Homework # 4:

Please circle your answers when appropriate! **As usual, 3rd edition references are in bold and in [ ].**
4th edition references are in italics, underlined and in {}.

1) 4.1, p. 133 **[4.1, p. 131.]** *{4.3.1, p. 131} But do the following instead:*

(a) within ± 0.15 standard deviation of the mean

(b) within ± 1.35 standard deviations of the mean

(c) within ± 2.25 standard deviations of the mean

(d) more than 3.25 standard deviations away from (above or below) the mean.

2) 4.5 p. 133 **[4.5, p. 131]** *{4.3.5, p. 131} But assume a mean of 89 lb, and a standard deviation of 6.25 lb.*

3) Not in 2nd edition. **[4.13 and 4.14, p. 133]** *{4.3.13 and 4.3.14, p. 132}***:** *Make sure to read the question as given here. It's been changed from what is in your text.*

4.13 The amount of growth, in a 15-day period, for a population of sunflower plants was found to follow a normal distribution with a mean of 3.48 cm and a standard deviation of 0.73 cm. What percentage of plants grow

(a) 3 cm or more? *(b) 4 cm or less?* *(c) between 2.75 and 3.75 cm?*

4.14 Refer to Exercise 4.13. *In what range do the middle 75% of all growth values lie?*

4) 4.13, p. 134 **[4.16, p. 133]** *{4.3.16, p. 132}. But do the following:*

a) What percentage of times were greater than 225 minutes

b) What is the 75th percentile of the times

c) no (c)

Now let's do some R...

(On next page)

5) We want to explore the effect of parameters on a binomial distribution.

Start up R or R-commander.

a) First we'll do our coin example from lecture. Let's calculate those probabilities and plot a graph so we know what our "distribution" looks like:

If you're using R from the command line, do the following:

`y <- c(0:10)`	we're giving y the numbers 0 through 10
`y`	if you want to see "y", you can do this step (not necessary)
`pr <- dbinom(y,10,.5)`	this gives us all the probabilities for the binomial. ".5" is the probability of success, "10" is the number of trials, and y is the number we're interested in (in this case, all the values from 0 to 10). Make sure you put in the numbers in the right order.
`pr`	this should give you all the probabilities.
`barplot(pr,names.arg = y)`	this generates a plot of "pr" (our probabilities) and tells it to use "y" to label our x axis.

Now answer the question at the end of this section.

If you're using R-commander, then do:

Distributions --> Discrete distributions --> Binomial distribution --> Plot binomial distribution.

Now enter 10 for "Binomial trials" and ".5" for "probability of success"

To get the actual probabilities you have to repeat this process:

Distributions --> Discrete distributions --> Binomial distribution --> Binomial probabilities

Enter the same information as before and the actual probabilities will be in the output window.

R and R commander: answer the question below:

(a) What does the graph look like? (It should be identical to the one some of you may have seen in lecture).

Make sure you present your graph as well. Instructions for cutting and pasting graphs were given in the previous homework assignment.

b) Let's try some different "parameters". Let's change .5 to .1 and see what happens (kind of like having a coin that comes up heads 90% of the time):

Repeat the above, but put use .1 for "probability of success" for both the graph and actual probabilities (for both the command line and R-commander, you essentially do the same thing you did, but change .5 to .1).

R and R commander: answer the question:

(b) What does the graph look like now? What changed? Why? Which probabilities are now higher? Why?

Again, make sure you present your graph.

6) Now let's try a normal distribution. The first plot will be easy, but the second will be more difficult to do, so follow the instructions carefully. First we'll plot a standard normal, then we'll see what happens if you change the parameters.

If you're using R from the command line:

```
x <- seq (-3.291, 3.291, length.out=100)
```

gives us 100 equally spaced values of x between -3.291 and 3.291. "seq" let's us generate a sequence of numbers, and "length.out" tells us how many numbers we want in the sequence.

```
plot(x, dnorm(x, mean = 0, sd = 1),
      col = "red", type = "l")
```

"plot" is a command to generate graphs (it's quite versatile, and we can't explain everything here). "dnorm" is essentially the equation for the normal distribution. This command will plot x and the value of x at each of the 100 points using the normal equation. The "mean", "sd" and "col" functions should be self-explanatory. type = "l" generates lines instead of points (try it without type = "l" to see the difference).

If you're using R-commander:

Distributions --> Continuous distributions --> Normal distribution --> Plot normal distribution.

You can just leave the defaults in place and plot it (i.e., just click "OK" when the box pops up asking you to enter the parameters).

R and R-commander:

At this point you should have a standard normal curve. Now we'll see what happens when we change the parameters. Unfortunately this can't be done easily from R-commander, but it will work if you carefully follow the instructions.

First, let's change the mean from 0 to 1:

If you're using R from the command line:

```
lines(x, dnorm(x, mean = 1, sd = 1), col = "blue")
```

"lines" will plot lines, but without erasing the previous plot. In other words, it'll plot the lines on the same graph as before. We changed the color to blue so you can see which plot is which.

If you're using R-commander:

The problem is that if you change the parameters and plot it again, R will simply adjust the scale on the graph and the graph you get will look exactly the same (try it! - but if you look at the x and y axes, you'll see that they have changed!). It'll also erase the old plot, which is not what we want.

So we need to plot our second graph on top of the first graph (don't close the graphics window).

In the script window, type:

```
x <- seq (-3.291, 3.291, length.out=100)
lines(x, dnorm(x, mean = 1, sd = 1), col = "blue")
```

highlight both lines and click the submit button. The R-code is explained above under using R from the command line.

R and R-commander: answer the following question:

(a) What happened to the normal distribution? How is it different from the first distribution you plotted?

You can hold off on copying and pasting your graph until you've done all of the steps below (you only need to hand in one graph for problem 6).

Now let's adjust the standard deviation:

If you're using R from the command line:

```
lines(x, dnorm(x, mean = 0, sd = 2), col = "green")
```

you should know how this works at this point.

If you're using R-commander:

Type (in the script window):

```
lines(x, dnorm(x, mean = 0, sd = 2), col = "green")
```

click on submit, etc.

R and R-commander: answer the following question:

(b) Again, what happened to the normal distribution? How is it different from the previous normal distributions you plotted?

Finally, let's adjust both the mean and standard deviation.

R and R-commander:

The code is below; you should know how to make it work by now:

lines(x, dnorm(x, mean = 1.5, sd = 1.7), col = "purple")

(c) How is this different from the above normal distributions?

At this point your graph should have four curves plotted on top of each other in your final graph. Copy this graph and paste it into a word processor or whatever you want to use so that you can print a copy.

(d) Can you understand now how important the parameters of a distribution are? What do the parameters of a distribution tell us about the distribution?

****

Copying and pasting graphs from R:

See instructions from the previous set of homework.

****

BIOL 214: Problems are due in recitation, Wednesday, June 19th.

BIOL 312: Problems are due at the beginning of lab, Thursday, June 20th.