Homework 2

Marc Hughes

Table of contents

Link to the Github repository

! Due: Tue, Feb 14, 2023 @ 11:59pm

Please read the instructions carefully before submitting your assignment.

1. This assignment requires you to only upload a `PDF` file on Canvas
2. Don't collapse any code cells before submitting.
3. Remember to make sure all your code output is rendered properly before uploading your submission.

Please add your name to the author information in the frontmatter before submitting your assignment

For this assignment, we will be using the Abalone dataset from the UCI Machine Learning Repository. The dataset consists of physical measurements of abalone (a type of marine snail) and includes information on the age, sex, and size of the abalone.

We will be using the following libraries:

```
library(readr)
library(tidyr)
```

```
Warning: package 'tidyr' was built under R version 4.2.2
```

```
library(ggplot2)
library(dplyr)
```

```
Warning: package 'dplyr' was built under R version 4.2.2

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(purrr)
```

```
Warning: package 'purrr' was built under R version 4.2.2
```

```
library(cowplot)
```

```
Warning: package 'cowplot' was built under R version 4.2.2
```

Question 1

30 points

EDA using `readr`, `tidyr` and `ggplot2`

1.1 (5 points)

Load the "Abalone" dataset as a tibble called `abalone` using the URL provided below. The `abalone_col_names` variable contains a vector of the column names for this dataset (to be consistent with the R naming pattern). Make sure you read the dataset with the provided column names.

```r
library(readr)
url <- "http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data"

abalone_col_names <- c(
  "sex",
  "length",
  "diameter",
  "height",
  "whole_weight",
  "shucked_weight",
  "viscera_weight",
  "shell_weight",
  "rings"
)

abalone <- read_csv(url, col_names = abalone_col_names)
```

```
`curl` package not installed, falling back to using `url()`
Rows: 4177 Columns: 9
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): sex
dbl (8): length, diameter, height, whole_weight, shucked_weight, viscera_wei...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

1.2 (5 points)

Remove missing values and NAs from the dataset and store the cleaned data in a tibble called df. How many rows were dropped?

```r
abalone
```

```
# A tibble: 4,177 x 9
   sex   length diameter height whole_weight shucked_wei~1 visce~2 shell~3 rings
   <chr>  <dbl>    <dbl>  <dbl>        <dbl>         <dbl>   <dbl>   <dbl> <dbl>
 1 M      0.455    0.365  0.095        0.514        0.224   0.101    0.15     15
 2 M      0.35     0.265  0.09         0.226        0.0995  0.0485   0.07      7
```

```
 3 F      0.53     0.42   0.135        0.677        0.256   0.142    0.21      9
 4 M      0.44     0.365  0.125        0.516        0.216   0.114    0.155    10
 5 I      0.33     0.255  0.08         0.205        0.0895  0.0395   0.055     7
 6 I      0.425    0.3    0.095        0.352        0.141   0.0775   0.12      8
 7 F      0.53     0.415  0.15         0.778        0.237   0.142    0.33     20
 8 F      0.545    0.425  0.125        0.768        0.294   0.150    0.26     16
 9 M      0.475    0.37   0.125        0.509        0.216   0.112    0.165     9
10 F      0.55     0.44   0.15         0.894        0.314   0.151    0.32     19
# ... with 4,167 more rows, and abbreviated variable names 1: shucked_weight,
#   2: viscera_weight, 3: shell_weight
```

```
df <- drop_na(abalone)
df
```

```
# A tibble: 4,177 x 9
   sex   length diameter height whole_weight shucked_wei~1 visce~2 shell~3 rings
   <chr>  <dbl>    <dbl>  <dbl>        <dbl>         <dbl>   <dbl>   <dbl> <dbl>
 1 M      0.455    0.365  0.095        0.514        0.224   0.101    0.15     15
 2 M      0.35     0.265  0.09         0.226        0.0995  0.0485   0.07      7
 3 F      0.53     0.42   0.135        0.677        0.256   0.142    0.21      9
 4 M      0.44     0.365  0.125        0.516        0.216   0.114    0.155    10
 5 I      0.33     0.255  0.08         0.205        0.0895  0.0395   0.055     7
 6 I      0.425    0.3    0.095        0.352        0.141   0.0775   0.12      8
 7 F      0.53     0.415  0.15         0.778        0.237   0.142    0.33     20
 8 F      0.545    0.425  0.125        0.768        0.294   0.150    0.26     16
 9 M      0.475    0.37   0.125        0.509        0.216   0.112    0.165     9
10 F      0.55     0.44   0.15         0.894        0.314   0.151    0.32     19
# ... with 4,167 more rows, and abbreviated variable names 1: shucked_weight,
#   2: viscera_weight, 3: shell_weight
```

No rows were dropped from the dataset.

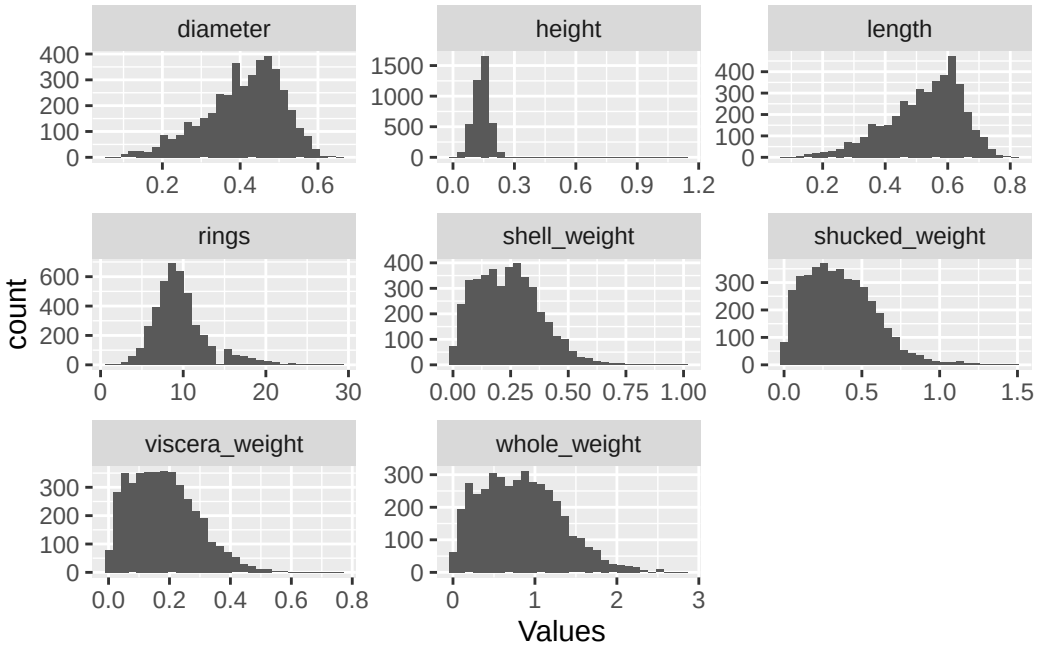
1.3 (5 points)

Plot histograms of all the quantitative variables in a **single plot** [1]

[1] You can use the `facet_wrap()` function for this. Have a look at its documentation using the help console in R

```r
new_df <-
  df %>%
  # using gather function to gather all variables into one column
  pivot_longer(!sex,
               names_to = "Variables",
               values_to = "Values")

plt <-
  ggplot(new_df, aes(x = Values)) +
  geom_histogram() +
  facet_wrap(~Variables, scales="free")
plt
```

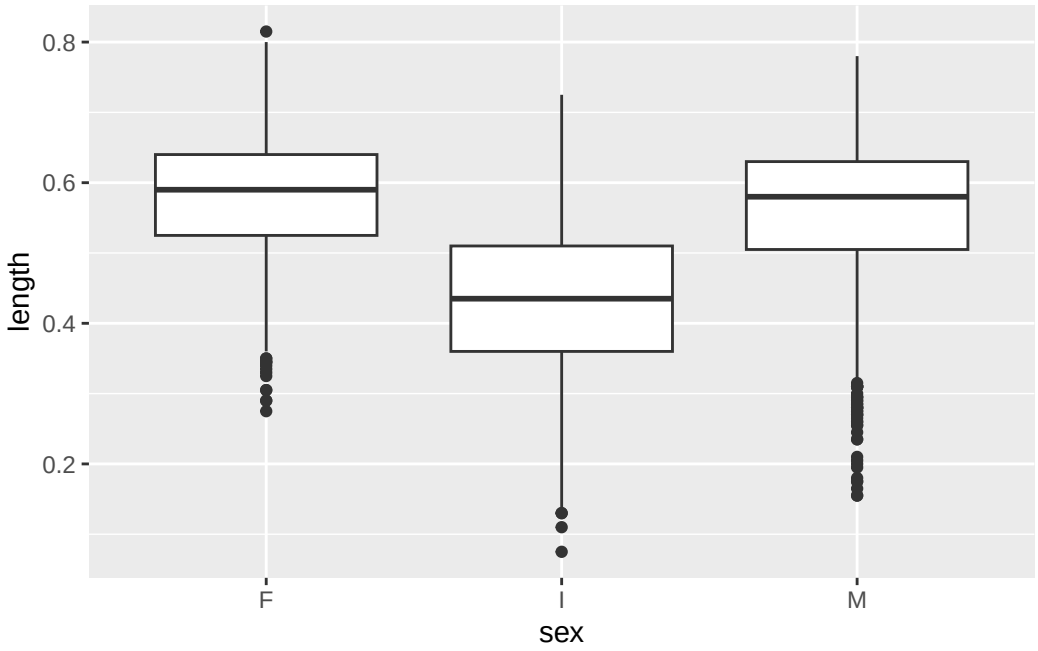
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



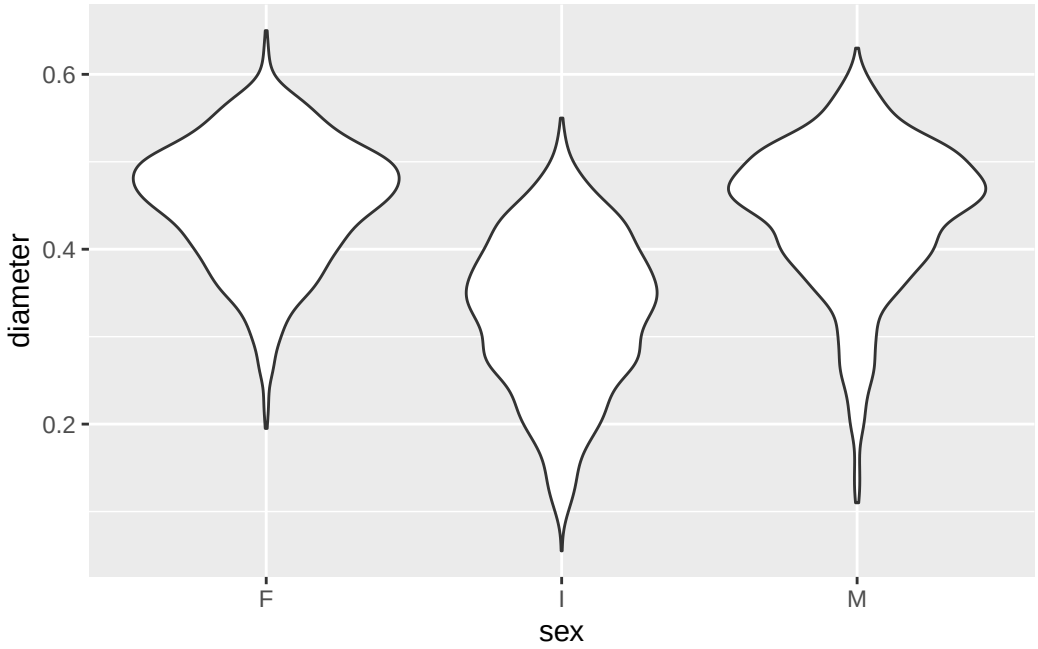
1.4 (5 points)

Create a boxplot of `length` for each `sex` and create a violin-plot of of `diameter` for each `sex`. Are there any notable differences in the physical appearences of abalones based on your analysis here?

```
box_plt <-
  ggplot(df, aes(x=sex, y=length)) +
  geom_boxplot()
box_plt
```




```
violin_plt <-
  ggplot(df, aes(x=sex, y=diameter)) +
  geom_violin()
violin_plt
```

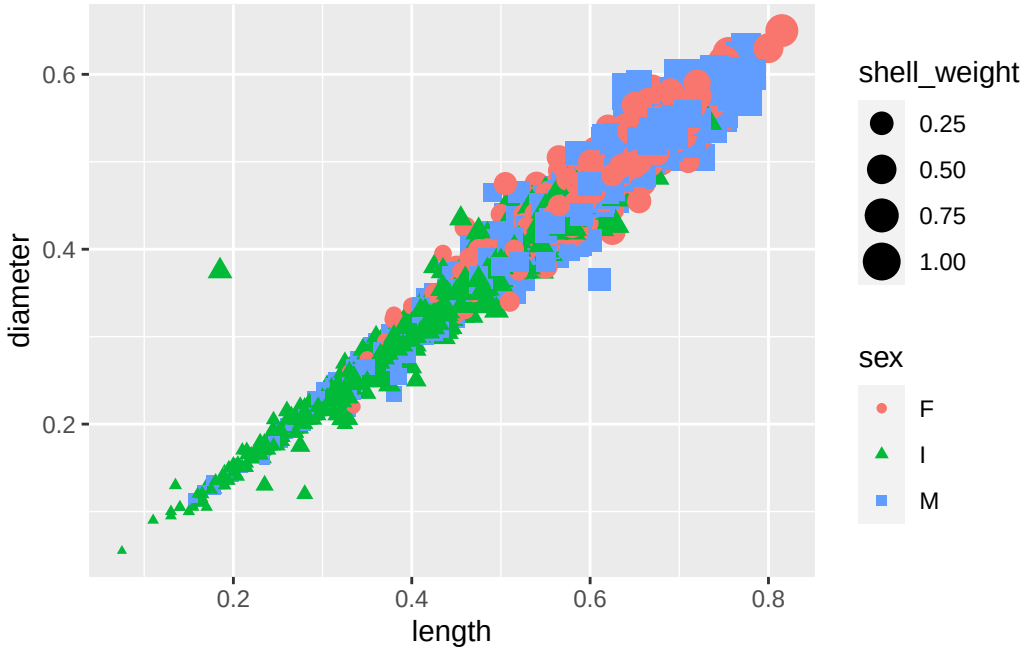


It would seem that the abalones of 'sex' "I" are significantly shorter in both 'diameter' and 'length' than those of 'sex' "M" and "F". In addition to this, the abalones of 'sex' "F" generally have a larger diameter than the other sexes.

1.5 (5 points)

Create a scatter plot of `length` and `diameter`, and modify the shape and color of the points based on the `sex` variable. Change the size of each point based on the `shell_wight` value for each observation. Are there any notable anomalies in the dataset?

```
scatter_plt <-
  ggplot(df, aes(x=length, y=diameter)) +
  # specifying the 'shape' to make the plot more readable
  geom_point(aes(shape=sex, color=sex, size=shell_weight))
scatter_plt
```



The plot created above shows that as 'length' increases so too does 'diameter'. When it comes to any notable anomalies that are displayed by the plot, a few come to mind. For example, there is an outlier abalone of 'sex' "I" represented by a triangle that is far larger in 'diameter' than other abalones of proportional 'length'. In addition to this, it has become evident that abalones of 'sex' "M" have a much more diverse size range than those of 'sex' "F". In other words, male abalones can be much smaller than female abalones while still growing relatively large in line with the average size of females.

1.6 (5 points)

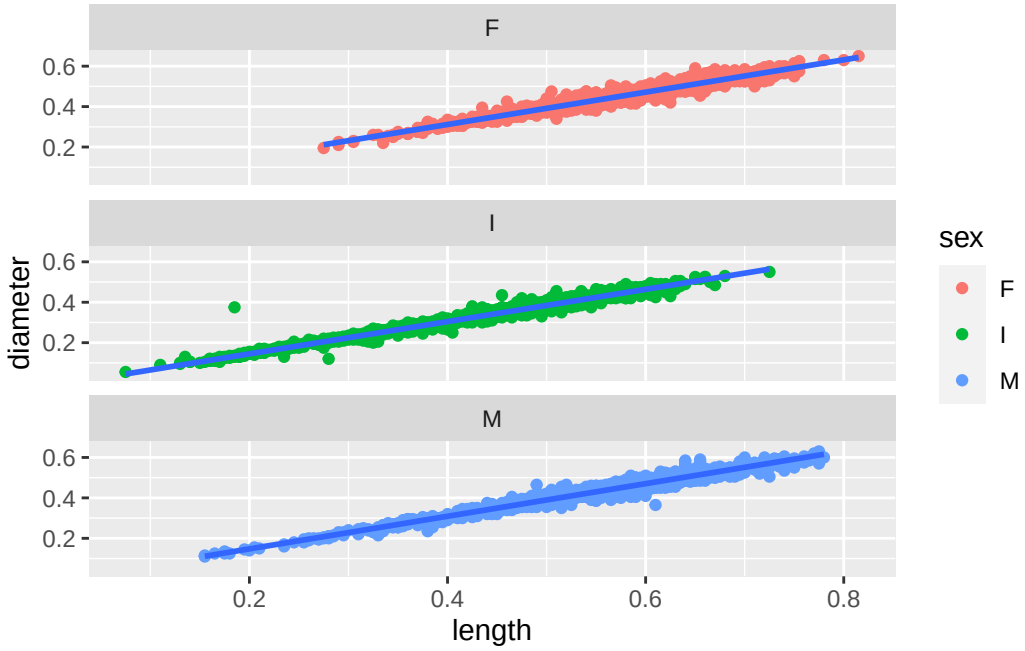
For each `sex`, create separate scatter plots of `length` and `diameter`. For each plot, also add a **linear** trendline to illustrate the relationship between the variables. Use the `facet_wrap()` function in R for this, and ensure that the plots are vertically stacked **not** horizontally. You should end up with a plot that looks like this: [2]

```
abalone_plt <-
  ggplot(df, aes(x=length, y=diameter)) +
  geom_point(aes(color=sex)) +
  geom_smooth(method = "lm") +
```

[2]Plot example for 1.6

```
  facet_wrap(vars(sex), dir="v")
abalone_plt
```

```
`geom_smooth()` using formula = 'y ~ x'
```



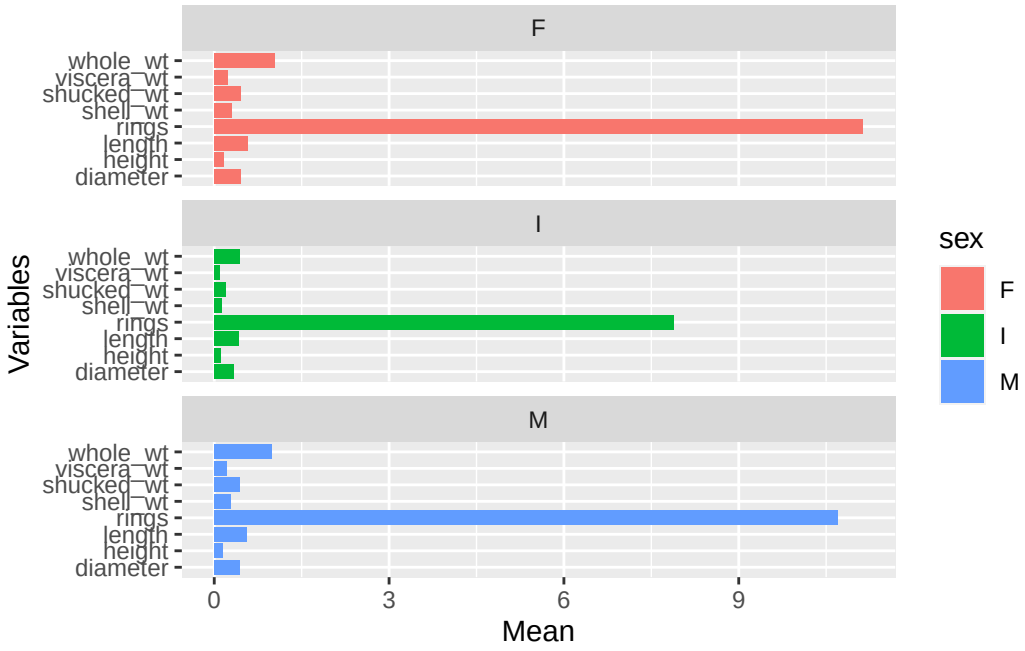
Question 2

> 40 points
>
> More advanced analyses using `dplyr`, `purrrr` and `ggplot2`

2.1 (10 points)

Filter the data to only include abalone with a length of at least 0.5 meters. Group the data by `sex` and calculate the mean of each variable for each group. Create a bar plot to visualize the mean values for each variable by `sex`.

```r
glyph_ready_df <-
  df %>%
  group_by(sex) %>%
  summarize(across(everything(),mean)) %>%
  filter("length" >= 0.5) %>%
  # renaming just to make cleaner
  rename("shell_wt" = shell_weight,
         "shucked_wt" = shucked_weight,
         "viscera_wt" = viscera_weight,
         "whole_wt" = whole_weight) %>%
  # using pivot longer to put all variables into one column
  pivot_longer(!sex,
               names_to = "Variables",
               values_to = "Mean")

ggplot(glyph_ready_df, aes(x=Mean, y=Variables)) +
  geom_bar(stat="identity", aes(fill=sex)) +
  facet_wrap(vars(sex), dir="v")
```



2.2 (15 points)

Implement the following in a **single command**:

1. Temporarily create a new variable called `num_rings` which takes a value of:

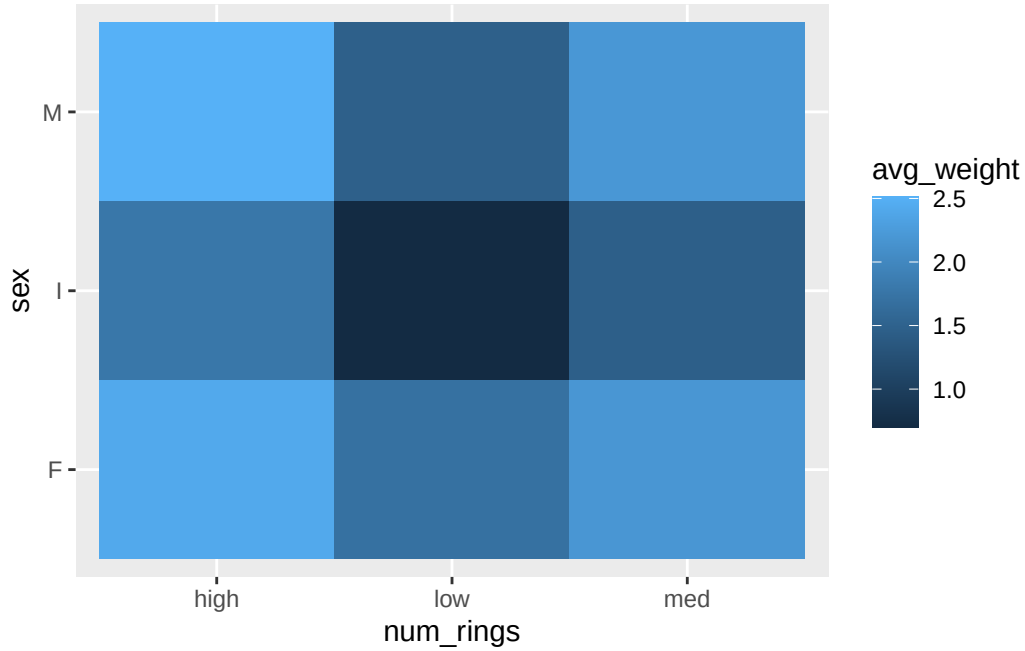
- `"low"` if `rings < 10`
- `"high"` if `rings > 20`, and
- `"med"` otherwise

2. Group `df` by this new variable and `sex` and compute `avg_weight` as the average of the `whole_weight + shucked_weight + viscera_weight + shell_weight` for each combination of `num_rings` and `sex`.

3. Use the `geom_tile()` function to create a tile plot of `num_rings` vs `sex` with the color indicating of each tile indicating the `avg_weight` value.

```
df %>%
  mutate(num_rings = ifelse(rings < 10, "low",
                            ifelse(rings > 20, "high", "med"))) %>%
  group_by(num_rings, sex) %>%
  summarize("avg_weight" = mean(whole_weight + shucked_weight + viscera_weight + shell_wei
  ggplot(aes(x=num_rings, y=sex)) +
  geom_tile(aes(fill=avg_weight))
```

```
`summarise()` has grouped output by 'num_rings'. You can override using the
`.groups` argument.
```



2.3 (5 points)

Make a table of the pairwise correlations between all the numeric variables rounded to 2 decimal points. Your final answer should look like this [3]

```
# creating new data frames with only numeric variables
df2 <- df %>%
  select(!sex)

# assigning variable of pairwise correlations
pair.cor <- round(cor(df2), 2)
pair.cor
```

	length	diameter	height	whole_weight	shucked_weight
length	1.00	0.99	0.83	0.93	0.90
diameter	0.99	1.00	0.83	0.93	0.89
height	0.83	0.83	1.00	0.82	0.77
whole_weight	0.93	0.93	0.82	1.00	0.97

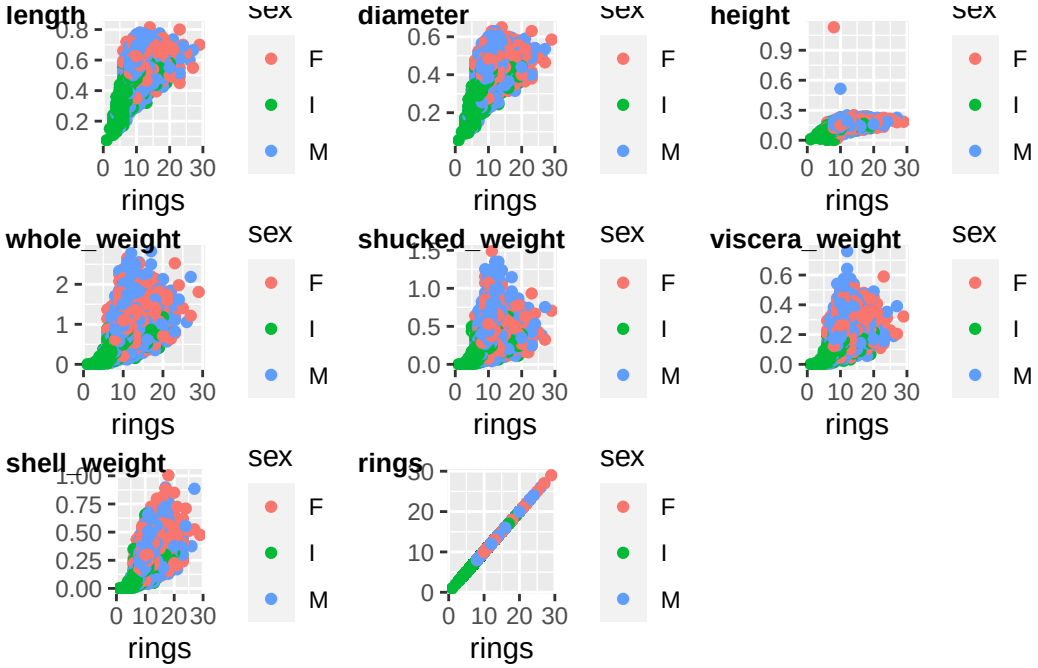
[3] Table for 2.3

```
shucked_weight   0.90     0.89   0.77         0.97           1.00
viscera_weight   0.90     0.90   0.80         0.97           0.93
shell_weight     0.90     0.91   0.82         0.96           0.88
rings            0.56     0.57   0.56         0.54           0.42
                viscera_weight shell_weight rings
length                    0.90         0.90  0.56
diameter                  0.90         0.91  0.57
height                    0.80         0.82  0.56
whole_weight              0.97         0.96  0.54
shucked_weight            0.93         0.88  0.42
viscera_weight            1.00         0.91  0.50
shell_weight              0.91         1.00  0.63
rings                     0.50         0.63  1.00
```

2.4 (10 points)

Use the `map2()` function from the `purrr` package to create a scatter plot for each *quantitative* variable against the number of `rings` variable. Color the points based on the `sex` of each abalone. You can use the `cowplot::plot_grid()` function to finally make the following grid of plots.

```r
# creating data frame with all quantitative variables
df_quant <-
  df %>%
  select(!sex)

# creating data frame with only rings variable
df_y <-
  df %>%
  select(rings)

# using 'map2()' function to make a plot of every quantitative variable vs rings
plot1 <- map2(df_quant, df_y, ~ ggplot(df) +
  geom_point(aes(x=rings, y=.x, color=sex)) +
  labs(x = "rings", y = " "))

# using 'plot_grid()' to put all plots into one grid
cowplot::plot_grid(plotlist = plot1, labels = colnames(df_quant), ncol = 3, label_size = 9
```



Question 3

> 30 points
>
> Linear regression using `lm`

3.1 (10 points)

Perform a simple linear regression with `diameter` as the covariate and `height` as the response. Interpret the model coefficients and their significance values.

```
# assigning 'x' and 'y'
x = df$diameter
y = df$height

# creating the model and summarizing
model <- lm(y ~ x)
```

```r
summary(model)
```

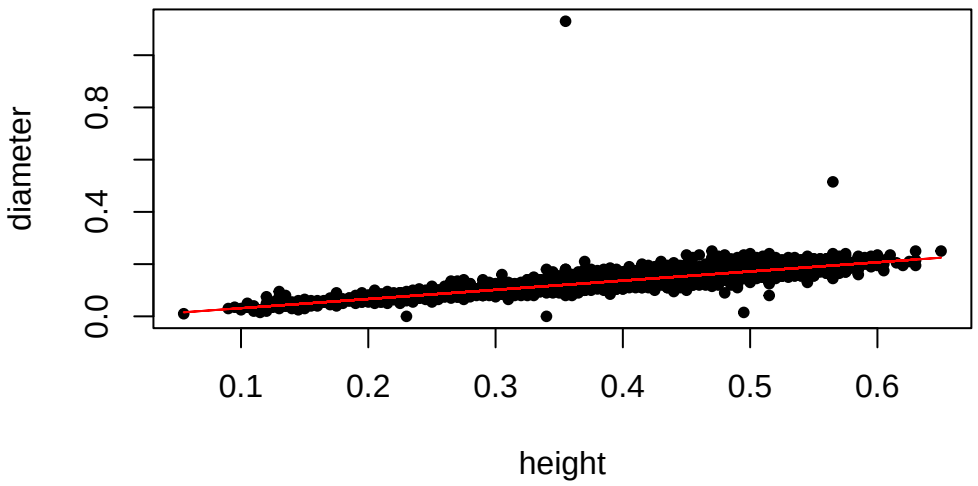
```
Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-0.15513 -0.01053 -0.00147  0.00852  1.00906

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.003803   0.001512  -2.515   0.0119 *
x            0.351376   0.003602  97.544   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0231 on 4175 degrees of freedom
Multiple R-squared:  0.695, Adjusted R-squared:  0.695
F-statistic:  9515 on 1 and 4175 DF,  p-value: < 2.2e-16
```

The model's β_0 coefficient represents the intercept and has a value of -0.003803. This means that the y-intercept of the simple linear regression model is in the negatives. The model's β_1 coefficient represents the slope and has a value of 0.351376. The p-value is very statistically significant therefore we reject the null hypothesis and accept the alternative hypothesis. The R-squared value, on the other hand, is not very large and is not significant.

3.2 (10 points)

Make a scatterplot of `height` vs `diameter` and plot the regression line in `color="red"`. You can use the base `plot()` function in R for this. Is the linear model an appropriate fit for this relationship? Explain.

```r
# creating scatterplot using 'plot()' function
plot(x, y, xlab="height", ylab="diameter", pch=20)
lines(x, fitted(lm(y ~ x)), col="red")
```



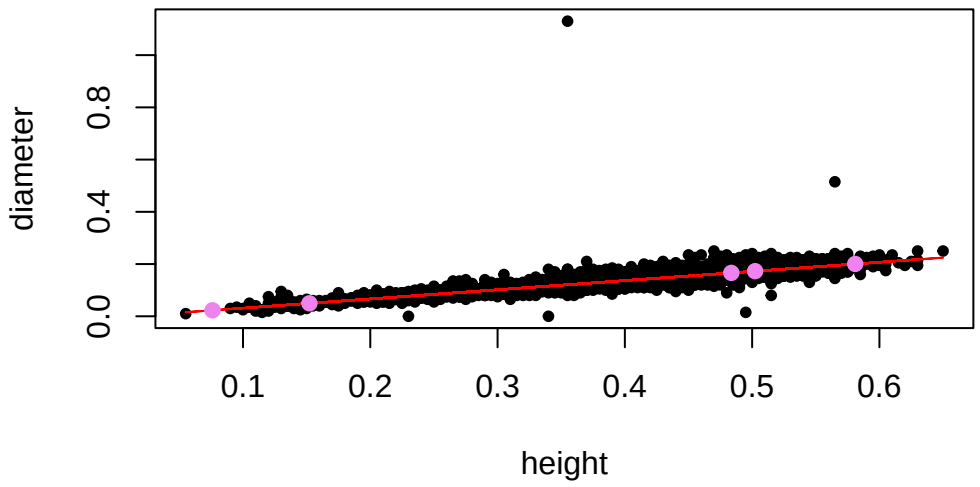
Yes this linear model is an appropriate fit for this relationship because it successfully minimizes the distance from each point to the trend line. In addition to this, the p-value is statistically significant, as mentioned above.

3.3 (10 points)

Suppose we have collected observations for "new" abalones with `new_diameter` values given below. What is the expected value of their `height` based on your model above? Plot these new observations along with your predictions in your plot from earlier using `color="violet"`

```
new_diameters <- c(
  0.15218946,
  0.48361548,
  0.58095513,
  0.07603687,
  0.50234599,
  0.83462092,
  0.95681938,
  0.92906875,
  0.94245437,
```

```
  0.01209518
)

# creating new variables
new_x <- data.frame(x = new_diameters)
new_y <- predict(model, new_x)

# plotting graph and adding predicted points
plot(x, y, xlab="height", ylab="diameter", pch=20)
lines(x, fitted(lm(y ~ x)), col="red")
points(new_x %>% unlist(), new_y, col="violet", pch=19)
```



The expected value of the "new" abalones height based on the model above is as follows:

```
new_y
```

```
           1            2            3            4            5            6
0.0496723682 0.1661276096 0.2003304536 0.0229141546 0.1727090665 0.2894625947
           7            8            9           10
0.3324002348 0.3226493217 0.3273527111 0.0004465615
```

Appendix

Session Information

Print your R session information using the following command

```
sessionInfo()
```

```
R version 4.2.1 (2022-06-23 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 22000)

Matrix products: default

locale:
[1] LC_COLLATE=English_United States.utf8
[2] LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

attached base packages:
[1] stats     graphics  grDevices datasets  utils     methods   base

other attached packages:
[1] cowplot_1.1.1 purrr_1.0.1   dplyr_1.1.0   ggplot2_3.4.1 tidyr_1.3.0
[6] readr_2.1.4

loaded via a namespace (and not attached):
 [1] pillar_1.8.1     compiler_4.2.1   tools_4.2.1      bit_4.0.5
 [5] digest_0.6.31    lattice_0.20-45  nlme_3.1-157     jsonlite_1.8.4
 [9] evaluate_0.20    lifecycle_1.0.3  tibble_3.1.8     gtable_0.3.1
[13] mgcv_1.8-40      pkgconfig_2.0.3  rlang_1.0.6      Matrix_1.4-1
[17] cli_3.6.0        parallel_4.2.1   yaml_2.3.7       xfun_0.37
[21] fastmap_1.1.0    withr_2.5.0      knitr_1.42       generics_0.1.3
[25] vctrs_0.5.2      hms_1.1.2        bit64_4.0.5      grid_4.2.1
[29] tidyselect_1.2.0 glue_1.6.2       R6_2.5.1         fansi_1.0.4
[33] vroom_1.6.1      rmarkdown_2.20   farver_2.1.1     tzdb_0.3.0
```

```
[37] magrittr_2.0.3   splines_4.2.1    scales_1.2.1     ellipsis_0.3.2
[41] htmltools_0.5.4  colorspace_2.1-0 renv_0.16.0-53   labeling_0.4.2
[45] utf8_1.2.3       munsell_0.5.0    crayon_1.5.2
```