Problem Set 1

JP

8/14/2021

```
set.seed(082421)
# creating 5 different categories, with different amounts of time they show up in the data
categories = rep(letters[1:5], times = c(5, 4, 6, 8, 7))

# 30 observations with the mean and standard deviation
test_scores = rnorm(n = 30, mean = 7.14, sd = 2.16)

# Making the categories act like categories rather than numerical values
categories = as.factor(categories)

# Observations (be careful R is case sensitive)
N = 30
```

I will do this two ways

Using the numerical values and then using the objects approach

Frequencies

```
# get the frequencies of all the categories
table(categories)
```

```
## categories
## a b c d e
## 5 4 6 8 7
```

```
# Frequency & Percentage of A
5/30
```

```
## [1] 0.1666667
```

```
(5/30)*100
```

```
## [1] 16.66667
```

```r
# Frequency & Percentage of B
4/30
```

```
## [1] 0.1333333
```

```r
(4/30)*100
```

```
## [1] 13.33333
```

```r
# Frequency & Percentage of C
6/30
```

```
## [1] 0.2
```

```r
(6/30)*100
```

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

```r
# Frequency & Percentage of D
8/30
```

```
## [1] 0.2666667
```

```r
(8/30)*100
```

```
## [1] 26.66667
```

```r
# Frequency & Percentage of E
7/30
```

```
## [1] 0.2333333
```

```r
(7/30)*100
```

```
## [1] 23.33333
```

```r
# cumulative frequencies of categories A & B
(5/30) + (4/30)
```

```
## [1] 0.3
```

```r
# another option is to do the calculations above and then just add up the frequency of A with the frequ
.167 + .133
```

```
## [1] 0.3
```

```r
# To get the cumlative percentage of all 5 categories
(5/30)*100 + (4/30)*100 + (6/30)*100 + (8/30)*100 + (7/30)*100
```

```
## [1] 100
```

```r
# you can also use the calculations above to add up all the percentages
16.667 + 13.333 + 20 + 26.667 + 23.333
```

```
## [1] 100
```

Frequencies using objects

```r
freq_a <- 5/N
freq_a
```

```
## [1] 0.1666667
```

```r
# Percent of category A
percent_a <- freq_a*100
percent_a
```

```
## [1] 16.66667
```

```r
freq_b <- 4/N
freq_b
```

```
## [1] 0.1333333
```

```r
percent_b <- freq_b*100
percent_b
```

```
## [1] 13.33333
```

```r
freq_c <- 6/N
freq_c
```

```
## [1] 0.2
```

```r
percent_c <- freq_c*100
percent_c
```

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

```r
freq_d <- 8/N
freq_d
```

```
## [1] 0.2666667
```

```r
percent_d <- freq_d*100
percent_d
```

```
## [1] 26.66667
```

```r
freq_e <- 7/N
freq_e
```

```
## [1] 0.2333333
```

```r
percent_e <- freq_e*100
percent_e
```

```
## [1] 23.33333
```

```r
# cumulative frequency of A & B
freq_ab = freq_a + freq_b
freq_ab
```

```
## [1] 0.3
```

```r
# To get the cumlative percentage of all 5 categories
total_percent = percent_a + percent_b + percent_c + percent_d + percent_e
total_percent
```

```
## [1] 100
```

```r
# Just a way to combine all the percentages together to see in one place
cbind(percent_a, percent_b, percent_c, percent_d, percent_e)
```

```
##      percent_a percent_b percent_c percent_d percent_e
## [1,]  16.66667  13.33333        20  26.66667  23.33333
```

```r
# find the mode
table(categories)
```

```
## categories
## a b c d e
## 5 4 6 8 7
```

```r
# which value occurs most often
```

Getting the Median Two ways

```r
sort(test_scores)
```

```
##  [1]  1.979125  3.537262  3.566831  4.282318  4.396029  4.506466  4.977139
##  [8]  5.668250  5.895740  6.085677  6.436928  6.440507  6.589584  6.821120
## [15]  7.060384  7.212301  7.288768  7.816361  7.868345  7.869582  8.239301
## [22]  8.691209  8.845786  8.924377  9.383258  9.407202 10.668180 10.949238
## [29] 12.055448 12.588050
```

```
#look for the middle value or in this case the 15th and 16th observations
# 7.060384 & 7.212301 and then divide by 2
(7.06 + 7.21)/2
```

```
## [1] 7.135
```

```
# find the median
sort(test_scores)
```

```
##  [1]  1.979125  3.537262  3.566831  4.282318  4.396029  4.506466  4.977139
##  [8]  5.668250  5.895740  6.085677  6.436928  6.440507  6.589584  6.821120
## [15]  7.060384  7.212301  7.288768  7.816361  7.868345  7.869582  8.239301
## [22]  8.691209  8.845786  8.924377  9.383258  9.407202 10.668180 10.949238
## [29] 12.055448 12.588050
```

```
sort(test_scores)[15:16]
```

```
## [1] 7.060384 7.212301
```

```
get_median = (7.06 + 7.21)/2
get_median
```

```
## [1] 7.135
```

```
median(test_scores)
```

```
## [1] 7.136343
```

Getting the Mean Two Ways

```
# look at test scores first in order for lowest to highest
sort(test_scores)
```

```
##  [1]  1.979125  3.537262  3.566831  4.282318  4.396029  4.506466  4.977139
##  [8]  5.668250  5.895740  6.085677  6.436928  6.440507  6.589584  6.821120
## [15]  7.060384  7.212301  7.288768  7.816361  7.868345  7.869582  8.239301
## [22]  8.691209  8.845786  8.924377  9.383258  9.407202 10.668180 10.949238
## [29] 12.055448 12.588050
```

```
# Now let's get the sum
1.98 + 3.54 + 3.57 + 4.28 + 4.40 + 4.51 + 4.98 + 5.67 + 5.90 + 6.09 + 6.44 + 6.44 + 6.59 + 6.82 + 7.06 +
```

```
## [1] 216.09
```

```
# Now let's decide the sum of the scores by the number of observations
(1.98 + 3.54 + 3.57 + 4.28 + 4.40 + 4.51 + 4.98 + 5.67 + 5.90 + 6.09 + 6.44 + 6.44 + 6.59 + 6.82 + 7.06
```

```
## [1] 7.203
```

```
# 7.20 is the mean of our scores
```

```
# now getting the mean with objects
sort(test_scores)
```

```
##  [1]  1.979125  3.537262  3.566831  4.282318  4.396029  4.506466  4.977139
##  [8]  5.668250  5.895740  6.085677  6.436928  6.440507  6.589584  6.821120
## [15]  7.060384  7.212301  7.288768  7.816361  7.868345  7.869582  8.239301
## [22]  8.691209  8.845786  8.924377  9.383258  9.407202 10.668180 10.949238
## [29] 12.055448 12.588050
```

```
# look at how many observations we have
N
```

```
## [1] 30
```

```
# Now lets get the sum of the scores
sum_of_scores = sum(test_scores)
sum_of_scores
```

```
## [1] 216.0508
```

```
# get the mean
xbar = sum_of_scores/N
# could also look at it this way
xbar2 = sum_of_scores/30

xbar
```

```
## [1] 7.201692
```

```
xbar2
```

```
## [1] 7.201692
```

```
# we get the same mean
```

Now deviations

```
sort(test_scores)
```

```
##  [1]  1.979125  3.537262  3.566831  4.282318  4.396029  4.506466  4.977139
##  [8]  5.668250  5.895740  6.085677  6.436928  6.440507  6.589584  6.821120
## [15]  7.060384  7.212301  7.288768  7.816361  7.868345  7.869582  8.239301
## [22]  8.691209  8.845786  8.924377  9.383258  9.407202 10.668180 10.949238
## [29] 12.055448 12.588050
```

```
# look at how far students deviated from the average score by doing each one individually
1.98 - 7.20
```

```
## [1] -5.22
```

```
3.54 - 7.20
```

```
## [1] -3.66
```

```
3.57 - 7.20
```

```
## [1] -3.63
```

```
4.28 - 7.20
```

```
## [1] -2.92
```

```
4.40 - 7.20
```

```
## [1] -2.8
```

```
4.51 - 7.20
```

```
## [1] -2.69
```

```
4.98 - 7.20
```

```
## [1] -2.22
```

```
5.67 - 7.20
```

```
## [1] -1.53
```

```
5.90 - 7.20
```

```
## [1] -1.3
```

```
6.09 - 7.20
```

```
## [1] -1.11
```

```r
6.44 - 7.20
```

```
## [1] -0.76
```

```r
6.44 - 7.20
```

```
## [1] -0.76
```

```r
6.59 - 7.20
```

```
## [1] -0.61
```

```r
6.82 - 7.20
```

```
## [1] -0.38
```

```r
7.06 - 7.20
```

```
## [1] -0.14
```

```r
7.21 - 7.20
```

```
## [1] 0.01
```

```r
7.29 - 7.20
```

```
## [1] 0.09
```

```r
7.82 - 7.20
```

```
## [1] 0.62
```

```r
7.87 - 7.20
```

```
## [1] 0.67
```

```r
7.87 - 7.20
```

```
## [1] 0.67
```

```r
8.24 - 7.20
```

```
## [1] 1.04
```

```r
8.69 - 7.20
```

```
## [1] 1.49
```

```r
8.85 - 7.20
```

```
## [1] 1.65
```

```r
8.92 - 7.20
```

```
## [1] 1.72
```

```r
9.38 - 7.20
```

```
## [1] 2.18
```

```r
9.41 - 7.20
```

```
## [1] 2.21
```

```r
10.67 - 7.20
```

```
## [1] 3.47
```

```r
10.95 - 7.20
```

```
## [1] 3.75
```

```r
12.06 - 7.20
```

```
## [1] 4.86
```

```r
12.59 - 7.20
```

```
## [1] 5.39
```

```r
# now to get the sum of the deviations
(-5.22) + (-3.66) + (-3.63) + (-2.92) + (-2.8) + (-2.69) + (-2.22) + (-1.53) + (-1.3) + (-1.11) + (-.76)
```

```
## [1] 0.09
```

```r
# remember to look at the test scores
sort(test_scores)
```

```
##  [1]  1.979125  3.537262  3.566831  4.282318  4.396029  4.506466  4.977139
##  [8]  5.668250  5.895740  6.085677  6.436928  6.440507  6.589584  6.821120
## [15]  7.060384  7.212301  7.288768  7.816361  7.868345  7.869582  8.239301
## [22]  8.691209  8.845786  8.924377  9.383258  9.407202 10.668180 10.949238
## [29] 12.055448 12.588050
```

```r
# lets remember what the mean was
xbar
```

```
## [1] 7.201692
```

```r
# look at how far students deviated from the average score using the objects we created earlier
deviation = test_scores - xbar
deviation
```

```
##  [1] -2.80566349 -3.66442975 -0.61210859 -0.76476402 -5.22256690  0.01060902
##  [7]  5.38635782  0.61466928 -2.22455345 -0.76118503 -2.69522654  2.18156621
## [13]  1.48951665  0.66788945  2.20550951  1.03760910 -1.11601555  1.72268513
## [19]  1.64409340  3.74754569 -3.63486163 -0.38057177  4.85375559  0.08707613
## [25]  3.46648728  0.66665299 -0.14130830 -2.91937412 -1.30595174 -1.53344238
```

```r
# look to see how far the worse test score was from the mean
sort(deviation)
```

```
##  [1] -5.22256690 -3.66442975 -3.63486163 -2.91937412 -2.80566349 -2.69522654
##  [7] -2.22455345 -1.53344238 -1.30595174 -1.11601555 -0.76476402 -0.76118503
## [13] -0.61210859 -0.38057177 -0.14130830  0.01060902  0.08707613  0.61466928
## [19]  0.66665299  0.66788945  1.03760910  1.48951665  1.64409340  1.72268513
## [25]  2.18156621  2.20550951  3.46648728  3.74754569  4.85375559  5.38635782
```

```r
# getting the sum of the deviations
sum(deviation)
```

```
## [1] 5.77316e-15
```