Creating Graphics with R

Katie Anderson (Data Services)

April 15, 2015

I. Base Graphics

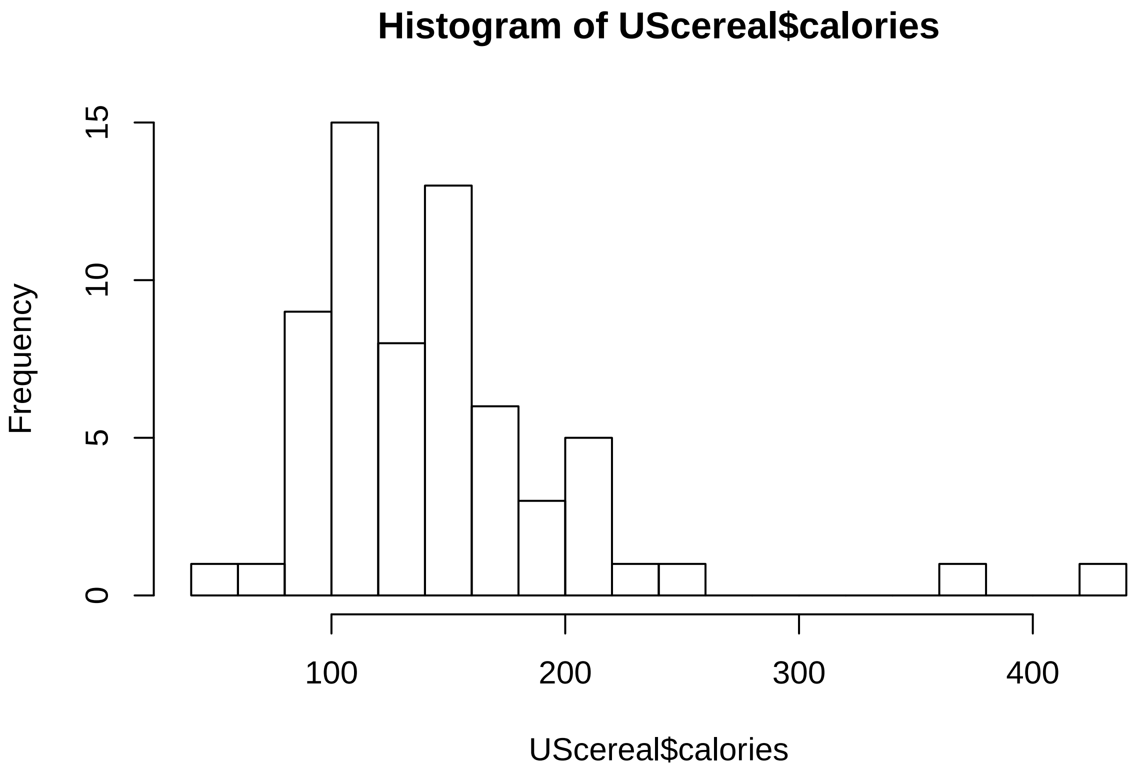
```r
library(MASS)
data(UScereal)
head(UScereal)
```

```
##                           mfr calories   protein      fat   sodium
## 100% Bran                   N 212.1212 12.121212 3.030303 393.9394
## All-Bran                    K 212.1212 12.121212 3.030303 787.8788
## All-Bran with Extra Fiber   K 100.0000  8.000000 0.000000 280.0000
## Apple Cinnamon Cheerios     G 146.6667  2.666667 2.666667 240.0000
## Apple Jacks                 K 110.0000  2.000000 0.000000 125.0000
## Basic 4                     G 173.3333  4.000000 2.666667 280.0000
##                               fibre    carbo   sugars shelf potassium
## 100% Bran                 30.303030 15.15152 18.18182     3 848.48485
## All-Bran                  27.272727 21.21212 15.15151     3 969.69697
## All-Bran with Extra Fiber 28.000000 16.00000  0.00000     3 660.00000
## Apple Cinnamon Cheerios    2.000000 14.00000 13.33333     1  93.33333
## Apple Jacks                1.000000 11.00000 14.00000     2  30.00000
## Basic 4                    2.666667 24.00000 10.66667     3 133.33333
##                           vitamins
## 100% Bran                 enriched
## All-Bran                  enriched
## All-Bran with Extra Fiber enriched
## Apple Cinnamon Cheerios   enriched
## Apple Jacks               enriched
## Basic 4                   enriched
```

A. Choosing the Right Graph for your Data

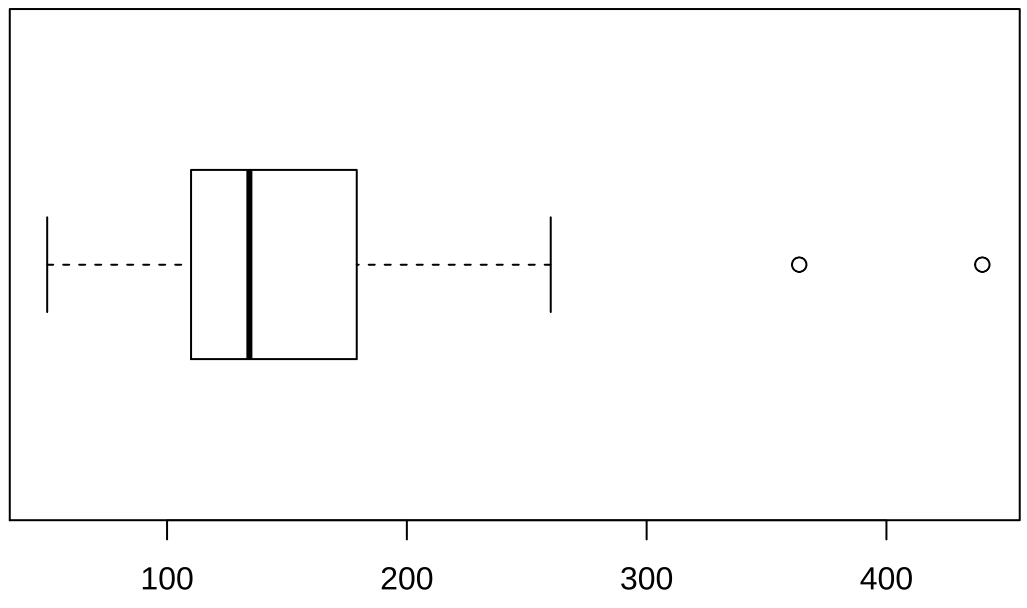
One Continuous Variable A histogram is a graph used to show a distribution of data. On the x-axis is for a continuous variable which the user is interested in. The y-axis typically displays either the frequency of values inside the bins or the density of the distribution. Bins, or the width of the bars, can be determined by the creator of the graph - but changing the width of the bins can potentially have a dramatic effect on the shape of the distribution.

```r
hist(UScereal$calories, breaks = 15)
```



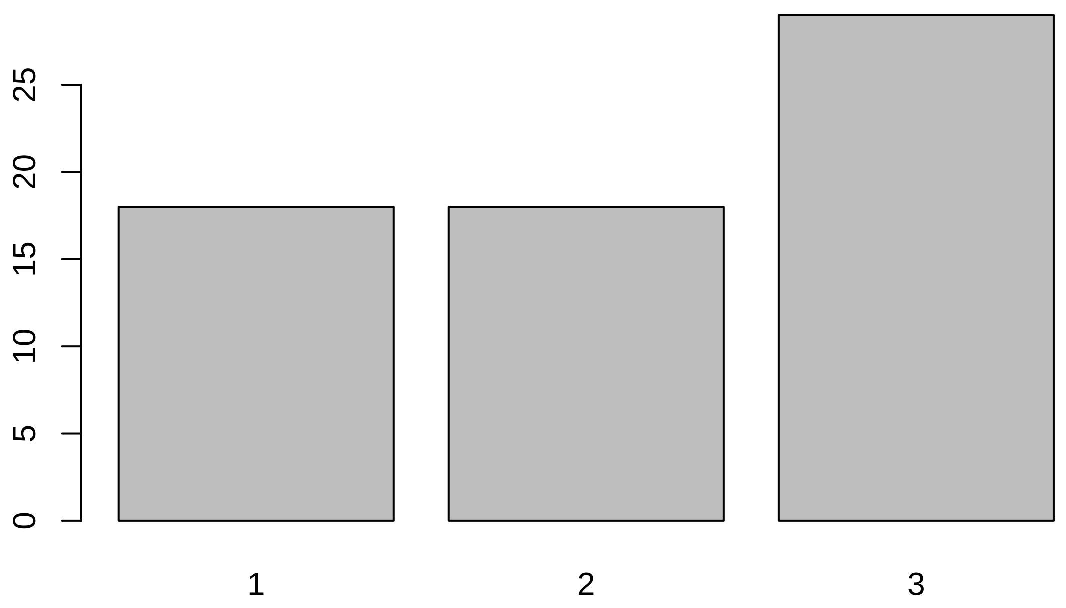
A box plot, also called a box and whisker plot, is used to show the distribution of data. The box, represents the Inter Quartile Range (IQR) which is the center 50% of the data - the edges of the box represent the 75th an 25th percentiles respectively. Typically there is a marker or line in the box, which indicates the median. The lines, or whiskers, add 1.5 times the IQR to the 75th percentile, or subtract 1.5 times from the 25th percentile. Any outliers are represented by points or markers outside of the whiskers reach. The horizontal = TRUE argument, displays the boxplot horizontally rather than vertically.

```r
boxplot(UScereal$calories, horizontal = TRUE)
```



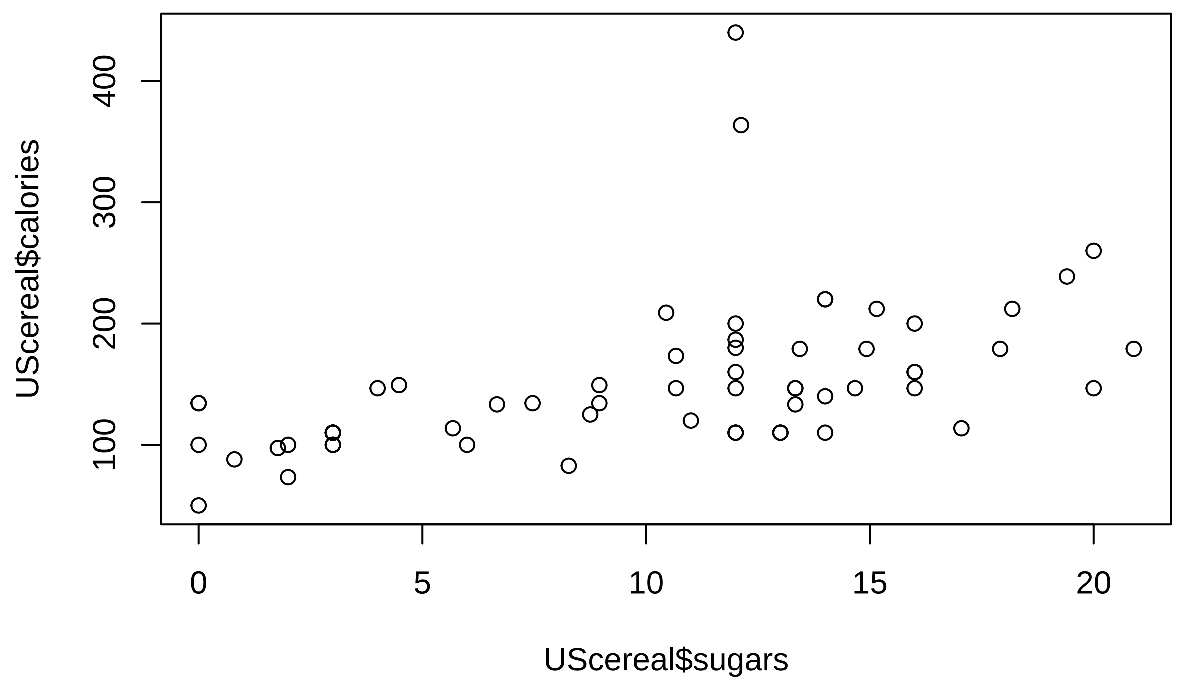
One Categorical (Discrete) Variable A bar plot, or bar graph is used to see the distribution between discrete categories. Typically the frequency of the categories appears on the y-axis, however sometimes a statistics such as the mean of a continuous variable is shown.

```r
barplot(table(UScereal$shelf))
```



Two Continuous Variables A scatterplot is a graph that shows the relationship between two continuous variables, where the dependent variable goes on the vertical axis (y-axis) and the independent variable is displayed on the horizontal axis (x-axis). Each point on the graph represents a single observation.

```r
plot(x = UScereal$sugars, y = UScereal$calories)
```

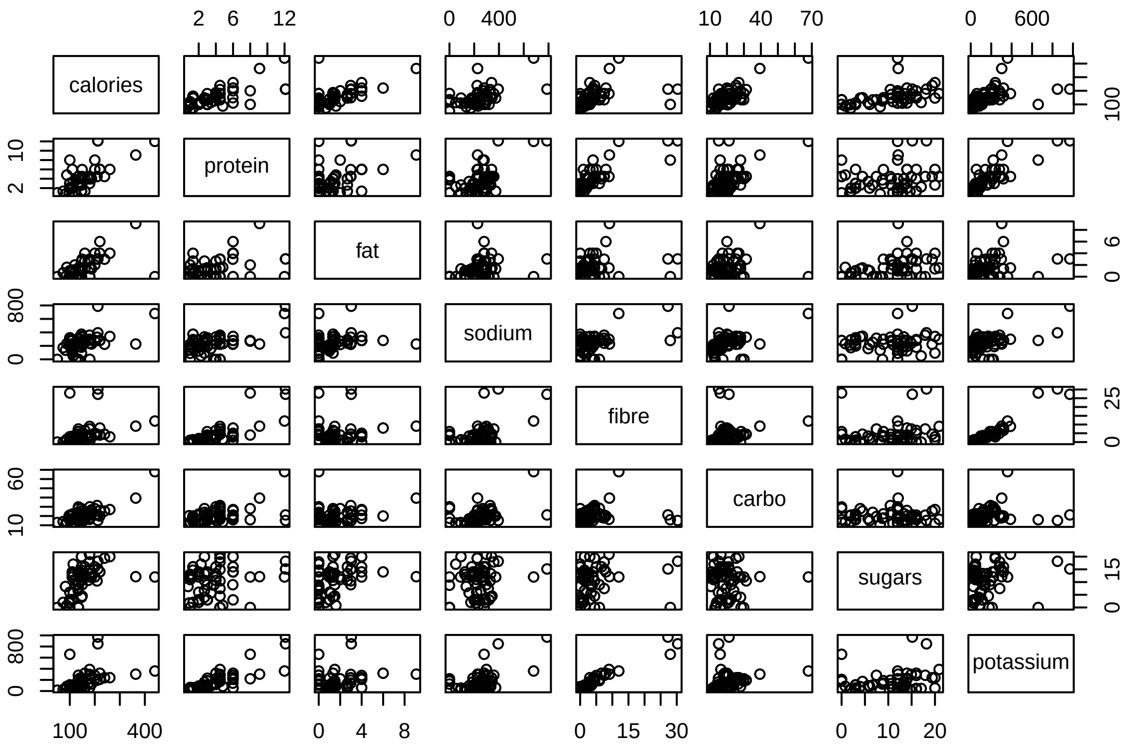


For many graphics in R, you can use either dollar sign notation, as shown above or use formula notation, as seen below:

```r
plot(calories ~ sugars, data = UScereal)
```

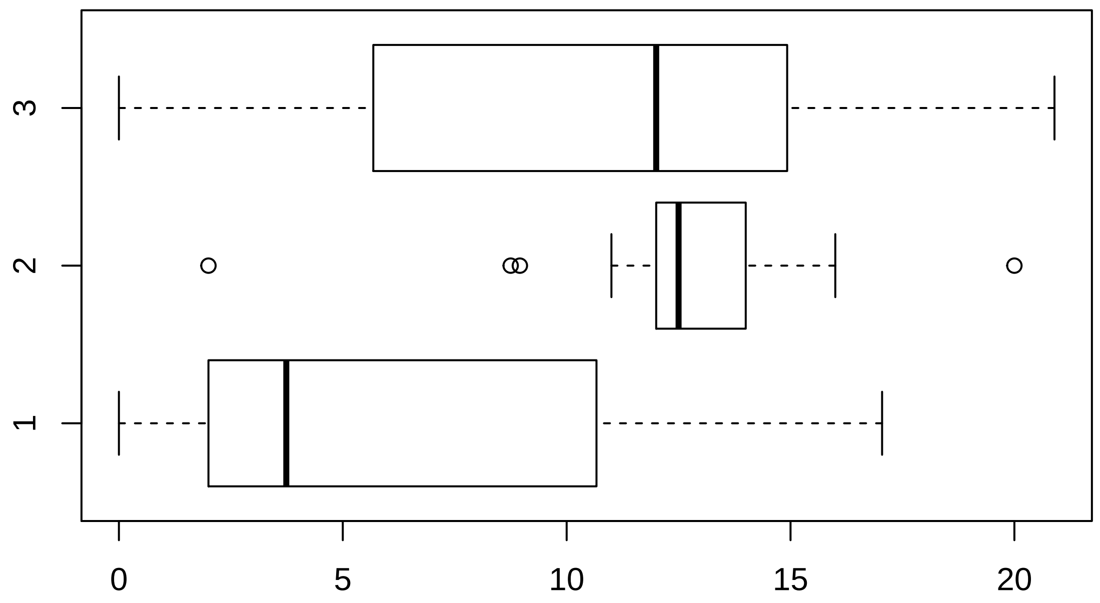
When you have many continuous variables, and you want to see the relationship between all possible pairs - you can produce a scatterplot matrix.

```r
plot(UScereal[, c(2:8, 10)])
```



One Continuous Variable and One Categorical Variable When you want to show the relationship between one continuous varibale and one categorical variable you can show the distributions of a continuous variable broken down by the groups of the categorical variable. For example, a boxplot broken up by groups:

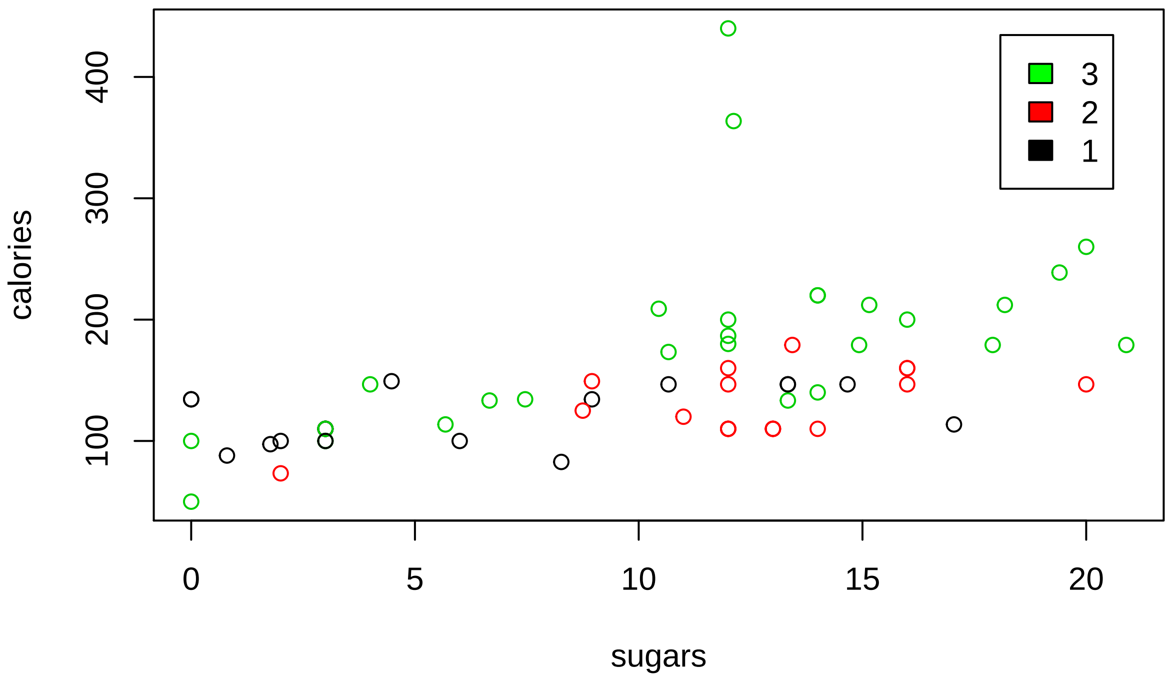
```r
boxplot(sugars ~ shelf, data = UScereal, horizontal = TRUE)
```



What advantages do we get by looking at the data in this way instead of looking at the boxplot from the first graph?

Two Continuous Variables and One Categroical Variable When you want to see the effect of a grouping variable on the relationship between two continuous variable, you can add a categorical variable as a color factor on a scatterplot. Of course you will need to add a legend which we discuss a little bit later.

```
plot(calories ~ sugars, data = UScereal, col = shelf)
legend('topright', inset = .05, legend = c(3,2,1), fill = c('green', 'red', 'black'))
```

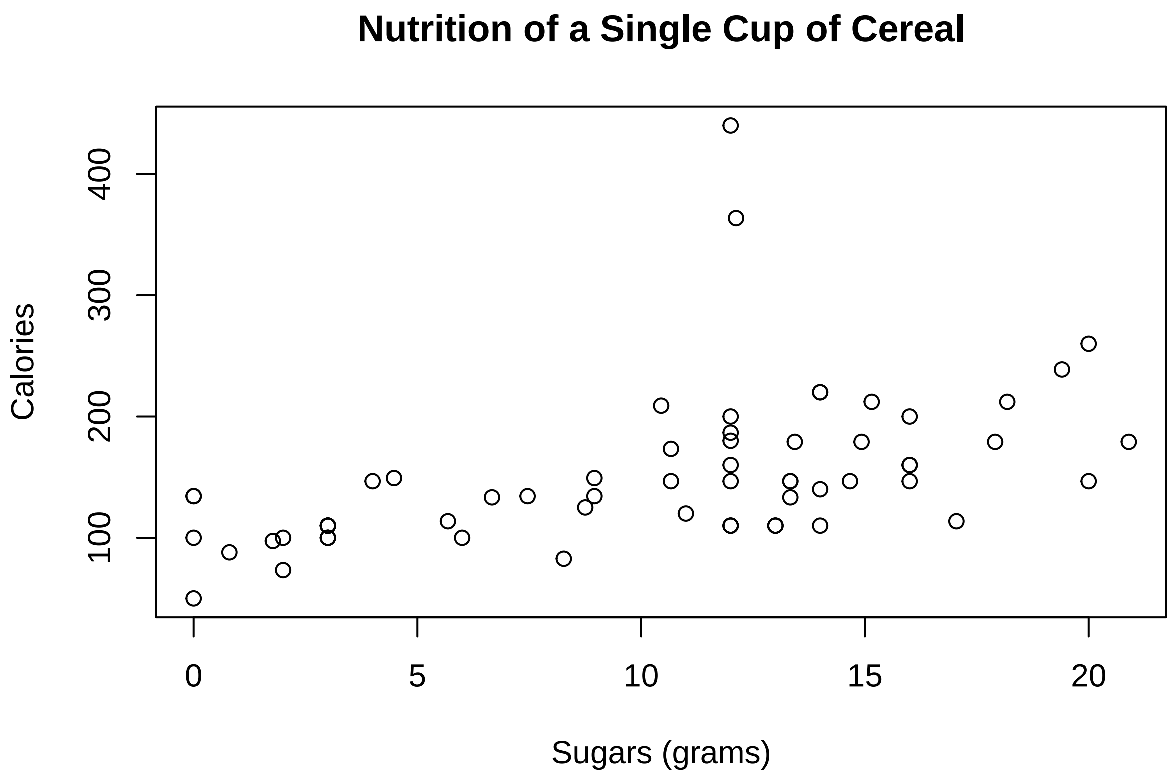


B. Graphing Elements

Each element of a graph should be chosen wisely. It is important that the aspects of your graph convey your story effectively.

Titles Your graph should be properly labeled, including axis labels and a title. Your axis titles should also include units of measurement. Your main title should describe what you are seeing without duplicating other information portrayed by the graph. For example, you may not want to title this graph 'Calories versus Sugars' because not only is that too vague, but the axis already provide that information.

```r
plot(calories ~ sugars, data = UScereal, ylab = 'Calories', xlab = 'Sugars (grams)', main = 'Nutrition
```

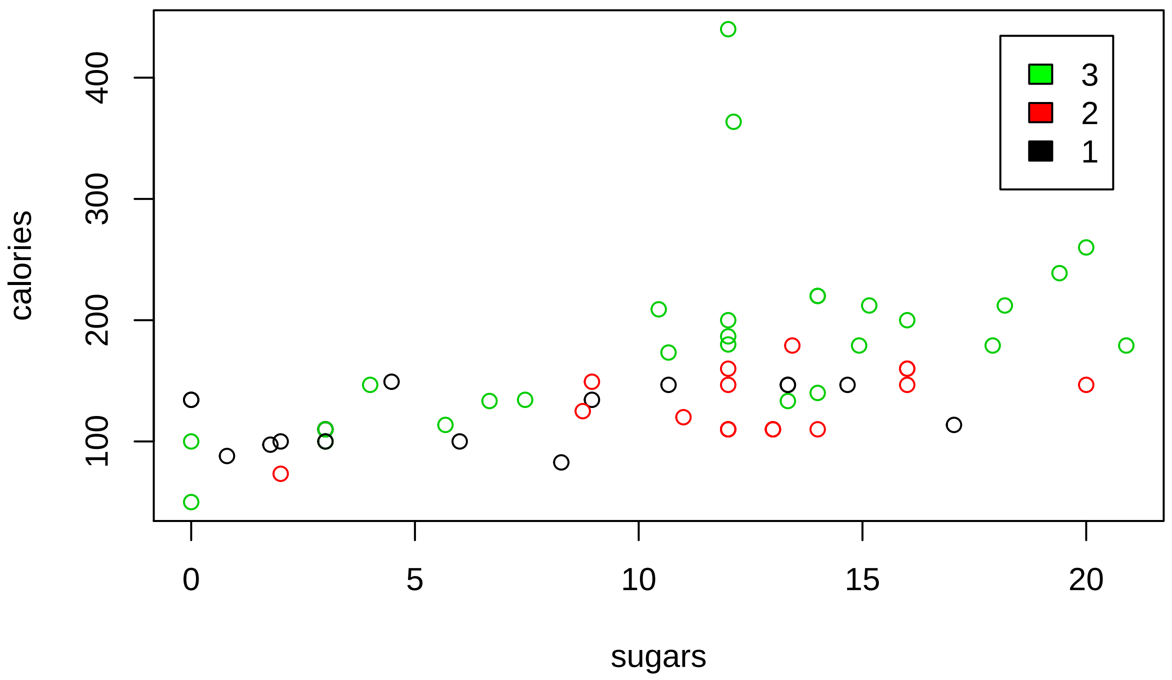


Titles can also be added on afterwards but make sure to use the ann = FALSE argument in the main plot if you are adding the x-axis and y-axis titles afterwards.

```r
plot(calories ~ sugars, data = UScereal, ann = FALSE)
title(main = 'Nutrition of a Single Cup of Cereal', ylab = 'Calories', xlab = 'Sugars (grams)')
```

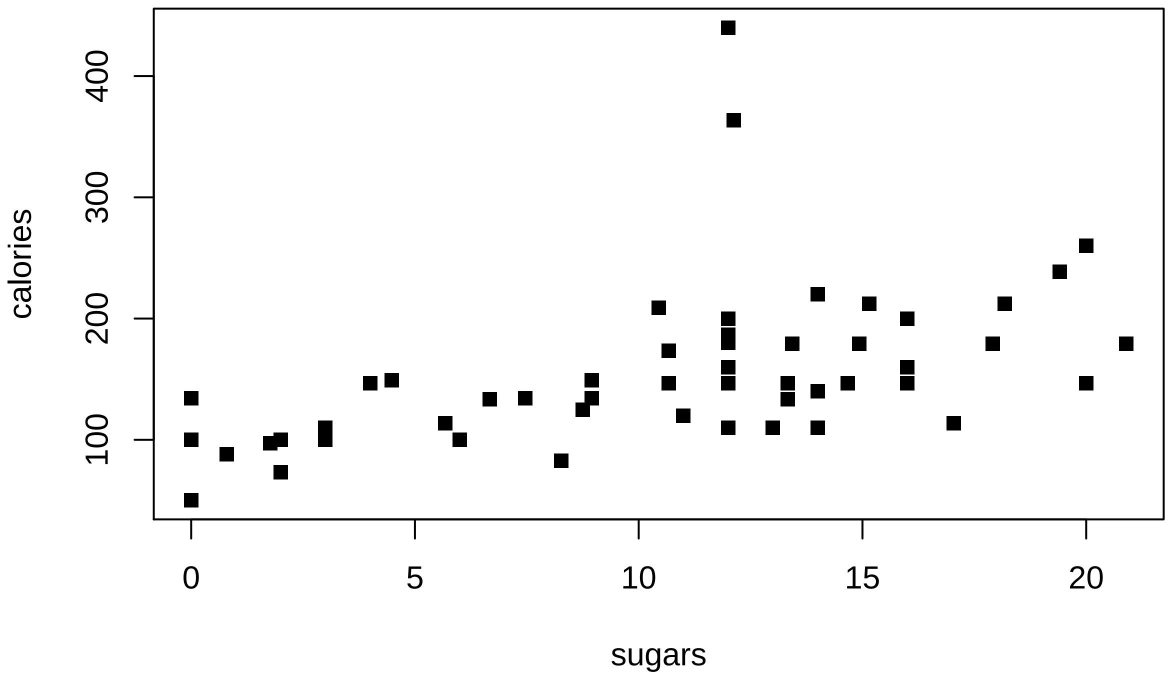
Legend Similar to the titles, the legend can be added after you create the plot.

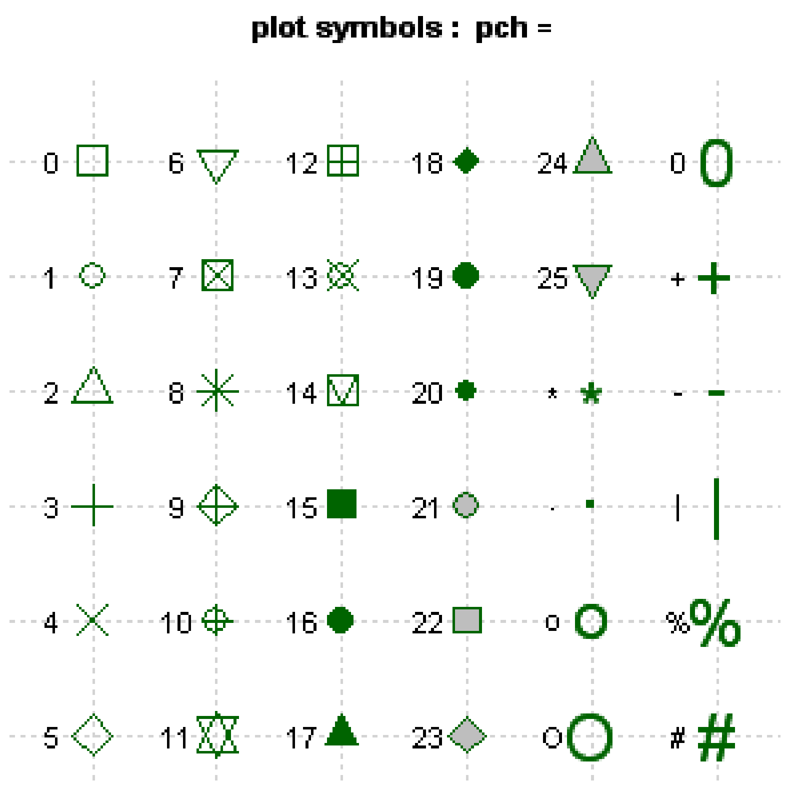
```r
plot(calories ~ sugars, data = UScereal, col = shelf)
legend('topright', inset = .05, legend = c(3,2,1), fill = c('green', 'red', 'black'))
```



Point Type and Color The pch argument assigns a point type. A list of what is available can be found on Quick-R.

```r
plot(calories ~ sugars, data = UScereal, pch = 15)
```

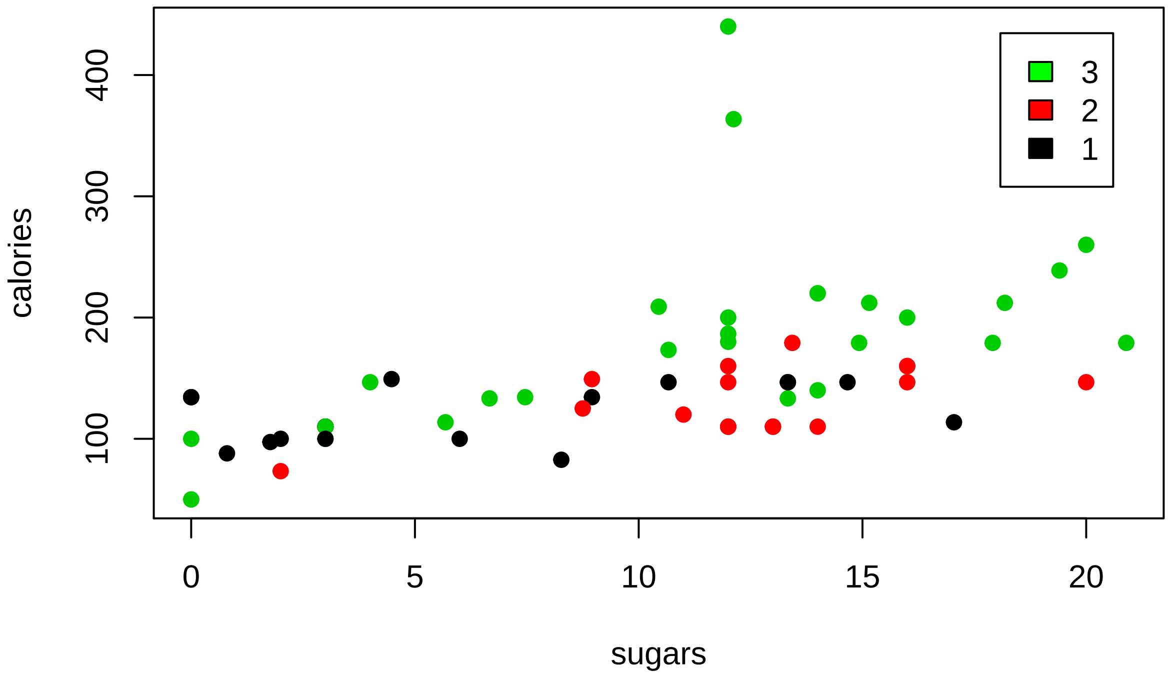




Source: Robert I. Kabacoff, Quick-R

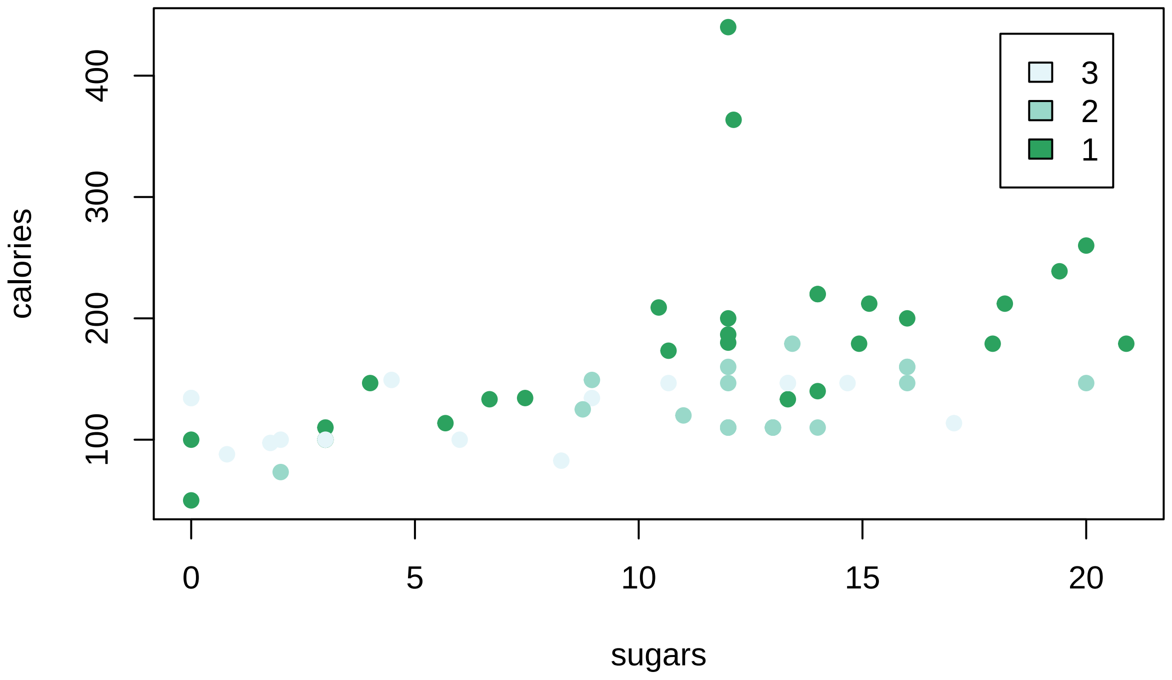
If you want to assign colors to these points, you need to include both the col and bg argumets which assing the border color and fill color respectively.

```r
plot(calories ~ sugars, data = UScereal, pch = 19, col = shelf, bg = shelf)
legend('topright', inset = .05, legend = c(3,2,1), fill = c('green', 'red', 'black'))
```



You can also represent colors by ther hexidecimal color code by assigning a palette. A pdf of all the possible colors that can be used in R can be found here. If you want to pick colors that go well together try using a tool such as ColorBrewer

```
palette(c('#e5f5f9', '#99d8c9', '#2ca25f'))
plot(calories ~ sugars, data = UScereal, pch = 19, col = shelf, bg = shelf)
legend('topright', inset = .05, legend = c(3,2,1), fill = c('#e5f5f9', '#99d8c9', '#2ca25f'))
```

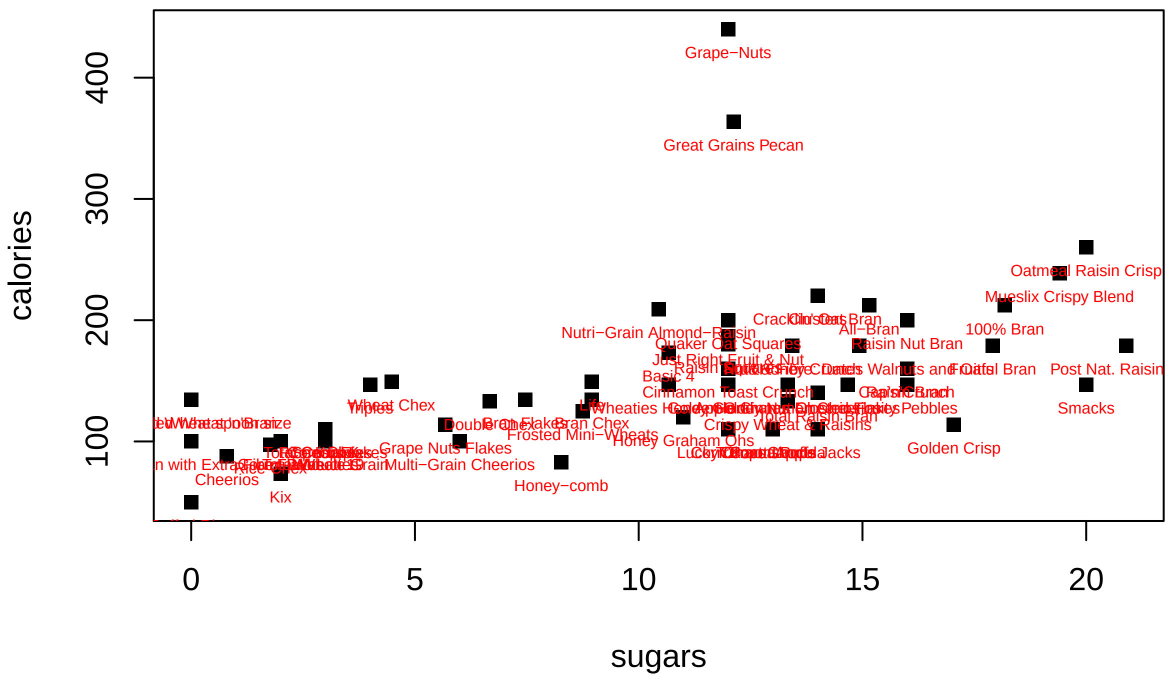


Remember each attribute of a graph should be representing one item. Meaning that if there are three groups for shelf we should either represent the different shelves by using different point types or using different colors and not both.

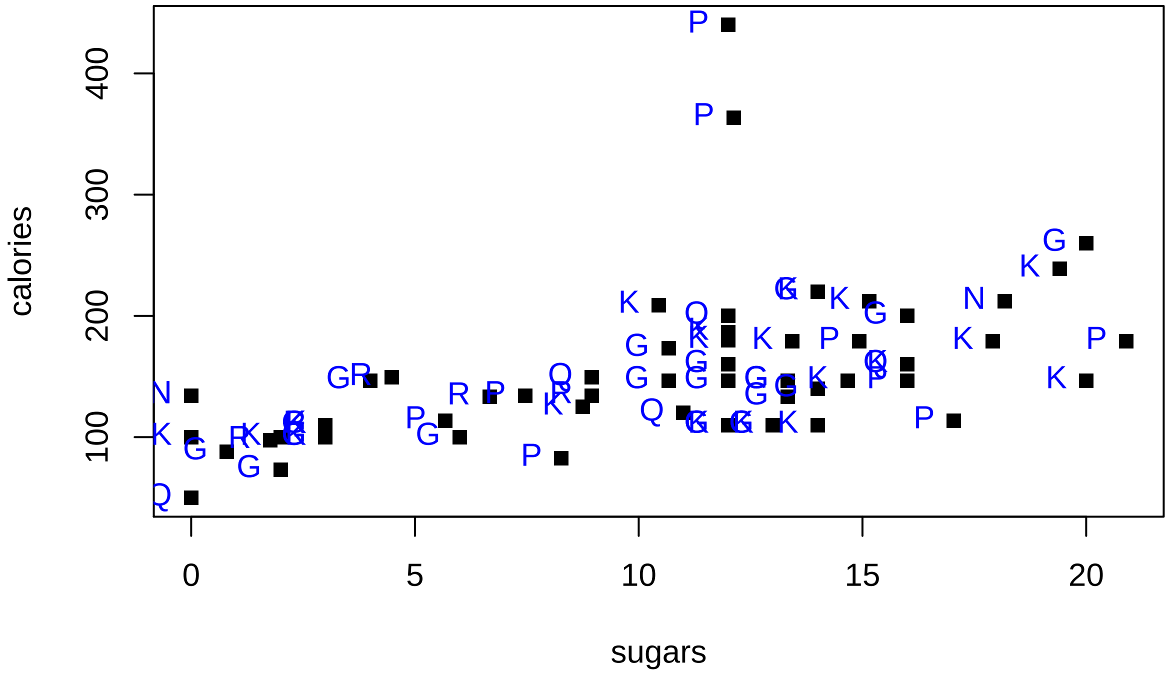
Label all Points

You can label the points with the *'text'* function by either assigning the row names, or a variable to be the labels.

```
plot(calories ~ sugars, data = UScereal, pch = 15)
text(UScereal$sugars, UScereal$calories, row.names(UScereal), col = "red", pos = 1, cex = .5)
```

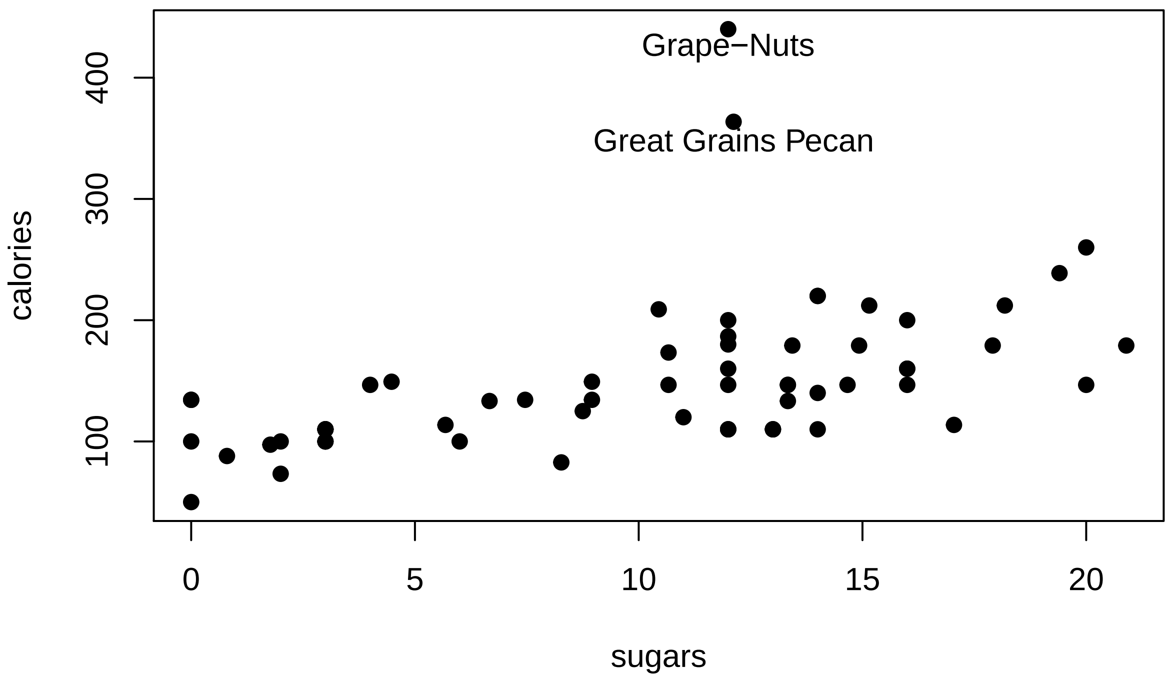


```r
plot(calories ~ sugars, data = UScereal, pch = 15)
text(UScereal$sugars, UScereal$calories, UScereal$mfr, col = "blue", pos = 2)
```



Identify Outliers You can identify outliers by manually determining a rule and then use the *'text'* function to label the points.

```r
plot(calories ~ sugars, data = UScereal, pch = 19)
outliers <- UScereal[which(UScereal$calories > 300),]
text(outliers$sugars, outliers$calories - 15, labels = row.names(outliers))
```

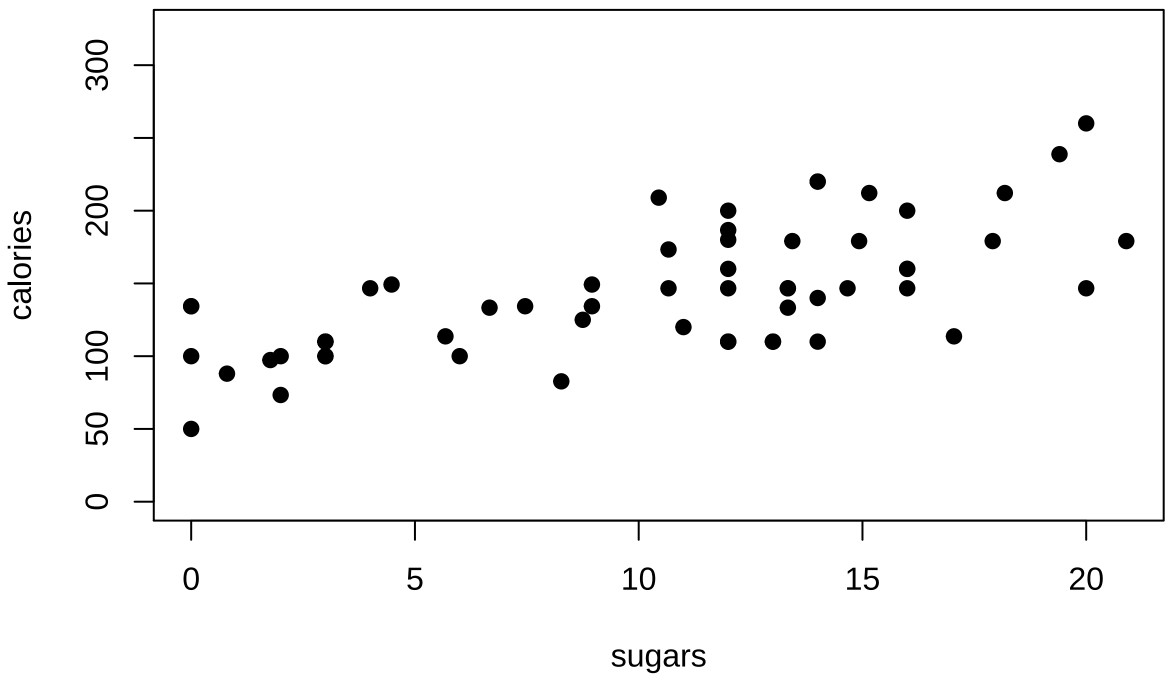


Or you use the *'identify'* fucntion to interactively choose outliers:

```
plot(calories ~ sugars, data = UScereal, pch = 19)
identify(UScereal$carbo, UScereal$calories, n = 2, labels = row.names(UScereal))
```

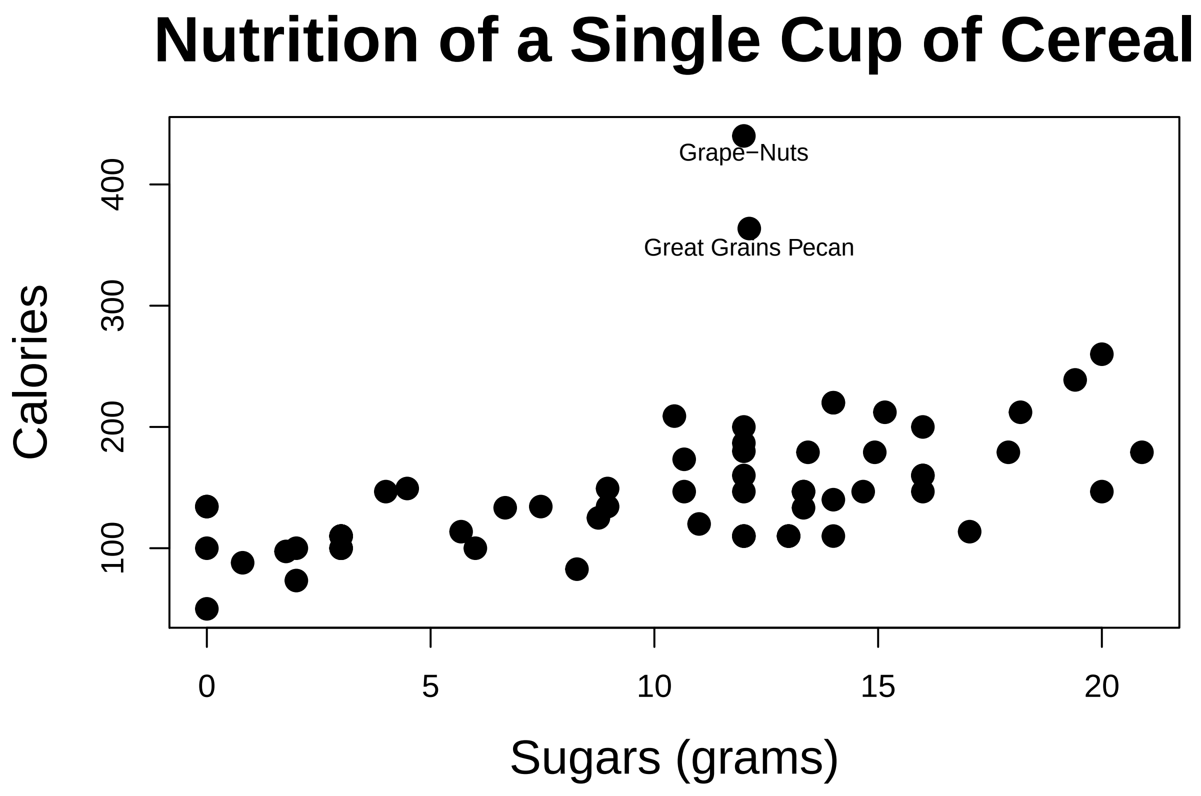
If you choose, you can change the window axis limits to remove the outliers from view:

```
plot(calories ~ sugars, data = UScereal, pch = 19, ylim = c(0, 325))
```



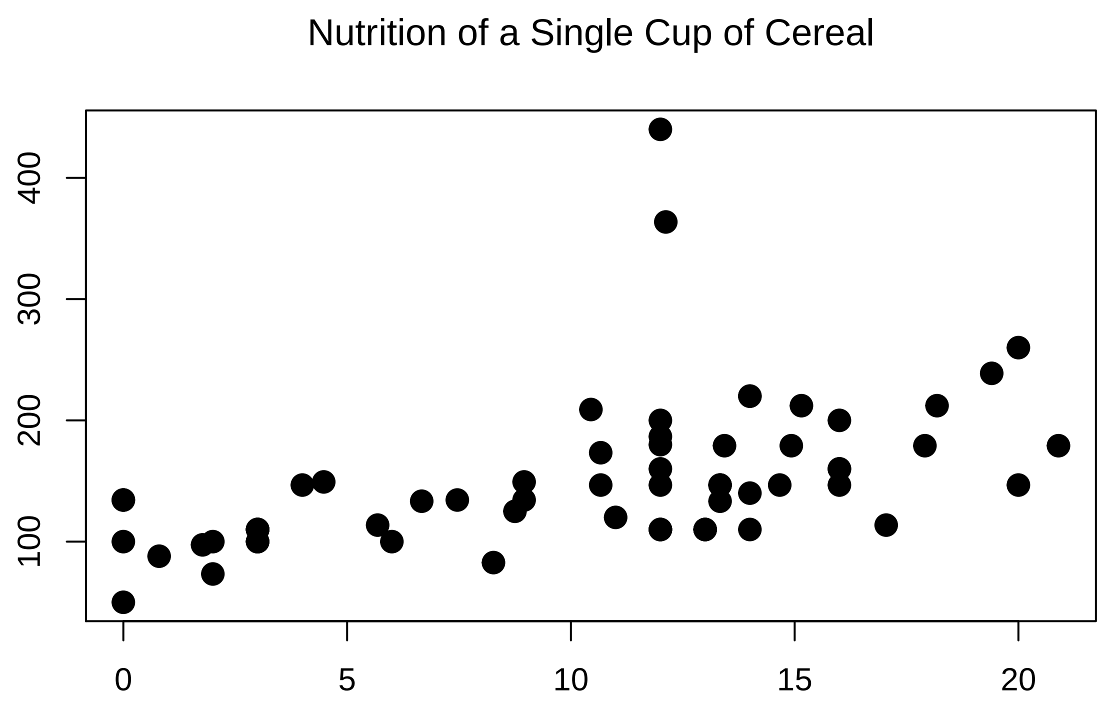
Size When changing the size of any element within the graph you will use a cex based argument. The cex is defaulted to 1, so that if you set the cex value to 1.5, the element will be 50% larger etc…

```
plot(calories ~ sugars, data = UScereal, pch = 19, ann = FALSE, cex = 1.5)
outliers <- UScereal[which(UScereal$calories > 300),]
text(outliers$sugars, outliers$calories - 15, labels = row.names(outliers), cex = .75)
title(main = 'Nutrition of a Single Cup of Cereal', ylab = 'Calories', xlab = 'Sugars (grams)', cex.mai
```



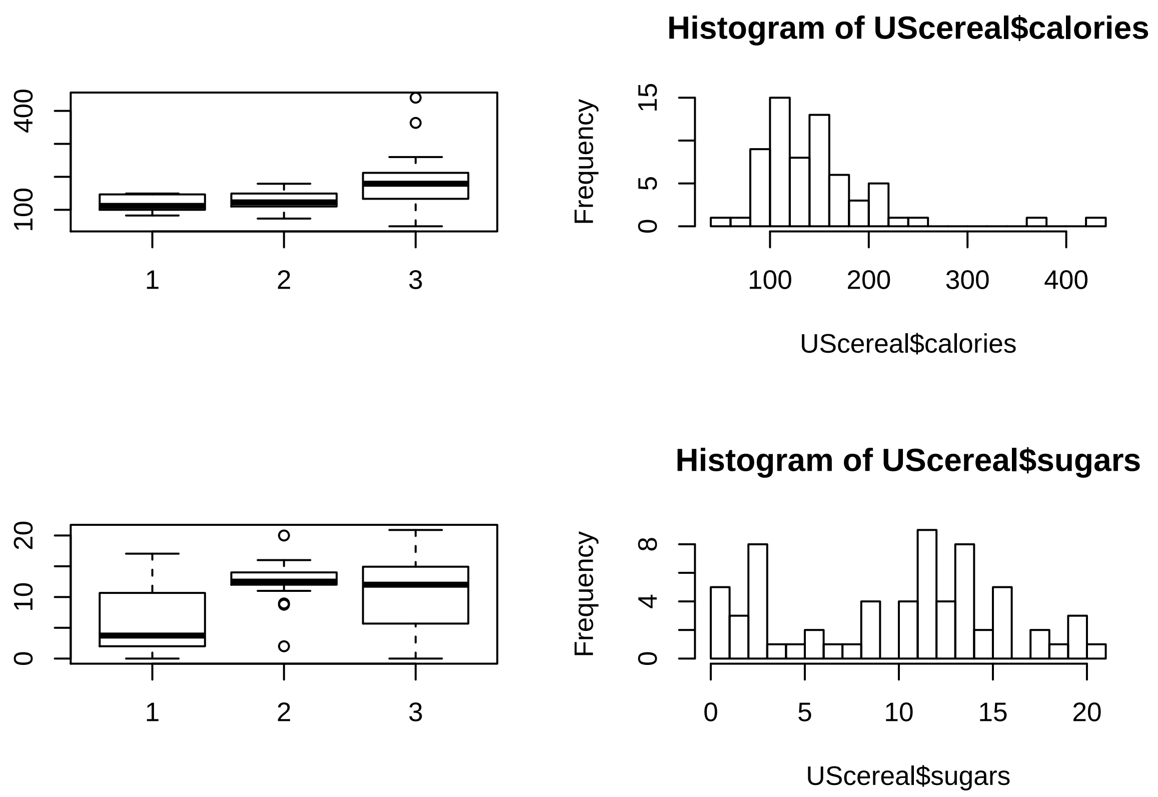
You can also play with font type, which takes on values from 1 to 5.

```
plot(calories ~ sugars, data = UScereal, pch = 19, ann = FALSE, cex = 1.5)
title(main = 'Nutrition of a Single Cup of Cereal', font.main = 1)
```



Combine Graphs into the Same Frame You can combine multiple graphs into the same frame, so that your graphs can appear right next to each other by using the *'par'* function.

```
par(mfrow = c(2, 2))
boxplot(calories ~ shelf, data = UScereal)
hist(UScereal$calories, breaks = 15)
boxplot(sugars ~ shelf, data = UScereal)
hist(UScereal$sugars, breaks = 15)
```



Just don't forget to reset the matrix when you are done.

```
par(mfrow = c(1, 1))
```

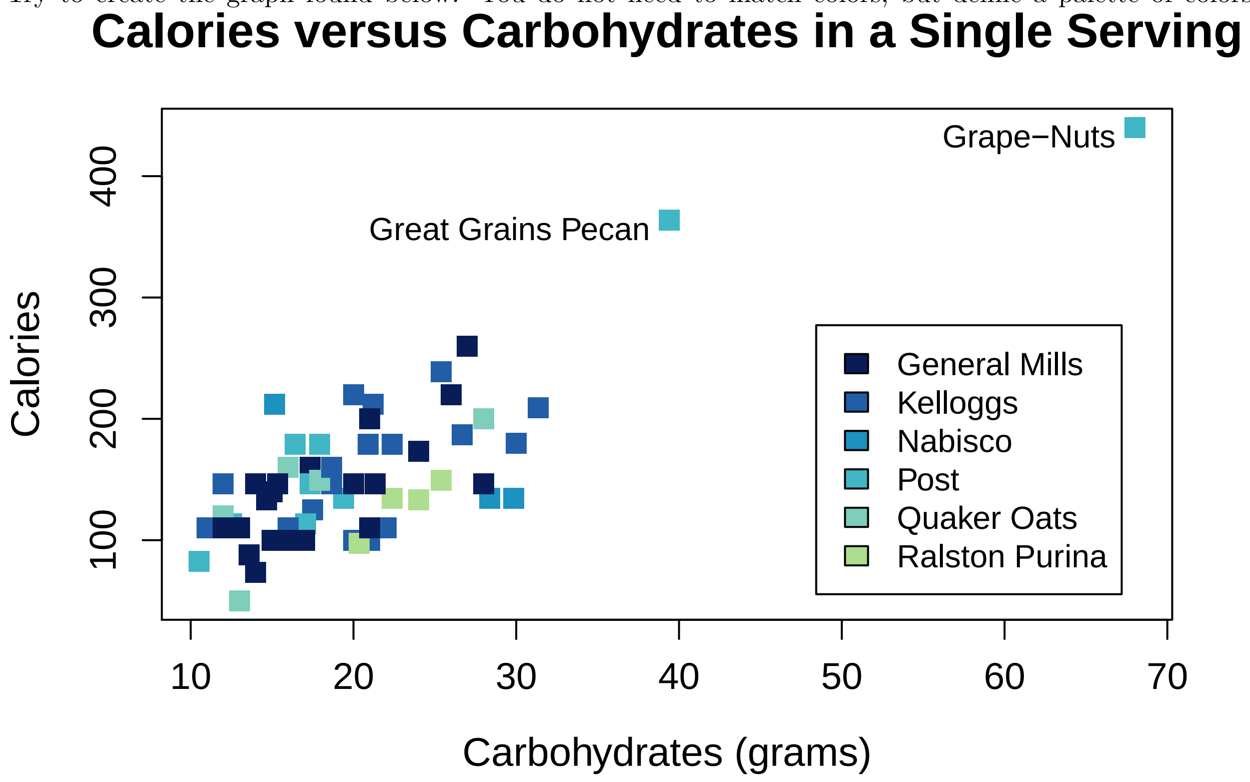
Margins

You can also change margins of the graph, the default is provided below (bottom, left, right, top).

```
par(mar = c(5.1, 4.1, 4.1, 2))
```

Exercise 1

Try to create the graph found below. You do not need to match colors, but define a palette of colors.



II. Advanced Graphics - ggplot2

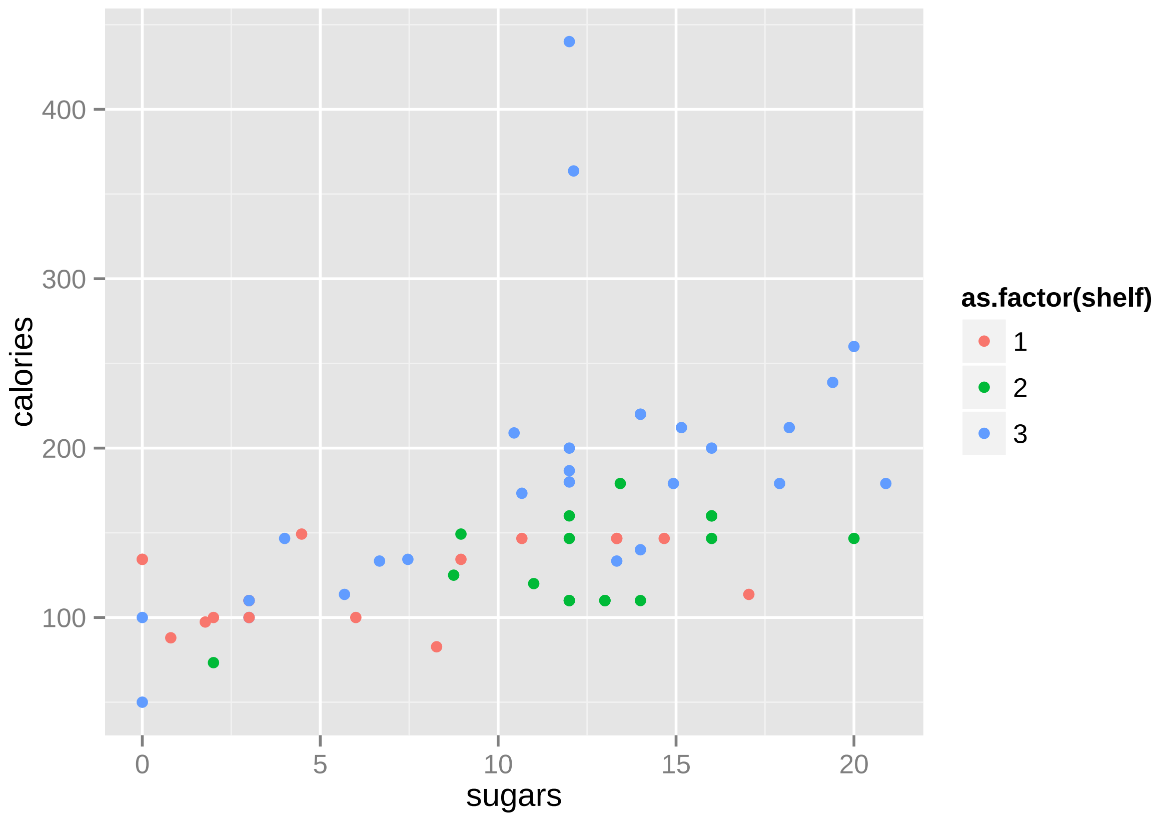
RStudio's ggplot2 cheat sheet is a great resource to keep on hand when you are getting started with ggplot2. Or you can use docs.ggplot2.org for more complete documentation on the functions. You can also check out the book, ggplot2: Elegant Graphics for Data Analysis by Hadley Wickham.

ggplot2 is a package based off of the Grammar of Graphics, a set of guidelines and rules to creating graphics. The way to create plots using ggplot2 starts with a geom to represent data points and then adding layers to change the aesthetics of the graph.

```
library(ggplot2)
```

The *'qplot'* function in ggplot2 creates a complete plot without adding layers but ultimately *'qplot'* has less functionality than the *'ggplot'* function with layers. Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

```
  qplot(x = sugars, y = calories, color = as.factor(shelf), data = UScereal)
```



A. The '*ggplot*' function

First you define your dataset and which variables you want to work with using the *'ggplot'* function. So for working with one variable, you would want to have one variable in the aesthetic function or if you are using two variables to make a graph, you can put two variables in the '*aes*' function etc...

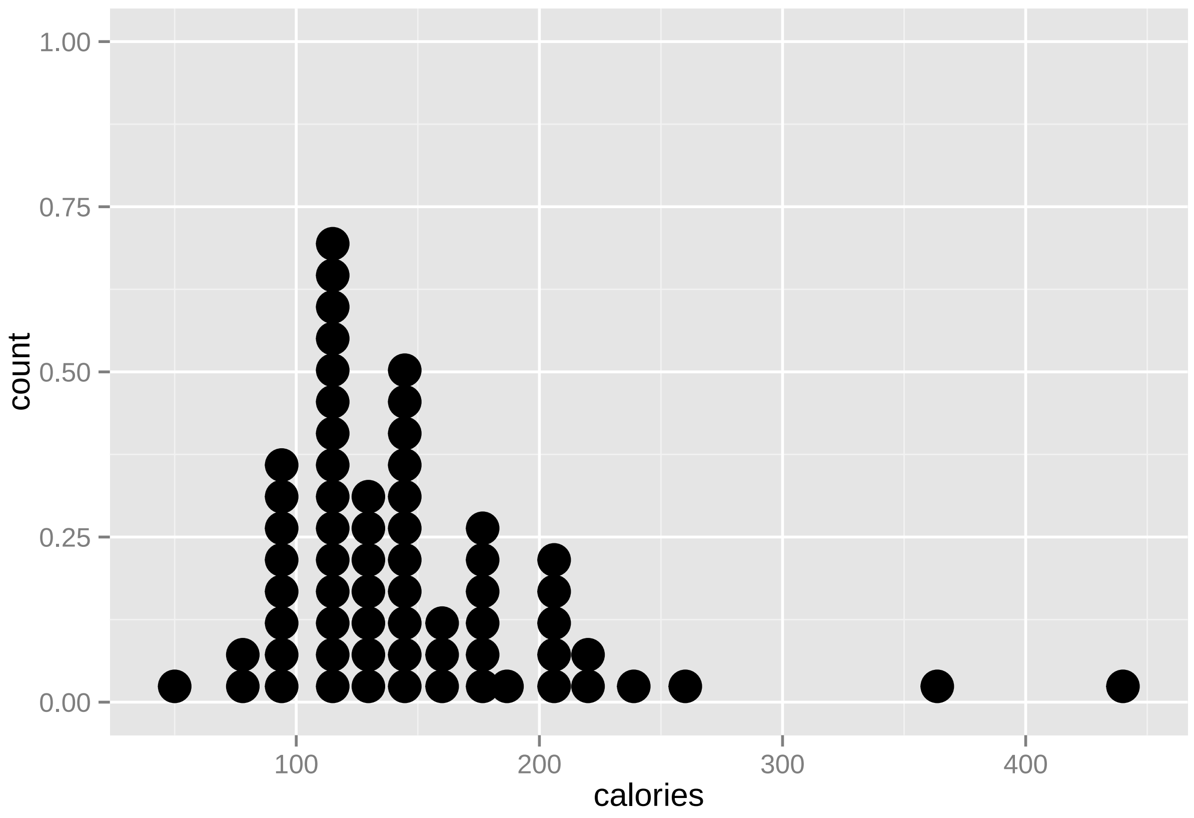
```
p1 <- ggplot(UScereal, aes(x = calories))

p2 <- ggplot(UScereal, aes(x = sugars, y = calories))
```

B. Layers

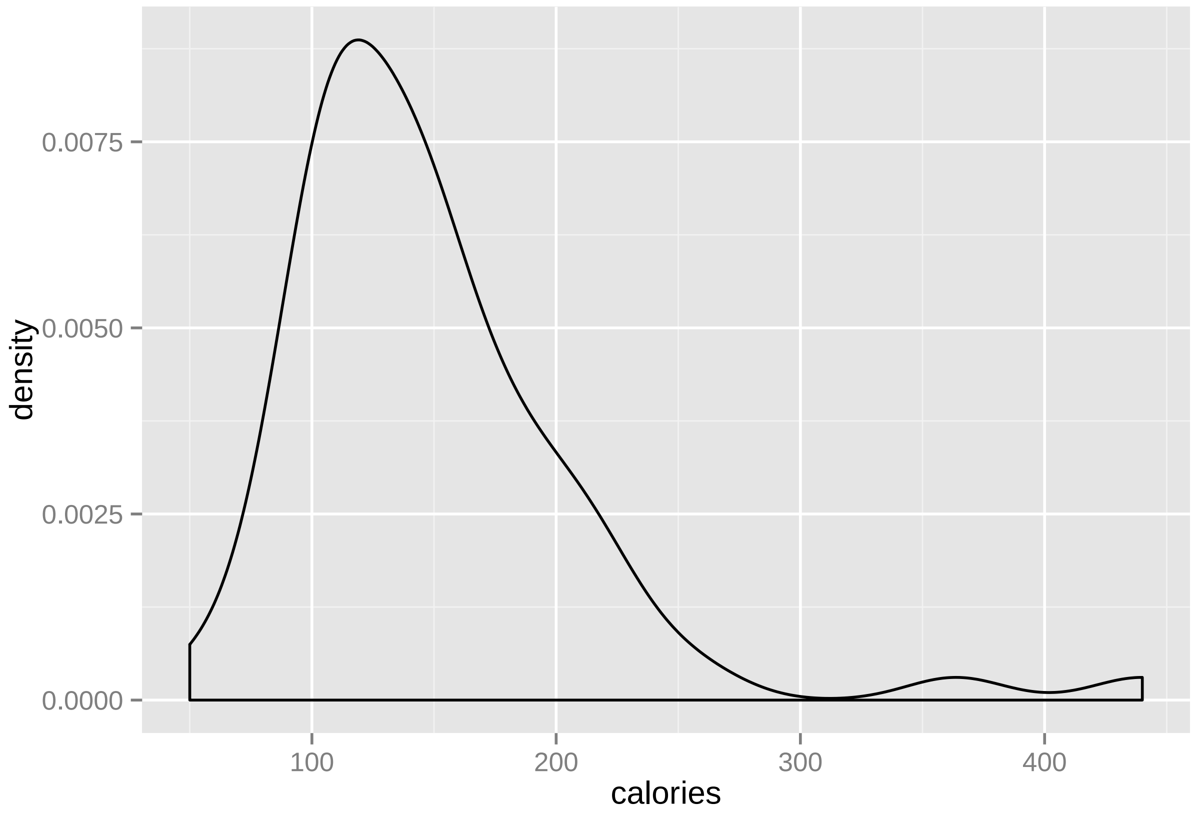
You can then apply different layers to create different graphics. You can use the same p1 object to create many different graphs, just by adding layers.

```
p1 + geom_dotplot()
```

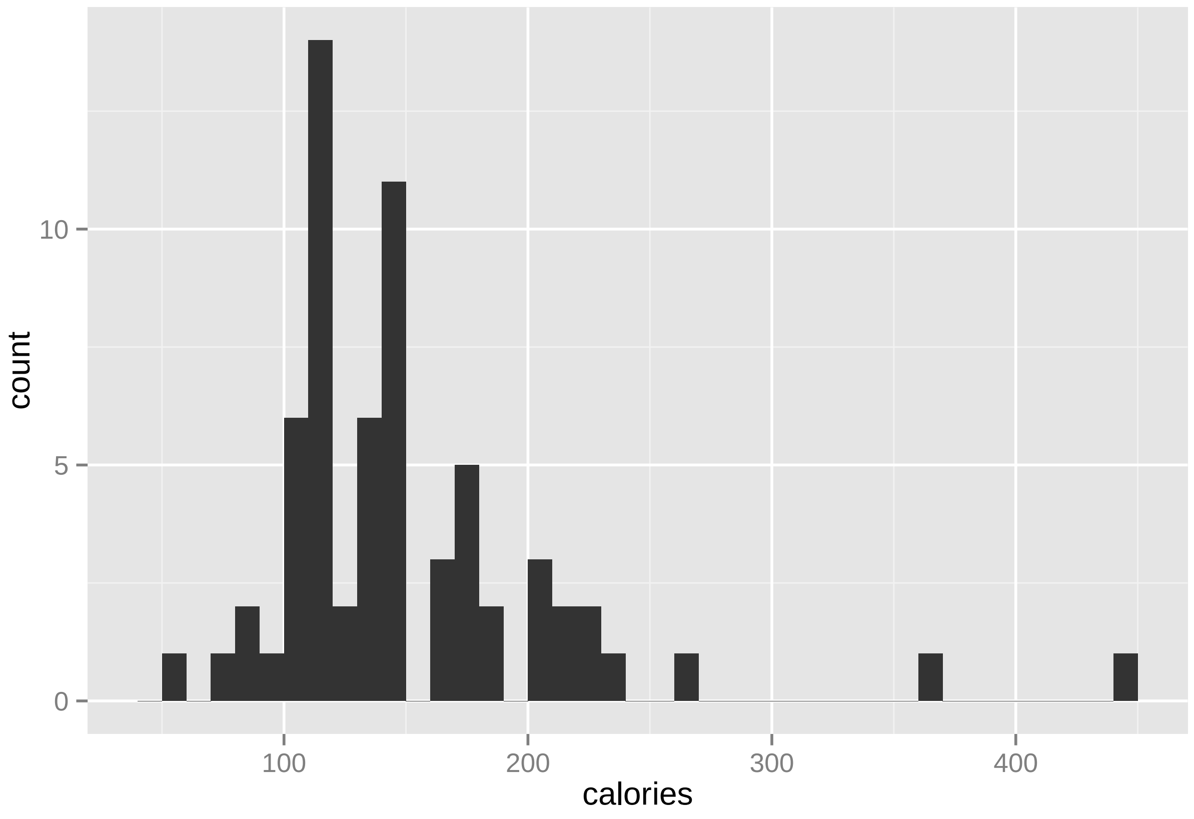
```
## stat_bindot: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



```
p1 + geom_density()
```

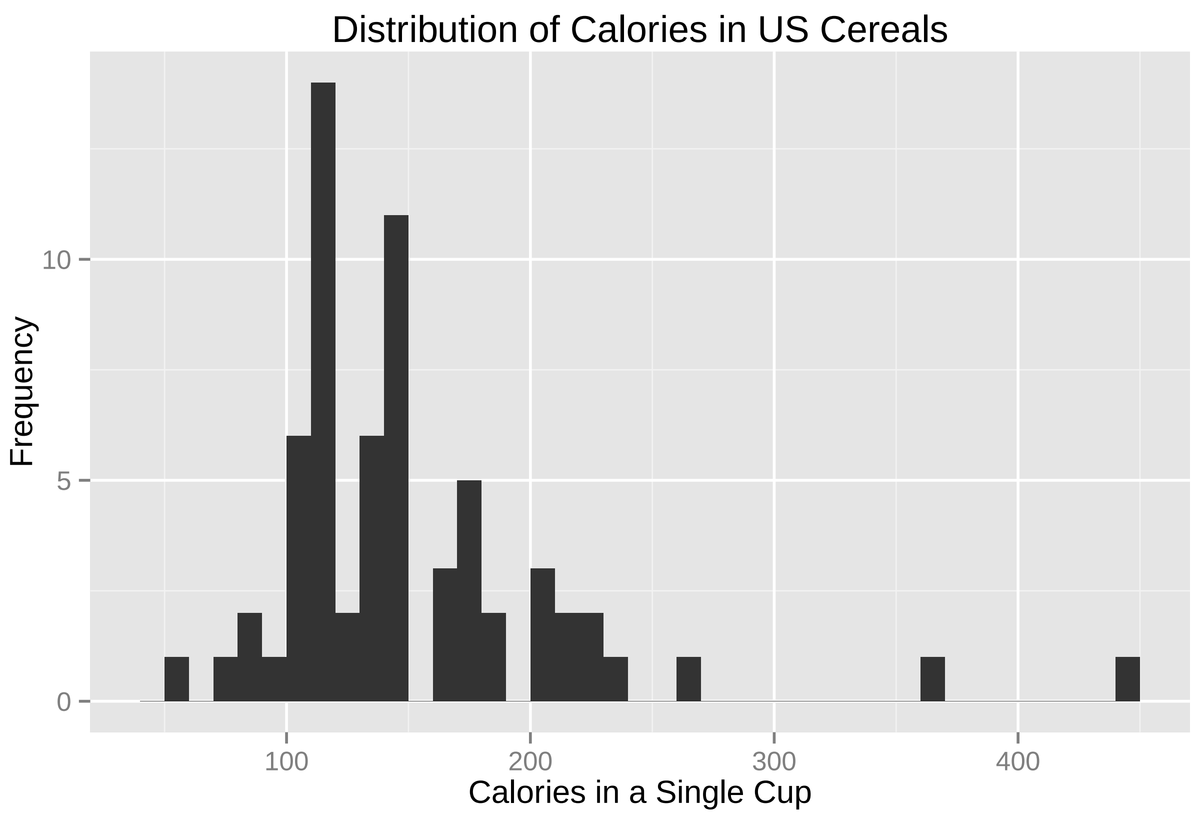


```
p1 + geom_histogram(binwidth = 10)
```

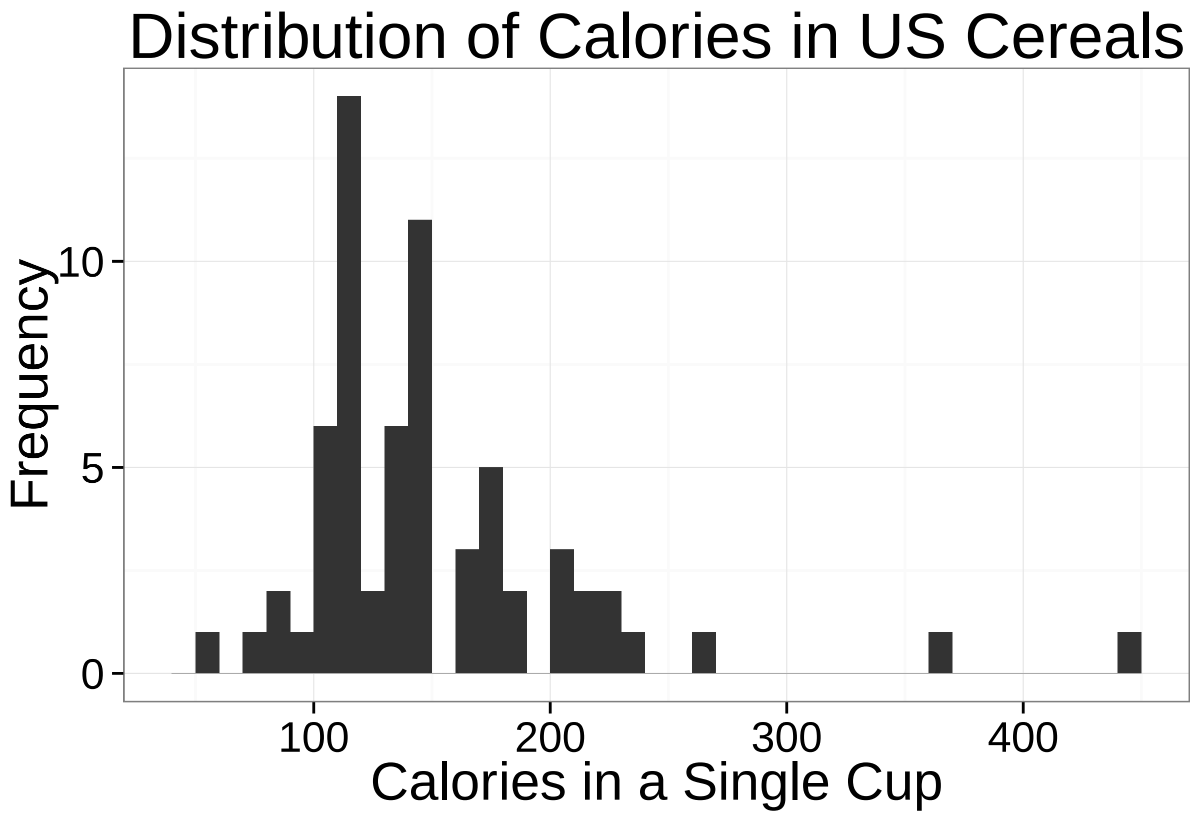


Or you can add multiple layers to the same graph. Such as adding titles or changing the size of elements or changing background colors etc…

```
p1 + geom_histogram(binwidth = 10) + ggtitle('Distribution of Calories in US Cereals') +
  xlab('Calories in a Single Cup') + ylab('Frequency')
```



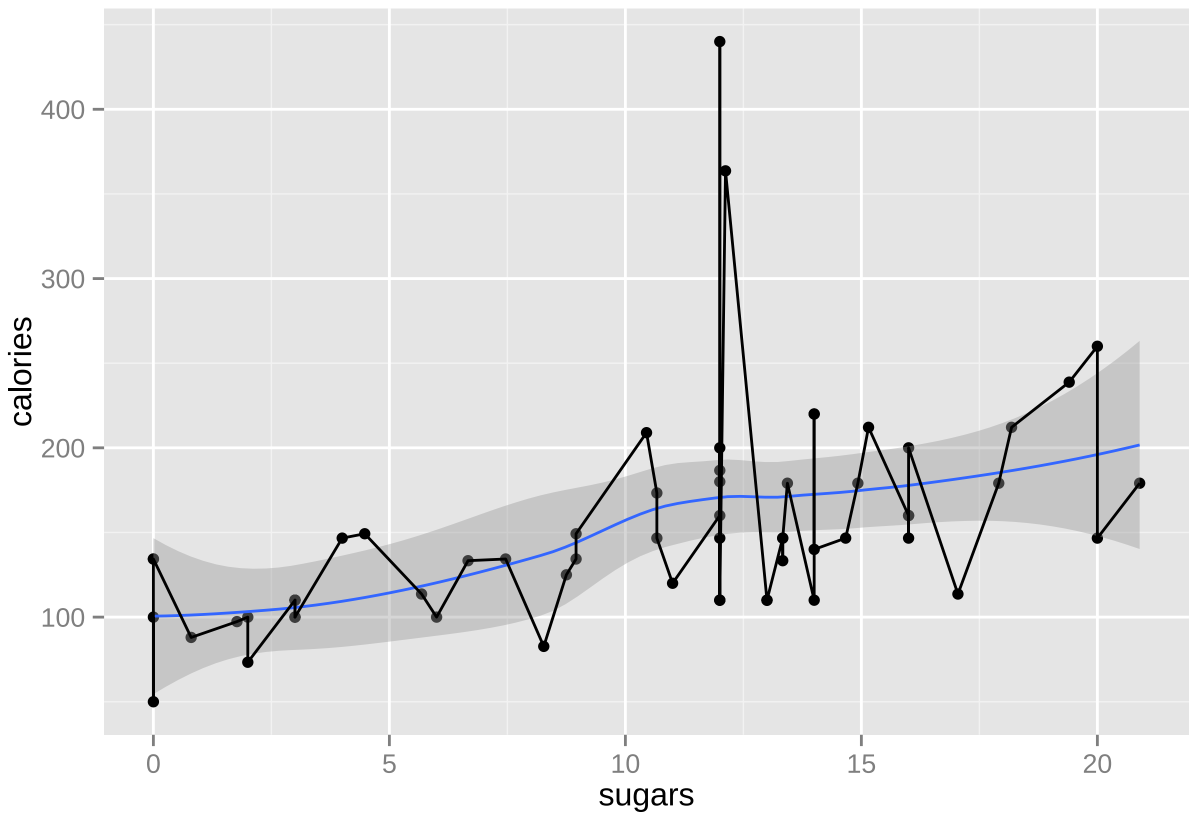
```
p1 + geom_histogram(binwidth = 10) + ggtitle('Distribution of Calories in US Cereals') +
  xlab('Calories in a Single Cup') + ylab('Frequency') + theme_bw() + theme(text = element_text(size = 
```



You can even add multipe '*geom_*' functions to the same ggplot object.

```
p2  + geom_point() + geom_smooth() + geom_line()
```

```
## geom_smooth: method="auto" and size of largest group is <1000, so using loess. Use 'method = x' to cl
```

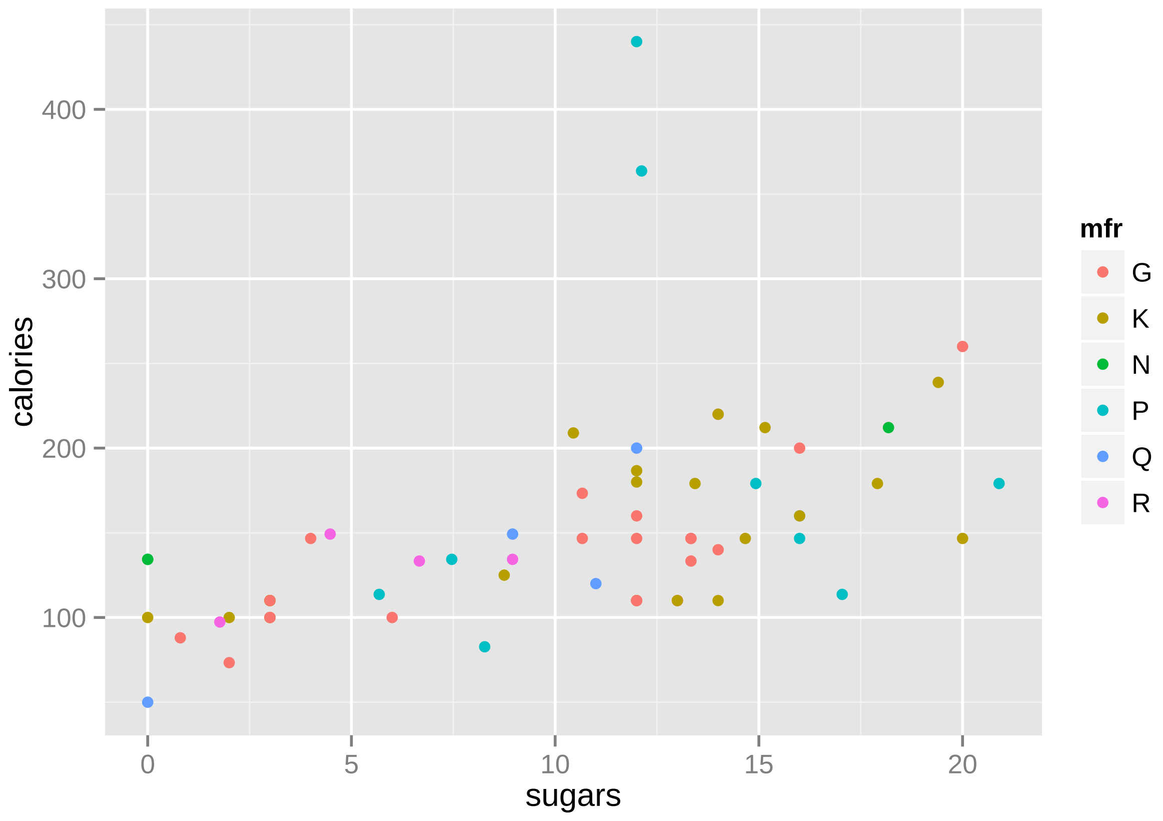


C. Aesthetics

Aesthetics can change either in the ggplot function or inside of a geom_function. Aesthetics can include x position, y position, size of elements, shape of elements and color of elements. An element is a geometric shape suchs as points, lines, line segments, bars or text. Each geometric shape has their own aesthetics. Ex. Points have their own shape and size.

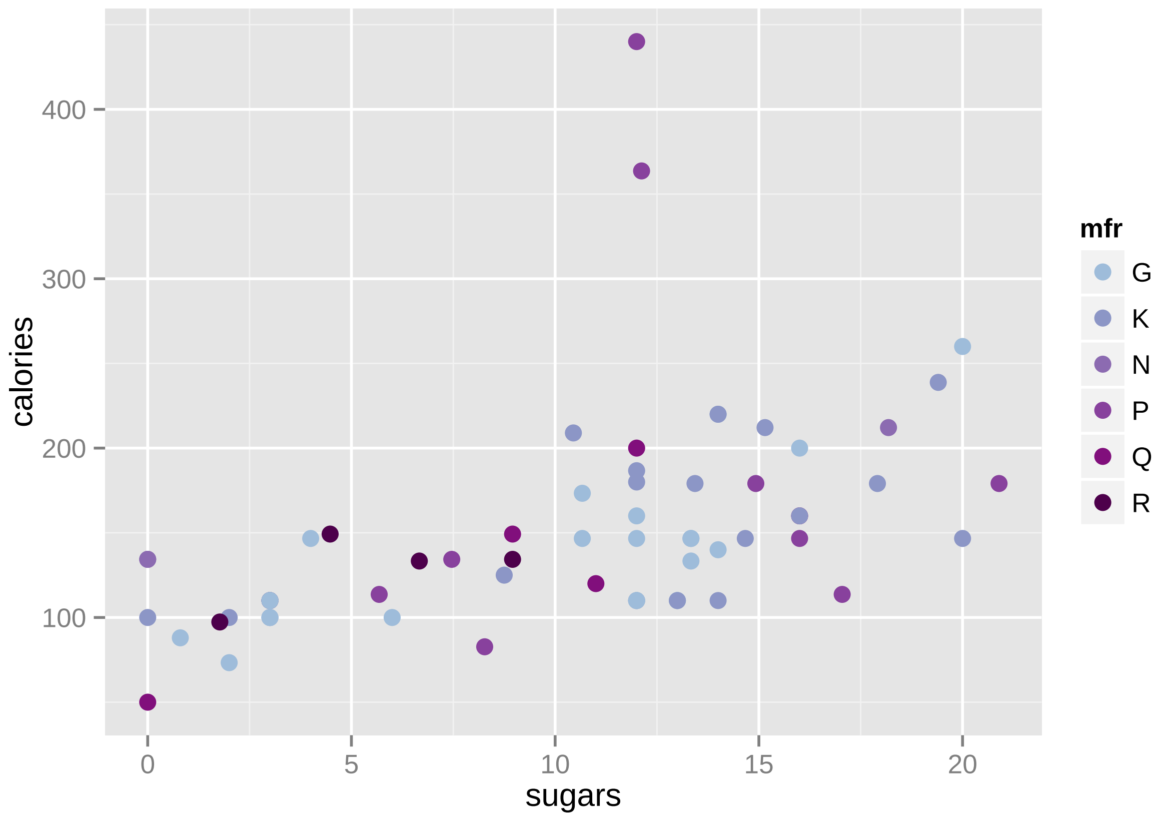
Color We can set up colors in both the '*ggplot*' function or the '*geom_functions*'.

```
p2 <- ggplot(UScereal, aes(x = sugars, y = calories, color = mfr))

p2 + geom_point()
```

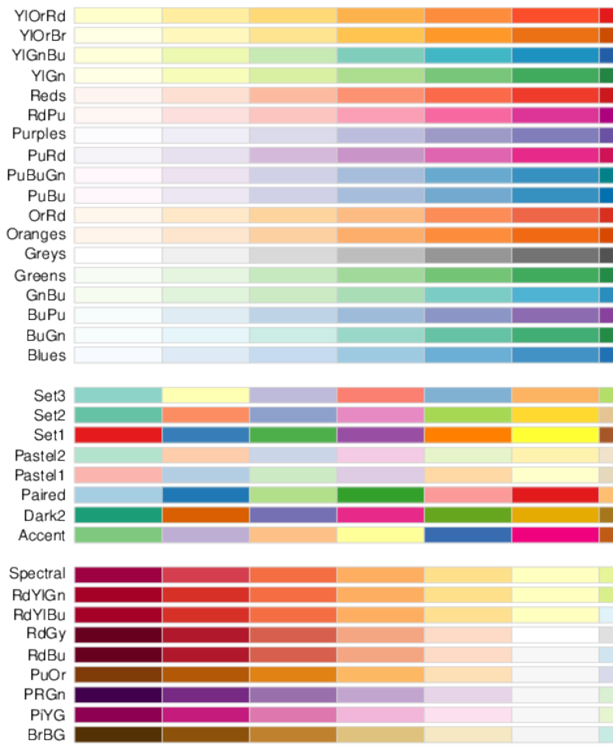


We can still define a palette of colors manually. Functions that start with '*scale_*' control how a plot maps data values to the visual values of an aestheti.

```
my_colors <- c('#9ebcda', '#8c96c6', '#8c6bb1', '#88419d', '#810f7c', '#4d004b')

p2 + geom_point(size = 3) + scale_color_manual(values = my_colors)
```

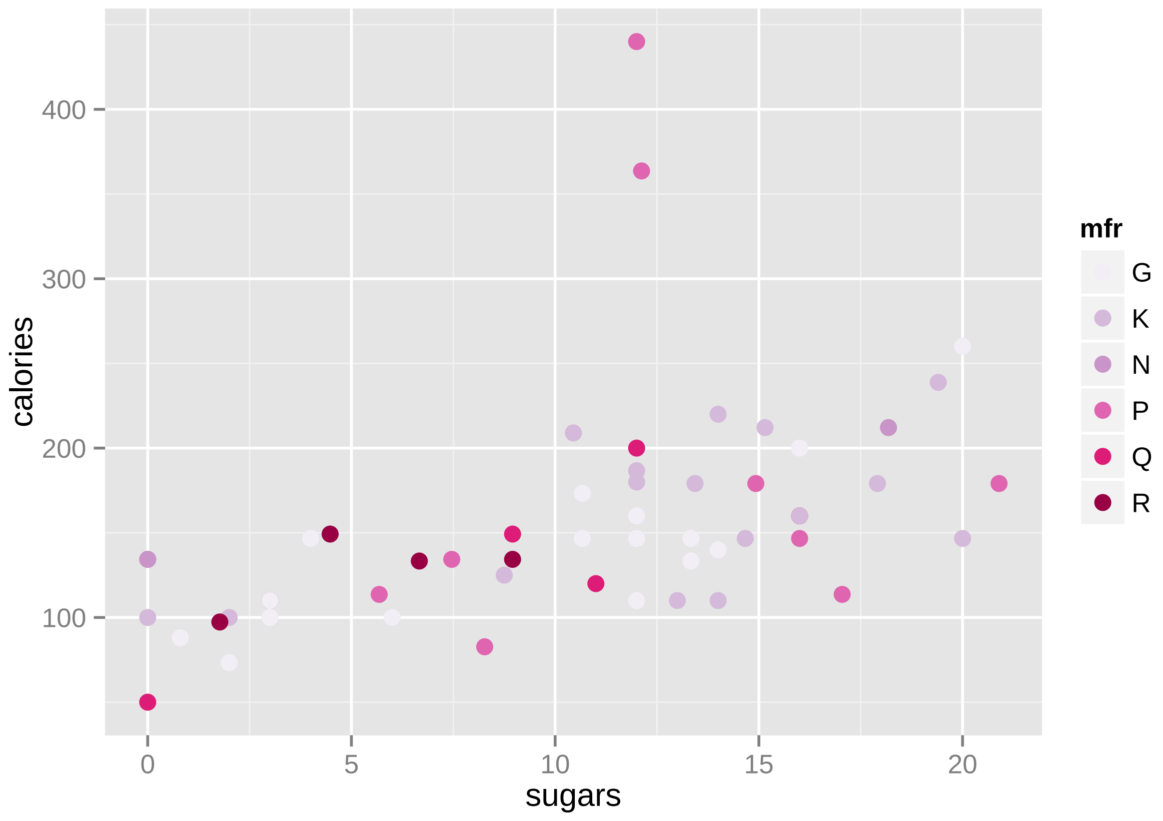


calories
400
300
200
100
0
5
10
15
20
sugars
mfr
G
K
N
P
Q
R



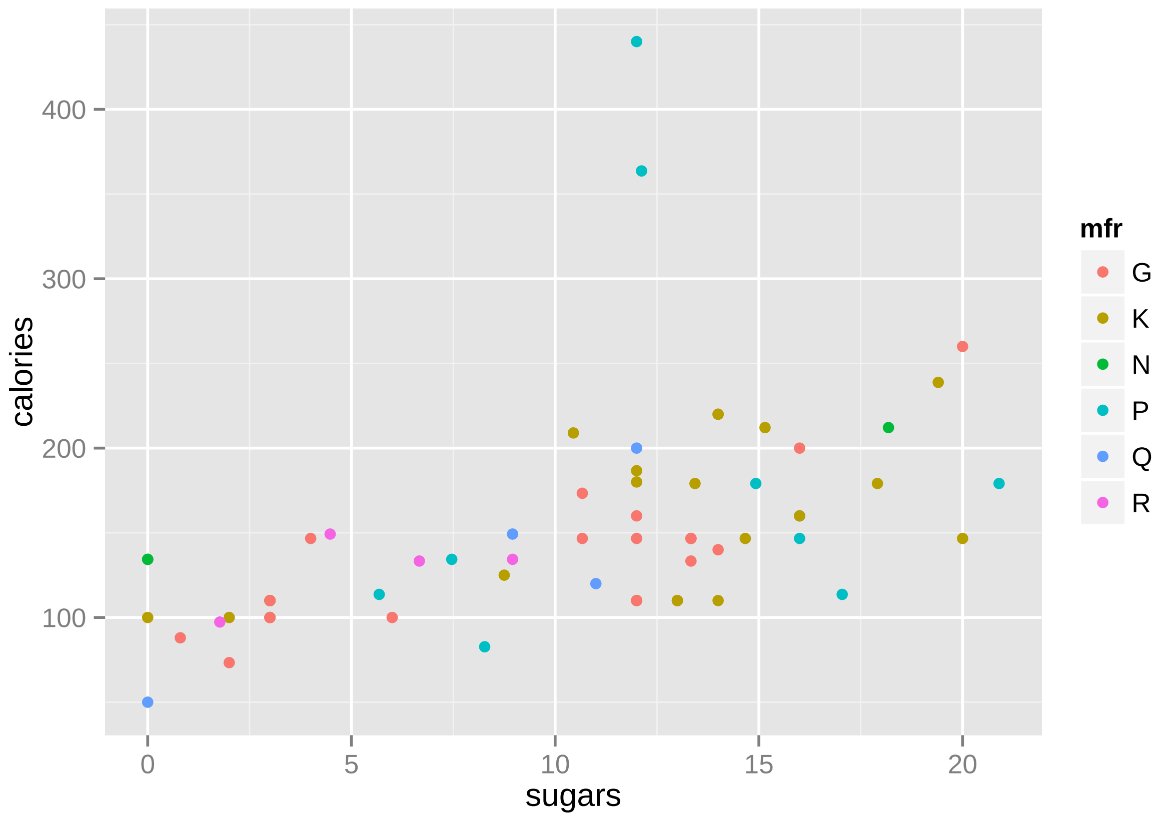
Or can also use predefined palettes from the '*RColorBrewer*' package.
Source: http://www.lgbmi.com/wordpress/wp-content/uploads/2012/08/colorbrewer_names.png

```
library(RColorBrewer)
p2 + geom_point(size = 3) + scale_color_brewer(palette = 'PuRd')
```



Here we map the color inside the geom_point function.

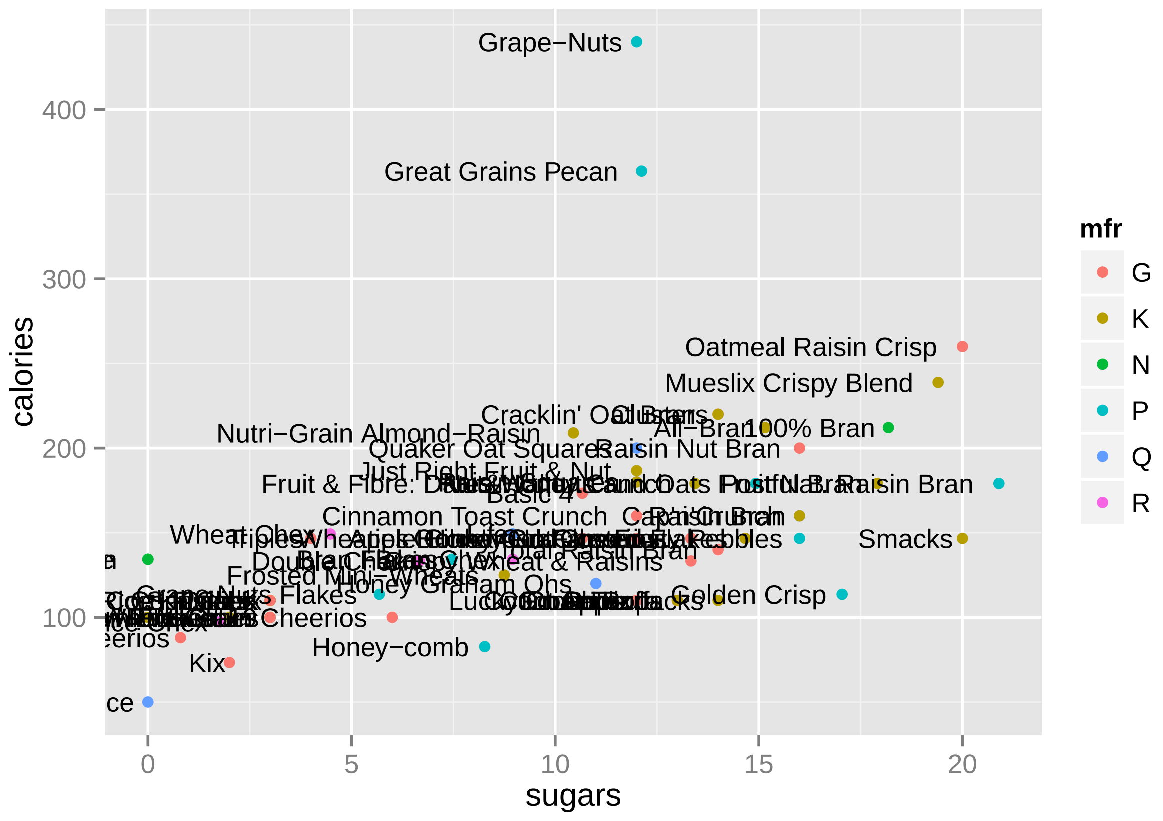
```r
p2 <- ggplot(UScereal, aes(x = sugars, y = calories))

p2 + geom_point(aes(color = mfr))
```



Point Labels

We can add labels to points using the '*geom_text*' function. Notice that the text is black - if we wanted the text the same color as the point, we may want to map the color aesthetic inside of the ggplot funciton as shown previously. The '*scale_size()*' removes the legend created by '*geom_text*'

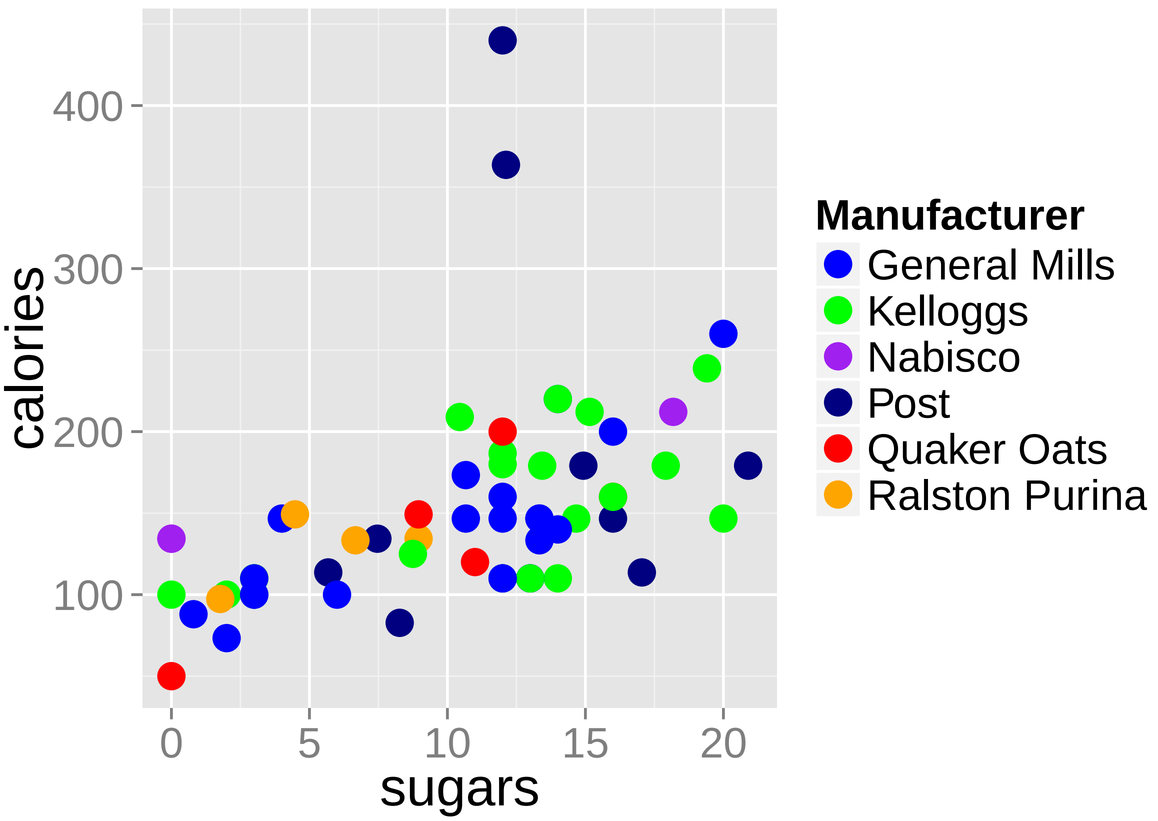
```
p2 + geom_point(aes(color = mfr)) + geom_text(aes(label = row.names(UScereal), size = .5), hjust = 1.1)
```



Edit the legend

We can adjust the legen label by using '*labs()*' and we can adjust the colors and labels in the legend by using the '*scale_color_manual*' function.

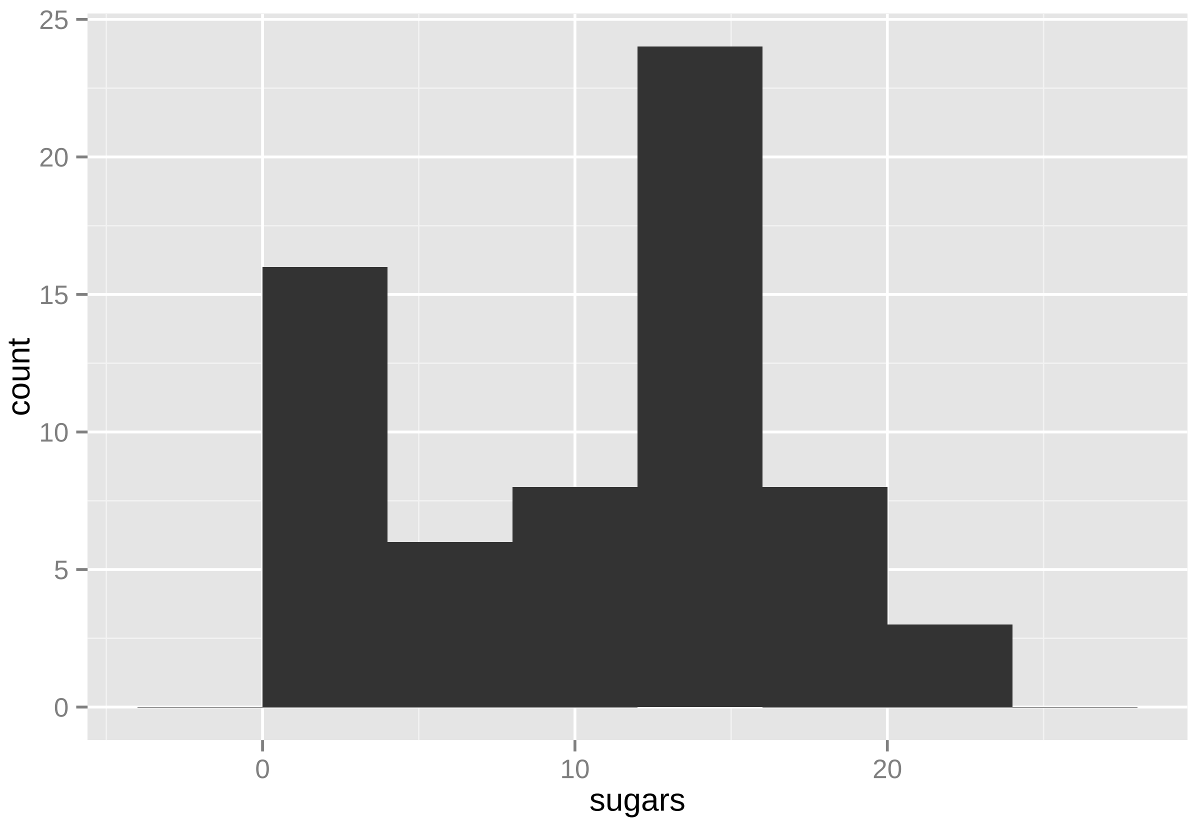
```
p2 + geom_point(aes(color = mfr), size = 5) + labs(color = 'Manufacturer') +
  scale_color_manual(values = c('blue', 'green', 'purple', 'navyblue', 'red', 'orange'),
                     labels = c('General Mills', 'Kelloggs', 'Nabisco', 'Post', 'Quaker Oats', 'Ralston
  theme(text = element_text(size = 20))
```



D. Faceting

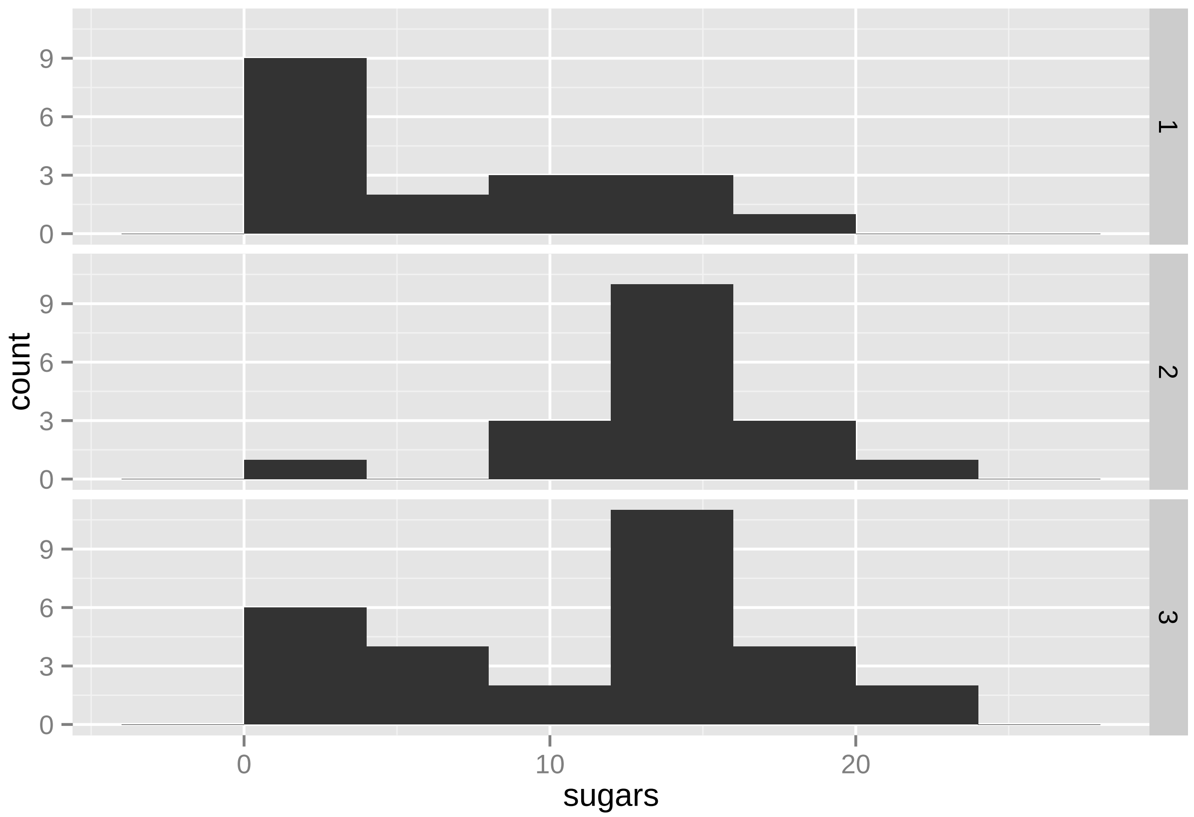
Divide a plot into subplots based on the valuesof one or more discrete variables.

```
p3 <- ggplot(UScereal, aes(x = sugars))
p3 + geom_histogram(binwidth = 4)
```

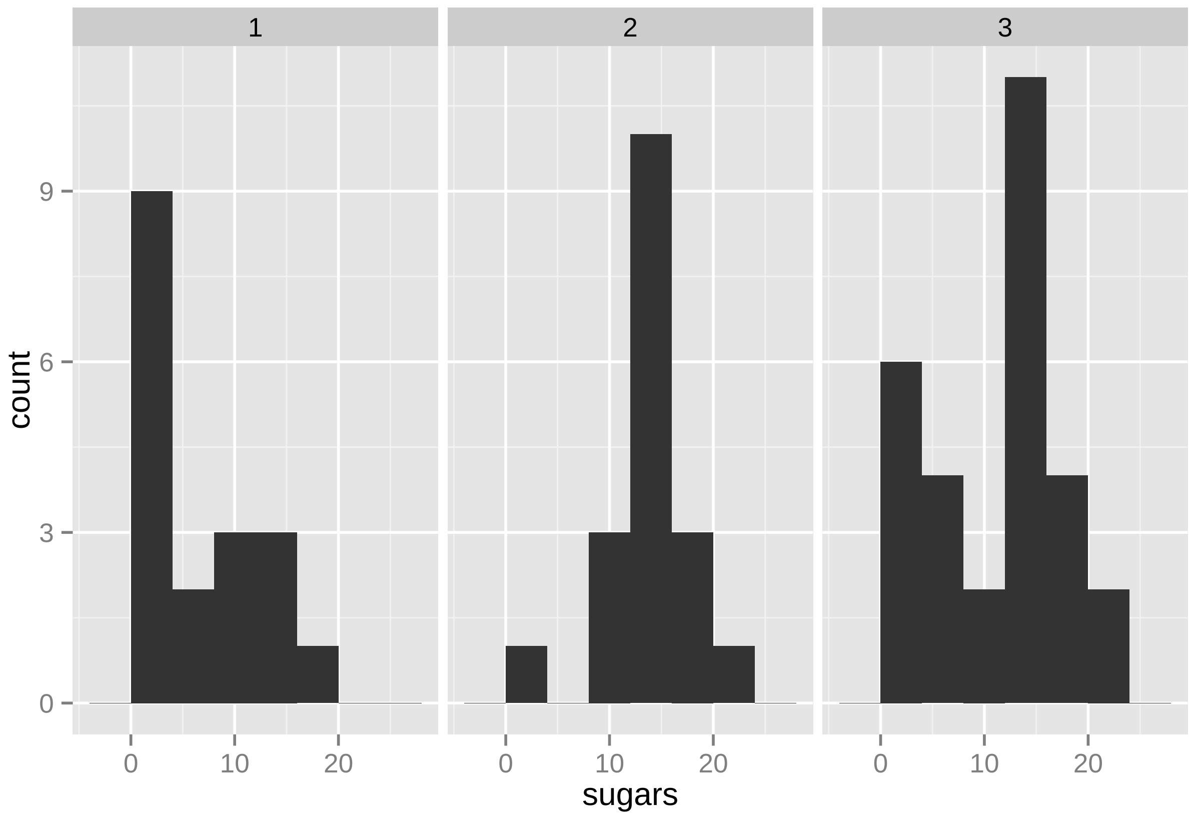


You can have the categories displayed vertically or horizontally, it depends which looks better for your data.

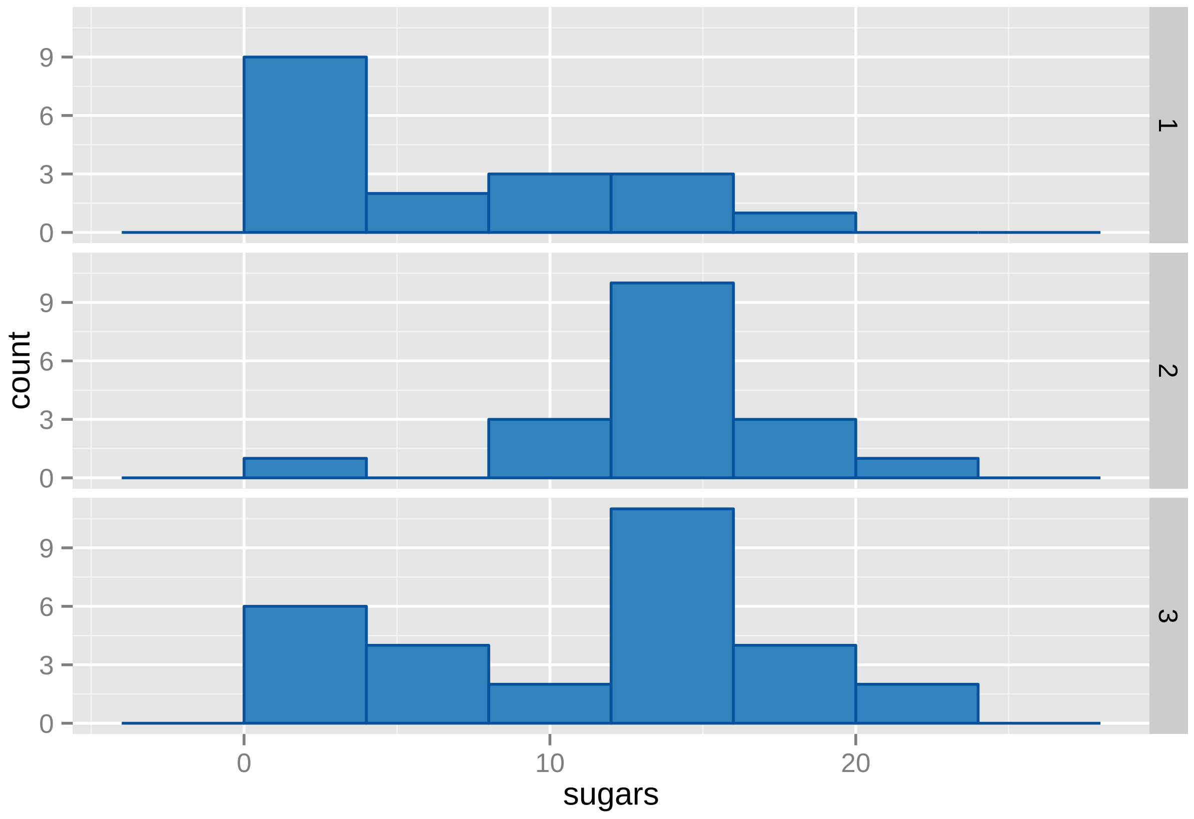
```
p3 + geom_histogram(binwidth = 4) + facet_grid(shelf ~ .)
```



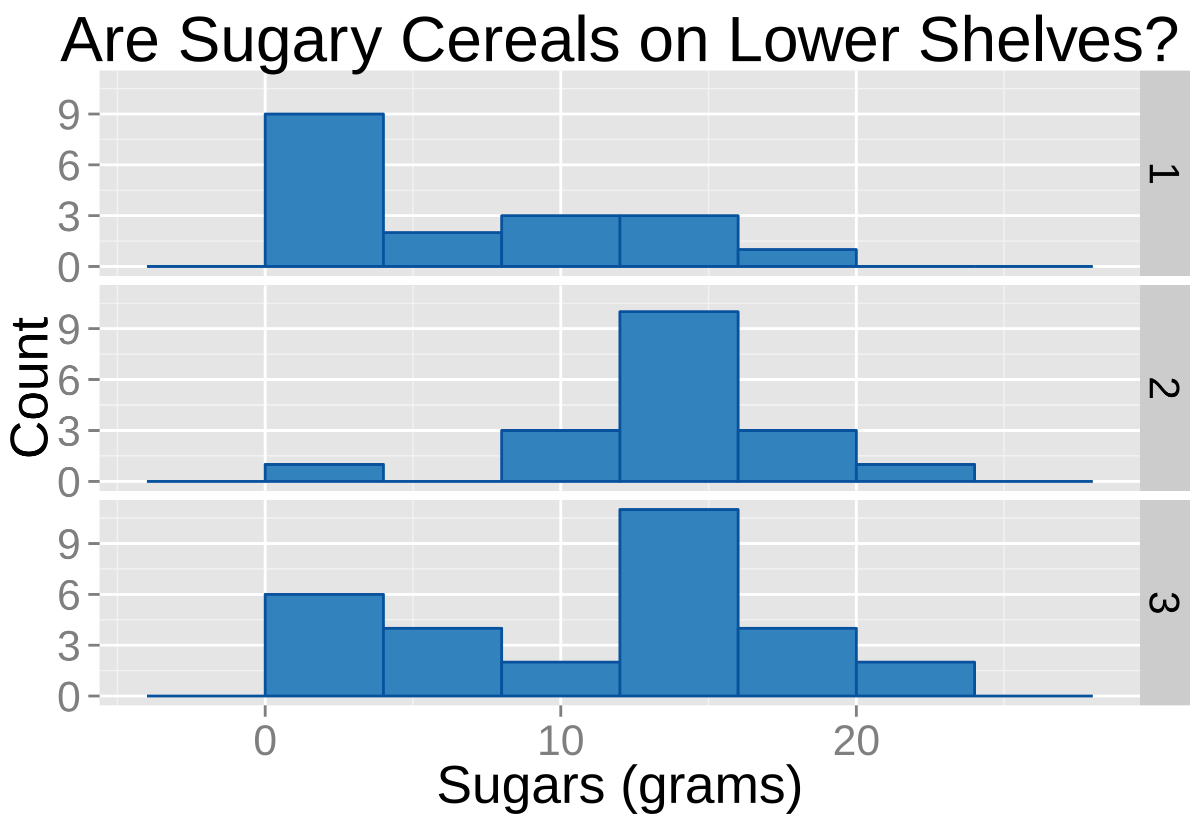
```
p3 + geom_histogram(binwidth = 4) + facet_grid(. ~ shelf)
```



```
p3 + geom_histogram(fill = '#3182bd', color = '#08519c', binwidth = 4) + facet_grid(shelf ~ .)
```

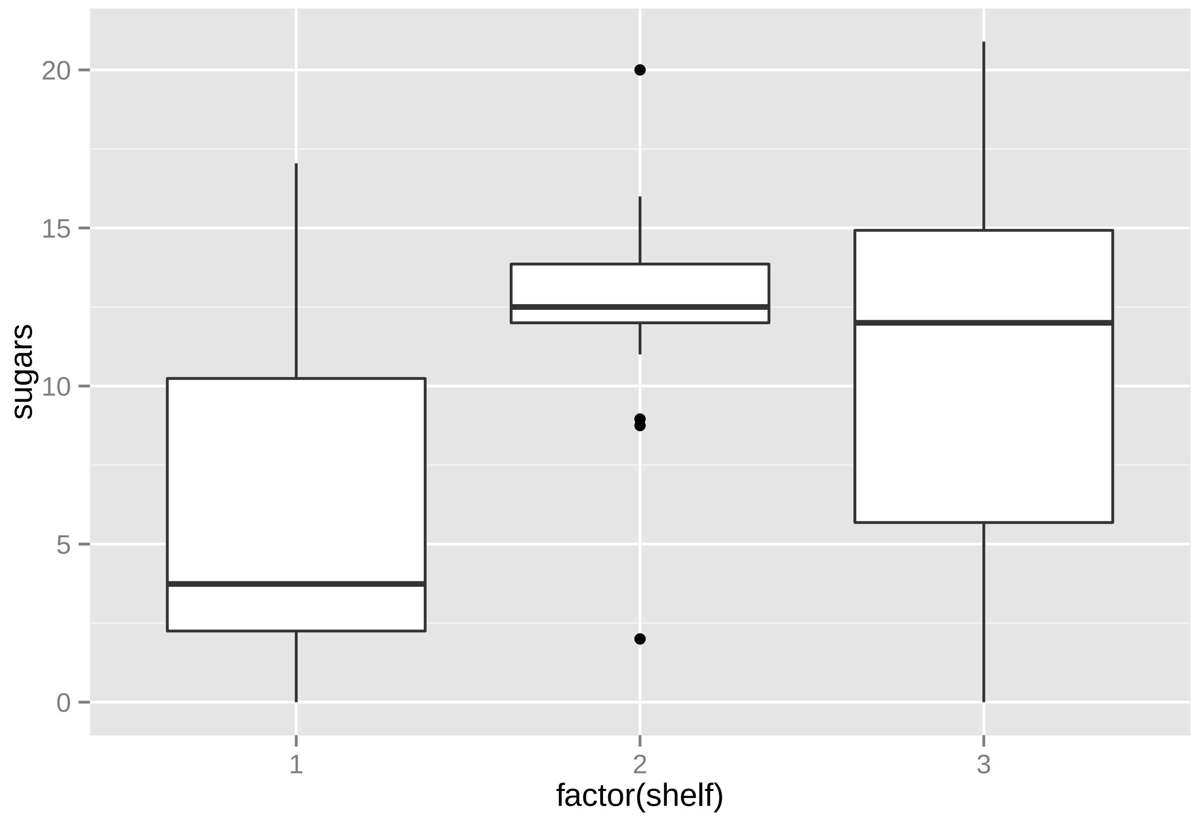


```
p3 + geom_histogram(fill = '#3182bd', color = '#08519c', binwidth = 4) + facet_grid(shelf ~ .) + theme(
  labs(title = 'Are Sugary Cereals on Lower Shelves?',
       x = 'Sugars (grams)', y = 'Count')
```

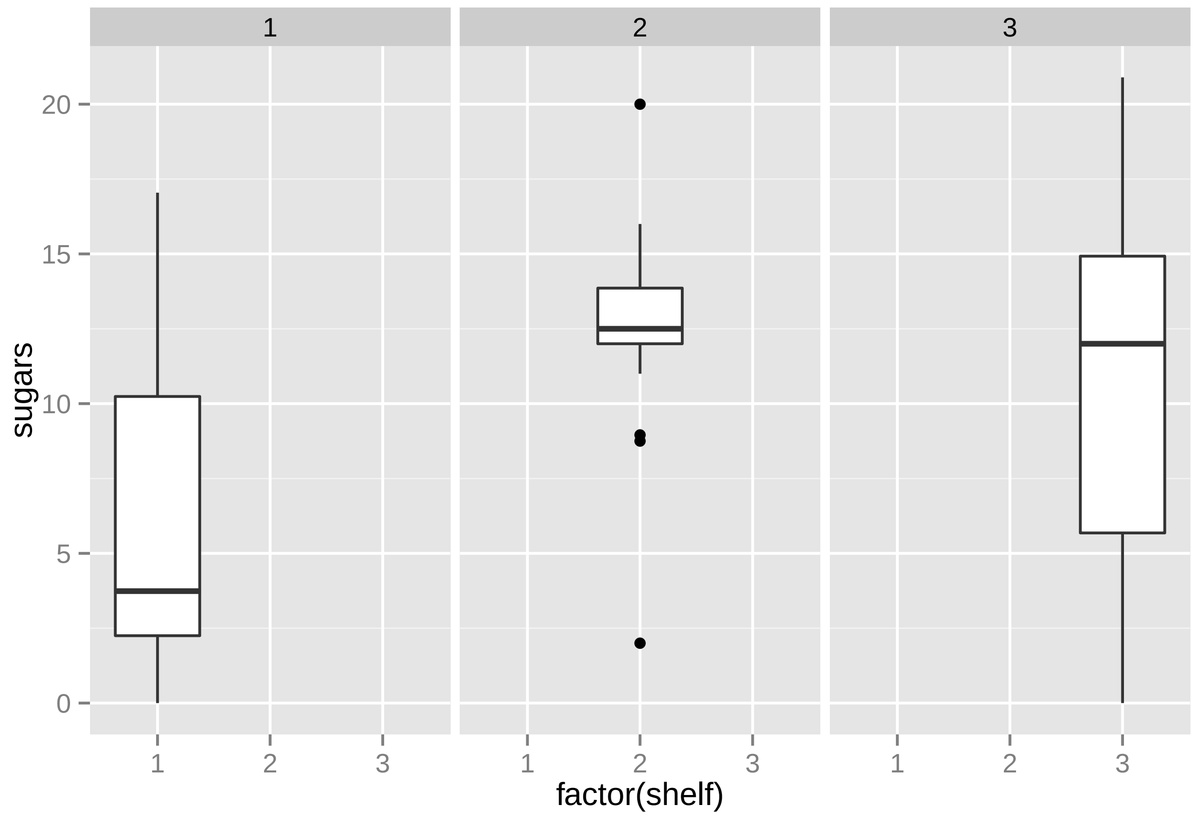


There are also some cases where it may not make sense to use faceting:

```
p4 <- ggplot(UScereal, aes(factor(shelf), sugars))
p4 + geom_boxplot()
```

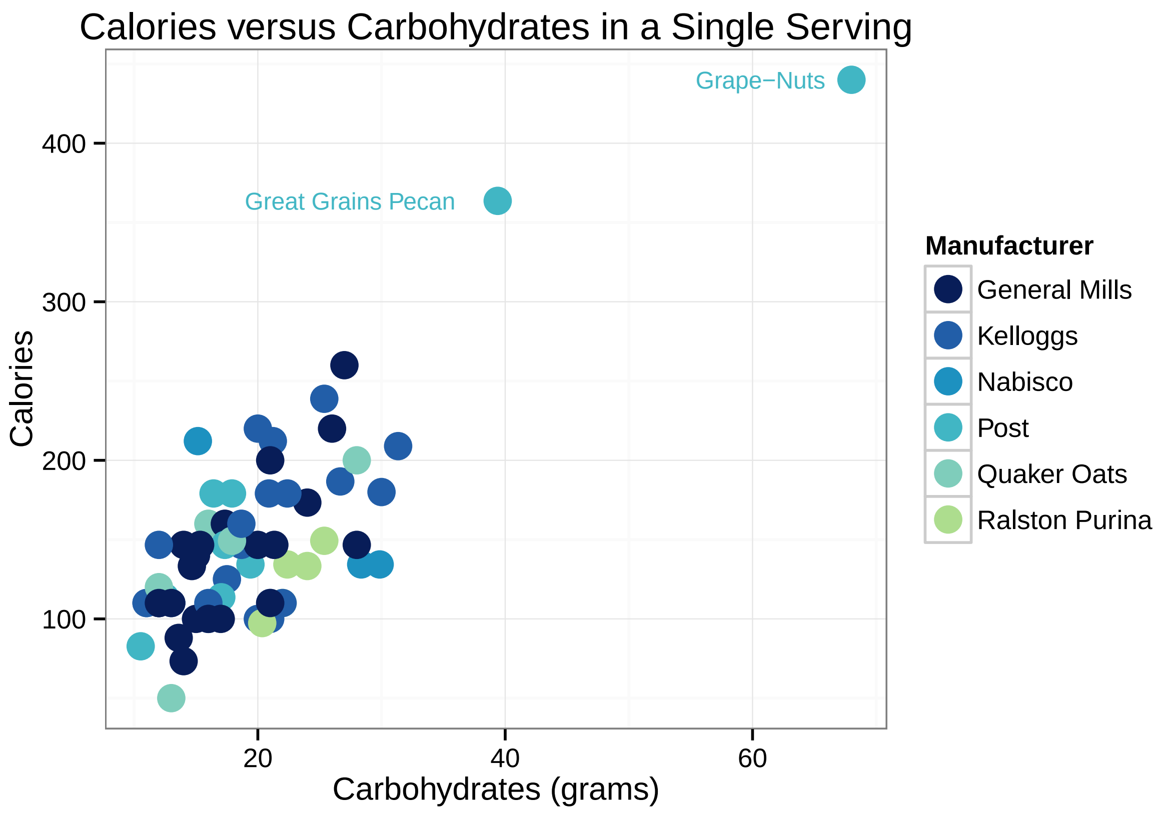


```
p4 + geom_boxplot() + facet_grid(. ~ shelf)
```



Exercise 2

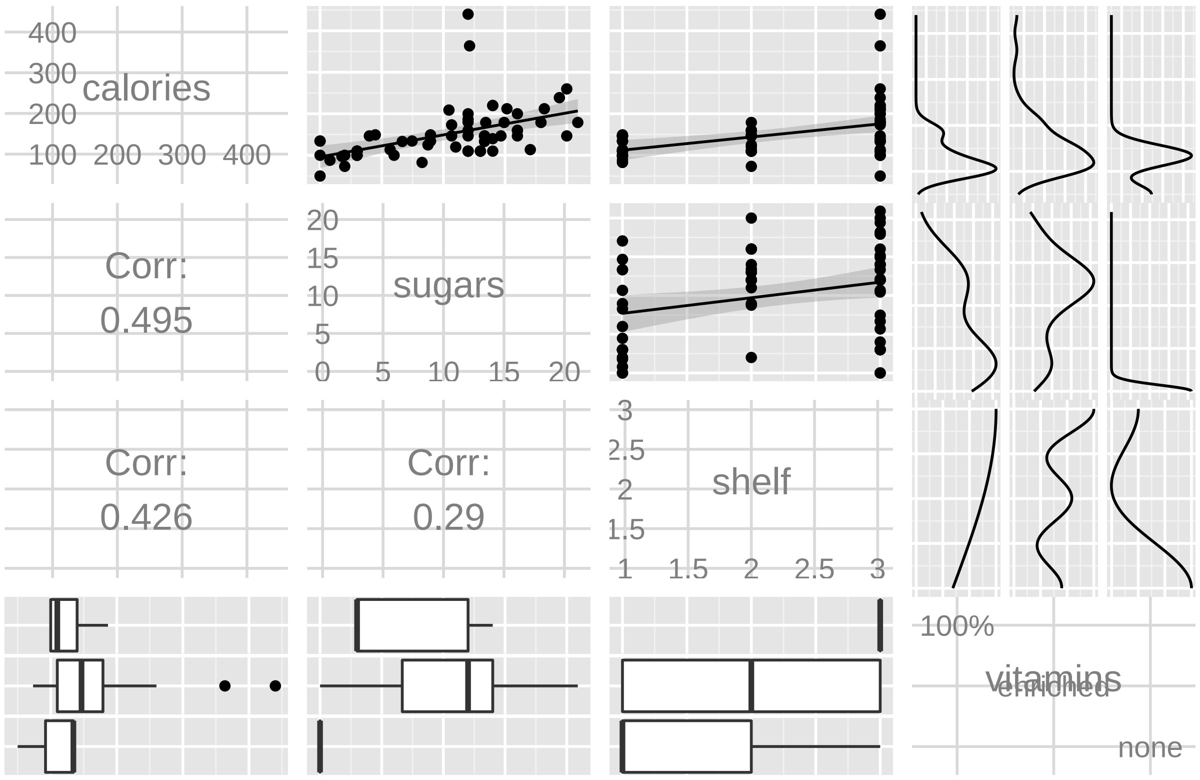
Try to create the graph found below. You do not need to match colors, but define a palette of colors.



III. Other Graphs

Scatterplot matrix using the '*GGally*' package. This is more than just a scatterplot matrix though, because we can have different graphs on each side of the diagonal and we can

```
library(GGally)
ggpairs(UScereal[, c(2, 8, 9, 11)],
        upper = list(continuous = 'smooth', combo = 'facetdensity', discrete = 'blank') ,
        lower = list(continuous = 'cor', combo = 'box'))
```


calories
400
300
200
100 200 300 400
sugars
20
15
10
5
0 5 10 15 20
Corr:
0.495
Corr:
0.426
Corr:
0.29
shelf
3
2.5
2
1.5
1 1.5 2 2.5 3
100%
vitamins
enriched
none