Seminar 5

Data Manipulation and Visualization

Data Manipulation

From this R tutorial document, we will learn basic data manipulation techniques with the 'dplyr' package. dplyr provides a set of useful functions for data manipulation. Make sure to install the 'dplyr' package before using the functions in the package (install.packages('dplyr')) and to store the exercise files (GDP.csv and continent.csv) to your project working directory .

```
> library(dplyr)          # load the dplyr package
```

We will conduct our exercise on World Bank GDP figures and continent information. The GDP figures data is in GDP.csv. Let's read in the file.

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

Note: NA means an empty or NULL value.

As you can see, each row is a country and observations over time are in columns.

Let's also get the second data including 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. As we learned in the previous exercises, if you want to append new observations (rows) and variables (columns) to an existing dataset, use the rbind and cbind functions respectively. Note that the columns/rows need to be compatible in such combinations.

```
> rbind(Continents[1:3,] ,Continents[231:233,]) # Append rows
    ISO2 Continent
1     AD         EU
2     AE         AS
3     AF         AS
231   UA         EU
232   UG         AF
233   UM         OC
>
```

```
> cbind(Continents[1:3,] ,Continents[231:233,]) # Append columns
  ISO2 Continent ISO2 Continent
1   AD         EU   UA         EU
2   AE         AS   UG         AF
3   AF         AS   UM         OC
>
```

Let me remind you the merge() function . If we want to combine two datasets based on the values of a common column, we can use the merge() function. Below is the syntax for the merge() function:

```
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's use the merge function to join Continents dataset to GDP dataset.

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

We 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's 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 cannot 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 we want to select certain rows of output based on a column, this is what we do. When we wanted to filter certain observations in the last R exercises, we used indexing with logical operations before. Let's select observations in 'Oceania (OC)' first.

First, let's see how this is done in R.

With indexing and subset:

```
> # 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
> |
```

As learned, an easier way is to use the subset function:

```
> 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   X2007   X2008   X2009   X2010   X2011   X2012
11   AU         OC  Australia 37035.05 38129.81 38840.24 39416.04 40643.45 41311.94 41170.05 41329.95 41706.00 42529.87
145  NZ         OC New Zealand 29754.46 30355.86 30984.47 31182.26 31953.38 31058.21 31398.28 31227.55 31683.45 32281.25
>
```

```
> cData[1:3,4:13] # Indexing by column numbers
      X2003     X2004     X2005     X2006     X2007     X2008     X2009     X2010     X2011     X2012
1        NA        NA        NA        NA        NA        NA        NA        NA        NA        NA
2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280 57379.972 56376.770 57044.578
3   1096.756  1066.685  1145.717  1173.001  1297.821  1310.717  1547.539  1637.297  1695.153  1893.076
>
```

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   X2008   X2009   X2010   X2011   X2012
1   AU         OC  Australia 37035.05 38129.81 38840.24 39416.04 40643.45 41311.94 41170.05 41329.95 41706.00 42529.87
2   NZ         OC New Zealand 29754.46 30355.86 30984.47 31182.26 31953.38 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.

With indexing and subset:

```
> # 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     X2010     X2011     X2012
1        NA         NA         NA        NA        NA        NA        NA        NA        NA        NA
2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280 57379.972 56376.770 57044.578
3   1096.756   1066.685   1145.717  1173.001  1297.821  1310.717  1547.539  1637.297  1695.153  1893.076
>
```

```
> cData[1:3,-(1:3)] # Negative indexing
      X2003      X2004      X2005     X2006     X2007     X2008     X2009     X2010     X2011     X2012
1        NA         NA         NA        NA        NA        NA        NA        NA        NA        NA
2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280 57379.972 56376.770 57044.578
3   1096.756   1066.685   1145.717  1173.001  1297.821  1310.717  1547.539  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     X2010     X2011     X2012
1        NA         NA         NA        NA        NA        NA        NA        NA        NA        NA
2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280 57379.972 56376.770 57044.578
3   1096.756   1066.685   1145.717  1173.001  1297.821  1310.717  1547.539  1637.297  1695.153  1893.076
>
```

The subset function can also handle this:

```
> subset(cData[1:3,], select = -c(ISO2, Continent, Country)) # Drop these columns
      X2003      X2004      X2005     X2006     X2007     X2008     X2009     X2010     X2011     X2012
1        NA         NA         NA        NA        NA        NA        NA        NA        NA        NA
2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280 57379.972 56376.770 57044.578
3   1096.756   1066.685   1145.717  1173.001  1297.821  1310.717  1547.539  1637.297  1695.153  1893.076
>
```

With dplyr:

```
> select(cData[1:3,], X2003:X2012) # All columns between X2003 and X2012
      X2003      X2004      X2005     X2006     X2007     X2008     X2009     X2010     X2011     X2012
1        NA         NA         NA        NA        NA        NA        NA        NA        NA        NA
2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280 57379.972 56376.770 57044.578
3   1096.756   1066.685   1145.717  1173.001  1297.821  1310.717  1547.539  1637.297  1695.153  1893.076
>
> select(cData[1:3,], -(ISO2:Country))
      X2003      X2004      X2005     X2006     X2007     X2008     X2009     X2010     X2011     X2012
1        NA         NA         NA        NA        NA        NA        NA        NA        NA        NA
2 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280 57379.972 56376.770 57044.578
3   1096.756   1066.685   1145.717  1173.001  1297.821  1310.717  1547.539  1637.297  1695.153  1893.076
>
```

Aggregating based on Groups

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

With dplyr :

```
> # Create grouped data
> contiData <- group_by(cData, Continent); contiData
Source: local data frame [208 x 13]
Groups: Continent [6]

    ISO2 Continent            Country     X2003     X2004     X2005     X2006     X2007     X2008     X2009     X2010     X2011
   <fctr>    <fctr>             <fctr>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
1      AD        EU            Andorra        NA        NA        NA        NA        NA        NA        NA        NA        NA
2      AE        AS United Arab Emirates 110549.415 111543.925 103139.799 96399.737 83655.038 73611.390 61725.280 57379.972 56376.770
3      AF        AS        Afghanistan  1096.756  1066.685  1145.717  1173.001  1297.821  1310.717  1547.539  1637.297  1695.153
4      AG        AN  Antigua and Barbuda 19566.329 20395.483 21414.230 24016.311 26007.825 25736.016 22388.957 20567.359 19987.924
5      AL        EU            Albania  6286.205  6699.225  7119.290  7537.454  8055.857  8747.208  9129.176  9559.157  9897.180
6      AM        AS            Armenia  4181.720  4635.303  5296.814  6019.860  6877.382  7382.525  6357.829  6507.914  6812.352
7      AO        AF             Angola  3818.663  4086.858  4667.346  5444.890  6452.560  7102.870  7038.957  7047.052  7094.084
8      AR        SA          Argentina        NA        NA        NA        NA        NA        NA        NA        NA        NA
9      AS        OC     American Samoa        NA        NA        NA        NA        NA        NA        NA        NA        NA
10     AT        EU            Austria 39732.713 40555.386 41142.303 42311.039 43673.542 44157.046 42335.528 43005.543 44239.864
# ... with 198 more rows, and 1 more variables: X2012 <dbl>
>
```

```
> summarise(contiData, count=n(), GDP2011 = mean(X2011, na.rm = T))
# A tibble: 6 × 3
  Continent count    GDP2011
      <fctr> <int>      <dbl>
1        AF    52  5098.267
2        AN    31 18527.820
3        AS    49 23932.691
4        EU    46 29737.968
5        OC    18  9593.810
6        SA    12 12191.071
>
```

Visualization

The numbers of packages that handle visualizations are many, but the ggplot package is the most popular tool among them all. We will focus on ggplot and discuss plotting histograms and scatter plots with qplot and matrix plots with ggcorrplot briefly. Make sure to install the 'ggplot2' package (install.packages('ggplot2')) before using the functions in the package.

Introducing the Dataset

We will analyze the Motor Trends data (http://www.jstor.org/stable/2530428) . The dataset was compiled from 1974 issues of Motor Trends magazine and is included with R Base package. Let's start with loading the dataset. You don't need to import any data from you project folder for this exercise.

```
> data(mtcars)
```

As we learned in the last exercise on packages, you can query the documentation for almost anything, including the datasets included in packages. The document includes descriptions of the variables.

```
?mtcars
```

Let's get a summary of the data.

```
> summary(mtcars)
      mpg             cyl             disp             hp             drat             wt             qsec             vs
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0   Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
 Median :19.20   Median :6.000   Median :196.3   Median :123.0   Median :3.695   Median :3.325   Median :17.71   Median :0.0000
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7   Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0   Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
       am              gear            carb
 Min.   :0.0000   Min.   :3.000   Min.   :1.000
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
 Median :0.0000   Median :4.000   Median :2.000
 Mean   :0.4062   Mean   :3.688   Mean   :2.812
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :1.0000   Max.   :5.000   Max.   :8.000
>
```

Present the number of cars with differing number of front gears

```
> table(mtcars$gear)

 3  4  5
15 12  5
>
```

Present a frequency table comparing two categorical variables

```
> table(mtcars[,c("am","cyl")])
   cyl
am   4  6  8
  0  3  4 12
  1  8  3  2
```

Return a correlation table for first 4 variables

```
> cor(mtcars[,1:4])
            mpg        cyl       disp         hp
mpg   1.0000000 -0.8521620 -0.8475514 -0.7761684
cyl  -0.8521620  1.0000000  0.9020329  0.8324475
disp -0.8475514  0.9020329  1.0000000  0.7909486
hp   -0.7761684  0.8324475  0.7909486  1.0000000
>
```

We will plot the above outcomes with the qplot(), ggplot(), and ggcorrplot() functions!

Plotting with qplot()

qplot simplifies the ggplot functionality by automating most common tasks. We will use qplot for most common plots. Again, make sure to install the ggplot2 package (install.packages('ggplot2')) .

> library(ggplot2) # Load the ggplot package

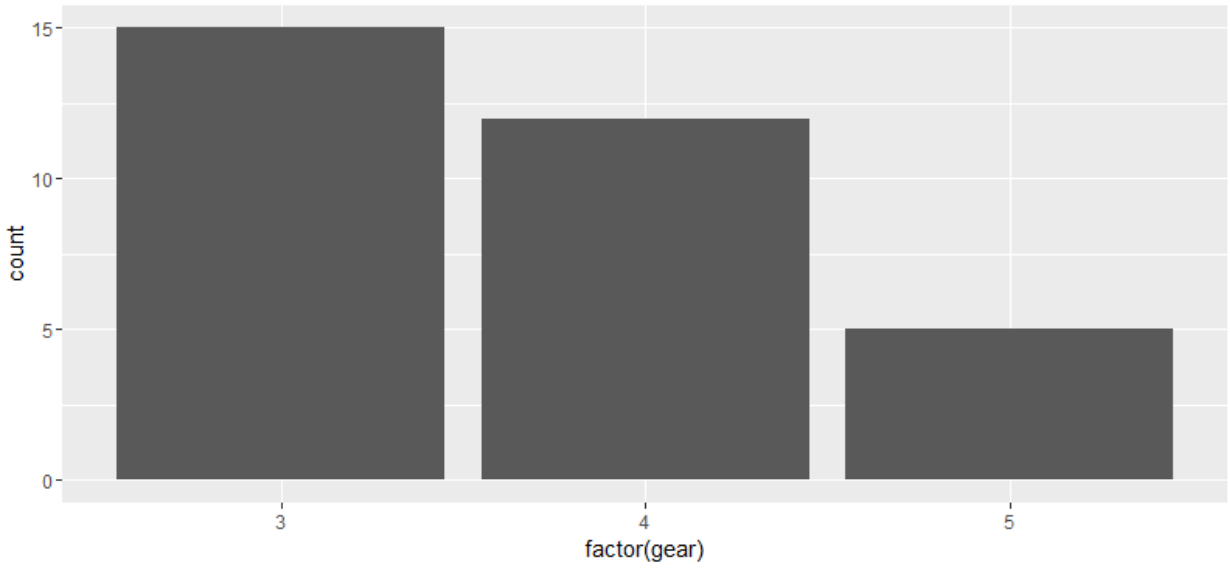
> ?qplot # Review function syntax

Histogram

W would use a histogram when you are interested in frequencies of certain categories.

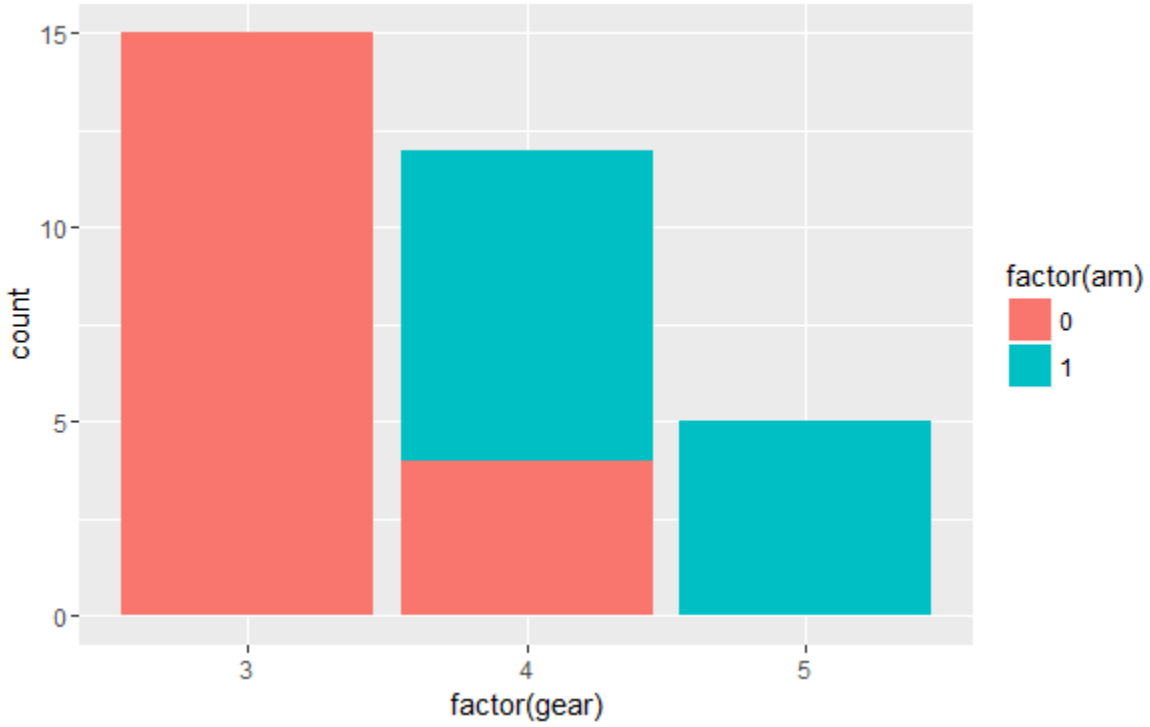
Let's report the number of cars with differing number of front gears

> qplot(factor(gear), data=mtcars, geom="bar") # I used factor to declare categorical



If we want to get fancy and want to report across two categorical variables, we can color the bars based on another variable.

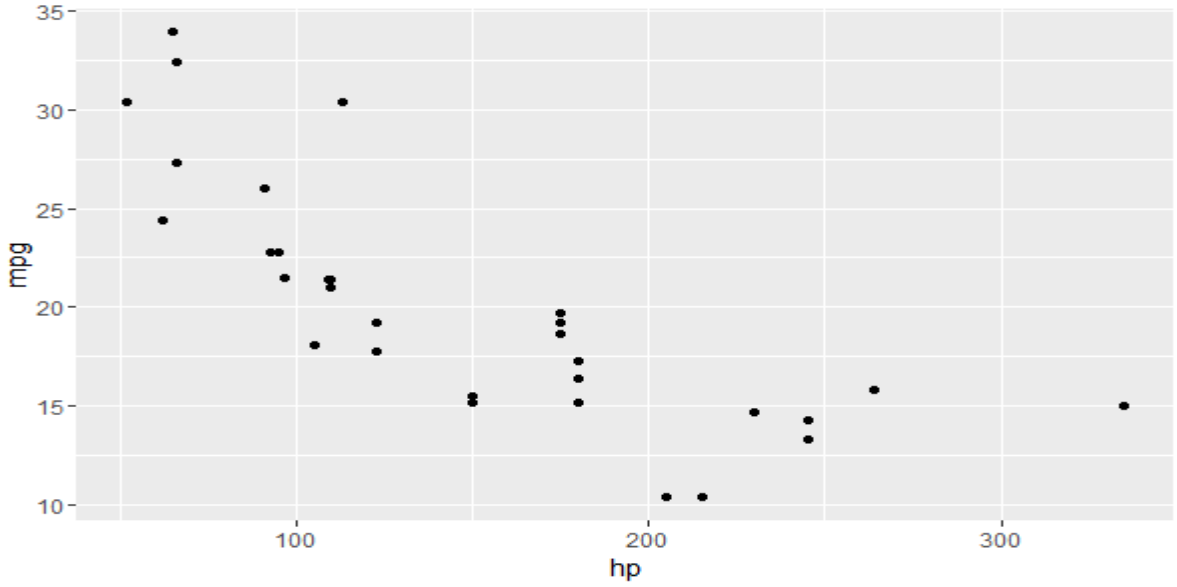
> qplot(factor(gear), data=mtcars, fill=factor(am), geom="bar")



Scatter Plots

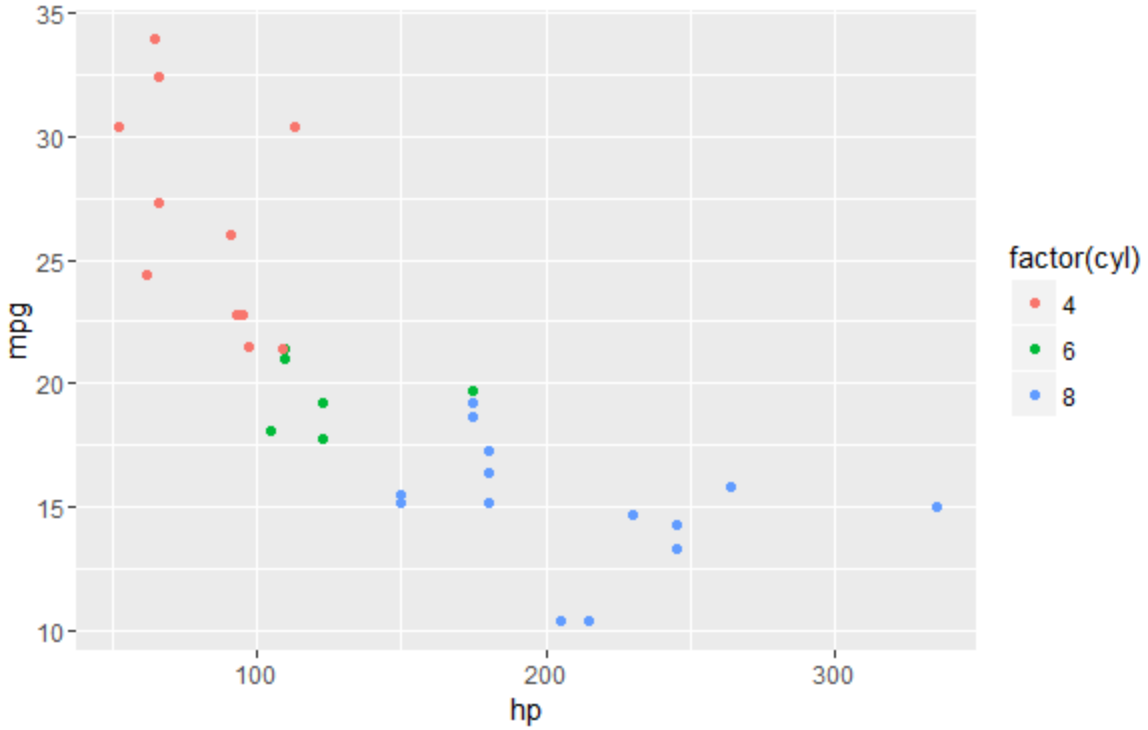
If you are interested in the relationship between two continuous variables, you can use scatter plots.

> qplot(hp, mpg, data=mtcars)



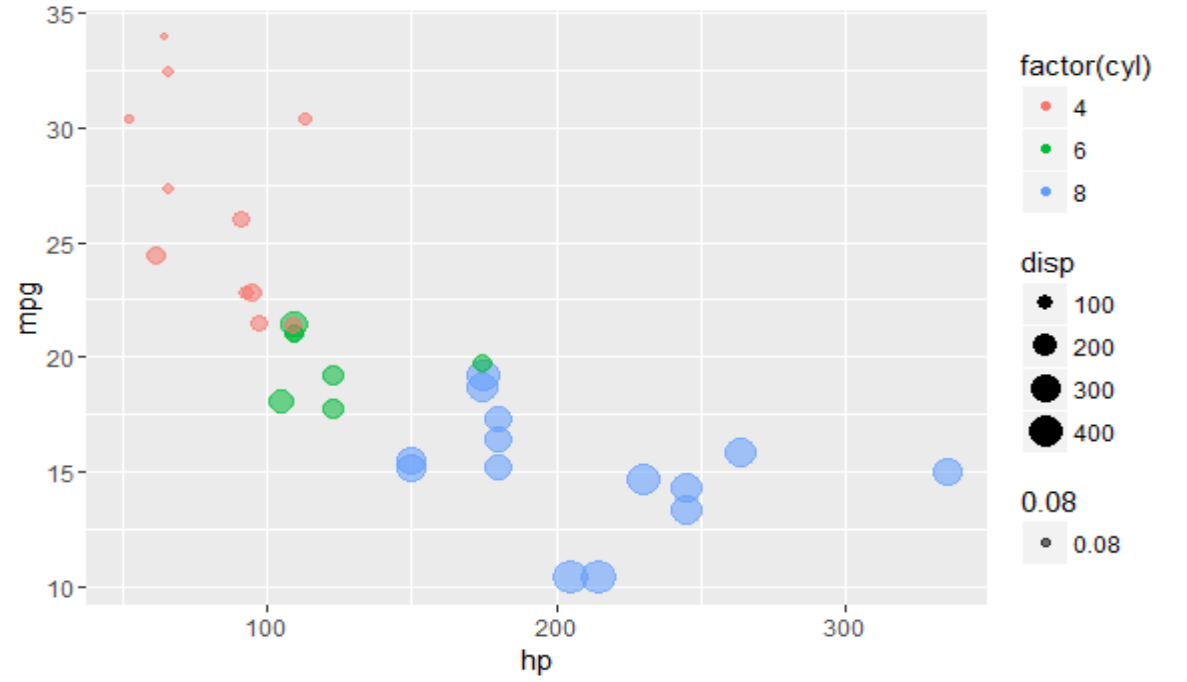
Let's impose an additional factor into the plot. Let's color the dots by the number of cylinders.

> qplot(hp, mpg, data=mtcars, color=factor(cyl))



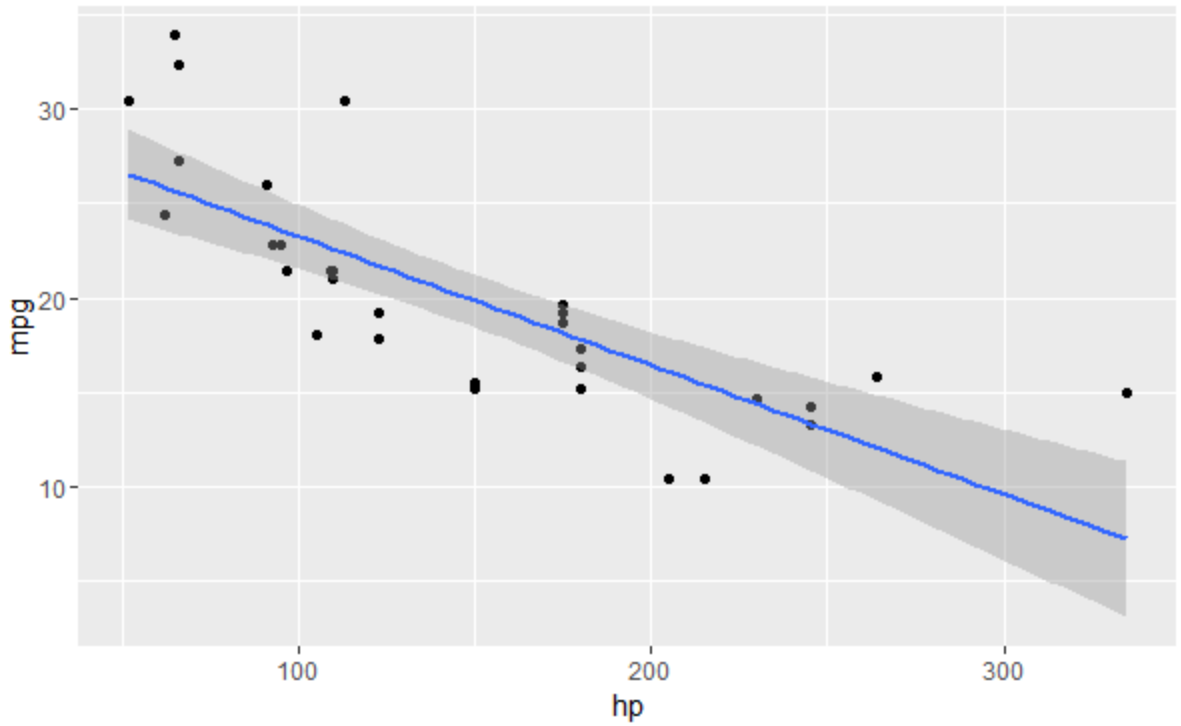
Size of dots depends on a continuous variable (displacement).

> qplot(hp, mpg, data=mtcars, color=factor(cyl), size=disp, alpha=.08)



Let us fit a regression line.

> qplot(hp, mpg, data=mtcars) + geom_smooth(method=lm)



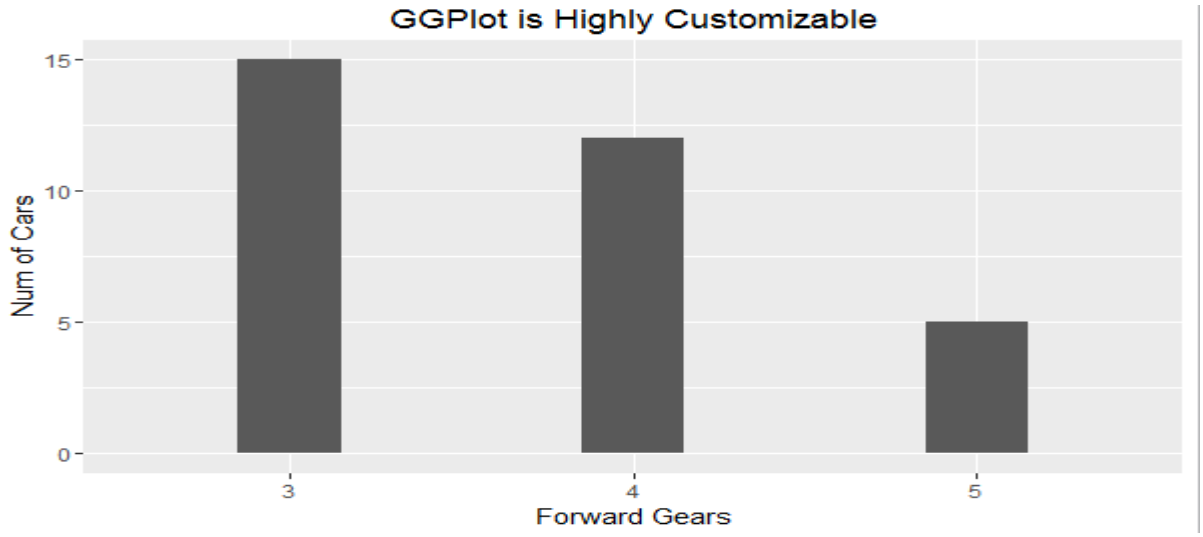
ggplot

qplot provides a convenient command for plotting. While qplot would address 90% of your plotting needs, ggplot is way more than qplot.

Histogram

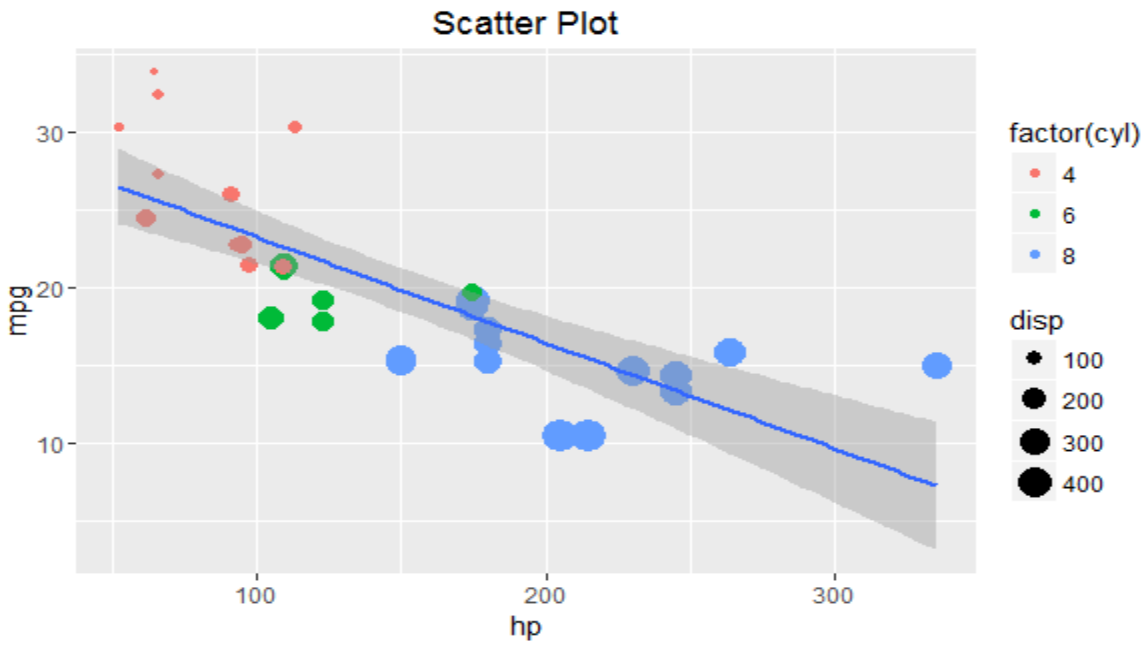
Initialize the plot with variables of interest (gear), and then instruct the ggplot function to plot bars of width .3

```
ggplot(mtcars, aes(factor(gear))) +
  geom_bar(stat = "count", width=0.3) +
  ggtitle('GGPlot is Highly Customizable') +
  xlab('Forward Gears') +
  ylab('Num of Cars')
```



Scatter Plot

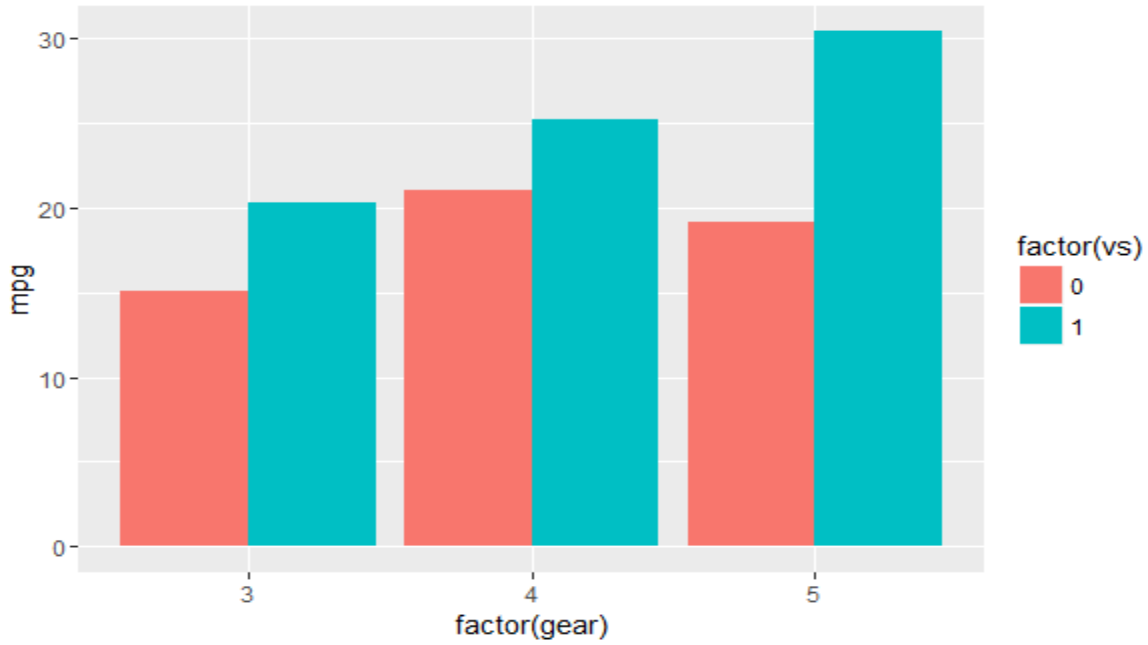
```
ggplot(mtcars, aes(x=hp, y=mpg)) +
  geom_point(aes(color=factor(cyl), size=disp)) + # For scatter plot
  geom_smooth(method=lm) + # Add a regression line
  ggtitle('Scatter Plot') # Add a title
```



Bar Charts

You can use bar charts when you want to visualize the relationship of a continuous variable over a categorical variable (eg. gender-height). Here I plot mean mpg over two categorical variables.

```
ggplot(mtcars,aes(x=factor(gear),y=mpg,fill=factor(vs)), color=factor(vs)) +
  stat_summary(fun.y=mean, position=position_dodge(), geom="bar")
```



Matrix Plot

You can use matrix plots when you want to see the correlations among multiple variables. Although the ggplot2 package provides the function generating matrix plots, it requires some pre-processing steps. You will easily create matrix plots with just a couple of steps by using the 'ggcorrplot' package provides.

Install the ggcorrplot pacakge

> install.packages('ggcorrplot')).

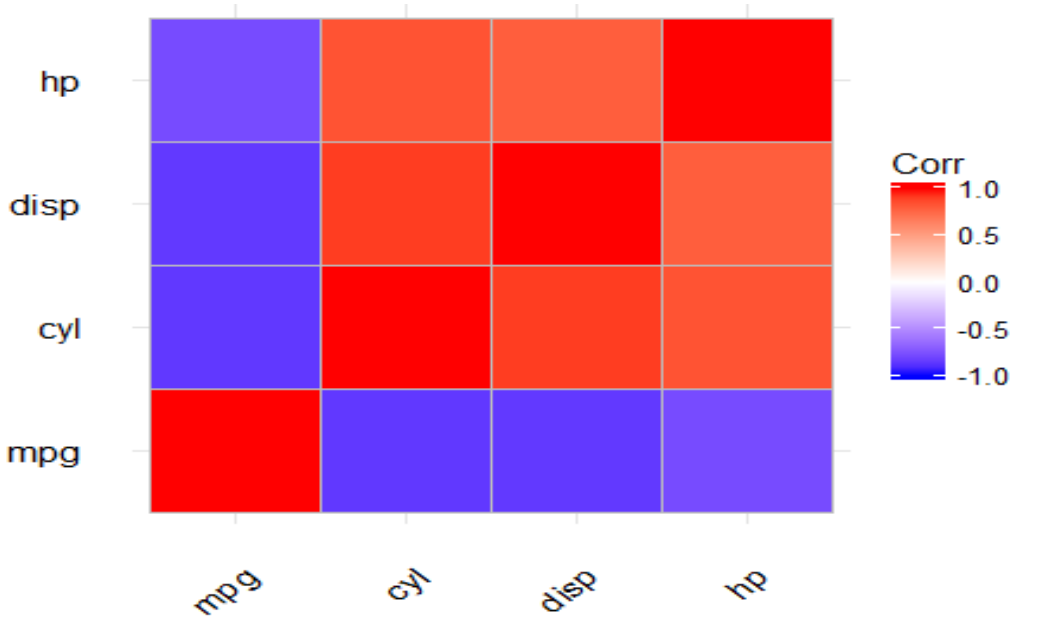
> library(ggcorrplot)

Compute a correlation matrix/table for the first 4 variables

> corr <- cor(mtcars[,1:4])

Visualize the correlation table

> ggcorrplot(corr)



You can also get the lower triangle correlation matrix with the coefficients, which is much easier to interpret.

> ggcorrplot(corr, hc.order = TRUE, type = "lower",lab = TRUE)