Topic 7o: Automating Values from Grouped Data

We are given a small frequency table of values in intervals.

interval	(15,22]	(22,29]	(29,36]	(36,43]	(43,50]	(50,57]	(57,64]	Total
count	15	46	54	25	45	74	16	275

We will go through the process outlined in the previous section. First, do this the long way.

```
     # get the midpoints and the frequencies
mid_p <- seq(15,57,7)+3.5
mid_p
freqs <- c(15,46,54, 25, 45, 74, 16)
freqs
     # find the total number of data values
num_x <- sum( freqs )
num_x
```

lines from the R script

Those saame lines and their output from the console

```
>      # first do the problem the long way
>      # get the midpoints and the frequencies
> mid_p <- seq(15,57,7)+3.5
> mid_p
[1] 18.5 25.5 32.5 39.5 46.5 53.5 60.5
> freqs <- c(15,46,54, 25, 45, 74, 16)
> freqs
[1] 15 46 54 25 45 74 16
>      # find the total number of data values
> num_x <- sum( freqs )
> num_x
[1] 275
```

Now find the mean.

```
     # get the sum of the midpoints times
     # the count (the value in freqs)
sum_x <- sum( mid_p * freqs)
sum_x
     # then the mean is sum_x/num_x
mu <- sum_x/num_x
mu
```

```
>      # get the sum of the midpoints times
>      # the count (the value in freqs)
> sum_x <- sum( mid_p * freqs)
> sum_x
[1] 11212.5
>      # then the mean is sum_x/num_x
> mu <- sum_x/num_x
> mu
[1] 40.77273
```

Then, for the standard deviation, using the alternative formula, we need to get the squares of the midpoint values. Then get the sum of the products of the squares of the midpoints and the frequencies, Then compute the alternative formula.

```
     # now to get the standard deviation
     # first create squares of the midpoints
mp_sq <- mid_p^2
mp_sq
     #  then find the sum of the products of
     #  the mp_sq and the counts
sum_xsq <- sum( mp_sq * freqs)
sum_xsq
     # finally use the alternative formula
     # to get the standard deviation
std_dev = sqrt( (sum_xsq - (sum_x^2)/num_x)/num_x)
std_dev
```

```
>      # now to get the standard deviation
>      # first create squares of the midpoints
> mp_sq <- mid_p^2
> mp_sq
[1]  342.25  650.25 1056.25 1560.25 2162.25 2862.25 3660.25
>      #  then find the sum of the products of
>      #  the mp_sq and the counts
> sum_xsq <- sum( mp_sq * freqs)
> sum_xsq
[1] 498760.8
>      # finally use the alternative formula
>      # to get the standard deviation
> std_dev = sqrt( (sum_xsq - (sum_x^2)/num_x)/num_x)
> std_dev
[1] 12.29879
```

Or, we could have used a shorter approach where we create, from the midpoint values and the frequencies, a variable that fits the original table. Then we get the mean and standard deviation from that variable.

```
32      # then there is the shorter way
33      # assuming that we know the midpoints and
34      # the frequencies  (lines 5-8 above )
35      # just make the data values
36 x <- rep( mid_p, freqs )
37 head(x, 19)
38 tail(x, 19)
39      # then just find the mean and
40      # standard deviation of x
41 mean( x )
42 source("../pop_sd.R")
43 pop_sd(x)
44      # and if this was a sample
45 sd(x)
```

```
>      # then there is the shorter way
>      # assuming that we know the midpoints and
>      # the frequencies  (lines 5-8 above )
>      # just make the data values
> x <- rep( mid_p, freqs )
> head(x, 19)
 [1] 18.5 18.5 18.5 18.5 18.5 18.5 18.5 18.5 18.5 18.5 18.5
[12] 18.5 18.5 18.5 18.5 25.5 25.5 25.5 25.5
> tail(x, 19)
 [1] 53.5 53.5 53.5 60.5 60.5 60.5 60.5 60.5 60.5 60.5 60.5
[12] 60.5 60.5 60.5 60.5 60.5 60.5 60.5 60.5
>      # then just find the mean and
>      # standard deviation of x
> mean( x )
[1] 40.77273
> source("../pop_sd.R")
> pop_sd(x)
[1] 12.29879
>      # and if this was a sample
> sd(x)
[1] 12.32121
```

Or we could take the much shorter way, use the function.

```
50 source("../get_from_table.R")
51 get_from_table( 15, 22, freqs)
```

```
> source("../get_from_table.R")
> get_from_table( 15, 22, freqs)
     size      mean        sd    pop sd
275.00000  40.77273  12.32121  12.29879
```