Preparing data for analysis and visualization in R

Observations and variables

Jenine Harris
Brown School

Defining observations and variables

- **observations**: the person, organization, or other thing being measured
- **variables**: the characteristics being measured
- In a typical data set, observations are the rows and variables are the columns.
- For example, a data set consisting of people (the observations) asked about their `voting` and `income` (variables) might look like this:

```
##   income voted
## 1  34000   yes
## 2 123000    no
## 3  21500    no
```

An example of data with observations and variables

In this very small data set, there are three **observations** (the rows) and two **variables** (the columns).

- The first observation is a person with an income of $34,000 that answered "yes" for voted.
- The second observation is a person with an income of $123,000 that answered "no" for voted.
- The third observation is a person with an income of $21,500 that responded "no" for voted.

The two variables are `income` and `voted`.

```
##   income voted
## 1  34000   yes
## 2 123000    no
## 3  21500    no
```

Entering and storing variables in R

- R stores information as **objects**, and then data analyses and data management are performed on these stored objects.
- Before an object can be used in data management or analyses in R, it has to be stored in the R environment.
- Information is stored as objects in R by **assigning** the information a name, which can be a single letter or some combinations of letters and numbers that will serve as the name of the object.

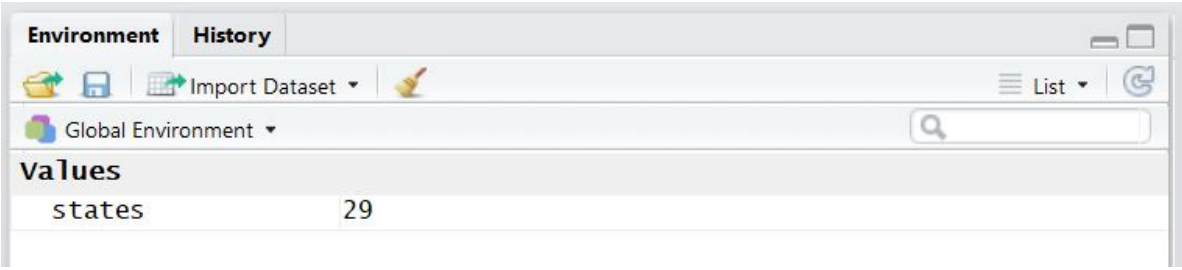
Using the assignment arrow

- Assigning an object to a name is done by using an arrow like this: `<-`.
    - The arrow separates the name of the object on the left from the object itself on the right, like this: `name <- object`.
    - An object can be as simple as one letter or number. An object can also be as complex as several data sets combined together.
- For example, let's say you wanted to create an object that stores the number of states with legal marijuana.
- Try storing `29` in an object called `states` by typing the following at the R prompt (`>`) in the Console pane of the R Studio window.
    - After typing the code, press Enter on the keyboard

```
states <- 29
```

Using the environment tab

- The `states <- 29` code does not result in anything actually happening in the Console window, instead the `states` object is stored and there is no result to display.
- R is storing the number 29 as an object called `states` to use in analysis.
- While nothing happened in the Console, something did happen.
    - The `states` object is now stored under the **Environment tab** in the top right pane of the R Studio window.
    - Notice that the window shows `states` and its value of 29 under a heading titled *values*.



Using the history tab

- There is also a **History** tab next to the **Environment** tab in the upper right pane.
- Clicked on the History tab to see `states <- 29`.
- This tab holds all of the code run since the History pane was last cleared.
- Double-click on any of the code shown in the History window and R will send the code to the Console, ready to run again. There is no need to type anything twice!
- Try clicking on the `states <- 29` code in the History pane to see what happens.

Viewing an object

- To see the value of the states object created in the Console pane, type the name of the object at the R prompt and press enter:

```
states
```

```
## [1] 29
```

- Notice that the value of `states` appears.

Using objects in mathematical expressions

- Once a value is stored, it can be used in mathematical expressions.
- To try using the `states` object in a mathematical expression, enter the expression `2 + states` at the R prompt and pressing Enter:

```
2 + states
```

```
## [1] 31
```

- Note that 31 is printed in the Console, which is the value of `2 + 29`.
- New vocabulary: When running the code results in something being displayed on the screen, the thing that is displayed is the **output**. For example, the 31 printed after running the code above is **output**.

Check your understanding

Assign your age in years to an object with your name. Add 5 to the object and press Enter to see how old you will be in 5 years.

Answer

```
ageYears <- 49

ageYears + 5
```

```
## [1] 54
```