Session 4:
Plotting Data - Histograms

You've come a long way. So far, you've learned a lot about getting your data into R and how to describe it.
In this session, you'll learn the basics about visualizing your data by making simple graphs.

Arguably, plotting your data is more important than performing a statistical analysis. In part, this is because the way in which you analyze your data will depend partly on what it looks like when you plot it. Due to the importance of plotting data, we'll devote an entire future lecture to it in semester 2. Today you'll begin learn the 'nuts and bolts' of plotting data; this session focuses on histograms, and Session 5 focuses on scatterplots

Histograms

1. What is a histogram used for?

A histogram is a type of bar graph. The shape of the histogram (i.e., the shape made by the bars in the plot) gives you an indication of how your data is distributed and can tell you a lot. We'll address why the shape of a histogram (i.e. the distribution of your data) matters in lecture. But below, we will introduce histograms and distributions through an example

2. An example histogram

We'll use the hemoglobin data we've used in Sessions 2 and 3. Recall that these data describe hemoglobin levels measured from 1962 people in four regions of the world: the USA, the Andes, Tibet and Ethiopia. These data were collected to examine how hemoglobin levels compare between high and low elevations.

You may still have the 'hemo' object in your session. You can check this using ls(), or typing 'hemo'. If not, start a new RStudio session, and re-importing the dataset into an object called, 'hemo' (see Session 3 for more detail):

```
hemo <- read.table("hemoglobin.csv",header=TRUE,sep=',')
```

Now, look at hemo, and note the type of information that is present in each column.

It would be interesting to see how hemoglobin levels compare between regions. To simplify, we'll focus on two of the four of the populations: the USA (measured at sea level) and Andes populations, and make separate histograms for each population.

To plot a histogram, we'll use the function, **hist()**. Enter ?hist() into RStudio to read about this function, now.

Let's make a histogram of the hemoglobin data only from the USA. To do this, we first need to determine where the USA data lie in the dataframe. Because the data are nicely organized, we can do this by simply looking at the dataframe.
Do this now. You should see that the USA data lie on rows 1 to 1704. Now, how do we tell R to plot only these rows of hemoglobin data? Recall that in Session 3 we learned how to specify only certain rows and columns from a dataframe. In our current example, we wish to plot the hemoglobin measurements, but we only wish to plot data from rows 1 to 1704. The relevant hemoglobin data are in column 2, so we specify these with: hemo[1:1704,2] (recall the use of square brackets, and that we specify rows before columns within the []'s).

Hence, to make a histogram of only the USA data, we place this argument in the hist() function. Enter:

```
hist(hemo[1:1704,2])
```

Likewise, to plot the data from the Andes (rows 1705 to 1775), we enter:

```
hist(hemo[1705:1775,2])
```

A tidier way of processing and plotting the data, that makes the code easier to follow or 'debug', is first to extract (or "subset") the hemo data into a vector:

```
USAdata<- hemo[1:1704,2])
AndesData<-hemo[1705:1775,2]
```

Then plot the histograms by typing:

```
hist(USAdata)
hist(AndesData)
```

Note that, depending on what we wish to plot, we cannot always specify the appropriate data by simply selecting a set of neighbouring rows of data. Instead, we sometimes need to extract the data using a different method, which is explained in Box 4.1; we highly recommend that you read this box, because it also includes additional R tips.

3. What does a histogram tell us?

A histogram displays our data as a frequency distribution, and provides an excellent summary of the data by graphically representing which values were most or less common. The histogram's x-axis displays the values that were measured (in this case, hemoglobin levels), and the y-axis indicates how many data points had that specific value (or lay within a narrow range of hemoglobin values). From the USA histogram, we see that more than 300 individuals (see the y-axis) have hemoglobin levels of about 15 (the x-axis). In contrast, only a very few individuals have hemoglobin levels as low as 10, and very few, again, have hemoglobin levels greater than 18. As well, note that the 'tail' on the left of the USA distribution is relatively long, but the tail on the right-hand side of the histogram is shorter.

Overall, the distribution of hemoglogin levels of people living at sea-level in the USA is mildly asymmetrical, with the most common values in the distribution at about 15, and the lower and upper values at about 10 and 19, respectively.

Now, look at the histogram for the Andes data (i.e., hist(hemo[1705:1775,2]))

Note a few differences from the USA distribution: the most common values are higher (at about 19, instead of 15 seen for the USA), and the lowest and highest hemoglobin values, about 14 and 25, respectively, are also higher than those for the USA population. Finally, note that the Andes distribution appears to be slightly more symmetrical than that from the USA.

Now, determine the mean, minimum and maximum hemoglobin values from each of the USA and Andes datasets, and compare these values to what you see in the histograms.

Plotting and analyzing data in this fashion can prompt you to ask pertinent questions and formulate more detailed hypotheses to explain the findings. For instance: why do hemoglobin levels tend to be higher in the Andes compared to the USA population? Look at the original article (see the reference for Beall et al. (2002), at the end of Session 2).

4. Changing the width of the bars

As explained above, each bar in a histogram represents the number of observations (i.e., data points) that fall within a range of data on the x-axis. For example, if the bar covers the range of 14 to 16 on the x-axis, the height of the bar represents how many data points had hemoglobin levels within this range of 14 to 16. The range of each bar is referred to as a 'bin'. It follows, then, that changing the width of the bars (or the 'bin width') will also change their height. So, how can we change the width of the bars?

The function **hist()** allows users to specify bar widths in a variety of ways, but we'll

focus on one approach.

We'll use the option, 'breaks'. This option allows the user to specify the number of bars that will be used in the plot; naturally, if we change the number of bars, their widths will change, too. You may have noticed, from help(histo), that the default value of breaks is "Sturges". This refers to one type of calculation of an appropriate bar width, according to a formula devised originally in 1926, by H.A. Sturges. Specifically:

$k = \log_2(n)+1$

…that is: the optimum number of histogram intervals ('bins'), k, is determined from the logarithm (base 2) of the total number of observations, n.

Let's examine the effect of varying the value of breaks on the width and number of bars in the histogram of the USA data..

First, let's specify that we want the data to be divided up into 100 bars; in other words, we'll take the range of data along the x-axis and divide it into 100, identically sized bins. We do this by typing:

```
hist(hemo[1:1704,2],breaks=100)
```

How does this figure compare to the first figure?

Now compare this with the effect of specifying many fewer bars, such as 2:

```
hist(hemo[1:1704,2],breaks=2)
```

There is no "right number" of bars to display your data and different bin sizes can reveal different features of the data. For instance, we get a better sense of the distribution of the data when there are 50 or 100 bars, compared to 2. The point here is that playing with the number of bars can sometimes give you a better interpretation of your data.

Box 4.1: Alternative approach to plot sub-sets of a dataframe

In the main text, we created separate plots for USA and Andes hemoglobin data by specifying certain rows. However, sometimes the way the data are organized will not allow such a simple approach. In this case, we can use a different approach to extract and plot a subset of our data. For example, to create a dataframe with only USA data, type the following:

```
USA <- hemo[which(hemo$population=="USA"),]
```

What is this command doing? To understand it, we'll work from the inside out.

First, note the portion:

```
hemo$population=="USA"
```

Recall that the $ allows you to select a named column. This portion of the command compares the data in the column, 'population' of hemo, and asks whether it is the same as (i.e., equals) USA (see Box 4.2 for an explanation of the double-equals sign, ==). It returns TRUE if a row comes from the USA population, and FALSE, otherwise.

In the next step outwards, we use the function, **which()**. As you might guess, which() only returns data for which the value inside the brackets is TRUE. In our example, when a row of data has "USA" in the population column (i.e., when hemo$population=="USA" yields TRUE), which() will return these data.

Next, note the location of the which() statement. It occurs at the point marked 'HERE':

```
hemo[HERE,]
```

Why is the which() command located there? Recall two points. First, when we use the square brackets with a dataframe, as we do here, we access locations within the dataframe. Second, note that the which() statement lies in the 'rows' portion of the square brackets; i.e., it lies before the comma. Hence, overall, the above command causes rows from hemo with data from the USA to be written to a new object, 'USA'.

Now, let's do the same thing for the Andes:

```
Andes <- hemo[which(hemo$population=="Andes"),]
```

Look at the new dataframes, USA and Andes, to make sure they're what you expect.

BOX 4.2: == versus =.

The 'equals' sign can be used in two ways in R, as well as in other programming languages. When a single equals sign is used (i.e., =), it serves as a command that makes the left-hand side of the equation equal to the right-hand side. This is very useful when writing a program. On the other hand, sometimes we wish to compare two things and ask whether they are the same; in this case, we use the double equals sign (i.e., ==) to compare the items.

5. Exercise:

This Session includes only one exercise; you will gain additional practice in your practical workshop later this semester.

There is some concern that weather is becoming more hazardous. The file hurricanes_AllYears.csv contains data on the severity of hurricanes for the years 2001 to 2010; the column 'category' represents the severity of the storm. These data come from:

http://en.wikipedia.org/wiki/2010_Atlantic_hurricane_season and similar articles for other years, from the website:

http://whitlockschluter.zoology.ubc.ca/data/chapter02

Do the following:

a) Import the dataset

b) Using methods described above or in Session 3, make 2 separate histograms for hurricane 'category': one histogram for data from 2001-2005, and a second histogram for 2006-2010. How do these histograms compare?

   Determine the average hurricane category over the years 2001 – 2005. How does this compare to the average over 2006 – 2010?