ECON 590 Week Two Handout

Basic Data Structures in R

Last class, we said that <- is the assignment operator. We created an object named a that stored the value 4 using the following command.

```
a <- 4
```

In R, we can create several different kinds of objects or data types using the assignment operator:

- **Numeric**: stores a single value (like a stores the number 4 above)
- **String**: stores a string of text (like "hello world")
- **Logical**: stores binary values True or False
- **Factor**: stores categories (like "Employed," "Employed Part-Time," and "Unemployed)

```
# Below are examples of various data types and functions to identify them.
# First, let's check if a is numeric. `is` functions will always return
# true or false.

a <- 4

is.numeric(a)
```

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

```
# Now we can create a string object and check it using is.character()

a.string <- "123 Main St"

is.character(a.string)
```

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

```
# When we create a logical object, note that we don't put TRUE in quotes
# (otherwise it would be a string object)

a.logical <- TRUE

is.logical(a.logical)
```

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

```
# We'll take a longer look at factor variables in a future class
```

Vectors

We also showed last class how we can create vectors storing values, like the following:

```
b.vector <- c(1, 2, 3)
```

In the code above, the `c()` function *combines* or *concatenates* the arguments in parentheses. Vectors are the basic data structure in `R`, so we want to make sure we understand how they work:

```
# Basic operations in R get applied to each element of a vector

b.vector + 2
```

```
## [1] 3 4 5
```

```
# We can access individual elements in a vector using []'s

b.vector[2]
```

```
## [1] 2
```

```
b.vector[2:3]
```

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

Data Frames in `R`

When we work with data in `R`, we'll generally be using `data frames`. Technically, data frames are just a collection or list of vectors. We can create a sample data set below.

```
# Define vectors storing various object types from above

person.ID <- c(12, 24, 54, 65)
address   <- c("123 Main St.", "274 Long St.", "789 Right St.", "467 Left St.")
employed  <- c(TRUE, TRUE, FALSE, TRUE)
wage.inc  <- c(12500, 15750, 0, 14100)

# Combine each of the individual vectors into a data frame

data <- data.frame(person.ID, address, employed, wage.inc)
```

You might know what each of the object types defined above are already, but you can check the "Environment" pane to confirm. Find our `data` object, and click the triangle button to see more information. We can access individual variables (or vectors) in the data frame using the `$` operator:

```
# Code below prints the entire person.ID vector then just particular rows

data$person.ID
```

```
## [1] 12 24 54 65
```

```
# We can also perform operations on the individual variables

mean(data$employed) # Note that T = 1, F = 0 here
```

```
## [1] 0.75
```

```
mean(data$wage.inc)
```

```
## [1] 10587.5
```

Accessing Data in Data Frames

There's two general ways of retrieving data from data frames. Generally, we'll want to use `tidyverse` commands where possible for larger data sets (more on that in the class activity for today). But its important to know the `base R` approach as well. In the section above, we showed we can use `$` to pick out specific variables or vectors.

In this section, we'll show several ways of accessing data within data frames.

```
# We can use brackets to access data within a particular variable. The command
# below returns the first address in our data set

data$address[1]
```

```
## [1] "123 Main St."
```

```
# We could also select multiple observations sequentially using the following

data$address[2:4]
```

```
## [1] "274 Long St."  "789 Right St." "467 Left St."
```

```
# Finally, you can provide a list of row indexes like so

data$address[c(1,3,4)]
```

```
## [1] "123 Main St."  "789 Right St." "467 Left St."
```

```
# One useful function to learn about the structure of a data frame is str()

str(data)
```

```
## 'data.frame':    4 obs. of  4 variables:
##  $ person.ID: num  12 24 54 65
##  $ address  : chr  "123 Main St." "274 Long St." "789 Right St." "467 Left St."
##  $ employed : logi  TRUE TRUE FALSE TRUE
##  $ wage.inc : num  12500 15750 0 14100
```