GRAPHICS AND EXPLORATORY ANALYSIS

Juanjo Medina

2.1 Introduction

A picture is worth a thousand words; when presenting and interpreting data this basic idea also applies. There has been, indeed, a growing shift in data analysis toward more visual approaches to both interpretation and dissemination of numerical analysis. Part of the new data revolution consists in the mixing of ideas from visualisation of statistical analysis and visual design. Indeed data visualisation is one of the most interesting areas of development in the field.

Good graphics not only help researchers to make their data easier to understand by the general public. They are also a useful way for understanding the data ourselves. In many ways it is very often a more intuitive way to understand patterns in our data than trying to look at numerical results presented in a tabular form.

One of the reasons why R is so popular is because it is capable of producing astonishing graphics such as this:

As with other aspects of R, there are a number of core functions that can be used to produced graphics. But it is by mastering user-designed graphic packages that you will get the most of R graphical power. Two popular packages for data visualisation in R are `lattice` and `ggplot2`. We will focus in the latter. It is a very flexible package that tries to implement the grammar of graphics. The `ggplot2` package has excellent online documentation. I also highly recommend the book by Winston Chang as a valuable reference.

There are two different ways of producing plots with `ggplot2`. The quick way (which gives you less control over the graphic) using the `qplot()` function (stands for quick plot) and the slightly brick-by-brick way using

the `ggplot()` function. The `ggplot()` function gives you more control over appearance. With `qplot()` you can produce plots quickly for exploration and then further explore the data or beautify your plot with `ggplot()`.

Unfortunately, we don't have the time to explain in all detail both functions so for most of the session we will use the `ggplot()` function. Although `qplot()` makes it easy to produce basic graphs, it may also delay understanding the philosophy of `ggplot2()`. The `ggplot()` function provides fuller implementation of the grammar of graphics and although it may have a steeper learning curve allows much more flexibility when building graphs.

The grammar of graphics defines various components of the graphic. Some of the most important are:

-**The data**: For using `ggplot2` the data has to be stored as a data frame

-**The geoms**: They describe the objects that represent the data (e.g., points, lines, polygons, etc.).

-**The aesthetics**: They describe the visual characteristics that represent data (e.g., position, size, colour, shape, transparency).

-**Facets**: They describe how data is split into subsets and displayed as multiple small graphs.

-**Stats**: They describe statistical transformations that typically summarise data.

Although this may sound abstract at this point, it will become clearer as we go along.

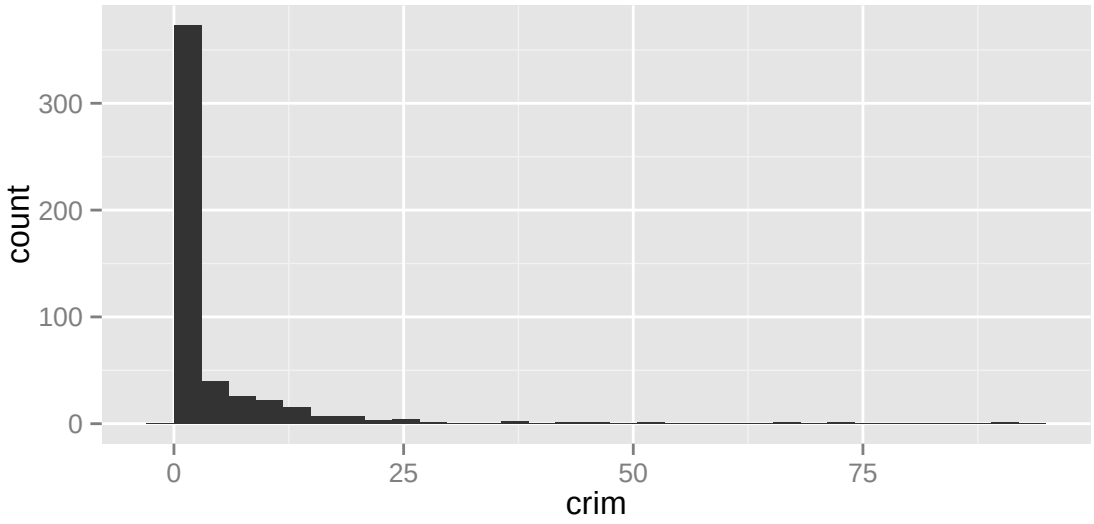
We will start by loading the `Boston` dataset that we introduced last week. If you don't remember how to load this data look at the handout from last week.

2.2 Histograms

Histograms are useful ways of representing quantitative variables visually. If you want a quick and dirty histogram you can use the `hist()` function from base graphics as we did last week. But in this course we are going to rely on `ggplot2`, so instead we will use the functions associated with this package.

To create a histogram with the `qplot` function is as simple as this:

```
library(ggplot2)
data(Boston, package="MASS")
qplot(crim, data = Boston)
```

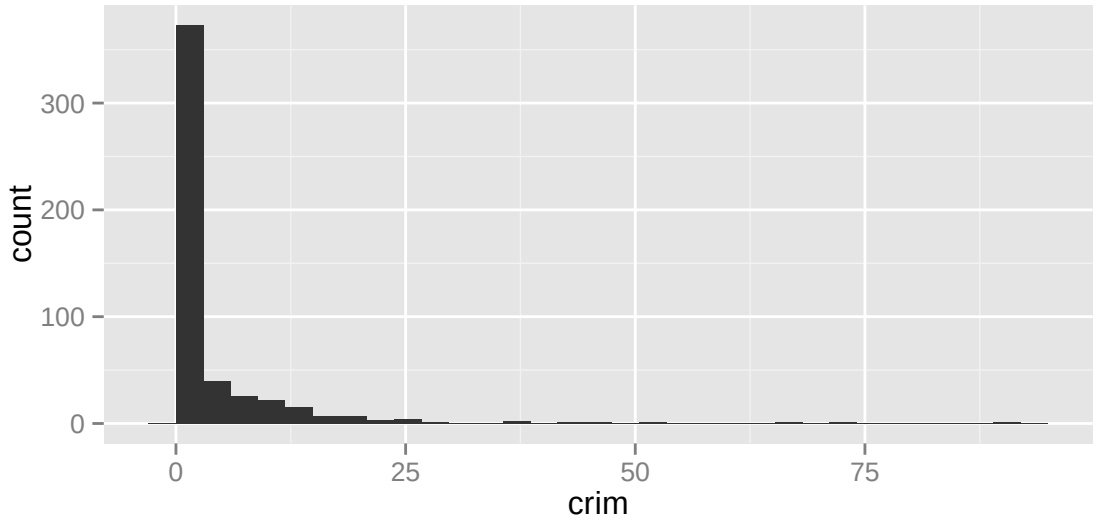


This is the distribution of the per capita crime rate by town in he suburbs of Boston. As you can see this is a highly skewed distribution. You better get used to the look of it. Crime data tend to look like this[1]. You will notice that all we have done is to tell R to use the variable we wanted to plot and the dataset where that variable is located [2].

As mentioned earlier, we will emphasise in this course the use of the `ggplot()` function. With `ggplot()` you start with a blank canvass and keep adding specific layers. The `ggplot()` function can specify the dataset and the aesthetics (the visual characteristics that represent the data). If you want to produce a histogram with the `ggplot` function you would use the following equivalent code:

```
#The "aes" argument defines the aesthetics for the plot, here we are telling R the data
#we will represent comes from the Boston data and our X variable, displayed in the X
#axis, will be Per capita crime rate. So we are specfying we are using the "crim"
#variable to display a position in the X axis.

#If we just run the function in the first line we won't be plotting anything, we have
#to tell R what type of object, or geom, is going to represent the data that we specified
#in the aesthetics. There are multiple geoms we can use. For a histogram we use the
#geom_histogram.

ggplot(Boston, aes(x = crim)) +
  geom_histogram()
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



So you can see that `ggplot` works in a way that you can add a series of additional specifications (layers, annotations). In this simple plot the `ggplot` function simply maps *crim* as the variable to be displayed (as one of the aesthetics) and the dataset. Then you add the `geom_histogram` to tell R that you want this variable to be represented as a histogram. Later we will see what other things you can add.

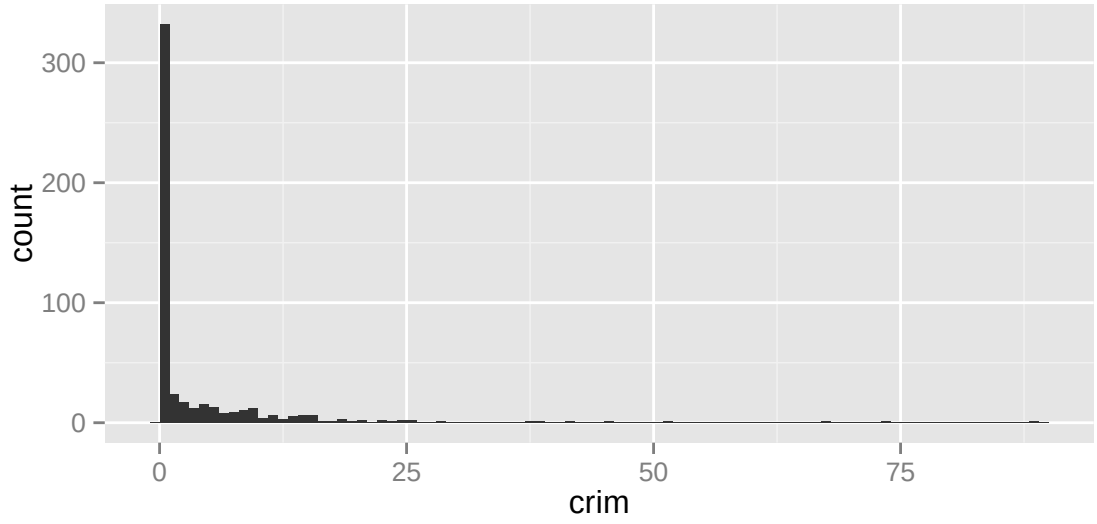
A histogram is simply putting cases in "bins" and then creates a bar for each bin. You can think of it as a visual grouped frequency distribution. The code we have used so far has used a bin-width of size range/30 as

[1] This is an interesting blog entry in solutions when you have highly skewed data.

[2] In the first lecture we introduced the idea that R is an object-oriented language, it is important you understand it is also *polymorphic*; which means that a single function can be applied to different types of inputs, which the function processes in the appropriate way. This will become clearer in a later example when we pass different inputs to the qplot function. In this case, since qplot only sees one quantitative variable it produces the histogram, an appropriate visualisation for this scenario.

R kindly reminded us. But you can modify this parameter if you want to get a rougher or a more granular picture. In fact, you should *always* play around with different specifications of the bin-width until you find one that tells the full story in a parsimonious way.

```
#We can pass arguments to the the geoms, here we are changing the size of the bins
#(for further details on other arguments you can check the help files)

ggplot(Boston, aes(x = crim)) +
  geom_histogram(binwidth = 1)
```



Using bin-width of 1 we are essentially creating a bar for every one unit increase in the percent rate of crime. We can still see that most towns have a very low level of crime.

```
#Let's sum the number of towns with a value lower than 1 in the per capita crime rate

sum(Boston$crim < 1)
```

```
## [1] 332
```

We can see that the large majority of towns, 332 out of 506, have a per capita crime rate below 1%. But we can also see that there are some towns that have a high concentration of crime. This is a spatial feature of crime; it tends to concentrate in particular areas and places. You can see how we can use visualisations to show the data and get a first feeling for how it may be distributed.

When plotting a continuous variable we are interested in the following features:

- **Asymmetry**: whether the distribution is skewed to the right or to the left.
- **Outliers**: Are there one or more values that seem very unlike the others?
- **Multimodality**: How many peaks has the distribution? More than one peak may indicate that the variable is measuring different groups.
- **Impossibilities or other anomalies**: Values that are simply unrealistic given what we are measuring (e.g., somebody with an age of a 1000 years). Sometimes you may come across a distribution of data with a very high (and implausible) frequency count for a particular value. Maybe you measure age and you have a large number of cases aged 99 (which is often a code used for missing data).

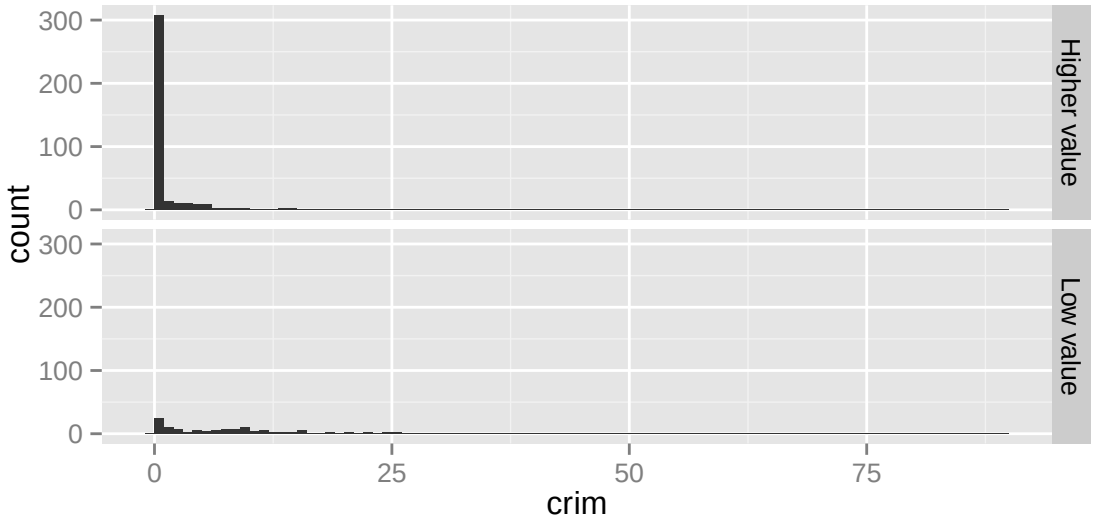
- **Spread**: this gives us an idea of variability in our data.

Often we visualise data because we want to compare distributions. **Most of data analysis is about making comparisons**. We are going to explore whether the distribution of crime is different for less affluent areas. The variable *medv* measures the median value of owner-occupied homes. For the purposes of this illustration I want to dichotomise this variable, to create a group of towns with particularly low values versus all the others. For further details in how to recode variables with R you may want to read the relevant sections in Quick R or the R Cookbook.

```
#How to create a categorical variable based on information from a quantitative variable

#First we tell R to create a new vector ("lowval") in the Boston data frame. This
#vector will be assigned the character value "Low value" when the condition within
#the square brackets is met. That is, we are saying that whenever the value in "medv"
#is below 17.02 then the new variable lowval will equal "Low value"
#I have chosen 17.02 as this is the first quartile for medv

Boston$lowval[Boston$medv <= 17.02] <- "Low value"

#Now we define the other values as "Higher Value"

Boston$lowval[Boston$medv > 17.02] <- "Higher value"

#The variable we created was a character vector, this step transforms it into a factor
#(many functions designed to work with categorical variables expect a factor as an
#input, not just a character vector)

Boston$lowval <- as.factor(Boston$lowval)
```

Now we can produce the plot:

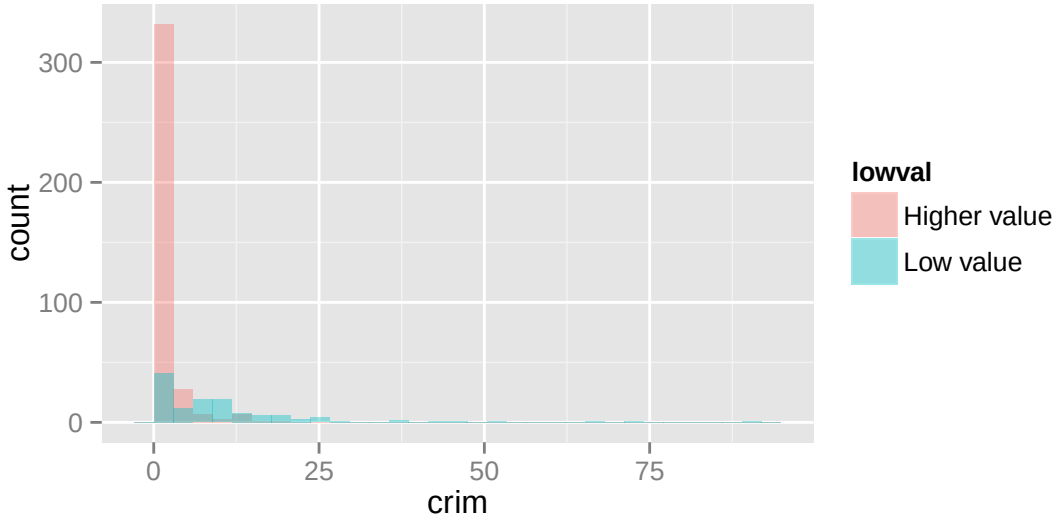
```
#Facets are another element of the grammar of graphics, we use it to define subsets of
#the data to be represented as multiple groups, here we are asking R to produce two
#plots defined by the two levels of the factor we just created.

ggplot(Boston, aes(x = crim)) +
  geom_histogram(binwidth = 1) +
  facet_grid(lowval ~ .)
```



Visually this may not look great, but it begins to tell a story. We can see that there is a considerable lower proportion of towns with low levels of crime in the group of towns that have cheaper homes. It is a flatter, less skewed distribution. You can see how the `facet_grid()` expression is telling R to create the histogram of the variable mentioned in the ggplot function for the groups defined by the categorical input of interest (the factor "lowval").

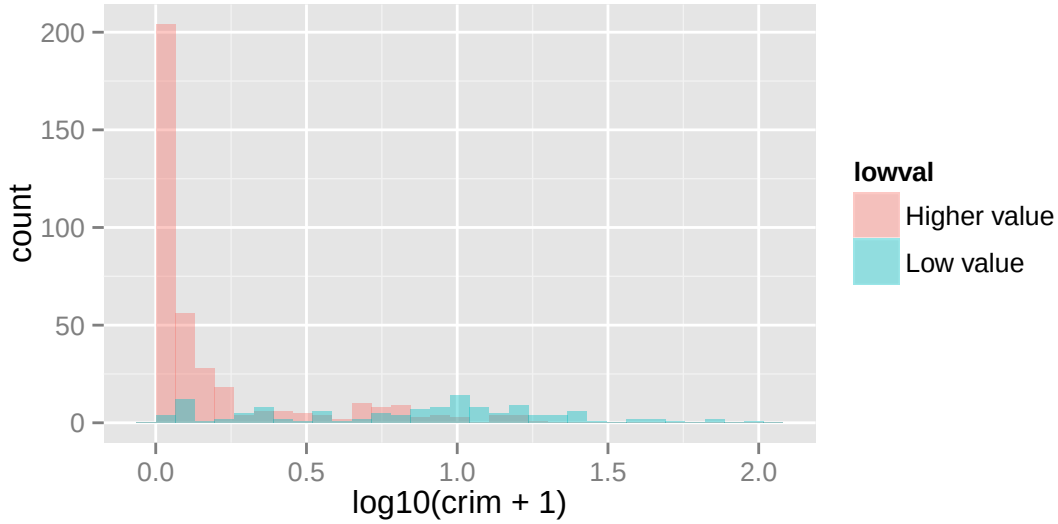
Instead of using facets, we could overlay the histograms with a bit of transparency. Transparencies work better in screens than in printed document, so keep in mind this when deciding whether to use them instead of facets. The code is as follows:

```
#"fill" identifies the factor grouping the cases
#"position = identity"" tells R to overlay the distributions
#and "alpha"" asks for the degree of transparency, a lower value will be more transparent

ggplot(Boston, aes(x = crim, fill = lowval)) +
  geom_histogram(position = "identity", alpha = 0.4)
```



In this case, part of the problem we have is that the skew can make it difficult to appreciate the differences. When you are dealing with skewed distributions such as this, it is sometimes convenient to use a transformation. We will come back to this later this semester. For now suffice to say, that taking the logarithm of a skewed variable helps to reduce the skew and to see patterns more clearly. In order to visualise the differences here a bit better we could ask for the logarithm of the crime per capita rate. Notice how I also add a constant of 1 to the variable *crim*, this is to avoid NA values if the value in *crim* is zero (you cannot take the log of 0).

```
ggplot(Boston, aes(x = log10(crim + 1), fill = lowval)) +
  geom_histogram(position = "identity", alpha = 0.4)
```

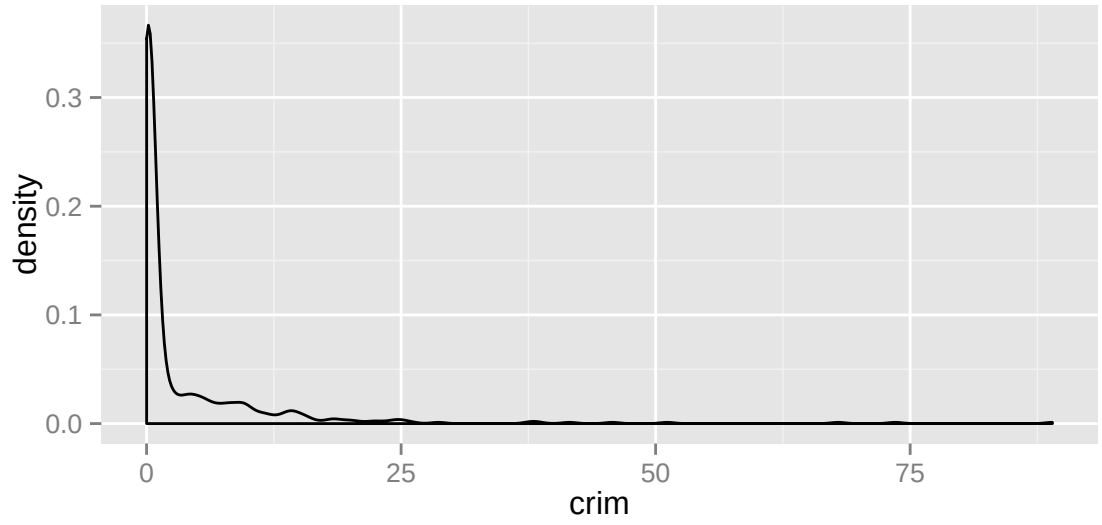


The plot now is a bit clearer. It seems pretty evident that the distribution of crime is slightly different between these two types of towns.

2.3 Density plots

For smoother distributions, you can use density plot. You should have a healthy amount of data to use these or you could end up with a lot of unwanted noise. Let's first look at the single density plot for all cases:

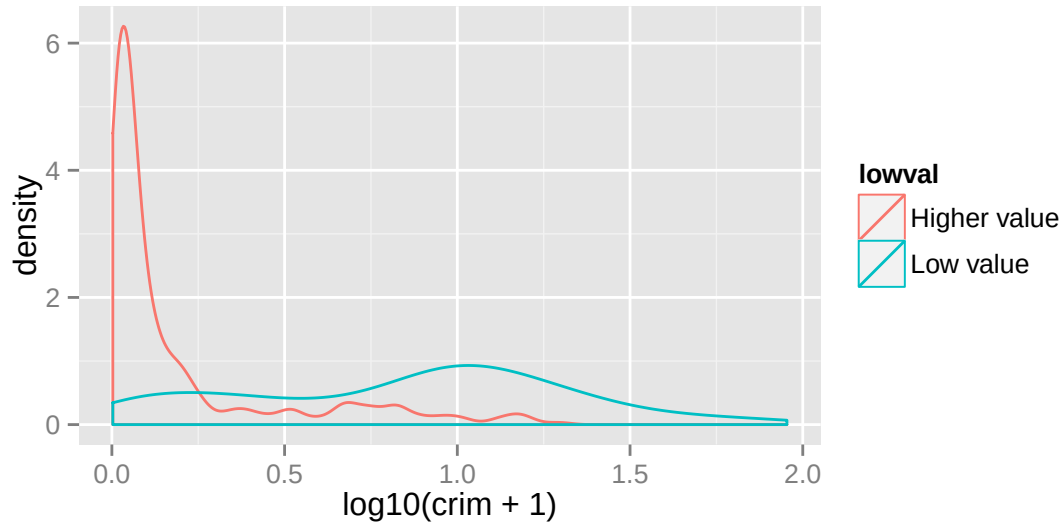
```
ggplot(Boston, aes(x = crim)) +
  geom_density()
```



In a density plot the area under the lines sum to 1 and the Y, vertical, axis now gives you the estimated probability for the values in the X, horizontal, axis. In this plot we can see that there is a high estimated probability of observing a town with near zero per capita crime rate and a low estimated probability of seeing towns with large per capita crime rates. As you can observe it provides a smoother representation of the distribution (as compared to the histograms).

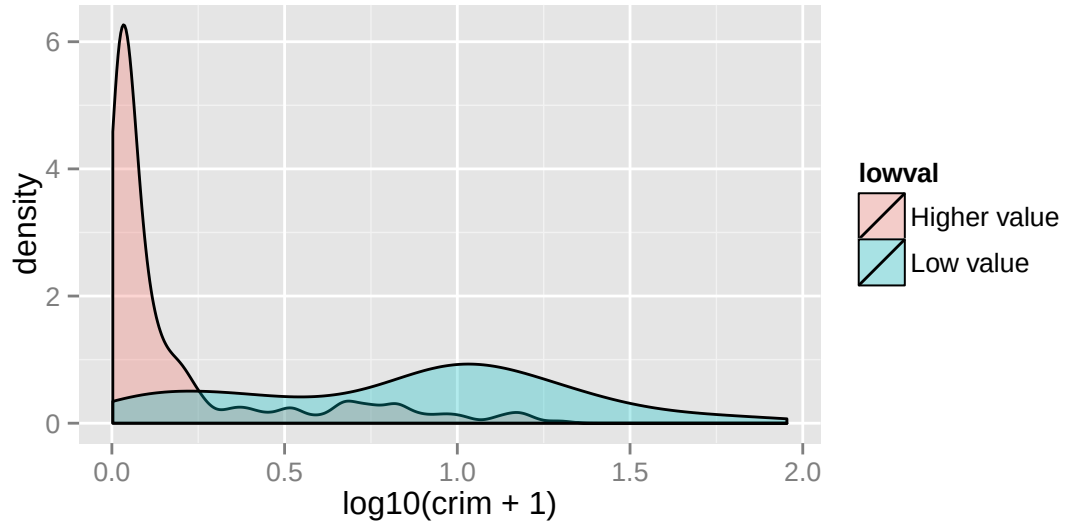
You can also use this to compare the distribution of a quantitative variable across the levels in a categorical variable (factor) and, as before, is possibly better to take the log of skewed variables such as crime:

```
#We are mapping "lowval" as the variable colouring the lines

ggplot(Boston, aes(x = log10(crim + 1), colour = lowval)) +
  geom_density()
```



Or you could use transparencies:

```
ggplot(Boston, aes(x = log10(crim + 1), fill = lowval)) +
  geom_density(alpha = .3)
```



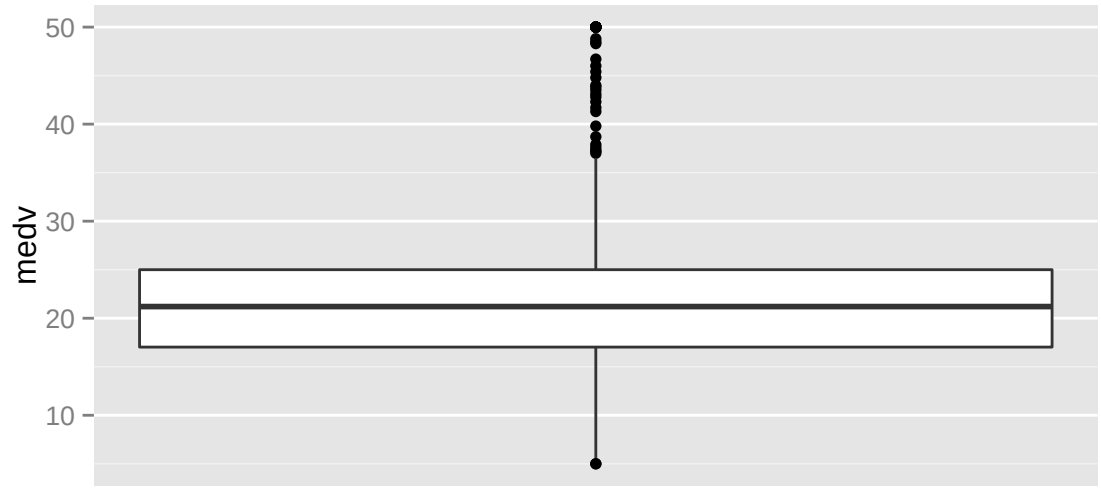
Did you notice the difference with the comparative histograms? By using density plots we are rescaling to ensure the same area for each of the levels in our grouping variable. This makes it easier to compare two groups that have different frequencies. The areas under the curve add up to 1 for both of the groups, whereas in the histogram the area within the bars represent the number of cases in each of the groups. If you have many more cases in one group than the other it may be difficult to make comparisons or to clearly see the distribution for the group with fewer cases. So, this is one of the reasons why you may want to use density plots.

2.4 Box plots

Box plots are an interesting way of presenting the 5 number summary in a visual way. This video may help you to understand them better. If we want to use `ggplot` for this we need some convoluted code, since `ggplot` assumes you want a boxplot to compare various groups. Therefore we need to set some arbitrary value for the grouping variable and we also may want to remove the x-axis tick markers and label.

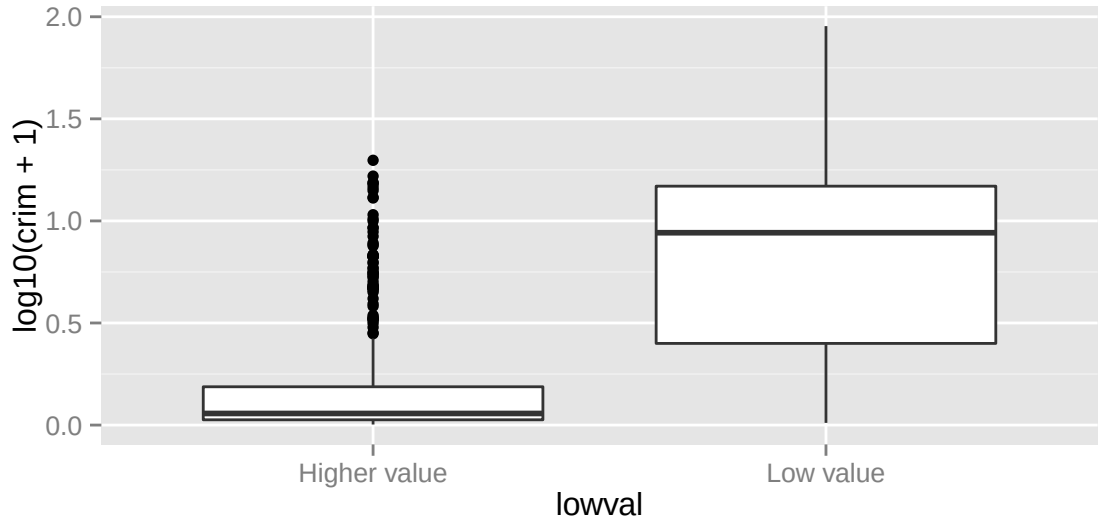
For this illustration, I am going to display the distribution of the median value of property in the various towns instead of crime.

```
ggplot(Boston, aes(x = 1, y = medv)) +
  geom_boxplot() +
  scale_x_continuous(breaks = NULL) + #removes the tick markers from the x axis
  theme(axis.title.x = element_blank())
```



Boxplots, however, really come to life when you do use them to compare the distribution of a quantitative variable across various groups. Let's look at the distribution of log(crime) across cheaper and more expensive areas:

```
ggplot(Boston, aes(x = lowval, y=log10(crim + 1))) +
  geom_boxplot()
```

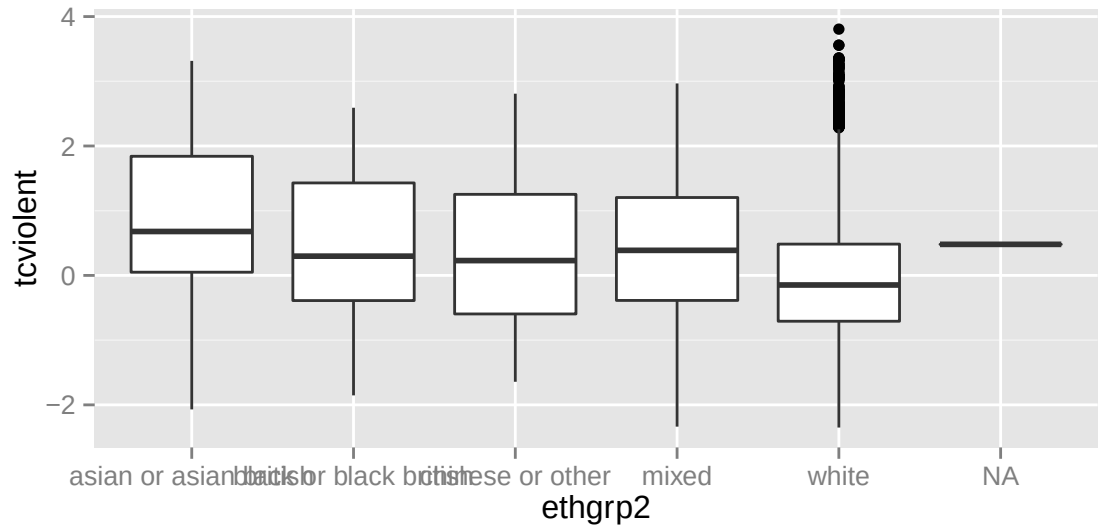


With a boxplot like this you can see straight away that the bulk of cheaper areas are very different from the bulk of more expensive areas. The first quartile of the distribution for low areas about matches the point at which we start to see **outliers** for the more expensive areas.

This can be even more helpful when you have various groups. Let's try an example using the `BCS0708` data frame we introduced last week. If you did not save your project after last week and need to load the data again but don't remember how, look at the notes from last week.

This dataset contains a quantitative variable that measures the level of worry for crime (`tcviolent`): high scores represent high levels of worry. We are going to see how the score in this variable changes according to ethnicity (`ethgrp2`).

```
#A comparative boxplot of ethnicity and worry about violent crime
ggplot(BCS0708, aes(x = ethgrp2, y = tcviolent)) +
  geom_boxplot()
```



Nice. But could be nicer. To start with we could order the groups along the X axis so that the ethnic groups are positioned according to their level of worry. Secondly, we may want to exclude the information for the NA cases on ethnicity (represented by a flat line).

```
#A nicer comparative boxplot (excluding NA and reordering the X variable)
ggplot(subset(BCS0708, !is.na(ethgrp2) & !is.na(tcviolent)),
       aes(x = reorder(ethgrp2, tcviolent, FUN = median), y = tcviolent)) +
        geom_boxplot()
```

The `subset` function is using a logical argument to tell R to only use the cases that do not have NA values in the two variables that we are using. The `reorder` function on the hand is asking R to reorder the levels in ethnicity according to the median value of worry of violent crime. Since we are using those functions

within the `ggplot` function this subsetting and this reordering are not introducing permanent changes in your original dataset. If you prefer to reorder according to the mean you only need to change that parameter after the `FUN` option (e.g, `FUN = mean`).

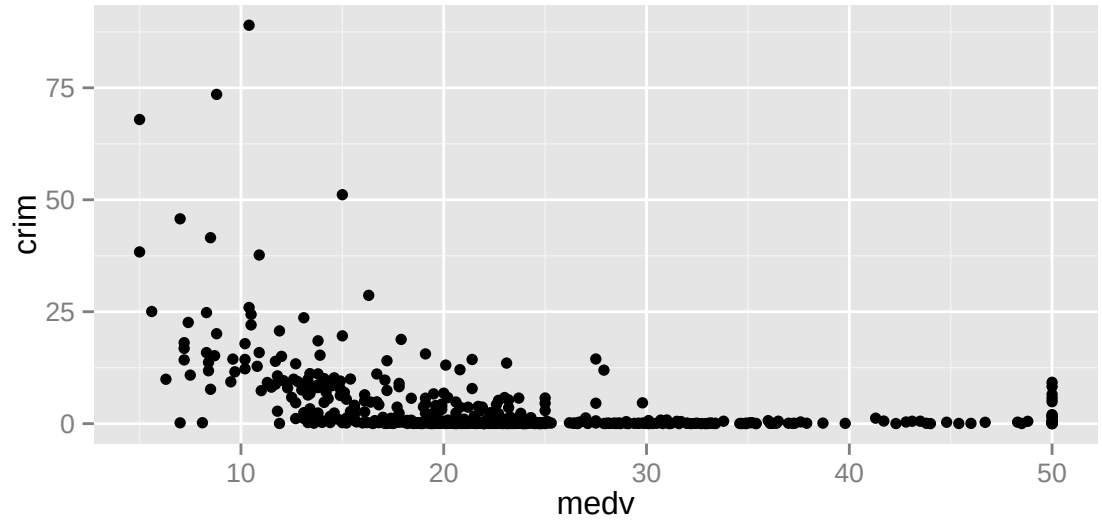
2.5 Scatter plots with two variables

When looking at the relationship between two quantitative variables nothing beats the **scatterplot**. This is a lovely article in the history of the scatterplot and this video may help you to understand them better.

A scatterplot plots one variable in the Y axis, and another in the X axis. Typically, if you have a clear outcome or response variable in mind, you place it in the Y axis, and you place the explanatory variable in the X axis.

This is how you produce a scatterplot with `ggplot()`:

```
#A scatterplot of crime versus median value of the properties
ggplot(Boston, aes(x = medv, y = crim)) +
  geom_point()
```



Each point represents a case in our dataset and the coordinates attached to it in this two dimensional plane are given by their value in the Y (crime) and X (median value of the properties) variables.

What do you look for in a scatterplot? You want to assess global and local patterns, as well as deviations. We can see clearly that at low levels of `medv` there is a higher probability in this data that the level of crime is higher. Once the median value of the property hits $30,000 the level of crime is nearly zero for all towns. So far so good, and surely predictable. The first reason why we look at scatterplots is to check our hypothesis (e.g., poorer areas, more crime).

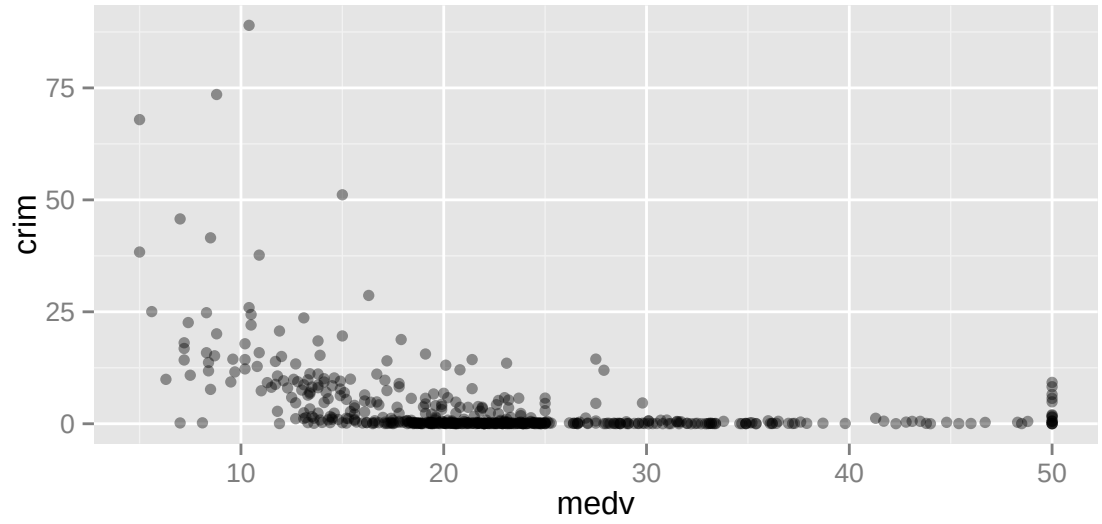
However, something odd seems to be going on when the median property value is around $50,000. All of the sudden the variability for crime goes up. We seem to have some of the more expensive areas also exhibiting some decent level of crime. What's going on? To be honest I have no clue (although I have some hypothesis)! This is my first look at this dataset. But the pattern at the higher level of property value is indeed odd.

This is the second reason why you want to plot your data before you do anything else. It helps you to detect apparent anomalies. I say this is an anomaly because the break in pattern is quite noticeable. It is hard to think of a natural process that would generate this sudden radical increase in crime once the median property value reaches the $50k mark. If you were analysing this for real, you would want to know what's driving this

pattern (e.g., find out about the original data collection, the codebook, etc.): perhaps the maximum median value was capped at $50K and we are seeing this as a dramatic increase when the picture is more complex? For now we are going to let this rest.

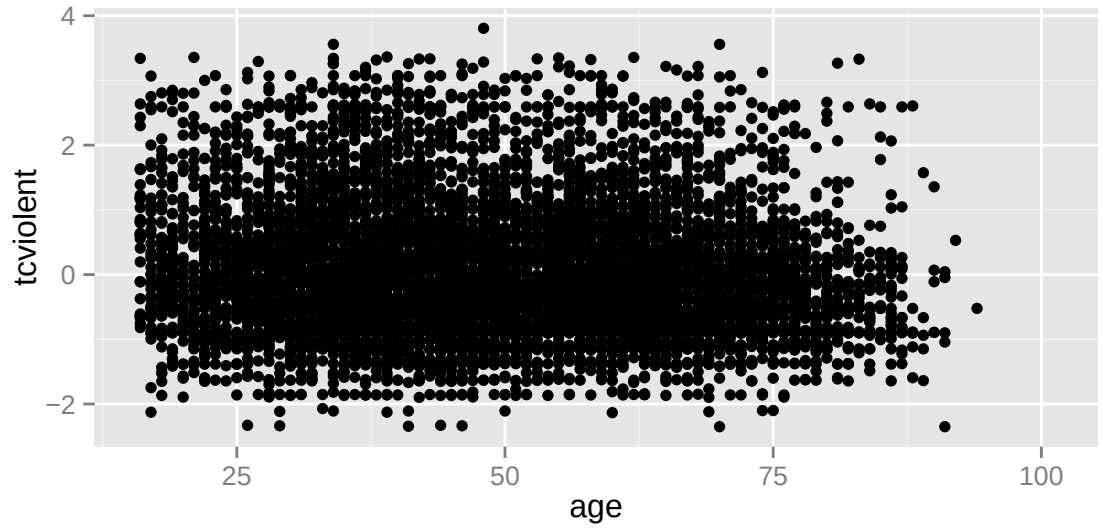
One of the things you may notice with a scatterplot is that even with a smallish dataset such as this, with just about 500 cases, **overplotting** may be a problem. When you have many cases with similar (or even worse the same) value, it is difficult to tell them apart. There's a variety of ways of dealing with overplotting. One possibility is to add some **transparency** to the points:

```
ggplot(Boston, aes(x = medv, y = crim)) +
  geom_point(alpha=.4) #you will have to test different values for alpha
```

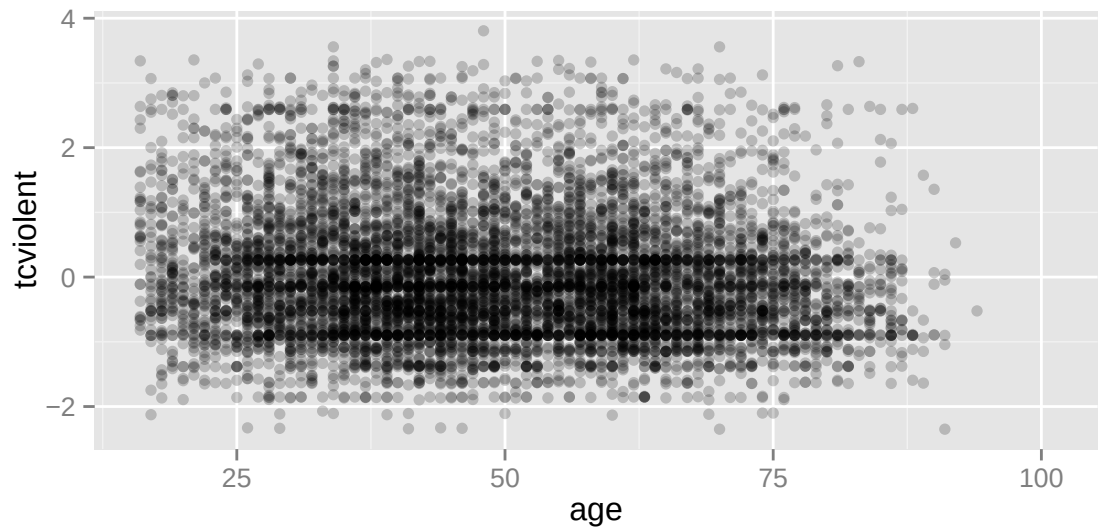


Why this is an issue may be more evident with the `BCS0708 data`. Compare the two plots:

```
ggplot(BCS0708, aes(x = age, y = tcviolent)) +
  geom_point()
```



```
ggplot(BCS0708, aes(x = age, y = tcviolent)) +
  geom_point(alpha=.2)
```

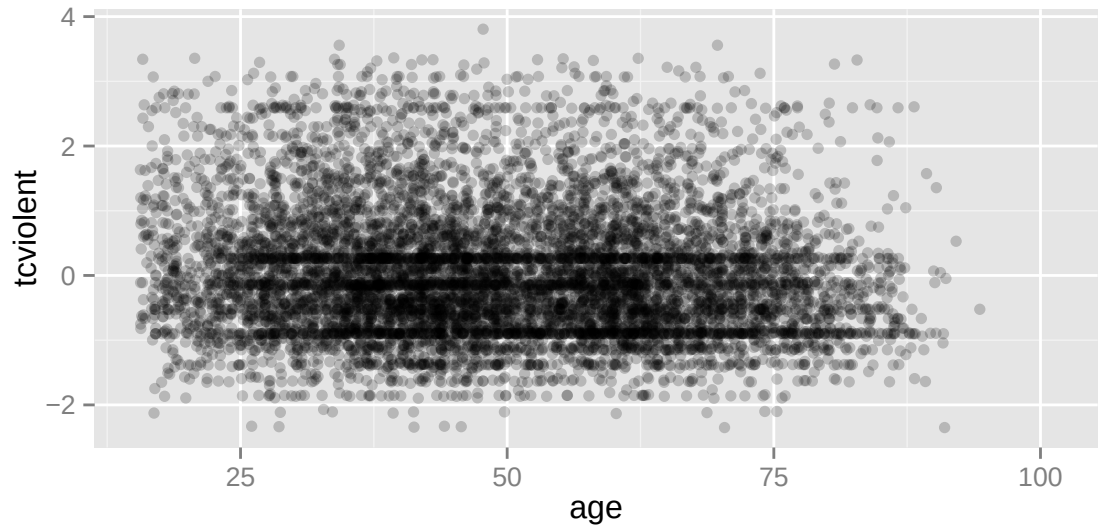


The second plot gives us a better idea of where the observations seem to concentrate in a way that we could not see with the first.

Overplotting can occur when a continuous measurement is rounded to some convenient unit. This has the effect of changing a continuous variable into a discrete ordinal variable. For example, age is measured in years and body weight is measured in pounds or kilograms. Age is a discrete variable, it only takes integer values. That's why you see the points lined up in parallel vertical lines. This also contributes to the overplotting in this case.

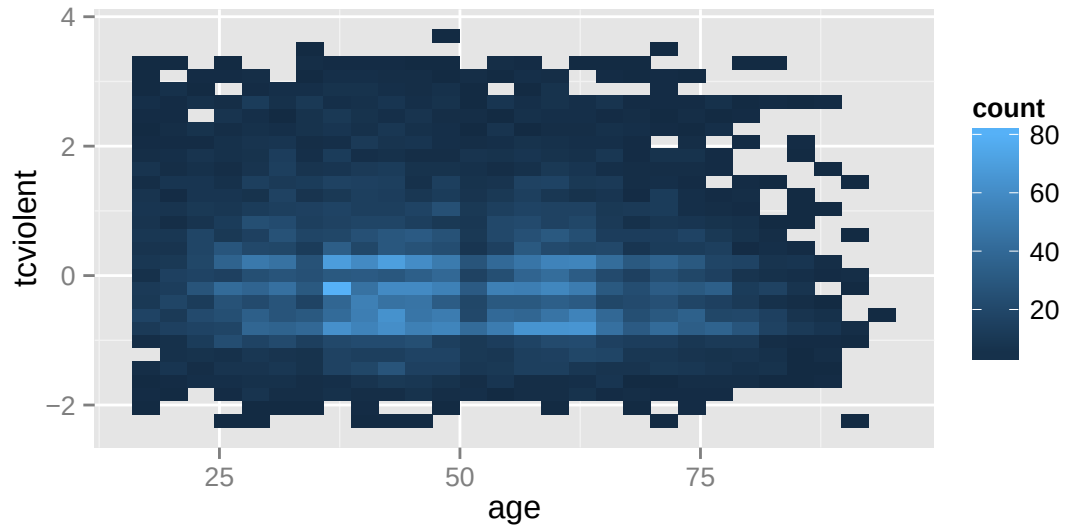
One way of dealing with this particular problem is by **jittering**. Jittering is the act of adding random noise to data in order to prevent overplotting in statistical graphs. In `ggplot` one way of doing this is by passing an argument to `geom_point` specifying you want to jitter the points. This will introduce some random noise so that age looks less discrete.

```
#Alternatively you could replace geom_point() with geom_jitter() in
#which case you don't need to specify the position
ggplot(BCS0708, aes(x = age, y = tcviolent)) +
  geom_point(alpha=.2, position="jitter")
```

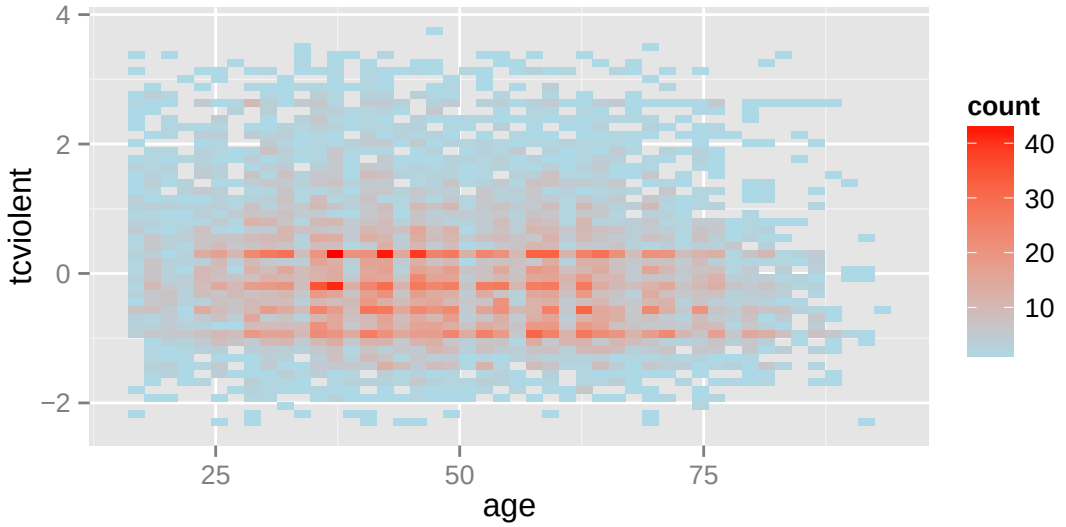


Another alternative for solving overplotting is to **bin the data** into rectangles and map the density of the points to the fill of the colour of the rectangles.

```
ggplot(BCS0708, aes(x = age, y = tcviolent)) +
  stat_bin2d()
```



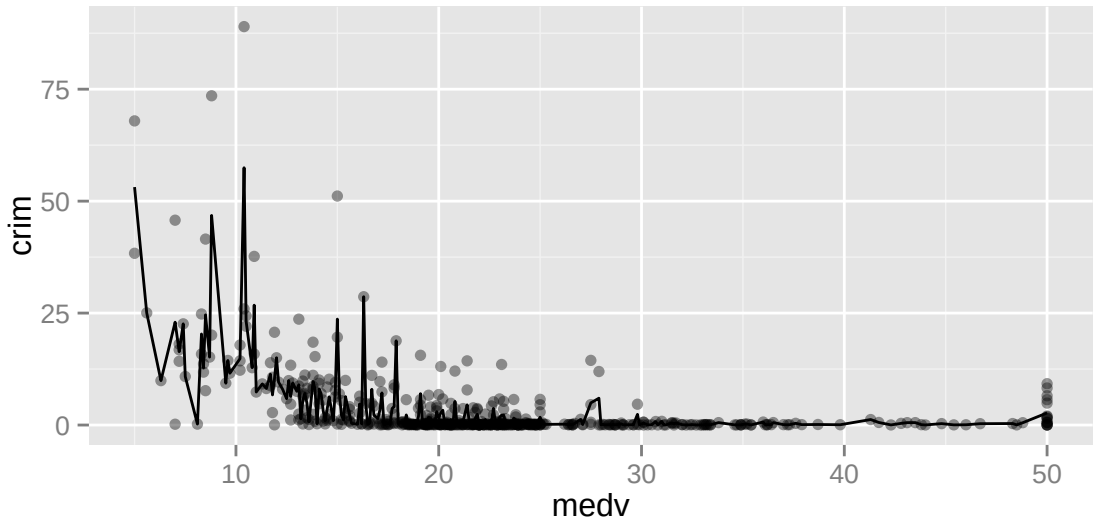

```
#The same but with nicer graphical parameters
ggplot(BCS0708, aes(x = age, y = tcviolent)) +
  stat_bin2d(bins=50) + #by increasing the number of bins we get more granularity
  scale_fill_gradient(low = "lightblue", high = "red") #change colors
```



What this is doing is creating boxes within the two dimensional plane; counting the number of points within those boxes; and attaching a colour to the box in function of the density of points within each of the rectangles.

When looking at scatterplots, sometimes it is useful to summarise the relationships by mean of drawing lines. You could for example add a line representing the **conditional mean**. A conditional mean is simply mean of your Y variable for each value of X. Let's go back to the Boston dataset. We can ask R to plot a line connecting these means using `geom_line()` and specifying you want the conditional means.

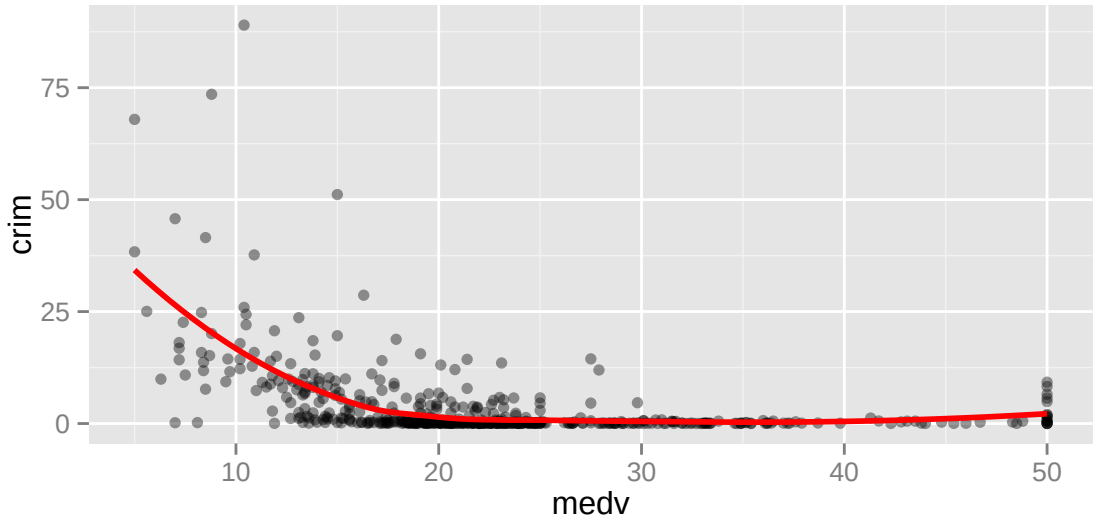
```
ggplot(Boston, aes(x = medv, y = crim)) +
  geom_point(alpha=.4) +
  geom_line(stat='summary', fun.y=mean)
```



With only about 500 cases there are loads of ups and downs. If you have many more cases for each level of X the line would look less rough. You can, in any case, produce a smoother line using `geom_smooth` instead. We will discuss later this semester how this line is computed (although you will see the R output tells you, you are using something call the "loess" method). For now just know that is a line that tries to *estimate* the typical value for Y for each value of X.

```
#We'll explain later this semester what the se argument does, colour is simply asking
#for a red line instead of blue (which I personally find harder to see. I'm also
#making the line a bit thicker with size 1)

ggplot(Boston, aes(x = medv, y = crim)) +
  geom_point(alpha=.4) +
  geom_smooth(colour="red", size=1, se=FALSE)
```

```
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to change t
```



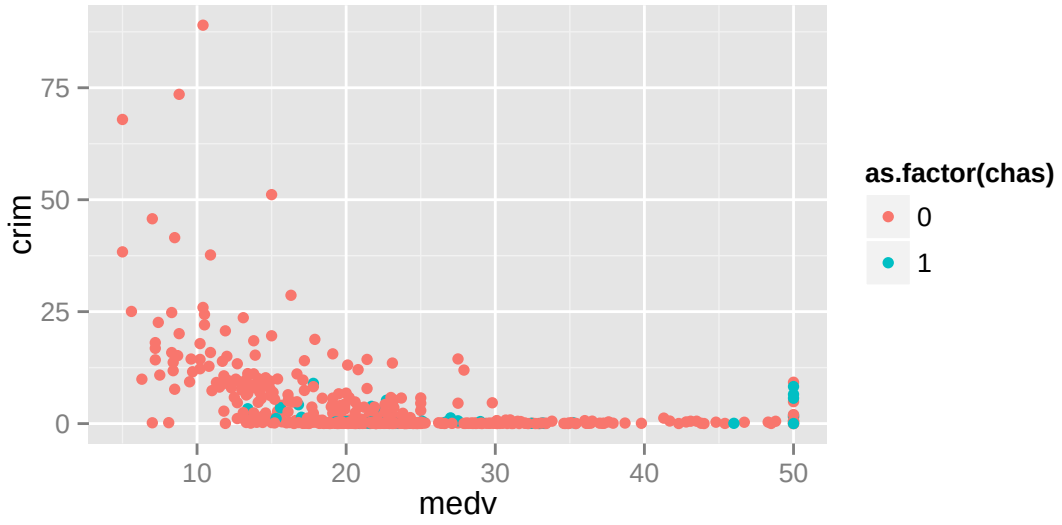
As you can see here you produce a smoother line than with the conditional means. The line, as the scatterplot, seems to be suggesting an overall curvilinear relationship that almost flattens out once property values hit $20k.

2.6 Scatter plots conditioning in a third variable

There are various ways to plot a third variable in a scatterplot. You could go 3D and in some contexts that may be appropriate. But more often than not it is preferable to use only a two dimensional plot.

If you have a grouping variable you could map it to the colour of the points as one of the aesthetics arguments. Here we return to the Boston scatterplot but will add a third variable, that indicates whether the town is located by the river or not.

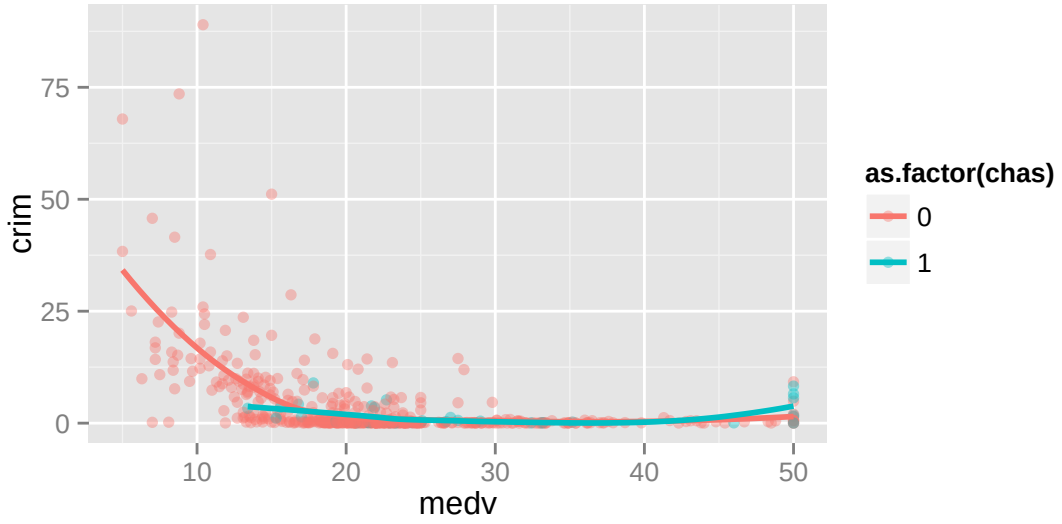
```
#Scatterplot with two quantitative variables and a grouping variable, we are telling R
#to treat "chas", a numeric vector, as a factor.
ggplot(Boston, aes(x = medv, y = crim, colour = as.factor(chas))) +
  geom_point()
```



Curiously, we can see that there's quite a few of those expensive areas with high levels of crime that seem to be located by the river.

As before you can add smooth lines to capture the relationship. What happens now, though, is that `ggplot` will produce a line for each of the levels in the categorical variable grouping the cases:

```
ggplot(Boston, aes(x = medv, y = crim, colour = as.factor(chas))) +
  geom_point(alpha=.4) + #I am doing the points semi-transparent to see the lines better
  geom_smooth(se=FALSE, size=1) #I am doing the lines thicker to see them better
```

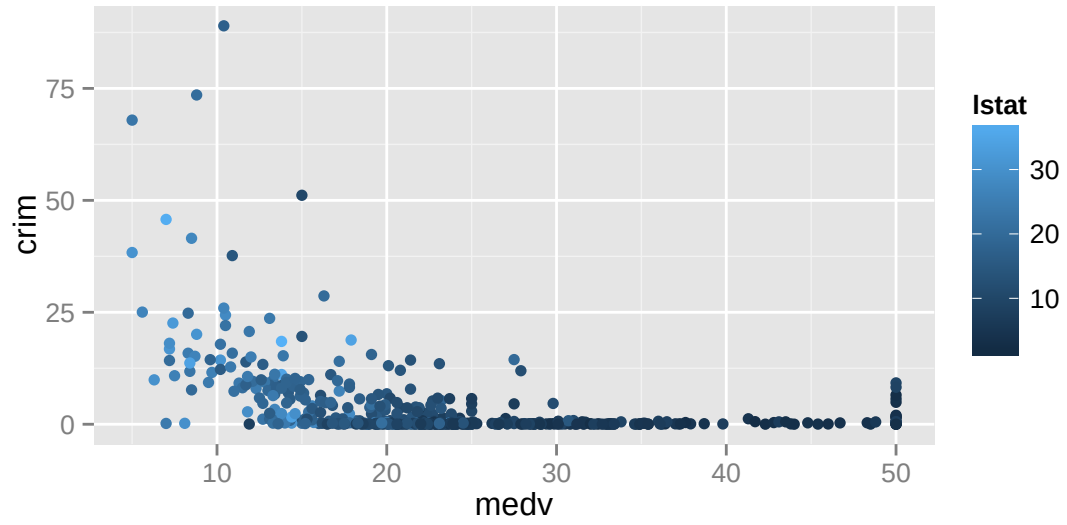


You can see how the relationship between crime and property values is more marked for areas not bordering the river, mostly because you have considerably fewer cheaper areas bordering the river. Notice as well the upward trend in the green line at high values of `medv`. As we saw there seems to be quite a few of those particularly more expensive areas that have high crime and seem to be by the river.

We can also map a quantitative variable to the colour aesthetic. When we do that, instead of different colours for each category we have a gradation in colour from darker to lighter depending on the value of the

quantitative variable. Below we display the relationship between crime and property values conditioning on the status of the area (high values in `lstat` represent lower status).

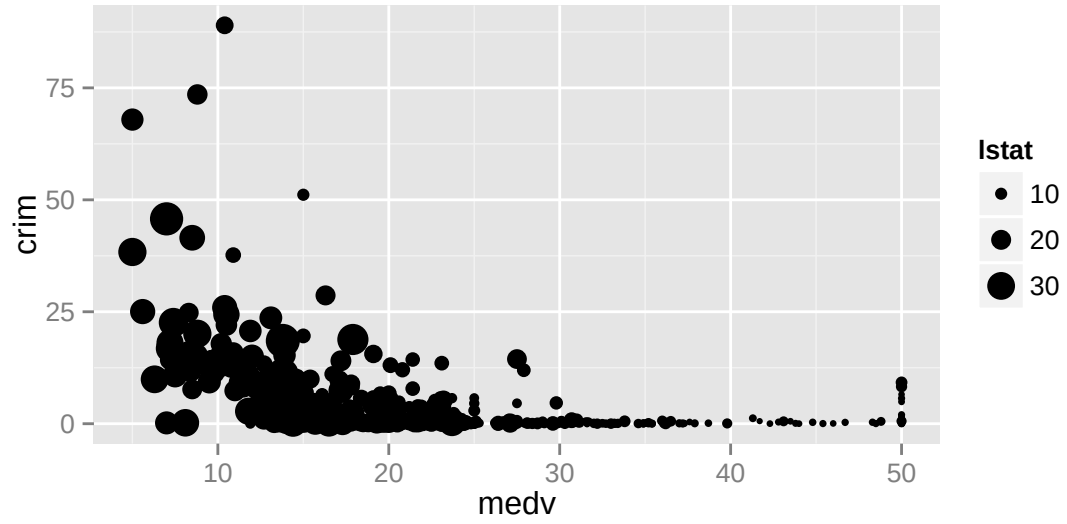
```
ggplot(Boston, aes(x = medv, y = crim, colour = lstat)) +
  geom_point()
```



As one could predict `lstat` and `medv` seem to be correlated. The areas with low status tend to be the areas with cheaper properties (and more crime) and the areas with higher status tend to be the areas with more expensive properties (and less crime).

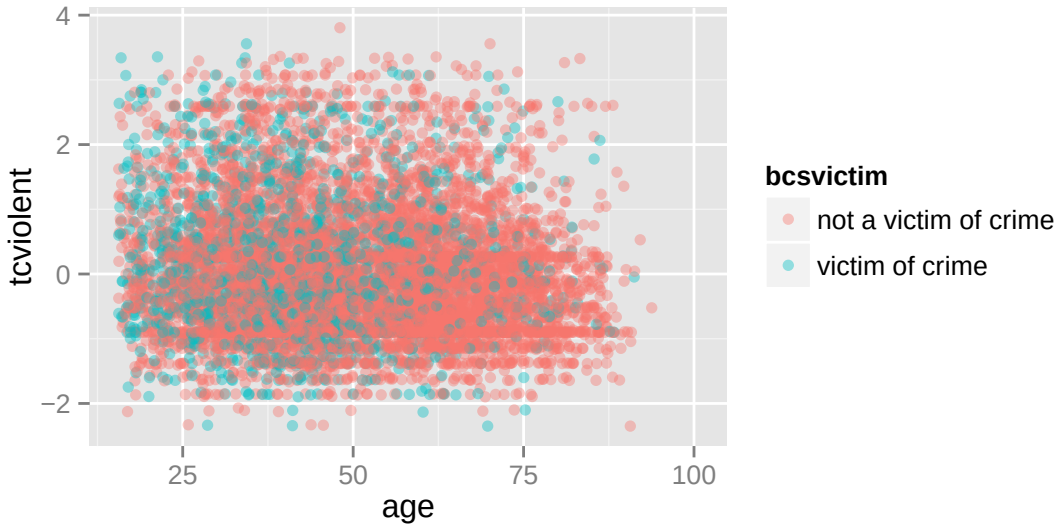
You could map the third variable to a different aesthetic (rather than colour). For example, you could map `lstat` to size of the points. This is called a **bubblechart**. The problem with this, however, is that it can make overplotting more acute sometimes.

```
ggplot(Boston, aes(x = medv, y = crim, size = lstat)) +
  geom_point() #You may want to add alpha for some transparency here.
```



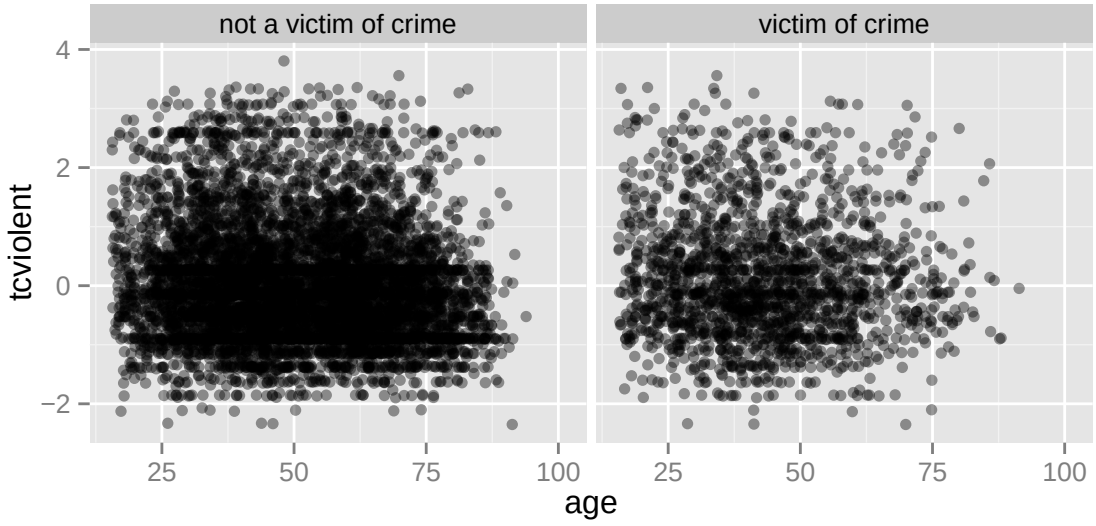
If you have larger samples and the patterns are not clear (as we saw when looking at the relationship between age and worry of violent crime) conditioning in a third variable can produce hard to read scatterplots (even if you use transparencies and jittering). Let's look at the relationship between worry for violent crime and age conditioned on victimisation during the previous year:

```
ggplot(BCS0708, aes(x = age, y = tcviolent, colour = bcsvictim)) +
  geom_point(alpha=.4, position="jitter")
```



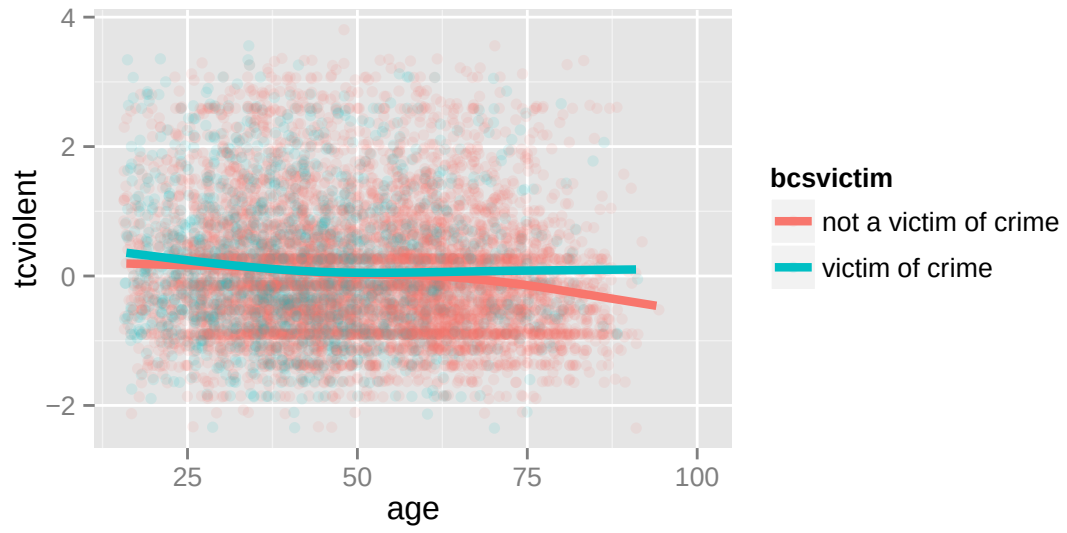
You can possibly notice that there are more green points in the left hand side (since victimisation tend to be more common among youth). But it is hard to read the relationship with age. We could try to use facets instead using `facet_grid`?

```
ggplot(BCS0708, aes(x = age, y = tcviolent)) +
  geom_point(alpha=.4, position="jitter") +
  facet_grid( .~ bcsvictim)
```



It is still hard to see anything, though perhaps you can notice the lower density the points in the bottom right corner in the facet displaying victims of crime. In a case like this may be helpful to draw a smooth line

```
ggplot(BCS0708, aes(x = age, y = tcviolent, colour = bcsvictim)) +
  geom_point(alpha=.1, position="jitter") +
  geom_smooth(size=1.5, se=FALSE)
```



What we see here is that for the most part the relationship of age and worry for violent crime looks quite flat, regardless of whether you have been a victim of crime or not. At least, for most people. However, once we get to the 60s things seem to change a bit. Those over 62 that have not been a victim of crime in the past year start to manifest a lower concern with crime as they age (in comparison with those that have been a victim of crime).

2.7 Scatterplot matrix

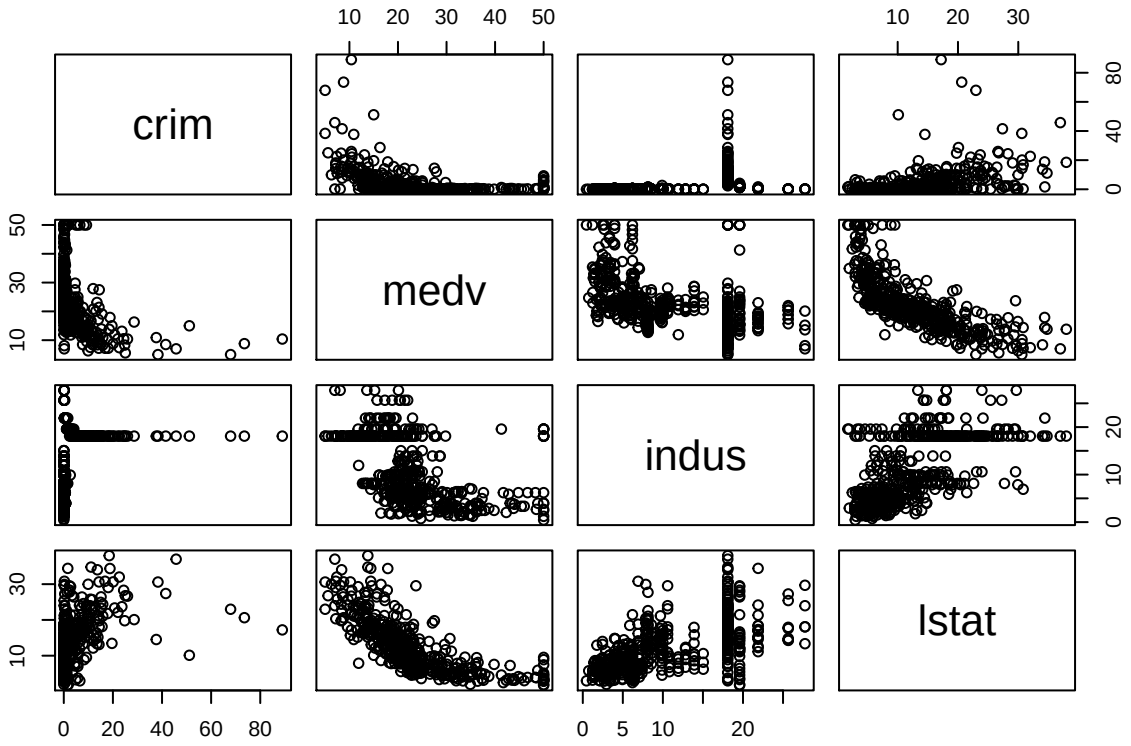
Sometimes you want to produce many scatterplots simultaneously to have a first peak at the relationship between the various variables in your data frame. The way to do this is by using a scatterplot matrix. Unfortunately, `ggplot2` doesn't do scatterplot matrix too well. For this it is better to use the graphics tools of base R.

Not to overcomplicate things we will only use a few variables from the `Boston` dataset:

```
#I create a new data frame that only contains 4 variables included in the Boston dataset
#and I am calling this new data frame object BostonSM
Boston_SM <- subset(Boston, select = c(crim, medv, indus, lstat))
```

Then we run the scatterplot matrix using the **pairs()** function:

```
pairs(Boston_SM)
```



We can modify this code to put additional information in the matrix. As you may have noticed the matrix is symmetrical, for every scatterplot in the upper left side there is a similar on the the bottom right side. We are going to modify this matrix so that the diagonal boxes not only display the name of the variable but also a histogram for the variables and to display the correlation coefficients in the upper left side (instead of printing twice the same scatterplot). We will also make the points smaller and will add a smooth line within each scatterplot.

The code for this is a bit complicated. We are going to introduce **customised functions** for the panels (I took them from Winston Chang book). First we are going to create these two customised functions. One to produce the correlations between each pair of variables and another for producing the histograms for each of the variables.

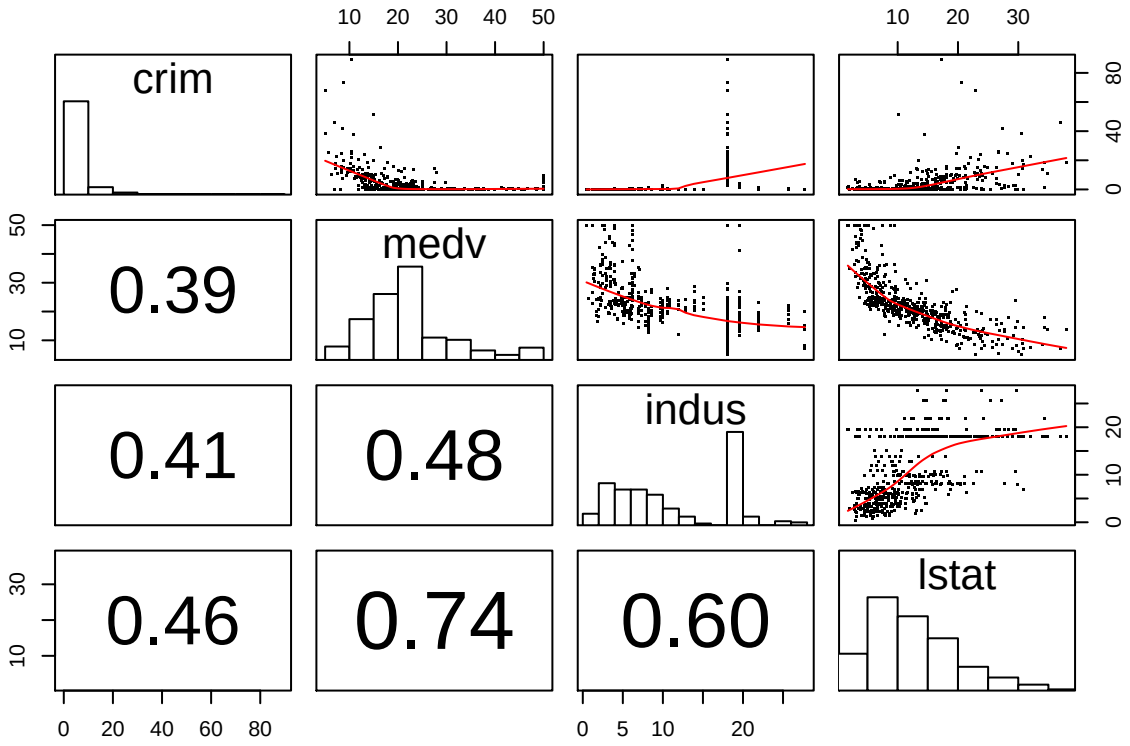
After that we will use the `pairs()` function inserting the inputs that result form using these customised functions. You are likely not to understand the code for the first part. Particularly because apart from doing computations is controlling the aspect of the results. Don't worry, I'm not expecting to understand this code. Just cut and paste. But I think it is important you understand that apart from using the functions created by others and included in packages, once you become proficient in R you can start producing your own functions to expand the range of what you can do with R.

```
panel.cor <- function(x, y, digits=2, prefix="", cex.cor, ...){
  usr <- par("usr")
  on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y, use="complete.obs"))
  txt <- format(c(r, 0.123456789), digits=digits)[1]
  txt <- paste(prefix, txt, sep="")
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
```

```
  text(0.5, 0.5, txt, cex = cex.cor * (1+r)/2)
}
panel.hist <-function(x, ...){
  usr <- par("usr")
  on.exit(par(usr))
  par(usr = c(usr[1:2], 0, 1.5))
  h <- hist(x, plot = FALSE)
  breaks <- h$breaks
  nB <- length(breaks)
  y <- h$counts
  y <- y/max(y)
  rect(breaks[-nB], 0, breaks[-1], y, col="white", ...)
}
```

We can now run the `pairs()` function using our customised functions as inputs.

```
pairs(Boston_SM, pch=".", #Produces smaller points to make it easier to see
      upper.panel=panel.smooth, #Adds for smoothed lines to be added
      lower.panel=panel.cor, #Ask for correlation coeffients in the upper panel
      diag.panel=panel.hist) #Ask for histograms in the diagonal
```

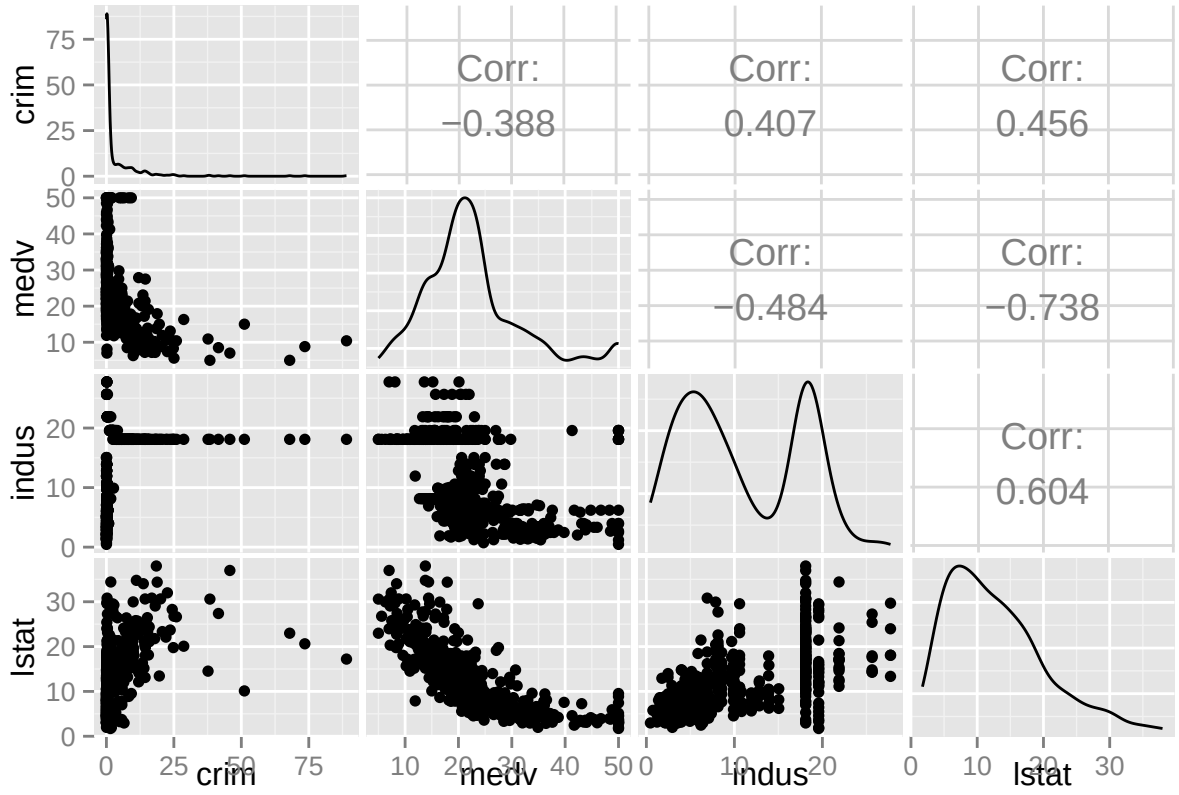


We will explain what correlation coefficients are and when they are appropriate later this semester. For now just focus in trying to understand what the plots show. What do you think may be going on with the distribution of "indus" (proportion of non-retail business acres per town)?

One of the beauties of `ggplot2` is that its functionality has been extended by a number of other packages. One of them is the `GGally` package that allows us to produce output similar to the one we just produced in

a much quicker and straightforward way and extend the correlation matrix to situations in which some of your variables are not numeric. You can easily produced **generalised pair plots** for these scenarios using this package.

```
library(GGally)
ggpairs(Boston_SM)
```

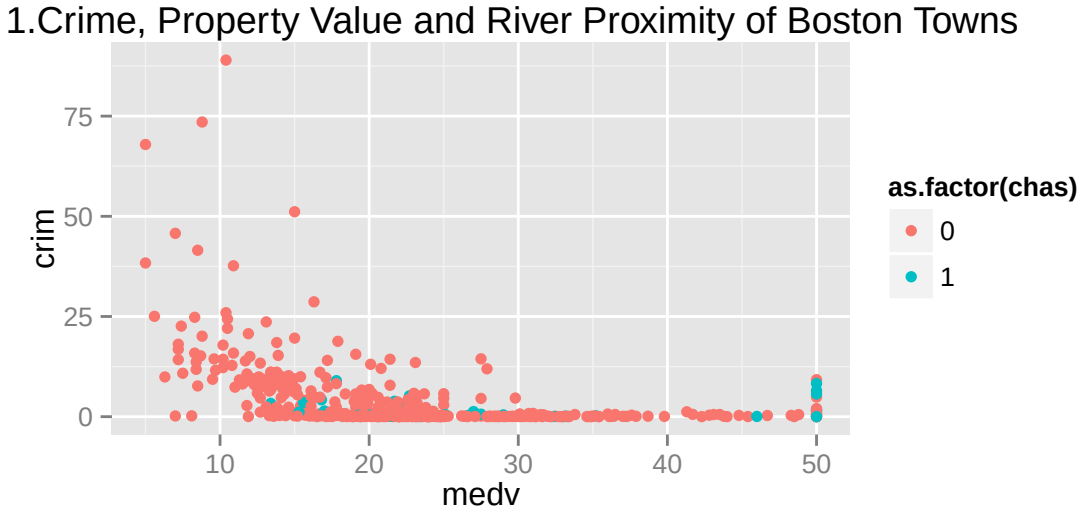


As you can see getting results, once you get the knack of it, is only half of the way. The other, and more important, half is trying to make sense of the results. R cannot do that for you. For this you need to use a better tool: **your brain** (scepticism, curiosity, creativity, a lifetime of knowledge) and what Kaiser Fung calls "numbersense".

2.8 Titles and legends in ggplot2

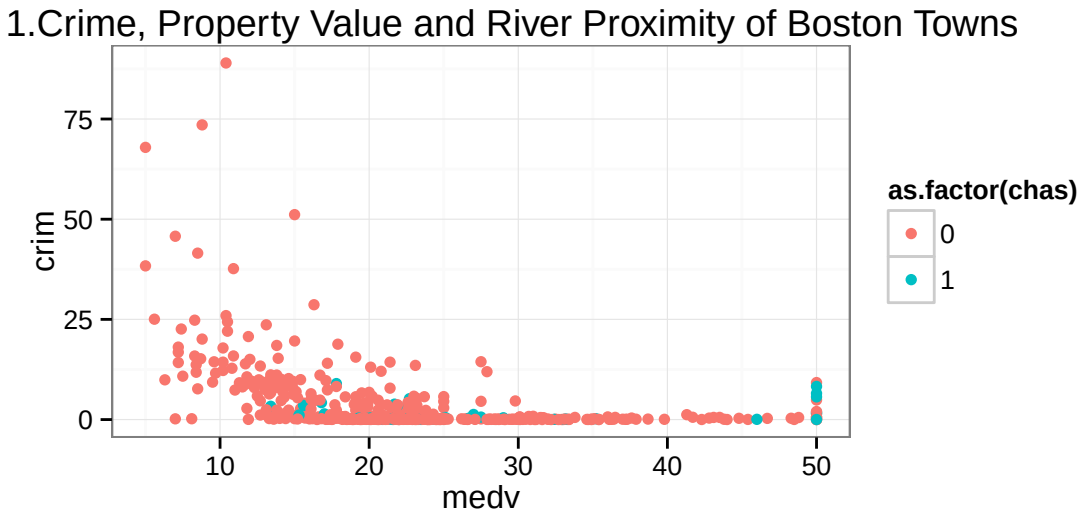
We have introduced a number of various graphical tools, but what if you want to customise the way the produce graphic looks like? Here I am just going to give you some code for how to modify the titles and legends you use. For adding a title for a `ggplot` graph you use `ggtitle()`.

```
ggplot(Boston, aes(x = medv, y = crim, colour = as.factor(chas))) +
  geom_point() +
  ggtitle("Fig 1.Crime, Property Value and River Proximity of Boston Towns")
```



If you don't like the default background theme for `ggplot` you can use a black and white background adding `theme_bw()` as a layer:

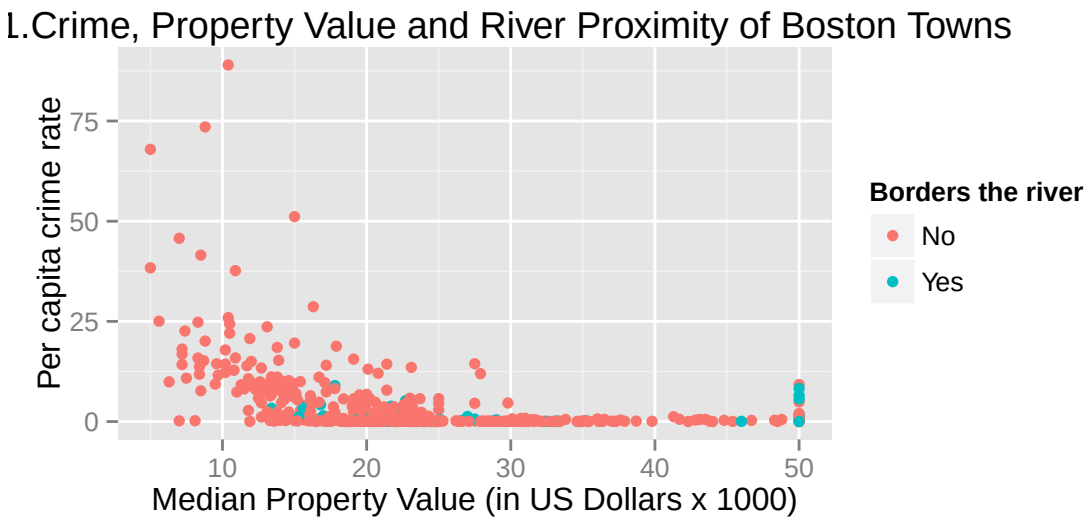
```
ggplot(Boston, aes(x = medv, y = crim, colour = as.factor(chas))) +
  geom_point() +
  ggtitle("Fig 1.Crime, Property Value and River Proximity of Boston Towns") +
  theme_bw()
```



Using `labs()` you can change the text of axis labels (and the legend title), which may be handy if your variables have cryptic names. Equally you can manually name the labels in a legend. The value for "chas" are 0 and 1. This is not informative. We can change that.

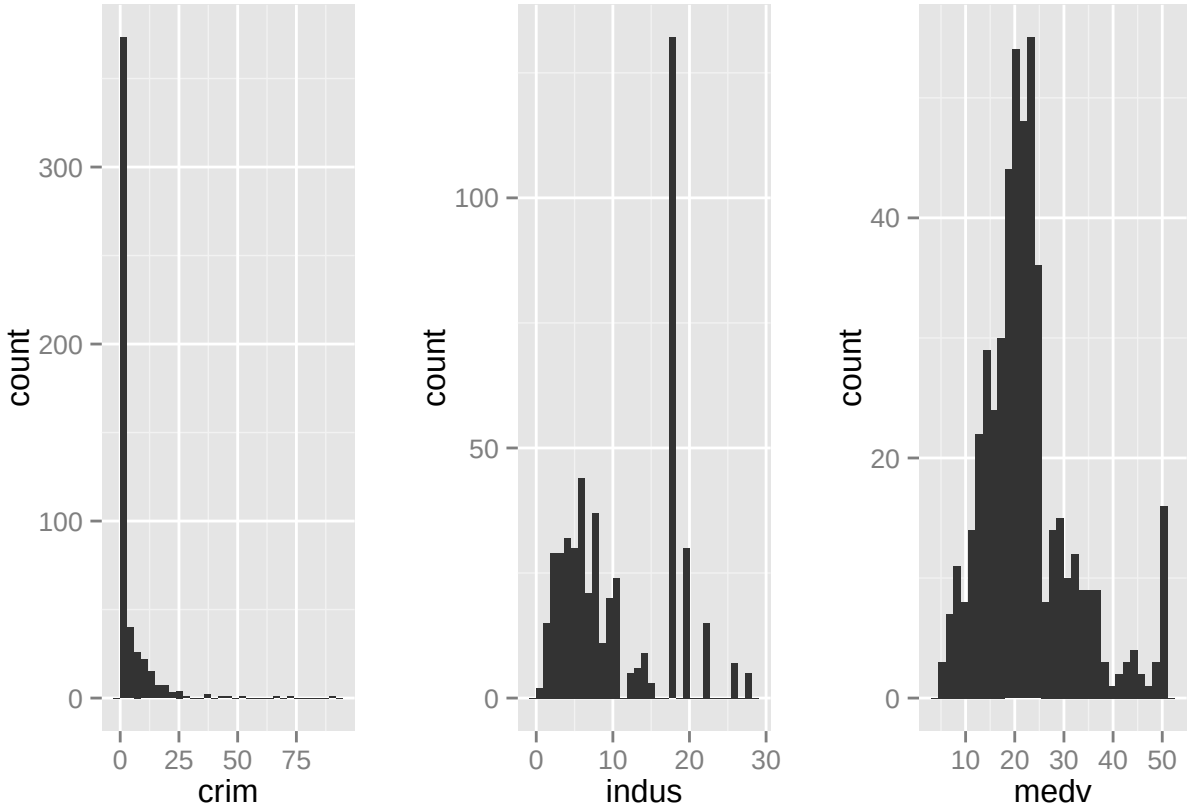
```
ggplot(Boston, aes(x = medv, y = crim, colour = as.factor(chas))) +
  geom_point() +
  ggtitle("Fig 1.Crime, Property Value and River Proximity of Boston Towns") +
  labs(x = "Median Property Value (in US Dollars x 1000)",
```

```
      y = "Per capita crime rate",
      colour = "Borders the river") +
  scale_colour_discrete(labels = c("No", "Yes"))
```



Sometimes you may want to present several plots together. For this the `gridExtra` package is very good. You will first need to install it and then load it. You can then create several plots and put them all in the same image.

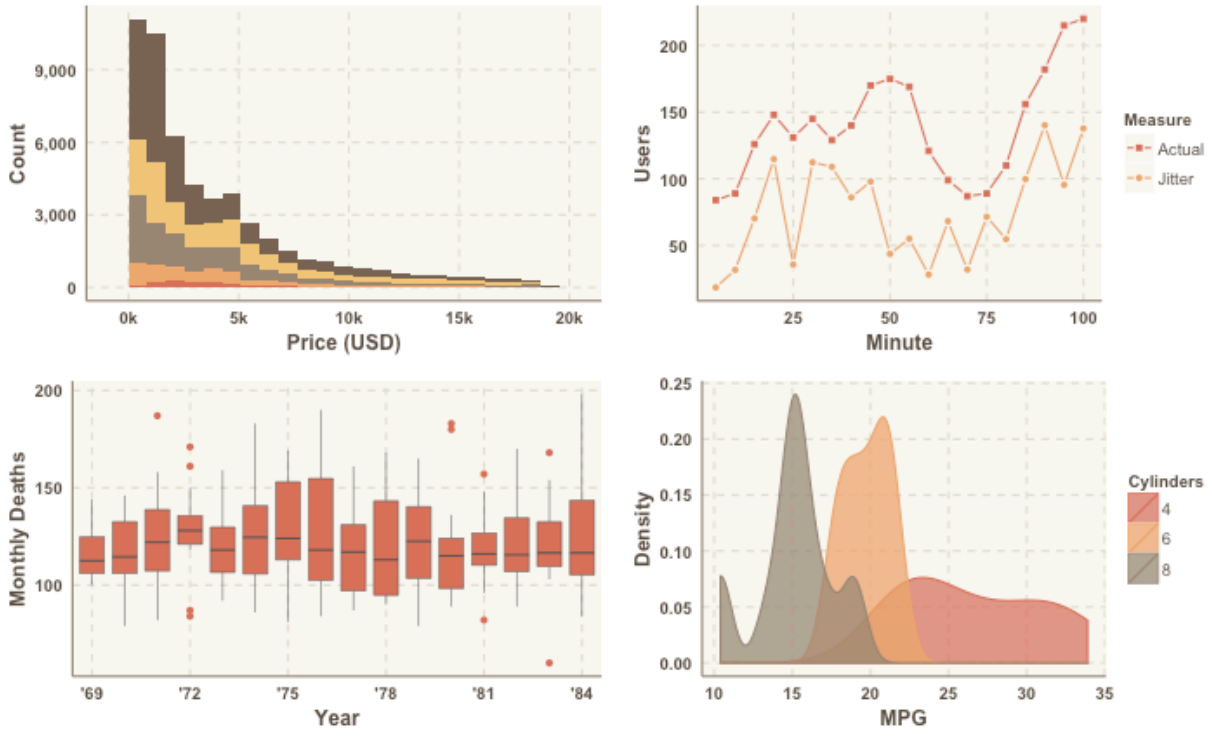
```
#You may need to install first with install.packages("gridExtra")
library(gridExtra)
#Store your plots in various objects
p1 <- qplot(x=crim, data=Boston)
p2 <- qplot(x=indus, data=Boston)
p3 <- qplot(x=medv, data=Boston)
#Then put them all together using grid.arrange()
#ncol tells R we want them side by side, if you want them one in top of the other
#try ncol=1, in this case, however ncol=2 would possibly be the better solution.
grid.arrange(p1, p2, p3, ncol=3)
```



There are packages that allow you to add new themes to ggplot2 graphics. The *ggthemes* package is one of them. You can install this package in the usual way. Check here for details. There is also another one called *ggthemr* still in development. If you want to install the latter you would need to use the following code, but keep in mind that this package is still not too stable. You may prefer to way until it is available in the CRAN repository.

```
devtools::install_github("ggthemr", "cttobin")
```

This is an example of the `dust` theme in this package. You can see other themes in the github page for the package.



2.9 Plotting categorical data: bar charts

You may be wondering what about categorical data? So far we have only discussed various visualisations where at least one of your variables is quantitative. When your variable is categorical you can use bar plots (similar to histograms). We map the factor variable in the aesthetics and then use the `geom_bar()` function to ask for a bar chart.

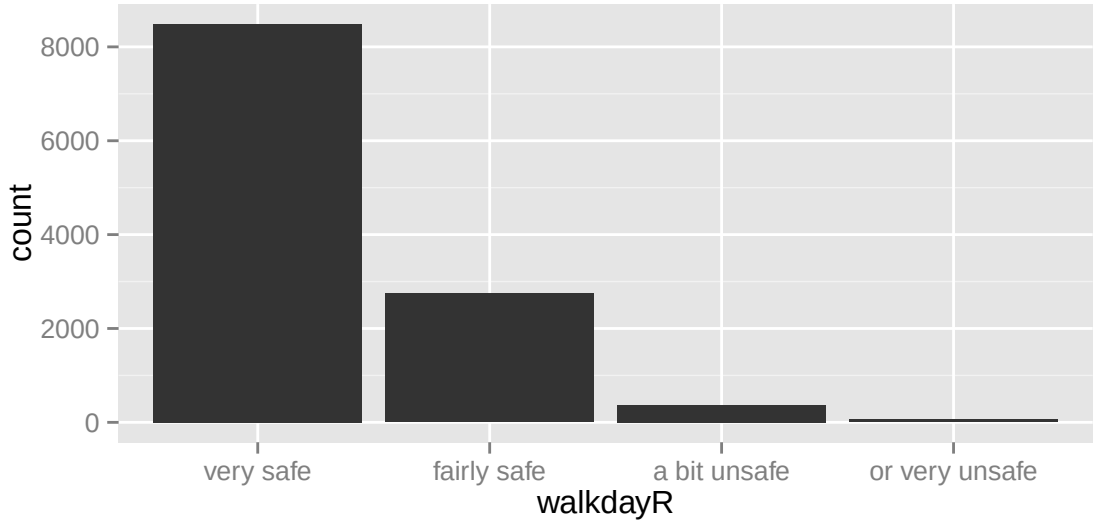
```
ggplot(BCS0708, aes(x=walkday)) +
  geom_bar()
```

Unfortunately, the levels in this factor are ordered by alphabetical order, which is confusing. We can modify this by reordering the factors levels first -click here for more details. You could do this within the ggplot function (just for the visualisation), but in real life you would want to sort out your factor levels in an appropriate manner more permanently. As discussed last week, this is the sort of thing you do as part of pre-processing your data. And then plot.

```
#Print the original order
print(levels(BCS0708$walkday))
```

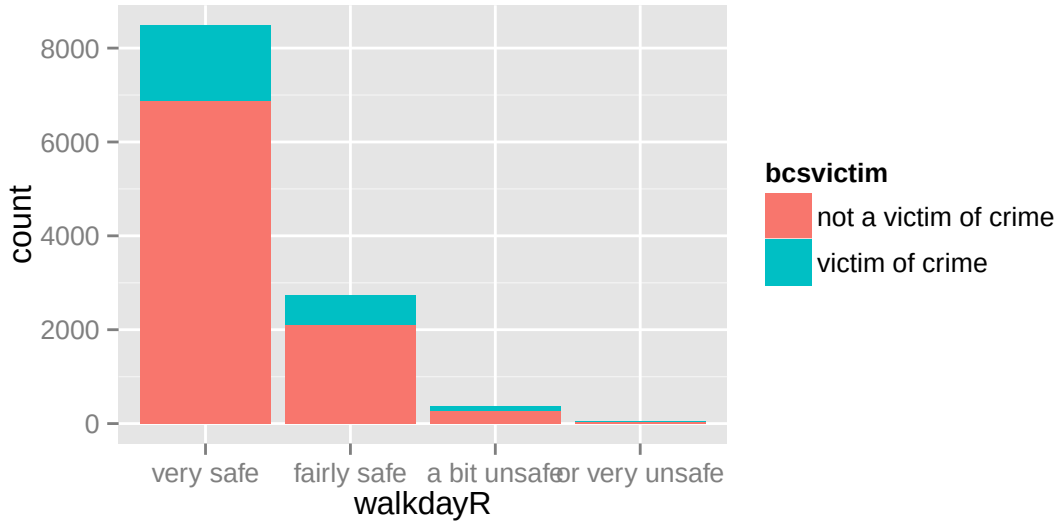
```
## [1] "a bit unsafe"   "fairly safe"    "or very unsafe" "very safe"
```

```
#Reordering the factor levels from very safe to very unsafe (rather than by alphabet).
#Notice that I am creating a new variable, it is often not unwise to do this to avoid
#messing up your original data.
BCS0708$walkdayR <- factor(BCS0708$walkday, levels=c('very safe',
  'fairly safe','a bit unsafe','or very unsafe'))
#Plotting the variable again (and subsetting out the NA data)
ggplot(subset(BCS0708, !is.na(walkdayR)), aes(x=walkdayR)) +
  geom_bar()
```



We can also map a second variable to the aesthetics, for example, let's look at victimisation in relation to feelings of safety. For this we produce a **stacked bar chart**.

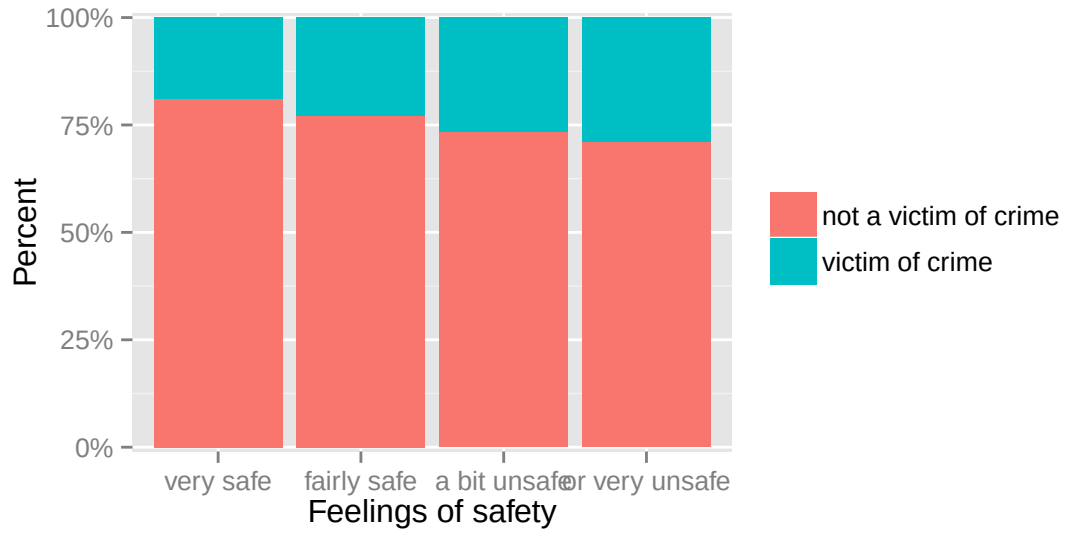
```
ggplot(subset(BCS0708, !is.na(walkdayR)), aes(x=walkdayR, fill=bcsvictim)) +
  geom_bar()
```



These sort of stacked bar charts are not terribly helpful if you are interested in understanding the relationship between these two variables. Instead what you want is a **proportional stacked bar chart**, that gives you the proportion of your "explanatory variable" (here victimisation in the past 12 months) within each of the levels of your "response variable" (here feelings of safety).

First we need to scale the data to 100% within each stack and then plot. The code here is also more complex. Again, I'm not expecting you to fully understand all of it. In your early days of using R, you will find yourself cutting and pasting chunks of code that you will only fully understand with practice.

```
#First we create a subset with the relevant data
.df1 <- data.frame(x = BCS0708$walkdayR, z = BCS0708$bcsvictim)
#Then we compute the proportions within the stacks
.df1 <- as.data.frame(with(.df1, prop.table(table(x, z), margin = NULL)))
#Finally we plot
# scale_y_continuous adapts the scale of the y axis and. in this case, asks to
#express the values in percentage terms
#xlab gives you another way of changing the label for the x axis
#whereas ylab changes the label for the Y axis
#guides in this case removes the title of the legent (bcsvictim),
#since the labels of the categories are self-explanatory
stbcf <- ggplot(.df1, aes(x = x, y = Freq, fill = z)) +
  geom_bar(position = "fill", stat = "identity") +
  scale_y_continuous(expand = c(0.01, 0), labels = scales::percent_format()) +
  xlab("Feelings of safety") +
  ylab("Percent") +
  guides(fill = guide_legend(title = NULL))
#Autoprint the plot, notice how earlier rather than printing directly I stored the plot
#in the object with this name (the reason is that I want to use this plot later and
#don't want to rerun all the code again)
stbcf
```

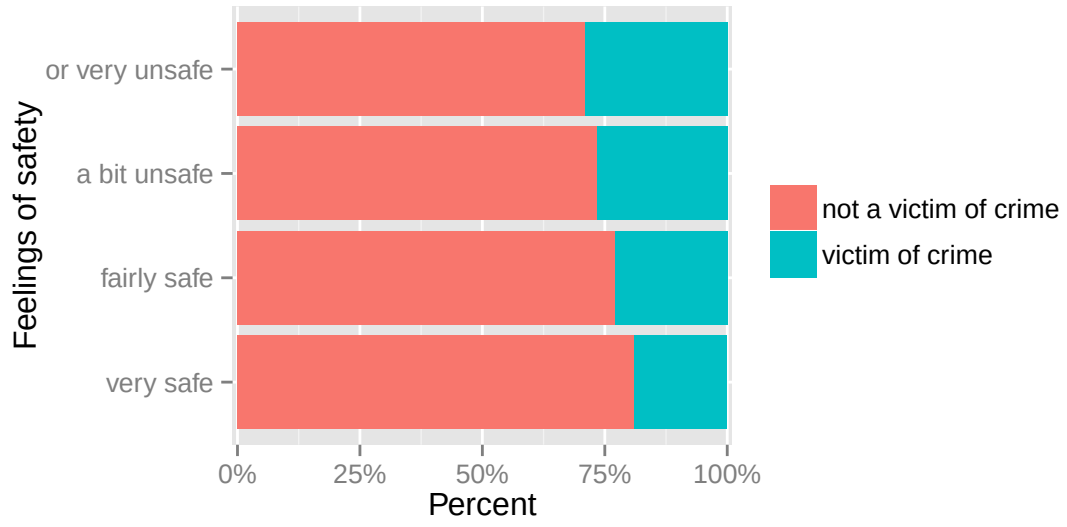



```
rm(.df1) #to remove the data frame we created
```

Here we can see that the those that express less feelings of safety are more likely to have experienced a victimisation in the past 12 months.

Sometimes, you may want to flip the axis, so that the bars are displayed horizontally. You can use the `coord_flip()` function for that.

```
#First we invoke the plot we created and stored earlier and then we add an additional #specification wi
stbcf + coord_flip()
```

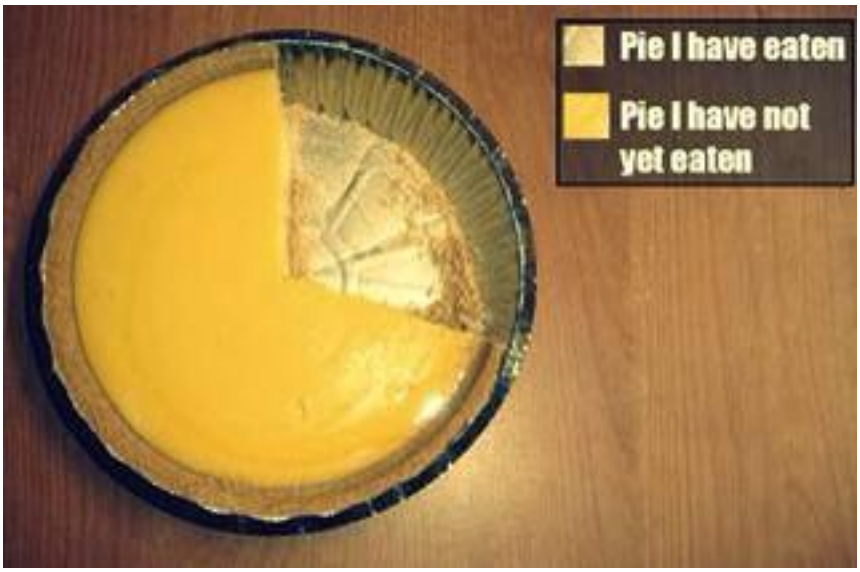


You can also use `coord_flip()` with other `ggplot` plots (e.g., boxplots).

A particular type of bar chart is the divergent stacked bar chart, often used to visualise **Likert scales**. You may want to look at some of the options available for it via the HH package, the likert package or sjPlot. But we won't cover them here in detail.

Keep in mind that knowing *how* to get R to produce a particular visualisation is only half the job. The other half is knowing *when* to produce a particular kind of visualisation. This blog, for example, discusses some of the problems with stacked bar charts and the exceptional circumstances in which you may want to use them.

There are other tools sometimes used for visualising categorical data. Pie charts is one of them. But don't use pie charts. They are evil. This is the only pie chart you will see in this course:



Also increasingly popular are **mosaic plots**. R is pretty good at doing them. However, we have covered already a lot of ground, plus I have some sympathy for Stephen Few's arguments against mosaic plots.

2.10 Further resources

By now you should know the path to data analysis wisdom will take time. The good news is that we live in a time where there are multiple (very often free) resources to help you along the way. The time where this knowledge was just the preserve of a few is long distant. If you are a motivated and disciplined person there is a lot that you can do to further consolidate and expand your data visualisation skills without spending money. I already recommended you the excellent ggplot2 online documentation. Here I just want to point you to a a few useful resources that you can pursue in the future.

First **MOOCS (Massive Online Open Courses)**. There are a couple of useful MOOCS that provide training in data visualisation for free (well, you can pay if you want a certificate, but you can also use the resources and learn for free):

- UDACITY Data Analysis with R. This course offered by programmers at Facebook will teach you loads of cool things about ggplot2 in a very interactive manner. The course is very well designed and you can take it at your own pace. Beware, the product placement is very much in your face, but other than that is a really good course.
- COURSERA Exploratory Data Analysis. The Coursera approach relies uses more traditional video lectures. It complements the UDACITY course in that it provides a general overview of graphics in base R, the lattice package and ggplot2. Also, the second part of the course covers slightly more specialised tools for identifying groups.
- You may also want to keep your eyes open for any new edition of Alberto Cairo's MOOC on "Data Visualisation and Infographics" which covers not the software but the principles and theory of good static and interactive visualisation. I understand he is planning to offer a new edition of this course during this academic year.

Second, **online tutorials**. One of the things you will also soon discover is that R users are keen to write "how to" guides in some of the 573 R devoted blogs or as part of Rpubs. Using google or similar you will often find solutions to almost any problem you will encounter using R. For example, in this blog there are a few tutorials that you may want to look to complement my own:

- Cheatsheet for visualising scatterplots
- Cheatsheet for visualizing distributions
- Cheatsheet for barplots

Third, *blogs on data visualisation.* If you use Feedly or a similar blog aggregator, you should add the following blogs to your reading list. They are written by leading people in the field and are full of useful advice: + Flowing Data + The Functional Art + Visual Business Intelligence + chartsnthings + Data Revelations

Fourth, **resources for visualisations we don't have the time to cover**. R is way ahead some of the more popular data analysis programs you may be more familiar with (e.g SPSS). There's many things you can do. First, if you like maps, **R can also be used to produce visualisations of spatial data**. There are various resources to learn how to do this. But you may want to start with the handouts produced by Dr. Nick Bearman (University of Liverpool):

- Getting started with maps in R
- Using google maps with R

Next semester we offer a course on crime mapping, but we will be using friendlier point-and-click user interfaces (ArcGis though soon migrating to QGis) for that (rather than R).

Second, you may be interested in producing interactive graphics in the internet. Data journalism is making big inroads into this area. You may want to have a look at the ggvis package, produced by the R Studio team, or rcharts.

Finally, you may want to consider some of the general books that will help you to produce better and more appropriate visualizations. These are some suggestions:

- The R Graphics Cookbook always sitting next to my desktop and possibly the most useful R reference book I always keep going back to.
- Graphical Data Analysis with R.
- The Truthful Art. You can check the author's list of recommended resources here.
- Data Points: Visualizations that mean something.

```
sessionInfo()
```

```
## R version 3.2.2 (2015-08-14)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 8 x64 (build 9200)
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.1252
## [2] LC_CTYPE=English_United Kingdom.1252
## [3] LC_MONETARY=English_United Kingdom.1252
## [4] LC_NUMERIC=C
```

```
## [5] LC_TIME=English_United Kingdom.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] gridExtra_2.0.0 GGally_0.5.0    mgcv_1.8-7      nlme_3.1-121
## [5] ggplot2_1.0.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.1      knitr_1.11       magrittr_1.5     MASS_7.3-43
##  [5] munsell_0.4.2    colorspace_1.2-6 lattice_0.20-33  stringr_1.0.0
##  [9] plyr_1.8.3       tools_3.2.2      grid_3.2.2       gtable_0.1.2
## [13] htmltools_0.2.6  yaml_2.1.13      digest_0.6.8     Matrix_1.2-2
## [17] reshape2_1.4.1   formatR_1.2      evaluate_0.7.2   rmarkdown_0.8
## [21] labeling_0.3     stringi_0.5-5    scales_0.3.0     reshape_0.8.5
## [25] proto_0.3-10
```