Skiing facts

Oh skiing during the holiday. Imagine the thrill of 0 degree weather on crisp clear days shushing down the hills at top speed. Imagine trying to learn how to snowboard and the bruised ribs that follow. Since I didn't want my ski break to end I've brought it into the classroom.

This project covers the notions of center, spread and position. These are measured with means, medians, variances, IQR's, percentiles and z-scores.

0.1 size of a data variable

The length of skis depends on several things: style of ski, ability of skier, age of ski (newer skis are generally shorter). A random sampling of skis on the hill gave these lengths

```
165 163 150 135 180 204 175 150 140 155
```

Enter the data into the variable `cms` as follows

```
> cms = c(165, 163, 150, 135, 180, 204, 175, 150, 140, 155)
```

First, how long is the data? This is usually denoted by n. We can find it with `length`

```
> length(cms)
[1] 10
> n = length(cms)
```

There are 10 numbers. The last line stores 10 into the variable `n`. Now we can refer to this instead of using `length(cms)`. This is convenient, but if you want to use a different variable, you'll have to do this again.

0.2 Problems

The time it takes to get down the hill varies alot depending on the abilities of the skier. Suppose, the following times were recorded for a random sample of skiers

```
3, 3, 5, 3, 7, 25, 65, 3, 4, 10, 15, 45, 4, 3
```

1. Enter in the data into a variable `time`
2. What is the length of the variable?

0.3 summation notation

The notation $\sum_i x_i$ can be confusing to learn. Using the computer, you just replace $\sum$ by `sum()`. For example to add the lengths we can do

```
> sum(cms)
[1] 1617
```

To find the mean, which is $\bar{x} = (1/n)\sum x_i$ we divide by `n`

```
> sum(cms)/n
[1] 161.7
```

We can check with `mean()`

```
> mean(cms)
[1] 161.7
```

So far so good. The median of the dataset can be found with the `median()` command.

```
> median(cms)
[1] 159
```

To do this by hand, we note that $10 = 2 \cdot 5$ so the median is the average of the fifth and sixth numbers after sorting.

```
> cms.sort = sort(cms)
> cms.sort
 [1] 135 140 150 150 155 163 165 175 180 204
> (155 + 163)/2
[1] 159
> (cms.sort[5] + cms.sort[6])/2
[1] 159
```

The last line shows how to do this using indices.

0.4 Problems

1. For the `time` variable, find the total sum. Compare the average found using this, to the output of `mean()`.
2. Find the median of `time`.
3. Draw by hand or make a stripchart using `stripchart(cms,method="stack")` of the `cms` variable. Identify on the graph the mean and median. Are they close by?

0.5 IQR

Both of the mean and median are output by the `summary()` function.

```
> summary(cms)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    135     150     159     162     173     204
```

This also gives the quartiles of the data set. The IQR is the difference between the third and first

```
> 173 - 150
[1] 23
> IQR(cms)                         # using function
[1] 22.5
```

Why the difference? Rounding in the output of summary. We find the quartiles directly with the `quantile()` function

```
> quantile(cms,.75)
  75%
172.5
> quantile(cms,.25)
25%
150
```

So the IQR is clearly 22.5.

The hinges are given in the book as the quantiles. These are output with the `fivenum()` function

```
> fivenum(cms)
[1] 135 150 159 175 204
```

0.6 Problems

1. The upper and lower hinges are the medians of the right and left half of the data. Show that the numbers above for `cms` are those.
2. The `quantile()` function finds percentiles on a scale of 0 to 1 (instead of 0 to 100%). Find the 80 percentile for `cms`.
3. What is the IQR of `time`?

0.7 Variance

The variance formula is

$$s^2 = \frac{1}{n-1}\sum_i (x_i - \bar{x})^2. \qquad \text{variance}$$

We can do this directly as follows. Notice we take it step by step to show the output of each command

```
> cms
 [1] 165 163 150 135 180 204 175 150 140 155
> cms - mean(cms)
 [1]   3.3   1.3 -11.7 -26.7  18.3  42.3  13.3 -11.7 -21.7  -6.7
> (cms - mean(cms))^2
 [1]   10.89    1.69  136.89  712.89  334.89 1789.29  176.89
 [8]  136.89  470.89   44.89
> sum( (cms - mean(cms))^2 )
[1] 3816
> sum( (cms - mean(cms))^2 )/ (n-1)
[1] 424
> var(cms)
[1] 424
```

Each command returns a new dataset of 10 numbers. So the `sum()` command adds up 10 numbers for you. Notice the work is already programmed in the `var()` command.

0.8 Problems

1. Find the standard deviation of `time`. (use `sd()`.) Compare to the IQR. Are they much different?
2. Find $\sum(x_i - \bar{x})$ for both the `time` and `cms` variables. Are the answers the same?

The quantile function answers “What data point is bigger than a certain percentage of the data”. The **percentile rank** attempt to answer the reverse: “What percent of the data is less than this data point.”

The book chooses to define this as

$$\text{percentile rank} = \frac{\text{no. less} + 1/2(\text{no. equal})}{n} \cdot 100\%.$$

It is easy to find the number less or the number less or equal using `sum()`. For example

```
> cms < 165
 [1] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE
> sum(cms < 165)
[1] 6
```

The first line is a question: how many are less than 165. The second line adds up all the `TRUE`'s. The number less or equal is

```
> sum(cms <= 165)
[1] 7
> sum(cms <= 165)/n * 100
[1] 70
```

So 70% are less than or equal 165.

To get the percentile rank annoyingly requires more typing

```
> cms == 165
 [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> sum(cms < 165) + (1/2) *sum(cms == 165)
[1] 6.5
> ( sum(cms < 165) + (1/2) *sum(cms == 165) ) /n * 100
[1] 65
```

So the percentile rank of 165 is 65%. Notice you use two equals signs `==` to ask about equality.

0.9 Problems

1. How many of `cms` are less than 150. Less than or equal? What is the percentile rank of 150?
2. For `time`, what is the percentile rank of 4?

0.10 *z*-scores

The *z*-score measures how far a data point is from the center of the data by a number of standard deviations. The definition of the *z*-score of x_i is

$$z_i = \frac{x_i - \bar{x}}{s}. \qquad z\text{-score}$$

For example, the *z*-score of 165 is

```
> (165 - mean(cms))/sd(cms)
[1] 0.1603
```

or 0.16. Whereas, the *z*-score of 204 is

```
> (204 - mean(cms))/sd(cms)
[1] 2.054
```

All the *z*-scores are found by the `scale()` command. These commands will do so and print the numbers you found the *z*-scores of.

```
> cbind(cms,scale(cms))
```

0.11 Problems

1. For `time()` what is the z-score of 15? 100?
2. The z-score of a number and its percentile rank are roughly linked. By Chebyshev's theorem, no more than $1/k^2$ can have a z-score bigger than k in absolute value. Verify this for $k = 2$ by counting the number of z-scores bigger than 2 in the variables `cms` and `time`.

Table contains information on ski resorts in New Hampshire. For these questions, answer the following by typing in the data and using the appropriate functions.

1. What is the average ticket price? What is the median price. Mark both on a strip chart. Is there a big difference?
2. How much variation in ticket price is there? What is bigger the IQR or the standard deviation?
3. What is the percentile rank of 49$? What is the z-score.
4. What is the minimum ticket price? Why don't most people ski there?
5. Compare the standard deviation and the IQR for the vertical fall. Are they similar? different?
6. What is the 80% percentile of vertical fall?
7. What is the percentile rank of 1000?
8. Make a stripchart of the number of lifts
9. Mark the mean and median amount on the stripchart for number of lifts. Are they similar in size?
10. Find the standard deviation and IQR for the number of lifts.

Table 1: NH ski facts

Where	vertical	no. lifts	price
skiway	968	3	34
sunapee	1510	9	49
Pats peak	710	8	40
Ragged Mtn	1250	9	45
Attitash	1750	12	49
Black Mtn.	1100	4	32
Cranmore	1200	9	32
King Pine	350	6	29
Wildcat	2112	4	52
Balsams	1000	4	30
Gunstock	1400	8	45
Bretton Wds	1500	8	53
Cannon Mtn	2146	7	42
Loon Mtn	2100	8	49
Waterville	2020	11	39