Intro to R Week 3 Lab 2

Lab 2 Week 3

General codes for understanding

Reading a Data Frame from a Text File

```
read.table
read.csv
```

things to watch out for

- make sure file is in working director
- header (sets the variable names)
- set stringsAsFactors=FALSE

Reading a data file

data <- read.csv("NSYL.csv")

Using `read.csv` potentially has some problems when analyzing data. So lets download `tidyverse` and use this package to read in data.

The `tidyverse` consists of a collection of packages to do *data wrangling* in R. All this functionality also exists in base R, but `tidyverse` makes it easier, more intuitive, and sometimes also faster.

We will use this package to:

- read and write spreadsheet-like data
- extract variables
- create new variables
- select rows
- create subsets of the data

```
install.packages("tidyverse") install.packages("foreign")
```

In addition, we will review several summary characteristics of data. We will also work with the package `foreign` to read and write **dbf** files, which we will use a lot later on.

Before we start, make sure you have set the current working directory. It should contain the files **NYLS.csv** and **NYLS.dbf**.

some useful commands

view the first 6 rows
`head` (data)
structure, gives types of columns
`str`
dimension, number of rows and columns
`dim`
statistical summary of variables: min, 1st quartile, median, mean, 3rd quartils, max
`summary`

Extracting variables

sometimes you will want to extract a vector to look at just a subset of data

extracting variables columns

`data_frame$variable_name`
[[“variable,name”]] [,column_number]

extracting a data frame

`[column_number]`

Writing a Data Frame to a file

`write.csf(df,filename)`
set row.names=FALSE
will be written to workign directory

Renaming variables with `rename`

The `rename` command lets us change a variable name. Again, we first pass the tibble, and then an expression with new name = old name. One way to remember the order is to think of it as computing a new variable, only the new variable is the same as the old variable, but with a different name.

Creating new variables with `mutate`

We typically want to carry out computations with the variables in our data frame and add the results to the data frame. This can be done using base R commands, but in the tidyverse it is easily accomplished by means of the `mutate` command. Again, we pass the data frame and the expression to be calculated as new_variable = expression. To make the addition permanent, we assign the result to a data frame (could be the original data frame).

For example, say we wanted to compute the math score change, i.e., the difference between MATHSCORE2010 and MATHSCORE2008 We use `mutate` with **df** and `mathdifference = MATHSCORE2010 - MATHSCORE2008`. We assign the result back to **df**. We list the contents of **df** to check.

Now, let’s also create a logical variable that is TRUE for those people with a positive growth, say `mathscore = mathdifference > 0`. Again, we add the variable to the existing data frame by assigning the result of the `mutate` operation back to **df**. We print the result to check.

Descriptive statistics using `summarize`

The `summarize` command computes specific descriptive statistics and assigns them to a new variable. All the statistics are then combined in a data frame with a single observation.

The values are the same as what we obtained from the `summary`. In addition to the `mean`, summarize can also yield the `median`, standard deviation (`sd`), inter-quartile range (`IQR`), minimum (`min`), maximum (`max`), quantile (`quantile(variable, percentile)`), counts of logical values (`sum`).

Summaries by subset using `group_by`

The main power of `summarize` is its use in combination with `group_by`, which computes the descriptive statistics for subsets of the data. For example, say we wanted to compute the mean population separately for the poeple that saw growth and those that saw decline.

We could of course use `filter` to create two separate data frames and repeat the calculation for each of them. Instead, we use `group_by` to make the grouping internal to the data frame and then apply the `summarize`.

A very useful summary statistic is `n()` which gives the count of observations by group, contained in the variable **count**.