Chapter 1 Sample Answers

Justin Touchon

7/19/2021

This page provides sample answers to the assignment at the end of Chapter 1 of Applied Statistics with R: A Practical Guide for the Life Sciences by Justin Touchon. Since the assignment is very open-ended, these answers are truly meant to be *examples* and are not the only answers you can get. My hope is that you are here either because 1) you want to check the answers you got to make sure you completed the assignments correctly, or 2) you got stumped and need some help. Either way, you will learn much more if you have already spent some time working through the assignments on your own. If you haven't done that, close this page and go work on it! :)

Assignment 1

Create a data frame that contains at least one factor with three categories and at least three columns of made up numeric data. Thus, it should have at least four columns. Make sure the columns have meaningful names. Have at least 10 rows per categories (i.e., at least 30 rows long).

How should you go about doing this? As with anything in R, there are of course many different ways. I will show you how I would do it. The basic idea here is to make a data frame with four columns and one of those columns needs to be a factor with three categories.

Let's start by making the factor. Since the data this book will be working with are ecological data about red-eyed treefrog tadpoles, I will stick with that general genre of data. Let's imagine our three categories are three different species of treefrogs. Let's choose hourglass treefrogs (the species pictured on the cover of the book), which we will abbreviate "HGTF", red-eyed treefrogs, which will be "RETF", and Rosenberg's gladiator frog, or "GLDF".

Three beautiful treefrogs from Central America. Photos by Justin Touchon.

In order to make our factor, we can combine the functions **rep()** and **c()**. Each species abbreviation is repeated 10 times, and those vectors are concatenated together to make a single vector.

```
#We first create our vector of species codes
species<-c(rep("HGTF",times=10), rep("RETF",times=10), rep("GLDF",times=10))
#By typing in the name of the vector, we can verify that it worked
species
```

```
##  [1] "HGTF" "HGTF" "HGTF" "HGTF" "HGTF" "HGTF" "HGTF" "HGTF" "HGTF" "HGTF"
## [11] "RETF" "RETF" "RETF" "RETF" "RETF" "RETF" "RETF" "RETF" "RETF" "RETF"
## [21] "GLDF" "GLDF" "GLDF" "GLDF" "GLDF" "GLDF" "GLDF" "GLDF" "GLDF" "GLDF"
```

If we want to specify that this should be a factor, we can do so with the **as.factor()** function. Just assign the object over itself, as is shown below.

```
species<-as.factor(species)
#Type the name of the object at the console to see the difference from the original vers
ion
species
```

```
##  [1] HGTF HGTF HGTF HGTF HGTF HGTF HGTF HGTF HGTF HGTF RETF RETF RETF RETF RETF
## [16] RETF RETF RETF RETF RETF GLDF GLDF GLDF GLDF GLDF GLDF GLDF GLDF GLDF GLDF
## Levels: GLDF HGTF RETF
```

Next, we need to make our numeric variables. Perhaps our experiment involves raising these tadpoles under three different conditions. For the sake of example, let's imagine that we raised them under three different diets, which we will just call "control", "dietA", and "dietB". You could imagine that the control is what we always feed our tadpoles, but diets A and B are experimental diets that we are interested in understanding the effects of. Let's also imagine that the two experimental diets are most beneficial to a particular species.

Let's create three numeric variables, one for each diet. Like above, we need to concatenate together (using the **c()** function) three vectors of numbers. We will create those numbers using the **rnorm()** function, which creates a vector of numbers randomly selected from a normal distribution with a given mean and standard deviation. If that doesn't make a lot of sense, don't worry, the normal distribution will be discussed in Chapter 3! Lastly, note that hourglass treefrog tadpoles are considerably smaller than either red-eyed treefrog or gladiator frog tadpoles, which are pretty similar in size.

```
#First, create a vector of total lengths measured from our hypothetical control diet.
control<-c(rnorm(n=10, mean=20, sd=1), rnorm(n=10, mean=35, sd=1), rnorm(n=10, mean=35,
 sd=1))
#Next, we will create a vector of total lengths measured from our hypothetical experimen
tal diet A.
#Let's imagine that this diet really benefits the red-eyed treefrog tadpoles, which are
 the second set of 10 animals in our data frame.
dietA<-c(rnorm(n=10, mean=20, sd=1), rnorm(n=10, mean=45, sd=1), rnorm(n=10, mean=35, sd
=1))
#Lastly, let's create a vector of total lengths measured from our hypothetical experimen
tal diet B.
#Let's imagine that this diet most benefits the gladiator frog tadpoles, which are the t
hird set of 10 animals in our data frame.
dietB<-c(rnorm(n=10, mean=20, sd=1), rnorm(n=10, mean=35, sd=1), rnorm(n=10, mean=50, sd
=1))
#We can see that these worked if we type the names of the vectors into the console.
control
```

```
##  [1] 20.28765 20.50313 21.12897 19.72175 20.14627 21.34154 21.60495 20.07540
##  [9] 20.24625 16.64097 35.12743 35.09776 34.15570 34.53447 35.67963 34.34170
## [17] 34.63109 34.51641 34.12231 34.09898 35.49607 34.10358 34.56424 35.36238
## [25] 35.70223 34.49017 34.55833 34.72064 35.11726 33.16159
```

```
dietA
```

```
##  [1] 20.95581 20.35380 20.90934 18.92906 18.77894 21.05896 19.36297 19.97784
##  [9] 19.61302 19.56862 46.67544 43.80609 45.32396 44.76575 44.57497 45.13672
## [17] 45.11032 43.15470 46.23609 45.06409 34.74257 35.04648 36.03387 33.86551
## [25] 35.68297 35.91487 33.97409 34.78326 35.29067 34.27993
```

```
dietB
```

```
##  [1] 18.51032 20.06070 19.22531 19.50250 20.61580 20.57504 21.27551 19.91593
##  [9] 19.18883 19.79930 35.80058 36.23139 35.48757 34.28855 34.47623 37.35941
## [17] 35.69033 35.04877 36.03441 33.45098 51.26033 51.43865 50.73609 50.48902
## [25] 51.20996 47.62562 51.09144 50.39724 49.54861 50.03212
```

Okay, now we have our four vectors. It's time to combine them into a data frame using the function **data.frame()**. Remember to assign your new data frame to an object!

```
tad_data<-data.frame(species, control, dietA, dietB)
tad_data
```

```
##    species  control    dietA    dietB
## 1     HGTF 20.28765 20.95581 18.51032
## 2     HGTF 20.50313 20.35380 20.06070
## 3     HGTF 21.12897 20.90934 19.22531
## 4     HGTF 19.72175 18.92906 19.50250
## 5     HGTF 20.14627 18.77894 20.61580
## 6     HGTF 21.34154 21.05896 20.57504
## 7     HGTF 21.60495 19.36297 21.27551
## 8     HGTF 20.07540 19.97784 19.91593
## 9     HGTF 20.24625 19.61302 19.18883
## 10    HGTF 16.64097 19.56862 19.79930
## 11    RETF 35.12743 46.67544 35.80058
## 12    RETF 35.09776 43.80609 36.23139
## 13    RETF 34.15570 45.32396 35.48757
## 14    RETF 34.53447 44.76575 34.28855
## 15    RETF 35.67963 44.57497 34.47623
## 16    RETF 34.34170 45.13672 37.35941
## 17    RETF 34.63109 45.11032 35.69033
## 18    RETF 34.51641 43.15470 35.04877
## 19    RETF 34.12231 46.23609 36.03441
## 20    RETF 34.09898 45.06409 33.45098
## 21    GLDF 35.49607 34.74257 51.26033
## 22    GLDF 34.10358 35.04648 51.43865
## 23    GLDF 34.56424 36.03387 50.73609
## 24    GLDF 35.36238 33.86551 50.48902
## 25    GLDF 35.70223 35.68297 51.20996
## 26    GLDF 34.49017 35.91487 47.62562
## 27    GLDF 34.55833 33.97409 51.09144
## 28    GLDF 34.72064 34.78326 50.39724
## 29    GLDF 35.11726 35.29067 49.54861
## 30    GLDF 33.16159 34.27993 50.03212
```

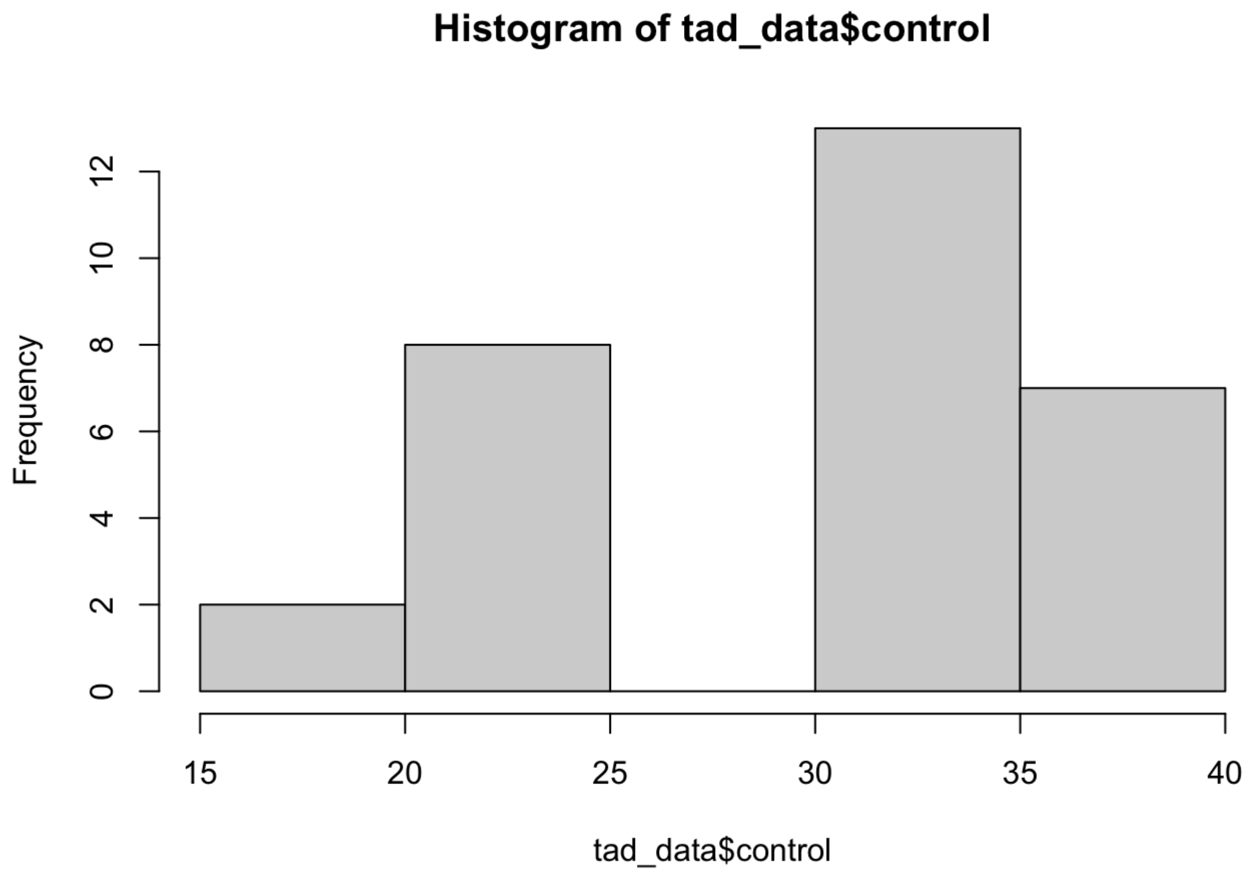
Hooray, you did it! As discussed earlier, you could choose anything to be your categories (e.g., strains of mice in a study, brands of cereal to eat, species of plants) and anything for your numeric variables (e.g., responses of mice to different tests, sugar, protein, and fat content of cereals, growth rates of plants under three different light conditions, etc.) Okay, onward and upward.

Assignment 2

Once you have a data frame, plot a histogram of your numeric variables. Try to dress it up by adding color, changing the limits of the axes, and adding a main title. How will you figure out how to do that? Look at the help file for hist()**.**

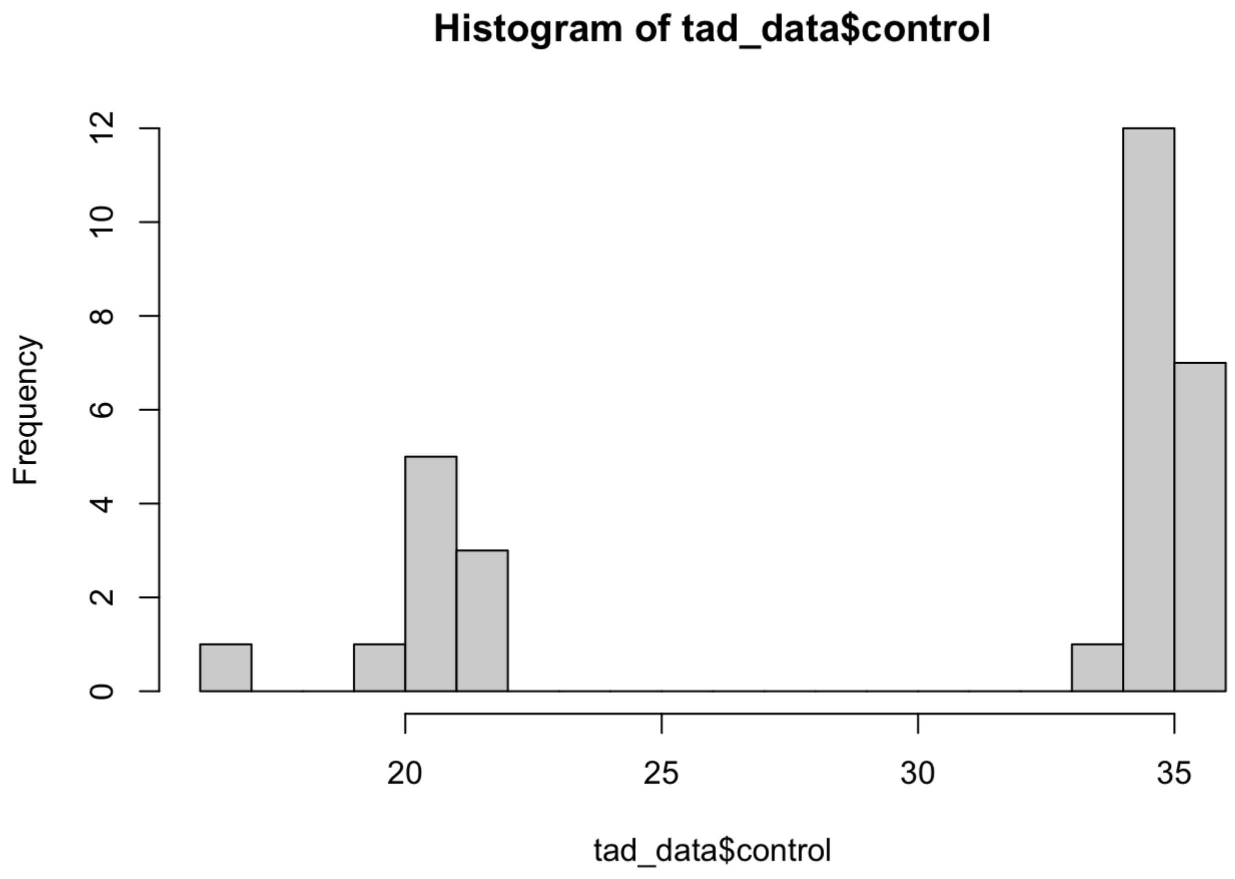
The basic goal here is to look at the values in your numeric vectors. If you look at the help file for the **hist()** function (which can be done by simply typing *?hist* in the console) you will see there are lots and lots of options. All of those are *optional* except for *x*, which is the vector of values you want to plot. In our data frame, we can access our numeric vectors with the **$** operator. For example, if we want to plot a histogram of the vector of sizes we measured in our control diet, we can type the following:

```
hist(tad_data$control)
```



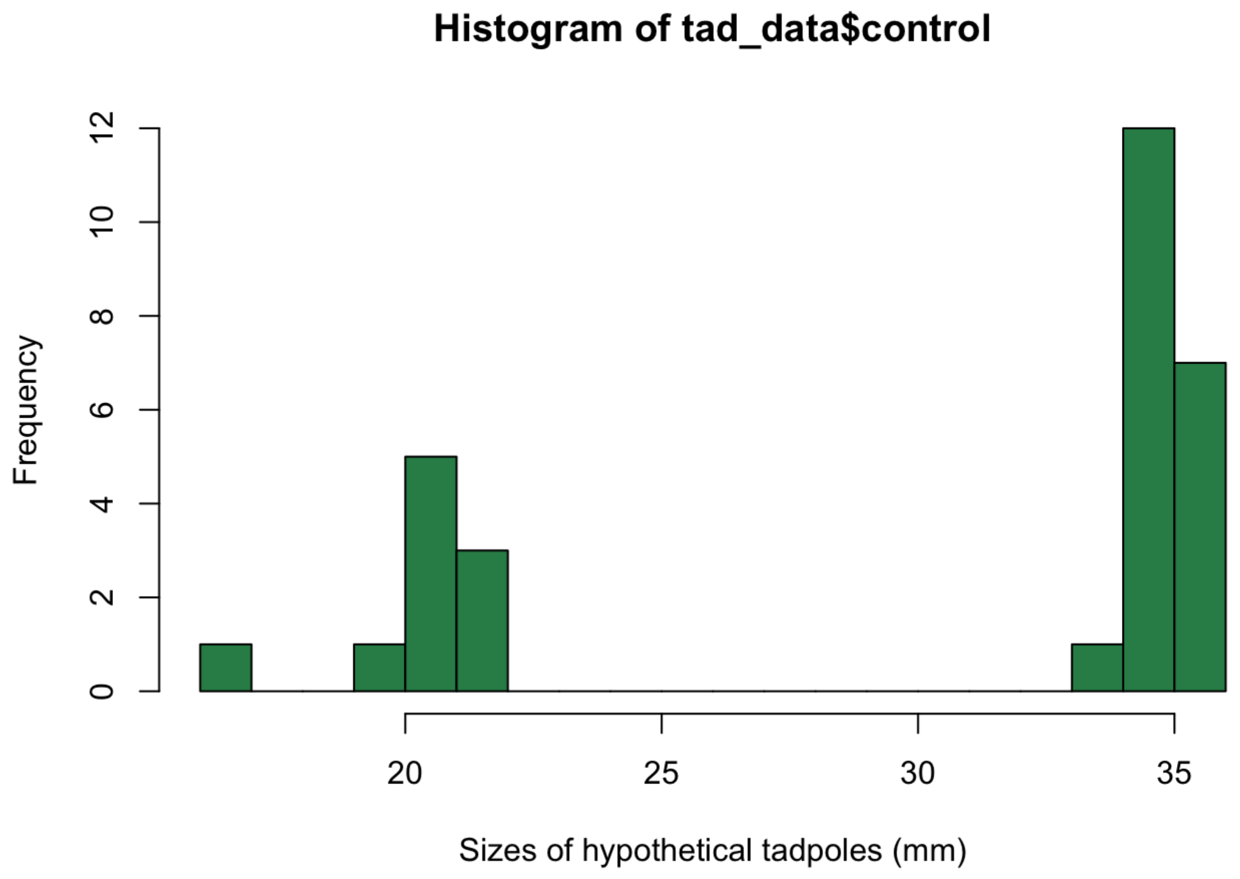
If you do that, you might get something very unhelpful, like I have here. In a function like **hist()**, R is going to try to choose a set of divisions in the data which it *thinks* will be most useful, but we can override them. Here, we want to set the number of *breaks* to use when plotting the histogram. Let’s go with 20 and see what that looks like.

```
hist(tad_data$control, breaks=20)
```



We can see a cluster of data representing our smallest species and a larger cluster o values representing the two larger species. You should also notice that by default R names our x-axis the variable we provided it. We can customize that by specify a new *label* for our x-axis, with the *xlab*= argument. We can also add color with the *col*= argument. Below, I've picked the color "seagreen" but you could of course pick whatever you want. (hint: type **colors()** in the console)

```
hist(tad_data$control, breaks=20, xlab="Sizes of hypothetical tadpoles (mm)", col="seagr
een")
```



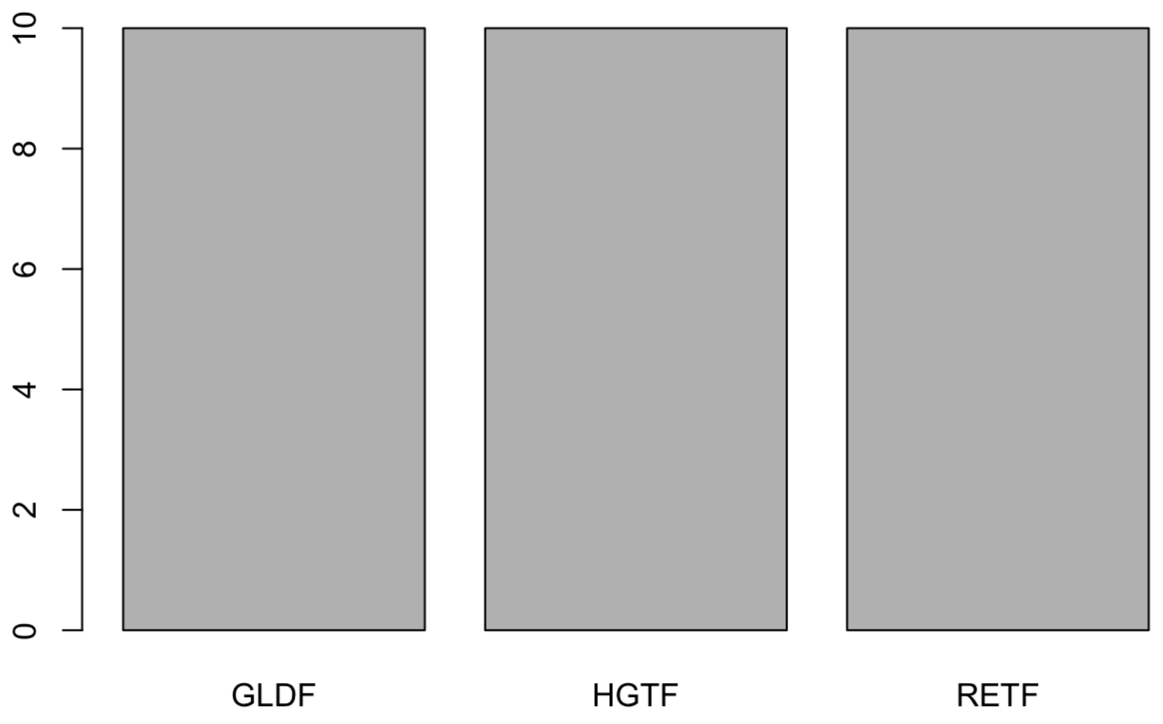
There are lots of other ways to dress up a figure. Play around with changing the title with the *main==* argument, or play with the range limits using *xlim=* or *ylim=*. You can even *add* one histogram to an existing plot, using the *add=T* argument.

Assignment 3

Use the plot() **function to try and plot your numeric and categorical variables. What happens when you give R different types of data to plot?**

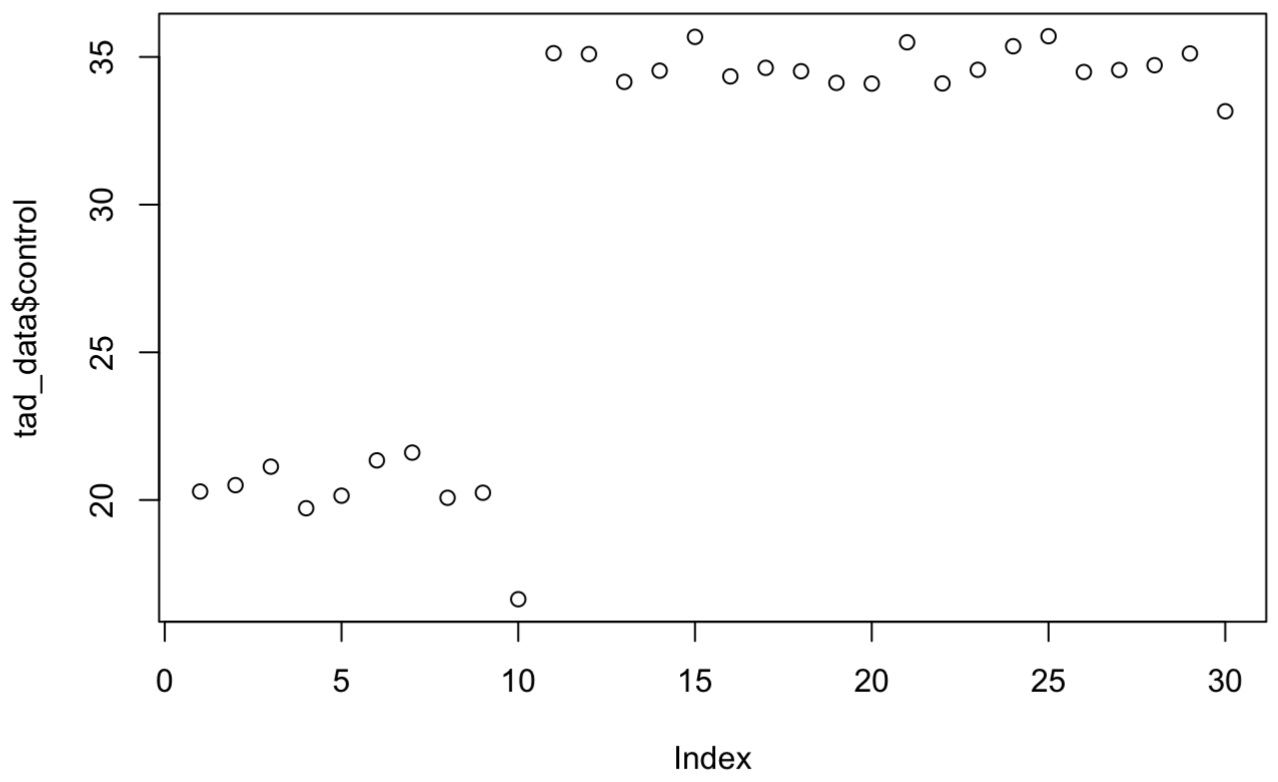
The **plot()** function is a very basic and fundamental function in R. We will explore its utility in the coming chapters, but for now let's explore what happens when we give it just a single vector of numbers. For example, if we provide **plot()** with our factor variable "species" it will plot a bar for each category with the height of the bar being now many observations are in each category. In this case, that means three bars that are each 10 high.

```
plot(tad_data$species)
```



What happens if we provide a vector of numeric values? R will plot each of the values in our vector on the y-axis, with the x-axis being the row of value. Thus, **plot()** just starts at the first row of the data frame and proceeds downward. In the figure below, the first 10 values (which correspond to our HGTF animals, which were the smallest) are smaller than values 11–30, which correspond to the two larger species.

```
plot(tad_data$control)
```



In the coming chapters we will explore what happens if you provide combinations of vectors as the x- and y-values, and we will begin learning how to use the ***ggplot2*** package to make beautiful figures. However, for now that's it! Great job.