# **Lab 2 – Creating Charts and Tables in R**

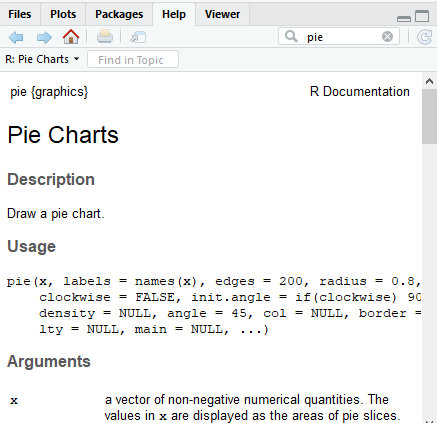
**To submit: answers to all numbered questions. When the question asks you to generate output in R, such as a graph, submit the output in the Word document as part of your answer. Also submit all commands you used to generate your output.**

# Pie charts in R

We will begin by making a pie chart. Pie charts are most useful for qualitative 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 arguments are pretty overwhelming, so let’s look at the first two:

|  |  |
| --- | --- |
| 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. Other objects are coerced by[as.graphicsAnnot](http://127.0.0.1:29446/help/library/graphics/help/as.graphicsAnnot). For empty or NA (after coercion to character) labels, no label nor pointing line is drawn. |

That is, 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. So let’s go back to the **console** window and create a vector with those percentages, in order:

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

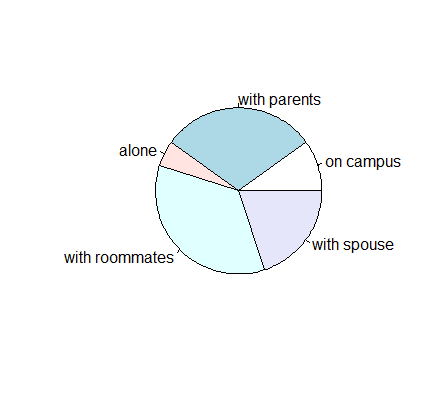
The second argument, **labels**, is exactly that. So let’s create that second vector, making sure the labels are in the correct order:

homeType<-c("on campus", "with parents", "alone", "with roommates", "with spouse")

Now we can create our pie chart:

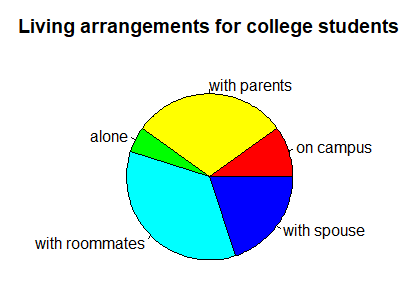
pie(percents, homeType)

A pie chart should appear in the **Plots** tab of the **miscellaneous** window. You can export it as a bitmap.



We can use the other arguments to change the colours on the pie chart and the title.

1. Read over the documentation for the **pie**command and create the following pie chart. Copy the pie chart into a Word document and include the command you used to produce it.



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**, whichgives data about 237 students. This data is part of the library **MASS**, so type the command

library(MASS)

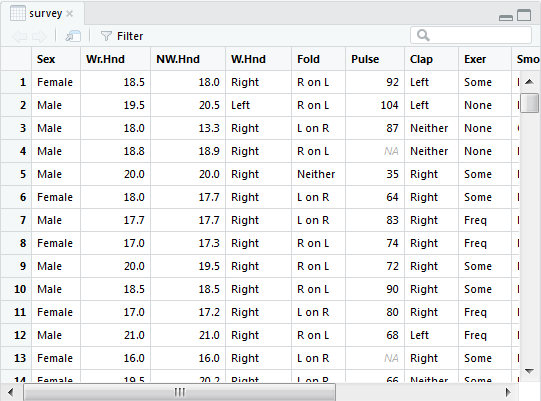
in the console to load it.

To view the **survey** dataset, type:

View(survey)

into the console window. (**Note: R is case-sensitive.)**

This will display your data in a tab in the **Script** window.



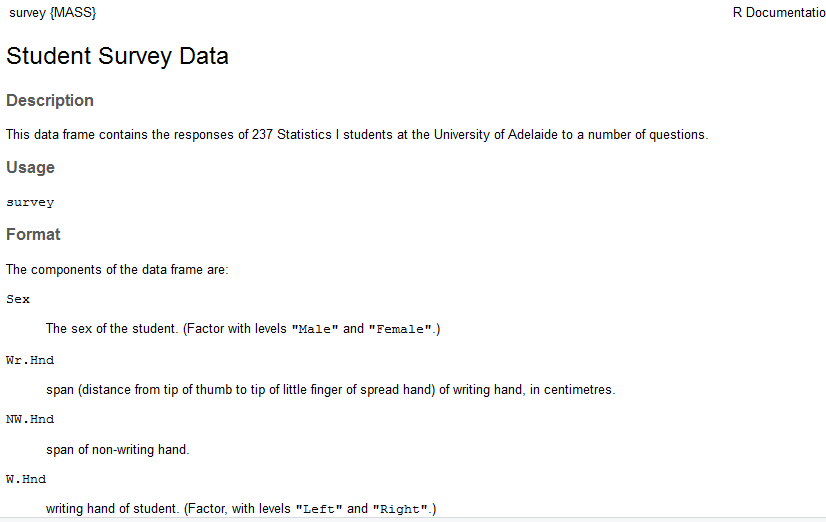
This is a useful command. Once we get to the stage of uploading your own data, it is very important to view the data to make sure it uploaded correctly!

You can also see part of your data in the console by typing **survey** in the console window.

There are many things we can do with this data. First, type

help(survey)

in the console and take a few minutes to read the information about this dataset, which appears in the **Help** window.



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

As before, we need two pieces of information: the categories, and the number of entries in each category. In this case, the categories are “Left” and “Right”, for the student’s writing hand; and the number of entries are obtained from the **W.Hnd** column of the data.

We can extract that column with the following command:

> hands<-survey$W.Hnd

This creates a list called “hands”, which consists of the entries of the **W.Hnd** column of the **survey** dataframe. If you type **hands** into the console, you can see this list.

The **table** command aggregates this information:

> tab<-table(hands)

> tab

hands

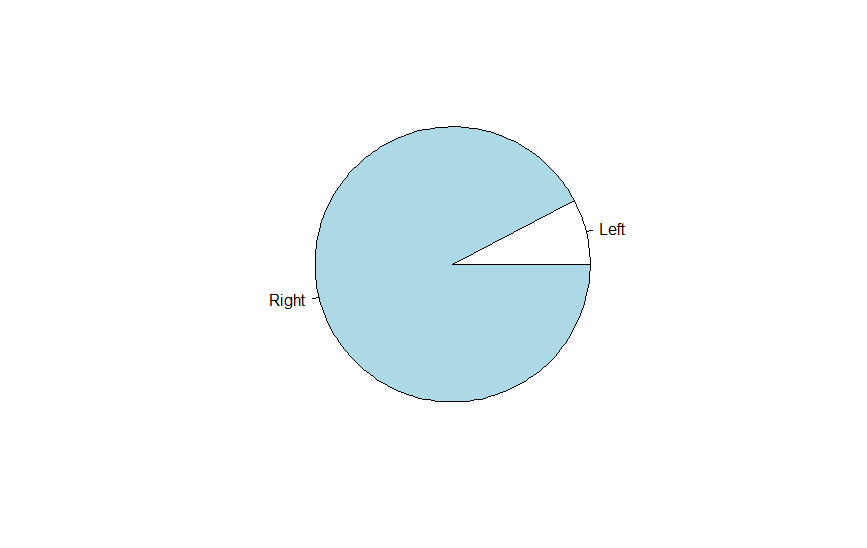
Left Right

18 218

In other words, 18 of the students surveyed are left-handed and 218 are right-handed.

We can then create a simple pie chart:

> pie(table(hands))



We can also add the numbers of students of each type by “pasting” those numbers to the labels:

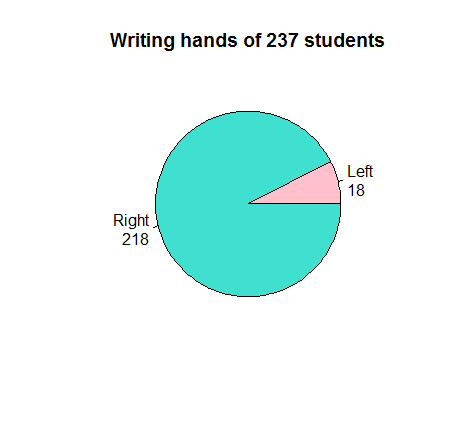
> lbls<-paste(rownames(tab), sep="\n", tab)

> lbls

[1] "Left\n18" "Right\n218"

The “\n” tells R to separate the column names and numbers with a new line.

1. Create the following pie chart. Include the command you used. (Hint: the colours involved are pink and turquoise.) Save the pie chart using the **Export** command above your chart. Copy the pie chart into a Word document and include the command you used to produce it.



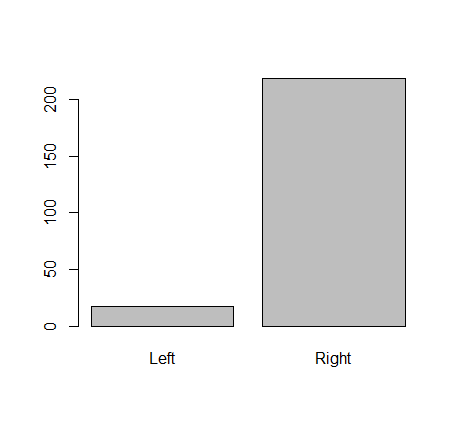
# Bar graphs in R

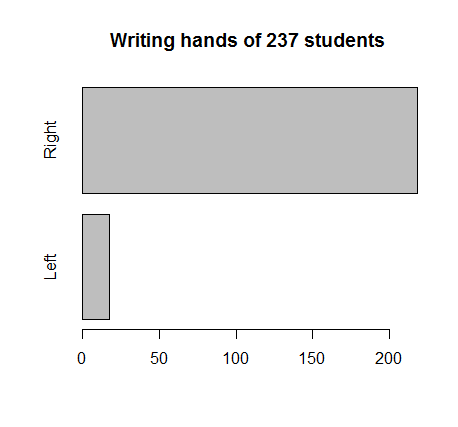
Alternatively, we can create a bar graph to represent the writing hands of the students.

The command

> barplot(tab)

based on the same table you used to create your pie chart, gives the following graph:

1. Read the documentation for the **barplot**() command and produce the following barplot:



# 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. We can extract it using the command

> survey$Height

This gives the data as a list. The first few rows look like this:

[1] 173.00 177.80 NA 160.00 165.00 172.72 182.88 157.00

[9] 175.00 167.00 156.20 NA 155.00 155.00 NA 156.00

[17] 157.00 182.88 190.50 177.00 190.50 180.34 180.34 184.00

We can use the **cbind** command 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

We can get a basic stem-leaf plot of heights using the command

> stem(survey$Height)

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

15 | 0224

15 | 555566777777899

16 | 00000000333333334444

16 | 555555555555555555677777777888888888888899999

17 | 000000000000000000111112222222233333333334

17 | 555555555566777778888999999

18 | 0000000000000000023333333344

18 | 55555555777888899

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.

1. Give the command you 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 | 0000000000000000001111

166 | 45000000066666

168 | 0000000059002

170 | 000000000000002222000005

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)

The command **stem.leaf.backback** allows us to plot the male and female heights back to back. The most basic plot provides only this data. We need two vectors: one for the men’s heights, and one for the women’s.

Create two lists: **MensHeights** and **WomensHeights**. You will have to use commands from last week’s lab.

The command **stem.leaf.backback(MensHeights, WomensHeights)** will provide a back-to-back stem plot, but it is not formatted very nicely. For one, there is too much data per stem. In addition, there are extra columns of numbers representing cumulative data totals, and this just adds to the clutter. We can increase the number of stems with the **m** argument, and get rid of the cumulative sums by setting **depths** to **FALSE**. We then get this back-to-back stem plot:

> stem.leaf.backback(MensHeights, WomensHeights, m=10, depths=FALSE)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1 | 2: represents 12, leaf unit: 1

MensHeights WomensHeights

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| 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 |5555555555555

| 16 |66

777| 16 |777777777

88| 16 |88888888

| 16 |999

000000| 17 |000000000000

111| 17 |111

22222| 17 |2222222

33| 17 |33

4| 17 |

555555| 17 |5555

666| 17 |6

77777| 17 |

88| 17 |8

99999| 17 |

0000000000000000| 18 |0

| 18 |

222222| 18 |

333| 18 |

44| 18 |

55555555| 18 |

| 18 |

777777| 18 |

8| 18 |

99| 18 |

000000| 19 |

1| 19 |

| 19 |

3| 19 |

| 19 |

5| 19 |

6| 19 |

| 19 |

| 19 |

| 19 |

0| 20 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

n: 118 118

NAs: 12 16

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Based on the back-to-back stem plots for men’s and women’s heights, what is one difference between the datasets that is immediately apparent? Explain, making reference to the plot.

# Frequency distributions in R

The stem plots for heights kept a lot of detail and our ability to customize them was limited. (What would have happened if we wanted to group by 3’s instead of by 1’s?). We probably don’t care how many students were, say, exactly 173 cm tall. If we just know how many were between 170 cm and 175 cm, that’s probably good enough.

We will make a frequency table for the **survey$Height** data.

Let’s say we want to group the data in increments of 5 cm. We will start by getting the range of the data with the command

> range(survey$Height)

The output is

[1] NA NA

Hmm…the “NA” entries in the **Height** field, which didn’t affect our stem plot, are suddenly causing problems. Let’s filter them out:

> numHeights<-survey$Height[!is.na(survey$Height)]

This gives a list of numerical heights only.

Now we can get the range:

> range(numHeights)

[1] 150 200

This tells us that the shortest student is 150 cm tall and the tallest is 200 cm tall. That tells us what our bins should be.

Next we cut the interval [150,200] into sub-ranges of 5cm each.

> breaks = seq(150, 205, by=5)

> breaks

[1] 150 155 160 165 170 175 180 185 190 195 200 205

We go up to 205 because otherwise our last bin is [195, 200) which excludes any student who is 200 cm tall.

We can now sort the heights into those intervals:

numHeights.cut = cut(numHeights, breaks, right=FALSE)

Finally, we are ready to make a frequency table of the heights:

> numHeights.freq = table(numHeights.cut)

> cbind(numHeights.freq)

numHeights.freq

[150,155) 6

[155,160) 13

[160,165) 20

[165,170) 45

[170,175) 42

[175,180) 27

[180,185) 28

[185,190) 17

[190,195) 8

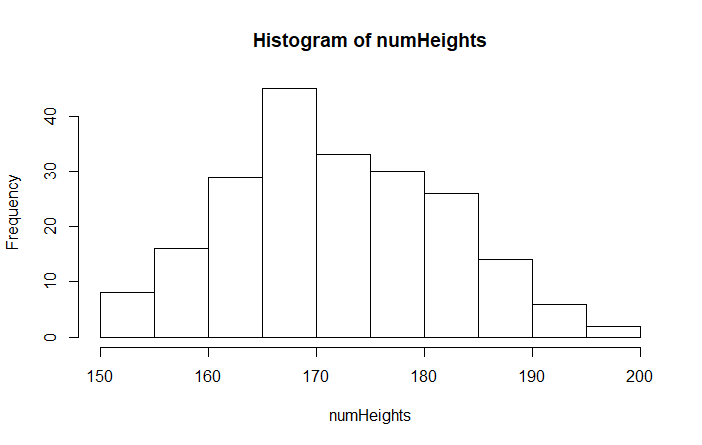
[195,200) 2

[200,205] 1

1. Using the **survey** data, createtwo frequency tables for student ages: one for male students, and one for female students. Choose a “reasonable” number of bins, and use the same bins for both groups. Include the tables as well as the commands you used to create them. Based on your tables, does it look like the distribution of ages differs significantly between male and female students? Explain.

# Histograms in R

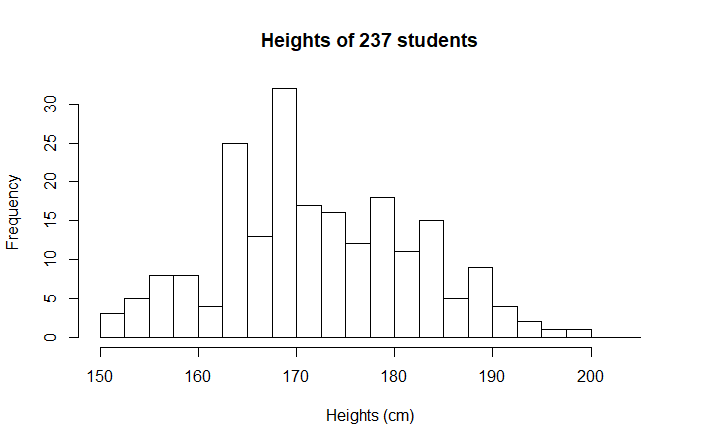
We can also display the height data from the last section in histogram form. The command **hist(numHeights)** gives this histogram:



There are a few things to note:

* The bins are each of width 5. This isn’t necessarily bad, but it might not be what we want.
* The label for the x-axis and the title aren’t great.

1. 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

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

> numHeights.cumfreq=cumsum(numHeights.freq)

> cbind(numHeights.cumfreq)

numHeights.cumfreq

[150,155) 6

[155,160) 19

[160,165) 39

[165,170) 84

[170,175) 126

[175,180) 153

[180,185) 181

[185,190) 198

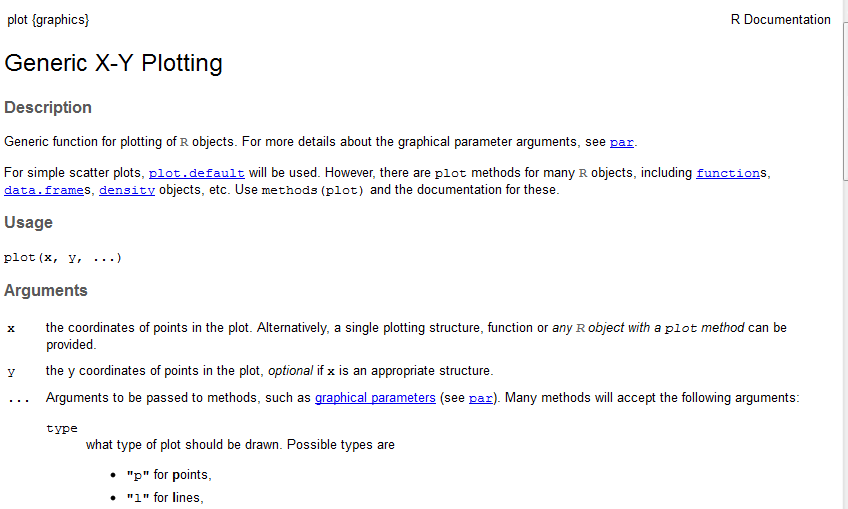
[190,195) 206

[195,200) 208

[200,205) 209

Here, we can see that 126 students are 175cm or shorter. We can verify this from the original frequency table.

# 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.

Note that the **plot** function reads in the ordered pairs as x and y coordinates, separately. The x-values in this case are the heights, which we have stored as **breaks**. The y-coordinates are our **numHeights.cumfreq** data.

However, if we display these two vectors, we will see that we have one more break than **cumfreq** value:

> plot(breaks, numHeights.cumfreq)

Error in xy.coords(x, y, xlabel, ylabel, log) :

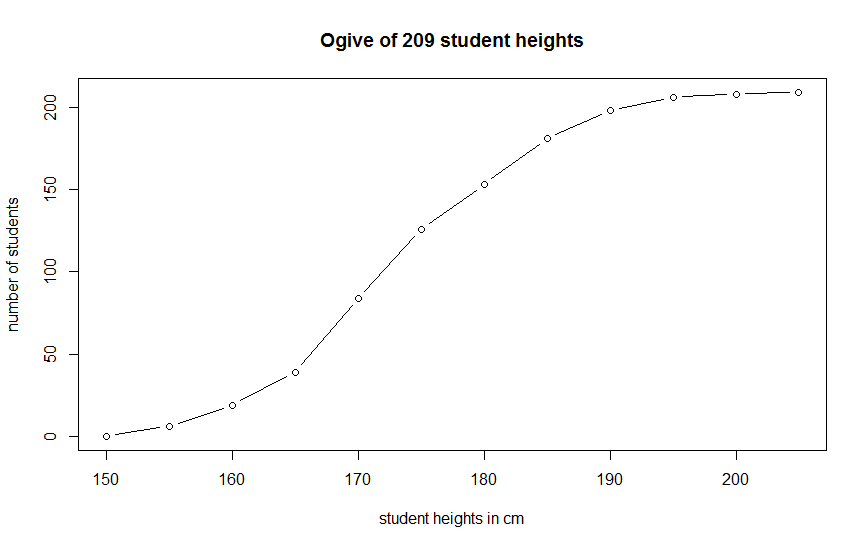
'x' and 'y' lengths differ

We want an (x,y) pair in our ogive to be of the form (height, number of students below that height). Note that for our data, we have no students who are less than 150 cm tall. Therefore we want the y-value corresponding to y=150 to be zero. So we can append a 0 to the beginning of our list of heights.

> freqs<-c(0, numHeights.cumfreq)

Now the values line up.

1. Read the help file for the **plot()** function and create the following ogive. Include the command you used.



# Scatterplots in R

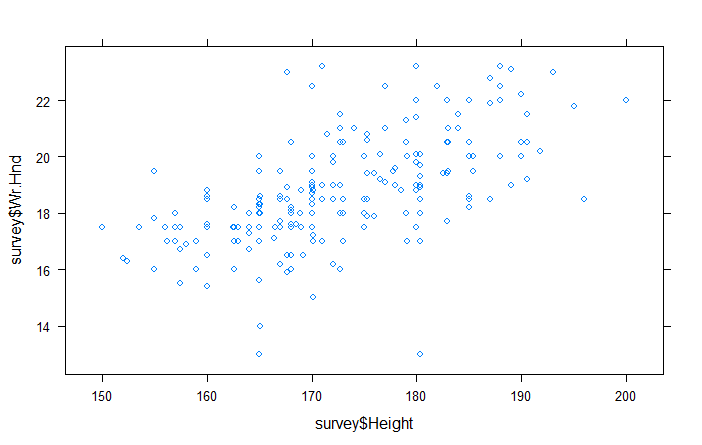
Sometimes we want to know how one variable is related to another. For example, how are height correlated. In general, we expect a tall person weigh more than a short person – but of course there are exceptions. We say there is a positive correlation between height and weight, but not a perfect correlation.

We will create a few scatterplots based on the **survey** data. First we will compare the **Height** variable, which gives student heights in centimeters to the **Wr.Hnd** variable, which gives the span of the student’s writing hand.

The most basic scatterplot is given by the command

> xyplot(survey$Wr.Hnd ~ survey$Height)

This produces the following plot



1. 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.
2. Create two new scatterplots: one that plots **Wr.Hnd** against **NW.Hnd**, and one that plots **Height** against **Pulse**. Both plots should have descriptive titles and axis labels. Include your plots along with the command(s) used to produce them, and answer the following questions: How do the graphs compare to one another, and to the **Wr.Hnd** vs **Height** graphs? What does this tell you about the correlation between each of the three variable pairs?