Visualising Amounts

We will look at how to represent variation with category values with the following charts

Bar charts Diverging Bar charts Cleveland Dot plots Lollipop plots Pareto Charts

### Bar Plots

Dataset : <https://scannain.com/Irish-box-office-top-ten/> (Irish Box Office Top 10, Weekend: October 25th to 27th, 2019)

ire\_box\_office <- read.csv("ire-movie-october25-2019.csv")  
  
# quick disoplay of variable abd variable types  
str(ire\_box\_office)

## 'data.frame': 10 obs. of 5 variables:  
## $ ï..rank : int 1 2 3 4 5 6 7 8 9 10  
## $ title : chr "Joker" "The Addams Family" "Maleficent: Mistress Of Evil" "Terminator: Dark Fate" ...  
## $ title\_short : chr "Joker" "Addams Family" "Maleficient" "Terminator" ...  
## $ weekend\_gross: int 374216 254192 163910 153624 74161 66844 50149 38760 36500 35663  
## $ gross : int 5124763 254192 511974 196706 74161 206742 201983 38760 140812 2848074

We’ll put this data into tabular form first. I want to show the following fields: rank, film title, weekend gross (in euro). The easiest way to create a formatted euro amount is to use the dollar\_format function using the scales library. Yes, dollar\_format to create a euro symbol!

Using, an adaption of this function to insert the euro sign, I create another column in the data weekend\_euro

library(scales)  
ire\_box\_office$weekend\_euro <- dollar\_format(prefix = "\u20ac", big.mark = ",") (ire\_box\_office$weekend\_gross)

I will use the kable library to format the table. First, Using dplyr I select the columns I want to show into a new data\_frame.

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

boxoffice\_ire\_display <- ire\_box\_office %>%  
 select(ï..rank, title, weekend\_euro,) %>%  
 rename(Rank = ï..rank,  
 Title = title,  
 `Weekend gross` = weekend\_euro)

I use the kableExtra library to create a nice looking table. There is a good tutorial [here] (<https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_html.html>) on using Kable and kableExtra. It contains the word ‘awesome’ in its title so you know it must be good.

library(knitr)  
library(kableExtra)

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

##   
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':  
##   
## group\_rows

boxoffice\_ire\_display%>%  
 kable(align = c('c', 'l', 'c'), caption = 'Highest grossing movies in Ireland for the weekend of October 25-27, 2019. Data source: Scannain.ie') %>%  
 kable\_styling(bootstrap\_options = c("striped", "hover", "condensed"), full\_width = F)

Highest grossing movies in Ireland for the weekend of October 25-27, 2019. Data source: Scannain.ie

Rank

Title

Weekend gross

1

Joker

€374,216

2

The Addams Family

€254,192

3

Maleficent: Mistress Of Evil

€163,910

4

Terminator: Dark Fate

€153,624

5

Abominable

€74,161

6

Shaun The Sheep Movie: Farmageddon

€66,844

7

Zombieland: Double Tap

€50,149

8

Countdown

€38,760

9

Hocus Pocus

€36,500

10

Downton Abbey

€35,663

Data where we have ranked amounts is commonly visualized with vertical bars of different lengths. For each movie, we draw a bar that starts at zero and extends all the way to the value we wish to represent. This type of visualization is called a bar plot or bar chart.

In ggpolot terms, the aesthetic that is used is the length of a column.As such the geometric we use is geom\_col.

You can create a basic bar plot here using the first 3 layers of grammar of graphics. As you can see, there is still some work to do before this plot is presentable. We will now step through what is required

I have used the theme\_set function to set the overall theme to be theme\_classic. I could have done this by simply adding + theme\_classic() to the plot. This clears away all default gridlines, gives white background to the plot.

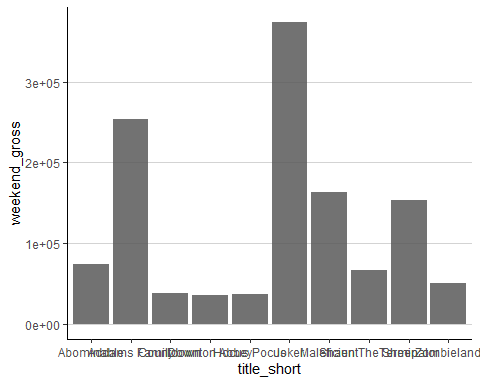
It makes sense to have some gridlines to help establish the values associated with each bar. I just use the major gridlines for the y axis, which can be defined by the panel.grid.major.y parameter within the theme function.

I’d like to be able to see the gridlines behind each bar - for the sake of visual continuity. I set the alpha value of the geom\_col geometric to be 0.8. As you may recall, the alpha value defines the transparency of a visual object.

library(ggplot2)

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

theme\_set(theme\_classic())  
  
ggplot(ire\_box\_office, (aes(x= title\_short, y=weekend\_gross))) +   
 geom\_col(alpha=0.85) +  
 theme(panel.grid.major.y = element\_line(size = 0.2, linetype = 'solid',colour = "lightgrey") )



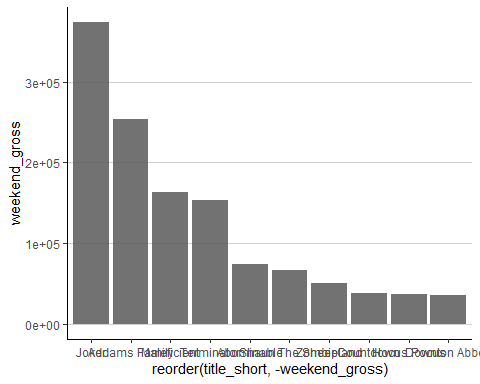
### Ordering the bars

We generally want to order the bars by their value on the y-axis.

We use the reorder function to reorder the bars by their euro value rather than by the default alphabetical ordering which is used by R to order a factor. An alternative approach would be to explicitly set the order of the levels in the factor; In this case, we would set the order of film titles in the title\_short factor.

The reorder function sorts in ascending order by default. To sort in descending order, just make the make the variable you wish to order negative

library(ggplot2)  
  
theme\_set(theme\_classic())  
  
ggplot(ire\_box\_office, (aes(x= reorder(title\_short, -weekend\_gross) , y=weekend\_gross))) +   
 geom\_col(alpha=0.85) +  
 theme(panel.grid.major.y = element\_line(size = 0.2, linetype = 'solid',colour = "lightgrey") )



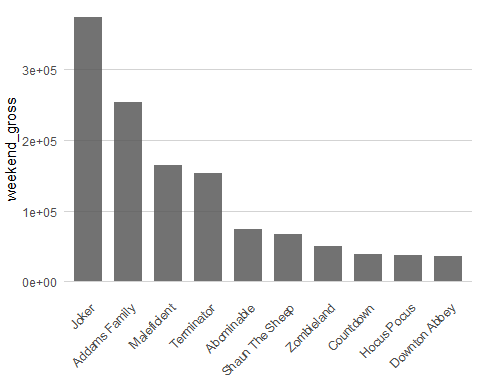
One problem we commonly encounter with vertical bars is that the labels identifying each bar take up a lot of horizontal space. To save horizontal space, we could place the bars closer together and rotate the labels.

This is non-data related adjustment and we do this in the theme function by setting the angle and vjust and hjust values of the axis.text.x parameter.

Note that I have removed the x-axis title and the axis line on both the x and y axes. Again, this was done in the theme function. Setting the value of any parameter in the theme function to element\_blank will make the corresponding visual feature disappear.

The width parameter in the geom\_col function allows you to specify the bar width - and in this case, I have set the width to 0.7 of the default. Try changing it to see the effect it has.

library(ggplot2)  
  
theme\_set(theme\_classic())  
  
ggplot(ire\_box\_office, (aes(x= reorder(title\_short, -weekend\_gross) , y=weekend\_gross))) +   
 geom\_col(alpha=0.85, width = 0.7) +  
 theme(  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.text.x = element\_text(angle = 45, vjust = 1, hjust = 1),  
 axis.title.x = element\_blank(),  
 plot.margin = margin(3, 6, 3, 3),  
 panel.grid.major.y = element\_line(size = 0.2, linetype = 'solid',colour = "lightgrey")  
   
 )



We still need to adjust the y-axis. The scale uses scientific notation, which in this case is not appropriate. It also has a title that needs reformatting. We can solve these issues by customising the x-axis scale. The function to use for this is scale\_x\_continuous.

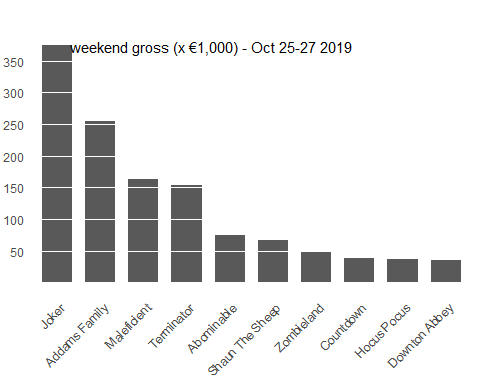
Within this function, I specify the axis breaks I want displayed with a vector of evenly spaced values. I also specify how I want these values to be labelled. This overrides the default breaks and labels imposed by ggplot.

Often an axis becomes crowded and ugly with large numbers, hence the use of the more compact scientific notation. When you don’t want to use this type of notation, you can specify the axis units to be in millions or thousands as I do here. I also specify the name or label I want for the y axis.

In this plot, I use a suggestion on bar plots from Edward Tufte - made in the spirit of reducing ink. I’ve made the gridlines white. Now gridlines are part of the overall background panel in a ggplot. If I bring the overall panel to the fore using panel.ontop = TRUE in the theme function, the background panel also comes to the fore and obscures the plot. To deal with this issue, you set panel.background to be blank.

Using white gridlines eliminates the gridlines on the plot that do not intersect with the bars, and removes the ticks. It also has the effect of breaking bars into visible units based on the break intervals. This allows for visual quantification of the difference between bars. For example it’s easy to see that the lowest ranking films are less all less than one unit high (< 50k), and that Maleficient and Terminator are both just over 3 units high (> 150k )

library(ggplot2)  
  
theme\_set(theme\_classic())  
  
ggplot(ire\_box\_office, (aes(x= reorder(title\_short, -weekend\_gross) , y=weekend\_gross))) +   
 geom\_col( width=0.7) +  
   
 scale\_y\_continuous(limits = c(0, 3.8e5),  
   
 breaks = c( 5e4, 1.0e5, 1.5e5, 2.0e5, 2.5e5,3e5,3.5e5 ),  
 labels = c("50", "100", "150", "200","250", "300", "350" ),  
 name = "weekend gross (x €1,000)") +  
   
 ggtitle("weekend gross (x €1,000) - Oct 25-27 2019 ") +  
   
 theme(  
 axis.title.y = element\_blank(),  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.text.x = element\_text(angle = 45, vjust = 1, hjust = 1),  
 axis.title.x = element\_blank(),  
 plot.title = element\_text(vjust = -10, hjust = 0.25, size = 11),  
 plot.margin = margin(6, 6, 3, 3),  
 panel.background = element\_blank(),  
 panel.grid.major.y = element\_line(size = 0.4, linetype = 'solid',colour = "white"),  
 panel.ontop = TRUE  
   
 )

 The gridlines or (‘Tufte’ lines that are drawn here) help in guiding your eye to the from the top of each bar to the corresponding horizontal location on the y axis. As presented, the plot gives us the ranking of the movies and an approximation of the gross revenues each generated. We also get a sense of the comparative differences in revenue for these movies.

If we really needed more precision in presenting the actual revenue values per movie, we could insert the value over the bar and drop the y-axis and gridlines entirely. However, the visualisation does lose something in terms of being able to see at a glance the comparative difference in revenue values per movie.

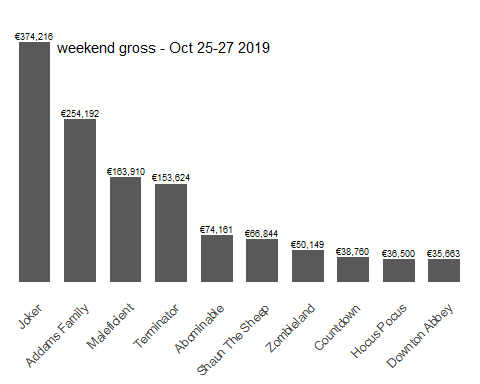
We use the geom\_text geometric to realise this. This demonstrates that an additional geom level aesthetic mapping is also possible. Here we specify that the label property of geom\_text is to be associated with the value of weekend\_euro, the variable I created earlier to hold formatted euro values of the weekend\_gross values in the dataset. The geom\_text elements will be placed at the x, y locations specified by the plot level aesthetic mapping, and the value of each geom\_text point will be taken from weekend\_euro.

I specify the size of the text and make a slight vertical adjustment with vjust so that the text hovers over the bar instead of just inside it.

To remove the y-axis, I have removed the scale\_y\_continuous function and in the theme function, I have set axis.title.y and axis.text.y to element\_blank.

I have added a title using ggtitle and moved it with the plot area, using the plot.title parameter in theme so as to make clear what the numbers represent.

library(ggplot2)  
  
theme\_set(theme\_classic())  
  
ggplot(ire\_box\_office, (aes(x= reorder(title\_short, -weekend\_gross) , y=weekend\_gross))) +   
 geom\_col(width=0.7) +  
 geom\_text(aes(label=weekend\_euro), size =2.5, vjust=-0.4) +  
 ggtitle("weekend gross - Oct 25-27 2019 ") +  
   
 theme(  
 axis.title.y = element\_blank(),  
 axis.text.y = element\_blank(),  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.text.x = element\_text(angle = 45, vjust = 1, hjust = 1),  
 axis.title.x = element\_blank(),  
 plot.margin = margin(6, 6, 3, 3),  
 plot.title = element\_text(vjust = -10, hjust = 0.2, size = 11)  
   
 )

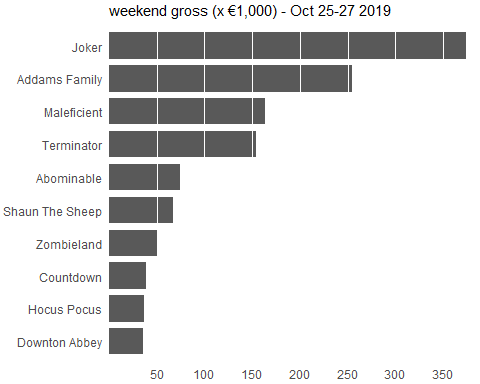
 While the film titles are now readable, there is a disadvantage in that the plot is vertically longer than it needs to be, and the reader has to slightly tilt their reading angle to read the titles on the x-axis.

An alternative is to swap the x and the y axis, so that the bars run horizontally. This is often done where there are more than, say, 6 categories to visualise.

Adding The coord\_flip function will swap the axes. As the film titles are now on the y axis and the revenue values are on the x axis, I have to also reinstate the x-axis title and remove the y -axis title.

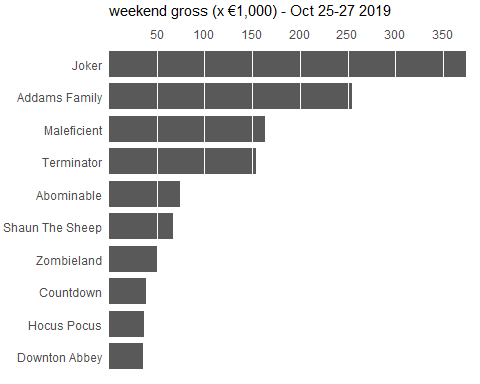
I’ve put in gridlines - this time, major gridlines for the x axis. This is done within the theme function by specifying values for the panel.grid.major.x parameter.

library(ggplot2)  
  
theme\_set(theme\_classic())  
  
ggplot(ire\_box\_office, (aes(x= reorder(title\_short, weekend\_gross) , y=weekend\_gross))) +   
 geom\_col( width=0.8) +  
 scale\_y\_continuous(limits = c(0, 3.8e5),  
 expand = c(0, 0),  
 breaks = c( 5e4, 1.0e5, 1.5e5, 2.0e5, 2.5e5,3e5,3.5e5 ),  
 labels = c("50", "100", "150", "200","250", "300", "350" ),  
 name = "weekend gross (x €1,000)") +  
   
 ggtitle("weekend gross (x €1,000) - Oct 25-27 2019 ") +  
  
 coord\_flip(clip = "off") +  
 theme(  
 axis.title = element\_blank(),  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.title.y = element\_blank(),  
 plot.title = element\_text(size = 11),  
 plot.margin = margin(3, 6, 3, 3),  
 panel.background = element\_blank(),  
 panel.grid.major.x = element\_line(size = 0.4, linetype = 'solid',colour = "white"),  
 panel.ontop = TRUE  
   
 )

 It could be argued that without the visible gridline that the scale on the axis might be hard to reconcile to the white break lines in the bars. An approach to address this is to put the x-axis at the top of the figure. Now this is not commonly done - and it is not completely straightforward to do this in ggplot either. However with ordered bars and the invisible ‘Tufte’ gridlines, I think it is permissible.

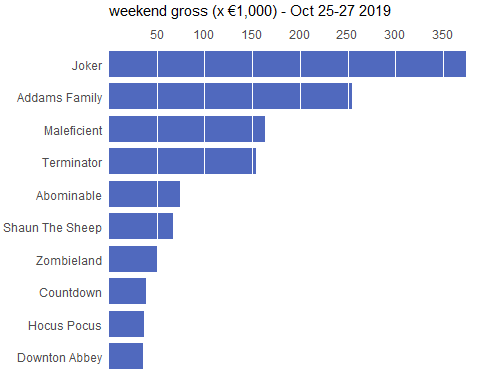
To accomplish this ggplot, I specify the axis as a duplicate axis sec.axis = dup\_axis(). As the name suggests, this creates a second duplicate axis (on top). In the theme function, I remove the axis from the bottom of the plot by setting the axis.title.x.bottom and axis.text.x.bottom attributes to element\_blank()

library(ggplot2)  
  
theme\_set(theme\_classic())  
  
ggplot(ire\_box\_office, (aes(x= reorder(title\_short, weekend\_gross) , y=weekend\_gross))) +   
 geom\_col( width=0.8) +  
   
   
 scale\_y\_continuous(sec.axis = dup\_axis(), limits = c(0, 3.8e5),  
 expand = c(0, 0),  
 breaks = c( 5e4, 1.0e5, 1.5e5, 2.0e5, 2.5e5,3e5,3.5e5 ),  
 labels = c("50", "100", "150", "200","250", "300", "350" ),  
 name = "weekend gross (x €1,000)") +  
 ggtitle("weekend gross (x €1,000) - Oct 25-27 2019 ") +  
  
 coord\_flip(clip = "off") +  
 theme(  
 axis.title = element\_blank(),  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.title.y = element\_blank(),  
 axis.title.x.bottom = element\_blank(),  
 axis.text.x.bottom = element\_blank(),  
 plot.title = element\_text(size = 11),  
 plot.margin = margin(3, 6, 3, 3),  
 panel.background = element\_blank(),  
 panel.grid.major.x = element\_line(size = 0.4, linetype = 'solid',colour = "white"),  
 panel.ontop = TRUE)

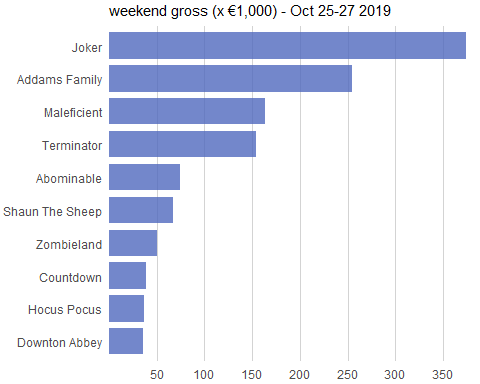
 After swapping the axes, we obtain a compact figure in which all visual elements, including all text, are horizontally oriented. As a result, the figure is much easier to read.

By default, bar graphs use dark grey bars. However we can customise the colour fill of the bars using the fill property

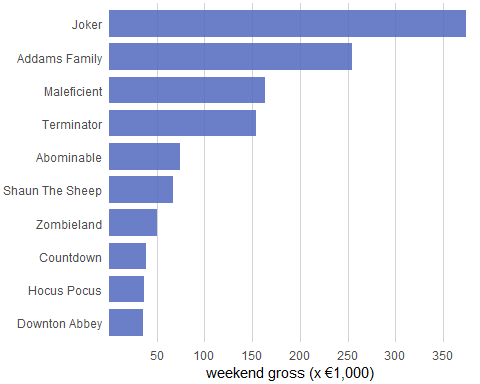
library(ggplot2)  
  
theme\_set(theme\_classic())  
  
ggplot(ire\_box\_office, (aes(x= reorder(title\_short, weekend\_gross) , y=weekend\_gross))) +   
 geom\_col( fill="#5069be", width=0.8) +  
   
   
 scale\_y\_continuous(sec.axis = dup\_axis(), limits = c(0, 3.8e5),  
 expand = c(0, 0),  
 breaks = c( 5e4, 1.0e5, 1.5e5, 2.0e5, 2.5e5,3e5,3.5e5 ),  
 labels = c("50", "100", "150", "200","250", "300", "350" ),  
 name = "weekend gross (x €1,000)") +  
 ggtitle("weekend gross (x €1,000) - Oct 25-27 2019 ") +  
  
 coord\_flip(clip = "off") +  
 theme(  
 axis.title = element\_blank(),  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.title.y = element\_blank(),  
 axis.title.x.bottom = element\_blank(),  
 axis.text.x.bottom = element\_blank(),  
 plot.title = element\_text(size = 11),  
 plot.margin = margin(3, 6, 3, 3),  
 panel.background = element\_blank(),  
 panel.grid.major.x = element\_line(size = 0.4, linetype = 'solid',colour = "white"),  
 panel.ontop = TRUE)

 And here is the same plot with the x-axis on the bottom and standard grey gridlines

library(ggplot2)  
  
theme\_set(theme\_classic())  
  
ggplot(ire\_box\_office, (aes(x= reorder(title\_short, weekend\_gross) , y=weekend\_gross))) +   
 geom\_col(alpha=0.8, fill="#5069be", width=0.8) +  
   
   
 scale\_y\_continuous(limits = c(0, 3.8e5),  
 expand = c(0, 0),  
 breaks = c( 5e4, 1.0e5, 1.5e5, 2.0e5, 2.5e5,3e5,3.5e5 ),  
 labels = c("50", "100", "150", "200","250", "300", "350" ),  
 name = "weekend gross (x €1,000)") +  
 ggtitle("weekend gross (x €1,000) - Oct 25-27 2019 ") +  
  
 coord\_flip(clip = "off") +  
 theme(  
 axis.title = element\_blank(),  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.title.y = element\_blank(),  
 plot.title = element\_text(size = 11),  
 plot.margin = margin(3, 6, 3, 3),  
 panel.grid.major.x = element\_line(size = 0.2, linetype = 'solid',colour = "lightgrey"))

 Here it is with the title removed and the axis title reinstated.

library(ggplot2)  
  
theme\_set(theme\_classic())  
  
ggplot(ire\_box\_office, (aes(x= reorder(title\_short, weekend\_gross) , y=weekend\_gross))) +   
 geom\_col(fill="#5069be",alpha = 0.85, width=0.8) +  
 scale\_y\_continuous(limits = c(0, 3.8e5),  
 expand = c(0, 0),  
 breaks = c( 5e4, 1.0e5, 1.5e5, 2.0e5, 2.5e5,3e5,3.5e5 ),  
 labels = c("50", "100", "150", "200","250", "300", "350" ),  
 name = "weekend gross (x €1,000)") +  
 coord\_flip(clip = "off") +  
 theme(  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.title.y = element\_blank(),  
 plot.margin = margin(3, 6, 3, 3),  
 panel.grid.major.x = element\_line(size = 0.2, linetype = 'solid',colour = "lightgrey")  
   
 )



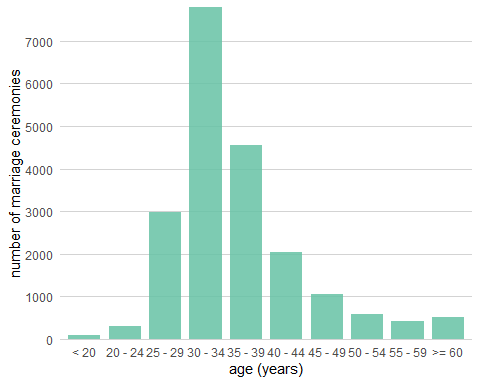
## Different orderings

In the previous example, we reordered the bars. We should only do this when there is no natural ordering to the categories the bars represent. Whenever there is a natural ordering (i.e. when our categorical variable is an ordered factor) we should retain that ordering in the visualization.

For example, this figure shows the age profiles of grooms in registered marriages in Ireland in 2018, ordered by age groups. In this case, the bars should be arranged in order of increasing age. Sorting by bar height while shuffling the age groups makes no sense. The data for this plot was obtained from the [Central Statistics Office]

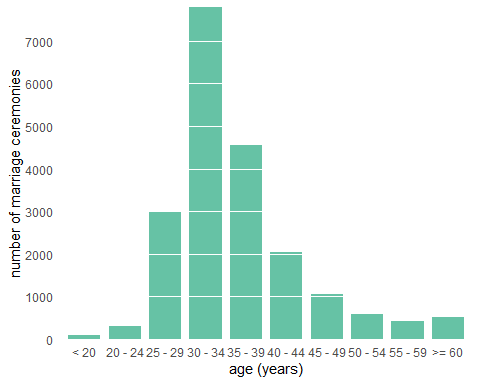
(<https://statbank.cso.ie/px/pxeirestat/statire/SelectTable/Omrade0.asp?Planguage=0>)

bride\_groom\_age <- read.csv("ire\_2018\_bride\_groom\_age.csv")  
  
library(ggplot2)  
library (dplyr)  
  
bride\_groom\_age$age <- factor(bride\_groom\_age$age, levels = c("< 20", "20 - 24", "25 - 29", "30 - 34", "35 - 39", "40 - 44","45 - 49", "50 - 54", "55 - 59",">= 60" ))  
  
bride\_groom\_age\_g<-bride\_groom\_age%>%filter(bride\_groom=="Groom")  
  
theme\_set(theme\_classic())  
  
ggplot(bride\_groom\_age\_g, (aes(x= age, y=ceremonies))) +   
 geom\_col(fill="#66c2a5", alpha=0.85, width =0.8) +  
 scale\_y\_continuous(  
 breaks = c(0, 1000, 2000, 3000, 4000, 5000, 6000, 7000),  
 expand = c(0, 0),  
 name = "number of marriage ceremonies") +  
 xlab(label = "age (years)") +  
   
 theme(  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 panel.grid.major.y = element\_line(size = 0.2, linetype = 'solid',colour = "lightgrey") )



Here is the same data plotted using the ‘Tufte’ gridlines approach.

library(ggplot2)  
  
theme\_set(theme\_classic())  
  
ggplot(bride\_groom\_age\_g, (aes(x= age, y=ceremonies))) +   
 geom\_col(fill="#66c2a5", width =0.8) +  
 scale\_y\_continuous(  
 breaks = c(0, 1000, 2000, 3000, 4000, 5000, 6000, 7000),  
 expand = c(0, 0),  
 name = "number of marriage ceremonies") +  
 xlab(label = "age (years)") +  
   
 theme(  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 panel.background = element\_blank(),  
 panel.grid.major.y = element\_line(size = 0.4, linetype = 'solid', colour = "white"),  
 panel.ontop = TRUE)

 ## Grouped and Stacked bars

Frequently, however, we are interested in two categorical variables at the same time. In a grouped bar plot, we draw a group of bars at each position along the x axis, determined by one categorical variable, and then we draw bars within each group according to the other categorical variable.

In the next plot, we will include the age profile of the brides in registered marriage ceremonies in Ireland.

We use colour fill as an aesthetic so that bars representing groom data will be one colour and bars representing bride data will be another. In the aes function we set fill = bride\_groom, the variable designating whether a data point is for bride or groom.

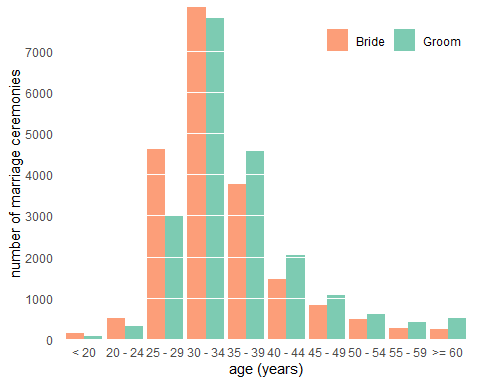
You must use the position = “dodge” in the geom\_col function, which tells the bars to avoid each other horizontally.

Check to what happens when you don’t use position = “dodge”

The default ggplot colours aren’t great. We can use the scale\_fill\_manual function to set our own pair of colour values, which are a pair of colour-blind friendly qualitative colours selected from the <http://colorbrewer2.org> website. I have set name = NULL in this function to tell ggplot not to render the name of the legend, which is often not required.

In order to reduce the width of the overall plot, I have moved the legend inside the plot from its default position on the right hand-side outside the plot. It is OK to do this, as long as the legend cannot be confused with or interfere with the perception of the data elements of the plot.

library(ggplot2)  
library (dplyr)  
  
  
theme\_set(theme\_classic())  
  
ggplot(bride\_groom\_age, (aes(x= age, y=ceremonies, fill=bride\_groom))) +   
 geom\_col(position="dodge", alpha=0.85) +  
 scale\_y\_continuous(  
 breaks = c(0, 1000, 2000, 3000, 4000, 5000, 6000, 7000),  
 expand = c(0, 0),  
 name = "number of marriage ceremonies") +  
 xlab(label = "age (years)") +  
 scale\_fill\_manual(values = c("#fc8d62","#66c2a5" ), name = NULL) +  
   
 theme(  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 panel.background = element\_blank(),  
 panel.grid.major.y = element\_line(size = 0.4, linetype = 'solid', colour = "white"),  
 panel.ontop = TRUE,  
 legend.position= c(0.8, 0.9), legend.direction="horizontal")



Here is another example, but this time with multiple bars. The data is from the Central Statistics Office and gives the distribution of principle economic status values per quarter in 2016.

By running the head function you can see that the data is in wide form, with Q1, Q2 etc represented as columns. ggplot tends to work best with data in long form, so we have to transform the dataframe.

Luckily, there is a tidyverse library called tidyr that allows us to do this easily using the gather function. Using this function, I create two new columns quarter and value, which will represent the transformed data in the new dataframe. You can run head on the new dataframe to see the transformation.

There is a nice explanation on tidyr here

I have used colours from the colourblind friendly palette of Okabe and Ito for this visualisation, but I have desaturated them a little as they were a little bright for my taste.

Note that this is an example where you have to choose your break intervals sensitively. Five out six bars have values below 500 and one bar has a value over 1750. If your break frequency is too great, the gridlines make the overall plot look too busy; At the same time, if your breaks are too far apart, say at levels 1000 and 2000, then it will be difficult to resolve values for the shorter bars against the axis.

As before, in order to reduce the width of the overall plot, I have moved the legend inside the plot from its default position on the right hand-side outside the plot.

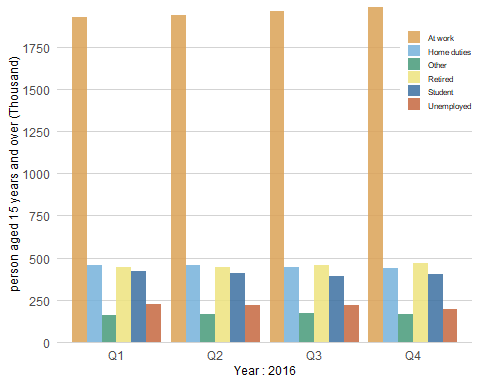
library(ggplot2)  
library (dplyr)  
library(tidyr)  
library(colorspace)  
  
econ\_stat\_per\_q <- read.csv("2016PrincipalEconomicStatusPerQuarter.csv")  
  
head(econ\_stat\_per\_q)

## status Q1.2016 Q2.2016 Q3.2016 Q4.2016  
## 1 At work 1926.1 1941.1 1961.7 1987.1  
## 2 Unemployed 222.7 221.8 220.6 194.1  
## 3 Student 419.6 408.8 392.9 402.8  
## 4 Engaged on home duties 455.0 454.6 445.2 436.0  
## 5 Retired from employment 441.7 444.3 457.7 467.1  
## 6 Other 161.1 167.1 170.6 167.8

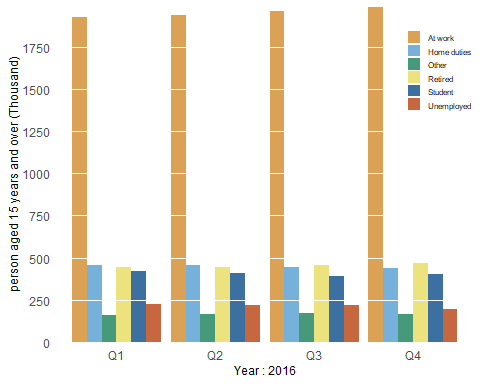
# convert the data from wide from to long form, which is required by ggplot  
econ\_stat\_per\_q %>% gather(quarter, value, Q1.2016:Q4.2016)->econ\_stat\_per\_q\_long  
  
head(econ\_stat\_per\_q\_long)

## status quarter value  
## 1 At work Q1.2016 1926.1  
## 2 Unemployed Q1.2016 222.7  
## 3 Student Q1.2016 419.6  
## 4 Engaged on home duties Q1.2016 455.0  
## 5 Retired from employment Q1.2016 441.7  
## 6 Other Q1.2016 161.1

theme\_set(theme\_classic())  
  
# colour blind friendly palette - slightly desaturated  
cbPalette <- desaturate(c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7"), 0.25)  
  
ggplot(econ\_stat\_per\_q\_long, (aes(x= quarter, y=value, fill=status))) +   
 geom\_col(position="dodge", alpha=0.85) +  
 scale\_y\_continuous(  
 breaks = c(0, 250, 500, 750, 1000, 1250, 1500, 1750, 2000),  
 expand = c(0, 0),  
 name = "person aged 15 years and over (Thousand)") +  
 xlab(label = "Year : 2016") +  
scale\_fill\_manual(values = cbPalette[2:8], name = NULL, labels = c("At work","Home duties","Other","Retired","Student","Unemployed")) +  
 scale\_x\_discrete(labels = c("Q1","Q2","Q3","Q4"))+  
   
 theme(  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.title = element\_text(size=9),  
 panel.grid.major.y = element\_line(size = 0.2, linetype = 'solid',colour = "lightgrey"),  
   
 legend.position= c(0.92, 0.82),  
 legend.text = element\_text(size = 6), # legend text was a little large  
 legend.key.size = unit(0.7, "lines"), # legend key size was a little large  
 legend.title = element\_text(size =10))

 I have produced a Tufte version of the same plot, which I think works.

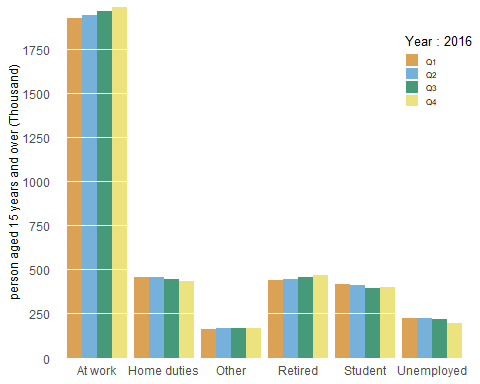
library(ggplot2)  
library (dplyr)  
library(tidyr)  
library(colorspace)  
  
theme\_set(theme\_classic())  
  
# colour blind friendly palette - slightly desaturated  
cbPalette <- desaturate(c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7"), 0.25)  
  
ggplot(econ\_stat\_per\_q\_long, (aes(x= quarter, y=value, fill=status))) +   
 geom\_col(position="dodge") +  
 scale\_y\_continuous(  
 breaks = c(0, 250, 500, 750, 1000, 1250, 1500, 1750, 2000),  
 expand = c(0, 0),  
 name = "person aged 15 years and over (Thousand)") +  
 xlab(label = "Year : 2016") +  
scale\_fill\_manual(values = cbPalette[2:8], name = NULL, labels = c("At work","Home duties","Other","Retired","Student","Unemployed")) +  
 scale\_x\_discrete(labels = c("Q1","Q2","Q3","Q4"))+  
   
 theme(  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.title = element\_text(size=9),  
 panel.background=element\_blank(),  
 panel.grid.major.y = element\_line(size = 0.4, linetype = 'solid',colour = "white"),  
 panel.ontop = TRUE,  
 legend.position= c(0.92, 0.82),  
 legend.text = element\_text(size = 6), # legend text was a little large  
 legend.key.size = unit(0.7, "lines"), # legend key size was a little large  
 legend.title = element\_text(size =10))# legend title was a little large)

 Grouped bar plots show a lot of information at once and they can be confusing. It is a little difficult is difficult to compare numbers of persons across quarters, although admittedly the numbers are relatively static.

If we care more about the overall pattern per economic status group over a given time span, we can group the bars by quarter, clearly showing the change in each group per quarter.

As such, the next plot provides an alternative bar grouping of the same data, where the reader is assumed to be more interested in comparing the quarterly differences within each socio-economic class

library(ggplot2)  
library (dplyr)  
library(tidyr)  
library(colorspace)  
  
theme\_set(theme\_classic())  
  
# colour blind friendly palette - slightly desaturated  
cbPalette <- desaturate(c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7"), 0.25)   
  
ggplot(econ\_stat\_per\_q\_long, (aes(x= status, y=value, fill=quarter))) +   
 geom\_col(position="dodge") +  
 scale\_y\_continuous(  
 breaks = c(0, 250, 500, 750, 1000, 1250, 1500, 1750, 2000),  
 expand = c(0, 0),  
 name = "person aged 15 years and over (Thousand)") +  
scale\_fill\_manual(values = cbPalette[2:8], name = "Year : 2016", labels = c("Q1","Q2","Q3","Q4")) +  
 scale\_x\_discrete(labels = c("At work","Home duties","Other","Retired","Student","Unemployed"))+  
   
 theme(  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.title.x = element\_blank(),  
 axis.title.y = element\_text(size = 9),  
 panel.background=element\_blank(),  
 panel.grid.major.y = element\_line(size = 0.4, linetype = 'solid',colour = "white"),  
 panel.ontop = TRUE,  
 legend.position= c(0.92, 0.82),  
 legend.text = element\_text(size = 6), # legend text was a little large  
 legend.key.size = unit(0.7, "lines"), # legend key size was a little large  
 legend.title = element\_text(size =10))# legend title was a little large)



The plot below provides an extended view of the data in the figure above. It provides an average Principal Economic Status figure for each group per year. The average is calculated from the quarterly figures per year.

As the bars within each group are ordered (by year), I want colour to indicate some progression in time.

As such I considered using a sequential palette (e.g. from light blue to dark blue). However, as we have 6 years to colour, I found the lightest colour in the sequence was too light. I decided to use the viridis palette as even the palette for the discrete scale suggests a sequence. This palette also has the added bonus of being colour blind friendly.

In the code below, I once again transform the data from wide form| to long form.

In the previous plot, you may have noticed that the ordering of the values of the status factor is alphabetical - At work, Home Duties, Other, etc. Having the value Other in the middle of the more meaningful values is a bit counter-intuitive. It would typically be the last category anyone might be interested in. Although the status variable is an unordered factor, we can impose a specific ordering on it that reflects the interest of our readers. I do this by explicitly resetting the ordering of the levels of the status factor in the dataframe.

library(RColorBrewer)  
  
econ\_stat <- read.csv("2012-2017-principle\_econ\_year.csv")  
  
head(econ\_stat)

## ï..status y2012 y2013 y2014 y2015 y2016 y2017  
## 1 At work 1780.100 1816.650 1853.575 1908.225 1954.000 1999.15  
## 2 Unemployed 361.175 325.600 290.825 246.150 214.800 185.10  
## 3 Student 409.675 414.175 412.400 406.675 406.025 415.20  
## 4 Engaged on home duties 504.000 481.800 475.200 471.925 447.700 429.65  
## 5 Retired from employment 392.175 404.275 413.050 423.475 452.700 472.50  
## 6 Other 147.250 150.275 151.850 155.100 166.650 172.45

# convert the data from wide from to long form, which is required by ggplot  
econ\_stat %>% gather(year, value, y2012:y2017)->econ\_stat\_long  
colnames(econ\_stat\_long)<-c("status","year","value")  
head(econ\_stat\_long)

## status year value  
## 1 At work y2012 1780.100  
## 2 Unemployed y2012 361.175  
## 3 Student y2012 409.675  
## 4 Engaged on home duties y2012 504.000  
## 5 Retired from employment y2012 392.175  
## 6 Other y2012 147.250

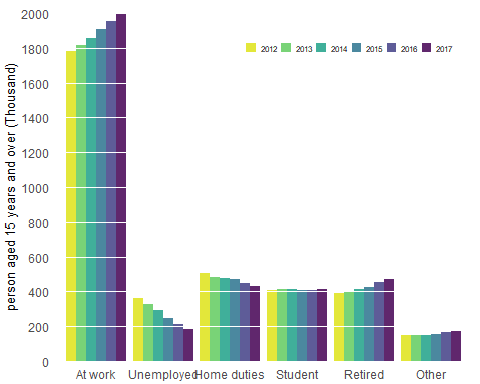
# set the ordering of status factor  
  
levels(econ\_stat\_long$status) # This gives you the current level ordering (alphabetic)

## NULL

econ\_stat\_long$status <- factor(econ\_stat\_long$status, levels = c("At work", "Unemployed", "Engaged on home duties", "Student", "Retired from employment", "Other"))  
  
levels(econ\_stat\_long$status) # This shows you the revised ordering (custom)

## [1] "At work" "Unemployed"   
## [3] "Engaged on home duties" "Student"   
## [5] "Retired from employment" "Other"

theme\_set(theme\_classic())  
  
fig<- ggplot(econ\_stat\_long, (aes(x= status, y=value, fill=year))) +   
 geom\_col(position="dodge", alpha=0.85) +  
 scale\_y\_continuous(  
 limits=c(0,2001),  
 breaks = seq(0, 2000, by =200),  
 expand = c(0, 0),  
 name = "person aged 15 years and over (Thousand)") +  
   
 scale\_fill\_viridis\_d(name = NULL, end = 0.95, direction =-1, labels = c("2012","2013","2014","2015","2016","2017")) + # end value =095 cuts out really bright yellow at end of palette  
   
 scale\_x\_discrete(labels = c("At work", "Unemployed", "Home duties", "Student", "Retired", "Other"), name = NULL)+  
   
 guides(fill = guide\_legend(nrow = 1))+ # ggplot will tend to wrap lonfg horizontal legends. This tells it to use just one row for the legend elements  
   
 theme(  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.title = element\_text(size=10),  
 panel.background=element\_blank(),  
 panel.grid.major.y = element\_line(size = 0.1, linetype = 'solid',colour = "white"),  
 panel.ontop = TRUE,  
   
 plot.margin = margin(10, 6, 3, 3),  
 legend.position= c(0.7, 0.9), legend.direction="horizontal",  
 legend.text = element\_text(size = 6), # legend text was a little large  
 legend.key.size = unit(0.6, "lines"),# legend keys were a little large  
 legend.spacing.x = unit(0.1, 'cm'))  
  
  
fig



#library(colorblindr)  
  
#cvd\_grid(fig)

### Faceted View

Both Figures encode a single categorical variable by position along the x axis and another by bar color. The encoding by position is easy to read while the encoding by bar color requires more cognitive effort, as we have to visually match the colors of the bars against the colours in the legend.

An approach that reduces this is to separate out the grouped plots into a panel of regular bar plots. In ggplot, this is known as faceting - and it is quite easy to facet an existing plot. The key idea is that you choose a variable whose values will determine the number of plots to be created. In this case, we will facet by the status variable, which will mean that ggplot will create 6 individual plots - one for each status value.

The principle changes here are as follows:

in the aes function x is set to year and y is set to value. This is because we want each panel to show values for the years 2012 - 2017. we set the fill colour to static colour as we would do for a single (non-grouped) bar plot

add the function facet\_wrap(~status, scales = “free\_x”) to the plot. This tells ggplot to create a set of plots (facets) — one for each value of the status variable. The facets can be placed next to each other, wrapping with a certain number of columns or rows. The scales parameter is set to free\_x. Normally, the axis scales on each facet are fixed, which means that they have the same size and range. They can be made independent, by setting scales to free, free\_x, or free\_y. In this case, we know that the year variable has the same values for each status value - so the use of scales = “free\_x” would seem to be redundant. However, by including it, it forces ggplot to include the x-axis scale under each plot, which I think is easier to read.By default, it would otherwise put the x-axis scale at the bottom of the panel. Remove it and see for yourself. Finally, I’ve made some modifications to the title of each facet - known as strip text. I’ve done this in the theme function.

#library(RColorBrewer)  
  
  
theme\_set(theme\_classic())  
  
fig<- ggplot(econ\_stat\_long, (aes(x= year, y=value))) +   
 geom\_col(fill="#5069be", alpha=0.85) +  
 scale\_y\_continuous(  
 limits=c(0,2001),  
 breaks = c(0, 500, 1000, 1500,2000),  
 expand = c(0, 0),  
 name = "person aged 15 years and over (Thousand)") +  
   
 scale\_x\_discrete(labels = c("2012","2013","2014","2015","2016","2017"), name = NULL) +  
   
 facet\_wrap(~status,ncol=3,scales = "free\_x") +  
   
 theme(  
 axis.line.y = element\_blank(),  
 axis.ticks.y = element\_blank(),  
 axis.line.x = element\_blank(),  
 axis.ticks.x = element\_blank(),  
 axis.title.x = element\_text(size=8),  
 axis.title.y = element\_text(size=8),  
 axis.text.x = element\_text(size=6),  
 axis.text.y = element\_text(size=6),  
  
 panel.spacing.y = grid::unit(14, "pt"),  
 strip.text = element\_text(size = 8),  
 strip.background = element\_blank(),  
 plot.margin = margin(10, 6, 3, 3),  
 panel.background=element\_blank(),  
 panel.grid.major.y = element\_line(size = 0.1, linetype = 'solid',colour = "white"),  
 panel.ontop = TRUE)   
  
fig

