Rstudio Tutorial

Junchen Feng

September 21, 2016

Welcome To R & Rstudio

R is the one of most popular software for data science (+statistics).

Rstudio is one the most popular integrated development environment(IDE) for R.

R has a more active user community than almost any other statistical software and is just as good in many aspects.

In this tutorial, we will learn the basics of R and Rstudio, including:

(1) An Introduction to RStudio
(2) Install and Use Packages
(3) Get Help!
(4) A Taste of Data Science

An Intro to Rstudio

We will start with a review of Rstudio's interface. Here is everything you need for this tutorial, which is about 5% of RStudio's total functionality. You are encouraged to explore more here.

- System Navigation

  Open files, create new scripts, and save scripts. That is all for now.
- Script

  The editing area. Write down your program and save it for later(or other people).
- Data/Variables

  Data and variables are the bread and butter for statistical analysis. Open and save dataset here.
- Command line/Result

  Tinker with your code in the console cmd (shortcut for command line) and see the fruit of your labor.
- Plot

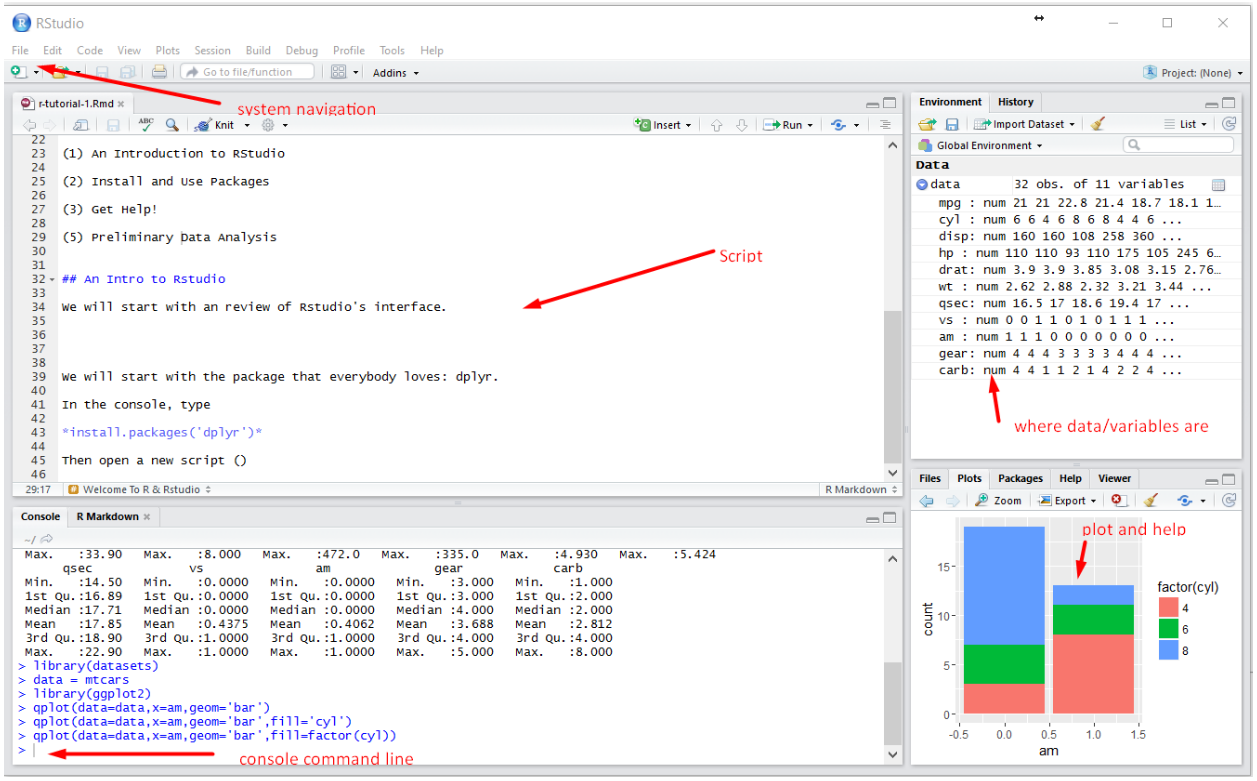
  Razzle-dazzle them with fancy lines and stuff.

Install and Use Package

The power of R lies in the ability to import packages that do all sorts of wonders. We will start with the package that everybody loves: ggplot2.

In the console, type

> *install.packages('ggplot2')*



```
(1) An Introduction to RStudio

(2) Install and Use Packages

(3) Get Help!

(5) Preliminary Data Analysis

## An Intro to Rstudio

We will start with an review of Rstudio's interface.



We will start with the package that everybody loves: dplyr.

In the console, type

*install.packages('dplyr')*

Then open a new script ()
```

```
Data
data           32 obs. of 11 variables
  mpg : num 21 21 22.8 21.4 18.7 18.1 1…
  cyl : num 6 6 4 6 8 6 8 4 4 6 ...
  disp: num 160 160 108 258 360 ...
  hp  : num 110 110 93 110 175 105 245 6…
  drat: num 3.9 3.9 3.85 3.08 3.15 2.76…
  wt  : num 2.62 2.88 2.32 3.21 3.44 ...
  qsec: num 16.5 17 18.6 19.4 17 ...
  vs  : num 0 0 1 1 0 1 0 1 1 1 ...
  am  : num 1 1 1 0 0 0 0 0 0 0 ...
  gear: num 4 4 4 3 3 3 3 4 4 4 ...
  carb: num 4 4 1 1 2 1 4 2 2 4 ...
```

```
~/
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0   Max.   :4.930   Max.   :5.424
      qsec             vs               am              gear            carb
 Min.   :14.50   Min.   :0.0000   Min.   :0.0000   Min.   :3.000   Min.   :1.000
 1st Qu.:16.89   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
 Median :17.71   Median :0.0000   Median :0.0000   Median :4.000   Median :2.000
 Mean   :17.85   Mean   :0.4375   Mean   :0.4062   Mean   :3.688   Mean   :2.812
 3rd Qu.:18.90   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :22.90   Max.   :1.0000   Max.   :1.0000   Max.   :5.000   Max.   :8.000
> library(datasets)
> data = mtcars
> library(ggplot2)
> qplot(data=data,x=am,geom='bar')
> qplot(data=data,x=am,geom='bar',fill='cyl')
> qplot(data=data,x=am,geom='bar',fill=factor(cyl))
> 
```

Figure 1:

Then open a new script (Click the icon) and choose R-script. Save it.

In the first line of the script, write

```r
library(ggplot2)
```

You will probably see some warnings in red, ignore them.

Get Help!

The best part of learning Rstudio is that you don't a teacher to become an expert.

The worst part of learning Rstudio is that you don't have someone to hold your hands.

If you are hardcore programmers, try the help function. For example:

help('qplot')

The help file of R studio gives you the description, argument explanations and a few examples.

If you are everybody else, don't panic. You can always Google or Youtube your way out of it. There is a magical website called **Stack Overflow** where all your prayers will be answered. If reading is not your thing, there is a ton of youtube tutorials that walk you through most of the elementary functionalities.

It won't be easy at the beginning if R is your first programming language. But nothing worthwhile in life comes easy. If you want to tag "data science"(or "statistics") in your LinkedIn profile, the gain is worth all the pain.

A Taste of Data Science

Your boss gives you a dataset of automobile design and performance extracted from the 1974 *Motor Trend* magazine. “Figure something out.” is the last word you heard from him.

```
autodata = mtcars
```

The dataset has the following variables:

- mpg: Miles/(US) gallon
- cyl: Number of cylinders
- disp: Displacement (cu.in.)
- hp: Gross horsepower
- drat: Rear axle ratio
- wt: Weight (1000 lbs)
- qsec: 1/4 mile time
- vs: V/S
- am: Transmission (0 = automatic, 1 = manual)
- gear: Number of forward gears
- carb: Number of carburetors

Summary Statistics

Check summary statistics is always a good habit. The summary statistics gives you the minimum, the maximum, four quartiles and the mean of all continous variables. For factor/logical varaibles, it reports frequency count.

```
summary(autodata)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  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
```

Recode

The most frequent recoding is to create factor varaibles. For example, am is a factor variable. 0 and 1 means nothing other than a place holder. The following code makes explicit what the value stands for.

```
autodata$trans = factor(autodata$am, labels=c('automatic','manual'))
```

Now let me explain each component. "autodata" is a dataframe that houses multiple variable. Most data read into R are converted into data frame by default. To extract a variable from the dataframe, use **$**. So

dataframe$variable -> autodata$trans

"factor()" is a function. It can take arguments and produce output. In this case, the output is a new variable. In some tutorial, you will see the assignment symbol is not "=" but "<-". For our purpose, it does not matter which one you choose.

output = function(arg) -> autodata$trans = factor(. . .)

"labels=. . . " changes the default value of the variable assigned by the function. Most R function set default values for auxilary varaibles so as to not overwhelm the user. You can assign custom value when you see fit.

"c('automatic','manual')" is the **list** data type. 'c' is short for concatenate, very 1980s. If you want to assign multiple values at once, usually you need to use the list. In addition, the rank of the string corresponds to the levels of the factor variable. Here 0='automatic'. If you reverse the order of the labels to c('manual','automatic'), then 0='manual'. Order matters so watch out.

Now summarize the variable gain to get the frequency counts

```
summary(autodata$trans)
```

```
## automatic    manual
##        19        13
```

Cross tabulates

For frequency count

```
table(autodata$trans, autodata$cyl)
```

```
##
##              4  6  8
##   automatic  3  4 12
##   manual     8  3  2
```

If you want to calculate cell percentage

```
ct = table(autodata$trans, autodata$cyl)
prop.table(ct)
```

```
##
##                   4       6       8
##   automatic 0.09375 0.12500 0.37500
##   manual    0.25000 0.09375 0.06250
```

If you want to have row percentage

```
prop.table(ct,1)
```

```
##
##                     4         6         8
##   automatic 0.1578947 0.2105263 0.6315789
##   manual    0.6153846 0.2307692 0.1538462
```

and column percentage is

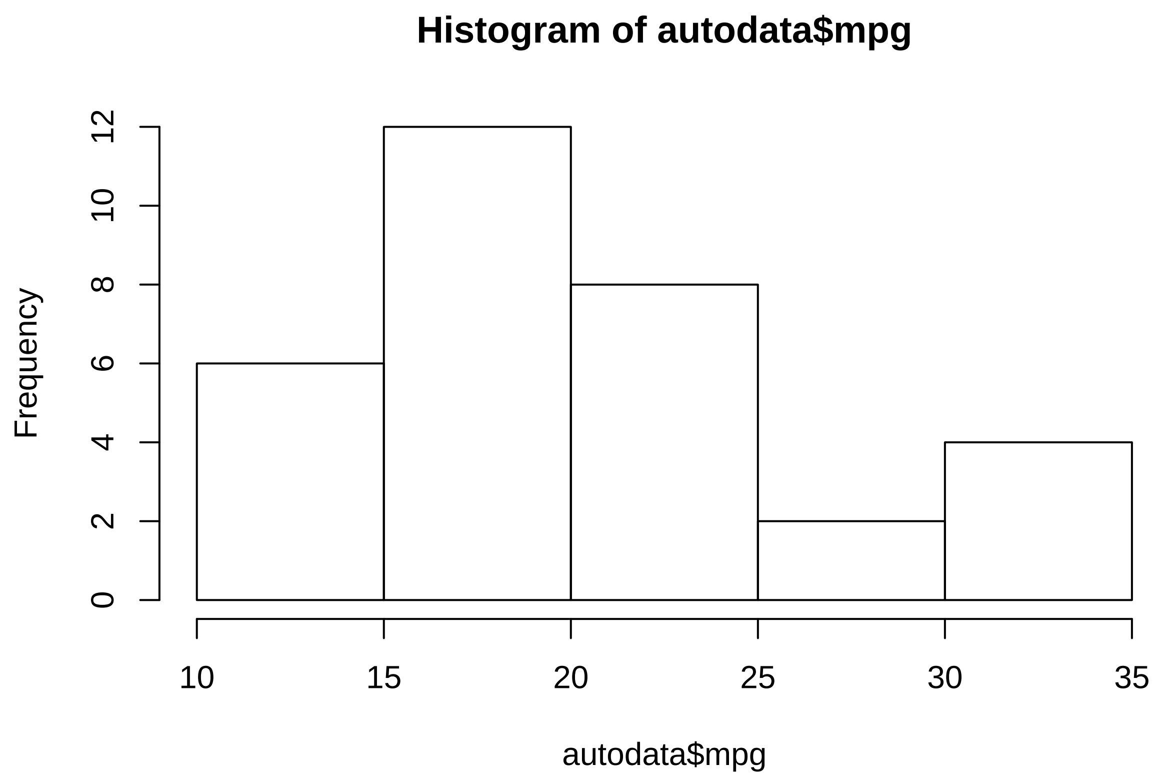
```
prop.table(ct,2)
```

```
##
##                     4         6         8
##   automatic 0.2727273 0.5714286 0.8571429
##   manual    0.7272727 0.4285714 0.1428571
```

Histogram

The native function for calculting histogram is hist. If you want to have the stats from the histogram, extract it from hist_stat.

```
hist_stat =hist(autodata$mpg)
```



I am not a big fan of the native R plotting devices, and so are most of the R users. Do the plotting like a pro with ggplot2.

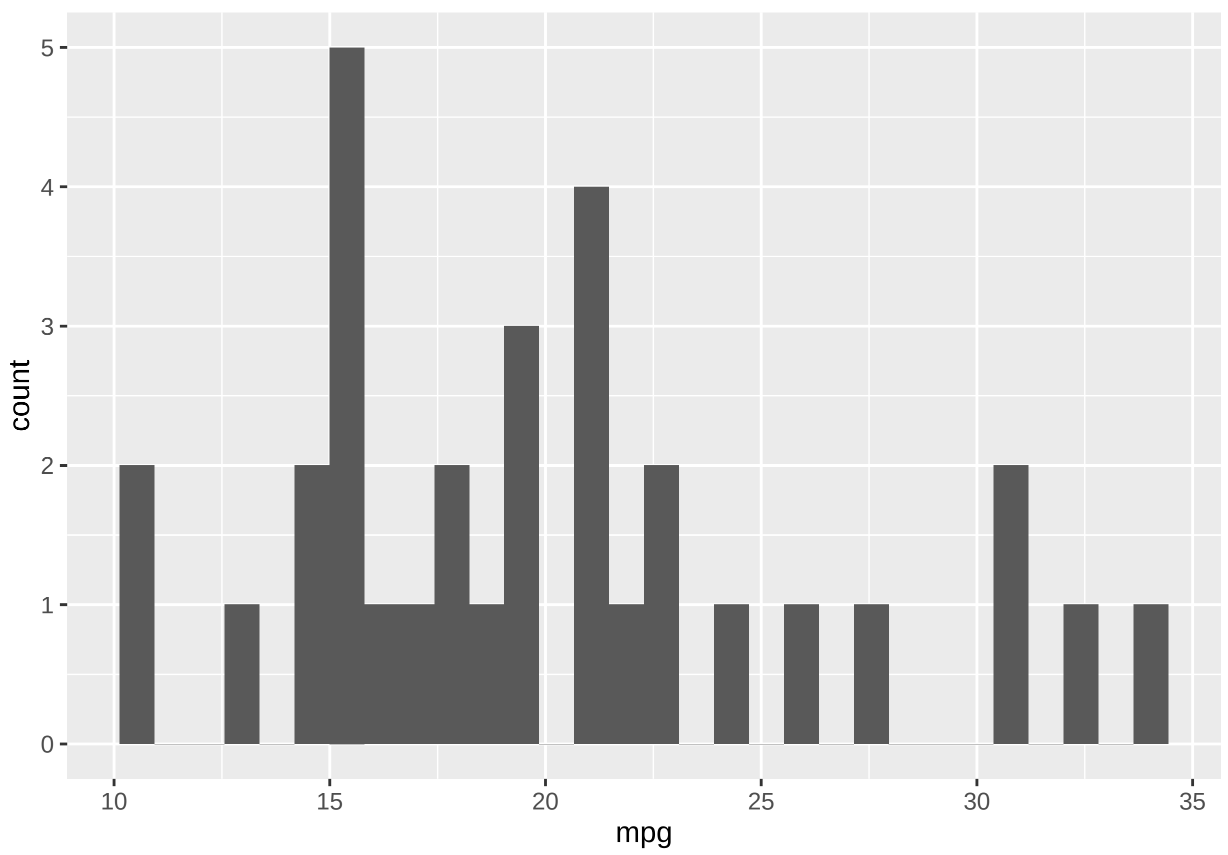
The basic grammar of qplot(quick plot) is simple.Here is the code that tells R to plot a ‘XXX’ figure of variable 1 (and variable 2) of the data frame df.

qplot(data=df, x=var1(, y=var2), geom=‘xxx’)

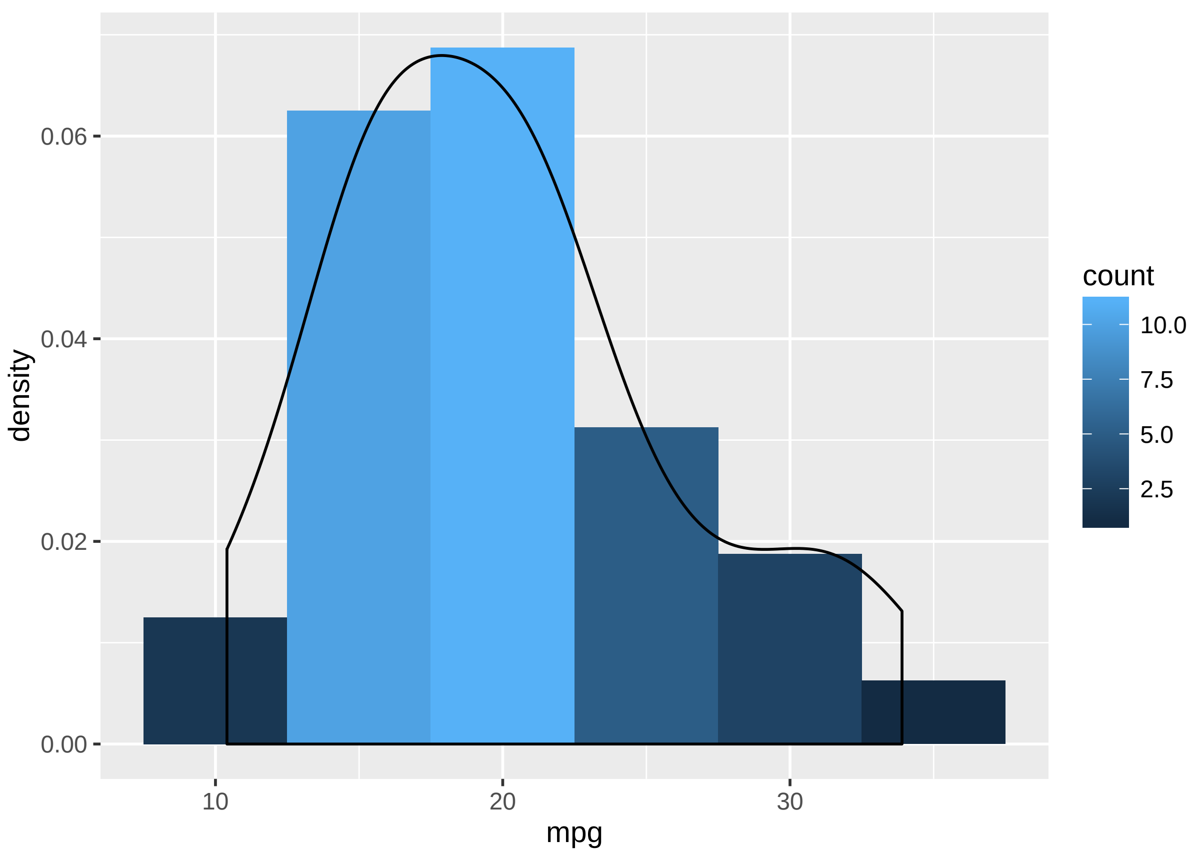
Here is the example with histogram.

```
qplot(data=autodata, x=mpg, geom='histogram')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



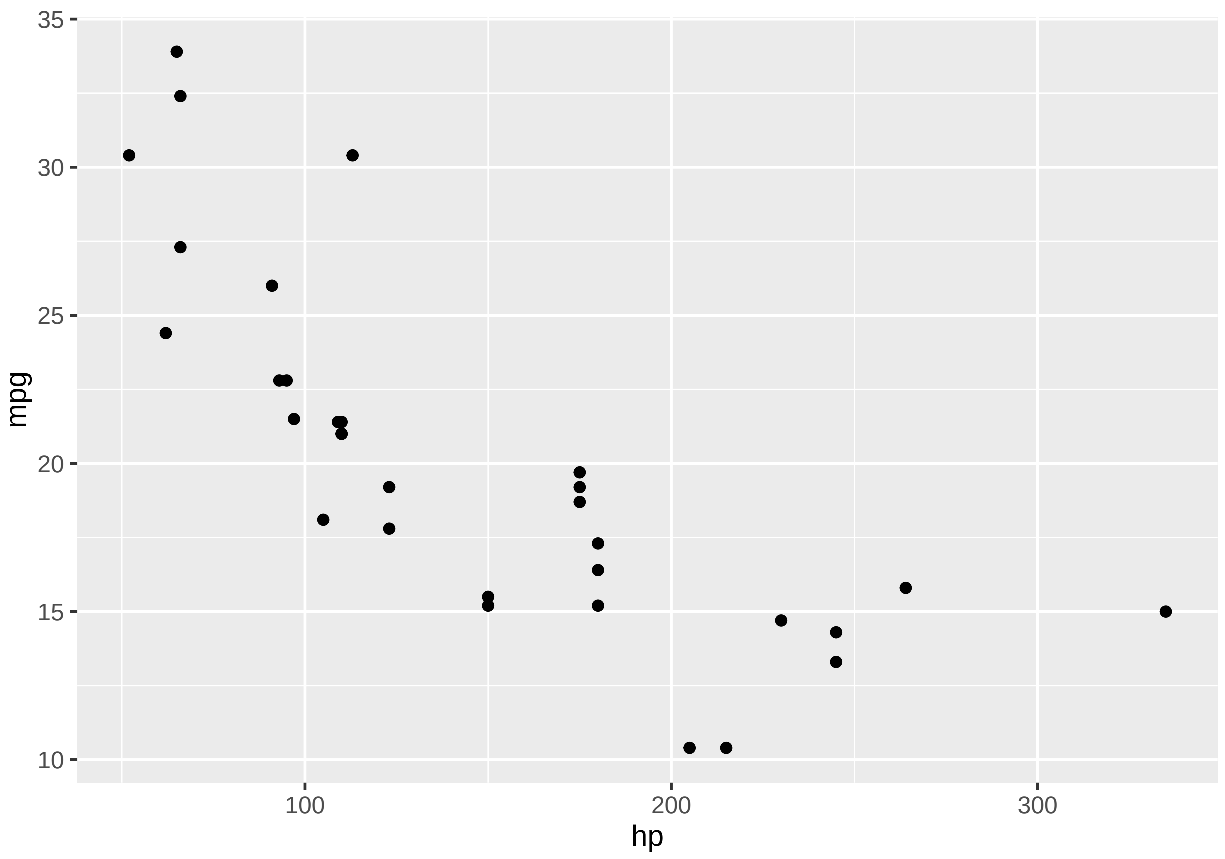
You can make it much fancy by diving deep into the ggplot2.



Scatter plot

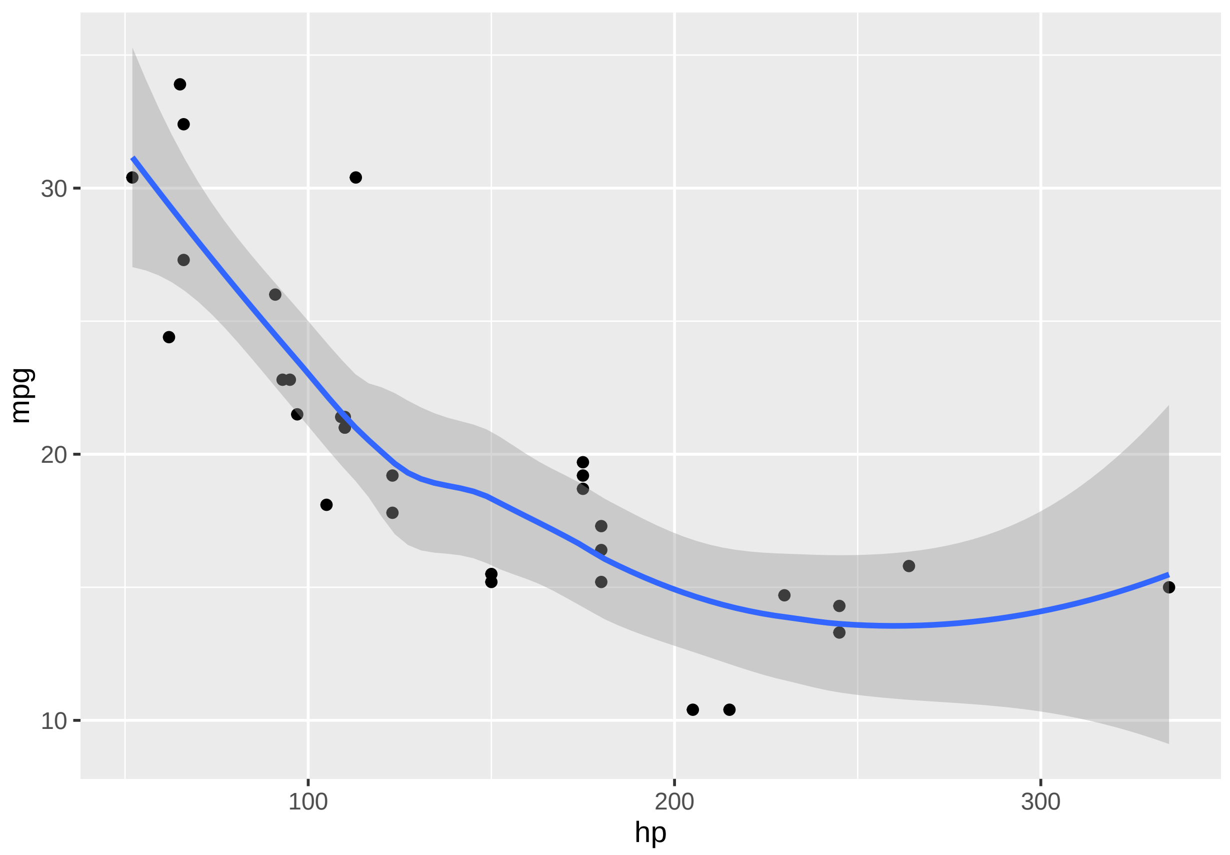
If you do not specify anything, the qplot function gives you the scatter plot.

```
qplot(data=autodata, x=hp, y=mpg)
```



Overlay the scatter plot with a smoothing function to show the trend.

```
qplot(data=autodata, x=hp, y=mpg) + geom_smooth()
```



Hooray, you have the discovery of the century. If you drive a bigger car, you burn more gasoline.

Your boss will be impressed!

Exercise

(1) Recode mumber of Cynlinders to a factor varaible with label 'Four','Six','Eight'.

(2) Crosstab number of cynlinders with transimission type

(3) Plot the historgram of weight

(4) Plot the relationship between weight and miles per galon

(5) **Challenge:** Download dplyr, another David Hadley's gift to humanity. Use the group_by() and summarize() to the average mpg by transimission type.