Data Analysis in R: Lecture Notes

PS239T

Spring 2018

For most applied research, conducting data analysis in R involves the following core tasks:

1. Constructing a dataset
2. Summarizing the structure and content of data
3. Carrying out operations and calculations across groups
4. Reshaping data to and from various formats
5. Testing relationships between variables, either descriptive or causal

This implies by extension that you will need to develop both a workflow (the process by which you conduct these tasks) and a way to keep track of the products (spreadsheets, charts, tables, etc.) of interest. For more on this way of thinking about programming, see this discussion.

As you become more advanced, I recommend switching to the above workflow. For now, let's clear our global environment and set up our working directory.

Setup environment

```r
# remove all objects
rm(list=ls())
# note that this does not remove any packages you might already have called, just objects

# set YOUR working directory
setwd(dir="~/YOUR_PATH_HERE/PS239T/04_r-data-analysis/")
```

1. Constructing a dataset

The first thing we want to do is load a dataset. This usually involves one or more of the following tasks:

a) ***Importing*** different types of data from different sources
b) ***Cleaning*** the data, including subsetting, altering values, etc.
c) ***Merging*** the data with other data

1a: Importing Data

First, let's load any packages we need.

For spreadsheet data that **is not** explicitly saved as a Microsoft Excel file, we need an additional package:

```r
# This package allows us to import non-Excel files of many different kinds:
# install.packages("foreign") # Only necessary one time
# Once it's installed, remember to load it (every time):
library(foreign)
```

Now, let's load some of the data we want to work with. In R, you can pretty much load as many datasets as you want at the same time (this is different than Stata, which some of you may be familiar with).

```r
# Basic CSV read: Import country-year data with header row, values separated by ",", decimals as "."
mydataset<-read.csv(file="data/country-year.csv", stringsAsFactors=F)
```

Let's load the PolityVI and CIRI datasets (both csvs):

```r
#impocountry.year
polity <- read.csv("data/Polity/p4v2013.csv", stringsAsFactors = F)
# take a quick peek
head(polity) # first 6 rows
```

```
##     cyear ccode scode     country year flag fragment democ autoc polity
## 1 7001800   700   AFG Afghanistan 1800    0       NA     1     7     -6
## 2 7001801   700   AFG Afghanistan 1801    0       NA     1     7     -6
## 3 7001802   700   AFG Afghanistan 1802    0       NA     1     7     -6
## 4 7001803   700   AFG Afghanistan 1803    0       NA     1     7     -6
## 5 7001804   700   AFG Afghanistan 1804    0       NA     1     7     -6
## 6 7001805   700   AFG Afghanistan 1805    0       NA     1     7     -6
##   polity2 durable xrreg xrcomp xropen xconst parreg parcomp exrec exconst
## 1      -6      NA     3      1      1      1      3       3     1       1
## 2      -6      NA     3      1      1      1      3       3     1       1
## 3      -6      NA     3      1      1      1      3       3     1       1
## 4      -6      NA     3      1      1      1      3       3     1       1
## 5      -6      NA     3      1      1      1      3       3     1       1
## 6      -6      NA     3      1      1      1      3       3     1       1
##   polcomp prior emonth eday eyear eprec interim bmonth bday byear bprec
## 1       6    NA     NA   NA    NA    NA      NA      1    1  1800     1
## 2       6    NA     NA   NA    NA    NA      NA     NA   NA    NA    NA
## 3       6    NA     NA   NA    NA    NA      NA     NA   NA    NA    NA
## 4       6    NA     NA   NA    NA    NA      NA     NA   NA    NA    NA
## 5       6    NA     NA   NA    NA    NA      NA     NA   NA    NA    NA
## 6       6    NA     NA   NA    NA    NA      NA     NA   NA    NA    NA
##   post change d4 sf regtrans
## 1   -6     88  1 NA       NA
## 2   NA     NA NA NA       NA
## 3   NA     NA NA NA       NA
## 4   NA     NA NA NA       NA
## 5   NA     NA NA NA       NA
## 6   NA     NA NA NA       NA
```

```r
# impocountry.year
ciri <- read.csv("data/CIRI/CIRI_1981_2011.csv", stringsAsFactors = F)
# take a quick peer
head(ciri)
```

```
##   X        CTRY YEAR CIRI COW POLITY UNCTRY UNREG UNSUBREG PHYSINT DISAP
## 1 1 Afghanistan 1981  101 700    700      4   142       62       0     0
## 2 2 Afghanistan 1982  101 700    700      4   142       62       0     0
## 3 3 Afghanistan 1983  101 700    700      4   142       62       0     0
## 4 4 Afghanistan 1984  101 700    700      4   142       62       0     0
## 5 5 Afghanistan 1985  101 700    700      4   142       62       0     0
## 6 6 Afghanistan 1986  101 700    700      4   142       62       0     0
##   KILL POLPRIS TORT OLD_EMPINX NEW_EMPINX ASSN FORMOV DOMMOV OLD_MOVE
## 1    0       0    0          0          2    0      0      1        0
## 2    0       0    0          2          1    0      0      0        0
## 3    0       0    0          0          0    0      0      0        0
## 4    0       0    0          1          1    0      0      0        0
## 5    0       0    0          0          0    0      0      0        0
## 6    0       0    0          0          1    0      0      0        0
##   SPEECH ELECSD OLD_RELFRE NEW_RELFRE WORKER WECON WOPOL WOSOC INJUD ccode
## 1      0      0          0          1      0     0     0     0     0   700
## 2      0      0          1          1      0     0     1     0     0   700
## 3      0      0          0          0      0     0     1     0     0   700
## 4      0      1          0          0      0     0     1     0     0   700
## 5      0      0          0          0      0     0     1     0     0   700
## 6      0      0          0          1      0     0     1     0     0   700
```

We can also import:

- Other types of Microsoft Excel files (.xls or .xlsx):
- Proprietary data formats (e.g.: .dta, .spss, .ssd) - these require the "foreign" package
- Data from the web

1b. Cleaning Data

Let's start with the Polity dataset on political regime characteristics and transitions. First, let's inspect the dataset.

```r
# Get the object class
class(polity)
```

```
## [1] "data.frame"
```

```r
# Get the object dimensionality
dim(polity) # Note this is rows by columns
```

```
## [1] 16727    36
```

```r
# Get the column names
colnames(polity)
```

```
##  [1] "cyear"    "ccode"    "scode"    "country"  "year"     "flag"    
##  [7] "fragment" "democ"    "autoc"    "polity"   "polity2"  "durable" 
## [13] "xrreg"    "xrcomp"   "xropen"   "xconst"   "parreg"   "parcomp" 
## [19] "exrec"    "exconst"  "polcomp"  "prior"    "emonth"   "eday"    
## [25] "eyear"    "eprec"    "interim"  "bmonth"   "bday"     "byear"   
## [31] "bprec"    "post"     "change"   "d4"       "sf"       "regtrans"
```

```r
# Get the row names
rownames(polity)[1:50] # Only the first 50 rows
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
## [15] "15" "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28"
## [29] "29" "30" "31" "32" "33" "34" "35" "36" "37" "38" "39" "40" "41" "42"
## [43] "43" "44" "45" "46" "47" "48" "49" "50"
```

```r
# View first six rows and all columns
head(polity)
```

```
##     cyear ccode scode     country year flag fragment democ autoc polity
## 1 7001800   700   AFG Afghanistan 1800    0       NA     1     7     -6
## 2 7001801   700   AFG Afghanistan 1801    0       NA     1     7     -6
## 3 7001802   700   AFG Afghanistan 1802    0       NA     1     7     -6
## 4 7001803   700   AFG Afghanistan 1803    0       NA     1     7     -6
## 5 7001804   700   AFG Afghanistan 1804    0        NA     1     7     -6
## 6 7001805   700   AFG Afghanistan 1805    0        NA     1     7     -6
##   polity2 durable xrreg xrcomp xropen xconst parreg parcomp exrec exconst
## 1      -6      NA     3      1      1      1      3       3     1       1
## 2      -6      NA     3      1      1      1      3       3     1       1
## 3      -6      NA     3      1      1      1      3       3     1       1
## 4      -6      NA     3      1      1      1      3       3     1       1
## 5      -6      NA     3      1      1      1      3       3     1       1
## 6      -6      NA     3      1      1      1      3       3     1       1
##   polcomp prior emonth eday eyear eprec interim bmonth bday byear bprec
## 1       6    NA     NA   NA    NA    NA      NA      1    1  1800     1
## 2       6    NA     NA   NA    NA    NA      NA     NA   NA    NA    NA
## 3       6    NA     NA   NA    NA    NA      NA     NA   NA    NA    NA
## 4       6    NA     NA   NA    NA    NA      NA     NA   NA    NA    NA
## 5       6    NA     NA   NA    NA    NA      NA     NA   NA    NA    NA
## 6       6    NA     NA   NA    NA    NA      NA     NA   NA    NA    NA
##   post change d4 sf regtrans
## 1   -6     88  1 NA       NA
## 2   NA     NA NA NA       NA
## 3   NA     NA NA NA       NA
## 4   NA     NA NA NA       NA
## 5   NA     NA NA NA       NA
## 6   NA     NA NA NA       NA
```

```
# View last six rows and all columns
tail(polity)
```

```
##          cyear ccode scode  country year flag fragment democ autoc polity
## 16722 5522008   552   ZIM Zimbabwe 2008    0        0     1     5     -4
## 16723 5522009   552   ZIM Zimbabwe 2009    0        0     3     2      1
## 16724 5522010   552   ZIM Zimbabwe 2010    0        0     3     2      1
## 16725 5522011   552   ZIM Zimbabwe 2011    0        0     3     2      1
## 16726 5522012   552   ZIM Zimbabwe 2012    0        0     3     2      1
## 16727 5522013   552   ZIM Zimbabwe 2013    2        0     5     1      4
##       polity2 durable xrreg xrcomp xropen xconst parreg parcomp exrec
## 16722      -4       9     2      1      4      2      3       3     3
## 16723       1       0     2      2      4      3      3       3     7
## 16724       1       1     2      2      4      3      3       3     7
## 16725       1       2     2      2      4      3      3       3     7
## 16726       1       3     2      2      4      3      3       3     7
## 16727       4       0     2      2      4      5      3       3     7
##       exconst polcomp prior emonth eday eyear eprec interim bmonth bday
## 16722       2       6    NA     NA   NA    NA    NA      NA     NA   NA
## 16723       3       6    -4      2   10  2009     1      NA      2   11
## 16724       3       6    NA     NA   NA    NA    NA      NA     NA   NA
## 16725       3       6    NA     NA   NA    NA    NA      NA     NA   NA
## 16726       3       6    NA     NA   NA    NA    NA      NA     NA   NA
## 16727       5       6     1      5   21  2013     1      NA      5   22
##       byear bprec post change d4 sf regtrans
## 16722    NA    NA   NA     NA NA NA       NA
## 16723  2009     1    1      5  1 NA        2
## 16724    NA    NA   NA     NA NA NA       NA
## 16725    NA    NA   NA     NA NA NA       NA
## 16726    NA    NA   NA     NA NA NA       NA
## 16727  2013     1    4      3  1 NA        1
```

```r
# Get detailed column-by-column information
str(polity)
```

```
## 'data.frame':    16727 obs. of  36 variables:
##  $ cyear   : int  7001800 7001801 7001802 7001803 7001804 7001805 7001806 7001807 7001808 7001809 ..
##  $ ccode   : int  700 700 700 700 700 700 700 700 700 700 ...
##  $ scode   : chr  "AFG" "AFG" "AFG" "AFG" ...
##  $ country : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
##  $ year    : int  1800 1801 1802 1803 1804 1805 1806 1807 1808 1809 ...
##  $ flag    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fragment: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ democ   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ autoc   : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ polity  : int  -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 ...
##  $ polity2 : int  -6 -6 -6 -6 -6 -6 -6 -6 -6 -6 ...
##  $ durable : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ xrreg   : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ xrcomp  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ xropen  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ xconst  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ parreg  : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ parcomp : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ exrec   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ exconst : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ polcomp : int  6 6 6 6 6 6 6 6 6 6 ...
##  $ prior   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ emonth  : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ eday    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ eyear   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ eprec   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ interim : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ bmonth  : int  1 NA NA NA NA NA NA NA NA NA ...
##  $ bday    : int  1 NA NA NA NA NA NA NA NA NA ...
##  $ byear   : int  1800 NA NA NA NA NA NA NA NA NA ...
##  $ bprec   : int  1 NA NA NA NA NA NA NA NA NA ...
##  $ post    : int  -6 NA NA NA NA NA NA NA NA NA ...
##  $ change  : int  88 NA NA NA NA NA NA NA NA NA ...
##  $ d4      : int  1 NA NA NA NA NA NA NA NA NA ...
##  $ sf      : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ regtrans: int  NA NA NA NA NA NA NA NA NA NA ...
```

We'll first want to subset, and maybe alter some values.

```r
# find column names
names(polity)
```

```
##  [1] "cyear"    "ccode"    "scode"    "country"  "year"     "flag"
##  [7] "fragment" "democ"    "autoc"    "polity"   "polity2"  "durable"
## [13] "xrreg"    "xrcomp"   "xropen"   "xconst"   "parreg"   "parcomp"
## [19] "exrec"    "exconst"  "polcomp"  "prior"    "emonth"   "eday"
## [25] "eyear"    "eprec"    "interim"  "bmonth"   "bday"     "byear"
## [31] "bprec"    "post"     "change"   "d4"       "sf"       "regtrans"
```

```r
# quickly summarize the year column
summary(polity$year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1800    1890    1955    1937    1988    2013
```

```
# subset the data
country.year <- subset(polity, year>1979 & year<2013, select=c(ccode,country,year,polity,democ,autoc))

# take a look
head(country.year)
```

```
##     ccode     country year polity democ autoc
## 181   700 Afghanistan 1980    -66   -66   -66
## 182   700 Afghanistan 1981    -66   -66   -66
## 183   700 Afghanistan 1982    -66   -66   -66
## 184   700 Afghanistan 1983    -66   -66   -66
## 185   700 Afghanistan 1984    -66   -66   -66
## 186   700 Afghanistan 1985    -66   -66   -66
```

```
# quickly summarize the polity column
summary(country.year$polity)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -88.000  -7.000   4.000  -1.833   9.000  10.000
```

```
# apply NA values
country.year$polity[country.year$polity < -10] <- NA
summary(country.year$polity)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -10.000  -6.000   5.000   1.808   9.000  10.000     239
```

```
  # Note how the summary has changed - minimum value and NAs have changed

# get a list of all the countries in the dataset
head(unique(country.year$country))
```

```
## [1] "Afghanistan" "Albania"     "Algeria"     "Angola"      "Argentina"
## [6] "Armenia"
```

```
# delete records
country.year <- country.year[-which(country.year$country=="Sudan-North"),]

# different way of deleting the same records
country.year <- country.year[!(country.year$country=="Sudan-North"),]
```

1c. Merging data

Oftentimes, we want to combine data from multiple datasets to construct our own dataset. This is called **merging**. In order to merge datasets, at least one column has be to shared between them. This column is usually a vector of keys, or unique identifiers, by which you can match observations.

For our data, each observation is a "country-year". But the "country" column is problematic. Some datasets might use "United States", others "USA", or "United States of America" – this makes it difficult to merge datasets.

So we'll use the "ccode" column, which is a numeric code commonly used to identify countries, along with "year". Together, this makes a unique id for each observation.

The first thing we want to do is inspect the dataset we want to merge and make it mergeable.

```
# get column names
names(ciri) # to be merged
```

```
##  [1] "X"          "CTRY"       "YEAR"       "CIRI"       "COW"
##  [6] "POLITY"     "UNCTRY"     "UNREG"      "UNSUBREG"   "PHYSINT"
## [11] "DISAP"      "KILL"       "POLPRIS"    "TORT"       "OLD_EMPINX"
## [16] "NEW_EMPINX" "ASSN"       "FORMOV"     "DOMMOV"     "OLD_MOVE"
## [21] "SPEECH"     "ELECSD"     "OLD_RELFRE" "NEW_RELFRE" "WORKER"
## [26] "WECON"      "WOPOL"      "WOSOC"      "INJUD"      "ccode"
```

```
# subset for the observations we care about (aka, we probably don't need all the variables)
ciri.subset <- subset(ciri, YEAR > 1979 & YEAR < 2013, select=c(YEAR,COW,UNREG,PHYSINT,SPEECH,NEW_EMPIN

# rename columns so that they are comparable to country.year
names(country.year)
```

```
## [1] "ccode"   "country" "year"    "polity"  "democ"   "autoc"
```

```
names(ciri.subset) <- c("year","ccode","unreg","physint","speech","new_empinx","wecon","wopol","wosoc",
names(ciri.subset)
```

```
##  [1] "year"       "ccode"      "unreg"      "physint"    "speech"
##  [6] "new_empinx" "wecon"      "wopol"      "wosoc"      "elecsd"
```

```
# merge
# merge format: merge(dataset1, dataset2, by=c(id variables), additional specifications as nec.)
country.year <- merge(country.year,ciri.subset,by=c("year","ccode"),all.x=TRUE)

# delete duplicates
which(duplicated(country.year))
```

```
##  [1]  189  336  481  628  774  920 1065 1211 1358 1505 1653
```

```
duplicates <- which(duplicated(country.year))
duplicates
```

```
##  [1]  189  336  481  628  774  920 1065 1211 1358 1505 1653
```

```
country.year <- country.year[-duplicates,]
```

We can keep doing this for many datasets until we have a brand-spanking new dataset!

Fast forward:

```
country.year <- read.csv("data/country-year.csv", stringsAsFactors = F)
names(country.year)
```

```
##  [1] "ccode"      "year"       "nyt.count"  "muslim"     "polity2"
##  [6] "physint"    "amnesty"    "statedept"  "gdp.pc.un"  "pop.wdi"
## [11] "wopol"      "wosoc"      "wecon"      "domestic9"  "region"
## [16] "idealpoint"
```

```
head(country.year)
```

```
##   ccode year nyt.count muslim polity2 physint amnesty statedept gdp.pc.un
## 1   100 1979        23      0       8      NA       5         3  1500.282
## 2   100 1980       105      0       8      NA       4         3  1752.542
## 3   100 1981        50      0       8       1       4         3  1867.043
## 4   100 1982        39      0       8       3       5         3  1954.792
## 5   100 1983        52      0       8      NA       4         3  1900.548
## 6   100 1984        66      0       8       1       4         3  1836.890
##    pop.wdi wopol wosoc wecon domestic9 region idealpoint
## 1 26326236    NA    NA    NA      1937     LA 0.12163210
## 2 26934591    NA    NA    NA      1875     LA 0.32799290
## 3 27550172     1     1     2      2750     LA 0.19557950
## 4 28173493     1     1     2      1250     LA 0.07314677
## 5 28803276     1     1     2      1250     LA 0.04930234
## 6 29438030     1     1     1      3750     LA 0.03903471
```

2. Summarizing

First let's get a quick summary of all variables.

```
summary(country.year)
```

```
##      ccode            year        nyt.count          muslim
##  Min.   : 20.0   Min.   :1979   Min.   :   1.00   Min.   :0.0000
##  1st Qu.:235.0   1st Qu.:1988   1st Qu.:   6.00   1st Qu.:0.0000
##  Median :451.0   Median :1998   Median :  21.00   Median :0.0200
##  Mean   :461.8   Mean   :1997   Mean   :  89.08   Mean   :0.2568
##  3rd Qu.:666.0   3rd Qu.:2006   3rd Qu.:  70.00   3rd Qu.:0.4800
##  Max.   :990.0   Max.   :2014   Max.   :6527.00   Max.   :1.0000
##  NA's   :36                     NA's   :690       NA's   :105
##     polity2          physint         amnesty        statedept
##  Min.   :-10.00   Min.   :0.000   Min.   :1.000   Min.   :1.000
##  1st Qu.: -6.00   1st Qu.:3.000   1st Qu.:2.000   1st Qu.:1.000
##  Median :  4.00   Median :5.000   Median :3.000   Median :2.000
##  Mean   :  1.75   Mean   :4.899   Mean   :2.712   Mean   :2.424
##  3rd Qu.:  9.00   3rd Qu.:7.000   3rd Qu.:3.000   3rd Qu.:3.000
##  Max.   : 10.00   Max.   :8.000   Max.   :5.000   Max.   :5.000
##  NA's   :1213     NA's   :1634    NA's   :1886    NA's   :867
##    gdp.pc.un            pop.wdi              wopol           wosoc
##  Min.   :    33.55   Min.   :9.471e+03   Min.   :0.000   Min.   :0.00
##  1st Qu.:   651.65   1st Qu.:1.665e+06   1st Qu.:2.000   1st Qu.:1.00
##  Median :  2004.29   Median :6.471e+06   Median :2.000   Median :1.00
##  Mean   :  7687.21   Mean   :3.107e+07   Mean   :1.782   Mean   :1.23
##  3rd Qu.:  7486.67   3rd Qu.:2.031e+07   3rd Qu.:2.000   3rd Qu.:2.00
##  Max.   :181492.97   Max.   :1.364e+09   Max.   :3.000   Max.   :3.00
##  NA's   :577         NA's   :94          NA's   :1619    NA's   :2862
##      wecon         domestic9          region            idealpoint
##  Min.   :0.000   Min.   :    0.0   Length:6380        Min.   :-2.4620
##  1st Qu.:1.000   1st Qu.:    0.0   Class :character   1st Qu.:-0.6718
##  Median :1.000   Median :    0.0   Mode  :character   Median :-0.2961
##  Mean   :1.318   Mean   :  772.4                      Mean   :-0.1095
##  3rd Qu.:2.000   3rd Qu.:  625.0                      3rd Qu.: 0.4543
##  Max.   :3.000   Max.   :31000.0                      Max.   : 2.7735
##  NA's   :1672    NA's   :701                          NA's   :113
```

Look at region:

```r
summary(country.year$region)
```

```
##    Length     Class      Mode 
##      6380 character character
```

Let's change this back to a factor.

```r
country.year$region <- as.factor(country.year$region)
summary(country.year$region)
```

```
## Africa   Asia   EECA     LA   MENA   West   NA's 
##   1586    903    853   1365    807    827     39
```

Sometimes we need to do some basic checking for the number of observations or types of observations in our dataset. To do this quickly and easily, `table()` is our friend.

Let's look the number of observations by region.

```r
table(country.year$region)
```

```
## 
## Africa   Asia   EECA     LA   MENA   West 
##   1586    903    853   1365    807    827
```

We can even divide by the total number of rows to get proportion, percent, etc.

```r
table(country.year$region)/nrow(country.year) # Shown as decimal
```

```
## 
##    Africa      Asia      EECA        LA      MENA      West 
## 0.2485893 0.1415361 0.1336991 0.2139498 0.1264890 0.1296238
```

```r
table(country.year$region)/nrow(country.year)*100 # Shown as regular percentage
```

```
## 
##   Africa     Asia     EECA       LA     MENA     West 
## 24.85893 14.15361 13.36991 21.39498 12.64890 12.96238
```

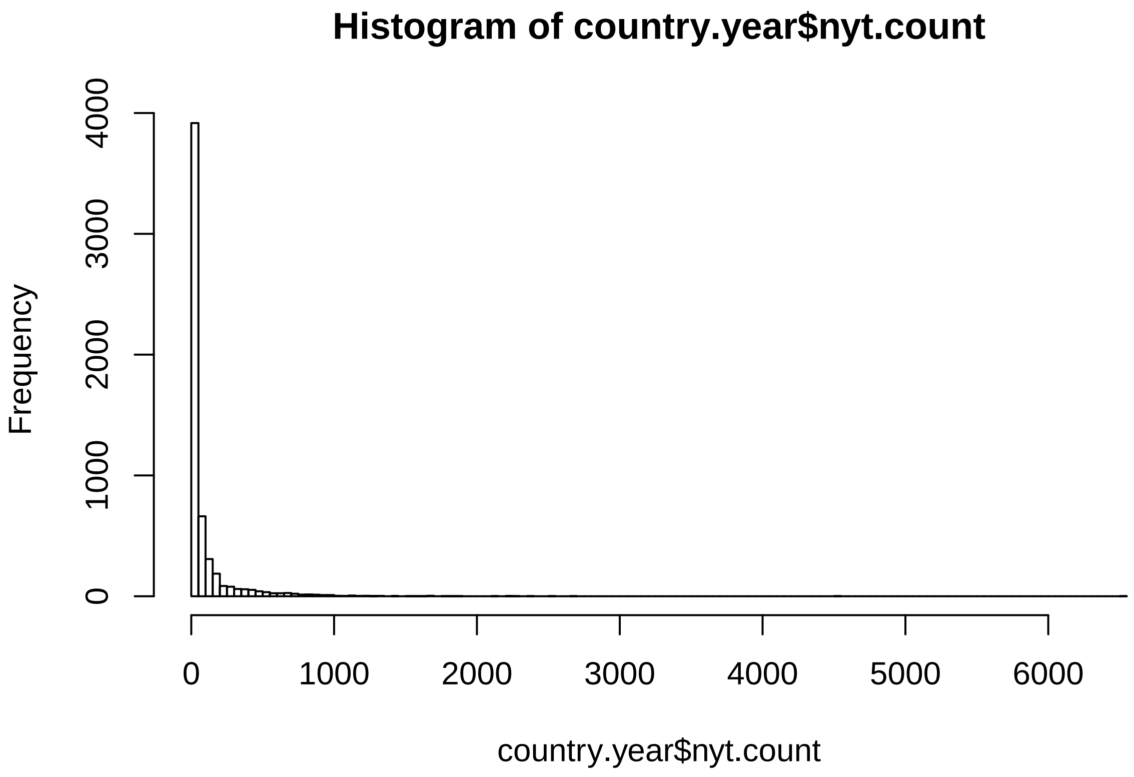
We can put two variables in there (check out what happens in early 1990s Eastern Europe!)

```r
table(country.year$year,country.year$region)
```

```
##       
##        Africa Asia EECA LA MENA West
##   1979     43   20   15 33   23   20
##   1980     44   20   15 33   23   20
##   1981     44   21   15 36   23   20
##   1982     44   21   15 37   23   20
##   1983     44   21   15 38   23   20
##   1984     44   22   15 38   23   20
##   1985     44   22   15 38   23   20
##   1986     44   22   15 38   23   20
##   1987     44   22   15 38   23   20
##   1988     44   22   15 38   23   20
##   1989     44   22   15 38   23   20
##   1990     43   21   14 37   22   20
##   1991     44   24   17 39   22   23
##   1992     44   24   24 39   22   23
##   1993     43   24   27 38   22   24
##   1994     41   23   28 36   22   24
##    1995      44    25    28  37    22    24
##    1996      44    25    28  38    22    24
##    1997      44    23    27  37    22    24
##    1998      45    23    27  38    22    24
##    1999      41    24    27  38    22    24
##    2000      43    27    29  38    22    24
##    2001      43    28    26  38    22    24
##    2002      44    29    29  39    22    25
##    2003      45    29    29  39    22    25
##    2004      45    29    29  39    22    25
##    2005      45    29    29  39    22    25
##    2006      45    29    30  39    22    25
##    2007      45    29    30  39    22    25
##    2008      45    29    30  39    22    25
##    2009      45    29    30  39    22    25
##    2010      45    29    30  39    22    25
##    2011      45    29    30  39    23    25
##    2012      45    29    30  39    23    25
##    2013      45    29    30  39    23    25
##    2014      45    29    30  39    23    25
```

Finally, let's quickly take a look at a histogram of the variable `nyt.count`:

```
hist(country.year$nyt.count, breaks = 100)
```



3. Calculating across groups

Let's say we want to look at the number of NYT articles per region.

```
summary(country.year$nyt.count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##    1.00    6.00   21.00   89.08   70.00 6527.00     690
```

```
sum(country.year$nyt.count[country.year$region=="MENA"],na.rm=T)
```

```
## [1] 146819
```

```
sum(country.year$nyt.count[country.year$region=="LA"],na.rm=T)
```

```
## [1] 62616
```

That can get tedious! A better way uses the popular new `dplyr` package, which uses a the ***split-apply-combine*** strategy. We **split** the data using some variable or variables to group our data, we **apply** some kind of function (either a built-in one, or one we write ourselves), and then we re-**combine** the data into a new dataset

```
# Install the "dplyr" package (only necessary one time)
# install.packages("dplyr")

# Load the "plyr" package (necessary every new R session)
library(dplyr)
```

```
## 
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
## 
##     filter, lag

## The following objects are masked from 'package:base':
## 
##     intersect, setdiff, setequal, union
```

All of the major dplyr functions have the same basic syntax. In this case, we're going to use summarise, which will let us do what took a few steps before in a single step!

Let's say we wanted to sum up all the NYT articles per region, and return those counts into their own dataframe. (This is often how we want our data organized for plotting, too.)

```
count.region <- summarise(group_by(country.year, region), region.sum=sum(nyt.count, na.rm=T))
count.region
```

```
## # A tibble: 7 x 2
##   region region.sum
##   <fct>       <int>
## 1 Africa      33681
## 2 Asia       104737
## 3 EECA        53569
## 4 LA          62616
## 5 MENA       146819
## 6 West       105359
## 7 <NA>           69
```

Note that many functions, like `sum`, are sensitive to missing values (NA); you should be sure to specify na.rm=T to avoid errors.

We can use even easier-to-read syntax using the chain or pipe operator (`%>%`), which passes the object or function from the line above into the next line for you:

```
count.region <- country.year %>% # put the dataset you want to work within here
  group_by(region)  %>% # next we have our grouping function
  summarise(region.sum=sum(nyt.count, na.rm=T)) # now we name our new variable and specify the function
count.region
```

```
## # A tibble: 7 x 2
##   region region.sum
##   <fct>       <int>
## 1 Africa      33681
## 2 Asia       104737
## 3 EECA        53569
## 4 LA          62616
## 5 MENA       146819
## 6 West       105359
## 7 <NA>           69
```

We can also split by multiple variables:

```
count.region.year <- country.year %>%
  group_by(region, year)  %>%
  summarise(nyt.count=sum(nyt.count, na.rm=T))
head(count.region.year)
```

```
## # A tibble: 6 x 3
## # Groups:   region [1]
##   region  year nyt.count
##   <fct>  <int>     <int>
## 1 Africa  1979       734
## 2 Africa  1980       841
## 3 Africa  1981       832
## 4 Africa  1982       718
## 5 Africa  1983       676
## 6 Africa  1984       641
```

Another very useful function is **arrange**, which orders a data frame on the basis of column contents.

```
# arrange by count, desc
by.count <- arrange(count.region.year, desc(nyt.count))
head(by.count)
```

```
## # A tibble: 6 x 3
## # Groups:   region [2]
##   region  year nyt.count
##   <fct>  <int>     <int>
## 1 MENA    2003      9742
## 2 MENA    2004      7693
## 3 MENA    2011      7047
## 4 MENA    1991      6296
## 5 MENA    2006      6268
## 6 Asia    2014      5944
```

```
# arrange by year, then count
by.year.count <- arrange(count.region.year, year, desc(nyt.count))
head(by.year.count)
```

```
## # A tibble: 6 x 3
## # Groups:   region [6]
##   region  year nyt.count
##   <fct>  <int>     <int>
## 1 Asia    1979      3140
## 2 MENA    1979      2641
## 3 West    1979      2208
## 4 LA      1979      1327
## 5 Africa  1979       734
## 6 EECA    1979       376
```

dplyr has a number of other useful functions (all of which follow the same syntax), which you'll see more of in your homework for this week.

4. Reshaping

Our country.year dataset, is currently in what's called "long" form: nyt article values are specified in the nyt.count column, and the country and year (aka, what uniquely identifies each value) are specified in each row. Let's say we wanted to make a new "wide" database, where each country has its own row, and the article counts within each year exist in multiple columns in that row.

Starting: country | year | nyt.count Brazil | 1976 | 434 Brazil | 1977 | 628 France | 1976 | 952 France | 1977 | 893

Ending: country | 1976.count | 1977.count Brazil | 424 | 628 France | 952 | 893

Base R does have commands for reshaping data (including **aggregate**, **by**, **tapply**, etc.), but each of their input commands are slightly different and are only suited for specific reshaping tasks. The **reshape2** package overcomes these argument and task inconsistencies, but is fairly slow. A recent alternative is **tidyr**, which has an easy syntax, interfaces well with dplyr, and works much faster:

```r
# install.packages("tidyr") # (only necessary one time)

# Load the "tidyr" package (necessary every new R session)
library(tidyr)
```

The package contains two major commands, **gather** (for our current purposes, that means reformat from wide to long) and **spread** (reformat from long to wide). Here, want the **spread** funciton.

```r
# here's our data, from when we used dplyr to organize it the way we wanted:
count.region.year <- country.year %>%
  group_by(region, year)  %>%
  summarise(nyt.count=sum(nyt.count, na.rm=T))

# now spread it:
region.count.wide <- spread(count.region.year, year, nyt.count)
region.count.wide[,1:10]

# write to csv
write.csv(region.count.wide,"region_year_counts.csv")
```

5. Description and Inference

Once we've imported our data, summarized it, carried out group-wise operations, and perhaps reshaped it, we may also want to assess quantitatively the relationships between our variables. We tend to describe these tests as falling into one of two categories: **descriptive**, which implies that we don't have a way to understand causation (e.g., did x cause y, or y cause x?), and **inferential**, in which we believe that we do have a way to assess whether the relationship between x and y (for instance) is **causal** (e.g., by using a randomized controlled trial).

5a. Descriptive tests

This often requires doing the following: 1) Assessing correlations 2) Carrying out classical hypothesis tests 3) Estimating regressions

Note that this class is not intended to serve as quantitative training and thus does not go into any nitty-gritty details of (for example) assessing causal identification; these are meant to be bare-bones instructions on how to run basic functions.

5a. Correlations

Often, when we want to quickly understand the relationship between two variables, and for whatever reason we want to summarize that quantitatively, we run a correlation (though, as you'll learn next week, exploring your data should be done **visually** FIRST, and correlations are no substitute for visualization).

```
# What's the relationship between population and GDP?
cor(country.year$gdp.pc.un, country.year$pop.wdi, use="pairwise.complete.obs", method="pearson")
```

```
## [1] -0.05683778
```

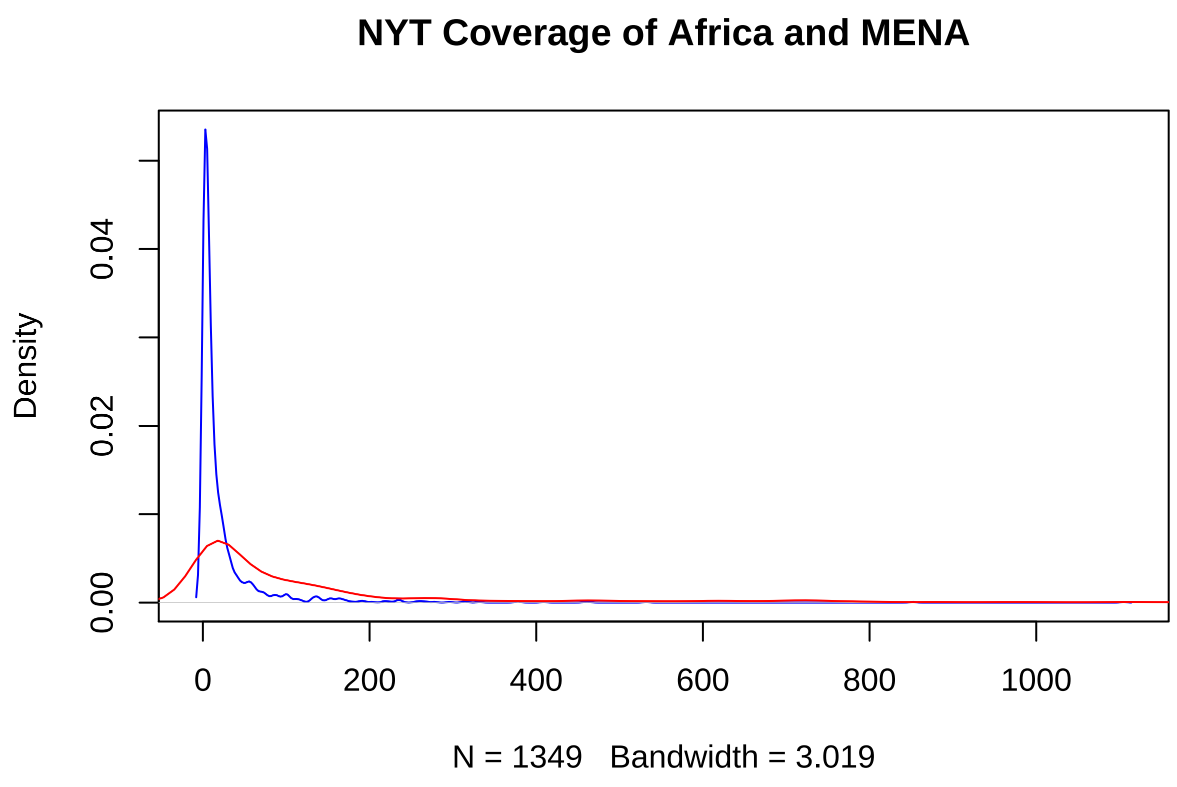
What happens if we change which observations are included? Does the correlation change if we use a different kind of test? Use `?cor` to see your options.

5b. Hypothesis Testing

Let's say we're interested in whether the New York Times covers MENA differently than the West in terms of quantity. One can test for differences in distributions in either a) their means using t-tests, or b) their entire distributions using ks-tests

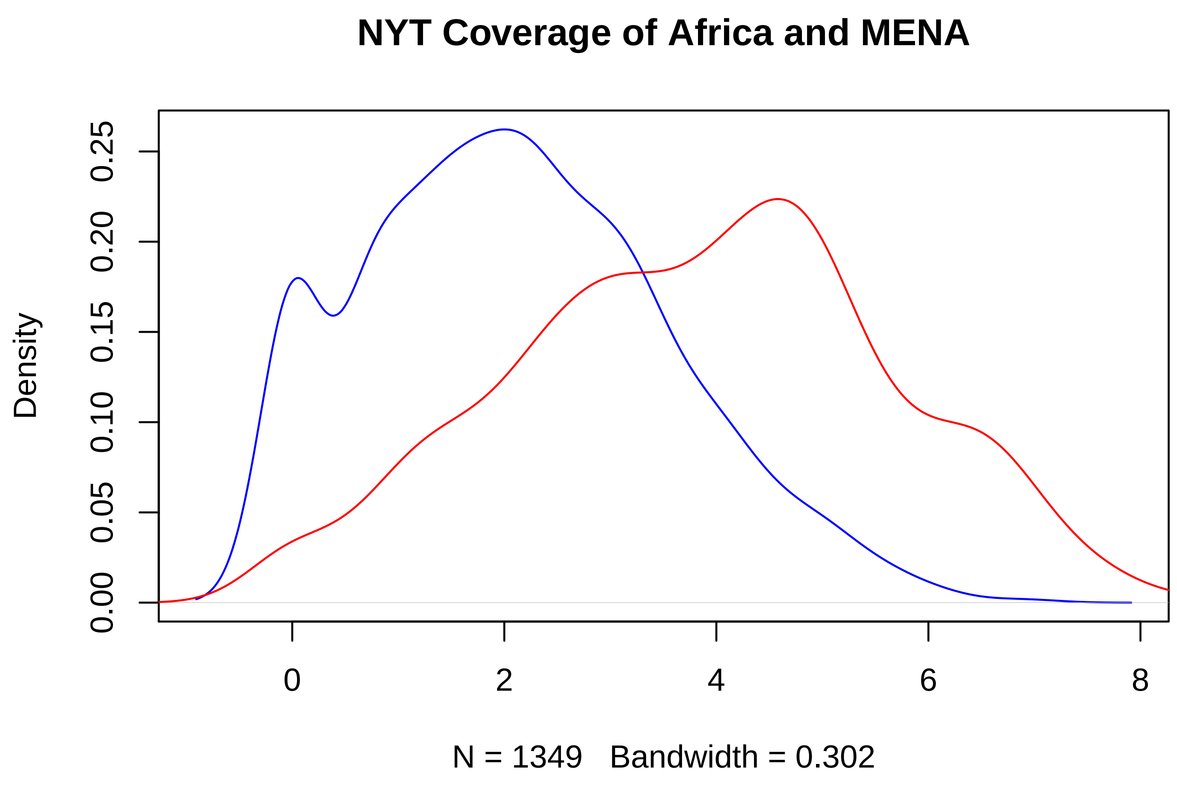
```
nyt.africa <- country.year$nyt.count[country.year$region=="Africa"]
nyt.mena <- country.year$nyt.count[country.year$region=="MENA"]

# this is a simple little plot, just to get us started:
plot(density(nyt.africa, na.rm = T), col="blue", lwd=1, main="NYT Coverage of Africa and MENA")
lines(density(nyt.mena, na.rm = T), col="red", lwd=1)
```



```r
# these are highly skewed, so let's transform taking the logarithm
nyt.africa.logged <- log(country.year$nyt.count[country.year$region=="Africa"])
nyt.mena.logged <- log(country.year$nyt.count[country.year$region=="MENA"])

plot(density(nyt.africa.logged, na.rm = T), col="blue", lwd=1, main="NYT Coverage of Africa and MENA")
lines(density(nyt.mena.logged, na.rm = T), col="red", lwd=1)
```



```r
# t test of means
t.test(x=nyt.africa.logged, y=nyt.mena.logged)
```

```
##
##  Welch Two Sample t-test
##
## data:  nyt.africa.logged and nyt.mena.logged
## t = -23.345, df = 1325.9, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.897417 -1.603250
## sample estimates:
## mean of x mean of y
##  2.112295  3.862628
```

```r
# ks tests of distributions
ks.test(x=nyt.africa.logged, y=nyt.mena.logged)
```

```
## Warning in ks.test(x = nyt.africa.logged, y = nyt.mena.logged): p-value
## will be approximate in the presence of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  nyt.africa.logged and nyt.mena.logged
## D = 0.41892, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

5c. Regressions

Running regressions in R is extremely simple, very straightforward (though doing things with standard errors requires a little extra work). **lm** is the most basic OLS regression you can run, and the most basic catch-all non-linear regression function in R is *glm*, which fits a generalized linear model with your choice of family/link function (gaussian, logit, poisson, etc.).

Remember, just like **you** are the only one who can prevent forest fires, **you** are the one responsible for thinking ahead of time about whether your regression is descriptive or inferential. This is determined by your research design, not your code.

Once you understand what you're estimating, the basic lm and glm calls look something like this:

```
lm(data=yourdata, y~x1+x2+x3+...)
glm(data=yourdata, y~x1+x2+x3+..., family=familyname)
```

In glm, here are a bunch of families and links to use (see `?family` for a full list), but some essentials are **binomial(link = "logit")**, **gaussian(link = "identity")**, and **poisson(link = "log")**

Example: suppose we want to explain the variation in NYT articles, and we think our variables give us sufficient leverage to make that assessment. A typical lm call would look something like this:

```
names(country.year)
```

```
##  [1] "ccode"      "year"       "nyt.count"  "muslim"     "polity2"
##  [6] "physint"    "amnesty"    "statedept"  "gdp.pc.un"  "pop.wdi"
## [11] "wopol"      "wosoc"      "wecon"      "domestic9"  "region"
## [16] "idealpoint"
```

```
reg <- lm(data = country.year, nyt.count ~ gdp.pc.un + pop.wdi + domestic9 + idealpoint)
summary(reg) # You'll almost always want to display your regression results using summary, since lm and
```

```
##
## Call:
## lm(formula = nyt.count ~ gdp.pc.un + pop.wdi + domestic9 + idealpoint,
##     data = country.year)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -897.9  -64.7  -32.0   -5.4 4336.3
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.245e+01  3.384e+00  12.545  < 2e-16 ***
## gdp.pc.un   8.564e-04  1.921e-04   4.457 8.49e-06 ***
## pop.wdi     4.713e-07  2.050e-08  22.985  < 2e-16 ***
## domestic9   2.938e-02  1.554e-03  18.905  < 2e-16 ***
## idealpoint  3.174e+01  3.288e+00   9.653  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 183 on 4981 degrees of freedom
##   (1394 observations deleted due to missingness)
## Multiple R-squared:  0.186,  Adjusted R-squared:  0.1854
## F-statistic: 284.6 on 4 and 4981 DF,  p-value: < 2.2e-16
```