Practice1

Prakhar

03/04/2021

## GGPlot

Every ggplot2 plot has 3 key components - Data - Aesthetic Mappings between variables in the data and visual properties - Geometrics Layer (geom) that describes what visual elements are used to visually render the data. In terms of the other layers, ggplot2 uses sensible default settings, which we will learn to override and customize

## Scaterplot

A scatterplot is simple graph in which the values of two variables are plotted along two axes. The resulting visualisation may reveal a correlation between the variables. In this session you will learn how to put together a a scatter plot like the one shown here.

data1 <- read.csv("country\_data.csv")  
# str gives the type of the data container (a data.frame), the variables and types  
str(data1)

## 'data.frame': 44 obs. of 3 variables:  
## $ var\_a : num 10.1 7.9 13 9 11 14 6 4 12 7 ...  
## $ var\_b : num 9.14 8.14 8.74 8.77 9.26 8.1 6.13 3.1 9.13 7.26 ...  
## $ country: chr "ireland" "ireland" "ireland" "ireland" ...

# head shows the first few rows of the data  
head(data1)

## var\_a var\_b country  
## 1 10.1 9.14 ireland  
## 2 7.9 8.14 ireland  
## 3 13.0 8.74 ireland  
## 4 9.0 8.77 ireland  
## 5 11.0 9.26 ireland  
## 6 14.0 8.10 ireland

### Defining aesthetic mapping

The aesthetic mapping for the first version of our scatter plot will map the values of var\_a to the scale of the x axis and the values of var\_b to the scale of the y-axis. We will ignore country for now. The aes() function is used to specify the X and Y axes.

# we first tell R that we will use functions from the ggplot2 package  
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.0.4

# the ggplot function takes the data and aesthetic mapping function aes as arguments  
g <-ggplot(data1, aes(x=var\_a, y=var\_b))  
  
g



### Adding a geometrics layer

A blank ggplot is drawn. Even though the x and y are specified, there are no points or lines in it. This is because, we have only created the first two layers - data and aesthetic mapping. We now need to tell ggplot the types of geometrics we want to use to visually represent the values of our variables. In ggplot these are known as geoms and every geom is prefixed by geom\_

For a list of the possible geoms see The Geoms Layer

We will use geom\_point which will place a point on the graph at each (var\_a, var\_b)

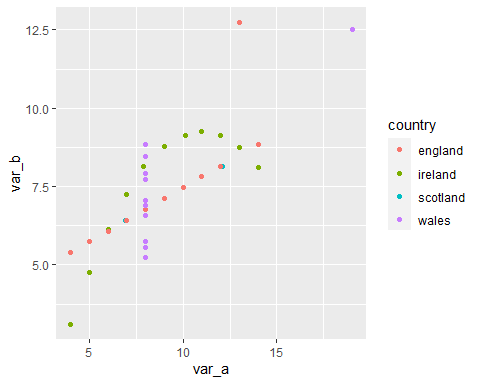
Note how we add the geom\_point using the + operator.

g <-ggplot(data1, aes(x=var\_a, y=var\_b)) +  
 geom\_point()   
  
g

 ### Adding an additional aesthetic mapping

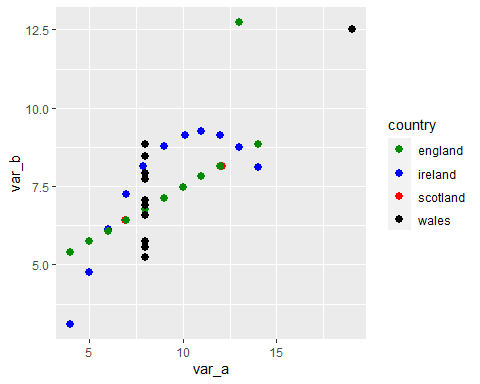
We have a very basic scatterplot, where each point represents a var\_a and var\_b value. We will now add another aesthetic mapping to handle the country factor. This variable has two values green and blue. We will map these to two colours. At first we will allow ggplot to assign the colours using its defaults. Then we will override the colours assigned.

g <-ggplot(data1, aes(x=var\_a, y=var\_b, colour=country)) +  
 geom\_point()   
  
g

 ### Changing the default colours

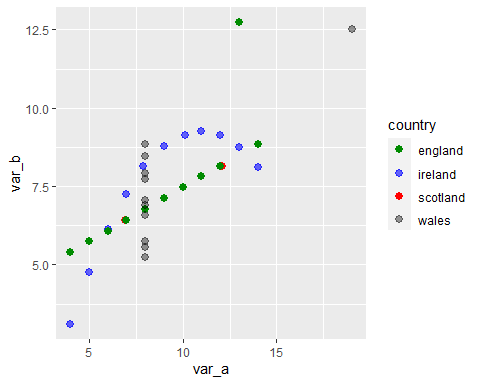
The colour aesthetic mapping visually groups the instances with the same level value. For easier interpretation, we will make the points with the level ireland green and those with the level scotland blue. To do this, we have to manually assign the colours by specifying a vector of colours, in this case, of size 2. We do this by passing the vector of colours to the scale\_colour\_manual function. In general, the functions of typescale\_ ‘x’ \_ manual enable you to override the default mappings ggplot makes to the visual scale of the aesthetic.

g <-ggplot(data1, aes(x=var\_a, y=var\_b, colour=country)) +  
 geom\_point(size=2.5) + # Note that I've increased the point size  
 # This function allows you to manually override the default colour mappings defined by the aes function.   
 scale\_colour\_manual(values = c( "green4","blue","red","black"))   
   
g

 ### Changing alpha values

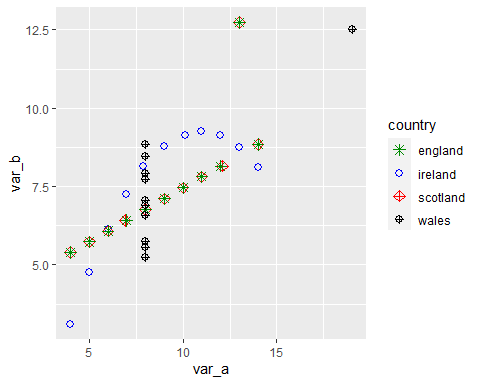
The alpha value controls the transparency of the geometric. Alpha values are in the range (0,1) where 0 means that the point is completely transparent or invisible. In this case, we want to lower the alpha value of the blue points. To manually set the transparency values of two sets of points to different values, we have to treat alpha as an aesthetic that is mapped to the variable that seperates the points. Here the points with the country value of blue should have a lower alpha than points with country value green. Therefore, we map the alpha scale to the values of the variable country. We then override the default mappings using a scale\_alpha\_manual function, just as we did when manually setting the colour of the points.

g <-ggplot(data1, aes(x=var\_a, y=var\_b, colour=country, alpha= country)) +  
 geom\_point(size=2.5) +   
 scale\_colour\_manual(values = c( "green4","blue","red","black")) +  
 scale\_alpha\_manual(values = c(1, 0.6,1,0.4))   
   
g

 ### Changing Geom Shape

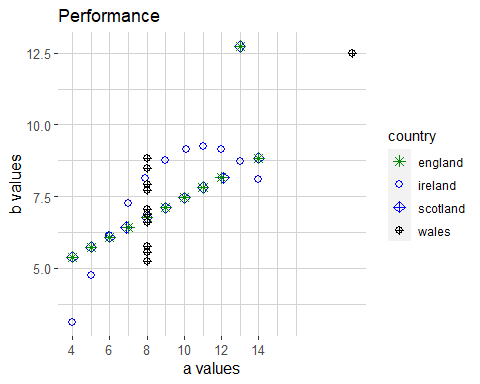
Another approach would be to change the shape of the points so that, even when they overlap on the graph, we can still perceive two different points.

g <-ggplot(data1, aes(x=var\_a, y=var\_b, colour=country, shape=country)) +   
 geom\_point(size=2.5) +   
 scale\_colour\_manual(values = c( "green4","blue","red","black")) +  
 scale\_alpha\_manual(values = c(1, 0.6,1,0.4)) +  
 scale\_shape\_manual(values = c(8, 1,9,10))   
   
   
g

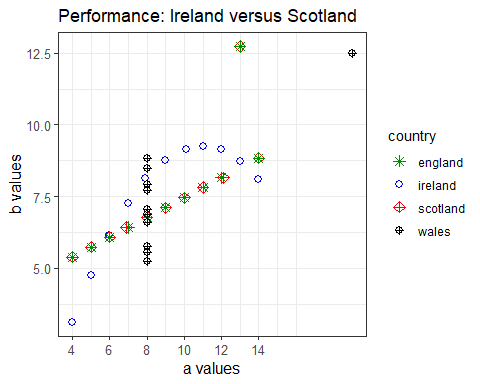
 ### Adjusting the themes layer

title : use the ggtitle function change the labels on the x and y axis : use xlab and ylab functions place more breaks in the x axis : as the x axis uses a continuous scale, we make adjustments to its default layout using the scale\_x\_continuous function. In this case we set the breaks () attribute using a vector of custom values Change the background : we can change the grey background by specifying the background colour as white in the theme function. However, as the default grid-lines are also white, we lose sight of them against the white background. We can specify the colour of the grid-lines as lightgrey as I have done below. Change the size of the axis text : The default size for the axis text is often too small in my opinion. You can change it within the theme function using axis.text = element\_text(size =10) Change the size of the axis title : Again I find that the default size of the axis title is often too small. In this case, I change it to size 12.

g <-ggplot(data1, aes(x=var\_a, y=var\_b, colour=country, shape=country)) +  
 geom\_point(size=2.5) +   
 scale\_colour\_manual(values = c( "green4","blue","blue","black")) +  
 scale\_shape\_manual(values = c(8, 1,9,10)) +  
 ggtitle("Performance") +  
 xlab("a values") + # new label for x axis  
 ylab ("b values") + # new label for x axis  
 scale\_x\_continuous(breaks= c(4,6,8,10,12,14)) +  
   
 theme(panel.background = element\_rect(fill = "white"),  
 panel.grid.major = element\_line(size = 0.25, linetype = 'solid',  
 colour = "lightgrey"),   
 panel.grid.minor = element\_line(size = 0.1, linetype = 'solid',  
 colour = "lightgrey"),  
 axis.text = element\_text(size =10),  
 axis.title = element\_text(size =12))  
g



g <-ggplot(data1, aes(x=var\_a, y=var\_b, colour=country, shape=country)) +  
 geom\_point(size=2.5) +   
 scale\_colour\_manual(values = c( "green4","blue","red","black")) +  
 scale\_shape\_manual(values = c(8, 1,9,10)) +  
 ggtitle("Performance: Ireland versus Scotland") +  
 xlab("a values") + # new label for x axis  
 ylab ("b values") + # new label for x axis  
 scale\_x\_continuous(breaks= c(4,6,8,10,12,14)) +  
   
 theme\_bw() + # prerolled theme   
 theme (axis.text = element\_text(size =10),  
 axis.title = element\_text(size =12)) # I still want to override the default axis text size  
  
g

 ### Final Touch

g <-ggplot(data1, aes(x=var\_a, y=var\_b, colour=country,shape=country)) + geom\_point(size = 2.5) + scale\_shape\_manual(values=c(20,8,15,5),labels=c("England","Ireland","Scotland","Wales")) + scale\_colour\_manual(values = c( "red","blue","green4","black"),labels=c("England","Ireland","Scotland","Wales")) + scale\_alpha\_manual(values = c(0.1,0.5,0.7,1),labels=c("England","Ireland","Scotland","Wales")) + ggtitle("Performance: Ireland/Scotland/England/Wales") + xlab("a values") + ylab ("b values") + scale\_x\_continuous(breaks = seq(0,20,2.5)) + scale\_y\_continuous(breaks = seq(0,15,2.5)) + theme(panel.background = element\_rect(fill="#E6E3C5"),panel.grid.major = element\_line(size = 0.25, linetype = 'solid',  
 colour = "white"),   
 panel.grid.minor = element\_line(size = 0.1, linetype = 'solid',  
 colour = "white"),  
 axis.text = element\_text(size =10),  
 axis.title = element\_text(size =12))  
  
  
g

