Week 6 - Review

Nicholas Kortessis

2025-02-19

Reviewing what we've learned in R so far

This week, we will review a little bit. Be sure to go back to previous lab activities if you need a refresher.

Object Types in R

R has many different object types. We have mostly been working with vectors and data frames.

Vectors

Here are some vectors, which are just a multiple items put together.

```
# A simple vector
vec <- c("Nicks", "BIO 380", "Lab")

# A list of 5 numbers equally spaced from 10 to 300.
my.vec <- seq(from = 10, to = 300, length = 5)

# Here is another that repeats 30 successes and 70 failures
another.vec <- rep(c("Success","Failure"), times = c(30,70))
```

You can find specific values of the vectors by using square brackets at the end of a named vector.

```
vec[2]
```

```
## [1] "BIO 380"
```

```
my.vec[3]
```

```
## [1] 155
```

```
another.vec[100]
```

```
## [1] "Failure"
```

```
another.vec[50:60]
```

```
##  [1] "Failure" "Failure" "Failure" "Failure" "Failure" "Failure" "Failure"
##  [8] "Failure" "Failure" "Failure" "Failure"
```

```
vec[-2]; vec[-c(2:3)]
```

```
## [1] "Nicks" "Lab"
```

```
## [1] "Nicks"
```

Matrices

We can also create matrices by taking vectors and organizing them by rows.

```r
a.matrix <- matrix(data = LETTERS[1:20],
                   ncol = 4,
                   nrow = 5,
                   byrow = T)
```

To get a specific value from a matrix, you need to indicate the row and the column where the item is listed. Like this.

```r
a.matrix[2,3]
```

```
## [1] "G"
```

```r
a.matrix[2,]
```

```
## [1] "E" "F" "G" "H"
```

```r
a.matrix[,3]
```

```
## [1] "C" "G" "K" "O" "S"
```

```r
a.matrix[-3,c(3:4)]
```

```
##      [,1] [,2]
## [1,] "C"  "D"
## [2,] "G"  "H"
## [3,] "O"  "P"
## [4,] "S"  "T"
```

Data frames

Data frames are special matrices that have column names associated with them. Let's load some data. The data for this week is on Canvas and is called `Cod.csv`. Load in this data using the `read.csv` command.

```r
str(cod.df)
```

```
## 'data.frame':    1254 obs. of  11 variables:
##  $ Sample    : int  5 365 468 509 469 508 7 467 1162 6 ...
##  $ Intensity : int  0 0 0 0 0 0 0 0 3 0 ...
##  $ Prevalence: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Year      : int  1999 2001 2001 2001 2001 2001 1999 2001 2001 1999 ...
##  $ Depth     : int  220 193 228 122 228 122 220 228 283 220 ...
##  $ Weight    : int  40 34 40 38 46 46 52 52 66 68 ...
##  $ Length    : int  17 17 17 17 18 18 19 19 19 20 ...
##  $ Sex       : int  0 0 2 2 2 2 0 2 2 0 ...
##  $ Stage     : int  0 0 1 1 1 1 0 1 1 0 ...
##  $ Age       : int  0 1 1 1 1 1 0 1 1 0 ...
##  $ Area      : int  2 3 4 3 4 3 2 4 2 2 ...
```

This is a dataset of Norwegian Cod that were caught between the years of 1999 and 2001. Information was taken about each fish (weight, length, sex, age, and location) and they checked to see if the fish had parasites. If it had parasites, it was recorded as prevalence = 1. The way to look at the column of prevalence is by using the `$` symbol.

```r
cod.df$Prevalence
```

```
##    [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
##   [75] 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0
##  [112] 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0
```

```
##  [149] 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0
##  [186] 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1
##  [223] 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 0
##  [260] 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
##  [297] 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
##  [334] 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
##  [371] 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1
##  [408] 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
##  [445] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
##  [482] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
##  [519] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [556] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [593] 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [630] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [667] 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
##  [704] 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
##  [741] 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
##  [778] 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [815] 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
##  [852] 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
##  [889] 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
##  [926] 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
##  [963] 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
## [1000] 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
## [1037] 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
## [1074] 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1
## [1111] 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1
## [1148] 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0
## [1185] 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 0 0 0 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 1 0 1
## [1222] 1 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1
```

Some of the structure of this is not so helpful. For example, the sex of the fish is specified by a number. Let's make it clearer.

```
cod.df[cod.df$Sex == 2,"Sex"] <- "Male"
cod.df[cod.df$Sex == 1,"Sex"] <- "Female"
cod.df[cod.df$Sex == 0,"Sex"] <- "Unknown"

head(cod.df$Sex)
```

```
## [1] "Unknown" "Unknown" "Male"    "Male"    "Male"    "Male"
```

Ah. That's better. How did this work? We used two things.

1. We isolated entries we were interested in by specifying rows with a logical statement and columns named "Sex".
2. The logical statement asked with of the entries in the "Sex" column have a particular value. The answer is a bunch of TRUE/FALSE statements. The ones that are true are indexed.

The end result for the first line of code is to assign the name `"Male"` to all rows in the "Sex" column where "Sex" is equal to 2.

Logical Statements

Logical statements use the following commands

1. `==` asks "is it equal to"
2. `!=` asks "is it NOT equal to"

3. `is.na` asks "is it NA"
4. `<` asks "is it less than" (`>` greater than)
5. `<=` asks "is it less than or equal to" (`>=` greater than or equal to)
6. `&` "AND"
7. `|` "OR"

Logical statements produce a new object that is either TRUE or FALSE (or a sequence of such TRUE/FALSE outputs for vectors and matrices). These are used to subset and find particular values in vectors, matrices, and dataframes.

Subsetting

An important thing to be able to do is to subset the data frame to look at particular components of the data frame. Say we want only the males. Here are a couple of ways to do it.

```
# One way
Male1.df <- subset(cod.df, subset = (Sex == "Male"))
# An alterantive
Male2.df <- cod.df[cod.df$Sex == "Male",]

# Check to see that they are the same.
identical(Male1.df, Male2.df)
```

```
## [1] TRUE
```

Checkpoint 1: Create a dataset that does not include the individuals with the Unknown sex. Call this `clean.cod.df`. Here are some other important functions to use in R for data frames and exploratory analysis

1. `str` - summarizes the data frame
2. `dim` - gives the number of rows (first output) and columns (second output)
3. `unique` - finds all the different *kinds* of values in an object
4. `table` - counts the number of items of each kind
5. `aggregate` - applies a simple function to all the entries of a particular kind.

Here are some examples.

```
str(clean.cod.df)
```

```
## 'data.frame':    1172 obs. of  11 variables:
##  $ Sample    : int  468 509 469 508 467 1162 1085 465 663 464 ...
##  $ Intensity : int  0 0 0 0 0 3 186 0 1 0 ...
##  $ Prevalence: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Year      : int  2001 2001 2001 2001 2001 2001 2000 2001 1999 2001 ...
##  $ Depth     : int  228 122 228 122 228 283 146 228 85 228 ...
##  $ Weight    : int  40 38 46 46 52 66 92 94 100 110 ...
##  $ Length    : int  17 17 18 18 19 19 22 23 23 24 ...
##  $ Sex       : chr  "Male" "Male" "Male" "Male" ...
##  $ Stage     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Age       : int  1 1 1 1 1 1 1 1 1 2 ...
##  $ Area      : int  4 3 4 3 4 2 3 4 1 4 ...
```

```
dim(clean.cod.df)
```

```
## [1] 1172   11
```

```
unique(clean.cod.df$Sex)
```

```
## [1] "Male"   "Female"
```

```r
table(clean.cod.df$Sex)
```

```
##
## Female   Male
##    574    598
```

```r
table(clean.cod.df$Sex, clean.cod.df$Stage)
```

```
##
##             1   2   3   4
##   Female 430  58   1  85
##   Male   405 149   1  43
```

```r
aggregate(Length ~ Sex, data = clean.cod.df, sd)
```

```
##      Sex   Length
## 1 Female 14.60530
## 2   Male 13.13844
```

Checkpoint 2: Use these functions to write code that answers the following question. a. How many fish are in the total data set (i.e., including the fish with unknown sex)?

b. What fraction are male, female, and unknown?

c. How many areas are in the study?

d. What are the average weights of fish for each different sex?

e. What is the prevalence of parasite infections by fish sex? Is parasite infection more common in one sex?

f. What is the prevalence of parasite infections by area? Is parasite infection more common in some areas? Which ones?

Plotting

We talked about different kinds of plots. We could

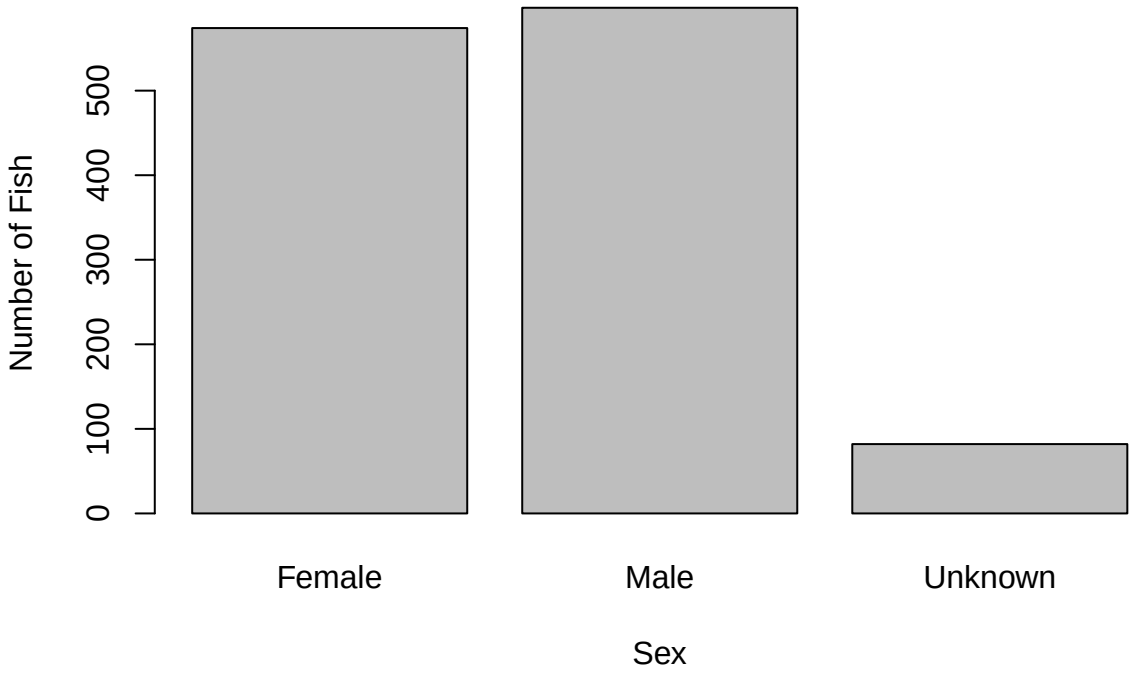
- plot amounts (or proportions) in different groups
- distributions of different groups
- relationships between multiple groups

Let's show some examples of each with this data set.

Amounts

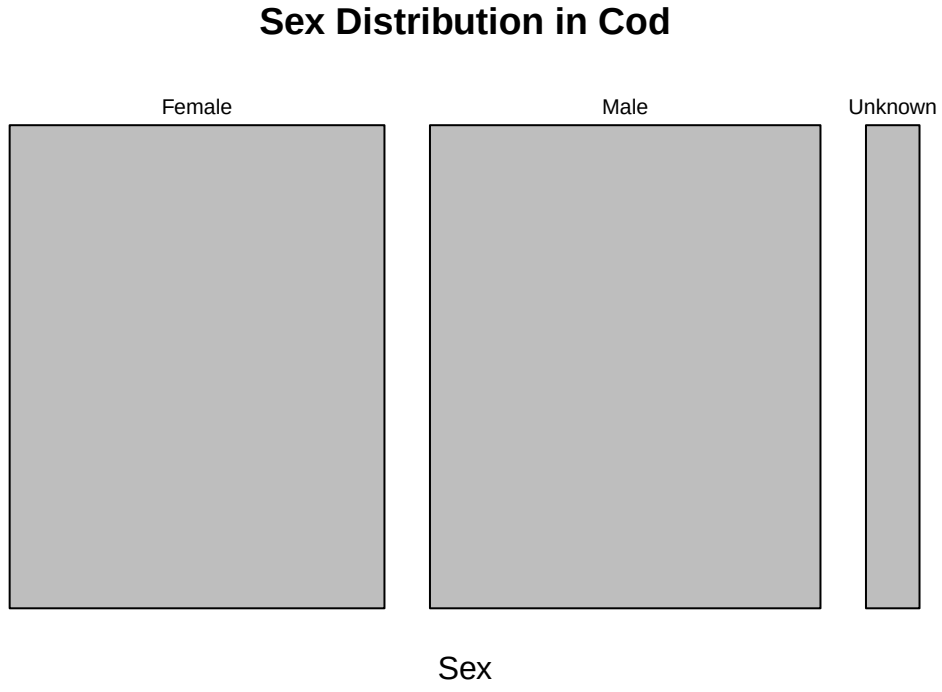
Lets look at absolute amounts. I asked how many there are of each sex. Let's visualize that data by throwing the `table` output into the `barplot` function.

```r
barplot(table(cod.df$Sex),
        xlab = 'Sex',
        ylab = 'Number of Fish')
```



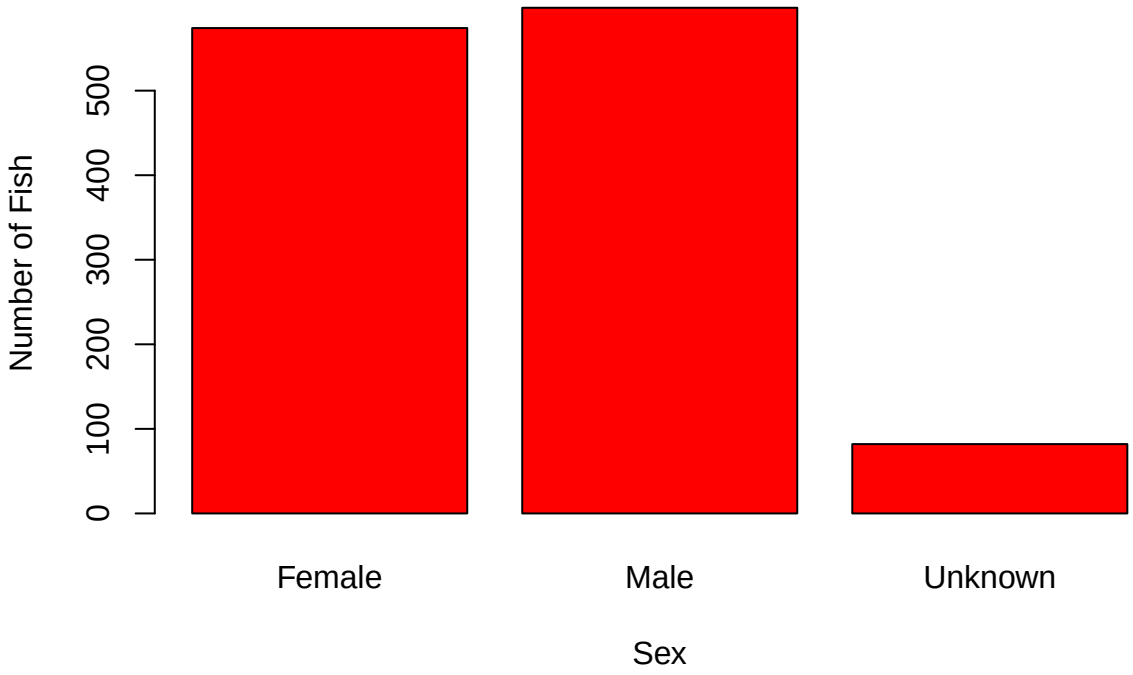
If we wanted this in versions of proportions, we could use a mosaic plot.

```
mosaicplot(Sex ~ 1, data = cod.df,
           main = 'Sex Distribution in Cod')
```



This is a good time to note that color can be very helpful. R takes normal color names to indicate colors. For example, let's make the barplots red.

```
barplot(table(cod.df$Sex),
        xlab = 'Sex',
        ylab = 'Number of Fish',
        col = 'red')
```



Maybe we want different colors for different groups. We could come up with color schemes ourselves, but there are a bunch out there that are very helpful. Let's use the package `ViridisLite`, which has some of my favorite color schemes (you can see them here) that are designed to be color-blind friendly and have continuous gradation in grayscale.

```
install.packages("viridisLite")
```

Now load the package.

```
library(viridisLite)
```

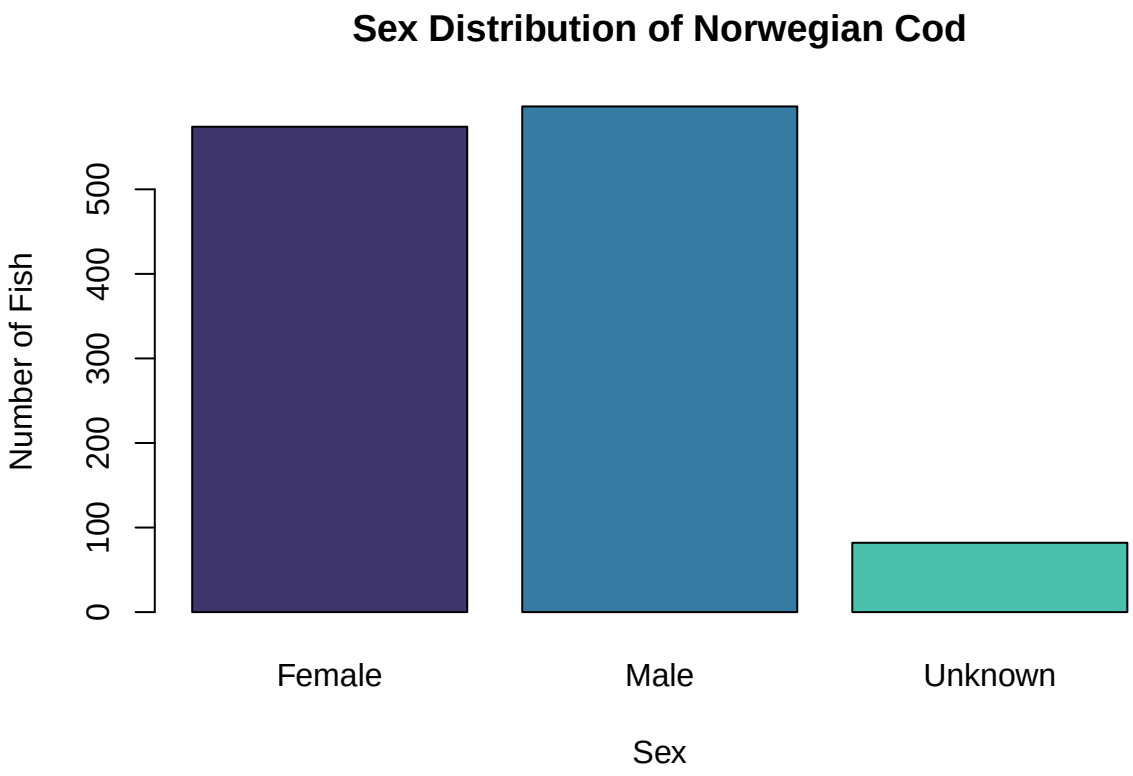
Let's use the function `mako`. The function just needs the number of colors you want. We want one for each sex in our data set, which we can get with `length(unique(x))`. We will also pass the arguments `begin` and `end`. These color schemes have continuous color gradation and span a wide range. You can narrow the range of colors to use. Here we will not use the 50% of the most extreme colors in the color scheme (check the function notation by writing `?viridis` or go to the website to learn more about what this means).

```
(our.colors <- mako(length(unique(cod.df$Sex)),
                    begin = 0.25,
                    end = 0.75))
```

```
## [1] "#3E356BFF" "#357BA2FF" "#49C1ADFF"
```

These numbers and symbols are one of many ways that R encodes color. We can just pass this to the color command in our plot.

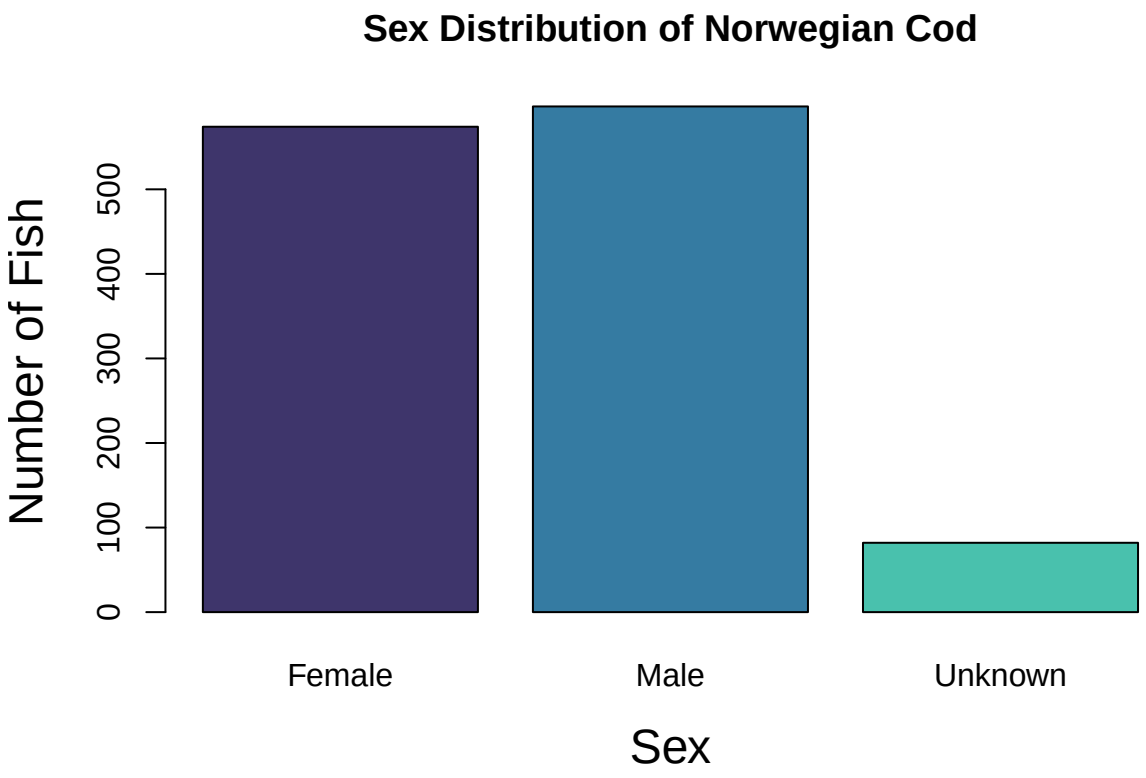
```
barplot(table(cod.df$Sex),
        xlab = 'Sex',
        ylab = 'Number of Fish',
        col = our.colors,
        main = 'Sex Distribution of Norwegian Cod')
```



That looks better.

If we want to increase the size of the axis labels, we use the argument `cex.lab`. If we want to increase the size of the axis values, we use `cex.axis`. Let's make the label names 50% bigger.

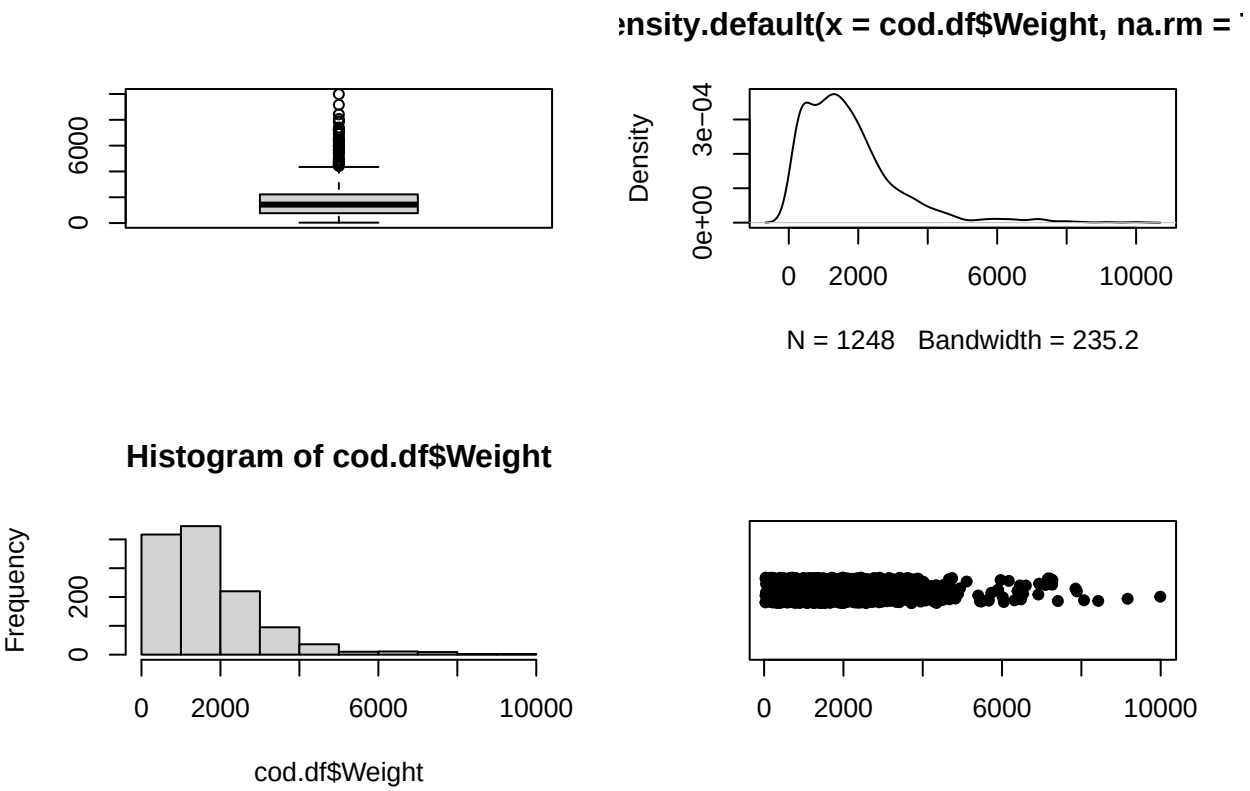
```
barplot(table(cod.df$Sex),
        xlab = 'Sex',
        ylab = 'Number of Fish',
        col = our.colors,
        main = 'Sex Distribution of Norwegian Cod',
        cex.lab = 1.5)
```



Distributions

We can use *density plots, boxplots, histograms, strip plots,* and others to illustrate distributions. Let's look at the distribution of fish sizes overall (ignoring sex). We will plot the four together using the plotting function `par(mfrow = c(nrow,ncol))`. This will be four panels that are 2x2.

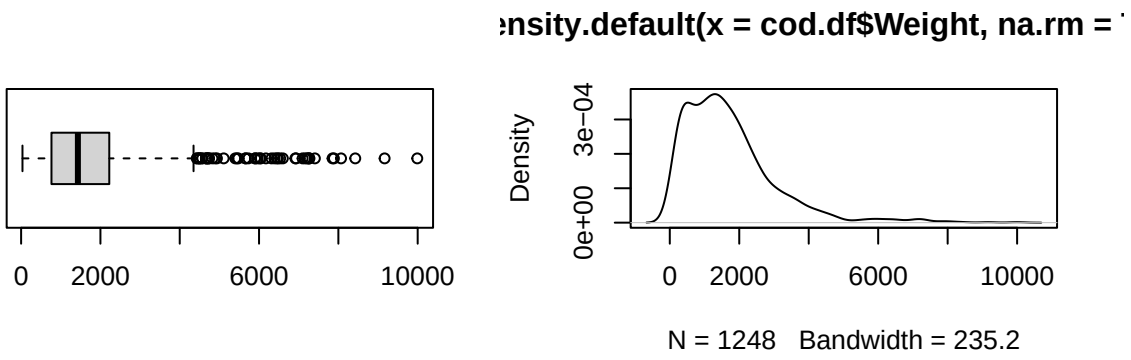
```
par(mfrow = c(2,2))
boxplot(cod.df$Weight)
plot(density(cod.df$Weight, na.rm = TRUE))
hist(cod.df$Weight)
stripchart(cod.df$Weight,
           method = "jitter",
           pch = 19)
```

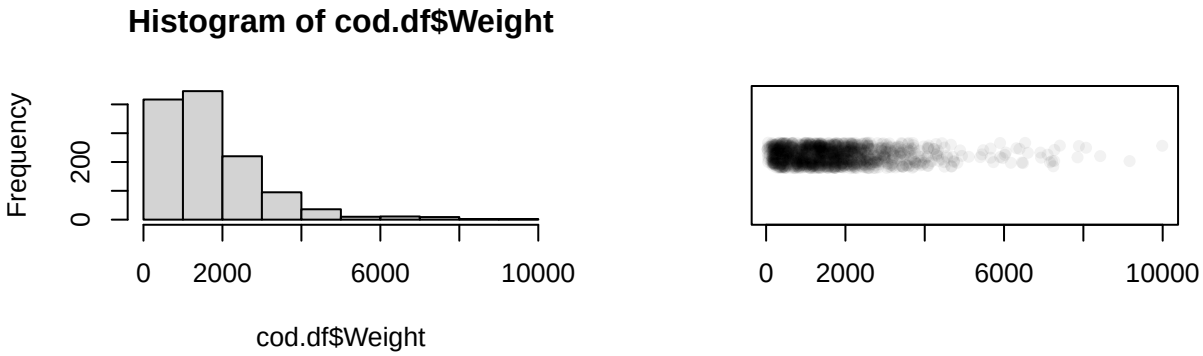


Two things.

1. I like them all to be either horizontal or vertically aligned when all in one panel. That means we need to rotate the boxplot to be horizontal to match the others. We can do this with `horizontal = TRUE`.
2. There are so many data points that the stripchart is ineffictive because they are all right on top of each other. We can fix this by specializing a transparent color with the function `rgb`. `rgb` creates a color from red, green, and blue hues and also takes an `alpha` value that gives how transparent the color should be. Let's see how these work.

```
par(mfrow = c(2,2))
boxplot(cod.df$Weight, horizontal = TRUE)

plot(density(cod.df$Weight, na.rm = TRUE))

hist(cod.df$Weight)

stripchart(cod.df$Weight,
           method = "jitter",
           pch = 19,
           col = rgb(0,0,0, alpha = 0.05))
```





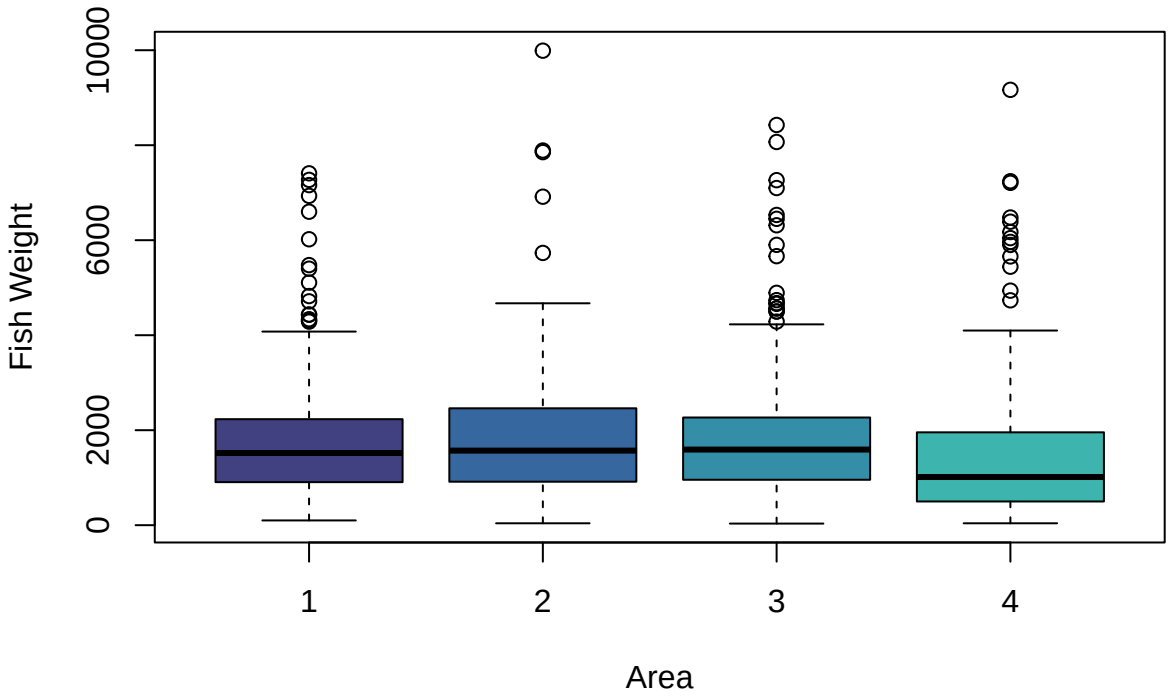
A summary of these weights is

```
summary(cod.df$Weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    34.0   765.5  1432.0  1704.3  2222.5  9990.0       6
```

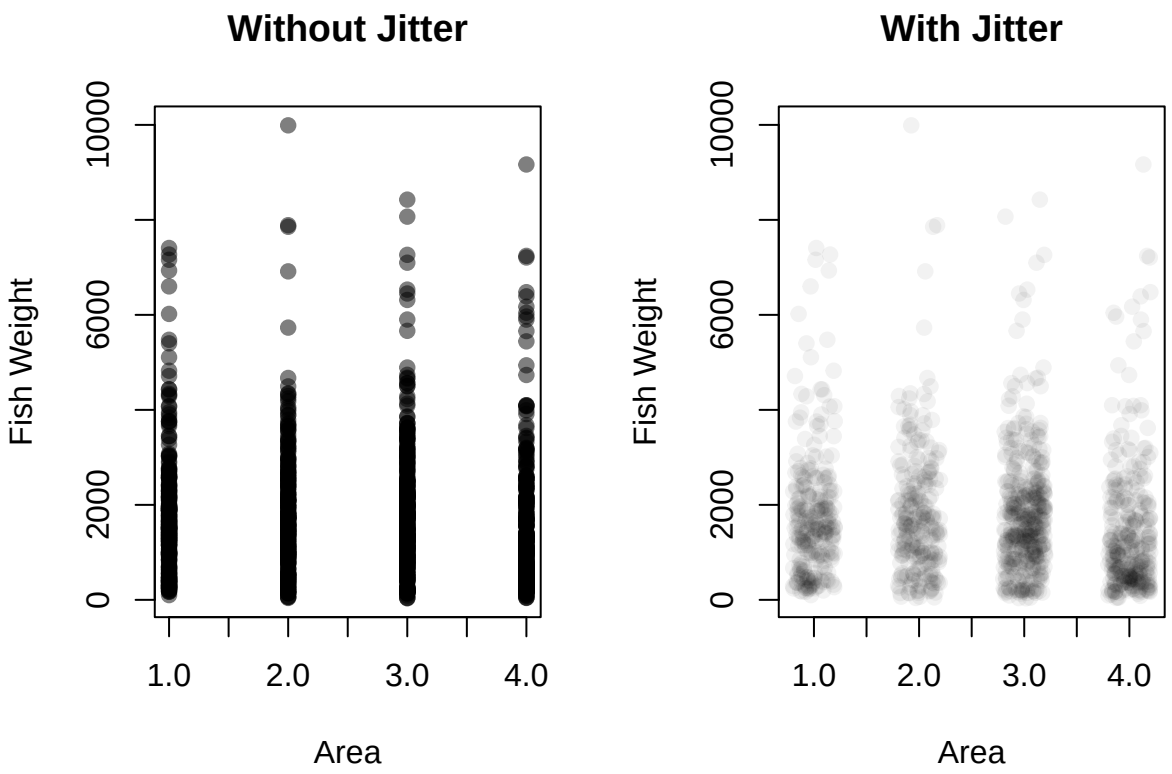
Sometimes we want to plot multiple distributions simultaneously. For example, we might want to know the weight of fish in different areas. In this case, use the function notation with ~ to mean "as a function of". For example, `Weight~Area` tells R to look at weight as a function of area.

```
boxplot(cod.df$Weight ~ cod.df$Area,
        xlab = 'Area', ylab = 'Fish Weight',
        col = mako(n = length(unique(cod.df$Area)), begin = 0.3, end = 0.7))
```



In this case, because `Area` is numerical, we could also plot them as individual points to make a stripchart. All the points will lie right on top of each other within the same area. To spread them out a bit, we can use the function `jitter` to add a bit of variation to the areas. This panel shows it with and without the jittering effect.

```
par(mfrow = c(1,2))
plot(cod.df$Area, cod.df$Weight,
     xlab = 'Area', ylab = 'Fish Weight',
     pch = 19, col = rgb(0,0,0,alpha = 0.5),
     main = "Without Jitter")
plot(jitter(cod.df$Area), cod.df$Weight,
     xlab = 'Area', ylab = 'Fish Weight',
     pch = 19, col = rgb(0,0,0,alpha = 0.05),
     main = 'With Jitter')
```



To see how this works, look at the difference between the `Area` vectors with and without jitter.

```
head(cod.df$Area)
```

```
## [1] 2 3 4 3 4 3
```

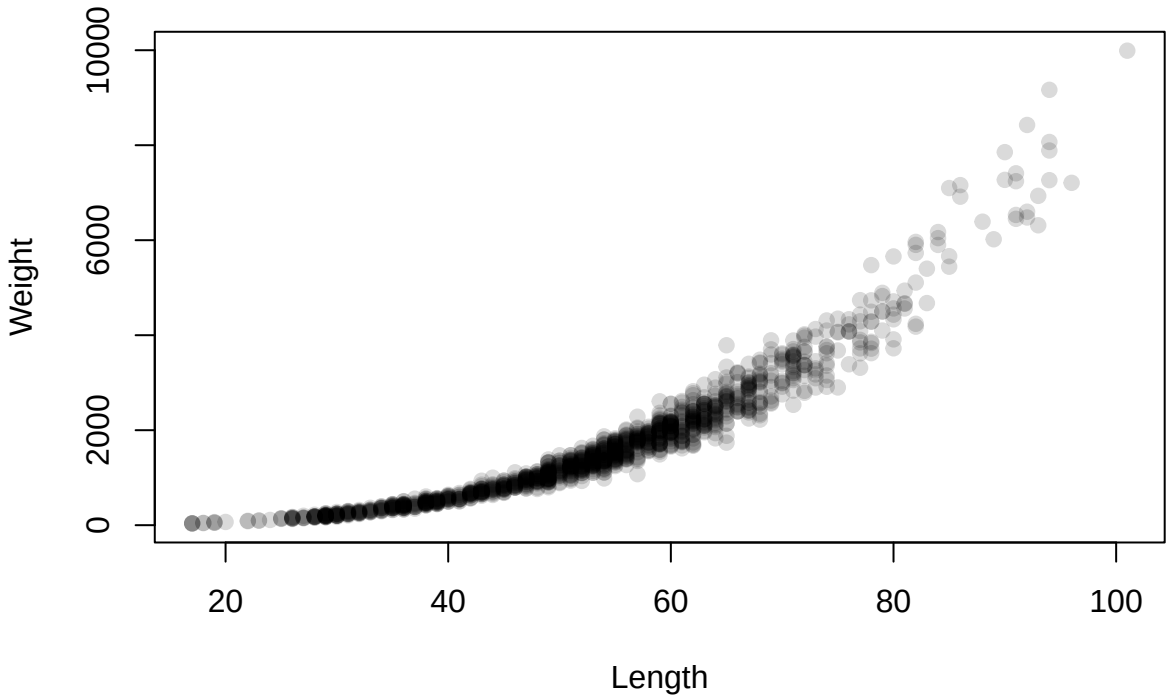
```
head(jitter(cod.df$Area))
```

```
## [1] 1.991141 2.879922 3.938863 3.077083 3.947335 3.171009
```

Scatterplots

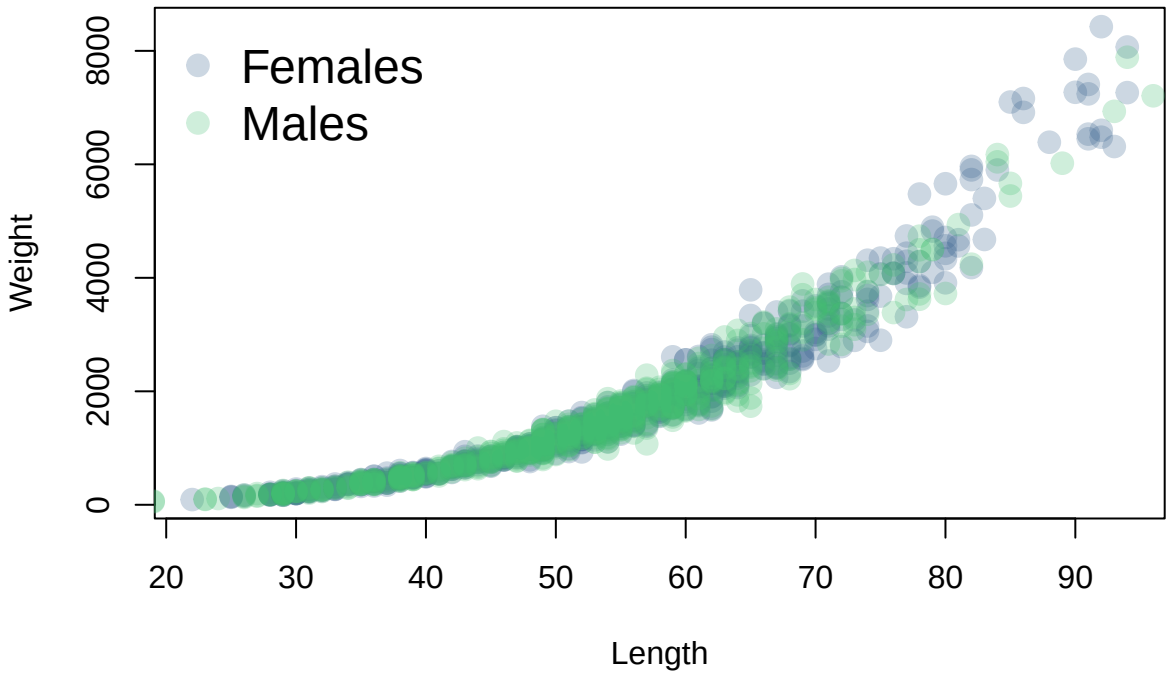
If we want to look at associations between two characteristics, we use scatterplots. For example, we could might be interested in the relationship between fish weight and fish length. We can do a basic scatterplot of the two using the basic function `plot` where each dot is a fish.

```
plot(cod.df$Length, cod.df$Weight,
     pch = 19,
     xlab = 'Length', ylab = 'Weight',
     col = rgb(0,0,0, alpha = 0.15))
```



Sometimes you want to see if different groups follow the same relationship. For example, let's see if male and female fish have different size relationships. Let's indicate that with color. We will use colors from the color scheme `viridis`. We only need two colors for males and females. Then we plot the females first and add the males second. We also want to add a legend to let us know the colors. We simply use the function `legend`. Last thing, let's make the points a big bigger (50% bigger) by using the plotting argument `cex = 1.5`.

```
# Get colors
my.colors <- viridis(2, begin = 0.3, end = 0.7, alpha = 0.25)
# Separate male and female data
male.df <- subset(cod.df, subset = (Sex == "Male"))
female.df <- subset(cod.df, subset = (Sex == "Female"))

# Plot females first
plot(female.df$Length, female.df$Weight,
     pch = 19, cex = 1.5,
     xlab = 'Length', ylab = 'Weight',
     col = my.colors[1])
# Add males to the plot using the function "points"
points(male.df$Length, male.df$Weight,
     pch = 19, cex = 1.5,
     col = my.colors[2])

legend('topleft', #position of legend
       legend = c("Females","Males"), # labels
       col = my.colors, # colors of labels
       pch = 19, cex = 1.5, # points to draw next to labels
       bty = 'n') # box type. 'n' for no box around legend
```



Other notes about plotting

If you want to add points to a plot, use the function `points(x,y)`

If you want to add a line to a plot, use the function `lines(x,y)`.

If you want to add a straight line to a plot, use the function `abline(h = y, v = x, a = intercept, b = slope)`. If you pick an h value, it makes a horizontal line at that y value. If you pick a v value, it draws a vertical line there.

Lines, points, and scatterplots have different symbols that can be controlled by `pch` (for **p**oint **ch**aracter). I like 19 because it is a filled circle.

Points can be made bigger or smaller using `cex`.

All plotting functions take color. You can specify colors with names, with red-green-blue characterization using `rgb`, or you can use a color scheme such as those in the package `viridis`. It's up to you.

You can also add text to figures using the function `text`. This is a bit harder to do, but might be worth your time looking into.

Saving plots

You can save plots using the function `pdf()` following by `dev.off()`.

This works in 3 steps. 1. Write `pdf(file = "FIGURENAME.pdf")`. 2. Run code to create your figure. 3. Write `dev.off()`. Steps 1 and 3 bracket the beginning and end points of telling your computer on what figures to save. The figure gets saved in your current working directory. Change your working directory if needed.

Let's save our fish plot.

```
# Step 1.Let R know you want to save the next plot and you
# are about to start plotting.
pdf(file = 'MyFishPlot.pdf')

# Step 2. Make the plot.
```

```
plot(female.df$Length, female.df$Weight,
     pch = 19, cex = 1.5,
     xlab = 'Length', ylab = 'Weight',
     col = my.colors[1])

points(male.df$Length, male.df$Weight,
     pch = 19, cex = 1.5,
     col = my.colors[2])

legend('topleft', #position of legend
       legend = c("Females","Males"), # labels
       col = my.colors, # colors of labels
       pch = 19, cex = 1.5, # points to draw next to labels
       bty = 'n') # box type. 'n' for no box around legend

# Step 3. Tell R you are done making the plot.
dev.off()
```

```
## pdf
##   2
```

Go check your working directory to see it.

Checkpoint 3-5: Make plots that show the data for the following three questions. For each plot, be sure to include the kinds of information needed on every graph: axis labels, legends (if necessary), and color coding to make the reader's job easier (if necessary).

Chekpoint 3. What is parasite infection prevalence across fish sexes?

Checkpoint 4. What is parasite infection prevalence across areas?

Checkpoint 5. What is the relationship between parasite infection and fish size? That is, are larger or smaller fish more at risk of parasite infection (jitter could be helpful here)?

Probability and Random Sampling

We also covered probability and sampling. We can sample from a vector (with or without replacement) using the function `sample`.

```
alphabet <- letters[1:26]
sample(alphabet, size = 10, replace = T) # With replacement
```

```
##  [1] "g" "b" "o" "r" "f" "e" "r" "t" "d" "b"
```

```
sample(alphabet, size = 10, replace = F) # Without replacement
```

```
##  [1] "v" "g" "f" "o" "k" "s" "w" "d" "l" "y"
```

We also learned how to use extract properties from different probability distributions. Here are the four pieces of information you can extract from named probability distributions.

- `r<name>(sample size, distribution parameters)` - take a random sample with a particular sample size.
- `d<name>(outcomes, distribution parameters)` - find the probability of a particular set of outcomes

- `p<name>(outcomes, distribution parameters)` - find the cumulative probability for a particular set of outcomes
- `q<name>(cumulative probability, distribution parameters)` - find the quantile for a particular set of cumulative probabilities

Here is an example with the binomial.

```
rbinom(5, size = 10, prob = 0.6)
```

```
## [1] 2 6 5 4 4
```

```
  # Possible number of successes in 5 samples where each
  # sample has 10 individuals and the chance of success is
  # 0.6.
dbinom(4:7, size = 10, prob = 0.6)
```

```
## [1] 0.1114767 0.2006581 0.2508227 0.2149908
```

```
  # Probabilities of getting 4,5,6,7 successes in a sample of
  # 10 individuals with probability of success equal to 0.6.
pbinom(4:7, size = 10, prob = 0.6)
```

```
## [1] 0.1662386 0.3668967 0.6177194 0.8327102
```

```
  # Cumulative probabilities of 4,5,6,7 successes out of 10 with
  # prob(success) = 0.6.
qbinom(c(0.5,0.9), size = 10, prob = 0.6)
```

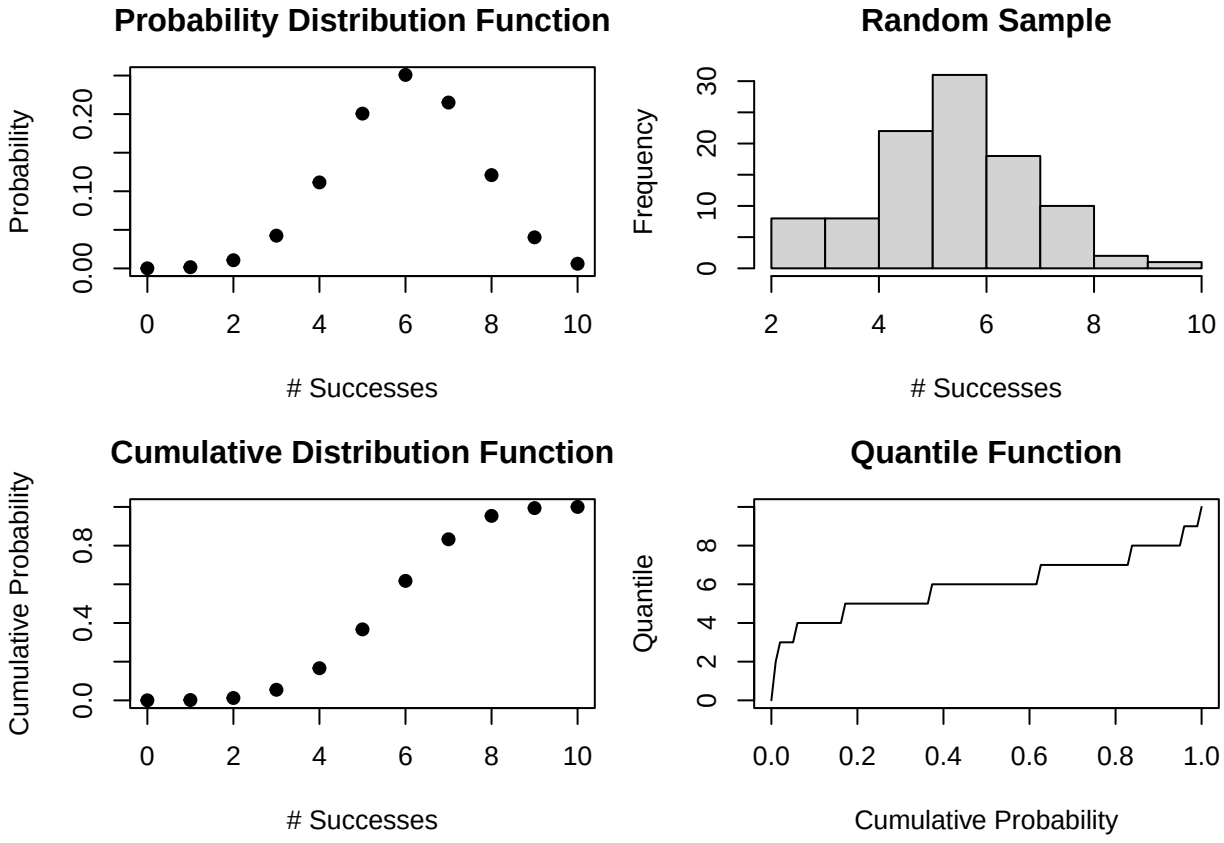
```
## [1] 6 8
```

```
  # 50% and 90% quantiles for a sample of 10 individuals where
  # prob(success) = 0.6.
```

We can also plot these distributions.

```
sample.size <- 10
prob.success <- 0.6
possible.outcomes <- 0:sample.size

# Probability Density function
probabilities <- dbinom(possible.outcomes,
                        size = sample.size,
                        prob = prob.success)

# Make a four panel plot with smaller margins
par(mfcol = c(2,2), mar = c(4,4,3,1))
plot(possible.outcomes, probabilities,
     xlab = '# Successes',
     ylab = 'Probability',
     main = 'Probability Distribution Function',
     pch = 19)

# Cumulative Distribution Function
cumulative.prob <- pbinom(possible.outcomes,
                          size = sample.size,
                          prob = prob.success)
plot(possible.outcomes, cumulative.prob,
     xlab = '# Successes',
```

```
     ylab = 'Cumulative Probability',
     main = 'Cumulative Distribution Function',
     pch = 19)

# Histogram of a 100 random samples from this
# distribution
hist(rbinom(100, size = sample.size, prob = prob.success),
     xlab = '# Successes',
     main = 'Random Sample')

# Quantile Function
quantile.probs <- seq(from = 0, to = 1, length = 100)
quantiles <- qbinom(quantile.probs,
                    size = sample.size,
                    prob = prob.success)
plot(quantile.probs, quantiles,
     xlab = 'Cumulative Probability',
     ylab = 'Quantile',
     main = 'Quantile Function',
     typ = 'l')
```



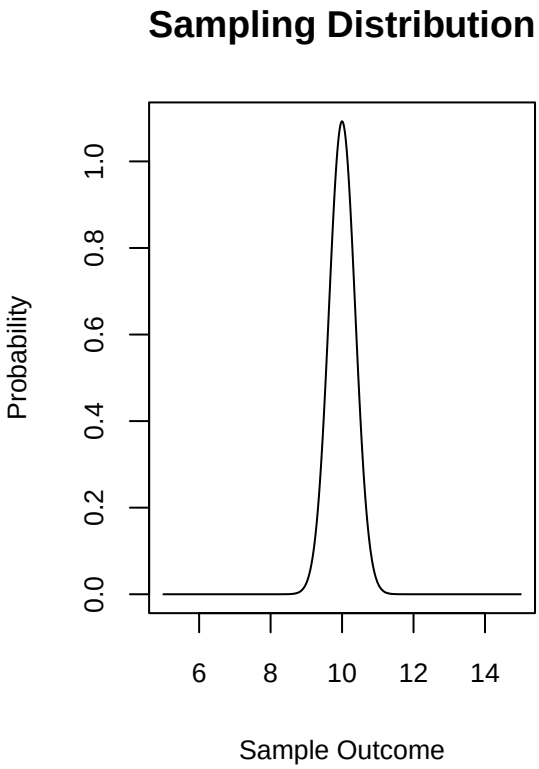
We can do the same thing using the normal distribution with

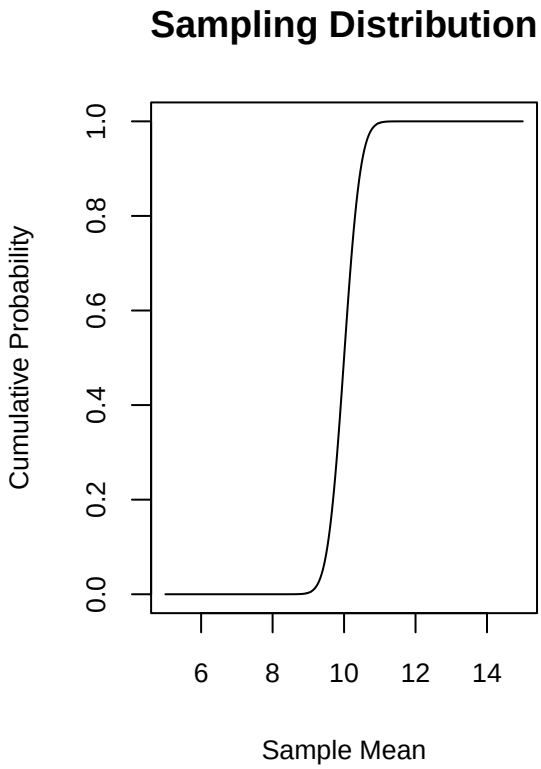
- `rnorm`
- `dnorm`
- `pnorm`

- qnorm

These can used to extract information such as confidence intervals from sampling distributions. For example, we know the mean of a sample will follow a normal distribution with enough individuals (by the central limit theorem). Let's take a look by assuming we are sampling 30 individuals from a population with a mean (μ) of 10 and the standard deviation (σ) of 2. Here is the sampling distribution.

```
# population mean
mu <- 10
# population sd
sigma <- 2

# Sample size
n <- 30

# sampling distribution mean
xbar.mu <- mu
# sampling distributrion standard deviation (=standard error)
xbar.sd <- sigma/sqrt(n)

pot.outcomes <- seq(from = 5, to = 15, length = 10000)
sam.prob <- dnorm(pot.outcomes, mean = xbar.mu, sd = xbar.sd)
sam.cumul.prob <- pnorm(pot.outcomes, mean = xbar.mu, sd = xbar.sd)

par(mfcol = c(1,2), cex.lab = 0.8, cex.axis = 0.8)
plot(pot.outcomes, sam.prob, typ = 'l',
     xlab = 'Sample Outcome',
     ylab = 'Probability',
     main = 'Sampling Distribution')
plot(pot.outcomes, sam.cumul.prob, typ = 'l',
     xlab = 'Sample Mean',
     ylab = 'Cumulative Probability',
     main = 'Sampling Distribution')
```





But the CLT is only an approximation for *large* sample sizes. What counts as large is difficult to determine. In class, I mentioned that an exact sampling distribution for data where individual observations are normally distributed, called the t-distribution. The t-distribution models the difference of the sample mean, $\bar{X}$, from the population mean, μ, in units of standard errors, $SE_{\bar{X}}$. That is, we can make a statistic

$$t = \frac{\bar{X} - \mu}{SE_{\bar{X}}}.$$

This sampling distribution of this statistics is a t-distribution. The t distribution has a single parameter, called the degrees of freedom, which is the sample size - 1 (i.e., $df = n - 1$).

The functions for the t-distribution are

- `rt(n, df)` - n samples with `df` degrees of freedom.
- `dt(t, df)` - probabilities of t values under `df` degrees of freedom
- `pt(t, df)` - cumulative probabilities of t values under `df` degrees of freedom
- `qt(cumul.prob, df)` - quantiles for specific cumulative probabilities with `df` degrees of freedom

Checkpoint 6: Plot the t-distribution for sample size of 5 and calculate its 2.5% and 97.5% quantiles that correspond to 95% of the most likely outcomes.

Checkpoint 7: Plot the t-distribution for sample size of 50 and calculate its 2.5% and 97.5% quantiles.

Standard Errors and Confidence Intervals

We can calculate standard errors and confidence intervals of the mean in two ways.

1. We can do it manually by using the binomial for categorical outcomes, t-distribution for normally distributed outcomes, or use the normal by leaning on the CLT.
2. We can use linear model structures that will use the t-distribution.

Let's look at each.

Categorical Characters

One categorical character we have is parasite prevalence. Let's take a look at the prevalence in our sample.

```
n <- length(cod.df$Prevalence)
num.infected <- sum(cod.df$Prevalence == 1)
```

We know that the number of occurrences of a categorical outcomes in a random sample follows a binomial distribution. We can use this to get the 95% confidence interval.

```
alpha <- 0.05
conf.levels <- c(alpha/2, 1 - alpha/2)

(conf.int <- qbinom(conf.levels, size = n, prob = num.infected/n))/n
```

```
## [1] 0.4362041 0.4920255
```

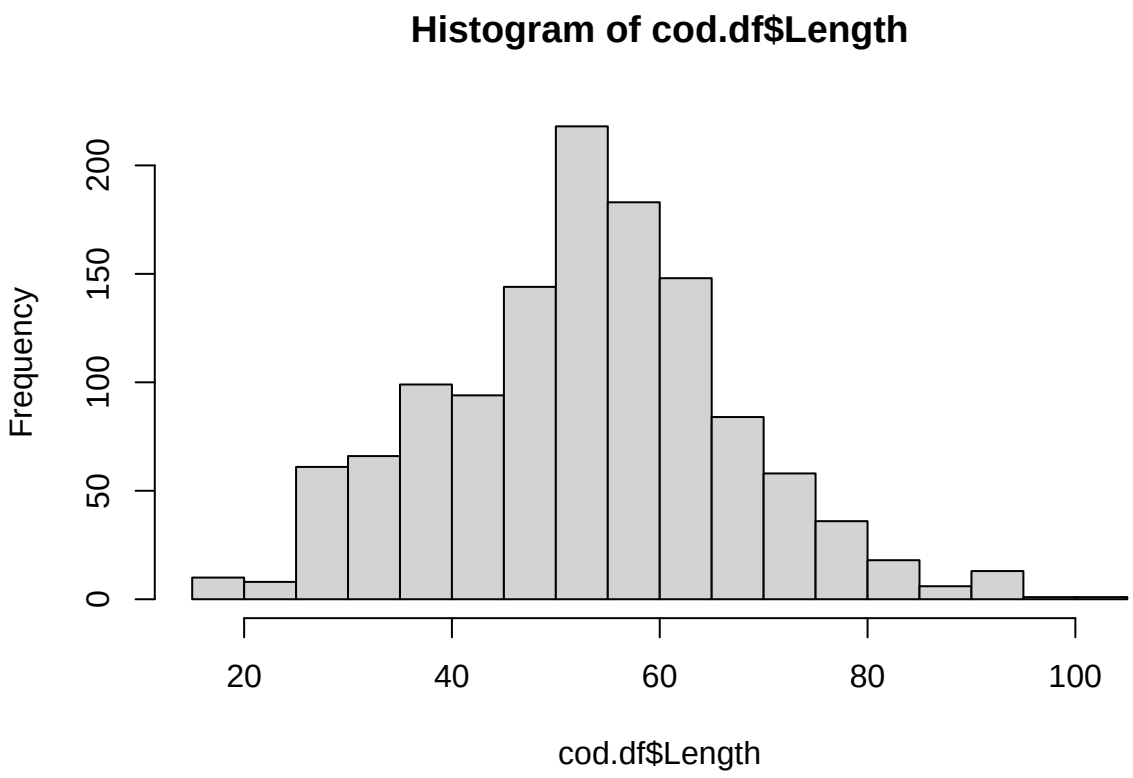
This says the probability that the true prevalence is between 43.6% and 49.2% is 0.95.

Checkpoint 8: Construct a 99% confidence interval for prevalence of parasite infection.

Characters that can reasonably be modeled by a normal

One character that can be modeled by a normal is fish length. Let's take a look.

```
hist(cod.df$Length)
```



Looks good enough for me. Let's find the mean and standard errors. Note that lengths are not available for some fish, so we need to only take the mean and standard deviation of fish we actually have lengths for.

```
mean.length <- mean(cod.df$Length, na.rm = T)
n.length <- sum(!is.na(cod.df$Length))
sd.length <- sd(cod.df$Length, na.rm = T)
se.length <- sd.length/sqrt(n.length)

n.length; mean.length; sd.length; se.length;
```

```
## [1] 1248
```

```
## [1] 53.44551
```

```
## [1] 14.10598
```

```
## [1] 0.3992968
```

There are 1248 fish we have lengths for. Their average length is 53 cm and we expect a randomly sampled fish to differ from the mean by 14 cm. So about 68% of fish are between the sizes of 53cm - 14cm = 39cm and 53cm + 14cm = 67cm. That also means that ~95% of fish are between the sizes of 53cm - 2*14cm = 25cm and 53cm + 2*14cm = 81 cm. That pretty much matches the histogram above.

This also says we expect the mean of the sample to differ only by about 0.4 cm for each sample of 1248 individuals!

Let's use the t-distribution to find the 99% confidence interval.

```
alpha <- 0.01
conf.levels <- c(alpha/2, 1-alpha/2)
qt(conf.levels, df = n.length -1)
```

```
## [1] -2.579778  2.579778
```

So this says that 99% of samples should be about 2.58 standard errors from the true mean value. We can use this to make a 99% confidence interval.

```
(fish.conf.int <- mean.length + qt(conf.levels, df = n.length - 1)*se.length)
```

```
## [1] 52.41542 54.47561
```

There it is. Presuming this data was collected as a random sample, the population meann is almost certainly near 53.4 cm over the time period of collection.

The other method is to do the same with the `lm` function.

```
length.mdl <- lm(Length ~ 1, data = cod.df)
confint(length.mdl, level = 0.99)
```

```
##                0.5 %   99.5 %
## (Intercept) 52.41542 54.47561
```

And the linear model gives the exact same result! (It should because it follows the exact same reasoning we have.)

Checkpoint 9: Find the 95% confidence intervals for fish lengths in each year of the study.

A synthesis challenge

I really want to know, what is the parasite prevalence in fish across space? You now have all the tools to do this.

Checkpoint 10: Estimate parasite prevalence (with uncertainty) in each location. Make a visualization to show the data that support this estimate. Finally, describe the pattern of prevalence across space in Norwegian Cod based on this data.