Week 4. Big Data Analytics - `data.frame` manipulation with `dplyr`

Hyeonsu B. Kang
hyk149@eng.ucsd.edu

April 2016

1 Dplyr

In the last lecture we have seen how to index an individual cell in a data frame, extract a vector of values from a column of a data frame, and subset a data frame using the age-weight-height data set. Today, we will learn how to use a package (`dplyr`) to efficiently manipulate data frames.

Download the data file from `https://goo.gl/pgMbKj` and place it in the working directory of your R project. Loading a csv file is pretty simple, you can use the built-in `read` function of R.

```
> getwd() # find your working directory
# place the data file in the directory and
> data = read.csv("")
> data
```

`NOTE`: You can set the working directory using `setwd(<new path>)`

1.1 Sanity Check

Let's first see how the data set is structured:

```
> nrow(data)
[1] 1704
> ncol(data)
[1] 6
> colnames(data)
[1] "country"   "year"      "pop"       "continent" "lifeExp"
[6] "gdpPercap"
```

There are 1,704 observations and 6 variables. A reasonable check might be to see whether observations have correct years, bigger than 0 population, life expectancy and GDP per capita.

```
> sum(data$year > 2016 | data$pop <= 0 | data$lifeExp <= 0 | data$gdpPercap <= 0)
[1] 0
```

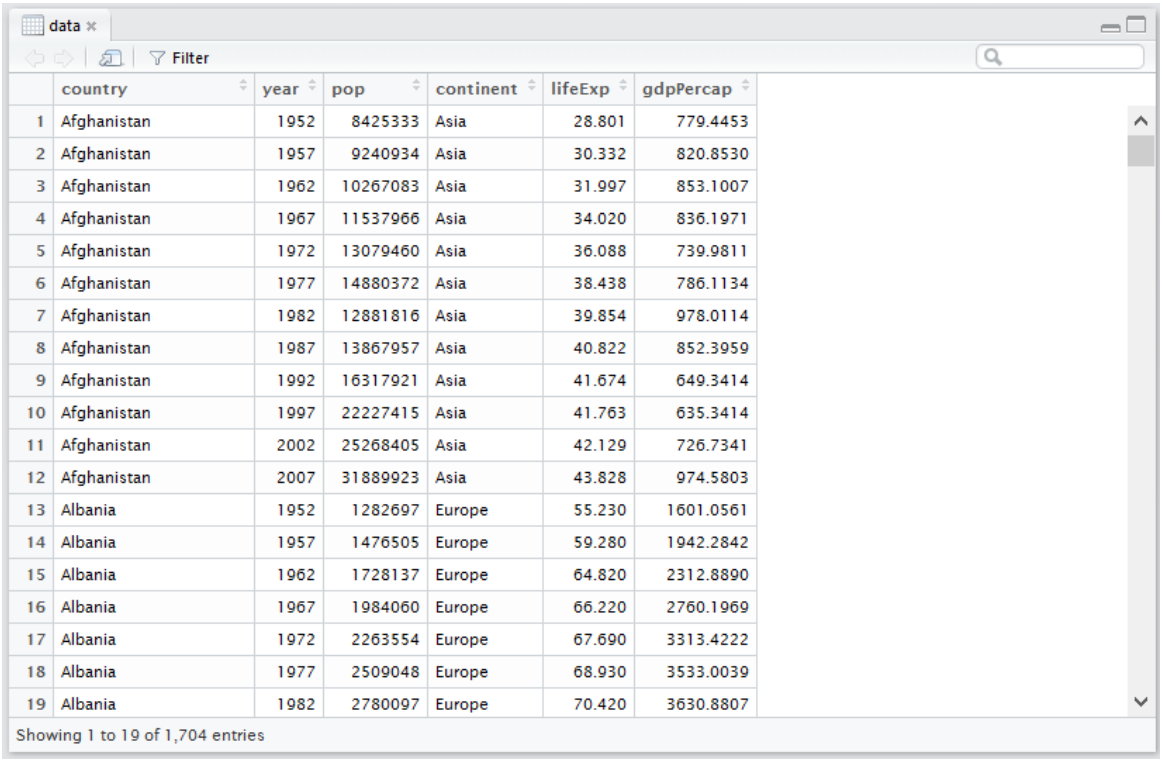
Good. The data does not seem to have invalid observations. Now, installing the `dplyr` package is possible as follows:

```
> install.packages("dplyr")
> library("dplyr")
```

1.2 `group_by()` and `summarize()`

Let us first generate some simple statistics. How can we compute yearly mean GDP per capita of different countries? If we were to use the elementary data frame manipulation operations, we would do something like below using logical indexing:

```
> unique(data$year)
 [1] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2007
> Y1952=data$year == 1952
> Y1957=data$year == 1957
> Y1962=data$year == 1962
> ...
> GDP1952=mean(data[Y1952,]$gdpPercap)
> GDP1952
[1] 3725.276
```



data ×

Filter

	country	year	pop	continent	lifeExp	gdpPercap
1	Afghanistan	1952	8425333	Asia	28.801	779.4453
2	Afghanistan	1957	9240934	Asia	30.332	820.8530
3	Afghanistan	1962	10267083	Asia	31.997	853.1007
4	Afghanistan	1967	11537966	Asia	34.020	836.1971
5	Afghanistan	1972	13079460	Asia	36.088	739.9811
6	Afghanistan	1977	14880372	Asia	38.438	786.1134
7	Afghanistan	1982	12881816	Asia	39.854	978.0114
8	Afghanistan	1987	13867957	Asia	40.822	852.3959
9	Afghanistan	1992	16317921	Asia	41.674	649.3414
10	Afghanistan	1997	22227415	Asia	41.763	635.3414
11	Afghanistan	2002	25268405	Asia	42.129	726.7341
12	Afghanistan	2007	31889923	Asia	43.828	974.5803
13	Albania	1952	1282697	Europe	55.230	1601.0561
14	Albania	1957	1476505	Europe	59.280	1942.2842
15	Albania	1962	1728137	Europe	64.820	2312.8890
16	Albania	1967	1984060	Europe	66.220	2760.1969
17	Albania	1972	2263554	Europe	67.690	3313.4222
18	Albania	1977	2509048	Europe	68.930	3533.0039
19	Albania	1982	2780097	Europe	70.420	3630.8807

Showing 1 to 19 of 1,704 entries

Figure 1: Gapminder Data

As you can see, there are mainly two steps: (1) make individual variables for each of the observation year, and (2) generate statistics for each of the variables. Of course, you would have to find the unique values of the year variable beforehand. Since the structure of computation is repetitive each year, one might make a module that generates a new variable for each of the unique year numbers and then stores the statistics into it and reuse it.

`Dplyr` makes the matter even simpler, the repetitive procedures are simply done by calling the `group_by` function:

```
> GDPbyYear = data %>%
+     group_by(year) %>%
+     summarize(MeanGDP=mean(gdpPercap))
> GDPbyYear
Source: local data frame [12 x 2]

    year   MeanGDP
   (int)     (dbl)
1   1952  3725.276
2   1957  4299.408
3   1962  4725.812
4   1967  5483.653
5   1972  6770.083
6   1977  7313.166
7   1982  7518.902
8   1987  7900.920
9   1992  8158.609
10  1997  9090.175
11  2002  9917.848
12  2007 11680.072
```

There is a new notation to become familiar with: `%>%`. This `dplyr` pipeline operator passes object from the left hand side as first argument (or . argument) of the function on right hand side. For example,

```
x %>% f(y) is the same as f(x, y)
y %>% f(x, ., z) is the same as f(x, y, z)
```

The `group_by` operation by itself simply outputs the grouped data frame. Since the number of variables nor observations changes, it is a little confusing whether or not the grouping operation is performed. This can be verified by looking at the data frames' internal representation using `str()`:

```
> groupedData=data%>%group_by(year)
> str(groupedData)
Classes 'grouped_df', 'tbl_df', 'tbl' and 'data.frame': 1704 obs. of
6 variables:
> str(data)
'data.frame':   1704 obs. of  6 variables:
```

So the `summarize()` function we used before was, in fact, operated over these internal groups formed in the data frame after `group_by()`. Grouping by multiple variables is also possible. For example, we can group the GDP per capita variable with respect of continent and year. Note that the order of grouping affects the representation of the resulting data frame:

```
> GDPByContiByYear = data %>%
+     group_by(continent, year) %>%
+     summarize(meanGDP = mean(gdpPercap))
> GDPByContiByYear
Source: local data frame [60 x 3]
Groups: continent [?]

   continent  year  meanGDP
      (fctr) (int)    (dbl)
1     Africa  1952 1252.572
2     Africa  1957 1385.236
3     Africa  1962 1598.079
4     Africa  1967 2050.364
5     Africa  1972 2339.616
6     Africa  1977 2585.939
7     Africa  1982 2481.593
8     Africa  1987 2282.669
9     Africa  1992 2281.810
10    Africa  1997 2378.760
..       ...   ...      ...

> GDPByYearByConti = data %>%
+     group_by(year, continent) %>%
+     summarize(meanGDP = mean(gdpPercap))
> GDPByYearByConti
Source: local data frame [60 x 3]
Groups: year [?]

    year continent   meanGDP
   (int)    (fctr)     (dbl)
1   1952    Africa  1252.572
2   1952  Americas  4079.063
3   1952      Asia  5195.484
4   1952    Europe  5661.057
5   1952   Oceania 10298.086
6   1957    Africa  1385.236
7   1957  Americas  4616.044
8   1957      Asia  5787.733
9   1957    Europe  6963.013
10  1957   Oceania 11598.522
..   ...       ...       ...
```

1.3 `filter()`

What if we wanted to compare different continents' GDP per capita in year 2002?

```
> Y2002GDPByConti = GDPByContiByYear %>%
+     filter(year == 2002)
> Y2002GDPByConti
Source: local data frame [5 x 3]
Groups: continent [5]

  continent  year   meanGDP
     (fctr) (int)     (dbl)
1    Africa  2002  2599.385
```

```
2  Americas  2002  9287.677
3      Asia  2002 10174.090
4    Europe  2002 21711.732
5   Oceania  2002 26938.778
```

We can use the `filter()` operation. This operation performs the task that is similar to logical indexing in default R. Now we can answer questions like 'which continent had the highest/lowest GDP per capita in year 2002?' or 'which country had the highest life expectancy in year 1997?'

```
> Y2002GDPByConti[which.max(Y2002GDPByConti$meanGDP),]
Source: local data frame [1 x 3]
Groups: continent [1]

  continent  year  meanGDP
     (fctr) (int)    (dbl)
1   Oceania  2002 26938.78
```

Similarly, for the lowest GDP per capita we can use `which.min()`.

1.4 `mutate()`

Appending a new column with, for example, log value of GDP per capita is also possible. To compute and append one or more new columns to a data frame we use `mutate()`

```
> LogGDPByContiByYear = GDPByContiByYear %>%
+     mutate(logGDP=log(meanGDP))
> LogGDPByContiByYear
Source: local data frame [60 x 4]
Groups: continent [5]

   continent  year  meanGDP   logGDP
      (fctr) (int)    (dbl)    (dbl)
1     Africa  1952 1252.572 7.132955
2     Africa  1957 1385.236 7.233626
3     Africa  1962 1598.079 7.376557
4     Africa  1967 2050.364 7.625773
5     Africa  1972 2339.616 7.757742
6     Africa  1977 2585.939 7.857844
7     Africa  1982 2481.593 7.816656
8     Africa  1987 2282.669 7.733101
9     Africa  1992 2281.810 7.732724
10    Africa  1997 2378.760 7.774334
..       ...   ...      ...      ...
```

Let us generate a histogram for logged GDP per capita per country in year 2002.

```
> Y2002LogGDP = data %>%
+     mutate(logGDP=log(gdpPercap)) %>%
+     filter(year==2002)
> hist(Y2002LogGDP$logGDP)
```

The resulting histogram is fig. 2.

1.5 Practice

Problem 1. Which country had the highest life expectancy in 2002 and what was the value of it?
Problem 2. Which continent had the lowest GDP per capita in 1967?
Problem 3. How does the distribution of GDP per capita look like in 1967?

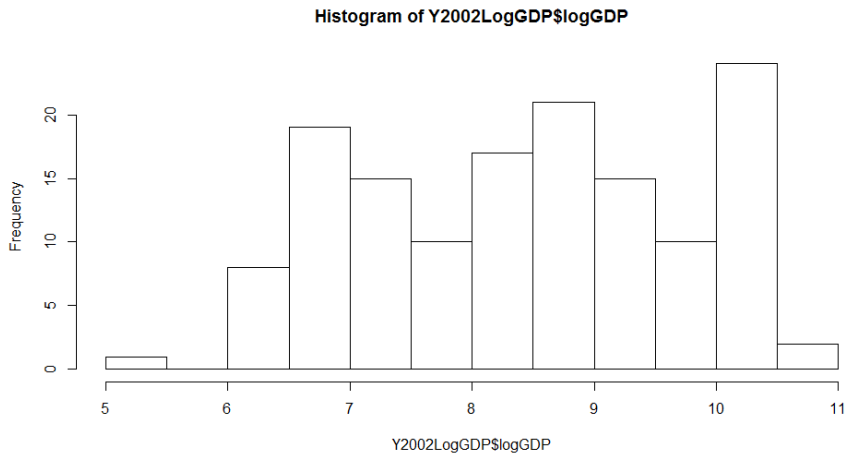


Figure 2: Logged GDP in 2002