Code

R Problem Set

STATS250 @ Suffolk University

Your Name Here

1 Directions

The problem set is worth **100** points.

Enter your answers in the empty code chunks.

Don't change anything in the chunk below, and make sure you run it before attempting any of the problems:

Hide

```
library(tidyverse)
library(ggpubr)
set.seed(2018) # Belgium win 3rd place in the World Cup
```

2 Basics

Calculate $\frac{(2+2)\times(3^2+5)}{(6/4)}$:

Hide

```
# your code here
```

Create a vector called “x” with the following values: 10, 15, 18, 20.

Hide

```
# your code here
```

Calculate the mean (`mean()`), median (`median()`) and standard deviation (`sd()`) of `x` .

Hide

```
# your code here
```

Draw two numbers from a standard normal distribution (`rnorm()`):

Hide

```
# your code here
```

Draw 100 numbers from a normal distribution with a mean of **7** and standard deviation of **4**, and store the output in an object called “x1”:

Hide

```
# your code here
```

Draw 100 numbers from a normal distribution with a mean of **5** and standard deviation of **2**, and store the output in an object called “x2”:

Hide

```
# your code here
```

Use `data.frame()` to combine `x1` and `x2` into a data frame called "df":

Hide

```
# your code here
```

Use the `$` indexing method to calculate:

- the mean of the first column of `df` :

Hide

```
# your code here
```

- the standard deviation of the second column of `df` :

Hide

```
# your code here
```

Use `head()` to print the first **3** rows of `df` :

Hide

```
# your code here
```

3 `dplyr`

Let's work with the data set `diamonds` :

Hide

```
data(diamonds)
head(diamonds)
```

Calculate the average price of a diamond:

Hide

```
# your code here
```

Use `group_by()` to group diamonds by **color**, then use `summarise()` to calculate the average price *and* the standard deviation in price **by color**:

Hide

```
# your code here
```

Use `group_by()` to group diamonds by **cut**, then use `summarise()` to count the number of observations **by cut**:

Hide

```
# your code here
```

Use `filter()` to remove observations with a depth greater than 62, then use `group_by()` to group diamonds by **clarity**, then use `summarise()` to find the maximum price of a diamond **by clarity**:

Hide

```
# your code here
```

Use `mutate()` and `log()` to add a new variable to the data called "log_price":

Hide

```
# your code here
```

4 ggplot2

Continue using `diamonds`.

Use `geom_histogram()` to plot a histogram of prices:

Hide

```
# your code here
```

Use `geom_density()` to plot the density of *log prices* (the variable you added to the data frame):

Hide

```
# your code here
```

Use `geom_point()` to plot carats against log prices (i.e. carats on the x-axis, log prices on the y-axis):

Hide

```
# your code here
```

Use `stat_summary()` to make a bar plot of **average** cut:

Same as above but change the theme to `theme_classic()`:

Hide

```
# your code here
```

Finally,

- create a bar plot for **average** color and assign it to the object "plot_color";
- create a scatter plot for depth against log pricse (depth on x-axis, log prices on y-axis) and assign it to the object "plot_depth";
- use `ggarrange` from `ggpubr` to combine `plot_color` and `plot_depth` into a single plot with automatic labels.

5 Inference

Use `t.test()` to test the following hypothesis on *log price*:

$$H_0 : \mu = 8$$
$$H_A : \mu \neq 8$$

Hide

```
# your code here
```

Use `lm()` to estimate the model

$$log(\text{price}) = \beta_0 + \beta_1 \text{carat} + \beta_2 \text{table} + \varepsilon$$

and store the output in an object called “m1”:

Hide

```
# your code here
```

Use `summary()` to view the output of “m1”:

Hide

```
# your code here
```

Use `lm()` to estimate the model

$$log(\text{price}) = \beta_0 + \beta_1 \text{carat} + \beta_2 \text{table} + \beta_3 \text{depth} + \varepsilon$$

and store the output in an object called “m2”:

Hide

```
# your code here
```

Use `summary()` to view the output of “m2”:

Hide

```
# your code here
```