STAT 33B Homework 1

ZIANG GAO (3033669851)

This assignment is due **Feb 5, 2020** by 11:59pm.

The purpose of this assignment is to help you get comfortable with writing R code and using R Markdown documents.

R Markdown Documents

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. 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 content as well as the output of any embedded R code chunks within the document. You can choose to knit your document to an HTML, Word, or PDF file.

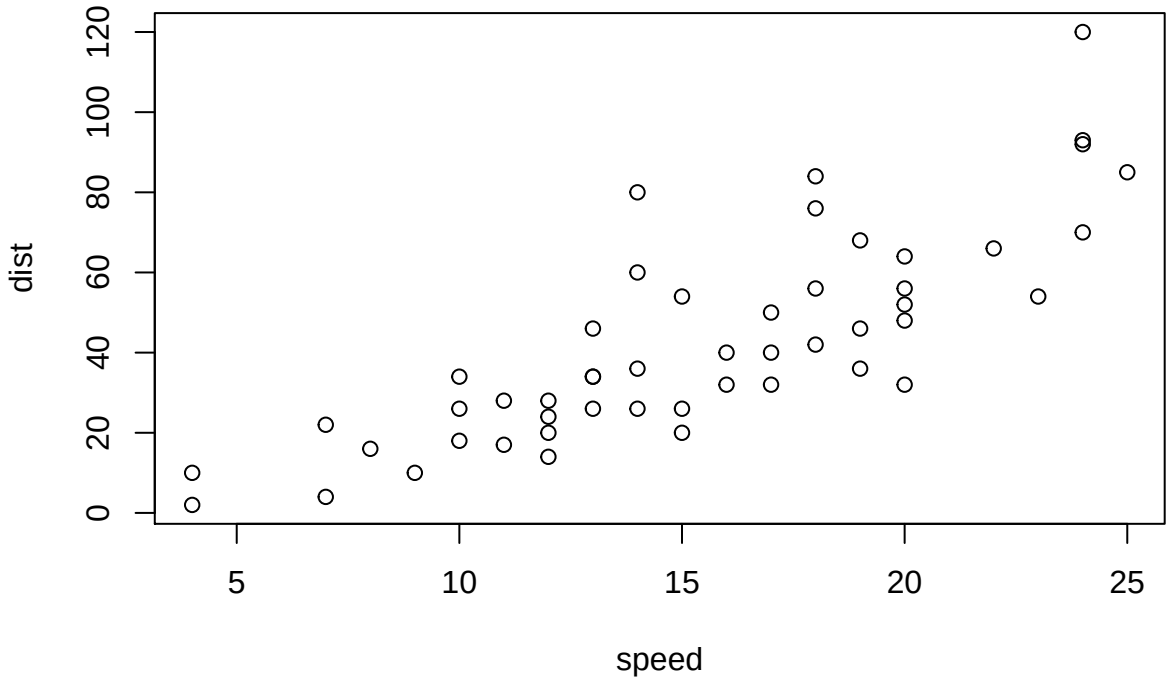
In order to knit to PDF, if you do not already have LaTex on your computer, you'll need to install the `tinytex` package in R. You can do this in R with the command `install.packages("tinytex")`. If you need help with this, come to office hours.

You can embed an R code chunk in the document like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Submission Instructions

Edit this file, knit to PDF, and:

- Submit the Rmd file on bCourses.
- Submit the PDF file on Gradescope.

If you think you'll need help with submission, please ask in office hours *before* the assignment is due.

Answer all questions with complete sentences, and put code in code chunks. You can make as many new code chunks as you like. Please do not delete the exercises already in this notebook, because it may interfere with our grading tools.

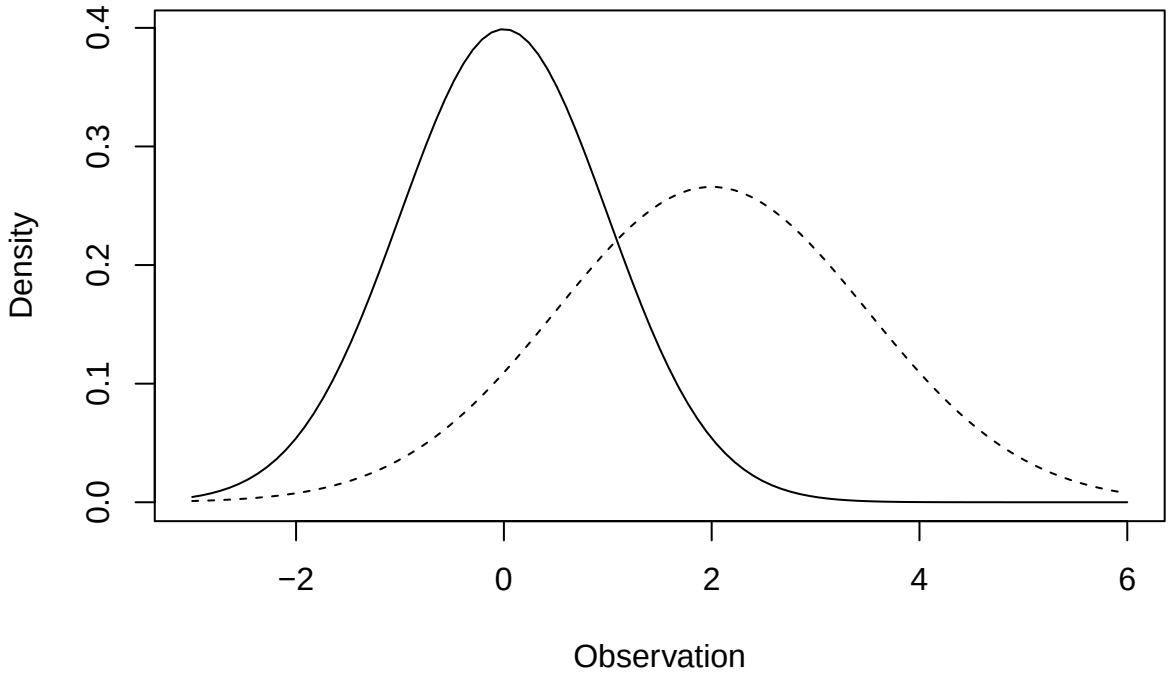
The Gaussian Distribution

The Gaussian or "normal" distribution is a bell-shaped probability distribution that plays an important role in statistics, as well as the natural and social sciences. The distribution is important because under weak conditions, the mean of a sample tends to be normally distributed as the size of the sample increases.

The family of normal distributions all have the same shape, and each is uniquely specified by its center (mean) and spread (standard deviation).

The "standard normal" distribution has center 0 and spread 1. This distribution is drawn in the figure below, along with a normal curve that has center 2 and spread 1.5 (dashed).

```
curve(dnorm, from = -3, to = 6, xlab = "Observation", ylab = "Density")
curve(dnorm(x, mean = 2, sd = 1.5), add = TRUE, lty = 2)
```



1. Read the documentation for `rnorm()` to figure out how to call `rnorm()` to generate 5 random values from a normal distribution with mean 10 and standard deviation 2.

```
rnorm(5, 10, 2)
```

```
## [1]  8.386456 13.945306  9.056453 12.683394 11.348721
```

2. Create a variable called `samp` that contains 1000 random values from a normal distribution with mean 1 and standard deviation 3.

```
# This problem uses R's pseudo-random number generator. The numbers generated
# are not truly random, but close enough for most statistical purposes.
#
# We can set the "seed" of the pseudo-random number generator to make the
# numbers it produces deterministic. This is allows others to run your code and
# get the same result you got.
#
# Do not modify the call to set.seed()
set.seed(33)

# Your code goes here
samp = rnorm(1000, 1, 3)
```

3. The 10% trimmed mean is obtained by taking the mean of the middle 80% of the values (removing the lowest 10% and the highest 10% of the values). Find the 10% trimmed mean of `samp`, and assign it to the variable `mean10`.

   Hint: Look at the help file for the `mean()` function.

```
# Your code goes here
mean10 = mean(samp, 0.1)
```

4. Use `sum()` to calculate the fraction of values in `samp` that are greater than 2.

   Write your code so that it coerces a logical vector to numeric vector automatically (that is, without using the function `as.numeric()`).

Write your code generically so that it doesn't depend on `samp` being a vector of 1000 elements. Hint: consider using the `length()` function.

Assign the result to the variable `above2`.

```
# Your code goes here
num = sum(samp > 2)
above2 = num / length(samp)
above2
```

```
## [1] 0.367
```

5. A percentile is a value at which a given percentage of samples fall at or below. For example, the 90th percentile is the value at which 90% of the samples fall at or below. Percentiles are sometimes also called quantiles.

   Calculate the 95th percentile of `samps` and assign this value to the variable `per95`.

   Hint: The `quantile()` function is helpful for this question.

```
# Your code goes here
per95 = quantile(samp, probs = 0.95)
per95
```

```
##      95%
## 6.059647
```

6. Calculate the greatest absolute distance from 1 for the values in `samps`. Assign this value to a variable called `max_dev`, short for "maximum deviation" from the mean. Hint: The functions `abs()` and `max()` are helpful for this question.

```
# Your code goes here
dist = abs(samp - 1)
max_dev = max(dist)
max_dev
```

```
## [1] 10.43959
```