Introduction to Data

James M. Murray, Ph.D.
University of Wisconsin - La Crosse

Updated: September 12, 2017

PDF file location: http://www.murraylax.org/rtutorials/data.pdf

HTML file location: http://www.murraylax.org/rtutorials/data.html

---

Note on required packages: The following code requires the packages in the `tidyverse`. The `tidyverse` contains many packages that allow you to organize, summarize, and plot data. If you have not already done so, download and install the libraries (needed only once per computer), and load the libraries (need to do every time you start R) with the following code:

`install.packages("tidyverse") # This only needs to be executed once for your machine`

`library("tidyverse") # This needs to be executed every time you load R`

---

1. Introduction

In this tutorial we will learn about what is data, what is a variable, and the types of scales that variables can be measured on. We will learn by doing using the data set, `countyrent.RData`, which is a merge of data from Zillow and the United States Census. The data set includes the median rental price for two bedroom apartments advertised on Zillow, and economic characteristics of the same counties provided by the U.S. Census.

You can download the data and load it into memory with a single step:

```
load(url("http://murraylax.org/datasets/countyrent.RData"))
```

The `load()` function above expects a path to a file name for an `.RData` file and loads the objects from that data file into memory. The file `countyrent.RData` includes one object, `countyrent.df`, which is a data frame that includes the data.

The `url()` function is embedded into the `load()` call above because `load()` expects a path name on the local file system (or an existing open connection), not a URL web address. The function `url()` creates the connection we need.

We can view the data in RStudio with the `View()` function, or by just clicking on the `countyrent.df` object in the upper-right `Environment` view.

```
View(countyrent.df)
```

You can call the `head()` function if you prefer just to view the top few rows of the data frame in the `Console` view:

```
head(countyrent.df)
```

```
##    FIPS RegionName State MedianRent
## 1 01003    Baldwin    AL    1010.27
## 2 01073  Jefferson    AL     785.79
## 3 01097     Mobile    AL     693.75
## 4 02020  Anchorage    AK    1293.13
## 5 04013   Maricopa    AZ    1019.83
## 6 04019       Pima    AZ     750.54
##                                         UrbanInfluence
## 1 In small metro area of less than 1 million residents
## 2          In large metro area of 1+ million residents
## 3 In small metro area of less than 1 million residents
## 4 In small metro area of less than 1 million residents
## 5          In large metro area of 1+ million residents
## 6 In small metro area of less than 1 million residents
##                EconTopology PovertyPerc MedianIncome
## 1                Recreation        12.9        52387
## 2            Nonspecialized        18.0        48415
## 3            Nonspecialized        18.4        42530
## 4 Federal/State Gov dependent         8.7        77791
## 5            Nonspecialized        16.3        56017
## 6            Nonspecialized        18.7        47107
```

If for some reason you prefer to view the *last* few rows in the Console view, you can do that too:

```r
tail(countyrent.df)
```

```
##      FIPS RegionName State MedianRent
## 327 53077     Yakima    WA     738.75
## 328 54061 Monongalia    WV     826.04
## 329 55025       Dane    WI    1098.75
## 330 55059    Kenosha    WI    1058.54
## 331 56021    Laramie    WY     750.00
## 332 56025    Natrona    WY    1001.00
##                                           UrbanInfluence
## 327 In small metro area of less than 1 million residents
## 328 In small metro area of less than 1 million residents
## 329 In small metro area of less than 1 million residents
## 330          In large metro area of 1+ million residents
## 331 In small metro area of less than 1 million residents
## 332 In small metro area of less than 1 million residents
##                  EconTopology PovertyPerc MedianIncome
## 327            Nonspecialized        19.1        46891
## 328 Federal/State Gov dependent        19.6        46718
## 329 Federal/State Gov dependent        11.2        65416
## 330            Nonspecialized        12.7        56414
## 331 Federal/State Gov dependent        10.6        59147
## 332          Mining dependent        11.0        56703
```

2. From Concepts to Data

It is useful to think explicitly what we mean by variables, data, measurement. They are commonly used words and you may have a good idea what they mean, but sometimes subtleties in their meaning have implications for how we work with them.

2.1 Variables and Observations

'Concept': Generalized idea of something that represents something interesting to someone, and that is likely related to other concepts.

For example, in the data set above, we could say that "How much people pay in rent" is a *concept.* It is not a variable.

'Variable': A variable is a concept that is carefully defined so that it can be precisely quantified or categorized.

How much people pay in rent is not yet a variable, because it is not precisely defined. Pay in rent for what? Apartments or houses? Number of bedrooms? When? This year, last year?

In the `countyrent.df` data frame, the variables are the column headings. To view the variables, call the `attr()` function:

```r
attr(countyrent.df, "names")
```

```
## [1] "FIPS"           "RegionName"     "State"          "MedianRent"
## [5] "UrbanInfluence" "EconTopology"   "PovertyPerc"    "MedianIncome"
```

The fourth variable is called `MedianRent` and is associated with our concept, "How much people pay in rent." Is it well defined? The `countyrent.df` data frame also includes variable labels, or descriptions. Not all data frames include variable labels It is an optional attribute of a data frame, but a very useful one for clearly defining variables. We can view the descriptions with a similar call to the `attr()` function,

```r
attr(countyrent.df, "variable.labels")
```

```
## [1] "Federal Information Processing Standards (FIPS) code uniquely identifying county"
## [2] "County name"
## [3] "State two letter code"
## [4] "Median Rent (All Rentals) in US$ in 2015"
## [5] "Degree of urban influence on county"
## [6] "Community specialization in economic output or employment"
## [7] "Percentage of population in 2015 with income below the poverty level"
## [8] "Median income in 2015"
```

You can see now that the fourth variable is precisely defined as the median rent for all rentals in U.S. dollars in 2015.

You may have noted that this description did not include an answer to the question, "Where?" This is defined based on the other variables. The second and third variables give the county name and U.S. state (and the first variable gives the unique FIPS code associated with each county). The `MedianRent` variable is the median rental price in 2015 for each given county.

'Observation': An observation is the particular value a variable takes for one of the items being measured, or the values for a set of variables for one of the items being measured.

'Observation level': A description of the unit, individual, entity, etc., that is measured.

In an R data frame, the observations are given by the rows. The observation level for the `countyrent.df` data frame is an *individual county.* Each county has a unique observation, which is a measurement for each of the variables.

If we are interested in viewing the first observation of our data frame, for Baldwin County, Alabama, we can scan across the first row in the data frame viewer, or by printing the first row and all columns to the console:

```r
slice(countyrent.df, 1)
```

```
## # A tibble: 1 x 8
##    FIPS RegionName State MedianRent
##   <chr>      <chr> <chr>      <dbl>
## 1 01003    Baldwin    AL    1010.27
## # ... with 4 more variables: UrbanInfluence <ord>, EconTopology <fctr>,
## #   PovertyPerc <dbl>, MedianIncome <dbl>
```

The code above only shows the first four variables. If you have a wider console view, you may see more, or even the whole slice.

2.2 Data and Statistics

Now that we understand what are concepts, variables, and observations, we can move on to more precisely defining data. First, let us understand the difference between a population and a sample.

'Population': All possible observations corresponding to a study. This is possibly infinite.

'Sample': A subset of the observations in population that are measured.

Usually it is impossible to collect all observations in a population, so we focus on a sample. One would hope that the sample gives an unbiased understanding of what the entire population looks like.

'Data': Measured values for all the variables for every observation in a sample. In a "flat" two-dimensional data set, the observations are typically represented by rows in a spreadsheet and the variables are represented by the columns.

'Statistics': Statistics is the study of using data from samples to make *inferences* about the population. Notice, that this goes beyond reporting computations that describe the values of the variables in the sample. The field of statistics is about using calculations from the samples to make probability statements about the entire population.

3. Scale of Measurement

Scale of measurement refers to how values for variables are defined, categorized, or quantified. There are four basic scales of measurement:

1. **'Nominal Scale':** Categorical variable where the categories cannot be ordered in any meaningful way.
2. **'Ordinal scale':** Categorical variable where the categories have a meaningful order, but the distances between values cannot be quantified.
3. **'Interval scale'** Quantitative variable where both levels and distances between levels can be quantified, but there is no meaningful zero. A "meaningful zero" is when the value zero indicates an absence of something (i.e. zero quantity).
4. **'Ratio scale'** Quantitative scale where levels and distances can be quantified and a value for zero is meaningful.

3.1 Nominal Scale

The variable, `EconTopology`, in the `countyrent.df` data frame is a categorical variable. Recall from the `variable.label` output above that `EconTopology` is a measure of community specialization in a particular type of employment or production. To see the `class` of an R objects, call the function, `class()`:

```
class(countyrent.df$EconTopology)
```

```
## [1] "factor"
```

In R-speak, the term "factor" is used to define categorical variables. You can view the different possible categories (or "levels" in R-speak) with a call to `levels()`:

```r
levels(countyrent.df$EconTopology)
```

```
## [1] "Nonspecialized"              "Mining dependent"
## [3] "Manufacturing dependent"     "Federal/State Gov dependent"
## [5] "Recreation"
```

You can notice from these descriptions that there is not any meaningful way to order the categories.

We can view some summary statistics of our nominal variable with a call to `summary()`.

```r
summary(countyrent.df$EconTopology)
```

```
##              Nonspecialized            Mining dependent
##                         225                           6
##     Manufacturing dependent Federal/State Gov dependent
##                          10                          62
##                  Recreation
##                          27
```

The `summary()` function determines what class of variable is passed to it and automatically determines what sort of summary information to give. In this case, the function outputs counts for each category. From above we see that there are 225 counties in the sample that are not specialized in any particular type of industry. There are 6 counties that are mining dependent, 10 that are manufacturing dependent, and so on.

3.2 Ordinal Scale

Satisfaction scales are typically ordinal scaled:

`Very Satisfied` > `Somewhat Satisfied` > `Somewhat Dissatisfied` > `Very Dissatisfied`.

As are frequency scales:

`Every day` > `More than once per week` > `Once per week` > `More than once per month` >

`Once per month` > `Less than once per month` > `Never`.

Ordinal data is set apart from quantitative data in that differences cannot be quantified. What is the difference between `Somewhat Satisfied` and `Somewhat Dissatisfied`? What is the difference between `Somewhat Dissatisfied` and `Very Dissatisfied`? Are these differences the same? These questions cannot be answered.

As a consequence, and strictly speaking, one cannot compute statistics such as a mean with ordinal data. Calculating a mean requires adding together numbers and dividing. Adding words, even those that can be ordered, does not have meaning. Instead of computing a mean, you can learn about central tendency with statistics such as the *median* or even the *interpolated median*. The median is the middle value when the data is ordered. Since ordinal data can be ordered, the median value can be selected.

In the data frame, `countyrent.df`, the variable `UrbanInfluence` is an ordinal variable. Let us view the class and levels:

```r
class(countyrent.df$UrbanInfluence)
```

```
## [1] "ordered" "factor"
```

```r
levels(countyrent.df$UrbanInfluence)
```

```
## [1] "In large metro area of 1+ million residents"
## [2] "In small metro area of less than 1 million residents"
## [3] "Micropolitan area adjacent to large metro area"
## [4] "Micropolitan area not adjacent to a metro area"
```

You can see from the output the class is a `factor`, meaning it is a categorical value. The output further details that it is an `ordered factor`, which is R-speak for an ordinal variable.

The levels (i.e. categories) refer to decreasing levels of urbanization. The first and highest level of urbanization refers to counties that include a large metropolitan area of one million or more residents. The other levels refer to progressively lower levels of urban influence.

We can call the `summary()` function as before to view some summary statistics for `UrbanInfluence`:

```r
summary(countyrent.df$UrbanInfluence)
```

```
##          In large metro area of 1+ million residents
##                                                  173
## In small metro area of less than 1 million residents
##                                                  153
##       Micropolitan area adjacent to large metro area
##                                                    2
##       Micropolitan area not adjacent to a metro area
##                                                    2
```

Like any other factor variable (i.e. categorical variable), we get the number of observations in each category.

Because `UrbanInfluence` is an ordinal variable, it is reasonable to compute the median. Unfortunately, we will get an error if we try:

```r
median(countyrent.df$UrbanInfluence)
```

```
## Error in median.default(countyrent.df$UrbanInfluence): need numeric data
```

The `median()` function is only defined for numeric data. One way to coerce the `median()` function to compute the median is to transform `UrbanInfluence` to numeric data. Factor variables can be transformed to numeric variables using the function `as.numeric()`.

Here is our call to the median function:

```r
median(as.numeric(countyrent.df$UrbanInfluence))
```

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

The median value equal to 1 implies the first level of `UrbanInfluence` is the median. In this case, the median urban influence in the sample is a metropolitan area of one million or more residents.

3.3 Interval Scale

An example of interval data is temperature. A temperature reading of 0 degrees Fahrenheit does not imply an absence of temperature.

For the most part, interval data can be treated like any other quantitative data. It is possible to add, subtract, multiply, and divide, and compute statistics such as means, standard deviations, correlations.

Once exception is that it is not meaningful to compute ratios. For example, you would not say that 80 deg F is "twice as hot" as 40 deg F. Even though 80 = 40 x 2, "twice as hot" on an interval scale does not have meaning. What is twice as hot as 0 deg F? What is twice as hot as -1 deg F?

There are no interval-scaled variables in the `countyrent.df` data frame.

3.4 Ratio Scale

There are three ratio variables in the `countyrent.df` data frame: `MedianRent`, `PovertyPerc`, and `MedianIncome`.

We can verify this for one of the variables with a call to `class()`:

```r
class(countyrent.df$MedianRent)
```

```
## [1] "numeric"
```

Numeric implies the variable is an interval or ratio variable. We know that rent is a ratio variable, because a value for zero literally means zero rental price, or free rent. Just because a value for zero is meaningful, does not imply that value exists in the data set.

We can view some summary statistics with a call to `MedianRent`:

```r
summary(countyrent.df$MedianRent)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   530.4   789.3  1023.1  1169.8  1356.5  4730.8
```

We want to be cautious when interpreting these values. Each county is given equal weight in these calculations, regardless of the population of the county. Also, these are rental prices for residences advertised on Zillow. Rental units in larger counties are more likely to use Zillow for advertising than in small communities.

We see that the nationwide mean of the median county rental prices is $1,169.80. The county with the lowest median rental price is has a median rent equal to $530.40. The county with the highest median rental price has median rent equal to $4,730.80.

Are you wondering what county that is in the United States where the median rental price is the highest in the country, and almost $5,000?! Let's find out!

```r
filter(countyrent.df, MedianRent>=4730.80)
```

```
##    FIPS    RegionName State MedianRent
## 1 06075 San Francisco    CA    4730.83
##                                UrbanInfluence   EconTopology PovertyPerc
## 1 In large metro area of 1+ million residents Nonspecialized        12.4
##   MedianIncome
## 1        90527
```

That's San Francisco! Median rental price is $4,730.83!

4. Filtering and Summarizing Data

Let's suppose that you are interested in understanding summary statistics for only counties with an economic specialization in recreation. There are several ways that we can filter the data to focus our analysis on this sub-sample.

4.1 Create Sub-Sample Data Frame

First, we can filter the data and save it as a separate data set. In the code that follows, we filter out the observations where `EconTopology` is equal to `"Recreation"` and save it to a new data frame we call `rent.rec.df`.

```r
rent.rec.df <- filter(countyrent.df, EconTopology=="Recreation")
```

To test for equality, we use the `==` operator, as in `EconTopology=="Recreation"`.

Now we can look at descriptive statistics for some of our variables:

```
summary(rent.rec.df$MedianRent)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   782.9   928.2  1241.7  1320.0  1436.5  3156.3
```

```
summary(rent.rec.df$UrbanInfluence)
```

```
##         In large metro area of 1+ million residents
##                                                  10
## In small metro area of less than 1 million residents
##                                                  16
##       Micropolitan area adjacent to large metro area
##                                                   1
##       Micropolitan area not adjacent to a metro area
##                                                   0
```

4.2 Filter and Pipe

Instead of saving a separate data frame (like `rent.rec.df` above), we can filter our data frame, then *pipe* the result to another function. The pipe operator is given by `%>%`. What is does is it takes the result of the function call that comes before the operator and passes that as an input to the function call that comes after the operator.

In the example below, we filter out counties with an economic specialization in recreation, then we use the `summarise()` function to calculate the means for `MedianRent` and `PovertyPerc`, saving these values as `MeanRent` and `MeanPoverty`

```
filter(countyrent.df, EconTopology=="Recreation") %>%
  summarise(MeanRent=mean(MedianRent), MeanPoverty=mean(PovertyPerc))
```

```
##   MeanRent MeanPoverty
## 1  1319.96    12.32593
```

4.3 Group and Pipe

Perhaps we are interested in counties that specialize in recreation, but we would also like to compare those results to other types of counties. Instead of filtering, we can group data by `EconTopology`, and calculate summary statistics for each group.

In the code below, we compute summary statistics for `MedianRent` for each type of county as defined in `EconTopology`. We again use pipes (`%>%` operator) to pipe the data frame to the `group_by()` function, then pipe that result to the summary function.

```
countyrent.df %>%
  group_by(EconTopology) %>%
  summarise(MeanRent=mean(MedianRent))
```

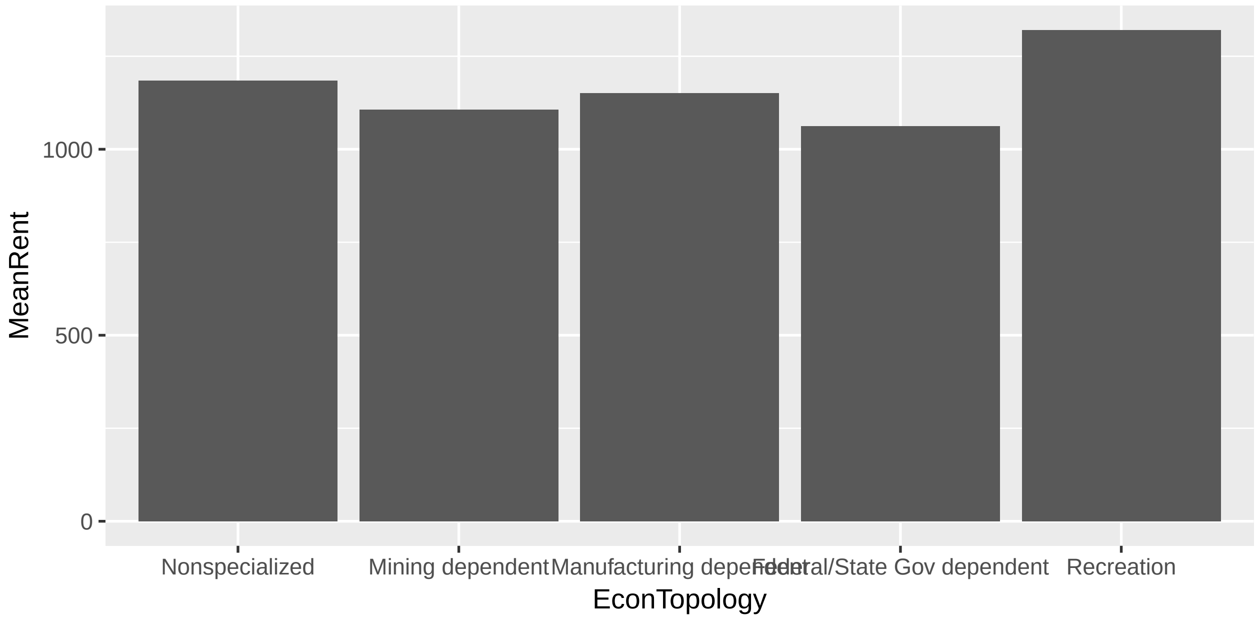
```
## # A tibble: 5 x 2
##                 EconTopology MeanRent
##                       <fctr>    <dbl>
## 1             Nonspecialized 1183.925
## 2           Mining dependent 1105.758
## 3     Manufacturing dependent 1151.169
## 4 Federal/State Gov dependent 1062.537
## 5                  Recreation 1319.960
```

Now we can compare average rent in counties specializing in recreation to other types of economic specialization. We can see from the table above, that (at least in our sample) counties that specialize in recreation have on average higher rent than other types of counties.

We will conclude with making a bar chart to visualize the results above. The code below uses the `ggplot()` function. The code below to create the chart is beyond the scope of this tutorial, so it is okay if you do not understand it.

```
# First create the summary statistics as above
# and save the result in renttable
countyrent.df %>%
  group_by(EconTopology) %>%
  summarise(MeanRent=mean(MedianRent)) ->
  renttable

# Call ggplot() with data frame renttable
ggplot(renttable, aes(x=EconTopology, y=MeanRent)) + geom_col()
```


MeanRent
Nonspecialized
Mining dependent
Manufacturing dependent
Federal/State Gov dependent
Recreation
EconTopology