Lab 2: Descriptive Statistics & Probability Distributions

ENVIRON 710: Applied Statistical Modeling[*]

The goals of this lab are to become familiar with discrete probability functions, specifically the binomial and Poisson distributions. The first part of the lab provides useful R functions, some of which we encountered in the last lab. The second part of the lab provides practice with exploratory statistics. The third part of the lab focuses on probability functions.

At the end of the lab, there are four problems to solve. Please show your code, type your answers to each problem, and plot graphs using R Markdown. *Submit your answers to the class Sakai site under the Assignments folder.*

More functions in R

In this lab, we introduce a few new R commands.

- `ls()` - provides a list of all the objects in the workspace
- `rm()` - removes individual objects from the workspace
- `is.na()` - logical function that identifies all NA's within a vector, returning either `TRUE` or `FALSE` for each value
- `dbinom()` - returns the height of the probability density function of the binomial distribution
- `pbinom()` - returns the cumulative density function of the binomial distribution; given a number, it computes the probability that a random number from a binomial distribution will be less than that number
- `rbinom()` - generates a random number from the binomial distribution defined by the probability (`prob`) of "successes" in a specified number (`size`) of trials
- `dpois()`, `ppois()`, `rpois()` - Same functions as above, but for the Poisson distribution defined by a single parameter, `lambda`, which specifies the mean and variance of the distribution

Download the database `AfrPlots.csv` from Sakai and read it into your workspace.

```
afdat <- read.csv("AfrPlots.csv", header = TRUE)
  head(afdat)
```

The database consists of tree plot data from Africa. In 30 1-ha plots, all the trees $\geq$ 10 cm dbh (diameter-at-breast height) were measured, and from these data the biomass, basal area, and other statistics were calculated for each plot. Note also that there were two census periods: the initial census and then a census four years later. To get started, let's calculate the mean basal area:

```
mean(afdat$BasalArea)
```

Sometimes you may want to remove variables from the workspace, particularly when you have created multiple variables and might confuse them. Variables can be removed one at a time using `rm()` as below. Or, multiple variables can be removed with `rm(list=ls(your objects here))`. You can remove *all* the objects in your workspace by leaving the list argument blank: `rm(list=ls())`.

[*]Created by John Poulsen with edits from TAs.

```r
(var0 <- seq(1:5))
 rm(var0)
```

In R, missing values are represented by the symbol NA (not available). Some functions, like `mean()` have arguments to remove NAs from the operation. Note what happens if you set `na.rm` to `TRUE`. `na.rm` is assumed to be FALSE if it is not explicitly set.

```r
var1 <- c(0, 4, NA, 2, NA, 7)

mean(var1)
mean(var1, na.rm = TRUE)
```

Other functions may not have built-in arguments to remove NAs, or you may have a different reason for removing NAs from a vector. To do this, employ the `is.na()` function. `is.na` is a logical function, meaning that it will return a TRUE or FALSE response for each value in the vector.

```r
is.na(var1)
```

To actually remove the NAs, define `var2` as all the values of `var1` where `is.na` is not (`!`) equal to TRUE. Note that TRUE and FALSE can be abbreviated as T and F. Here these two lines do the same thing.

```r
var2 <- var1[!is.na(var1)]

var2a <- var1[is.na(var1) == FALSE]
```

To code an observation as missing, you can simply assign it NA. For example, let's replace the value of 7 in `var2` with NA. The double equal sign (==) is a logical operator that means *exactly equal to*. This line of code says replace all values in `var2` that are exactly equal to 7 with `NA`.

```r
var2[var2 == 7] <- NA
```

Exploratory data analysis in R

Skewness and kurtosis

We have talked about measures of location and spread like the mean and standard deviation. By contrast, skewness and kurtosis are measures of shape. A histogram can give you the general idea of the shape of a distribution of data, but skewness and kurtosis give more precise, numerical evaluations. Why do we care? Many classical statistical tests and models depend on assumptions of a normal distribution (a bell-shaped curve), with skewness and *excess* kurtosis of 0. Significant skewness and kurtosis indicate that data are not normally distributed. Let's examine the skewness and kurtosis of the distribution of numbers of trees in the 30 field plots.

To Do

Create a boxplot of the `Trees` (number of trees per plot) data from `afdat`.

```r
dukeblue <- "#00009C"

ggplot2::ggplot(afdat, aes(x=Trees)) +
          geom_histogram(fill = dukeblue, colour = "white", bins = 10) +
            xlab(expression(paste("No. of Trees ", ha^-1))) +
          ylab("Count") + theme_bw()
```

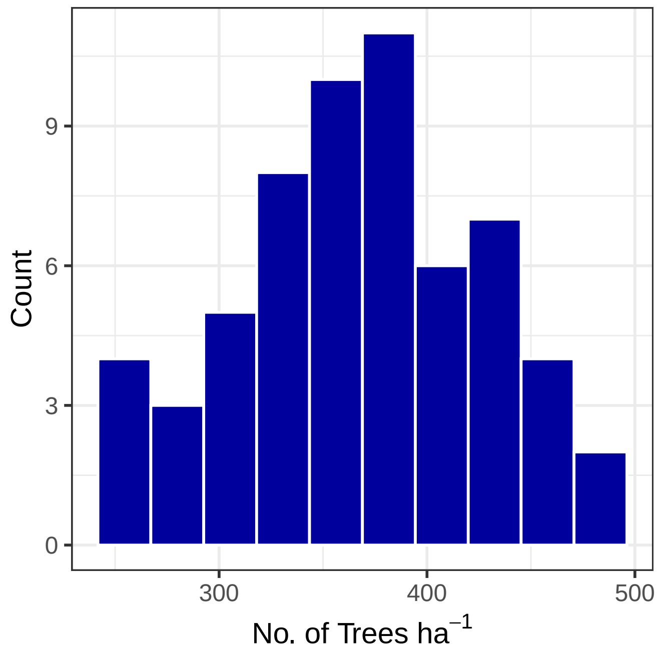

Figure 1: Distribution of number of trees in 1-ha plots in Africa.

Let's look at this histogram of the data. I did a few things here to make it pretty. First, I defined a color called *dukeblue*. I then filled the bars of the histogram with it using the `fill` argument. Don't like the choice of color? See *Rcolornames* in R Shortcuts on Sakai that gives a list of all the possible colors. Second, I specified the color white to outline the bars. Third, R automatically puts the data into 30 bins; I changed it to 10 bins to better represent the data. I also added x- and y-axis labels and specified `theme_bw()`, which I think is prettier than the standard grey background. Remove `+ theme_bw()` and run the code to see what it looks like with the standard theme.

Skewness is a measure of the extent to which a probability distribution of a random variable leans to one side of the mean. If a distribution has a negative skew (left-skewed or left-tailed), the left tail is longer and the mass of the distribution is concentrated to the right. As a rule, the mean of the data is less than the median for a left-skewed distribution. For positive skew or right-skewed distribution, it would be just the opposite. Skewness of 0 means the distribution is symmetrical around the mean and the mean is equal to the median.

Skewness can be calculated as:

$$g_1 = \frac{1}{ns^3}\sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^3$$

where Y_i is the ith data point, $\overline{Y}$ is the mean, s is the sample standard deviation, and n is the number of data points.

Rather than write the equation every time we want to calculate skewness, it can be written as a function in R.

```
myskewness <-  function(y) {
 n <- length(y)
  skew <- 1/(n*sd(y)^3)*sum((y-mean(y))^3)
```

```
    skew
}
```

Note the features of a function include the command `function()` that defines the object as a function. The function is then specified within the curly brackets and must correctly perform a task and return any results, which may be stored in other objects like vectors or data frames. To call this function to find the skewness of the variable `Trees`, type:

```
myskewness(afdat$Trees)
```

Are `Trees` positively or negatively skewed?

To Do

Calculate the skewness of the variable `BasalArea`.

Kurtosis is a measure of the flatness or peakedness of the probability distribution of a random variable. Distributions with negative kurtosis are called platykurtic, whereas distributions with positive kurtosis are called leptokurtic. See the following general rules.

- Kurtosis = 3: a normal distribution has kurtosis of exactly 3 (excess kurtosis = 0). Any distribution with kurtosis of approximately 3 is called mesokurtic.
- Kurtosis > 3 (excess kurtosis > 0): leptokurtic distribution, sharper than a normal distribution, with values concentrated around the mean and thicker tails. This means there is a high probability for extreme values.
- Kurtosis < 3 (excess kurtosis < 0): platykurtic distribution, flatter than a normal distribution with a wider peak. The probability for extreme values is less than for a normal distribution, and the values are widely spread around the mean.

Excess kurtosis can be calculated as:

$$g_2 = \left[\frac{1}{ns^4}\sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^4\right] - 3$$

with the variables having the same meanings as above. The reference standard is a normal distribution, which has a kurtosis of 3. In token of this, excess kurtosis is often calculated as here by subtracting 3 from the measurement.

In future labs we will estimate whether skewness or kurtosis is significantly different from a normal distribution - that is, their values are too high to be explained as random variation from a normal distribution.

Measures of location and spread

We have talked about measures of location and spread in lecture. There are several measures of Central Tendency, including the median, mean, trimmed mean, harmonic mean, and geometric mean. Similarly, there are several measures of spread, including variance, standard deviation, and coefficient of variation.

The variance of a sample of data can be calculated as:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^2$$

where s^2 is the sample variance, n is the sample size, $\overline{Y}$ is the sample mean and Y_i are the individual values of Y. The standard deviation is the square root of the variance. And, the coefficient of variation is the ratio of the standard deviation to the mean, reported as a percentage.

To Do

Use R's functions (`var`, `sd`, `mean`) to find the variance, standard deviation, and coefficient of variation of `BasalArea`.

Discrete probability in R

To explore discrete probability distributions in R, we first need to generate some random numbers. It is, in fact, almost impossible to generate random numbers on a computer, so we actually use a pseudo-random sample. To reproduce the results, we need to set the seed for our pseudo-random number generator to a value (let's use 1001) using the `set.seed()` function. This sets the number generator to always start at 1001 so that you will get reproducible results. For example, let's randomly choose 10 numbers from 1 to 100.

```
set.seed(1001)
 sample(x = 1:100, size = 10)
```

Run both lines of code multiple times. You should get the same sequence of ten numbers each time. Pseudo random number generators are not really random, but algorithmic: given the same seed, you get the same algorithmically-generated pseudo-random sequence. Again, this is driven by the desire for reproducible results. If you change the seed, or if you don't set the seed before sampling, you will get different results. Now try it again, but generate a sequence of 100 pseudo-random numbers instead of just ten. The initial ten numbers are the same as the first set, given you set the same seed.

```
set.seed(1001)
 sample(x = 1:100, size = 100)
```

Binomial distribution

Remember from lecture that a Bernoulli trial is an experiment with two possible outcomes, like flipping a coin or recording whether a species is present or absent. The outcome is referred to as a success or failure. Most often in environmental sciences, we are interested in what happens over a sequence of Bernoulli trials. For our random variable, X, we will count the number of successes, such as the number of times out of 10 that we flip "heads".

To simulate a Bernoulli trial, we can write:

```
set.seed(1001)
 dbinom(1, size=1, prob=0.5)
```

Here `dbinom()` is the density of the binomial distribution. For discrete data (not continuous data), it returns the probability of a specific value of X. How is this different from the results we obtained using `rbinom()` in the previous lab?

Because the values of the binomial distribution are discrete, the probability mass function is discrete, not continuous like a normal distribution. Therefore, we visualize it with a sequence of spikes. Let's represent a

Bernoulli distribution where the probability of 0 is 0.3 and the probability of 1 is 0.7:

```
xp <- data.frame(cbind(x = c(0, 1), p=c(0.3, 0.7)))

ggplot(xp, aes(x=x, y=p)) + geom_bar(stat="identity", width = 0.005) +
       geom_point(colour = "red", size = 2) +
        ylab("Probability") + xlab("X") + xlim(-0.5, 1.5) +
        theme_bw()
```

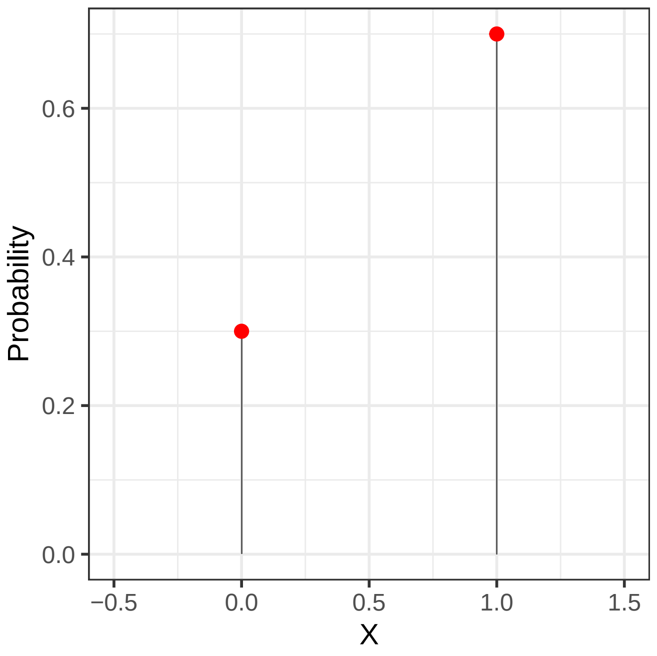

Figure 2: Bernoulli distribution where P(X=1) = 0.7.

Now, let's toss a 'fair' coin ten times. What is the probability of obtaining 3 heads? What is the probability of obtaining 3 or fewer heads? First, let's visualize the distribution. Here `n` is the number of flips; `p` is the probability of getting a heads, `x` is all possible outcomes (number of heads), and `pr` determines the probability of getting 0 to 10 heads out of 10 flips of a coin.

```
 n <- 10
 p <- 0.5
 x <- 0:10
 pr <- dbinom(x, size = n, prob = p)

 xp <- data.frame(cbind(x, pr))

 ggplot(xp, aes(x=x, y=pr)) + geom_bar(stat="identity", width = 0.05) +
       geom_point(colour = "red", size = 2) +
        ylab("Probability") + xlab("X, number of heads") +
         scale_x_continuous(breaks = c(seq(0, 10, by = 2))) +
        theme_bw()
```

To find the probability of a single value, $P(X = 3) = 0.12$, input one value of x into `dbinom()`:

```
dbinom(x = 3, size = 10, prob = 0.5)
```

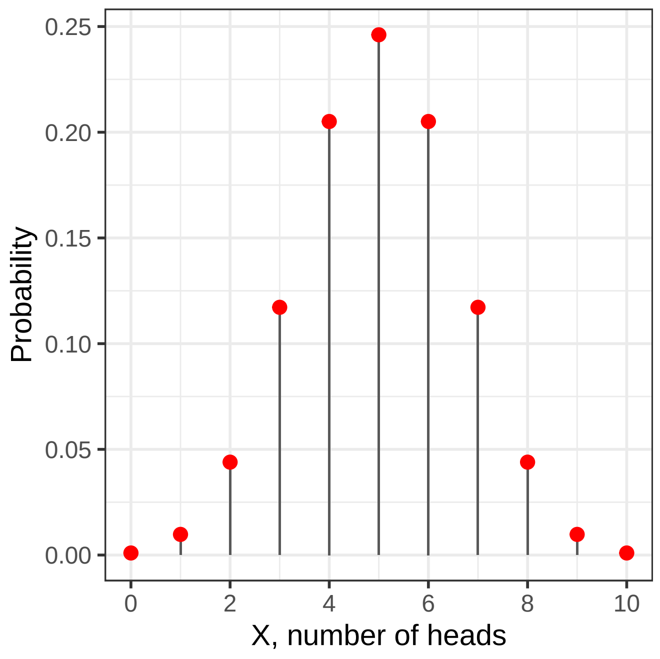

Figure 3: Bernoulli distribution of the probability of flipping 0 to 10 heads in 10 flips of a fair coin.

```
## [1] 0.1171875
```

To find the probability of obtaining 3 or fewer heads, $P(X \leq 3)$, one approach is to realize that the probability of obtaining 3 or fewer heads is the sum of the individual probabilities. Add up the probabilities of obtaining 0, 1, 2, or 3, heads.

```r
sum(dbinom(0:3, size = 10, prob = 0.5))
```

So what is the `dbinom()` function doing? It is calculating the binomial probability, using the equation:

$$P(X) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$$

.

To Do

Write the equation for binomial probability in R. Use it to calculate the probability of obtaining 3 or fewer heads in ten flips of a fair coin. How does your answer compare to the probability found using `dbinom()`?

We can also use *cumulative probability* to find this probability. Cumulative probability is the measure of the chance that two or more events happen. We can make use of the cumulative probability function for the binomial distribution, `pbinom()`, which provides the sum of the probabilities of all values "less than or equal" to the specified value.

```r
pbinom(q = 3, size = 10, prob = 0.5)
```

Poisson probabilities

Like the binomial distribution, the Poisson distribution is a discrete probability function. However, instead of modeling successes and failures, it models counts of outcomes. If lambda, λ, is the mean occurrence of a count, then the probability of having x occurrences is given as:

$$P(x) = \frac{\lambda^x}{x!}e^{-\lambda}$$

where $x = 0, 1, 2, 3, ...$

Instead of using `dbinom()` to find the probability of an occurrence, we replace it with `dpois()` for the Poisson distribution. For example, if the mean number of lightening strikes on top of Mt. Baldy is 3 per year, then the probability of the mountain only being struck once in 2014 is found by:

```
 dpois(x = 1, lambda = 3)
```

Like above, if we wanted to know the probability of witnessing more than 5 strikes in a year, we could use the cumulative probability function, `ppois()`. Note the argument `lower.tail = FALSE`, this tells `ppois()` that we want the probability on the right hand side of the distribution - the probability greater than 5.

```
ppois(q = 5, lambda = 3, lower.tail = FALSE)
```

```
## [1] 0.08391794
```

What is the probability of the mountain being struck 3 times? 10 times? $<$ 2 times?

Normal probabilities

Unlike the binomial and Poisson, the normal distribution is a continuous probability function. It can be used to model numbers that can take on infinitely many, uncountable values. The probability *density* function is given as:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where x is any value of a continuous random variable, σ is the standard deviation and μ is the population mean.

Plugging in a value of x and the mean and standard deviation, we can calculate a height f(x) of the density function. For example, we could use `dnorm()` similar to `dbinom()` and `dpois()` before. Here, we calculate the density at -1 for a standard normal distribution, $X \sim N(0, 1)$:

```
  dnorm(x = -1, mean = 0, sd = 1)
```

```
x <- seq(-3, 3, length = 1000)
 y <- dnorm(x, mean = 0, sd = 1)
xy <- data.frame(cbind(x,y))

 ggplot(xy, aes(x=x, y=y)) + geom_line(colour = dukeblue) +
         ylab("Probability density") + xlab("X") +
          geom_segment(aes(x = -3, y = dnorm(-1, 0, 1), xend = -1,
                       yend = dnorm(-1, 0, 1)), linetype = "dashed") +
          geom_segment(aes(x = -1, y = 0, xend = -1,
                       yend = dnorm(-1, 0, 1))) +
        theme_bw()
```

Again, this is the density or height of the curve at that point, not its probability. We could estimate the probability of getting a -1 by finding the area under the curve between -0.999 and -1.001:

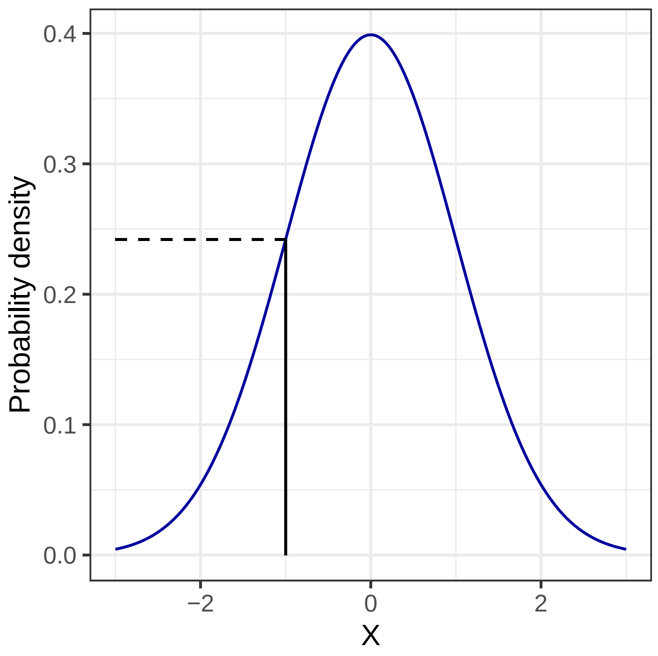

Figure 4: Density of a Normal curve at an x-value of -1.

```r
dnorm(x = -0.99999, mean = 0, sd = 1) - dnorm(x = -1.00001, mean = 0, sd = 1)
```

This probability is very small, and would get smaller and smaller if we tried to estimate it with more precision. On continuous distributions, we find probabilities for area under the curve, not for individual numbers.

But that is okay because we usually want to calculate bigger areas under the Normal curve. For example, we ask questions like ‘what is the probability of getting a value small than -1’. This can be determined by using the *cumulative probability* function `pnorm()`.

```r
pnorm(q = -1, mean = 0, sd = 1)
```

This shaded area represents the cumulative probability between -4 and -1 for the distribution.

```r
x1 <- seq(-4, 4, length = 1000)
xy.dat <- data.frame(cbind(x = seq(-4, 4, length = 1000),
                           y = dnorm(x1, mean = 0, sd = 1)))

 q <- ggplot(data = xy.dat, aes(x, y)) +
       geom_line(stat = 'identity', col = dukeblue) +
        ylab("Probability density") + xlab("X") +
         xlim(-4, 4) +
          annotate("text", x = 3, y = 0.38,
                   label = TeX("$X \\sim N(0, 1)")) +
         theme_bw()

q + stat_function(fun = dnorm, xlim = c(-4, -1), geom = "area",
                    fill = "#001A57", alpha = 0.7) +
                   annotate("text", x = -3, y = 0.12,
                           label = TeX("P(X$\\leq -1) = "), size = 3) +
                   annotate("text", x = -3, y = 0.1,
                  label = "pnorm(q = -1,", size = 3) +
         annotate("text", x = -3, y = 0.08,
                  label = " mean = 0, sd = 1)", size = 3)
```

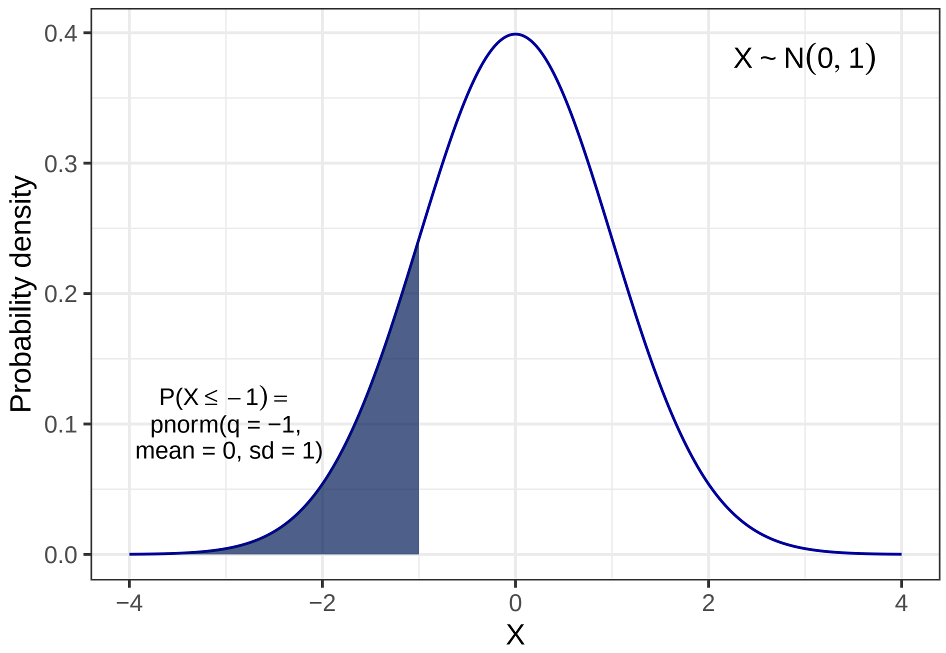

Figure 5: Normal distribution with shading demonstrating that the probability of getting a number less than -1 is 15.9%.

Problems

Problem 1

Summarize the data for mean growth `MeanGr` of the plots in `afdat`. Mean growth is the difference in average DBH in cm for each plot between the first tree census and the second census; thus, there are only data for `MeanGr` during the second census. Include the following in your analysis:

a. Plot both a histogram and a boxplot of `MeanGr`. Make sure to correctly label the axes.
b. Write a function for excess kurtosis (using the equation above) and calculate the kurtosis of the distribution of mean growth data.
c. Report the mean, median, standard deviation and coefficient of variation of the `MeanGr`.

Problem 2

Say you flip a fair coin 20 times. What is the probability of obtaining more than 5 heads: P(X > 5)? Show how you would answer this question using (a) `dbinom()` or `pbinom()` and (b) calculating it with your binomial equation from above.

Problem 3

Suppose there are 20 multiple-choice questions on a Stats quiz. Each question has four possible answers, only one of which is correct. Find the probability of answering 17 or more answers correctly if you attempt to answer them at random.

Problem 4

If there are 4 butterflies feeding at a flower per hour on average, find the probability of having 9 or more butterflies feeding at the flower in a particular hour. Although the possible maximum count of butterflies could be much greater, limit the maximum number of butterflies to 13. (a) Write out the Poisson formula in R for $P(X = 9)$; and, (b) calculate the probability, $P(X \geq 9)$, using `dpois()` or `ppois()`.