Lab 4 - Analyzing Proportions

Contents

Lab Objectives

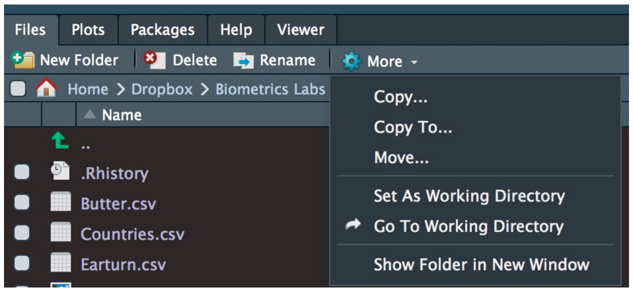
- Use R to conduct binomial testing
- Use R to compute a complete binomial distribution
- Use R to calculate confidence intervals for proportions using the Agresti-Coull method
- Practice using R

Set your working directory

Before getting started, set your working directory to where the course files are located. The code below is commented out, since you will have to provide your own working directory.

```r
# setwd('Path to your files') # i.e.
# C:/Users/YourName/Documents/Biometrics Labs/ note: this
# depends on your computer's OS
```

In RStudio, you can also change your working directory using the graphical interface:



Exercise 1: Binomial testing in R

The binomial distribution

The binomial distribution describes the probability of a given number of successes out of n trials, where each trial independently has a p probability of success. It is given by the following equation:

$$Pr\left(X\right) = \binom{n}{X} p^{X} \left(1-p\right)^{n-X}$$

X = total number of *successes* (pass or fail, heads or tails etc.)

p = probability of a success on an individual trial

n = number of trials

The binomial test compares the observed number of successes in a data set to that expected under a null hypothesis. Today we will learn how to conduct this test in R.

Ear turn

Do people typically use a particular ear preferentially when listening to strangers? To answer this question, a researcher approached and spoke to strangers in a noisy nightclub, and an observer recorded whether the person turned their left or right ear toward the researcher. This data is found in the file Earturn.csv.

Then, load the file Earturn.csv into a variable called 'd', and examine it's data structure using the **structure()** function and the **head()** or **tail()** functions to get a quick glance at the data. Head shows the top 6 rows, tails displays the bottom 6 rows:

```
d <- read.csv("Earturn.csv")
str(d)
```

```
## 'data.frame':    25 obs. of  1 variable:
##  $ Ear: Factor w/ 2 levels "left","right": 2 2 2 2 2 2 2 2 2 2 ...
```

```
head(d)
```

```
##     Ear
## 1 right
## 2 right
## 3 right
## 4 right
## 5 right
## 6 right
```

```r
tail(d)
```

```
##     Ear
## 20 left
## 21 left
## 22 left
## 23 left
## 24 left
## 25 left
```

You should see that d is loaded as a data.frame, which is R terminology for a means of storing data in a table like format. In this case, the data appear to be 25 observations with only 1 variable. In other words, there are 25 rows and 1 column of information.

What the output also notes is that the data.frame contains 1 variable that called "Ear", which is listed a Factor w/ 2 levels: "left", "right". This means, then that the file contains row-based entries on each measurement, whether it is right or left. This is a long form way of encoding data and will need to be summarised before the binomial tests can be applied.

Perform an Exact Binomial Test, using the **binom.test()** function. This function performs an exact test of a simple null hypothesis about the probability of success in a Bernoulli experiment.

However, the function requires that it be provided with a "vector of length 2, giving the numbers of successes and failures".

We will use the **xtabs()** function to perform a cross tabulation of the data and count up how many right and left turns there are. The tilde (~) is often used to specify groups or special formulae in R. Putting the results of the **xtabs()** function into the new variable success.fail helps to follow the example:

```r
xtabs(~d$Ear)
```

```
## d$Ear
##  left right
##     6    19
```

```r
success.fail <- xtabs(~d$Ear)
success.fail
```

```
## d$Ear
##  left right
##     6    19
```

```r
binom.test(success.fail, p = 0.5, conf.level = 0.95)
```

```
##
##  Exact binomial test
##
## data:  success.fail
## number of successes = 6, number of trials = 25, p-value = 0.01463
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.09356444 0.45128802
## sample estimates:
## probability of success
##                   0.24
```

There are often multiple ways to do things in R. You can manually specify x and n and input these as arguments in the function, binom.test:

```r
X <- 6
n <- 19 + 6
binom.test(X, n, p = 0.5, conf.level = 0.95)
```

```
##
##  Exact binomial test
##
## data:  X and n
## number of successes = 6, number of trials = 25, p-value = 0.01463
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.09356444 0.45128802
## sample estimates:
## probability of success
##                   0.24
```

If you want to simply have the p value returned, create a result variable and then call p value from the result, but using the dollar ($) sign in R, which is used typically to specify a named element (p.value) that is stored in your variable of interest.

```r
result <- binom.test(X, n, p = 0.5, conf.level = 0.95)
result$p.value
```

```
## [1] 0.0146333
```

Return to the data and change one right ear to a left ear. Re-run the Binomial test. Is the difference still significant? If so, return to the data and change another right ear to a left ear and re-run the binomial test until you find the point at which the proportion is no longer significantly different from the null hypothesis.

Note: the that Whitlock and Schluter advise that the p value returned from R's binom.test is not the same as one determined empirically, and often the solution is to multiply the number by 2. (see. p 196 Whitlock and Schluter re: p values and binomial tests.)

```r
success.fail <- c(7, 18)  # 6+1 and 19-1
result <- binom.test(success.fail, p = 0.5, conf.level = 0.95)
result$p.value * 2
```

```
## [1] 0.0865705
```

Note that you can also change the test proportion according to the null hypothesis. The null hypothesis for the ear turned is that there is no difference between left and right ears, so 0.5 is appropriate. However, this may change for other tests.

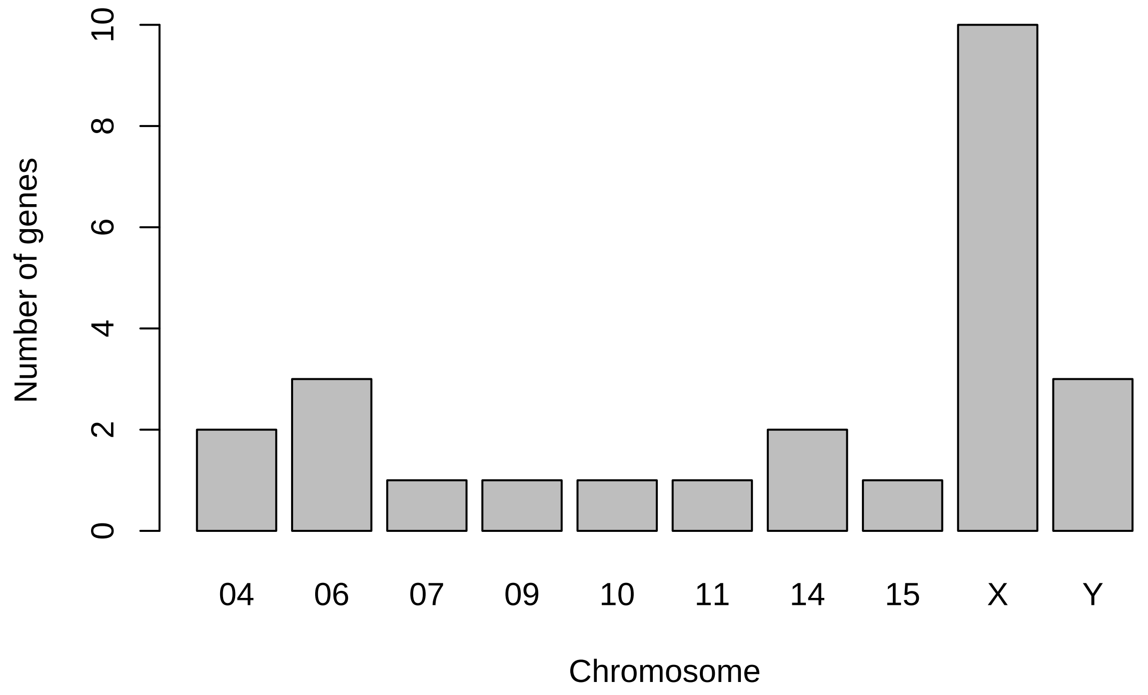
Answer question 1 on Sakai

Spermatogenesis genes

A study of genes involved in spermatogenesis was carried out to test the hypothesis that spermatogenesis genes should occur disproportionately often on the X chromosome, which contains 6.1% of the total genes. The researchers found that 40% of the analyzed spermatogenesis genes were on the X chromosome. This data is found in the file Spermatogenesis.csv.

Read in the data, summarise the number of genes per chromosome using the **xtabs()** function or the **table()** function and plot the results using **barplot()**.

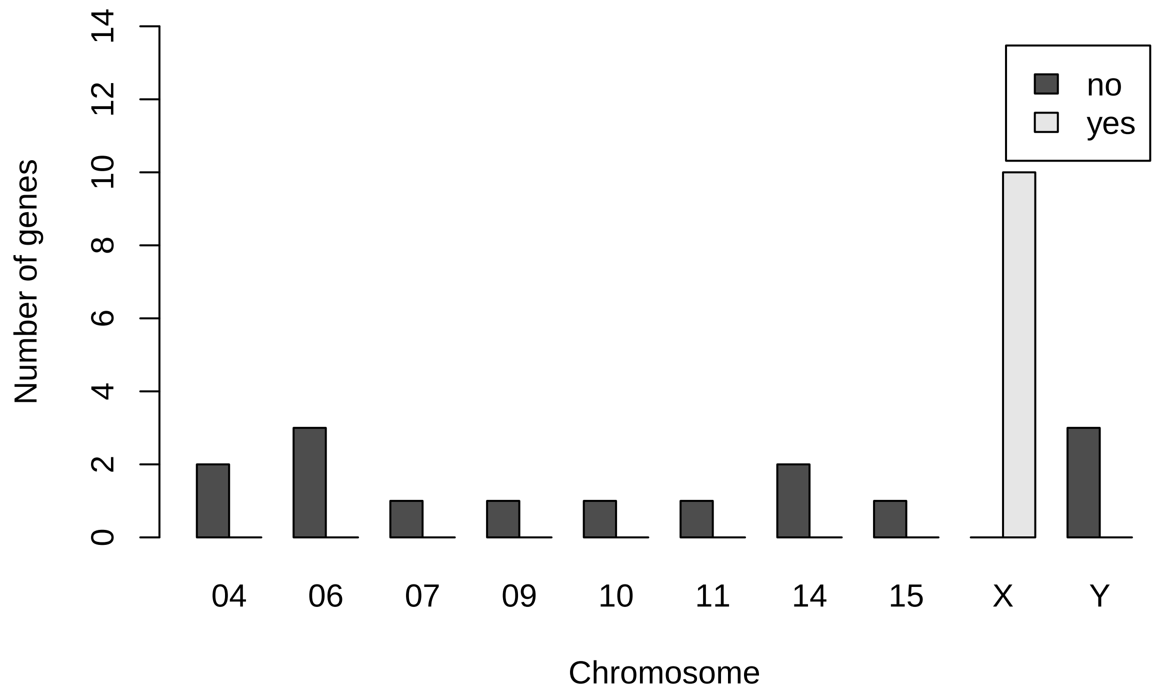
```
d <- read.csv("Spermatogenesis.csv")
counts <- xtabs(~d$chromosome)
barplot(counts, beside = T, xlab = "Chromosome", ylab = "Number of genes")
```



```
counts <- table(d$onX, d$chromosome)
counts
```

```
##
##       04 06 07 09 10 11 14 15  X  Y
##   no   2  3  1  1  1  1  2  1  0  3
##   yes  0  0  0  0  0  0  0  0 10  0
```

```
barplot(counts, beside = T, xlab = "Chromosome", ylab = "Number of genes",
    legend = rownames(counts), ylim = c(0, 14))
```



The barplot function can handle data that are grouped. In this case, by the onX variable, which yields 'no' or 'yes' as potential categories. Note: this is a strange way to plot your data, as the grouping variable is redundant.

Now, peform a binomial test of the frequency of genes on the X chromosome relative to the null expectation:

```
X = 10
n = sum(counts)
p = 0.061
p.value <- binom.test(X, n, p)$p.value
p.value  # one-tailed p value
```

```
## [1] 9.93988e-07
```

```
p.value * 2  # two-tailed equivalent
```

```
## [1] 1.987976e-06
```

Now imagine we are interested in the proportion of spermatogenesis genes on the Y chromosome. The Y chromosome contains only 1.2% of the total genes in the mouse genome. Make use of the **sum()** command and a boolean expression using the == command to count the observed number of genes on the Y, and compare to that expected under the null hypothesis.

```
d$chromosome == "Y"  # this should return a series of TRUE and FALSE values
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [23]  TRUE  TRUE  TRUE
```

```
sum(d$chromosome == "Y")  # sum up the number of TRUE statements
```

```
## [1] 3
```

```
X = sum(d$chromosome == "Y")
n = sum(counts)
# Now run the appropriate binomial test to answer the
# question
```

Make note of the observed proportion of genes on the Y chromosome and the P-value for a two-tailed test.

Answer question 2 on Sakai

Groundhog Prognostication

According to http://www.stormfax.com/ghogday.htm, the groundhog's ability to predict spring is about 39% accurate, even though the actual spring data were not provided in the link.

```
d <- read.csv("GroundhogDay.csv")
str(d)
```

```
## 'data.frame':    130 obs. of  2 variables:
##  $ Year  : int  1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 ...
##  $ Result: Factor w/ 5 levels "No Appearance by Groundhog",..: 5 5 2 3 2 2 2 2 2 2 ...
```

```
counts <- table(d$Result)
sum(counts)
```

```
## [1] 130
```

From the table above, there are 10 records with no data and 130 records in total thus n = 130-10 = 120, and the success rate is 39% or 0.39, as reported on the website above.

```
n = 130 - 10
X = 0.39 * n
```

Note this yields a fraction value for X, not a whole number, so we are not working with the actual data, but the binomial test below will return an error if we do not provide whole numbers, so we will have to round the x off to the nearest integer before running the binomial test, to test whether the groundhog's predictions differ from a 50% random chance prediction:

```
X <- round(0.39 * n)
X
```

```
## [1] 47
```

```
binom.test(X, n, p = 0.5, alternative = "two.sided")$p.value
```

```
## [1] 0.02208272
```

Punxsutawney Phil's prediction rate falls significantly below that of random chance!!

However, see a more detailed mathematical and scientific analysis here that suggests the groundhog has become a much more savvy groundhog, acheiving ~70% accuracy between 1955-1999:

https://www.jstor.org/stable/pdf/2687216.pdf?refreqid=excelsior%3Af9de80994249fe0e5103a0399b8b1e38

Lost Wallets

In a study in Scotland, researchers left a total of 240 wallets around Edinburgh, as though the wallets were lost. Each contained contact information including an address. Data on the number of wallets that were returned is found in the file Wallets.csv.

Load the file in using code derived from the examples above.

Answer question 3 and 4 on Sakai

Exercise 2: Computing a binomial distribution in R

Choosing wines

One function of R that we have not yet examined is its use to compute various statistical values as new variables. We previously examined an example in which 15 participants chose between two types of wine (cheap and expensive). If there was no preference between the two types of wine, then we would expect the proportion of individuals preferring each type of wine to be approximately 0.5 (50%).

The binomial distribution provides the probability distribution for the number of successes in a fixed number of trials, when the probability of success is the same in each trial. Applied to our example, the binomial distribution can tell us the probability of having a particular number of participants (e.g. 4, 8, 14) prefer the expensive wine over the cheap wine.

We will use the **dbinom()** function to return a result of a binomial distribution, wherein we can obtain the probability of obtaining a particular number of 'successes' (i.e., 4, 8, or 14).

```
n = 15
p = 0.5
```

Probability of 4 people

```
NumPick <- 4
dbinom(NumPick, size = n, prob = p)
```

```
## [1] 0.04165649
```

Probability of 8 people

```
NumPick <- 8
dbinom(NumPick, size = n, prob = p)
```

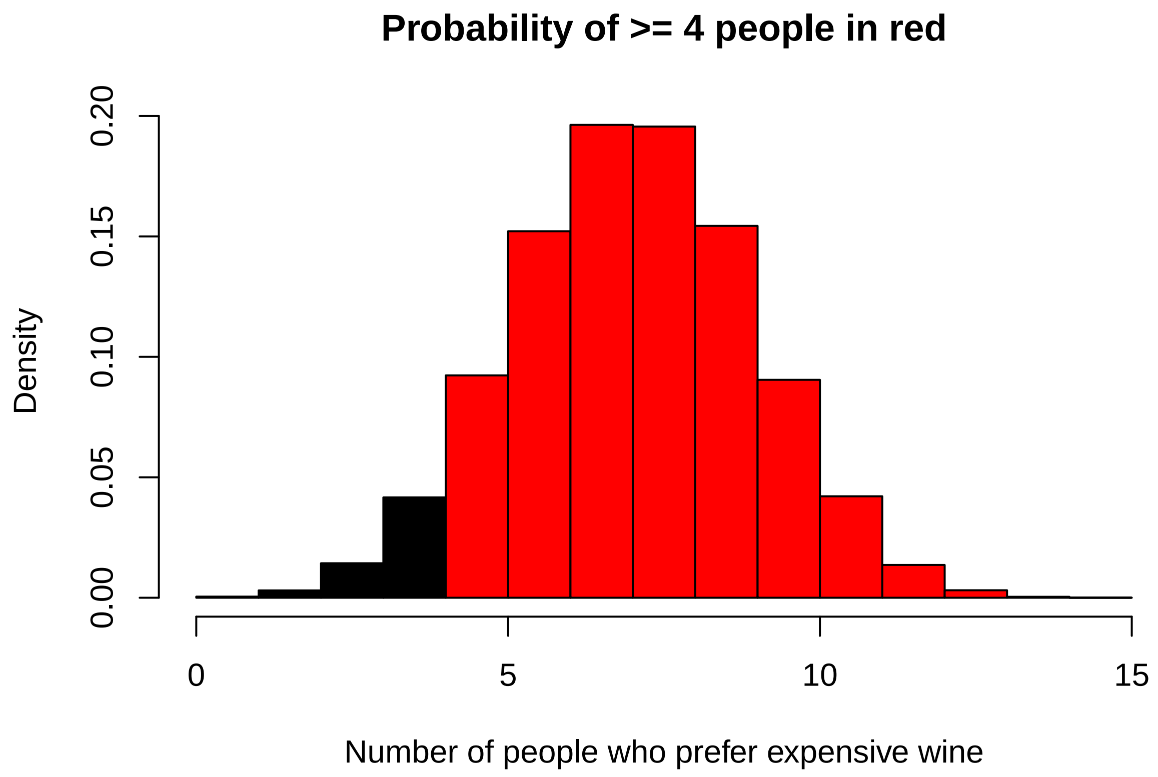
```
## [1] 0.1963806
```

Probability of 14 people

```
NumPick <- 14
dbinom(NumPick, size = n, prob = p)
```

```
## [1] 0.0004577637
```

Or, if you were interested in a range of probabilities, i.e. number >= 4:



You would need to sum across the densities obtain for each of those values. In this case, summing the density from 4 to 15 will encompass all possibilities, since you cannot have more than 15 out of 15:

```
probs <- dbinom(4:15, size = n, prob = p)
sum(probs)
```

```
## [1] 0.9824219
```

Answer question 5 on sakai

Heterozygous crosses

In a cross between two heterozygotes, 25% of the offspring is expected to be homozygous recessive. In pea plants, green pod colour is dominant, and yellow pod colour is recessive. In a cross between two heterozygous pea plants you obtain 20 offspring.

Compute the binomial distribution for obtaining 10 yellow plants out of n = 20.

```
dbinom(x = 10, size = 20, p = 0.25)
```

```
## [1] 0.009922275
```

Answer question 6 on sakai

Exercise 3: 95% Confidence intervals for proportions

To practice the Agresti-Coull method for calculating the 95% confidence interval of a proportion, we will need the following equations:

$$p' = \frac{X+2}{n+4}$$

$$p' - 1.96\sqrt{\frac{p'(1-p')}{n+4}} < p < p' + 1.96\sqrt{\frac{p'(1-p')}{n+4}}$$

Murphy's law

In a test of Murphy's Law, pieces of toast were buttered on one side and then dropped. Murphy's Law predicts that they will land butter-side down. Out of 9821 slices of toast dropped, 6101 landed butter-side down.

Open the file Butter.csv, which contains the data from the Murphy 's Law experiment described above.

```
d <- read.csv("Butter.csv")
str(d)
```

```
## 'data.frame':    9821 obs. of  1 variable:
##  $ sideDown: Factor w/ 2 levels "butter side",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
levels(d$sideDown)
```

```
## [1] "butter side" "dry side"
```

```
sum(d$sideDown == "butter side")
```

```
## [1] 6101
```

```
X <- 6101
n <- 9821
pp <- (X + 2)/(n + 4)
Conf.Range <- 1.96 * sqrt(pp * (1 - pp)/(n + 4))
```

Compute the confidence intervals and perform a binomial test on the data.

Answer questions 7 and 9 on sakai

Surgical infections

Out of 67410 surgeries tracked in a study in the UK, 2832 were followed by surgical site infections.

Calculate the 95% CI for the proportion of surgical site infections.

If a 95% confidence interval does not include the value provided in the null hypothesis, then the null hypothesis will usually be rejected. This makes sense, as it indicates that the value of the null hypothesis is not one of the most plausible values given the data.

```
X <- 2832
n <- 67410
pp <- (X + 2)/(n + 4)
Conf.Range <- 1.96 * sqrt(pp * (1 - pp)/(n + 4))
```

Answer question 8 on sakai

Exercise 4: R Practice

Country Data

A variety of population statistics collected from 209 different countries is found in the file Countries.sav.

Open the file Countries.csv

```
d <- read.csv("Countries.csv")
str(d)
```

```
## 'data.frame':    209 obs. of  40 variables:
##  $ Country                            : Factor w/ 209 levels "Afghanistan
##  $ Continent                          : Factor w/ 8 levels "Asia
##  $ Adolescent_fert                    : int  NA 16 8 NA NA 140 NA 58 30 25 ...
##  $ Agricultural_land                  : int  380480 11230 411500 50 260 575900 140 1293550 13900
##  $ Aid_per_capita                     : int  NA 101 11 NA NA 27 94 3 64 NA ...
##  $ Air_transport                      : int  NA 195702 3037298 NA NA 239795 778260 6938436 55579
##  $ Arable_land_hectares               : num  7910000 578000 7450000 2000 1000 3300000 8000 28500
##  $ Birth_rate_crude                   : int  NA 17 21 NA NA 48 NA 18 12 NA ...
##  $ Death_rate_crude                   : int  NA 6 5 NA NA 21 NA 8 9 NA ...
##  $ Electric_consumption               : num  NA 3.68e+09 2.95e+10 NA NA ...
##  $ Fertility_rate_total               : int  NA 2 2 NA NA 7 NA 2 1 NA ...
##  $ Forest_area                        : int  8670 7940 22770 180 160 591040 90 330210 2830 NA ..
##  $ GDP                                : num  NA 4.79e+09 6.97e+10 NA NA ...
##  $ Health_expenditure                 : int  20 169 108 NA 2556 36 503 484 88 NA ...
##  $ Hospital_beds                      : int  NA 3 NA NA 3 NA 2 NA 4 NA ...
##  $ Immunization_DPT                   : int  76 98 88 NA 98 47 99 92 90 NA ...
##  $ Inflation_consumerpricesannual     : int  NA 2 2 NA NA 25 NA 10 1 3 ...
##  $ Internet_users                     : int  1 6 6 NA 33 1 35 18 5 24 ...
##  $ Landarea_sqkm                      : num  652090 27400 2381740 200 470 ...
##  $ Life_expectancy_birth_female       : int  NA 79 73 NA NA 44 NA 79 75 NA ...
##  $ Life_expectancy_birth_male         : int  NA 73 70 NA NA 40 NA 71 68 NA ...
##  $ Life_expectancy_birth_totalyears   : int  NA 76 72 NA NA 42 NA 75 71 NA ...
##  $ Literacy_rate_adult                : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ Low_birthweight_babies             : int  NA 7 NA NA NA NA 5 NA 8 NA ...
##  $ Militaryexpenditure                : int  10 1 3 NA NA 5 NA 1 3 NA ...
##  $ Mortalityrate_adultfemale          : int  NA 52 107 NA NA 439 NA 82 89 78 ...
##  $ Mortalityrate_adultmale            : int  NA 109 125 NA NA 492 NA 171 198 139 ...
##  $ Mortalityrate_infant               : int  NA 16 34 NA 3 154 10 14 23 NA ...
##  $ Personalcomputers                  : int  0 2 1 NA NA 1 15 9 10 8 ...
##  $ Physicians                         : int  0 NA NA NA NA NA NA NA NA NA ...
##  $ Populationgrowth                   : int  NA 1 1 2 0 3 1 1 0 1 ...
##  $ Population_total                   : int  NA 3153731 32854159 58300 66200 16095214 83039 3874
##  $ Population_female                     : int  NA 50 49 NA NA 51 NA 51 53 52 ...
##  $ Prevalence_HIV                        : int  0 0 0 NA NA 4 NA 1 0 NA ...
##  $ Prevalence_overweightchildrenunder5   : int  NA NA NA NA NA NA NA 10 12 NA ...
##  $ Ratiooffemaletomale_primaryenrollment : int  59 NA 93 NA 95 NA NA NA 104 97 ...
##  $ Ratiooffemaletomale_secondaryenrollment: int  33 NA 108 NA 112 NA NA NA 103 103 ...
##  $ Roads_totalnetwork_km                 : int  NA NA NA NA NA NA NA NA 7515 NA ...
##  $ Surfacearea_sq.km                     : num  652090 28750 2381740 200 470 ...
##  $ TaxrevenueofGDP                       : int  NA NA 31 NA NA NA NA NA 14 NA ...
```

```
anyNA(d)
```

```
## [1] TRUE
```

```
sum(is.na(d))
```

```
## [1] 2064
```

Unfortunately there are quite a few missing data points in the file, as the function **anyNA()** and **is.na()** reveal. This may present some challenges. You may need to include na.rm=T when calling mean, sum, or other functions.

```
mean(d$Agricultural_land)
```

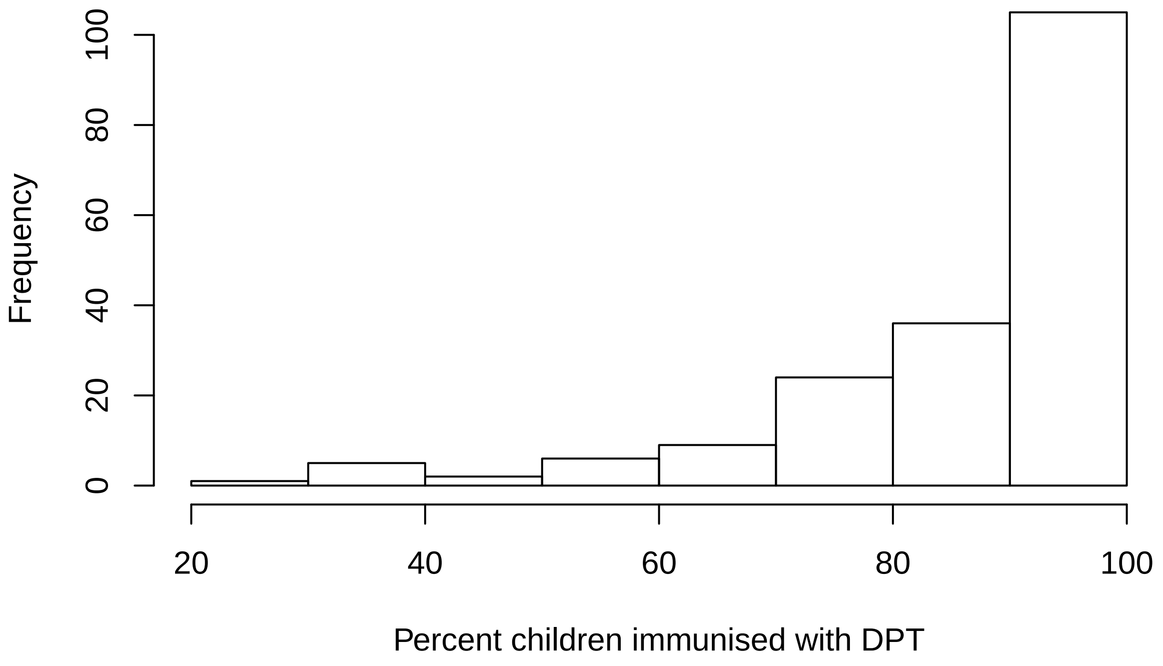
```
## [1] NA
```

```
mean(d$Agricultural_land, na.rm = T)
```

```
## [1] 235563.9
```

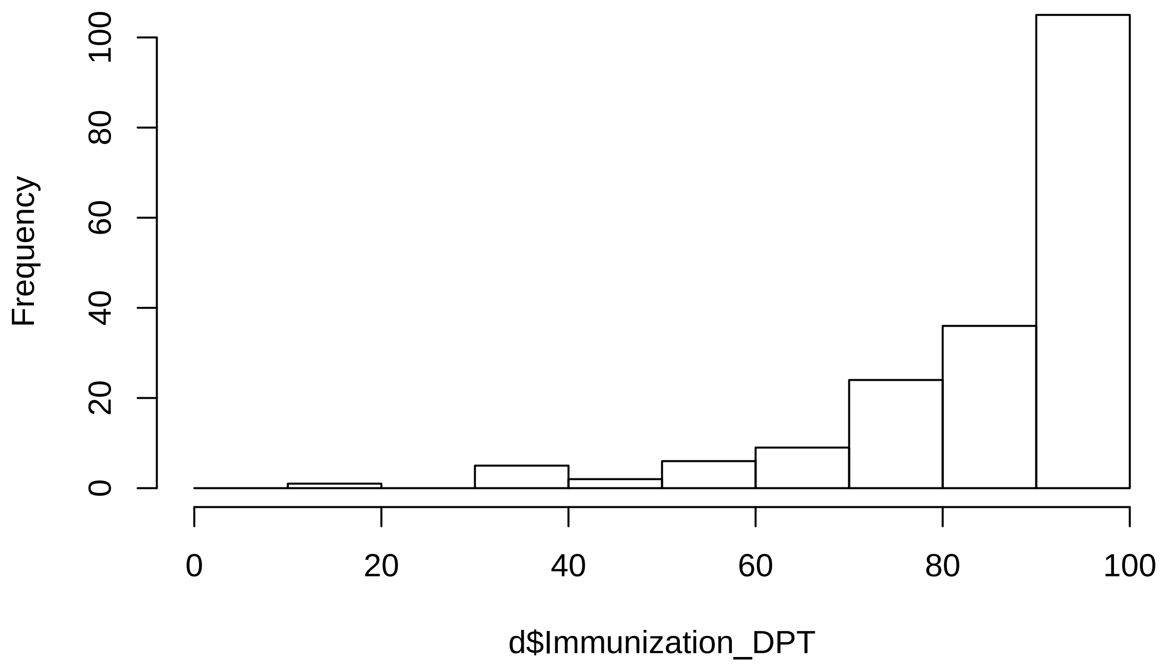
DPT is a combination vaccine against diphtheria, pertussis and tetanus. Use the **hist()** function to plot a histogram for the Immunization DPT variable, which contains data on the % of children ages 12 – 23 months who received this vaccine in different countries.

```
hist(d$Immunization_DPT, xlab = "Percent children immunised with DPT",
    main = NULL)
```



Change the binning to a custom interval width of 10, by setting the breaks = seq(0,100,10):

```r
hist(d$Immunization_DPT, breaks = seq(0, 100, 10), main = NULL)
```



Histograms can also be returned as objects to work from. Typically you would also set plot=FALSE when doing this:

```r
h <- hist(d$Immunization_DPT, plot = FALSE)
str(h)
```

```
## List of 6
##  $ breaks  : int [1:9] 20 30 40 50 60 70 80 90 100
##  $ counts  : int [1:8] 1 5 2 6 9 24 36 105
##  $ density : num [1:8] 0.000532 0.00266 0.001064 0.003191 0.004787 ...
##  $ mids    : num [1:8] 25 35 45 55 65 75 85 95
##  $ xname   : chr "d$Immunization_DPT"
##  $ equidist: logi TRUE
##  - attr(*, "class")= chr "histogram"
```

```r
h$breaks
```

```
## [1]  20  30  40  50  60  70  80  90 100
```

```r
h$counts
```

```
## [1]   1   5   2   6   9  24  36 105
```

Use this function to generate histograms for personal computers per 100 people, and internet users per 100 people.

To count up particular information, use the sum and boolean commands.

First, the number of countries with >90% immunisation

```r
sum(d$Immunization_DPT > 90, na.rm = T)
```

```
## [1] 105
```

Then, the number of countries with immunisation rates >= 70 and <80%:

```r
sum(d$Immunization_DPT >= 70 & d$Immunization_DPT < 80, na.rm = T)
```

```
## [1] 20
```

If we are interested in the descriptive statistics (values) only, we may choose to use the function available in a different package, such as **stat.desc()** from the **pastecs** package, or the **describe()** function from the **psych** package.

If running this on your own computer, you will likely have to install the pastecs package first (remove the # from in front of install.packages("pastecs"), below.

Use the **stat.desc** function to produce a table containing mean, standard deviation, minimum, maximum, and SE mean for the three variables examining life expectancy at birth (male, female, and total).

```r
# install.packages('pastecs')
library(pastecs)
options(scipen = 9)
variable <- d$Immunization_DPT
cbind(stat.desc(variable, basic = T))
```

```
##                       [,1]
## nbr.val        188.0000000
## nbr.null         0.0000000
## nbr.na          21.0000000
## min             20.0000000
## max             99.0000000
## range           79.0000000
## sum          16268.0000000
## median          93.0000000
## mean            86.5319149
## SE.mean          1.1110899
## CI.mean.0.95     2.1918814
## var            232.0898851
## std.dev         15.2344965
## coef.var         0.1760564
```

Alternatively, the **psych** package has a **describe()** function that provides similar summary information:

```r
# install.packages('pysch')
library(psych)
describe(variable)
```

```
##    vars   n  mean    sd median trimmed  mad min max range  skew kurtosis
## X1    1 188 86.53 15.23     93   89.49 7.41  20  99    79 -1.83     3.33
##      se
## X1 1.11
```

The variable population_female indicates the percent of the total population that is female. Calculate mean and 95% confidence interval for this variable.

Answer questions 10-14 on Sakai