Data Manipulation

Irfan Kanat

11/09/2015

In this document, I will try to walk you through some simple data manipulatione examples. As one might guess, data manipulation is sometimes more of an art, than science. Thus, what I can effectively teach you will be limited. We will learn some basic manipulations with dplyr package and vanilla R. dplyr provides a set of sensible functions to throw data around.

```
library(dplyr)
```

We will conduct our exercise on Worldbank GDP figures and continent information.

The GDP figures data is in GDP.csv. Let us read in the file.

```
GDP<-read.csv("GDP.csv")
head(GDP)
```

```
##       Country ISO2     X2003     X2004     X2005     X2006     X2007
## 1       Aruba   AW        NA        NA        NA        NA        NA
## 2     Andorra   AD        NA        NA        NA        NA        NA
## 3 Afghanistan   AF  1096.756  1066.685  1145.717  1173.001  1297.821
## 4      Angola   AO  3818.663  4086.858  4667.346  5444.890  6452.560
## 5     Albania   AL  6286.205  6699.225  7119.290  7537.454  8055.857
## 6  Arab World <NA> 11595.944 12407.232 12856.602 13434.034 13860.418
##       X2008     X2009     X2010     X2011     X2012
## 1        NA        NA        NA 36016.484        NA
## 2        NA        NA        NA        NA        NA
## 3  1310.717  1547.539  1637.297  1695.153  1893.076
## 4  7102.870  7038.957  7047.052  7094.084  7230.497
## 5  8747.208  9129.176  9559.157  9897.180 10157.164
## 6 14377.830 14354.814 14759.051 14825.910 15342.795
```

As you can see, the data is in what we call the wide format. Each row is a country and observations over time are in columns.

Let us also get the second data set: Continents and country codes.

```
Continents<-read.csv("continent.csv")
head(Continents)
```

```
##   ISO2 Continent
## 1   AD        EU
## 2   AE        AS
## 3   AF        AS
## 4   AG        AN
## 5   AI        AN
## 6   AL        EU
```

Combine Two Datasets

The most basic operation you can do with two datasets is to combine them. If you want to append new observations to an existing dataset, use rbind. If you want to append new variables to an existing dataset, use cbind. Note that the columns/rows need to be compatible in such combinations.

```
# Append one dataset to another
# Append rows
rbind(Continents[1:3,] ,Continents[231:233,])
```

```
##     ISO2 Continent
## 1     AD        EU
## 2     AE        AS
## 3     AF        AS
## 231   UA        EU
## 232   UG        AF
## 233   UM        OC
```

```
# Append columns
cbind(Continents[1:3,] ,Continents[231:233,])
```

```
##   ISO2 Continent ISO2 Continent
## 1   AD        EU   UA        EU
## 2   AE        AS   UG        AF
## 3   AF        AS   UM        OC
```

If you want to combine two datasets based on the values of a common column however, you will need to do a merge. Merging is similar to a join operation in SQL if you are familiar with it.

Below is the syntax for merge()

```
merge(x, y, by = intersect(names(x), names(y)),
      by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,
      sort = TRUE, suffixes = c(".x",".y"))
```

As you can see, a lot of the parameters are optional (have default values). Let us use merge to join Continents dataset to GDP dataset.

```
cData <- merge(Continents, GDP)
head(cData)
```

```
##   ISO2 Continent              Country      X2003      X2004      X2005
## 1   AD        EU              Andorra         NA         NA         NA
## 2   AE        AS United Arab Emirates 110549.415 111543.925 103139.799
## 3   AF        AS          Afghanistan   1096.756   1066.685   1145.717
## 4   AG        AN  Antigua and Barbuda  19566.329  20395.483  21414.230
## 5   AL        EU              Albania   6286.205   6699.225   7119.290
## 6   AM        AS              Armenia   4181.720   4635.303   5296.814
##       X2006     X2007     X2008     X2009     X2010     X2011     X2012
## 1        NA        NA        NA        NA        NA        NA        NA
## 2 96399.737 83655.038 73611.390 61725.280 57379.972 56376.770 57044.578
## 3  1173.001  1297.821  1310.717  1547.539  1637.297  1695.153  1893.076
## 4 24016.311 26007.825 25736.016 22388.957 20567.359 19987.924 20577.292
## 5  7537.454  8055.857  8747.208  9129.176  9559.157  9897.180 10157.164
## 6  6019.860  6877.382  7382.525  6357.829  6507.914  6812.352  7290.639
```

I did not need to specify which column to merge on as by has a default value of intersect(names(x), names(y)). This means, if there are common column names between two datasets the two will be merged on common column names.

Let us go over some of the more commonly used parameters.

by, by.x, by.y: column name to merge on between quotation marks. If the two datasets have different names, use by.x and by.y to separately specify the column names.

```
merge(Continents, GDP, by = "ISO2")
```

all, all.x, all.y: It determines what to do with rows that can not be matched in both datasets. From an SQL perspective, the all parameters specify the type of join operation. all.x = TRUE left join (keep rows from left table even if not matched), all.y = TRUE left join, and all = TRUE for an outer join.

Subset Rows Based on Column Values

If you want to select certain rows of output based on a column, this is what you do. When we wanted to filter certain observations in the first session we used indexing with logical operations before.

Let us select observations in Oceania first.

First let us see how this is done in R:

```
# displaying the first 6 columns to conserve space
# !is.na bit is required due to how R matches the == with NA's
cData[cData$Continent == "OC" & !is.na(cData$Continent),1:6]
```

```
##     ISO2 Continent                   Country     X2003     X2004     X2005
## 9     AS        OC            American Samoa        NA        NA        NA
## 11    AU        OC                 Australia 37035.053 38129.815 38840.241
## 60    FJ        OC                      Fiji  6804.258  7149.390  7168.896
## 61    FM        OC    Micronesia, Fed. Sts.  3320.158  3220.141  3301.392
## 75    GU        OC                      Guam        NA        NA        NA
## 98    KI        OC                  Kiribati  1801.492  1825.938  1791.892
## 123   MH        OC          Marshall Islands  3182.515  3183.256  3271.808
## 129   MP        OC Northern Mariana Islands        NA        NA        NA
## 138   NC        OC             New Caledonia        NA        NA        NA
## 145   NZ        OC               New Zealand 29754.461 30355.860 30984.473
## 149   PF        OC          French Polynesia        NA        NA        NA
## 150   PG        OC          Papua New Guinea  1756.367  1760.134  1779.310
## 157   PW        OC                     Palau 14695.686 15798.908 15837.395
## 165   SB        OC           Solomon Islands  1509.057  1543.241  1587.110
## 189   TO        OC                     Tonga  4955.380  4973.058  5059.245
## 192   TV        OC                    Tuvalu  3189.964  3128.561  2995.404
## 203   VU        OC                   Vanuatu  2508.348  2542.075  2609.884
## 204   WS        OC                     Samoa  4969.090  5166.430  5347.258
```

A better way is to subset the data (subset is part of the base package):

```
subset(cData[,1:6], Continent == "OC")
```

```
##     ISO2 Continent                   Country     X2003     X2004     X2005
```

```
## 9     AS        OC           American Samoa        NA        NA        NA
## 11    AU        OC                Australia 37035.053 38129.815 38840.241
## 60    FJ        OC                     Fiji  6804.258  7149.390  7168.896
## 61    FM        OC    Micronesia, Fed. Sts.  3320.158  3220.141  3301.392
## 75    GU        OC                     Guam        NA        NA        NA
## 98    KI        OC                 Kiribati  1801.492  1825.938  1791.892
## 123   MH        OC         Marshall Islands  3182.515  3183.256  3271.808
## 129   MP        OC Northern Mariana Islands        NA        NA        NA
## 138   NC        OC             New Caledonia        NA        NA        NA
## 145   NZ        OC              New Zealand 29754.461 30355.860 30984.473
## 149   PF        OC          French Polynesia        NA        NA        NA
## 150   PG        OC          Papua New Guinea  1756.367  1760.134  1779.310
## 157   PW        OC                     Palau 14695.686 15798.908 15837.395
## 165   SB        OC          Solomon Islands  1509.057  1543.241  1587.110
## 189   TO        OC                     Tonga  4955.380  4973.058  5059.245
## 192   TV        OC                    Tuvalu  3189.964  3128.561  2995.404
## 203   VU        OC                   Vanuatu  2508.348  2542.075  2609.884
## 204   WS        OC                     Samoa  4969.090  5166.430  5347.258
```

With dplyr:

```
filter(cData[,1:6], Continent == "OC")
```

```
##    ISO2 Continent                 Country     X2003     X2004     X2005
## 1    AS        OC           American Samoa        NA        NA        NA
## 2    AU        OC                Australia 37035.053 38129.815 38840.241
## 3    FJ        OC                     Fiji  6804.258  7149.390  7168.896
## 4    FM        OC    Micronesia, Fed. Sts.  3320.158  3220.141  3301.392
## 5    GU        OC                     Guam        NA        NA        NA
## 6    KI        OC                 Kiribati  1801.492  1825.938  1791.892
## 7    MH        OC         Marshall Islands  3182.515  3183.256  3271.808
## 8    MP        OC Northern Mariana Islands        NA        NA        NA
## 9    NC        OC             New Caledonia        NA        NA        NA
## 10   NZ        OC              New Zealand 29754.461 30355.860 30984.473
## 11   PF        OC          French Polynesia        NA        NA        NA
## 12   PG        OC          Papua New Guinea  1756.367  1760.134  1779.310
## 13   PW        OC                     Palau 14695.686 15798.908 15837.395
## 14   SB        OC          Solomon Islands  1509.057  1543.241  1587.110
## 15   TO        OC                     Tonga  4955.380  4973.058  5059.245
## 16   TV        OC                    Tuvalu  3189.964  3128.561  2995.404
## 17   VU        OC                   Vanuatu  2508.348  2542.075  2609.884
## 18   WS        OC                     Samoa  4969.090  5166.430  5347.258
```

You can also filter based on multiple columns. Let us say we are interested in countries in Oceania that are rich (GDP greater than 3rd quartile).

```
cData[cData$Continent == "OC" & !is.na(cData$Continent) & cData$X2011 > 23000 & !is.na(cData$X2011),]
```

```
##     ISO2 Continent     Country    X2003    X2004    X2005    X2006
## 11    AU        OC   Australia 37035.05 38129.81 38840.24 39416.04
## 145   NZ        OC New Zealand 29754.46 30355.86 30984.47 31182.26
##        X2007    X2008    X2009    X2010    X2011    X2012
## 11  40643.45 41311.94 41170.05 41329.95 41706.00 42529.87
## 145 31953.38 31058.21 31398.28 31227.55 31683.45 32281.25
```

I believe you would agree that, it is not very convenient. Filter to the rescue.

```
filter(cData, Continent == "OC" & X2011 > 23000)
```

```
##   ISO2 Continent     Country    X2003    X2004    X2005    X2006    X2007
## 1   AU        OC   Australia 37035.05 38129.81 38840.24 39416.04 40643.45
## 2   NZ        OC New Zealand 29754.46 30355.86 30984.47 31182.26 31953.38
##      X2008    X2009    X2010    X2011    X2012
## 1 41311.94 41170.05 41329.95 41706.00 42529.87
## 2 31058.21 31398.28 31227.55 31683.45 32281.25
```

Selecting Certain Columns

Let us say we are interested only in the GDP figures and not in any of the country identifiers. We would want to select only certain columns.

Traditional R ways:

```
# Limiting number of rows to 3 to conserve space
cData[1:3,4:13] # Indexing by column numbers
```

```
##        X2003      X2004      X2005     X2006     X2007     X2008     X2009
## 1         NA         NA         NA        NA        NA        NA        NA
## 2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280
## 3   1096.756   1066.685   1145.717  1173.001  1297.821  1310.717  1547.539
##       X2010     X2011     X2012
## 1        NA        NA        NA
## 2 57379.972 56376.770 57044.578
## 3  1637.297  1695.153  1893.076
```

```
cData[1:3,-(1:3)] # Negative indexing
```

```
##        X2003      X2004      X2005     X2006     X2007     X2008     X2009
## 1         NA         NA         NA        NA        NA        NA        NA
## 2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280
## 3   1096.756   1066.685   1145.717  1173.001  1297.821  1310.717  1547.539
##       X2010     X2011     X2012
## 1        NA        NA        NA
## 2 57379.972 56376.770 57044.578
## 3  1637.297  1695.153  1893.076
```

```
cData[1:3,grep("X", colnames(cData))] # Another way based on partial matching column name
```

```
##        X2003      X2004      X2005     X2006     X2007     X2008     X2009
## 1         NA         NA         NA        NA        NA        NA        NA
## 2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280
## 3   1096.756   1066.685   1145.717  1173.001  1297.821  1310.717  1547.539
##       X2010     X2011     X2012
## 1        NA        NA        NA
## 2 57379.972 56376.770 57044.578
## 3  1637.297  1695.153  1893.076
```

subset can also handle this:

```
subset(cData[1:3,], select = -c(ISO2, Continent, Country)) # Drop these columns
```

```
##        X2003      X2004      X2005     X2006     X2007     X2008     X2009
## 1         NA         NA         NA        NA        NA        NA        NA
## 2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280
## 3   1096.756   1066.685   1145.717  1173.001  1297.821  1310.717  1547.539
##       X2010     X2011     X2012
## 1        NA        NA        NA
## 2 57379.972 56376.770 57044.578
## 3  1637.297  1695.153  1893.076
```

dplyr way:

```
select(cData[1:3,], X2003:X2012) # All columns between X2003 and X2012
```

```
##        X2003      X2004      X2005     X2006     X2007     X2008     X2009
## 1         NA         NA         NA        NA        NA        NA        NA
## 2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280
## 3   1096.756   1066.685   1145.717  1173.001  1297.821  1310.717  1547.539
##       X2010     X2011     X2012
## 1        NA        NA        NA
## 2 57379.972 56376.770 57044.578
## 3  1637.297  1695.153  1893.076
```

```
select(cData[1:3,], -(ISO2:Country))
```

```
##        X2003      X2004      X2005     X2006     X2007     X2008     X2009
## 1         NA         NA         NA        NA        NA        NA        NA
## 2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280
## 3   1096.756   1066.685   1145.717  1173.001  1297.821  1310.717  1547.539
##       X2010     X2011     X2012
## 1        NA        NA        NA
## 2 57379.972 56376.770 57044.578
## 3  1637.297  1695.153  1893.076
```

Aggregating based on Groups

Let us say we want to calculate the average GDP per continent in 2011 and the number of countries in each continent.

R way:

```
Cont1 <- aggregate(cData$X2011 ~ cData$Continent, FUN=function(x)mean(x, na.rm=T))
Cont1
```

```
##   cData$Continent cData$X2011
## 1              AF    5098.267
## 2              AN   18527.820
## 3              AS   23932.691
## 4              EU   29737.968
## 5              OC    9593.810
## 6              SA   12191.071
```

```
Cont2 <- aggregate(cData$X2011 ~ cData$Continent, FUN=function(x) length(x))
Cont2
```

```
##   cData$Continent cData$X2011
## 1              AF          51
## 2              AN          26
## 3              AS          46
## 4              EU          40
## 5              OC          13
## 6              SA          11
```

```
Cont <- merge(Cont1, Cont2, by='cData$Continent')
Cont
```

```
##   cData$Continent cData$X2011.x cData$X2011.y
## 1              AF      5098.267            51
## 2              AN     18527.820            26
## 3              AS     23932.691            46
## 4              EU     29737.968            40
## 5              OC      9593.810            13
## 6              SA     12191.071            11
```

```
rm(Cont1, Cont2)
```

dplyr way:

```
# Create grouped data
contiData <- group_by(cData, Continent)
contiData
```

```
## Source: local data frame [208 x 13]
## Groups: Continent
##
##    ISO2 Continent              Country      X2003      X2004      X2005
## 1    AD        EU              Andorra         NA         NA         NA
## 2    AE        AS United Arab Emirates 110549.415 111543.925 103139.799
## 3    AF        AS          Afghanistan   1096.756   1066.685   1145.717
## 4    AG        AN  Antigua and Barbuda  19566.329  20395.483  21414.230
## 5    AL        EU              Albania   6286.205   6699.225   7119.290
## 6    AM        AS              Armenia   4181.720   4635.303   5296.814
## 7    AO        AF               Angola   3818.663   4086.858   4667.346
## 8    AR        SA            Argentina         NA         NA         NA
## 9    AS        OC       American Samoa         NA         NA         NA
## 10   AT        EU              Austria  39732.713  40555.386  41142.303
## ..  ...       ...                  ...        ...        ...        ...
## Variables not shown: X2006 (dbl), X2007 (dbl), X2008 (dbl), X2009 (dbl),
##   X2010 (dbl), X2011 (dbl), X2012 (dbl)
```

```
# Create variables on the fly
summarise(contiData, count=n(), GDP2012 = mean(X2012, na.rm = T))
```

```
## Source: local data frame [6 x 3]
##
##   Continent count   GDP2012
## 1        AF    52  5424.917
## 2        AN    31 17795.502
## 3        AS    49 24344.030
## 4        EU    46 29655.267
## 5        OC    18  9795.367
## 6        SA    12 12541.870
```

Converting Data to Long Format

So far our data had remained in wide format, yet for most statistical analysis it is more convenient to have it in long format.

Let us learn the use of reshape function. Below is the syntax.

```
reshape(data, varying = NULL, v.names = NULL, timevar = "time",
        idvar = "id", ids = 1:NROW(data),
        times = seq_along(varying[[1]]),
        drop = NULL, direction, new.row.names = NULL,
        sep = ".",
        split = if (sep == "") {
            list(regexp = "[A-Za-z][0-9]", include = TRUE)
        } else {
            list(regexp = sep, include = FALSE, fixed = TRUE)}
        )
```

data: data.frame to be reshaped. varying: names of variables that refer to single variables in long format. idvar: identifier for observations.

```
cData_Long <- reshape(cData, varying=4:13, direction="long", sep="")
head(cData_Long)
```

```
##        ISO2 Continent              Country time          X id
## 1.2003   AD        EU              Andorra 2003         NA  1
## 2.2003   AE        AS United Arab Emirates 2003 110549.415  2
## 3.2003   AF        AS          Afghanistan 2003   1096.756  3
## 4.2003   AG        AN  Antigua and Barbuda 2003  19566.329  4
## 5.2003   AL        EU              Albania 2003   6286.205  5
## 6.2003   AM        AS              Armenia 2003   4181.720  6
```

```
# Drop the unnecessary id column
cData_Long <- subset(cData_Long, select = -c(id, Country))
# Drop the missing observations
cData_Long <- na.exclude(cData_Long)
# Get rid of empty levels
cData_Long$ISO2 <- droplevels(cData_Long$ISO2)
# Rename variable X to GDP
colnames(cData_Long)[4] <- "GDP"
# Sort based on ISO2 and Year
cData_Long<-cData_Long[order(cData_Long$ISO2,cData_Long$ISO2),]
head(cData_Long)
```

```
##        ISO2 Continent time       GDP
## 2.2003   AE        AS 2003 110549.41
## 2.2004   AE        AS 2004 111543.92
## 2.2005   AE        AS 2005 103139.80
## 2.2006   AE        AS 2006  96399.74
## 2.2007   AE        AS 2007  83655.04
## 2.2008   AE        AS 2008  73611.39
```

Traditional Transformations

Often times we will find ourselves creating transformed variables. For logs and other linear transformations, our job is easy. We can just use the function name like so:

```
cData_Long$logGDP<-log(cData_Long$GDP)
head(cData_Long)
```

```
##        ISO2 Continent time       GDP   logGDP
## 2.2003   AE        AS 2003 110549.41 11.61322
## 2.2004   AE        AS 2004 111543.92 11.62217
## 2.2005   AE        AS 2005 103139.80 11.54384
## 2.2006   AE        AS 2006  96399.74 11.47626
## 2.2007   AE        AS 2007  83655.04 11.33446
## 2.2008   AE        AS 2008  73611.39 11.20656
```

What is more intriguing is to create transformations based on a grouping variable such as cumulative sums for each country. There are many ways to do this, here is one that is done with vanilla R:

```
# ASSUMING YOUR DATA IS SORTED
cData_Long$cumGDP <- ave(cData_Long$GDP, cData_Long$ISO2, FUN=cumsum)
```

Or lagged values for each country.

```
# ASSUMING YOUR DATA IS SORTED
# Write a function to lag
# A function, given a vector, shifts every observation by 1 and drops the last one.
lg <- function(x) c(NA,x[1:(length(x)-1)])
lg(1:10) # See how it works
```

```
##  [1] NA  1  2  3  4  5  6  7  8  9
```

```
# Run the function with group averages function
cData_Long$lagGDP <- ave(cData_Long$GDP, cData_Long$ISO2, FUN=lg)
```

```
## Warning in `split<-.default`(`*tmp*`, g, value = lapply(split(x, g), FUN)):
## number of items to replace is not a multiple of replacement length
```

```
head(cData_Long)
```

```
##        ISO2 Continent time       GDP   logGDP   cumGDP    lagGDP
## 2.2003   AE        AS 2003 110549.41 11.61322 110549.4        NA
## 2.2004   AE        AS 2004 111543.92 11.62217 222093.3 110549.41
## 2.2005   AE        AS 2005 103139.80 11.54384 325233.1 111543.92
## 2.2006   AE        AS 2006  96399.74 11.47626 421632.9 103139.80
## 2.2007   AE        AS 2007  83655.04 11.33446 505287.9  96399.74
## 2.2008   AE        AS 2008  73611.39 11.20656 578899.3  83655.04
```

Save dataset for later use.

```
write.csv(cData_Long, file="cDataLong.csv", row.names=F)
```
