Hierarchical Proportions

Some data is hierarchical and visualizing the overall proportions within each category cannot be achieved using the techniques we’ve seen up to now

The traditional approach would be to represent tree structures like a file system as a rooted directed graph with the root node at the top of the page and the child nodes below the parent will lines connecting them.

## Tree Maps

A Tree Map is a 2D representation of hierarchical information. It represents the quantities for each category using area size.

Each category is assigned a rectangle area with subcategory rectangles nested inside it. The area assigned to a category is in proportion to the quantity value of the category . The area assigned to a category is in proportion to the quantity value of the category, and is proportional to other quantities within the parent category. The area size of the parent category is the total of its subcategories.

Tree maps have a lof of information to show and can provide an intuitive overview of the categorical proportions in heirarchcal data. There are some drawbacks.

All Labels may not display

While our brains are good at comparing the lengths of objects (e.g bars) or the positions of objects (e.g. dots) we compare sizes of rectangles with less precision

This is really a case where the visualisation gives us the overall gist of the story

library(ggplot2)

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

library(treemapify)

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

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

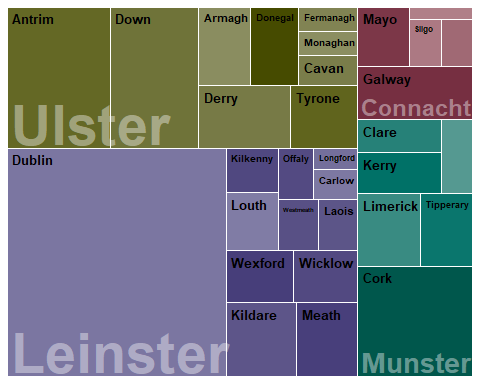
ire\_counties <- read.csv("ire\_counties.csv")  
  
  
ire\_counties <- ire\_counties %>% rename("Density" = "Density....kmÂ².")%>%rename("Area"="Area..kmÂ².")%>% rename("Rank" = "ï..Rank")  
  
  
ire\_counties$Province <- factor(ire\_counties$Province, levels = c("Connacht", "Ulster", "Munster","Leinster"))  
  
head(ire\_counties)

## Rank County Area Density Province Population  
## 1 1 Cork 7500 72.3 Munster 542868  
## 2 2 Galway 6149 42.0 Connacht 258058  
## 3 3 Mayo 5586 23.3 Connacht 130507  
## 4 4 Donegal 4861 32.6 Ulster 159192  
## 5 5 Kerry 4807 30.7 Munster 147707  
## 6 6 Tipperary 4305 37.2 Munster 159553

Thanks to Claus Wilke who provided the solution on Stack Overflow on how to colour the tree map using several sequential scales.

<https://stackoverflow.com/questions/50163072/different-colors-with-gradient-for-subgroups-on-a-treemap-ggplot2-r>

#ire\_counties$Province <- factor(ire\_counties$Province, levels = c("Athena", "Hermes", "Demeter","Zeus"))  
  
# code to add colors to data frame follows  
# first the additional packages needed  
  
library(colorspace)   
library(scales)  
  
# number of palettes needed  
  
n <- length(unique(ire\_counties$Province))  
  
# now calculate the colors for each data point  
ire\_counties\_df2 <- ire\_counties %>%  
 mutate(index = as.numeric(factor(Province))- 1) %>%  
 group\_by(index) %>%  
 mutate(  
 max\_area = max(Area),  
 colour = gradient\_n\_pal(  
 sequential\_hcl(  
 6,  
 h = 360 \* index[1]/n,  
 c = c(45, 20),  
 l = c(30, 80),  
 power = 0.6)  
 )(1- (Area/max\_area))  
 )  
  
ggplot(data = ire\_counties\_df2 , aes(area = Population , fill = colour, subgroup = Province)) +  
geom\_treemap(colour = "white", size = 0.5\*.pt, alpha = NA) +  
   
 geom\_treemap\_text(aes(label = County), colour = "black" , size =10, place = "topleft",fontface = "bold",padding.x = grid::unit(1.5, "mm"),padding.y = grid::unit(1.5, "mm")) +  
  
 geom\_treemap\_subgroup\_border(colour = "white", size =0.5) +   
   
 geom\_treemap\_subgroup\_text(grow = FALSE, colour = "#FAFAFA", size = 42, place ="bottomleft", fontface = "bold", alpha = 0.4) +  
   
 scale\_fill\_identity()+  
   
 coord\_cartesian(clip = "off") +  
 guides(colour = "none", fill = "none")



There are three aesthetics at play here.

Tile area, representing quantity population. The population for each county and province, is represented by portion of area it occupies in the overall 2D space.

A qualitative colour scale representing the parent level categories – e. olive green for Ulster, purple for Leinster, Crimson for Connact and Turquoise for Munster

sequential colour scales – one associated with the colour assigned to each province. Each colour scale represents quantity county area. In the colour gradient used, a dark shade indicates a high value and a light shade indicates a low value of county area.

The treemap gives you no indication of scale. In terms of mapping tile area to quantity, I feel that adding a legend (for example where a square of given area indicates a particular value) would NOT help.

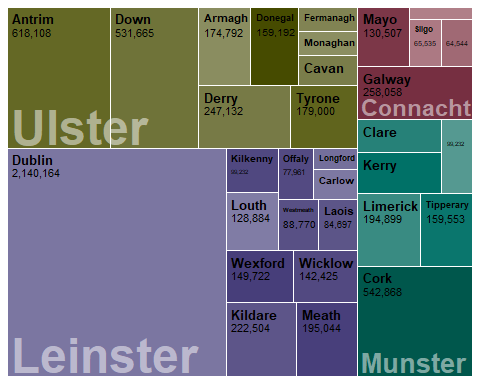
For that matter, adding 4 legends – one for each sequential scale used, would make the graphic cluttered looking, apart from the fact that matching colours is an imprecise task at the best of times.

There isn’t necessarily a solution to this as visualising nested proportions can be tricky.

However, it is important to have some sense of scale, and most tree maps include an indication of quantity in at least some of the tiles

The values included in the following tree map are the population values `and the idea is that they are confirmatory indications of quantity rather than the primary indicators.

ggplot(data = ire\_counties\_df2 , aes(area = Population , fill = colour, subgroup = Province)) +  
   
geom\_treemap(colour = "white", size = 0.5\*.pt, alpha = NA) +  
   
 geom\_treemap\_text(aes(label = County), colour = "black" , size =10, place = "topleft",fontface = "bold",padding.x = grid::unit(1.5, "mm"),padding.y = grid::unit(1.5, "mm")) +  
   
geom\_treemap\_text(aes(label = format(Population, nsmall=0, big.mark=",",trim=TRUE)), color = "black", size = 8, place = "topleft", min.size = 3, padding.x = grid::unit(1.5, "mm"), padding.y = grid::unit(15, "points"))+  
  
  
 #geom\_treemap\_text(aes(label = Area), color = "black", size = 8, place = "topleft", fontface = "bold", padding.y = grid::unit(13, "points")) +  
  
 geom\_treemap\_subgroup\_border(colour = "white", size =0.5) +   
   
 geom\_treemap\_subgroup\_text(grow = FALSE, colour = "#FAFAFA", size = 36, place ="bottomleft", fontface = "bold", alpha = 0.5) +  
   
 scale\_fill\_identity()+  
   
 coord\_cartesian(clip = "off") +  
 guides(colour = "none", fill = "none")



Here is an example of a tree map using widely used treemap conventions.

The principle difference from the last example is that the area of each province is outlined by thick grey borders and expanded text labels

It is also typical that the font size expands or contracts to fill the space available in sub-category tiles, in this case the tiles representing counties

While this graphic easier to achieve (it is essentially the default in the treemapify library ), it has a number of issues

The province labels intrude upon the county areas and labels.

The varying size of the county labels conflict with the tile area size. For example, the label for Dublin has the same size as the label for Westmeath, one of the smallest tiles

However, with a single hue, it is arguably easier to compare colour gradient values across all tiles.

library(RColorBrewer)  
   
  
ggplot(data = ire\_counties\_df2 , aes(area = Population , fill = Area, subgroup = Province)) +  
   
geom\_treemap(colour = "white", size = 0.15\*.pt, alpha = 1) +  
   
geom\_treemap\_text(aes(label = County), colour = "black" , grow = FALSE, place = "centre",fontface = "italic", reflow=T,padding.x = grid::unit(1.5, "mm"),padding.y = grid::unit(1.5, "mm")) +  
   
   
 #geom\_treemap\_text(aes(label = Area), color = "black", size = 8, place = "topleft", fontface = "bold", padding.y = grid::unit(13, "points")) +  
   
 geom\_treemap\_subgroup\_border(colour = "darkgrey", size =6) +   
   
 geom\_treemap\_subgroup\_text(grow = TRUE, place = "centre", colour = "#FAFAFA", min.size =0, alpha = 0.3) +  
   
#scale\_fill\_distiller(palette ="Blues", direction = 1) +  
   
scale\_fill\_gradient( low = "#56B1F7", high = "#132B43")+  
   
 coord\_cartesian(clip = "off") +  
 guides(colour = "none", fill = "none")

