HW5

Sneha Karanjai

2022-09-26

Contents

Purpose and Data

The goal of this homework is to test and strengthen our knowledge and understanding of Summarizing Data in R. We will use the horseshoe crab data set. The data set has the following variables

- y : whether the female crab has a satellite
- satell : number of satellites
- color : female crab's color (2 = "light", 3 = "medium", 4 = "dark", and 5 = "darker")
- spine : spine condition (1 = "both good", 2 = "one worn or broken", and 3 = "both worn or broken")
- weight: female crab weight (g)
- width: female carapace width (cm)

Tasks

1. Read in data

A look at the data shows that there are varied delimiters between values. For this, we will use the `read_table()` function from the `readr` package. `read_table()` is designed to read the type of textual data where each column is separated by one (or more) columns of space. We notice that this results in an empty column in the end because of the additional space character at the end of each `y` values so we use the `select()` function to only select the columns we will work with.

```
# Reading in the data
library(readr)
library(dplyr)
```

```
##
## 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
```

```
data <- read_table("crabs.txt", col_names = TRUE) %>%
  select(c(color, spine, width, satell, weight, y))
```

```
## Warning: Missing column names filled in: 'X7' [7]
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   color = col_double(),
##   spine = col_double(),
##   width = col_double(),
##   satell = col_double(),
##   weight = col_double(),
##   y = col_double(),
##   X7 = col_logical()
## )
```

```
data
```

```
## # A tibble: 173 x 6
##    color spine width satell weight     y
##    <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>
##  1     3     3  28.3      8   3050     1
##  2     4     3  22.5      0   1550     0
##  3     2     1  26        9   2300     1
##  4     4     3  24.8      0   2100     0
##  5     4     3  26        4   2600     1
##  6     3     3  23.8      0   2100     0
##  7     2     1  26.5      0   2350     0
##  8     4     2  24.7      0   1900     0
##  9     3     1  23.7      0   1950     0
## 10     4     3  25.6      0   2150     0
## # ... with 163 more rows
```

The next task is to convert the columns that needs to be in the factor format. Factor in R is a variable used to categorize and store the data, having a limited number of different values. The three variables that need to be converted to factor are : `color`, `spine`, `y`. The raw data has these coded with integers. We will convert them to factors and then store the actual interpretation of the integer values.

```
# convert columns to factors
factor_cols <- c("color", "spine", "y")
data[factor_cols] <- lapply(data[factor_cols], factor)

levels(data$color) <- c("2" = "light",
                        "3" = "medium",
                        "4" = "dark",
                        "5" = "darker")

levels(data$spine) <- c("1" = "Both Good",
                        "2" = "One Worn/Broken",
                        "3" = "Both Worn/Broken")

levels(data$y) <- recode(data$y,
                         "0" = "At least 1 Sattelite",
                         "1" = "No Sattelite")

data
```

```
## # A tibble: 173 x 6
##    color  spine            width satell weight y
##    <fct>  <fct>            <dbl>  <dbl>  <dbl> <fct>
##  1 medium Both Worn/Broken  28.3      8   3050 At least 1 Sattelite
##  2 dark   Both Worn/Broken  22.5      0   1550 No Sattelite
##  3 light  Both Good         26        9   2300 At least 1 Sattelite
##  4 dark   Both Worn/Broken  24.8      0   2100 No Sattelite
##  5 dark   Both Worn/Broken  26        4   2600 At least 1 Sattelite
##  6 medium Both Worn/Broken  23.8      0   2100 No Sattelite
##  7 light  Both Good         26.5      0   2350 No Sattelite
##  8 dark   One Worn/Broken   24.7      0   1900 No Sattelite
##  9 medium Both Good         23.7      0   1950 No Sattelite
## 10 dark   Both Worn/Broken  25.6      0   2150 No Sattelite
## # ... with 163 more rows
```

2. Creating two-way contintency table between the satellite and spine variables.

Contingency tables are very useful to condense a large number of observations into smaller to make it easier to maintain tables. A contingency table shows the distribution of a variable in the rows and another in its columns. They are a way of summarizing categorical variables. in R, to create a contintency table, we use the `table()` function. `table()` uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels.

```
# two-way contintency table
table(data$y, data$spine)
```

```
##
##                           Both Good One Worn/Broken Both Worn/Broken
##   No Sattelite                11                8                43
##   At least 1 Sattelite        26                7                78
```

The contintency table shows that there are :

- 11 Female Crabs with both good spines but no Sattelite.
- 26 Female Crabs with both good spines and Atleast 1 Sattelite.
- 8 Female Crabs with One worn/broken spine and no Sattelite.
- 7 Female Crabs with One worn/broken spine and Atleast 1 Sattelite.
- 43 Female Crabs with both worn/broken spines and no Sattelite.
- 78 Female Crabs with both worn/broken spines and Atleast 1 Sattelite.

3. Creating three way table between the color, spine, and satellite variables

Similar to two-way contintency table, we use the `table()` function. The order matters here. The table will be split into the number of the categories of the third variable mentioned. In this case, it is the `y` variable.

```
tb <- table(data$color, data$spine, data$y)
tb
```

```
## , ,  = No Sattelite
##
##
##          Both Good One Worn/Broken Both Worn/Broken
##   light          2               0                1
##   medium         8               5               13
##   dark           0               2               16
##   darker         1               1               13
##
## , ,  = At least 1 Sattelite
##
##
##          Both Good One Worn/Broken Both Worn/Broken
##   light          7               2                0
##   medium        16               3               50
##   dark           3               2               21
##   darker         0               0                7
```

To print out a two-way table between spine and satellite for crabs with 'darker' color we use slicing/indexing. 'darker' is the third index in the contintency table. So we choose all spine and satellite values where color=darker.

```
tb[4, ,]
```

```
##
##                    No Sattelite At least 1 Sattelite
##   Both Good                   1                    0
##   One Worn/Broken             1                    0
##   Both Worn/Broken           13                    7
```

To interpret the contintency table :

- There is a female darker crab with both good spines and no satellite.
- There is a female darker crab with one worn/broken spines and no satellite.
- There are 13 female darker crabs with both worn/broken spines and no satellite.

Similarly we can also interpret the "At least 1 Satellite" column.

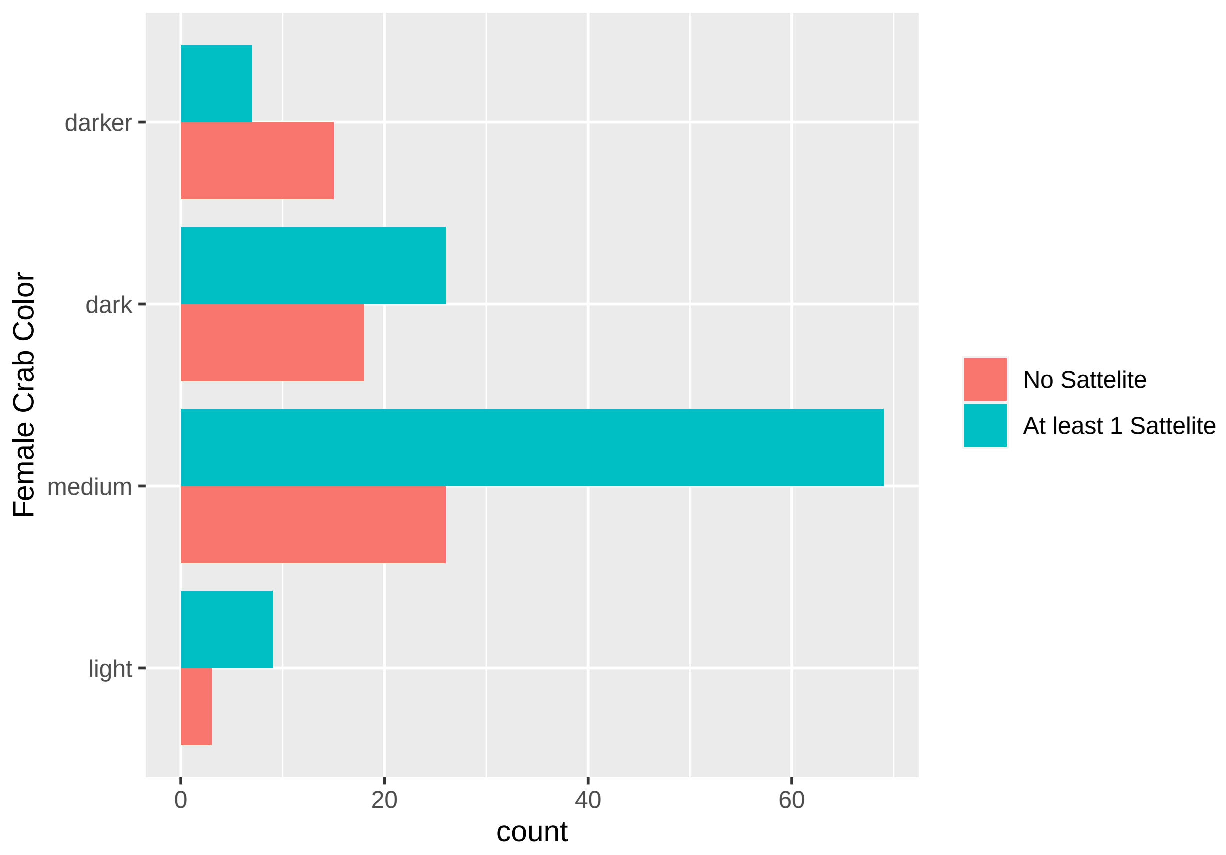
4. Recreating Single Plots

We use the `ggplot()` function to create a baseline graph. This specifies the data to use, what the x and y axis would be. Sometimes it also specifies which variable to use to fill the graph. Additionally, we use layers with `+` to add on to the baseline graph.

1. Satellite distribution among crabs of color

The `aes()` is the aesthetic function where we specify the "x-axis" variable and "fill" to fill the colors of the bars with the y variable. `geom_bar()` plots a bar graph and the "width" specifies the width of each bar. `labs()` is a function to specify the x-axis label. `position_dodge()` helps in building a side by side plot over a stacked plot. In the `theme()` function, we ensure that the legend has no title to match the aesthetics of the given document. `coord_flip()` flips the default vertical bar plot to a horizontal one.

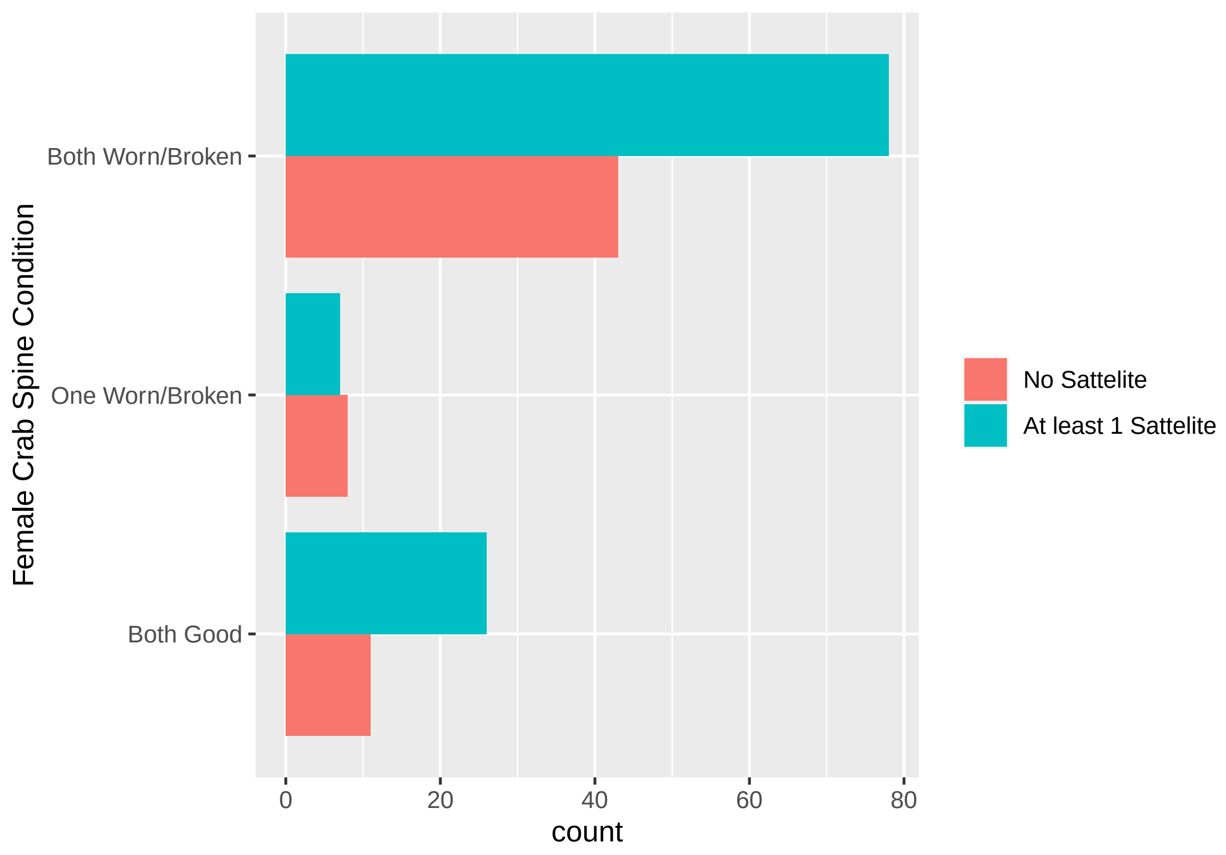
```r
library(ggplot2)
ggplot(data = data, aes(x = color, fill = y)) +
  geom_bar(position = position_dodge(), width = 0.85) +
  labs(x = "Female Crab Color") +
  theme(legend.title = element_blank()) +
  coord_flip()
```



Clearly, the majority of female crabs lie in the medium color with most of the medium color crabs having at least 1 Satellite. It is also note-worthy that except darker color, all female crabs in majority have at least 1 satellite. "Darker" is the only color where the number of female crabs have no satellite over female crabs having atleast 1 satellite.

2 Distribution of Satellite by the spine condition.

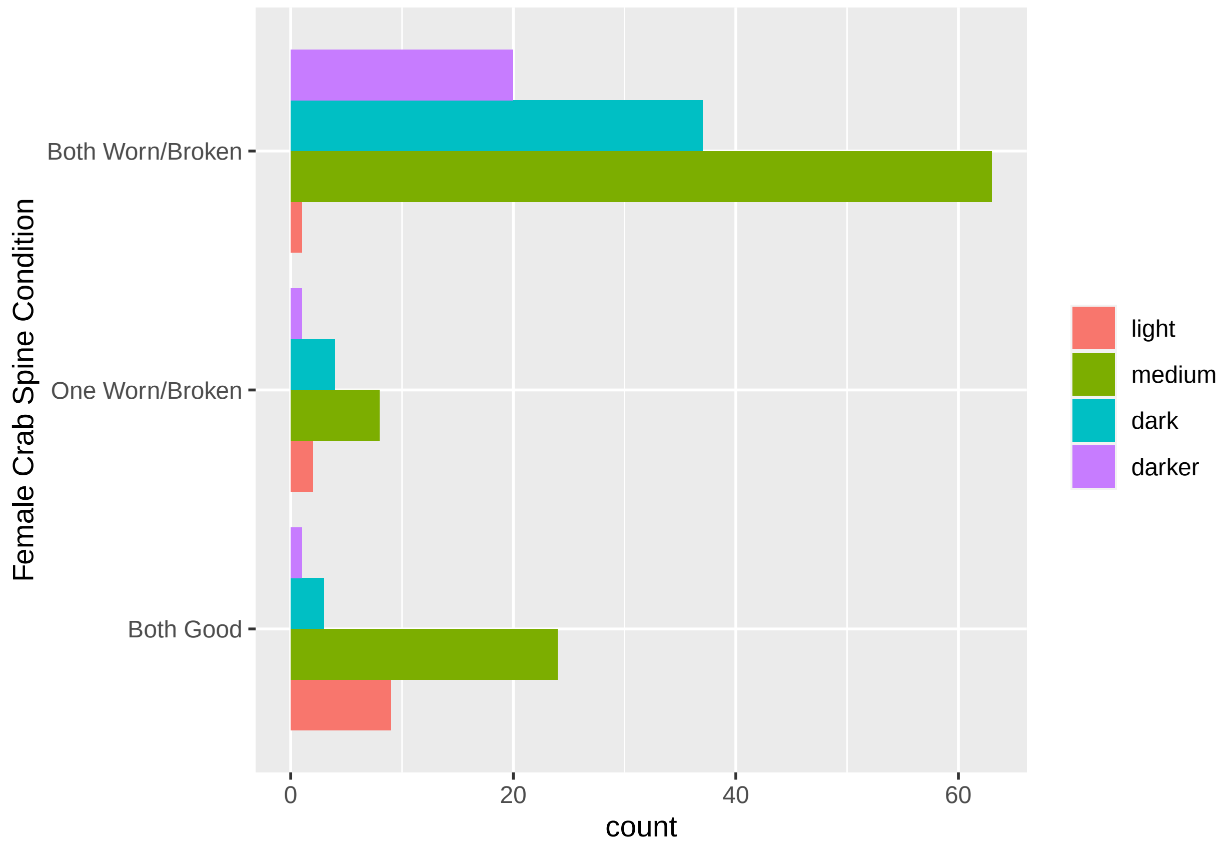
```
ggplot(data = data, aes(x = spine, fill = y)) +
  geom_bar(position = position_dodge(), width = 0.85) +
  labs(x = "Female Crab Spine Condition") +
  theme(legend.title = element_blank()) +
  coord_flip()
```



This is interesting for a person with no prior crab knowledge. Female crabs with both their spines broken/worn out have more satellites than female crabs with at least 1 working spine.

3. Distribution of female crabs according to their color and spine condition

```
ggplot(data = data, aes(x = spine, fill = color)) +
  geom_bar(position = position_dodge(), width = 0.85) +
  labs(x = "Female Crab Spine Condition") +
  theme(legend.title = element_blank()) +
  coord_flip()
```



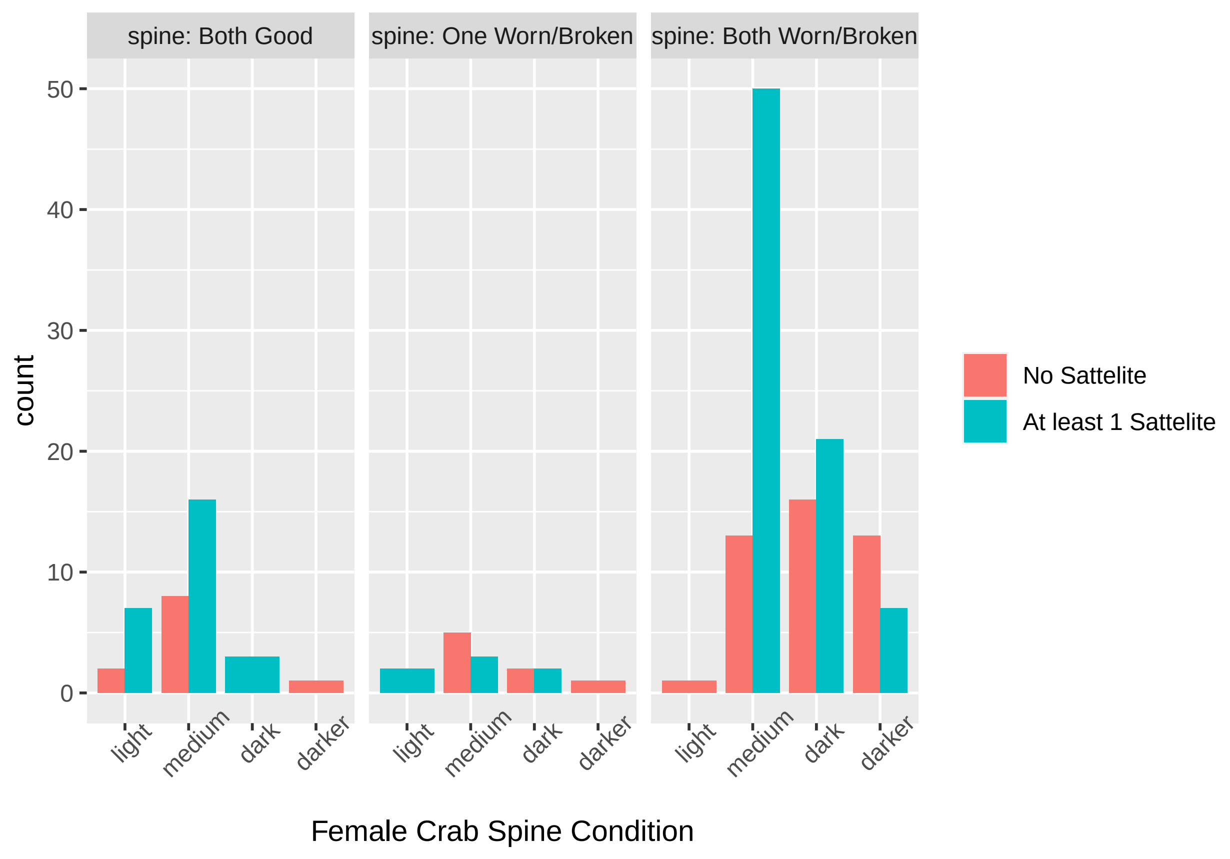
Satellites seem to have a preference with medium colored female crabs. light crabs with both spines broken/worn out, darker crabs with one broken/work out spine, and darker crabs with both good spines have the lease number of count of female crabs.

5. Recreating Side by side plots

1. Spine condition vs Satellite

`facet_wrap()` wraps a 1d sequence of panels into 2d. It divides each of the label of "spine" variable to a side by side individual plots. We also use the axis.text.x to tilt the x axis labels by an angle of 45 degrees.

```
ggplot(data = data, aes(x = color, fill = y)) +
  geom_bar(position = position_dodge(), width = 0.85) +
  facet_wrap(vars(spine), labeller = label_both) +
  labs(x = "Female Crab Spine Condition") +
  theme(legend.title = element_blank(),
        axis.text.x = element_text(angle = 45))
```



Color and Spine conditions are an important factor deciding if a female crab will have a Satellite or not. Medium is definitely a popular color that attracts fertilization with the satellites. Surprisingly, female crabs with both broken spines have more success with satellites than female crabs with functioning spines. There are no female crabs who have darker color and both working spines with atleast 1 satellite.

6. Summary statistics

To create this custom summary statistics, we will utilize `group_by()` function and `summarise()`. We group by color and y, and summarise by mean, standard deviation, median, IQR of the weight column.

```
data %>%
  group_by(color, y) %>%
  summarise(Avg = mean(weight), Sd = sd(weight), Median = median(weight), IQR = IQR(weight))
```

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

```
## # A tibble: 8 x 6
## # Groups:   color [4]
##   color  y                      Avg    Sd Median   IQR
##   <fct>  <fct>                <dbl> <dbl>  <dbl> <dbl>
## 1 light  No Sattelite         2525   152.   2600  138.
## 2 light  At least 1 Sattelite 2664.  442.   2700  650
## 3 medium No Sattelite         2242.  482.   2200  512.
## 4 medium At least 1 Sattelite 2649.  615.   2700  800
## 5 dark   No Sattelite         1907.  348.   1900  462.
## 6 dark   At least 1 Sattelite 2571.  495.   2575  719.
## 7 darker No Sattelite          2162.  436.   2150  650
## 8 darker At least 1 Sattelite 2200   523.   2100  200
```

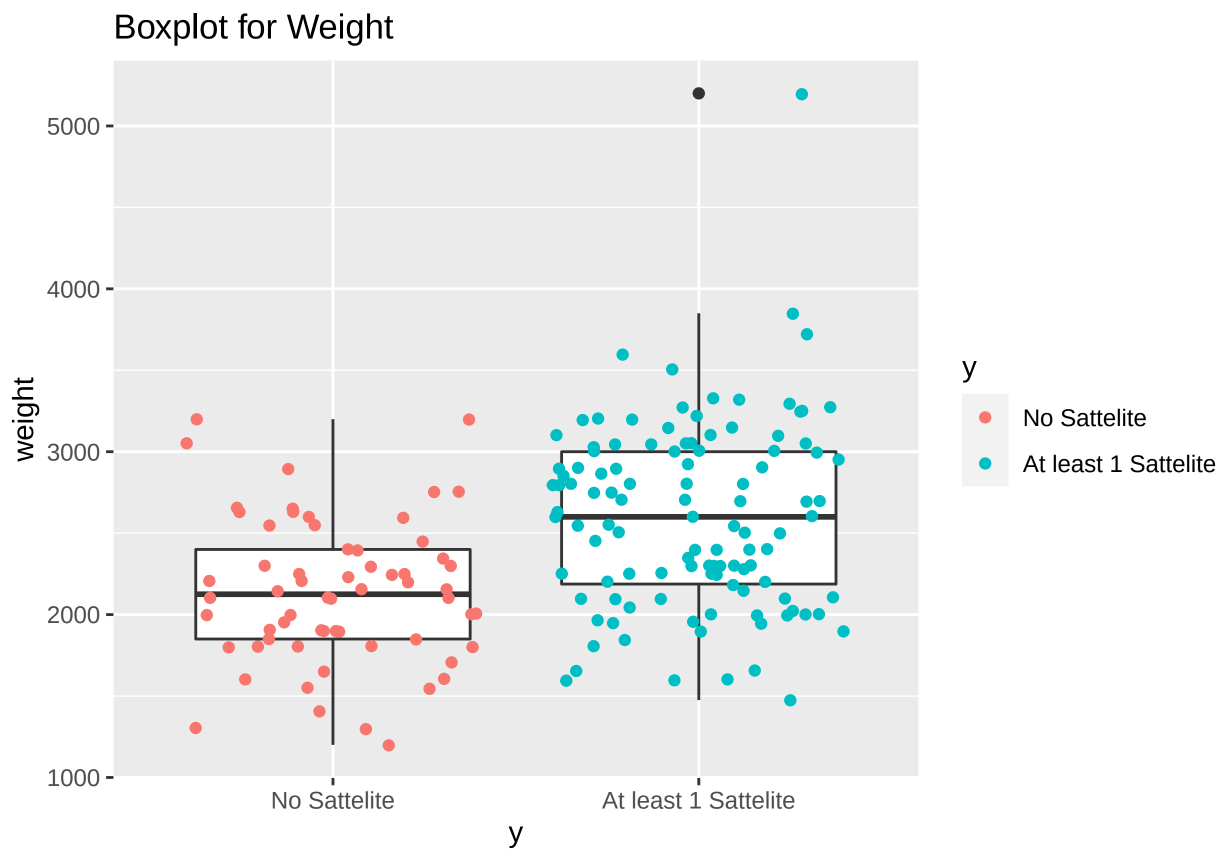
Female light crabs with Atleast 1 Sattelite have the highest average weight of 2664g. This category also has a high standard deviation of 442 which means that the raw data is spread out and farther away from the mean of 2664g. The median is the middle value of the female crabs of light color and having at least 1 satellite is 2700g which means half of the data lies beneath this value and half of the data lies above this. The distance between the first quartile and the third quartile of this category is 650.

7. Boxplots

1. Boxplot for weight

The jitter geom is a convenient shortcut for geom_point(position = "jitter"). It adds a small amount of random variation to the location of each point, and is a useful way of handling overplotting caused by discreteness in smaller datasets. We achieve this in `geom_jitter()`.

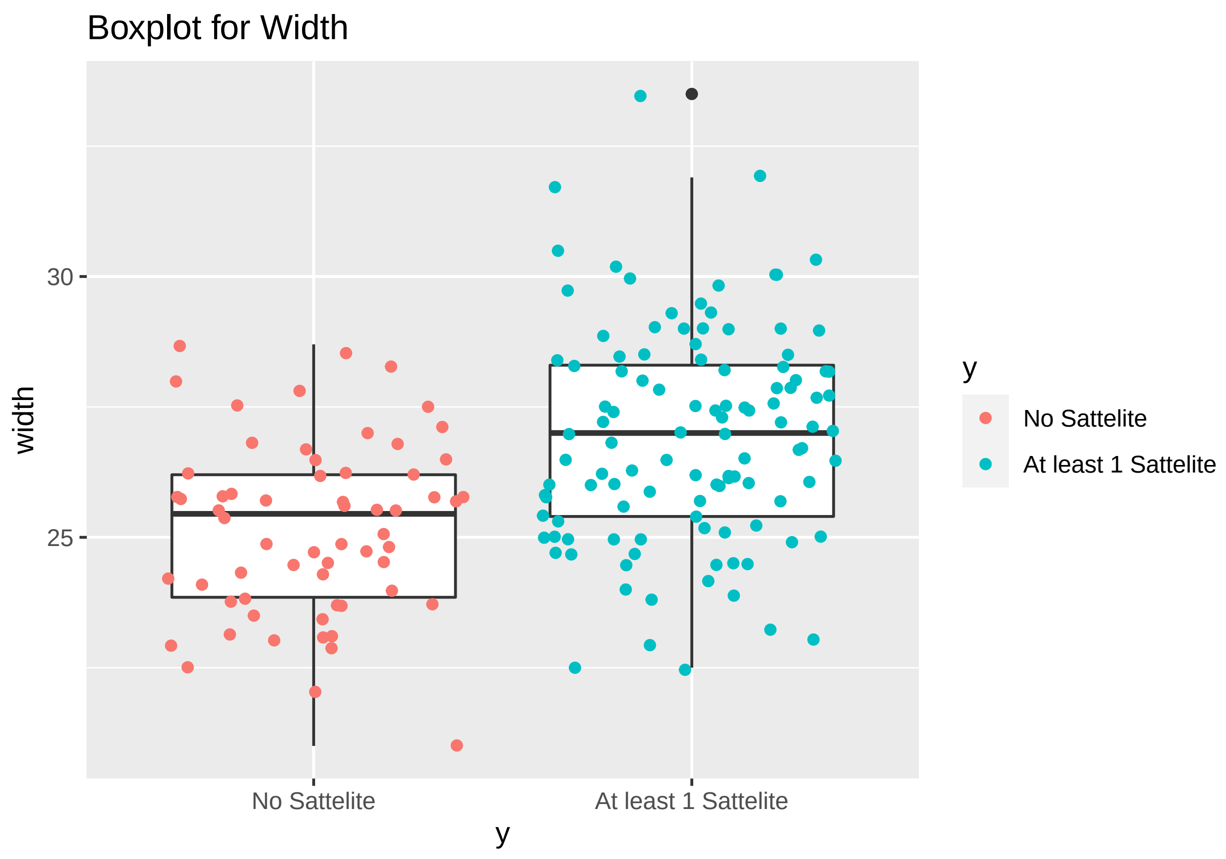
```
ggplot(data = data, aes(x = y, y = weight)) +
  geom_boxplot() +
  geom_jitter(aes(colour = y)) +
  labs(title = "Boxplot for Weight")
```



There is more uncertainty in weights of female crabs with at least 1 satellite. We also notice an outlier in this category.

2. Boxplot for width

```
ggplot(data = data, aes(x = y, y = width)) +
  geom_boxplot() +
  geom_jitter(aes(colour = y)) +
  labs(title = "Boxplot for Width")
```



There is more uncertainty in width of female crabs with at least 1 satellite. We also notice an outlier in this category.

8. Correlation and Scatterplot

We use the `cor()` function to find the correlation between weight and width.

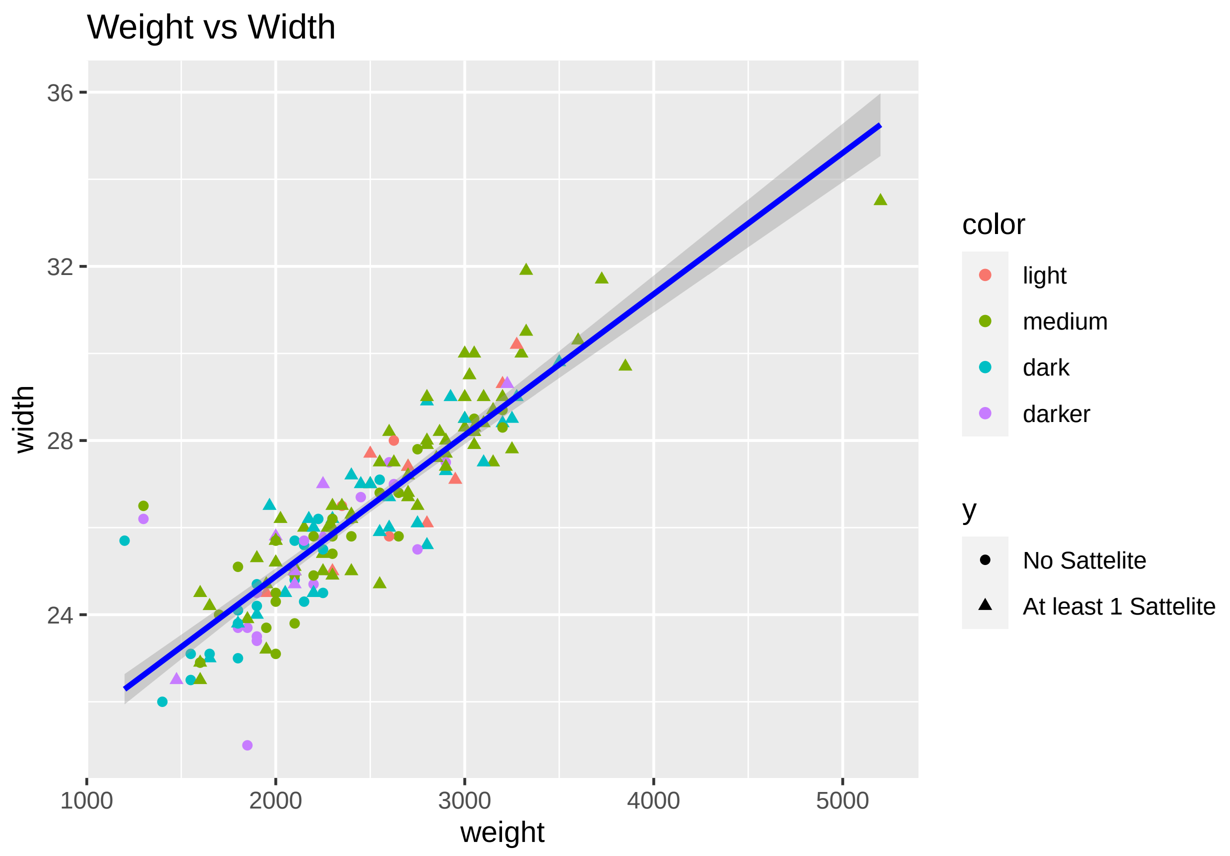
```
cor(data$weight, data$width)
```

```
## [1] 0.8868715
```

The reported correlation is 0.887. This is a positive strong correlation. This means as weight of a female crab increases the width also tends to increase.

```r
ggplot(data = data, aes(x = weight, y = width, colour = color)) +
  geom_point(aes(shape = y)) +
  geom_smooth(method = lm, col = "Blue") +
  labs(title = "Weight vs Width")
```

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



Weight and width show a near perfect positive correlation where as weight increases, width also increases. The scatter plot shows a dominant of medium female crabs with atleast 1 satellite. There are some data points that does not follow this strong relationship like dark and darker crabs with no satellites. We also notice an outlier that could be influencing the best fit line. It would be interesting to see how the line changes if this outlier is removed.