        Max.     :35 Max.     :74
```

Notice the summary for `sex` is simpler—all we get is frequencies, since the values of this variable aren’t numerical.

Some other useful functions:

The size of a data frame – first rows (subjects), then columns (variables):

```
> dim(data)
[1] 5 3
```

The length of a vector (i.e. of a single variable). This tells us how many observations we have:

```
> length(data$sex)
```

Reorder a vector from lowest to highest

```
> sort(data$age)
```

Components of a data frame

Components of a data frame work just like components of a vector, but you have to specify both the row and then the column.

```
> data[2,1]
```

This gives the second entry in the first column, meaning the value of Variable 1 for Subject 2.

You can also get a whole row by omitting the column index, or a whole column by omitting the row index (just be sure not to forget the comma). This shows all the data from subject 1 (the first row):

```
> data[1,]
```

This shows variable 1 for all subjects (the first column):

```
> data[,1]
```

As with components of vectors, you can look at multiple rows or columns at once. Here are the data for the first 3 subjects:

```
> data[1:3,]
```

You could store this as a new data frame

```
> datanew = data[1:3,]
```

Another way to access a single variable from a data frame is with the `$` symbol.

```
> data$age
[1] 19 23 4 35 34
> data$age[1]
[1] 19
```

Plotting data

You can make a histogram using the `hist()` function

```
> hist(X)
```

Set the number of bins:

```
> hist(X, breaks=10)
```

Make a scatterplot to see the relationship between two variables:

```
> plot(X,Y)
```

Each point is this plot corresponds to a single subject. The horizontal position of the point is that subject's value for `X`. The vertical position of the point is that subject's value for `Y`.

Frequency distribution tables

Create a set of data. Include at least 20 scores, and make sure there are many repetitions.

```
> X = c(..., ..., ...)
```

First, look at the values in the sample and the frequencies of those values. The function `factor(X)` tells R to treat `X` as a nominal variable, meaning just a collection of values without any numerical meaning (like `sex` above).
If we input this to the `summary()` function, we get frequencies of all the values.

```
> f = summary(factor(X))
> f
```

This is a frequency distribution table. It shows you the frequency `f(x)` for every value `x`. Notice that the total of
the frequencies equals the size of the sample.

```
> n = length(X)
> n
> sum(f)
```

`f` is very similar to a histogram. In fact we can make a bar plot of `f` to get a histogram.
```
> barplot(f)
```

Cumulative distributions

Pick any value in your sample.
```
> x = ___
```

To find the value of the cumulative distribution at `x`, we need to know how many scores are less than or equal to `x`. There are two ways to do this.

First, we can count the raw scores. Recall that this command:
```
> X <= x
```
will give a `TRUE/FALSE` vector showing which entries in `X` are less than or equal to `x`. If we take the sum of this vector, `sum()` will add up all the `TRUE`s, which tells us the total number of scores that are less than or equal to `x`.
```
> count = sum(X <= x)
```

The second solution is to use the frequency table, and add up the frequencies for all values less than or equal to `x`. To do this, you need to find which entry `x` is in the table. If it were the third entry (meaning `x` is the third smallest of the values that appear in the sample), then you would enter
```
> count = sum(f[1:3])
```

These two uses of `sum()` work in different ways. The first *counts* how many times `X` is less than or equal to `x`, whereas the second adds up the *frequencies* in `f` that correspond to values less than or equal to `x`. You should be able to see why both methods give the same result.

Either way you do it, you can then get the cumulative distribution by dividing the count by the sample size.
```
> count/n
```

To get the whole cumulative distribution function, use `ecdf()`, which stands for empirical cumulative distribution function.
```
> F = ecdf(X)
```

`ecdf()` is an unusual function because the result it gives, `F`, is a function itself. You can use the function `F` to evaluate the cumulative distribution at any value. Start with `x` to verify you get the same result as above. Then try some other values, including values inside, below, and above the raw distribution.
```
> F(x)
```

To see the whole function at once, plot it. Notice how it starts at zero, it jumps up at every value in the sample, and it ends at one.
```
> plot(F)
```

Loading data

If you have data stored as a text file, R will read it in and create a data frame. You can do this with internet files or files on your local machine. (The `header=TRUE` input tells the `read.table()` function that the first row of the data file shows the variable names.)

```
> data = read.table("Location of lab2-1.txt",header=TRUE)
Ex: > data = read.table("/Users/Sam/Desktop/lab2-1.txt",header=TRUE)
```

Recall that the first thing you should do when opening a new data file is to get a summary:

```
> summary(data)
   exam1           exam2             final      
Min.   :31.52 Min.   :50.65 Min.   :56.74
1st Qu.:62.78 1st Qu.:70.30 1st Qu.:68.44
Median :72.84 Median :81.81 Median :79.07
Mean   :71.01 Mean   :79.37 Mean   :78.25
3rd Qu.:77.10 3rd Qu.:91.27 3rd Qu.:88.59
Max.   :95.57 Max.   :98.10 Max.   :97.99
```

This shows that you have three variables, and it gives you their names and some descriptive statistics.

Assignment

1. Load the dataset called "lab2-2.txt" from the course website.

2. Display a summary of all the variables. Determine how many variables and how many subjects there are. Write how you interpret the output (e.g., "12 variables, 5 subjects").

3. Display the data for the last subject. Notice the values for this subject are senseless, meaning the data were input incorrectly and can't be used. Create a new data frame from your original one that excludes this subject. Use the new data frame for all subsequent questions.

4. Plot age against depression. Don't copy the plot into the submission window, but type a quick sentence about what pattern your plot shows.

5. Make a histogram of extraversion, with a reasonable number of bins. You should try a few different options and decide on which looks most useful. Don't copy your plot into the submission page; just your command(s).

6. Create a frequency table for years of education. Use the table to determine how many people have not graduated high school.