R Markdown

`hw1.Rmd` is an R Markdown document (a modification of the default one created by File -> New File -> R Markdown…). Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. This document is configured to produce a PDF. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both text (such as these paragraphs) as well as the output of any embedded R code "chunks" within the document. (You may be asked to install some packages to enable this, and/or extra software if you are working on a desktop or laptop. See here if you think that's the case.)

R code chunks are indicated like this:

```r
nums <- c(1, 2, 3, 4)
print(nums)
```

```
## [1] 1 2 3 4
```

When knitted, the output of the lines appears right after the chunk.

Editing and Submitting

To work on this assignment, simply edit this file and save it as you go. If you're familiar with git, feel free to utilize its features. If not, don't worry - the only thing to know is that to sync down the latest assignments, click the gray-green-and-red "Git" icon in the toolbar, and select "pull branches."

To submit this assignment, click the "knit" button, download the PDF that is generated, and submit it to Canvas.

Before you knit, be sure to clean up any debug code or extraneous printouts, though you of course may add explanatory comments (in fact you should, if you think a bit of code is confusing).

Problem 1

The function `rexp()` generates a numeric vector of a specified length, filled with random numbers from an exponential distribution. To generate a length-200 vector from an exponential distribution with "rate parameter" 1.5 we can use `rexp(200, rate = 1.5)`.

Since this is a homework we want repeatable randomness; to make the something like `rexp()` produce the same set every time we can run `set.seed()` to "seed" the random number generator to a specific value (an arbitrary numeric works–we're going to go with 42).

NOTE: This update adds the call to `RNGkind("Super")` to force the random number generator to use a particular method to avoid this: https://community.rstudio.com/t/getting-different-results-with-set-seed/31624/5

```
RNGkind("Super")
set.seed(42)
sample <- rexp(200, rate = 1.5)
print(sample[1:24])               # just showing first 24 to save space
```

```
##  [1] 0.36383964 0.45126898 0.49182558 0.84342096 0.86167556 0.55811506
##  [7] 0.21151951 0.48586652 0.40264938 0.48044846 0.05148179 0.07481878
## [13] 0.86044553 1.13469110 0.15932612 0.51576468 0.23620008 0.15790444
## [19] 0.11943451 1.19480448 1.45703598 0.72478392 1.72425664 0.05783676
```

Since this is an exponential distribution, the median of the sample (`median()`) will be less than the mean (`mean()`).

Your task: Add lines to the above code chunk so that it also *prints out the number of entries* greater than or equal to the median, and less than or equal to the mean. To help, the output should be `[1] 32`.

Problem 2

R comes with a number of built-in datasets; one of my favorites is `letters`, which is just a vector of the 26 lower-case letters. Let's suppose these are scrabble tiles, and generate a set of 26 random score points between 1 and 8. (`sample()` produces a vector of elements sampled from another; here we're sample from the vector `1:8` to get 26 outputs, with replacement during the sampling process).

```
print(letters)
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q"
## [18] "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

```r
# set seed just before producing any random samples
RNGkind("Super")
set.seed(42)
scores <- sample(1:8, size = 26, replace = TRUE)
print(scores)
```

```
##  [1] 7 7 3 4 4 3 6 3 7 3 5 8 2 4 4 2 5 4 5 1 6 5 8 3 6 3
```

Your task: Add lines to the chunk above so that it *also prints out the set of letters whose scores are greater than 5*. It should be a vector of length 8, starting `[1] "a" "b" "g" ...`