Discovering Objects

Directions: Follow along with the slides and answer the questions in **BOLDED** font in your journal.

What are objects?

- When we're coding: the numbers, variables, functions and data stored on the computer are all different types of **objects**.
  - **Objects** assign a *name* and *type* to a set of *values*.
  - *Functions*, like `histogram`, are also objects with names and coded *steps* instead of sets *values*
- **Objects** then are the groups of values, variables or steps needed by a function, that we can use to easily analyze data.

Take this line of code, for example:

```
bargraph(~helmet, data = cdc)
```

- `bargraph` is a **function** object.
  - This function takes other objects, `helmet` & `cdc` in this case, and performs all the necessary steps needed to produces a bargraph.
- `helmet` is a **vector** object.
  - The different values of the `helmet` variable are stored, in order, for us to use.
- `cdc` is a **data frame** object.
  - It stores all of the different values of *all* of the variables variables in our data set and keeps them all ordered.

Why are objects important?

- By using *objects*, we simplify the code writing process:
  - We can use the *values* of our data sets by calling their *variable names*.
  - We can call *functions* instead of having to continually write the same steps over and over and over . . .

A simple use of objects

- Load the `cdc` data into RStudio.
- We can use the following to look at the first 25 values of the `weight` variable

```
head(~weight, data = cdc, n = 25)
```

- **Write down how you would add these 25 values together.**
- **What would you do if someone asked you to compute the total combined weight of *everyone* in the data?**

A simple use of objects

- Here's how we can answer these questions using objects.
- First we save our values as new object named `first_25`.

```
first_25 <- head(~weight, data = cdc,
                 n = 25)
```

- And then we use the `sum` function on this object:

```
sum(first_25)
```

- And if we wanted the total weight of everyone?

```
sum(~weight, data = cdc)
```

A couple things to notice (1/2):

- When we save objects and give them a name, we want to choose names that are:
- *Short*. So we don't have to waste time writing our code.
- *Descriptive*. So we can remember what the object contains.
- **In your opinion, was `first_25` a good name for the first 25 values of our `weight` variable? Why or why not?**
- **What would be a good name for the value of `sum(first_25`?**

A couple things to notice (2/2):

- When we ran:

```
sum(~weight, data = cdc)
```

- You may have noticed a warning message popped up:

```
Warning message:
In sum(~weight, data = cdc) : The data contains 979 missing values
```

- This message is telling us that 979 people in our data didn't report their weight.
- We denote **missing values** in our data as `NA`, which stands for *not available*.

What just happened?

- We created a **vector** of the first 25 `weights` in our data and **assigned** it the name `first_25`.
  - The arrow, `<-`, is how we assigned the *values* from `head(~weight, data = cdc, n = 25)` to the *name* `first_25`.
  - A **vector** is a type of object that stores one or more values and keeps them in order.
- When then put this object into the `sum()` function.
  - The function

More about objects

- Every *object* needs 3 things:
    1. The *name* we can call to use the object.
    2. The *values* contained in the object.
    3. The *type* of object.

Our example explained:

```
first_25 <- head(~weight, data = cdc,
                 n = 25)
```

- `first_25` is the *name* of our object.
- We **assign** the *values* of our object, `head(~weight, data = cdc, n = 25)`, using the `<-` symbol.
- And since we're storing values from a single variable, and keeping them in order, we're creating a *vector* type object.

Objects and their info

- **Run the following commands and give a brief description about what each function does:**

```
names(cdc)
```

```
print(~age, data = cdc)
```

```
View(first_25)
```

```
str(cdc)
```

- **What information gets printed for the last command?**

On your own:

- **Choose a variable besides `weight`:**
- **Create a *vector* of the last 50 values using the `tail` function (Look at how you used the `head` function for a hint at how to use the `tail` function).**
- **Give your *vector* object a name and write it down. Is the name short & descriptive?**
- **For these 50 values, create either a `histogram` or a `bargraph`.**
- **How did you know which plot is appropriate for your values?**
- **Which particular values seemed to occur the most often?**