

Lab 2 – Creating Charts and Tables in R

This lab contains some numbered questions. You are required to submit:

1. One pdf document (Lab 2.pdf) that contains:
 - your written answers to the question
 - the commands you used to answer the question, when asked
2. One R script (Lab 2.R) that contains all the code used to produce your answers. The script must run without errors.

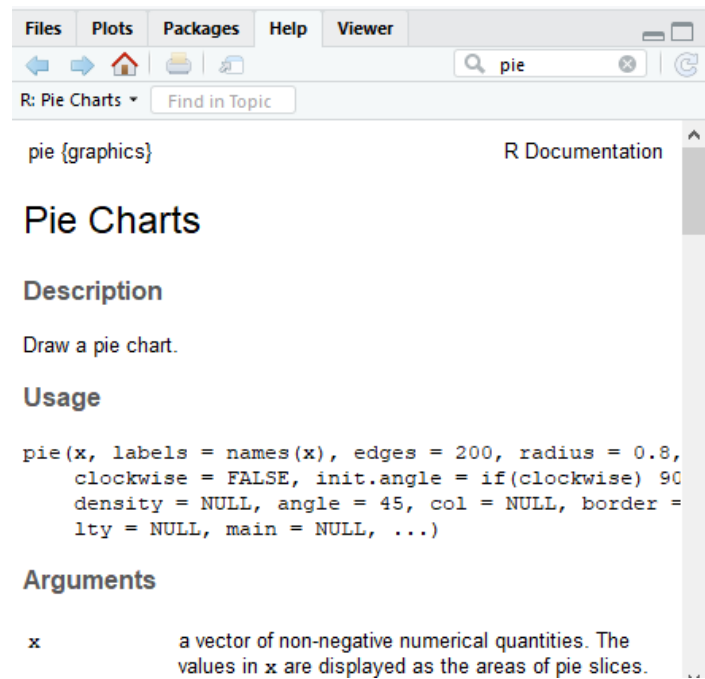
Submit your files to the Lab 2 folder in Learning Hub by 11:59pm next school day.

Pie Charts in R

We will begin by making a pie chart. Pie charts are most useful for categorical data.

Let's start with this data: "At a local college, 10% of students live on campus, 30% live with their parents, 5% live alone, 35% live off-campus with roommates, and 20% live with a spouse." We wish to present it in a pie chart.

We need to know how to store this data in a way that can be converted to a pie chart. Go to the **Help** tab of the **Miscellaneous** window and type "pie". This gives you a choice between two similar commands for making pie charts. Here is what R's help file tells you about the "pie" command:



The screenshot shows the R Help window with the 'pie' command selected. The window has tabs for Files, Plots, Packages, Help, and Viewer. The Help tab is active, and the search bar contains 'pie'. The search results show 'R: Pie Charts' selected. The documentation for 'pie' is displayed, including a description, usage, and arguments.

R: Pie Charts Find in Topic

pie (graphics) R Documentation

Description

Draw a pie chart.

Usage

```
pie(x, labels = names(x), edges = 200, radius = 0.8,
    clockwise = FALSE, init.angle = if(clockwise) 90
    density = NULL, angle = 45, col = NULL, border =
    lty = NULL, main = NULL, ...)
```

Arguments

x a vector of non-negative numerical quantities. The values in **x** are displayed as the areas of pie slices.

The first two arguments are the most important.

- `x` a vector of non-negative numerical quantities. The values in `x` are displayed as the areas of pie slices.
- `labels` one or more expressions or character strings giving names for the slices. ... For empty or `NA` ... labels, no label nor pointing line is drawn.

So, the first argument is a list of numbers that will represent areas. For our data, this would be the percentage of students in each type of living arrangement.

Let's go back to the **console** window and create a vector with those percentages, in order:

```
> percents <- c(10, 30, 5, 35, 20)
```

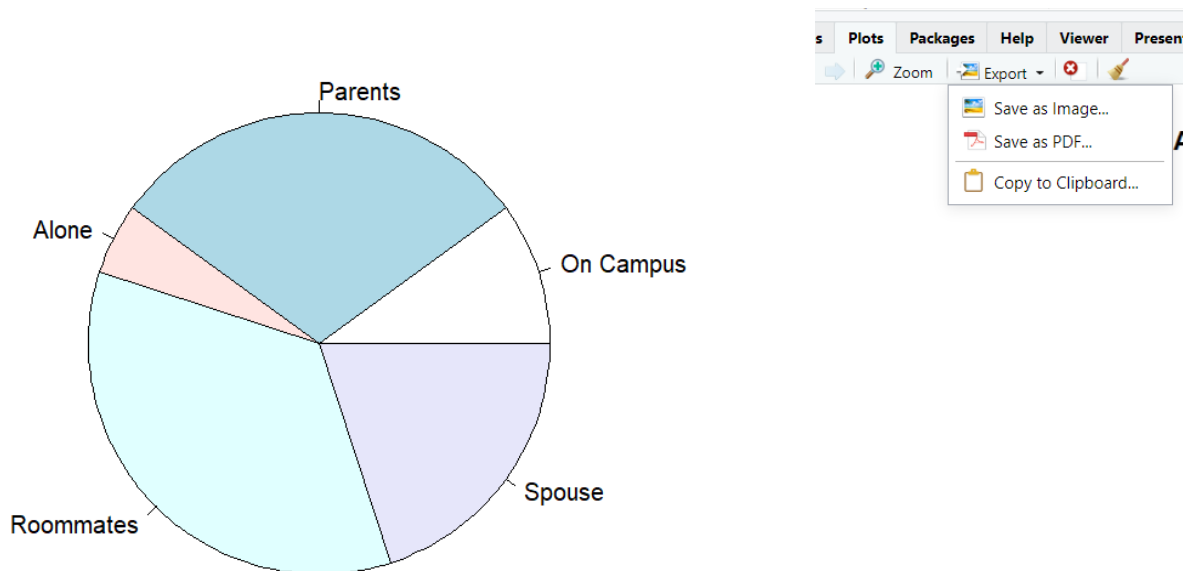
and also a vector with the labels:

```
> home.type <- c("On Campus", "Parents", "Alone", "Roommates", "Spouse")
```

Now we can create our pie chart:

```
> pie(percents, home.type)
```

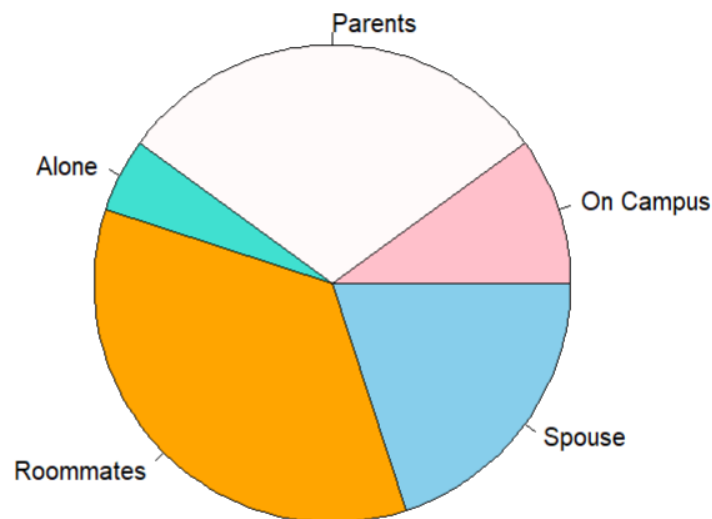
A pie chart will appear in the **Plots** tab. You can export it or screenshot it.



1. (This is the first Lab 2 question) Read over the documentation for the **pie** command and create the following pie chart. Copy/paste your pie chart into your solutions document.

- include a title in the pie chart
- color names can be found here: <https://r-charts.com/colors/>

Living Arrangements of Students



We can also create pie charts from built-in datasets. In this example, we are going to use the data from the built-in dataframe **survey**, which gives data about 237 students. This data is part of the library **MASS**, so first load that library and then view it:

```
> library(MASS)
> View(survey)
```

To see just the first few rows of the data frame, run the following:

```
> head(survey)
```

	Sex	Wr.Hnd	NW.Hnd	W.Hnd	Fold	Pulse	Clap	Exer	Smo
1	Female	18.5	18.0	Right	R on L	92	Left	Some	
2	Male	19.5	20.5	Left	R on L	104	Left	None	
3	Male	18.0	13.3	Right	L on R	87	Neither	None	
4	Male	18.8	18.9	Right	R on L	NA	Neither	None	
5	Male	20.0	20.0	Right	Neither	35	Right	Some	
6	Female	18.0	17.7	Right	L on R	64	Right	Some	
7	Male	17.7	17.7	Right	L on R	83	Right	Freq	
8	Female	17.0	17.3	Right	R on L	74	Right	Freq	
9	Male	20.0	19.5	Right	R on L	72	Right	Some	
10	Male	18.5	18.5	Right	R on L	90	Right	Some	
11	Female	17.0	17.2	Right	L on R	80	Right	Freq	
12	Male	21.0	21.0	Right	R on L	68	Left	Freq	
13	Female	16.0	16.0	Right	L on R	NA	Right	Some	
14	Female	19.5	20.2	Right	L on R	66	Neither	Some	

Before going further, let's read about the data set:

```
> help(survey)
```

Student Survey Data

Description

This data frame contains the responses of 237 Statistics I students at the University of Adelaide to a number of questions.

Usage

```
survey
```

Format

The components of the data frame are:

Sex

The sex of the student. (Factor with levels "Male" and "Female".)

Wr.Hnd

span (distance from tip of thumb to tip of little finger of spread hand) of writing hand, in centimetres.

NW.Hnd

span of non-writing hand.

W.Hnd

writing hand of student. (Factor, with levels "Left" and "Right".)

Pie charts are best for qualitative data. Let's create one for the variable W.Hnd, the student's writing hand.

There are only two possible values of W.Hnd, which R refers to as "levels". To see what they are, use:

```
> levels(survey$W.Hnd)
```

Extract the data column with the following command:

```
> hands <- survey$W.Hnd
```

This creates a vector called "hands", containing the values of the variable W.Hnd for all students surveyed. To generate a frequency table for this variable, use:

```
> freq.tab <- table(hands)
```

Retrieve the values for Left or Right using:

```
> freq.tab["Left"]
```

```
> freq.tab
hands
Left Right
18    218
```

The table `freq.tab` is actually a *named vector*. Pull out the names using:

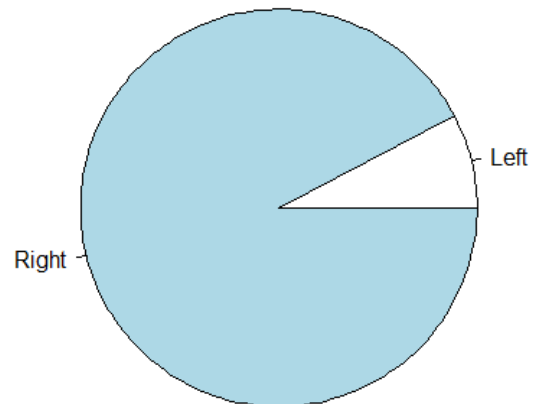
```
> names(freq.tab)
```

R will use these names as the labels if we make a pie chart:

```
> pie(freq.tab)
```

equivalently, doing all this in one command:

```
> pie( table( survey$W.Hnd ))
```



Suppose we want to augment the labels to include the number of students in each category. We can create new labels like this:

```
> new.labels <- paste(names(freq.tab), "\n(", freq.tab, ")", sep="")
```

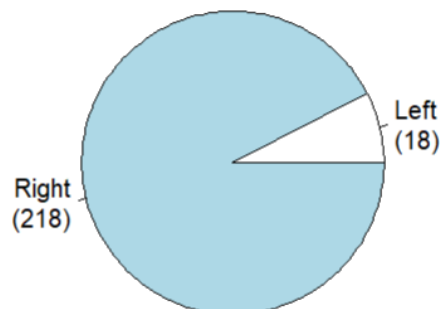
(The “\n” means *newline*.)

```
> new.labels
```

```
[1] "Left\n(18)"    "Right\n(218)"
```

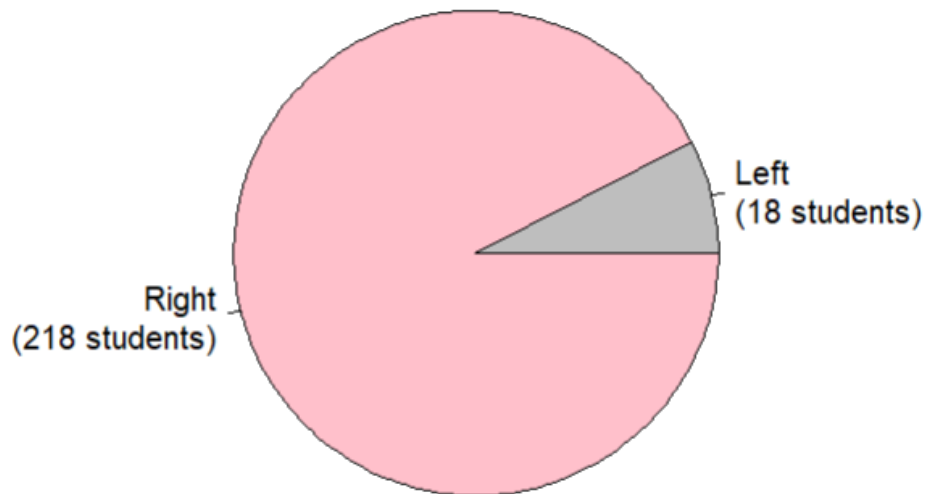
Then use the new labels to create a better pie chart:

```
> pie( freq.tab, new.labels)
```



2. Create the following pie chart. Include the command you used. **Export** the pie chart and paste it into your solutions.

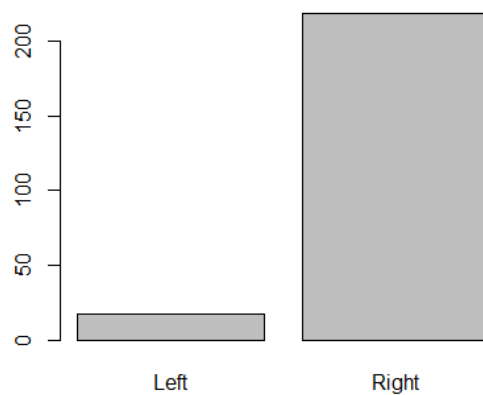
Writing Hand of Students (n =236)



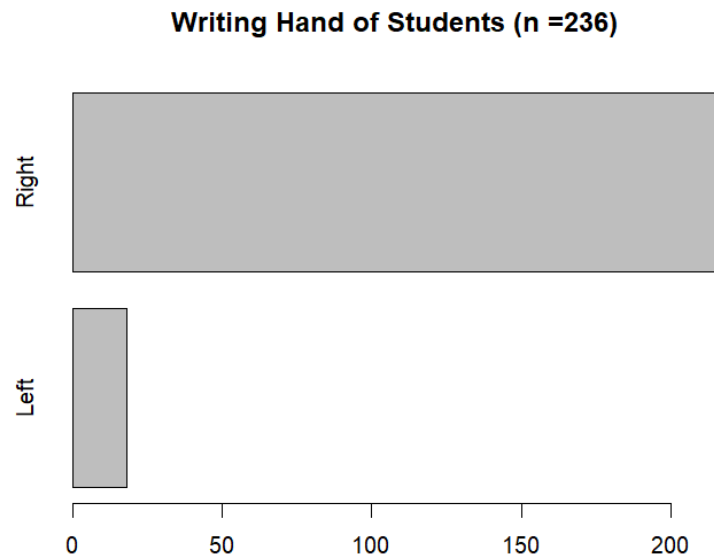
Bar graphs in R

Alternatively, we can create a bar graph to represent the writing hands of the students. The following command uses the same table you used to create your pie chart.

```
> barplot(freq.tab)
```



3. Read the documentation for **barplot** command and produce the following chart:



Stem Plots in R

For quantitative data, other kinds of charts are preferable. We are going to create a stem plot for the students' heights. We only need one column for our stem plot – the **Height** column. Extract it using:

```
> survey$Height
```

Have a look at the first few rows:

```
> head( survey$Height )
```

We can use **cbind()** to get the data as a column:

```
> cbind(survey$Height)
[,1]
[1,] 173.00
[2,] 177.80
[3,]  NA
[4,] 160.00
[5,] 165.00
```

Create a default stem-leaf plot of heights using:

```
> stem(survey$Height)
```

The decimal point is 1 digit(s) to the right of the |

```
15 | 0224
15 | 555566777777899
16 | 0000000033333334444
16 | 5555555555555555677777788888888888899999
17 | 000000000000000001111122222223333333334
17 | 55555555556677778888999999
18 | 0000000000000000023333333344
18 | 5555555777888899
19 | 00011123
19 | 56
20 | 0
```

Note that the first row contains heights between 150 cm and 154 cm; the second row contains heights between 155 cm and 159 cm. There are a lot of heights between 165 cm and 169 cm, and so that row is quite large. Read up on the **stem** function to see how you can customize your stem plot.

4. Give the command used to produce the following stem plot.

The decimal point is at the |

```
150 | 0
152 | 045
154 | 9900
156 | 02000555
158 | 000
160 | 00000000
162 | 56666000
164 | 000000000000000001111
166 | 45000000066666
168 | 0000000059002
170 | 00000000000002222000005
172 | 00000007777770000
174 | 00000033333
176 | 005500088
178 | 00500011
180 | 00000000333333333
182 | 059999000
184 | 0000000044
186 | 000
188 | 000000
190 | 0005558
192 | 0
194 | 0
196 | 0
198 |
200 | 0
```


Back-to-back Stem Plots

Suppose we are interested in comparing the heights of male and female students. We can create back-to-back stem plots to compare the two sets of heights.

To do this, we need to install the **aplpack** package.

```
> install.packages("aplpack")
> library(aplpack)
```

(Note: If you're on a Mac, you may get an error message and a prompt to install an additional package called XQuartz.)

The command **stem.leaf.backback** allows us to plot the male and female heights back to back.

We need two vectors: one for the men's heights, and one for the women's. Using commands like in Lab 1, create two lists: `mens.heights` and `womens.heights`.

Create the plot using:

```
> stem.leaf.backback( mens.heights, womens.heights)
```

The result probably does not fit nicely on your screen. It looks like:

```
> stem.leaf.backback( mens.heights, womens.heights)
```

1 2: represents 12, leaf unit: 1				
	mens.heights		womens.heights	
1	4	15*	02234	5
		15.	5566777777899	18
3	00	16*	000000222223334444	36
13	8877755555	16.	55555555555566777777778888888999	(35)
30	4332222111000000	17*	00000000000011122222233	31
51	99999887777766655555	17.	555568	7
(27)	443332222200000000000000	18*	0	1
28	9987777755555555	18.		
11	31000000	19*		
3	65	19.		
1	0	20*		
<hr/>				
n:	118		118	
NAs:	12		16	

In this plot, there is too much data per stem. We can increase the number of stems with the **m** argument. Second, there are extra columns of numbers representing cumulative data totals. Get rid of the cumulative sums by setting **depths** to **FALSE**.

```
> stem.leaf.backback(mens.heights, womens.heights, m=10, depths=FALSE)
```

```

1 | 2: represents 12, leaf unit: 1
  Mens.heights      Womens.heights
-----|-----
      | 15 | 0
      | 15 | 
      | 15 | 22
      | 15 | 3
4 | 15 | 4
      | 15 | 55
      | 15 | 66
      | 15 | 777777
      | 15 | 8
      | 15 | 99
00 | 16 | 000000
      | 16 | 
      | 16 | 22222
      | 16 | 333
      | 16 | 4444
55555 | 16 | 55555555555555
      | 16 | 66
777 | 16 | 777777777
88 | 16 | 88888888
      | 16 | 999
000000 | 17 | 0000000000000
111 | 17 | 111
22222 | 17 | 2222222

```

(plot continues...)

5. Give the R command that produces the back-to-back stem plot that begins with this output, and provide your output as well:

```

1 | 2: represents 12, leaf unit: 1
  Mens.heights      Womens.heights
-----|-----
      | 15* | 02234
      | 15. | 5566777777899
00 | 16* | 000000222223334444

```

6. Based on the back-to-back stem plots, what general observation can you make about womens' and mens' heights based on this survey? Make clear reference to the plot(s).

Frequency distributions in R

The stem plots for heights are useful, but they have serious limitations. (For example, what if we had thousands of data values?) It is useful to condense the raw data into a frequency table.

We will make a frequency table for the **survey\$Height** data using classes 5 cm in width. Start by getting the range of the data

```
> range(survey$Height)
```

Unfortunately, the output is unusable since Height contains NA values. To ignore NA values, use:

```
> numerical.heights <- survey$Height[ !is.na(survey$Height) ]
```

Now we can get the range:

```
> range(numerical.heights)
[1] 150 200
```

This tells us that the shortest student is 150 cm tall and the tallest is 200 cm tall. We will cut the interval [150, 200] every 5 cm:

```
> upper.limits <- seq(145, 200, by=5)
```

We start at 145 so that our first bin of (145, 150] will include any students who are 150 cm. We can now assign a class to each numerical height:

```
> heights.classes = cut(numerical.heights, upper.limits,
right=TRUE)
```

(Here `right=TRUE` says that `upper.limits` contains the *right* limit of each interval.) Finally, we make a frequency table of the heights:

```
> heights.freq = table(heights.classes)
> cbind( heights.freq)
```

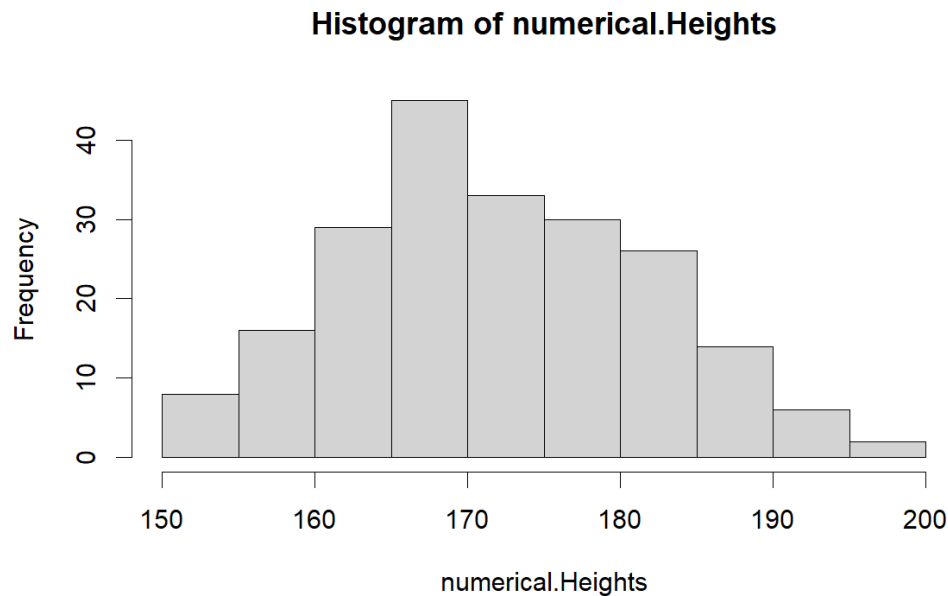
	heights.freq
(145,150]	1
(150,155]	7
(155,160]	16
(160,165]	29
(165,170]	45
(170,175]	33
(175,180]	30
(180,185]	26
(185,190]	14
(190,195]	6
(195,200]	2

- Using the **survey** data, create two frequency tables: one for the ages of Male students and one for the ages of Female students. Choose a reasonable number of classes and use the same class limits for both groups. Submit the commands as well as the resulting tables. Based on your tables, does the distribution of ages differs significantly between male and female students? Explain.

Histograms in R

We can also display the height data from the last section in histogram form:

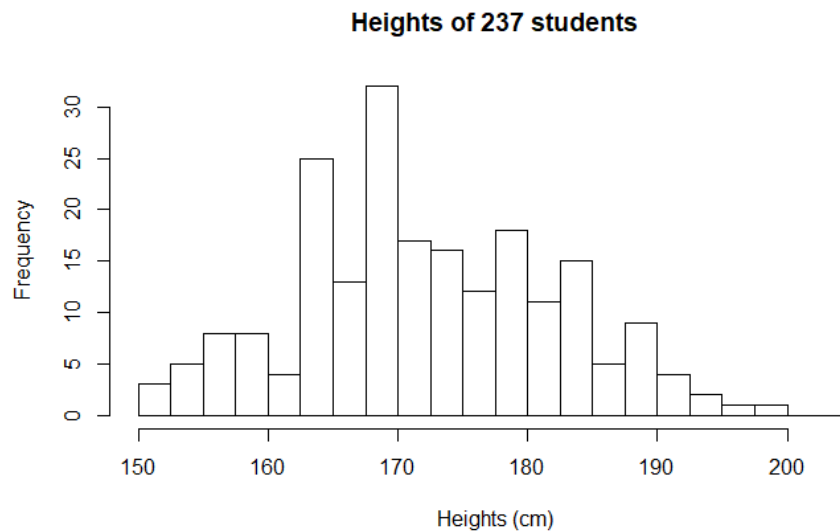
```
> hist(numerical.heights)
```



There are a few things to note:

- The bins are each of width 5. This might not be what we want.
- The label for the x-axis and the title are inadequate.

8. Go to the **Help** tab and look up the **hist** function. Produce the following histogram. Include the histogram as well as the command(s) you used to produce it.



Cumulative frequency tables in R

Sometimes we are interested in the cumulative frequencies of data. For instance, we might want to know how many students are ≤ 175 cm in height. We can create a *cumulative* frequency table based on the frequency table we already created:

```
> cumul.freq <- cumsum(heights.freq)
> cbind( cumul.freq )
```

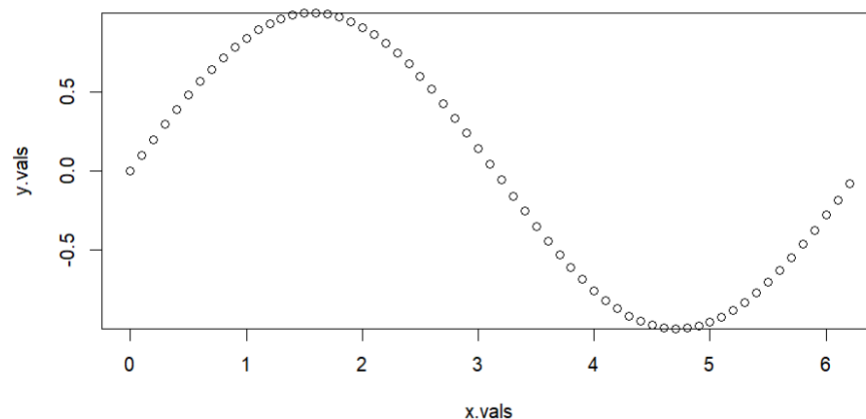
	cumul.freq
(145,150]	1
(150,155]	8
(155,160]	24
(160,165]	53
(165,170]	98
(170,175]	131
(175,180]	161
(180,185]	187
(185,190]	201
(190,195]	207
(195,200]	209

Here, we can see that 131 students are ≤ 175 cm in height.

Ogives in R

We can represent the cumulative frequencies in an *ogive* by plotting the cumulative frequencies against the heights. The **plot** function allows us to do this. Try using **plot**:

```
> x.vals <- seq(0, 2*pi, 0.1)
> y.vals <- sin(x.vals)
> plot(x.vals, y.vals)
```



To plot an *ogive* for student heights, we will use **plot** with

- upper limits as x coordinates
- cumulative frequency as y coordinates

Notice that these two vectors have different lengths:

```
> length( upper.limits )
> length( cumul.freq )
```

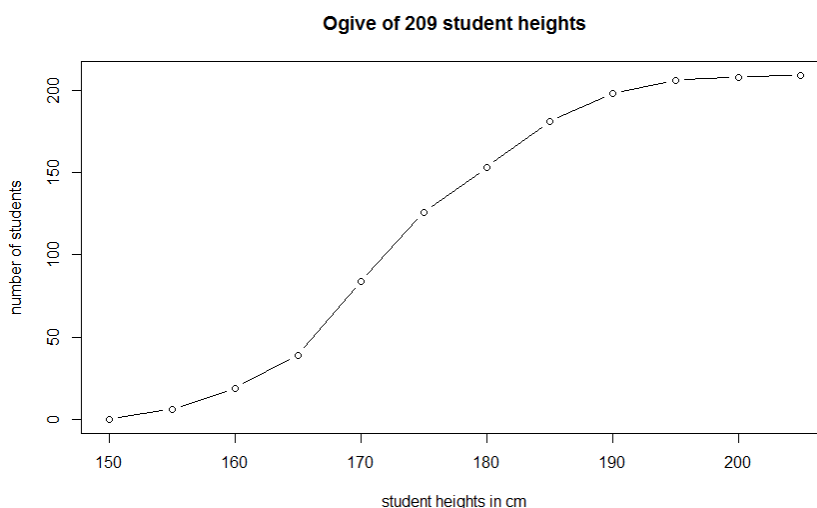
We need to insert a 0 at the beginning of **cumul.freq** before we can plot the ogive:

```
> cumul.freq <- c(0, cumul.freq)
```

Now plot:

```
> plot(breaks, cumul.freq, type="p", pch=19 )
```

9. Read the help file for **plot** and create the following ogive. Include the command you used.

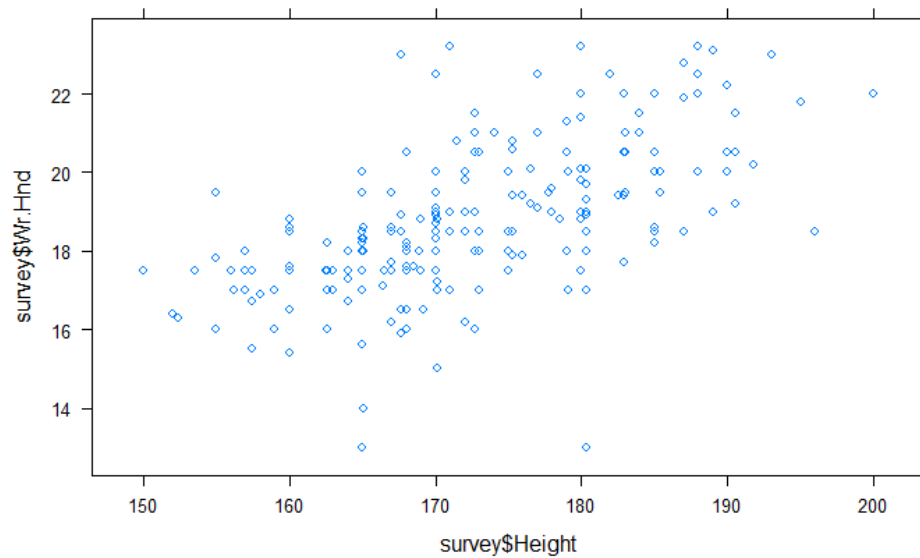


Scatterplots in R

Scatter plots help to visualize the relationship between two numerical variables. For example, we can see how height and weight are correlated. In general, we expect a tall person to weigh more than a short person – but of course there are exceptions. (We say there is a *positive correlation* between height and weight, but not a perfect correlation.)

To look for a relationship between the variables **Height** variable and the **Wr.Hnd** variable (which gives the span of the student's writing hand), make the scatterplot:

```
> library(mosaic)
> xyplot(survey$Wr.Hnd ~ survey$Height)
```



10. Use the help function to give descriptive axis labels and a title to the scatterplot above. Include your scatterplot and the command(s) you used to produce it.
11. Create two scatterplots: one that plots **Wr.Hnd** (y-axis) against **NW.Hnd** (x-axis) and one that plots **Height** (y-axis) against **Pulse** (x-axis). Both plots should have descriptive titles and axis labels. Include your plots along with the command(s) used to produce them.

Which of these two pairs of variables are more *strongly* correlated? Explain what you see in the scatter plots that indicates that one pair of variables is more strongly correlated.