Lab2

Exercises

Irimie Fabio

Contents

Exercise 1 - Create a new mean and sd function

A

Create the Lab2 project. Use the same structure used for Lab1:

- scripts,
- plots,

- data

B

Install the palmerpenguins package, load the penguins dataset or, alternatively, download the .RData object from moodle and import it after placing it inside the data directory of the project (hint: use the load() function).

```
library(palmerpenguins)
data(penguins)
```

C

Compute the mean, the standard deviation, and the median for the numeric variables of the dataset.

```
# Means
cat("Means: \n")
## Means:
colMeans(penguins[, c(3:6, 8)], na.rm = TRUE)
##    bill_length_mm     bill_depth_mm flipper_length_mm       body_mass_g 
##          43.92193          17.15117         200.91520        4201.75439 
##              year 
##        2008.02907
cat("\n")

# Medians
cat("Medians: \n")
## Medians:
sapply(penguins[, c(3:6, 8)], median, na.rm = TRUE)
##    bill_length_mm     bill_depth_mm flipper_length_mm       body_mass_g 
##             44.45             17.30            197.00           4050.00 
##              year 
##           2008.00
cat("\n")

# Standard deviations
cat("Standard deviations: \n")
## Standard deviations:
sapply(penguins[, c(3:6, 8)], sd, na.rm = TRUE)
##    bill_length_mm     bill_depth_mm flipper_length_mm       body_mass_g 
##         5.4595837         1.9747932        14.0617137       801.9545357 
##              year 
##         0.8183559
cat("\n")
```

D

Create a function called stat_auto that simultaneously returns both the mean and the standard deviation of a given vector (hint: return an object of type list or simply a vector). Then try it on the same numeric variables in C. to check the results (hint: if you obtain NA maybe you forgot to remove NA terms in the vector).

```
stat_auto <- function(vec, na.rm = FALSE) {
  if (na.rm) {
    mean <- mean(vec, na.rm = TRUE)
```

```r
    sd <- sd(vec, na.rm = TRUE)

    return(list("mean" = mean, "sd" = sd))
  }

  mean <- mean(vec)
  sd <- sd(vec)

  return(list("mean" = mean, "sd" = sd))
}

sapply(penguins[, c(3:6, 8)], stat_auto, na.rm = TRUE)
```

```
##      bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
## mean 43.92193       17.15117      200.9152          4201.754
## sd   5.459584       1.974793      14.06171          801.9545
##      year
## mean 2008.029
## sd   0.8183559
```

E

Create a function called stat_manual that simultaneously returns both the mean and the standard deviation of a given vector without using the mean() and the sd() functions (hint: you can use length(), sum(), and na.omit() functions). Then try it on the same numeric variables in C. to check the results.

```r
stat_manual <- function(vec, na.rm = FALSE) {
  if (na.rm) {
    sum <- sum(vec, na.rm = TRUE)
    mean <- sum / na.omit(length(vec))

    sum <- sum((vec - mean)^2, na.rm = TRUE)
    denom <- na.omit(length(vec)) - 1
    varianza <- sum / denom

    sd <- sqrt(varianza)

    return(list("mean" = mean, "sd" = sd))
  }

  sum <- sum(vec)
  mean <- sum / length(vec)

  sum <- sum((vec - mean)^2)
  denom <- length(vec) - 1
  varianza <- sum / denom

  sd <- sqrt(varianza)

  return(list("mean" = mean, "sd" = sd))
}

sapply(penguins[, c(3:6, 8)], stat_manual, na.rm = TRUE)
```

```
##      bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
```

```
## mean 43.66657       17.05145      199.7471          4177.326
## sd   5.449612       1.971543      14.06909          799.985
##      year
## mean 2008.029
## sd   0.8183559
```

Exercise 2 - Table of frequencies

A

In the penguins dataset, transform a numeric variable to a categorical one by aggregating values into classes. Consider the flipper length variable and create 10mm wide classes using the cut() function (hint: use the range() function to determine the min and max values of the variable, then define a sequence for the cuts).

```r
r <- range(penguins$flipper_length_mm, na.rm = TRUE)
splits <- seq(r[1], r[2], 10)
splits <- append(splits, r[2])

classes <- cut(penguins$flipper_length_mm, splits, ordered_result = TRUE)

cat("Splits: ", splits, "\n")
```

```
## Splits:  172 182 192 202 212 222 231
```

```r
cat("Classes: \n")
```

```
## Classes:
```

```r
classes
```

```
##   [1] (172,182] (182,192] (192,202] <NA>      (192,202] (182,192]
##   [7] (172,182] (192,202] (192,202] (182,192] (182,192] (172,182]
##  [13] (172,182] (182,192] (192,202] (182,192] (192,202] (192,202]
##  [19] (182,192] (192,202] (172,182] (172,182] (182,192] (182,192]
##  [25] (172,182] (182,192] (182,192] (182,192] <NA>      (172,182]
##  [31] (172,182] (172,182] (182,192] (182,192] (192,202] (192,202]
##  [37] (182,192] (172,182] (172,182] (182,192] (172,182] (192,202]
##  [43] (182,192] (192,202] (182,192] (182,192] (172,182] (172,182]
##  [49] (182,192] (182,192] (182,192] (182,192] (182,192] (192,202]
##  [55] (182,192] (182,192] (182,192] (192,202] (172,182] (192,202]
##  [61] (182,192] (192,202] (182,192] (182,192] (182,192] (182,192]
##  [67] (192,202] (182,192] (182,192] (192,202] (182,192] (182,192]
##  [73] (192,202] (192,202] (182,192] (192,202] (182,192] (182,192]
##  [79] (182,192] (192,202] (182,192] (192,202] (182,192] (192,202]
##  [85] (182,192] (192,202] (182,192] (182,192] (182,192] (182,192]
##  [91] (192,202] (202,212] (182,192] (182,192] (182,192] (202,212]
##  [97] (182,192] (192,202] (172,182] (182,192] (182,192] (202,212]
## [103] (182,192] (182,192] (192,202] (182,192] (192,202] (182,192]
## [109] (172,182] (192,202] (192,202] (182,192] (192,202] (192,202]
## [115] (182,192] (192,202] (182,192] (192,202] (182,192] (182,192]
## [121] (182,192] (192,202] (172,182] (192,202] (182,192] (192,202]
## [127] (182,192] (192,202] (182,192] (202,212] (182,192] (192,202]
## [133] (192,202] (192,202] (182,192] (182,192] (182,192] (192,202]
## [139] (182,192] (192,202] (192,202] (182,192] (182,192] (182,192]
## [145] (182,192] (182,192] (182,192] (182,192] (192,202] (192,202]
## [151] (182,192] (192,202] (202,212] (222,231] (202,212] (212,222]
```

```
## [157] (212,222] (202,212] (202,212] (212,222] (202,212] (212,222]
## [163] (212,222] (212,222] (212,222] (212,222] (202,212] (212,222]
## [169] (202,212] (212,222] (202,212] (212,222] (212,222] (212,222]
## [175] (212,222] (212,222] (212,222] (212,222] (212,222] (212,222]
## [181] (202,212] (212,222] (212,222] (202,212] (202,212] (222,231]
## [187] (212,222] (212,222] (212,222] (212,222] (202,212] (202,212]
## [193] (202,212] (222,231] (202,212] (212,222] (212,222] (212,222]
## [199] (202,212] (222,231] (212,222] (212,222] (202,212] (212,222]
## [205] (202,212] (222,231] (212,222] (212,222] (202,212] (212,222]
## [211] (202,212] (222,231] (202,212] (212,222] (212,222] (222,231]
## [217] (212,222] (222,231] (212,222] (222,231] (212,222] (222,231]
## [223] (212,222] (212,222] (212,222] (212,222] (212,222] (222,231]
## [229] (202,212] (212,222] (212,222] (222,231] (202,212] (212,222]
## [235] (202,212] (222,231] (202,212] (222,231] (212,222] (212,222]
## [241] (202,212] (222,231] (212,222] (222,231] (202,212] (222,231]
## [247] (212,222] (222,231] (212,222] (212,222] (202,212] (222,231]
## [253] (212,222] (222,231] (212,222] (222,231] (212,222] (212,222]
## [259] (202,212] (212,222] (202,212] (202,212] (212,222] (222,231]
## [265] (212,222] (222,231] (212,222] (222,231] (212,222] (212,222]
## [271] (212,222] <NA>      (212,222] (212,222] (202,212] (212,222]
## [277] (182,192] (192,202] (192,202] (182,192] (192,202] (192,202]
## [283] (172,182] (192,202] (192,202] (192,202] (192,202] (192,202]
## [289] (182,192] (192,202] (182,192] (192,202] (192,202] (172,182]
## [295] (182,192] (192,202] (172,182] (182,192] (182,192] (192,202]
## [301] (192,202] (192,202] (192,202] (192,202] (182,192] (202,212]
## [307] (182,192] (192,202] (182,192] (202,212] (192,202] (192,202]
## [313] (192,202] (202,212] (182,192] (202,212] (202,212] (182,192]
## [319] (192,202] (192,202] (192,202] (192,202] (182,192] (202,212]
## [325] (182,192] (192,202] (192,202] (192,202] (192,202] (202,212]
## [331] (182,192] (192,202] (182,192] (202,212] (192,202] (192,202]
## [337] (202,212] (182,192] (192,202] (202,212] (192,202] (192,202]
## [343] (202,212] (192,202]
## 6 Levels: (172,182] < (182,192] < (192,202] < ... < (222,231]
```

B

Use the table() function on the new variable generated by cut(). Then transform it into a data.frame object. Rename the columns accordingly using the colnames() function (hint: the second column correspond to the absolute frequencies).

```r
df <- data.frame(table(classes))
colnames(df) <- c("Classes", "AbsoluteFrequencies")
df
```

```
##     Classes AbsoluteFrequencies
## 1 (172,182]                  22
## 2 (182,192]                  96
## 3 (192,202]                  85
## 4 (202,212]                  47
## 5 (212,222]                  67
## 6 (222,231]                  24
```

C

Add the the columns for: relative frequencies, cumulative absolute frequencies, and cumulative relative frequencies.

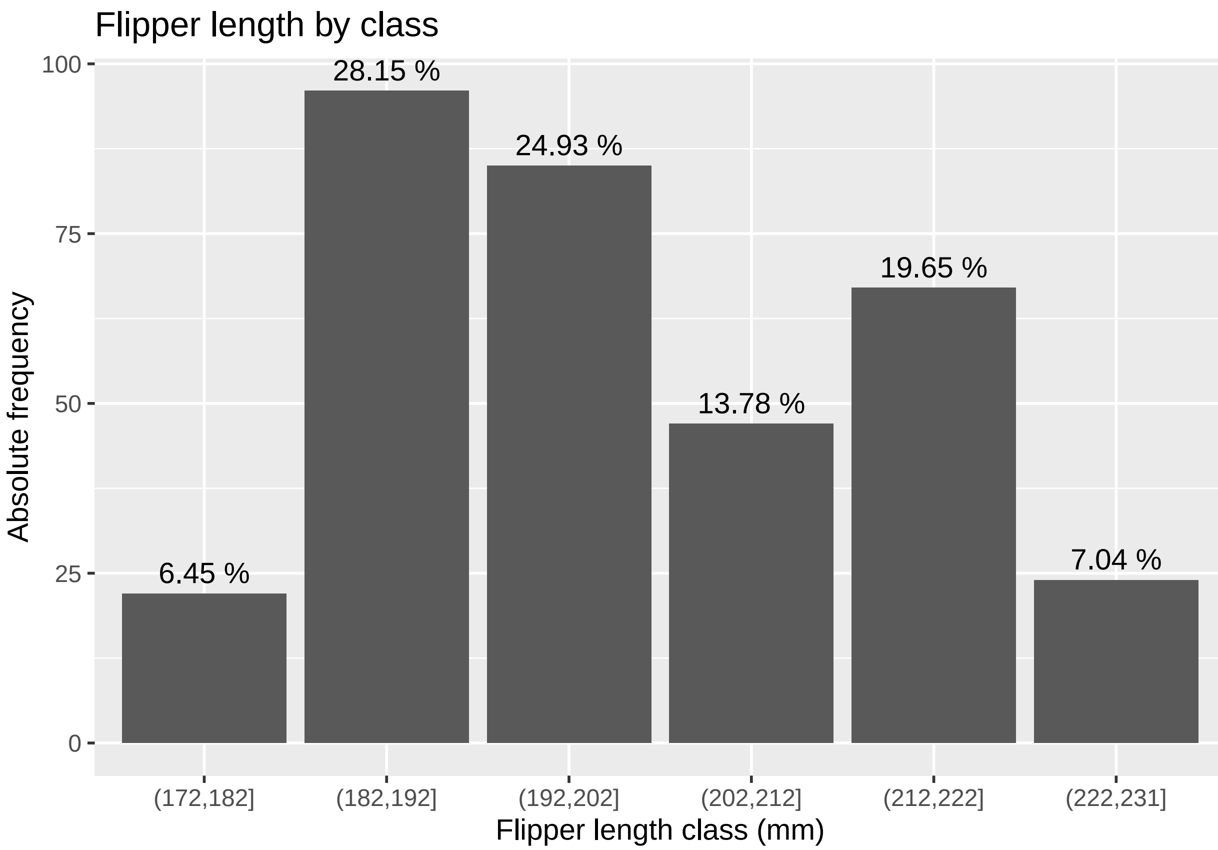
```
df$CumAbsFreq <- cumsum(df$AbsoluteFrequencies)
df$RelativeFrequencies <- df$AbsoluteFrequencies / sum(df$AbsoluteFrequencies)
df$RelAbsFreq <- cumsum(df$RelativeFrequencies)
df
```

```
##     Classes AbsoluteFrequencies CumAbsFreq RelativeFrequencies
## 1 (172,182]                  22         22          0.06451613
## 2 (182,192]                  96        118          0.28152493
## 3 (192,202]                  85        203          0.24926686
## 4 (202,212]                  47        250          0.13782991
## 5 (212,222]                  67        317          0.19648094
## 6 (222,231]                  24        341          0.07038123
##   RelAbsFreq
## 1 0.06451613
## 2 0.34604106
## 3 0.59530792
## 4 0.73313783
## 5 0.92961877
## 6 1.00000000
```

D

Use the geom_col() function to plot the frequence of each class. Then, using the geom_text(aes(label = ...)) function, add the relative frequence as a percentage above each column (hint: substitute the ... with the relative frequency values. Use the round() function to choose the appropriate number of digits).

```
library(ggplot2)
ggplot(
  data = df,
  aes(
    x = df$Class,
    y = df$AbsoluteFrequency,
  )
) +
  geom_col(aes(y = df$AbsoluteFrequencies)) +
  geom_text(
    aes(
      label = paste(round(df$RelativeFrequencies * 100, 2), "%"),
      y = df$AbsoluteFrequencies,
    ),
    vjust = -0.5,
  ) +
  labs(
    title = "Flipper length by class",
    x = "Flipper length class (mm)",
    y = "Absolute frequency",
  ) +
  theme(legend.position = "bottom")
```



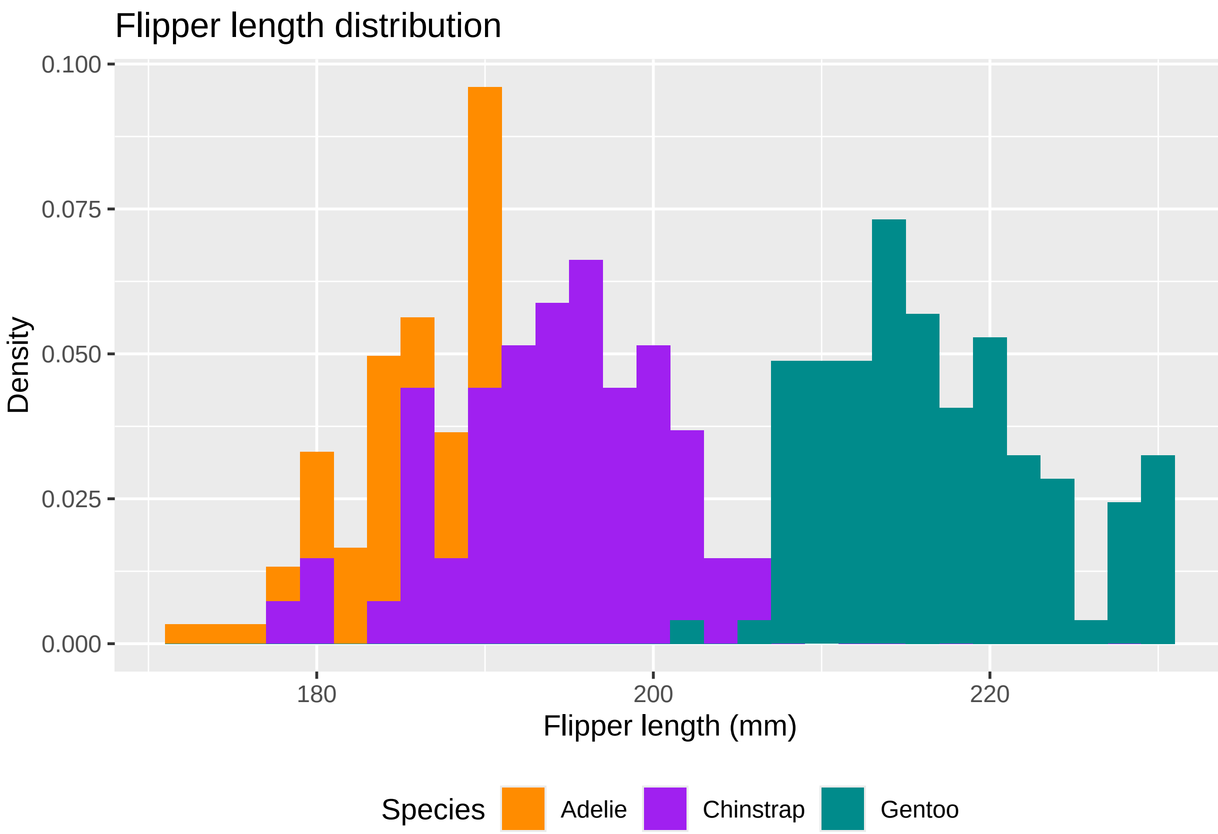
Exercise 3 - Histogram, Boxplot and quartiles

A

Using the geom_histogram() function of the ggplot2 package plot the flipper length distribution coloring each species with a different color (hint: use the fill argument of the aes() function to fill the histogram area and the position = "identity" argument of the geom_histogram()). Play with the binwidth argument. Try to insert y = ..density.. in aes(). Do you notice any change?

```
ggplot(
  data = penguins,
  aes(
    x = penguins$flipper_length_mm,
    fill = penguins$species,
  )
) +
  scale_fill_manual(values = c("darkorange", "purple", "cyan4")) +
  geom_histogram(
    position = "identity",
    binwidth = 2,
    aes(y = ..density..),
  ) +
  labs(
    title = "Flipper length distribution",
    x = "Flipper length (mm)",
    y = "Density",
    fill = "Species",
```

```r
) +
  theme(legend.position = "bottom")
```



B

About the flipper length, for each species of penguins compute the: 1. Sample mean 2. Sample median 3. Sample standard deviation (use a division by $n-1$) 4. Sample variance

(hint: to choose only a specific species use penguins[penguins$species == "Gentoo",])

```r
species <- unique(penguins$species)
summary <- data.frame(
  Specie = species
)

for (specie in species) {
  summary$Mean[which(specie == species)] <-
    mean(penguins[penguins$species == specie, ]$flipper_length_mm,
      na.rm = TRUE
    )
  summary$Median[which(specie == species)] <-
    median(penguins[penguins$species == specie, ]$flipper_length_mm,
      na.rm = TRUE
    )
  summary$StandardDeviation[which(specie == species)] <-
    sd(penguins[penguins$species == specie, ]$flipper_length_mm,
      na.rm = TRUE
    )
```

```r
  summary$Variance[which(specie == species)] <-
    var(penguins[penguins$species == specie, ]$flipper_length_mm,
      na.rm = TRUE
    )
}
summary
```

```
##      Specie     Mean Median StandardDeviation Variance
## 1    Adelie 189.9536    190          6.539457 42.76450
## 2    Gentoo 217.1870    216          6.484976 42.05491
## 3 Chinstrap 195.8235    196          7.131894 50.86392
```

C

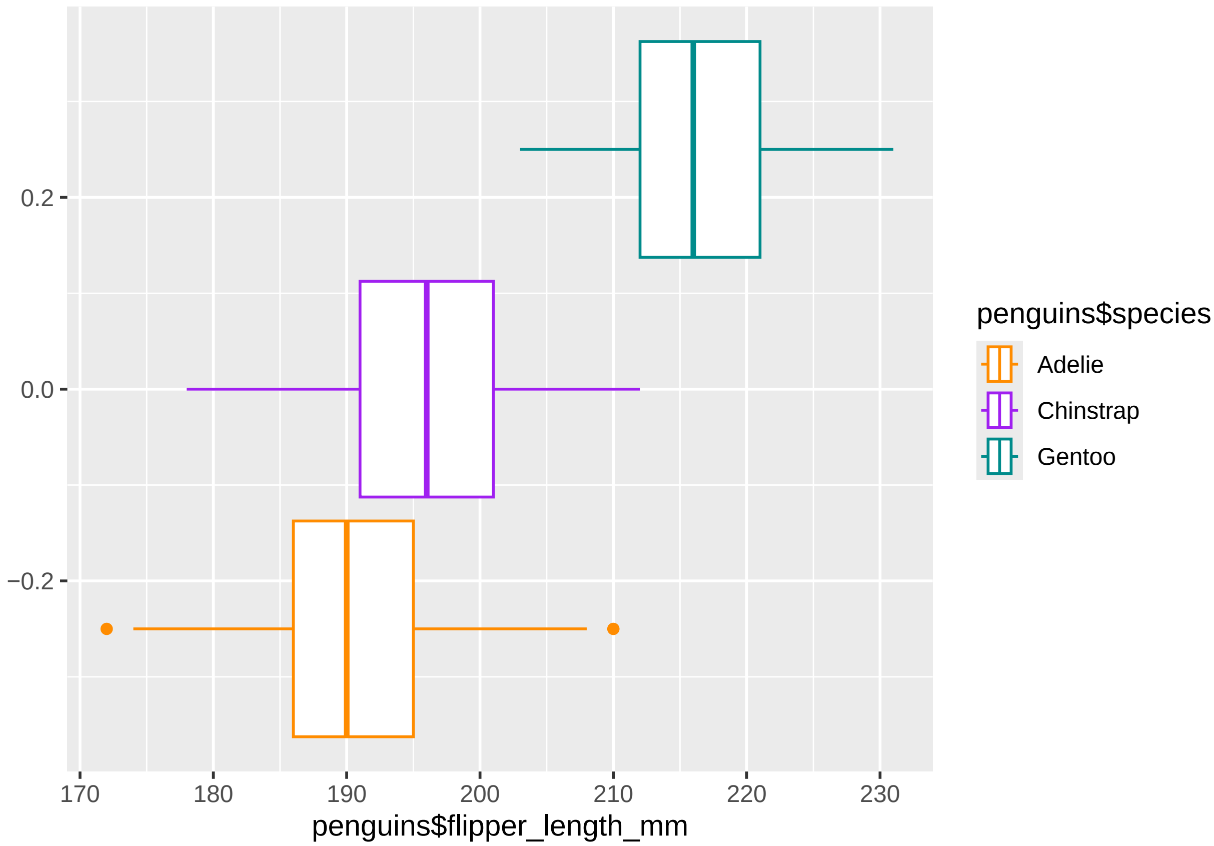
Using the geom_boxplot() function of the ggplot2 package plot the boxplot for the flipper length variable coloring each species with a different color (hint: use the color argument of the aes() function).

```r
ggplot(
  penguins,
  aes(
    x = penguins$flipper_length_mm,
    color = penguins$species
  )
) +
  scale_color_manual(values = c("darkorange", "purple", "cyan4")) +
  geom_boxplot(
    aes(x = penguins$flipper_length_mm)
  )
```

```
## Warning: Use of `penguins$flipper_length_mm` is discouraged.
## i Use `flipper_length_mm` instead.

## Warning: Use of `penguins$species` is discouraged.
## i Use `species` instead.

## Warning: Removed 2 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



D

Compute the flipper length quartiles for the "Gentoo" penguins (Q1, Q2, Q3).

```
gentoo <- penguins[penguins$species == "Gentoo", ]
quartiles <- quantile(
  gentoo$flipper_length_mm,
  c(0.25, 0.5, 0.75),
  na.rm = TRUE
)
quartiles
```

```
## 25% 50% 75%
## 212 216 221
```

E

Calculate the flipper length 40th percentile for the "Adelie" penguins.

```
adelie <- penguins[penguins$species == "Adelie", ]
p40 <- quantile(
  adelie$flipper_length_mm,
  0.4,
  na.rm = TRUE
)
p40
```

```
## 40%
```

```
## 189
```

Exercise 4 - Multiple boxplots from scratch

A

Generate random data with some structure, and create one data set for each day of the week (hint: use the for() cycle, data should have 7 columns). At the end you should obtain a matrix with N rows (N = number of random number to generate each time) and 7 columns (one for each day of the week).

```r
set.seed(123)
n <- 10
days <- c(
  "Monday",
  "Tuesday",
  "Wednesday",
  "Thursday",
  "Friday",
  "Saturday",
  "Sunday"
)
data <- matrix(nrow = n, ncol = 7)

for (i in 1:7) {
  data[, i] <- round(runif(n, min = 0, max = 100), 0)
}

df <- data.frame(data)
colnames(df) <- days
df
```

```
##    Monday Tuesday Wednesday Thursday Friday Saturday Sunday
## 1      29      96        89       96     14        5     67
## 2      79      45        69       90     41       44      9
## 3      41      68        64       69     41       80     38
## 4      88      57        99       80     37       12     27
## 5      94      10        66        2     15       56     81
## 6       5      90        71       48     14       21     45
## 7      53      25        54       76     23       13     81
## 8      89       4        59       22     47       75     81
## 9      55      33        29       32     27       90     79
## 10     46      95        15       23     86       37     44
```

B

Go from a wide to a long data format. You should create a data.frame object with exactly two columns. One contains the values created in A., the other contains the corresponding day of the week.

```r
df_long <- data.frame(
  "Day" = rep(days, each = n),
  "Value" = c(
    df$Monday,
    df$Tuesday,
    df$Wednesday,
    df$Thursday,
```

```r
    df$Friday,
    df$Saturday,
    df$Sunday
  )
)
df_long
```

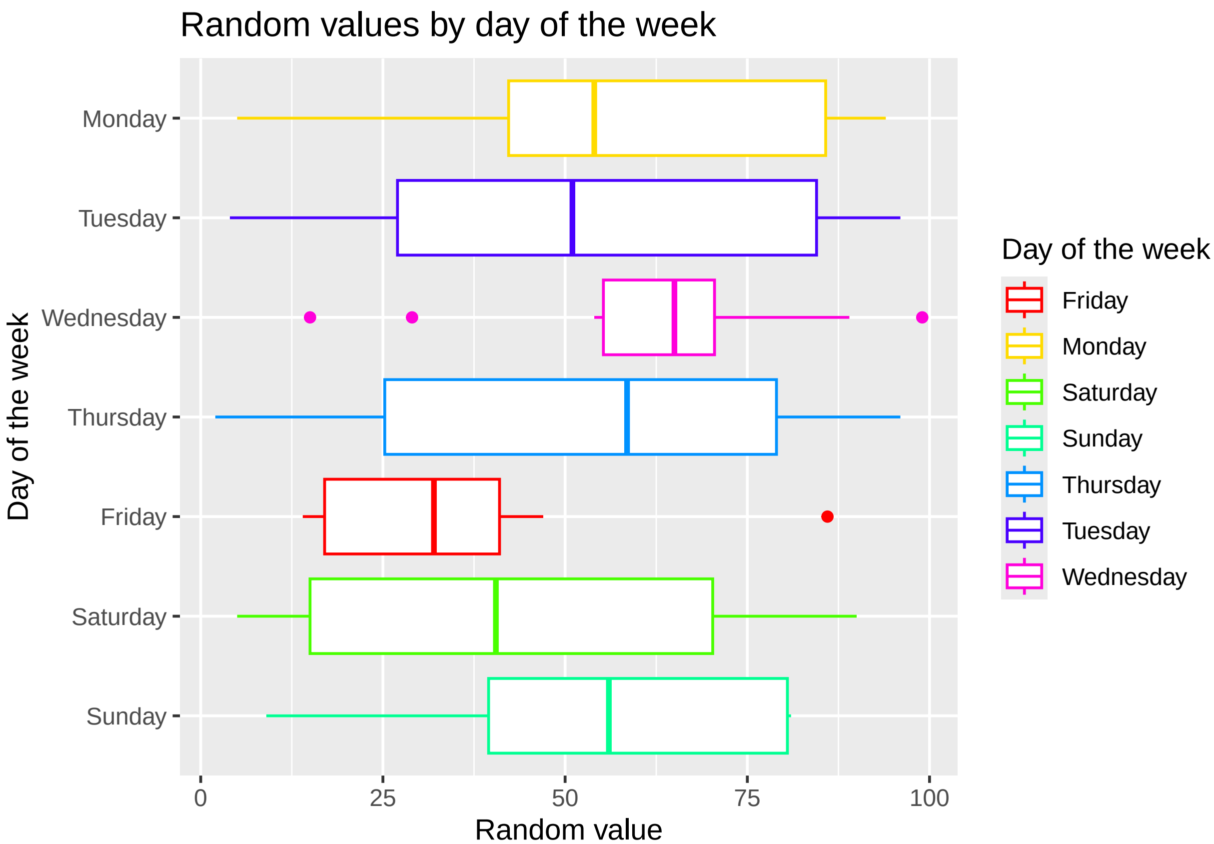
```
##          Day Value
## 1     Monday    29
## 2     Monday    79
## 3     Monday    41
## 4     Monday    88
## 5     Monday    94
## 6     Monday     5
## 7     Monday    53
## 8     Monday    89
## 9     Monday    55
## 10    Monday    46
## 11   Tuesday    96
## 12   Tuesday    45
## 13   Tuesday    68
## 14   Tuesday    57
## 15   Tuesday    10
## 16   Tuesday    90
## 17   Tuesday    25
## 18   Tuesday     4
## 19   Tuesday    33
## 20   Tuesday    95
## 21 Wednesday    89
## 22 Wednesday    69
## 23 Wednesday    64
## 24 Wednesday    99
## 25 Wednesday    66
## 26 Wednesday    71
## 27 Wednesday    54
## 28 Wednesday    59
## 29 Wednesday    29
## 30 Wednesday    15
## 31  Thursday    96
## 32  Thursday    90
## 33  Thursday    69
## 34  Thursday    80
## 35  Thursday     2
## 36  Thursday    48
## 37  Thursday    76
## 38  Thursday    22
## 39  Thursday    32
## 40  Thursday    23
## 41    Friday    14
## 42    Friday    41
## 43    Friday    41
## 44    Friday    37
## 45    Friday    15
## 46    Friday    14
```

```
## 47    Friday    23
## 48    Friday    47
## 49    Friday    27
## 50    Friday    86
## 51  Saturday     5
## 52  Saturday    44
## 53  Saturday    80
## 54  Saturday    12
## 55  Saturday    56
## 56  Saturday    21
## 57  Saturday    13
## 58  Saturday    75
## 59  Saturday    90
## 60  Saturday    37
## 61    Sunday    67
## 62    Sunday     9
## 63    Sunday    38
## 64    Sunday    27
## 65    Sunday    81
## 66    Sunday    45
## 67    Sunday    81
## 68    Sunday    81
## 69    Sunday    79
## 70    Sunday    44
```

C

Plot the seven boxplots (one for each day of the week) in one graph, horizontally oriented (hint: coord_flip() function translates the axes, the "limits" argument of scale_x_discrete() allows you to reorder the axis labels).

```
ggplot(
  df_long,
  aes(
    x = Day,
    y = Value,
    color = Day
  )
) +
  scale_color_manual(values = rainbow(7)) +
  scale_x_discrete(limits = rev(days)) +
  geom_boxplot() +
  coord_flip() +
  labs(
    title = "Random values by day of the week",
    x = "Day of the week",
    y = "Random value",
    color = "Day of the week"
  )
```



Exercise 5 - Exploratory analysis of data

Penguins dataset does not contain their weights and flipper lengths only. Many other variables are available. Let's explore it a little more:

A

How many islands are there? And how many penguins are present in each isle? Are the 3 species of penguins living together? (hint: use the table() function).

```
cat("The number of islands is: ", length(unique(penguins$island)), "\n")
```

```
## The number of islands is:  3
```

```
table(penguins$island)
```

```
##
##    Biscoe     Dream Torgersen
##       168       124        52
```

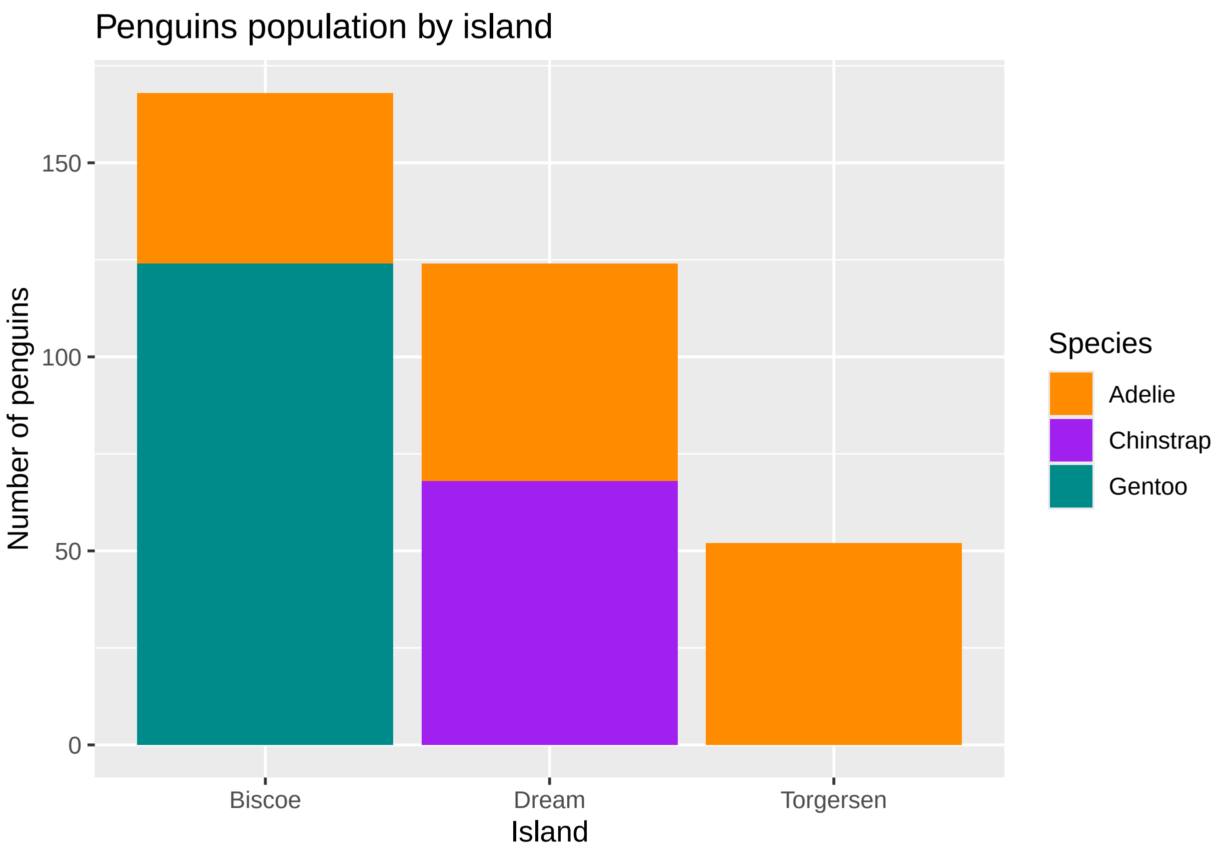
```
table(penguins$island, penguins$species)
```

```
##
##             Adelie Chinstrap Gentoo
##   Biscoe        44         0    124
##   Dream         56        68      0
##   Torgersen     52         0      0
```

B

Try to use the geom_bar() or geom_col() functions to graphically represent the population of each island, colored by species (hint: islands in the x-axis, number of penguins in the y-axis).

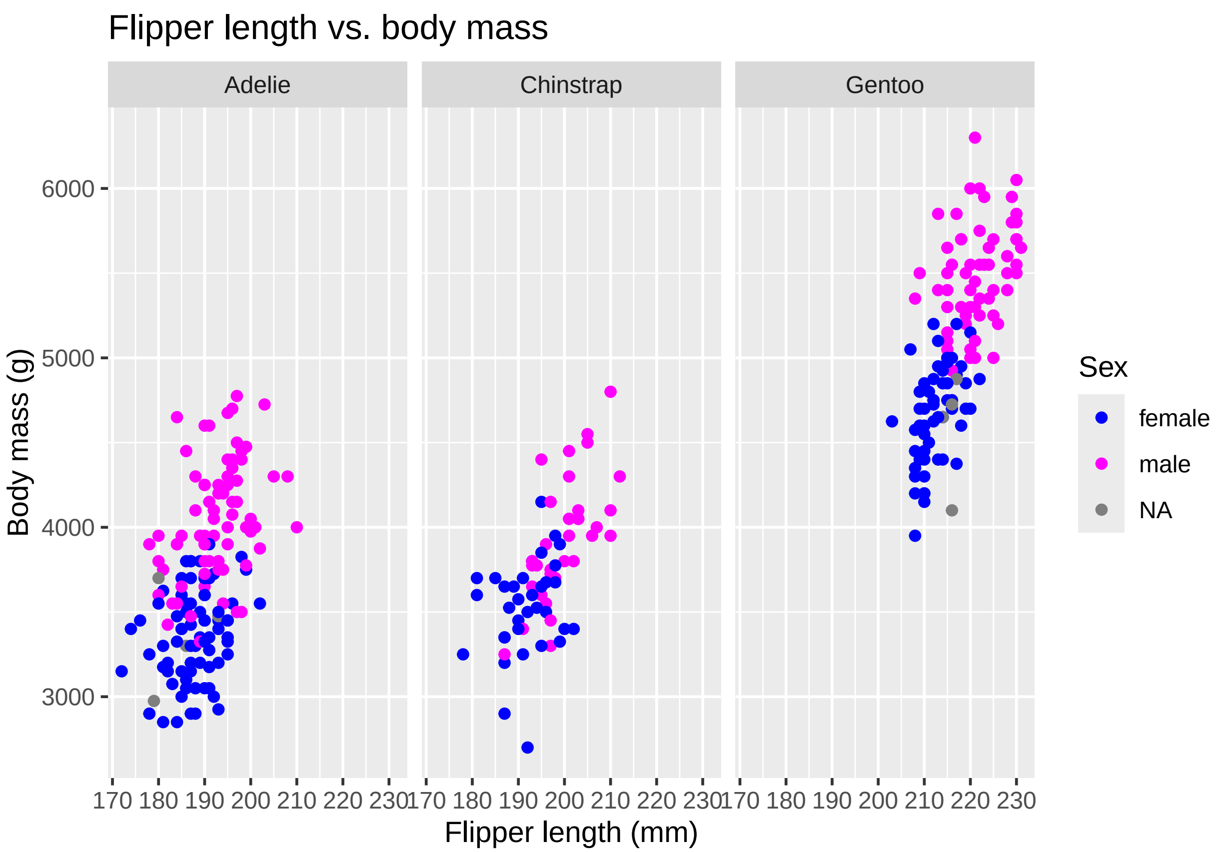
```r
ggplot(
  penguins,
  aes(
    x = island,
    fill = species
  )
) +
  geom_bar() +
  scale_fill_manual(values = c("darkorange", "purple", "cyan4")) +
  labs(
    title = "Penguins population by island",
    x = "Island",
    y = "Number of penguins",
    fill = "Species"
  )
```



C

Use a scatter plot to represent flipper length vs. body mass. Color the point according to the “sex” variable. Try to use facets to see if there are differences across species (hint: use facet_grid(~ species) function to add facets for species).

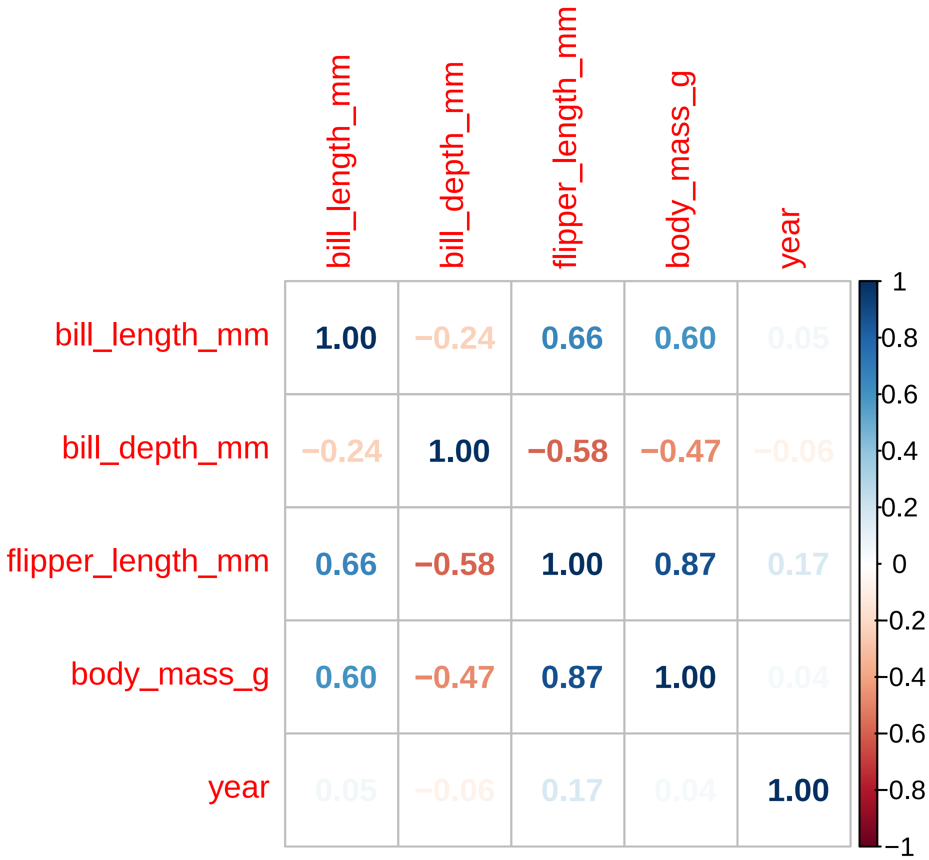
```r
ggplot(
  penguins,
  aes(
    x = penguins$flipper_length_mm,
    y = penguins$body_mass_g,
    color = penguins$sex
  )
) +
  geom_point() +
  facet_grid(~species) +
  scale_color_manual(values = c("blue", "magenta")) +
  labs(
    title = "Flipper length vs. body mass",
    x = "Flipper length (mm)",
    y = "Body mass (g)",
    color = "Sex"
  )
```



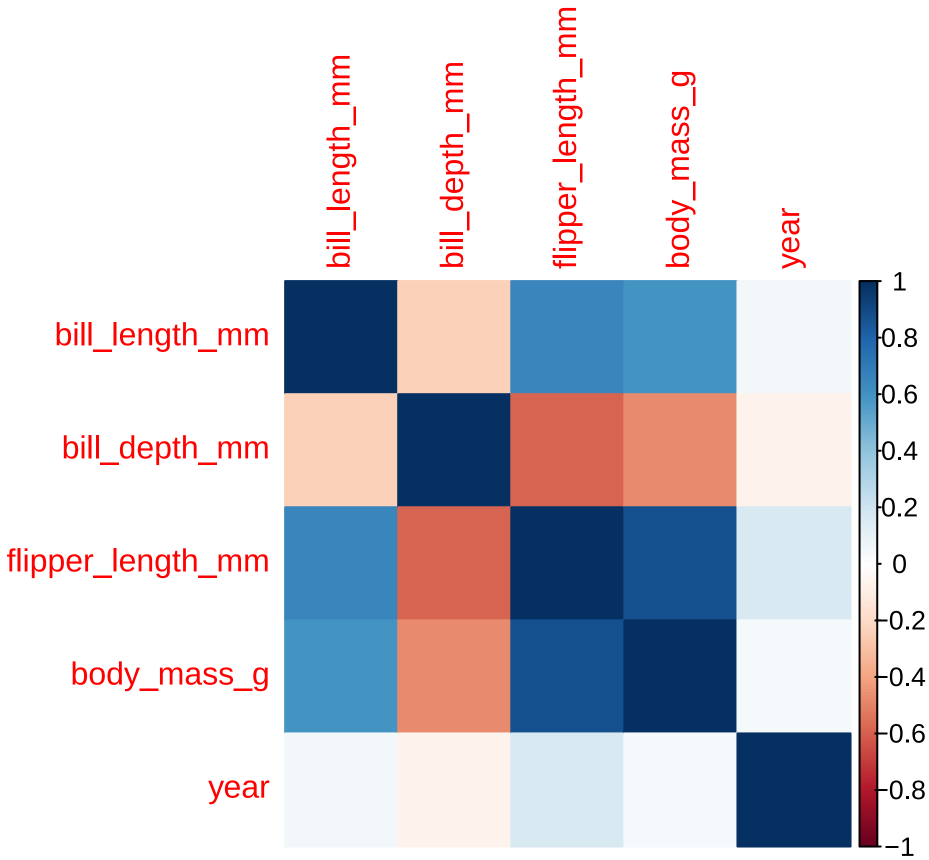
D

The numeric variables shows some interesting relationships. Are they correlated? Use the cor() and the corrplot() functions to study correlations between numeric variables (hint: try to google corrplot() to see which package you have to install to use it).

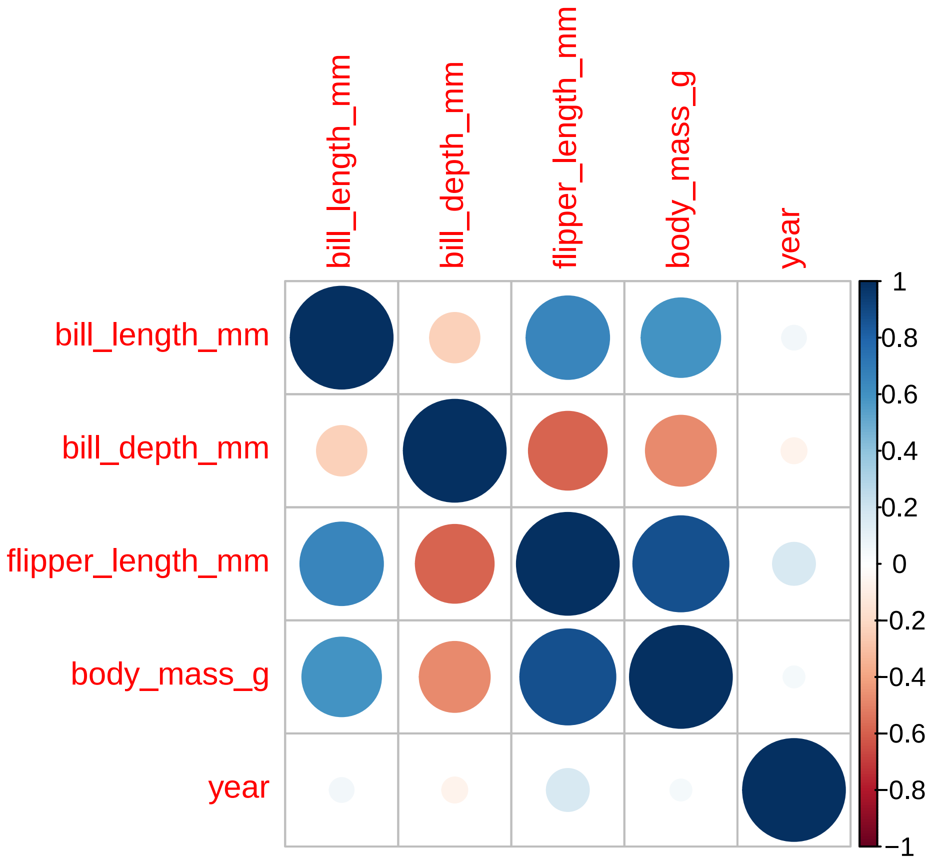
```r
library(corrplot)
correlation <- cor(penguins[, c(3:6, 8)], use = "complete.obs")
corrplot(correlation, method = "number")
```



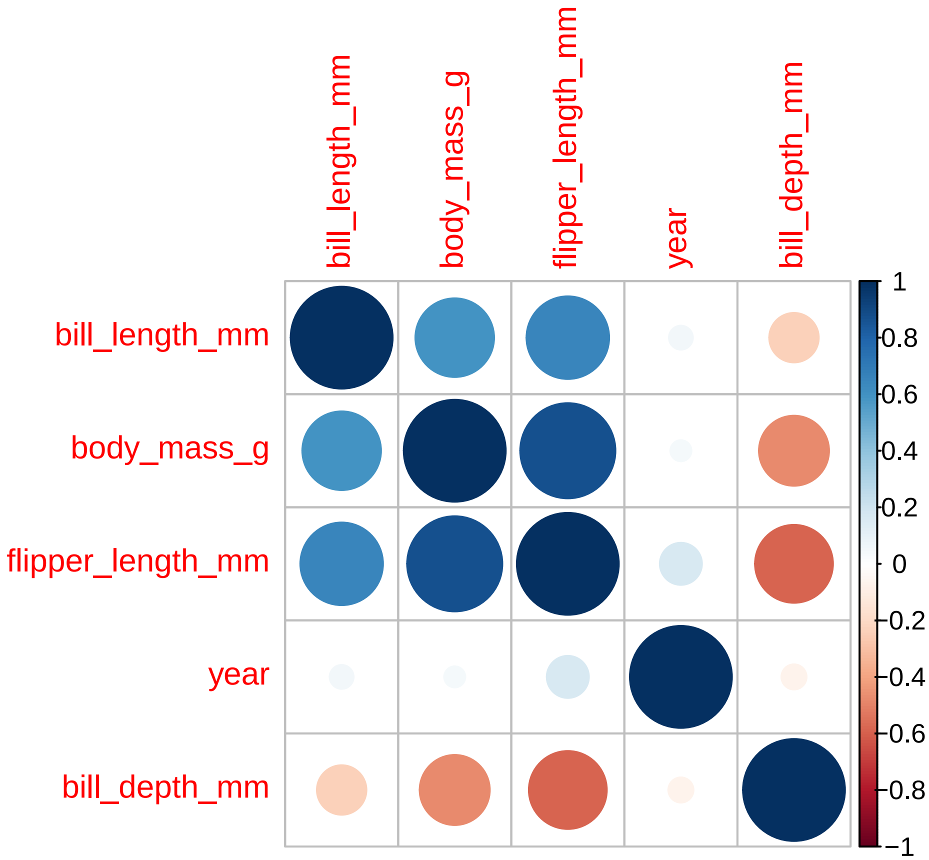

```
corrplot(correlation, method = "color")
```




```
corrplot(correlation)
```




```r
corrplot(correlation, order = "AOE")
```



E

Choose a pair of numeric variables, compute the correlation between them without using the cor() function (hint: remember the formula).

```
x <- penguins$flipper_length_mm
y <- penguins$body_mass_g

n <- length(x)
mean_x <- mean(x, na.rm = TRUE)
mean_y <- mean(y, na.rm = TRUE)

covariance <- sum((x - mean_x) * (y - mean_y), na.rm = TRUE) / (n - 1)
sd_x <- sqrt(sum((x - mean_x)^2, na.rm = TRUE) / (n - 1))
sd_y <- sqrt(sum((y - mean_y)^2, na.rm = TRUE) / (n - 1))

correlation <- covariance / (sd_x * sd_y)
cat(
  "The correlation between flipper length and body mass is: ",
  correlation, "\n"
)
```

```
## The correlation between flipper length and body mass is:  0.8712018
```

F

Plot the scatter plot for bill length vs. bill depth. Color the points by species. Use the function geom_smooth(formula = "y ~ x") to add a line to represent the linear relationship between the two variables. Then, again, use geom_smooth(formula = "y ~ x") colored by species. What are you noticing?

```r
ggplot(
  penguins,
  aes(
    x = penguins$bill_length_mm,
    y = penguins$bill_depth_mm,
    color = penguins$species
  )
) +
  geom_point() +
  geom_smooth(
    formula = "y ~ x",
    method = "lm",
    aes(color = penguins$species),
    se = FALSE
  ) +
  scale_color_manual(values = c("darkorange", "purple", "cyan4")) +
  labs(
    title = "Bill length vs. bill depth",
    x = "Bill length (mm)",
    y = "Bill depth (mm)",
    color = "Species"
  )
```