MTH 3220 Lab 1

Due Thu., Sept. 5

1 Part A: One-Sample t Confidence Interval for μ

1.1 Exhaust Hydrocarbons Data Set

The table below shows data on the amounts (in g/mi) of **hydrocarbon (HC) emissions** in the exhaust of $\boldsymbol{n = 46}$ randomly selected vehicles of the same type, measured under standard conditions prescribed by the U.S. Environmental Protection Agency.

0.50,	0.65,	0.46,	0.41,	0.41,	0.39,	0.44,	0.55,	0.72,	0.64,	0.83,	0.38,	0.38,	0.50,
0.60,	0.73,	0.83,	0.57,	0.34,	0.41,	0.37,	1.02,	0.87,	1.10,	0.65,	0.43,	0.48,	0.41,
0.51,	0.41,	0.47,	0.52,	0.56,	0.70,	0.51,	0.52,	0.51,	0.52,	0.57,	0.51,	0.36,	0.48,
0.52,	0.61,	0.58,	0.46,	0.47,	0.55								

1. Use `c()` and `<-` to create a vector containing the **HC emissions** data.

2. The function `t.test()` takes as its main argument a vector `x` and computes a ***95% one-sample t confidence interval*** (and carries out a *one-sample t test*) for a population mean μ. But `t.test()` accepts other optional arguments too. Among its arguments are:

`x`	a data vector.
`y`	another optional data vector.
`alternative`	the direction for the alternative hypothesis, one of `"two.sided"`, `"less"`, or `"greater"`.
`mu`	the null hypothesized value for the unknown population mean (or difference in means if you are performing a two-sample test), with default value 0.
`conf.level`	the confidence level for a confidence interval for the unknown population mean (or difference in means for a two-sample problem), with default value 0.95.

Use `t.test()` to compute a **95% CI** for the true (unknown) population mean HC emission μ, for example by typing (assuming you named your data vector `hc`):

```
t.test(x = hc, conf.level = 0.95)
```

2 Part B: Confidence Interval for a Population Proportion p

2.1 Political Poll Results

If a random sample is from a population of *successes* and *failures*, then a **$(1-\alpha)100\%$ *confidence interval*** for the population **proportion** of successes p is

$$\hat{P} \pm z_{\alpha/2}\sqrt{\frac{\hat{P}(1-\hat{P})}{n}},$$

where $z_{\alpha/2}$ is the **$(1-\alpha/2)100$*th percentile*** the N$(0,1)$ distribution.

(A slightly more accurate version of the CI, called the ***score CI***, is given in the textbook.)

A November 18, 2015, online CNN article had the headline "Poll: Clinton trails GOP rivals in Colorado."

The headline referred to a Quinnipiac poll which found that "Dr. Ben Carson tops Democratic front-runner Hillary Clinton **52 - 38** percent in a general election matchup."

Referring to its survey methodology, Quinnipiac states "This RDD (Random Digital Dialing) telephone survey was conducted from November 11 - 15, 2015 throughout the state of Colorado. Responses are reported for **1,262** self-identified registered voters."

1. The function `prop.test()` takes arguments

`x`	the number of successes in the sample.
`n`	the sample size.
`conf.level`	the confidence level for a confidence interval for the unknown population proportion, with default value 0.95.

   It returns the ***sample proportion $\hat{P}$*** along with a ***95% confidence interval for p***.

   Assume that **656** (52%) of the **1,262** respondents support Carson, and use `prop.test()` to compute a ***95% CI*** for the true (unknown) population **proportion p** that supports him.

3 Part C: One-Sample t Test

Between 1948 and 1958, the U.S. conducted 43 nuclear tests at Enewetak Atoll in the Pacific Ocean. Cleanup and restoration of the site for the return of the atoll to the Enewetak people began in 1977 and lasted to 1980. Concern has been expressed by the people of Enewetak and others over the possible aquatic impacts from the radioactive contaminants.

The following data are measured values of the radioactive contaminant radiocesium (^{137}Cs, in pCi/L) in 10 water specimens taken from the site in 1981.

69, 77, 227, 215, 28, 153, 25, 325, 55, 42

In this problem, we'll carry out a one-sample t test to decide if the data provide statistically significant evidence that the water's **population mean radiocesium** concentration is less than 215 pCi/L.

The hypotheses are

$$\begin{aligned} H_0 : \mu &\geq 215 \\ H_a : \mu &< 215 \end{aligned}$$

1. Use `c()` and `<-` to create a vector containing the **radiocesium** data.

2. The function `t.test()` takes as its main argument a vector `x` and carries out a ***one-sample t test*** for a population mean μ and reports the **test statistic** and **p-value**.

   We specify the null-hypothesized value μ_0 (**215** in our case) via the argument `mu`. By default, `t.test()` carries out a **two-tailed** test. To carry out a **one-tailed test**, we specify `"less"` or `"greater"` for the optional argument `alternative`. (In this case, the reported CI is one-sided.)

   Use `t.test()` to test the hypotheses stated above. For example, if you named your **radiocesium** data set `cs`, you'd type:

   ```
   t.test(x = cs, mu = 215, alternative = "less")
   ```

3. The one-sample t test rests on the assumption that either the sample was drawn from a **normal population** or ***n*** **is large**. If neither condition is met, the p-value from the test might be wrong.

   The **normality assumption** is considered to be met if

   - A ***histogram*** of the data is roughly **bell-shaped**.
   - Points in a ***normal probability plot*** hug a **straight line**.

   The function `hist()` will produce a histogram. Among its arguments are:

`x`	a data vector.
`col`	a color used to fill the histogram bars.
`xlab`	a label for the x-axis.
`ylab`	a label for the y-axis.
`main`	a main title.

   The function `qqnorm()` will produce a normal probability plot. Among its arguments are:

`y`	a data vector.
`pch`	an (optional) plot character type.

   The `qqline()` function will add a line to the plot. Among its arguments are:

`y`	a data vector.
`col`	a color used for the line.
`lwd`	the line width.

Now **check the normality assumption** by making a **histogram** and a **normal probability plot** of the **radiocesium** data. For example, if you named your **radiocesium** data set `cs`, you could type:

```
hist(x = cs, col = "blue", main = "Histogram of Radiocesium",
       xlab = "Radiocesium")
```

For example, you could type:

```
qqnorm(y = cs, pch = 19)
qqline(y = cs, col = "blue", lwd = 2)
```

(The argument `pch = 19` produces points that are solid circles, `col = "blue"` produces a blue line, and `lwd = 2` uses a line width twice as big as the default width).

4 Part D: Two-Sample *t* Test and Confidence Interval

4.1 Fruits and Vegetables Data Set

A random sample of **fruits** and another of **vegetables** were selected and the **moisture** contents (by percent) measured in each piece of food. The data are below.

Fruits		**Vegetables**	
	Moisture		Moisture
Apricot	86	Artichoke	85
Banana	75	BambooShoots	91
Avocado	72	Beets	88
Blackberry	88	Broccoli	89
Clementine	87	Cucumber	95
Fig	79	IcebergLettuce	96
PinkGrapefruit	92	Mushroom	92
Mango	84	Radish	95
		Tomato	94

Here are the data in a more convenient format:

Fruits	86,	75,	72,	88,	87,	79,	92,	84	
Vegetables	85,	91,	88,	89,	95,	96,	92,	95,	94

1. Use `c()` and `<-` to create two vectors containing the fruit and vegetable moisture contents.

2. The `boxplot()` function will produce **boxplots**. Among its arguments are:

`x, ...`	a numeric vector for specifying data from which the boxplots are to be produced, and (optionally) additional data vectors to be plotted.
`names`	group labels which will be printed under each boxplot. Can be a character vector.
`col`	colors to be used to color the bodies of the box plots.

Use `boxplot()` to make side-by-side boxplots of the two samples (but ***don't print*** the boxplots), for example by typing:

```r
boxplot(Fruits, Veggies, names = c("Fruits", "Vegetables"), col = "lightblue")
```

(assuming you named your vectors `Fruits` and `Veggies`).

3. We want to to decide **which type of food**, if any, has the **higher** mean **moisture content** by performing a ***two-sample t test*** of the hypotheses

$$H_0 : \mu_1 - \mu_2 = 0$$
$$H_a : \mu_1 - \mu_2 \neq 0$$

where $\boldsymbol{\mu_1}$ is the population mean moisture content for **fruits** and $\boldsymbol{\mu_2}$ is the population mean for **vegetables**.

If the two samples were selected **independently** of each other, the appropriate test is the ***two-sample t test***, with **test statistic**

$$t \;=\; \frac{\bar{X} - \bar{Y} - 0}{\sqrt{S_1^2/m + S_2^2/n}}$$

and **p-value** from the tails of the ***t*** **distribution** with **df**

$$\nu \;=\; \frac{\left(\frac{s_1^2}{m} + \frac{s_2^2}{n}\right)^2}{\frac{(s_1^2/m)^2}{m-1} + \frac{(s_2^2/n)^2}{n-1}} .$$

The function `t.test()` will carry out a ***two-sample t test***.

Use `t.test()` to carry out the test, for example by typing:

```r
t.test(x = Fruits, y = Veggies, mu = 0, alternative = "two.sided")
```

4. The function `t.test()` also computes the ***two-sample*** t ***confidence interval***

$$\bar{X} - \bar{Y} \pm t_{\alpha/2,\nu}\sqrt{\frac{S_1^2}{m} + \frac{S_2^2}{n}}$$

for $\boldsymbol{\mu_1 - \mu_2}$. The optional argument `conf.level` is used to specify the level of confidence. It has a default value of `0.95`.

Use `t.test()` to compute the **95% CI** for $\boldsymbol{\mu_1 - \mu_2}$.

5. The t test and CI require an assumption that the two samples came from **normal** distributions.

Use `hist()` to look at **histograms** of the two samples (but ***don't print*** the histograms). Are you concerned that the **normality assumption** may not be met for these two samples?