UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Grado
Estado del arte de visualización circular de datos con R

Autor: Pablo García Lacalle
Tutor: Juan José Cuadrado Gallego

TRIBUNAL

Presidente:
Vocal 1:
Vocal 2:

Calificación:

Índice

1. Introducción **5**

2. Ańalisis de los Diagramas **5**
2.1. Diagrama de Anillos . . . 5
2.2. Diagrama de Barras Radiales . . . 5
2.3. Diagrama en Espiral . . . 5
2.4. Diagrama de Columnas Radiales . . . 6
2.5. Diagrama de Sectores . . . 6
2.6. Diagrama Rose of Nightingale . . . 6
2.7. Diagrama de Cuadricula Circular Ordenada . . . 7
2.8. Diagrama de Cuadricula Circular Desordenada . . . 7
2.9. Diagrama Rayos de Sol . . . 7
2.10. Diagrama Radar . . . 8
2.11. Diagrama de Columnas Radiales Separadas . . . 8
2.12. Diagrama Flow . . . 8
2.13. Diagrama de Burbujas . . . 8
2.14. Diagrama de Mapa de Árbol Circular . . . 9
2.15. Diagrama de Mapa de Árbol Voronoi . . . 9
2.16. Diagrama de Planos Circulares . . . 9
2.17. Diagrama de Carreteras Cirular . . . 9
2.18. Diagrama de Mapa de Esferas . . . 10
2.19. Dendograma Circular . . . 10
2.20. Diagrama de Cuerdas . . . 10
2.21. Diagrama de Redes . . . 10

3. Visualización en R **10**
3.1. Diagrama de anillos . . . 10
3.1.1. Paquete CMplot . . . 11
3.1.2. Paquete ggplot2Donut . . . 11
3.1.3. Paquete ggplot2Pie . . . 11
3.1.4. Paquete ggplot2 . . . 11
3.1.5. Comparaciones . . . 11
3.2. Diagrama de Barras Radiales . . . 11
3.2.1. Paquete ggplot2 . . . 11
3.2.2. Paquete Plotrix . . . 15
3.2.3. Paquete circlize . . . 17
3.2.4. Comparaciones . . . 18
3.3. Diagrama Espiral . . . 18
3.3.1. Paquete ggplot2 . . . 18
3.3.2. Comparaciones . . . 22
3.4. Diagrama de Columnas Radiales . . . 22
3.4.1. Paquete ggplot2 . . . 22
3.4.2. Paquete plotrix . . . 25
3.4.3. Paquete cplots . . . 26
3.4.4. Comparaciones . . . 28
3.5. Diagrama de Sectores . . . 28
3.5.1. R Básico . . . 28
3.5.2. Paquete ggplot2 . . . 29
3.5.3. Paquete plotrix . . . 30
3.5.4. Paquete rCharts . . . 32
3.5.5. Paquete plotly . . . 33
3.5.6. Paquete ggiraphExtra . . . 33
3.5.7. Comparaciones . . . 34
3.6. Diagrama de Rose of Nightingale . . . 34

3.6.1. Paquete ggplot2 . . . 34
3.6.2. Paquete ggiraphExtra . . . 39
3.6.3. Comparaciones . . . 41
3.7. Diagrama de Cuadricula Circular Ordenada . . . 41
3.7.1. Paquete circlize . . . 41
3.7.2. Comparaciones . . . 41
3.8. Diagrama Rayos de Sol . . . 41
3.8.1. Paquete ggraph . . . 41
3.8.2. Paquete plotly . . . 43
3.8.3. Paquete webr . . . 45
3.8.4. Paquete ggiraphExtra . . . 46
3.8.5. Comparaciones . . . 46
3.9. Diagrama Radar . . . 46
3.9.1. Paquete fmsb . . . 46
3.9.2. Paquete plotly . . . 47
3.9.3. Paquete plotrix . . . 48
3.9.4. Paquete ggiraphExtra . . . 50
3.9.5. Comparaciones . . . 52
3.10. Diagrama de Columnas Circulares Separadas . . . 52
3.10.1. Paquete ggplot2 . . . 52
3.10.2. Paquete circlize . . . 52
3.10.3. Comparaciones . . . 52
3.11. Diagrama Flow . . . 52
3.11.1. Paquete ggplot2 . . . 52
3.11.2. Paquete circlize . . . 52
3.11.3. Paquete ggiraphExtra . . . 52
3.11.4. Comparaciones . . . 52
3.12. Diagrama de Burbujas Circular . . . 52
3.12.1. Paquete ggraph-ggforce . . . 53
3.12.2. Paquete packcircles . . . 54
3.12.3. Comparaciones . . . 55
3.13. Diagrama de Mapa de Árbol Circular . . . 55
3.13.1. Paquete ggraph . . . 55
3.13.2. Paquete ciclrpackeR . . . 57
3.13.3. Comparaciones . . . 59
3.14. Diagrama Mapa de Árbol Voronoi . . . 59
3.14.1. Paquete voronoiTreemap . . . 59
3.14.2. Comparaciones . . . 62
3.15. Diagrama de Mapa de Carreteras . . . 62
3.16. Dendograma Circular . . . 63
3.16.1. Paquete ggraph . . . 63
3.16.2. Paquete circlize . . . 63
3.16.3. Paquete basicR . . . 63
3.16.4. Paquete ggplot2 . . . 63
3.16.5. Comparaciones . . . 64
3.17. Diagrama de Cuerdas . . . 64
3.17.1. Paquete Circlize . . . 64
3.17.2. Paquete chorddiag . . . 66
3.17.3. Comparaciones . . . 68
3.18. Diagrama de Redes . . . 68
3.18.1. Paquete qgraph . . . 69
3.18.2. Paquete igraph . . . 69
3.18.3. Comparaciones . . . 69

4. Diagramas no Realizables **69**

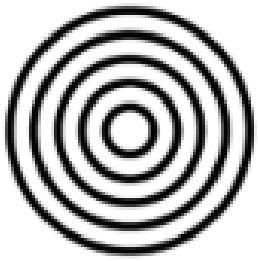
5. Comparativa **69**

Referencias **69**

1. Introducción

2. Ańalisis de los Diagramas

2.1. Diagrama de Anillos



2.2. Diagrama de Barras Radiales

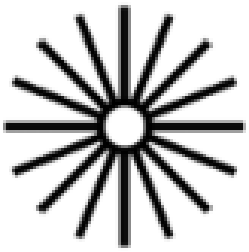
Un gráfico de Barras Radial o también llamado de Barras Circular es un gráfico de barras tradicional, pero en vez de usar un sistema de coordenadas cartesianas como el tradicional este utiliza coordenadas polares sobre el eje Y para dar el formato circular. En este diagrama las barras se colocan formando círculos concéntricos con el círculo que actúa como plantilla. Donde los divisores radiales sirven como escala de la longitud de dichas barras.
Con este diagrama existe un importante problema, el cual radica en que al tener cada barra un radio diferente según donde se posicionen en el círculo resultando las barras exteriores aparentemente más grande que los interiores aunque tengan el mismo valor. Por eso se suelen malinterpretar sus longitudes.
Debido a este problema este gráfico normalmente se usa para propositos estéticos y no prácticos como el tradicional.

2.3. Diagrama en Espiral

El gráfico en espiral expone los conjuntos de datos siguiendo una espiral de Arquímedes. Este diagrama se usa primordialmente para representar el paso del tiempo de unos datos, haciendo coincidir el origen de la espiral con el origen temporal del conjunto de datos y según se avanza en la espiral los datos avanzan en la variable tiempo.
Este tipo de gráfico es muy útil para representar como se actualizan los datos en cada rango de tiempo, para representar tendencias y para representar patrones periódicos.

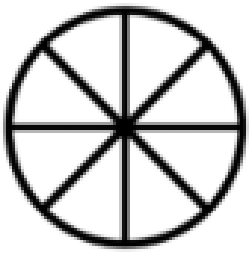
2.4. Diagrama de Columnas Radiales

Este diagrama llamado de Columnas Radiales o de Columnas Circular o de Estrella es otra de las formas de crear un gráfico de barras con coordenadas polares, pero esta vez con el eje X. Esta vez los divisores radiales se convierten en las barras y la longitud se mide con la distancia del centro al círculo concéntrico donde termine.
Este gráfico tiene un problema con la rapidez con la que se asimilan la información, ya que nunca va a ser tan rápido destacar una barra como en gráfico de barras con coordenadas cartesianas, por su uso de la escala radial.
Las barras en este diagrama se implementan de centro hacia el exterior, pero si se quiere representar valores negativos solo hace falta subir la escala 0 a concéntricos exteriores.



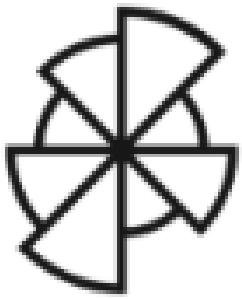
2.5. Diagrama de Sectores

El diagrama de Sectores o de Tarta es uno de los más conocidos y empleados en empresas, oficinas, en general por todos. Este gráfico funciona dividiendo el círculo en sectores (o porciones de tarta) dependiendo la longitud del arco del sector con el porcentaje del total que ocupen esos datos, haciendo concurrir el 100 porciento con los 360 grados del círculo.
Un porblema de este tipo es que no se pueden representar grandes cantidades de datos porque se perdería la ventaja de fácil identificación de la información. También es dificil hacer comparaciones entre varios gráficos de sectores.
Sin embargo su principal ventaja es que es muy útil para visualizar la distribución proporcional del conjunto de datos.

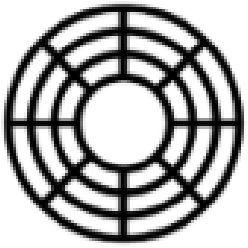


2.6. Diagrama Rose of Nightingale

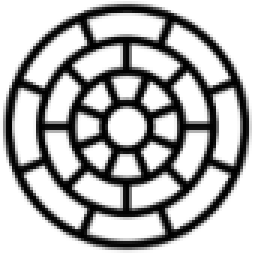
Es gráfico Rose of Nightingale o de Área Polar debe su nombre al médico Florence Nightingale que sirvió en la guerra de crimea para dar a visualizar las muertes evitables durente esa guerra. Se construye como un gráfico de sectores apilados donde cada segmento es un porcentaje del total.
Para representar este gráfico hay que tener en cuenta que los segmentos no se miden en longitud como el de Columnas Radiales, sino que hay que usar el valor del área del segmento.
Sin embargo, este gráfico tiene un problema principal, el cual consiste en que en los sectores más externos se malinterpreta su valor debido al cálculo del área del segmento.



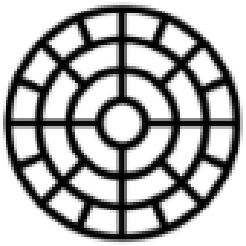
2.7. Diagrama de Cuadricula Circular Ordenada



2.8. Diagrama de Cuadricula Circular Desordenada

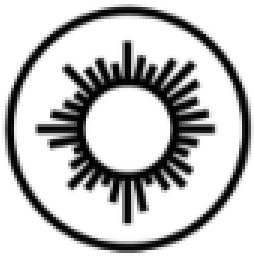


2.9. Diagrama Rayos de Sol



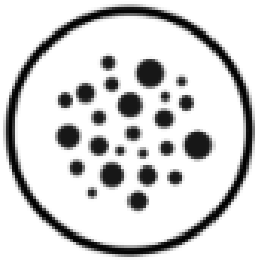
2.10. Diagrama Radar

2.11. Diagrama de Columnas Radiales Separadas

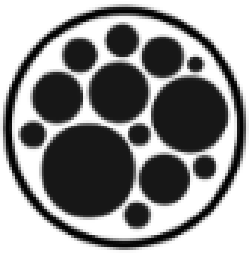


2.12. Diagrama Flow

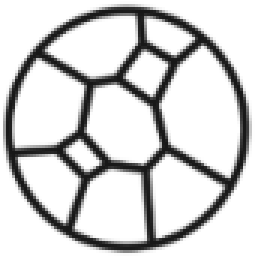
2.13. Diagrama de Burbujas



2.14. Diagrama de Mapa de Árbol Circular



2.15. Diagrama de Mapa de Árbol Voronoi

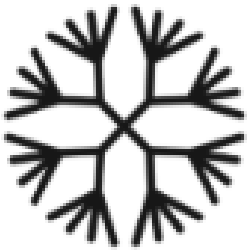


2.16. Diagrama de Planos Circulares

2.17. Diagrama de Carreteras Cirular

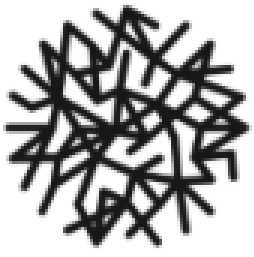
2.18. Diagrama de Mapa de Esferas

2.19. Dendograma Circular



2.20. Diagrama de Cuerdas

2.21. Diagrama de Redes



3. Visualización en R

3.1. Diagrama de anillos

Explicación.

3.1.1. Paquete CMplot

Ejemlpo.

3.1.2. Paquete ggplot2Donut

Ejemlpo.

3.1.3. Paquete ggplot2Pie

Ejemlpo.

3.1.4. Paquete ggplot2

Ejemlpo.

3.1.5. Comparaciones

3.2. Diagrama de Barras Radiales

Hay varias formas de aproximarse para realizar este tipo de diagramas. Una de ellas consiste en realizar un diagrama de barras tradicional y convertir su eje y al formato radial, la otra forma se realiza creando un círculo como plantilla y arcos como las barras (círculos concéntricos).
En este apartado se encuentran un paquete para la primera fórmula y dos para la segunda, pero todos los paquetes dan un gráfico muy similar aunque unos dan más posiblilidades de personalización, pero son más complejos.

3.2.1. Paquete ggplot2

En el punto anterior ya se ha visto este paquete y durante todos los demás puntos se demostrará lo versátil que es. En este tipo de grafo lo que hay que hacer es un buen diagrama de barras tradicional con este paquete y aplicarle la coordenada polar, la cual transforma la coordenada deseada de formato lineal a circular.
Primero se preparan los datos y se carga de la libreria:

```
Category <- c("Electronics", "Appliances", "Books", "Music", "Clothing",
              "Cars", "Food", "Hygiene",
              "Health/OTC", "Hair Care")
Percent <- c(81, 77, 70, 69, 69, 68, 62, 62, 61, 60)
internetImportance<-data.frame(Category,Percent)
```

Se insertan las etiquetas.

```
internetImportance$Category <-
  paste0(internetImportance$Category," - ",internetImportance$Percent,"%")
```

Y se ordenan de mayor a menor por como trabaja la libreria al dibujar el diagrama de fuera hacia dentro.

```
internetImportance$Category <-
  factor(internetImportance$Category,
         levels=rev(internetImportance$Category))
library(ggplot2)
```

Y después de esto se realiza el plot del diagrama circular:
Gracias a esta función se prepara el espacio para dibujar el gráfico con los parámetros básicos de los datos, y la estética del gráfico las categorías se representan en el eje x y el porcentaje en el eje y rellenando la figura geométrica que quiera representar en función de las categorías.

```
ggplot(internetImportance, aes(x = Category, y = Percent,
                               fill = Category)) +
```

Se va a elegir la figura geométrica de la barra para representar el gráfico que por ahora está en formato lineal.

```
  geom_bar(width = 0.9, stat="identity")  +
```

Después se transforma el eje y del formato lineal al formato radial.

```
  coord_polar(theta = "y") +
```

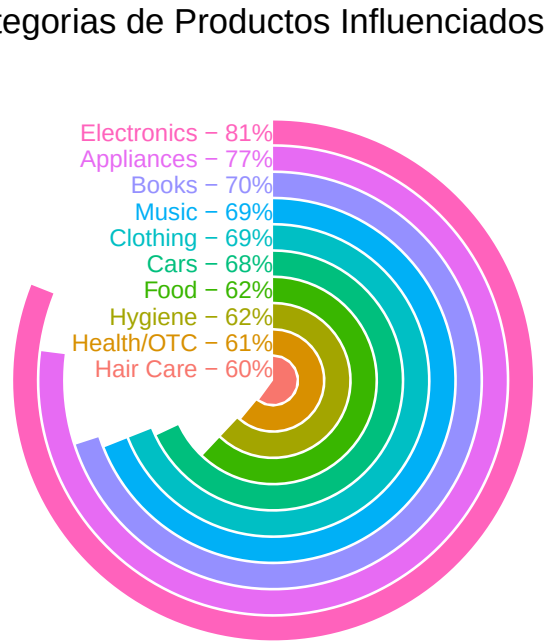
Para la estética del gráfico se eliminan las etiquetas de los ejes. Y representando los 360 grados del círculo al 100 % de los porcentajes.

```
  xlab("") + ylab("") +
  ylim(c(0,100)) +
```

Se establece el título y el texto de las barras, eliminando el fondo y la leyenda del gráfico para mejor representación.

```
  ggtitle("Categorias de Productos Influenciados por Internet") +
  geom_text(data = internetImportance, hjust = 1, size = 3,
            aes(x = Category, y = 0, label = Category, colour = Category)) +
  theme_minimal() +
  theme(legend.position = "none",
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_blank(),
        axis.text.y = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks = element_blank())
```

Y la figura queda como se ve a continuación.



Ahora se va a implementar uno de los grafos del libro [1, pág 76] , para ello se vuelven a preparar los datos como antes:

```
Name <- rev(c("ANTIMONY","INDIUM","SILVER","COPPER","TITANIUM","TANTALUM","PHOSPHORUS",
              "ALUMINIUM","GAS","OIL","COAL","AGRICUL. LAND","CORAL REEFS", "RAINFOREST"))
Value <- rev(c(8, 12, 17, 32, 44, 46, 76, 80, 35, 37, 42, 69, 88, 100))
Group <- rev(c("minerals","minerals","minerals","minerals","minerals","minerals",
           "minerals","minerals","fossil fuels","fossil fuels","fossil fuels",
           "ecosystems","ecosystems","ecosystems"))
stockCheck<-data.frame(Name,Value,Group)
#Etiquetas
stockCheck$Label <- paste0(stockCheck$Name," ")
```

Para poder tener un espacio vacio en el centro hay que incluir varias lineas vacias para que al dibujar la función las borre, pero dejando el espacio en blanco

```
len <- 5
df2 <- data.frame(Name = letters[1:len], Value = rep(NA, len),
                  Group = rep(NA, len), Label = rep("", len))
stockCheck <- rbind(stockCheck, df2)
#ordenar por grupos
stockCheck$Name <- factor(stockCheck$Name,
                          levels = rev(stockCheck$Name[order(stockCheck$Group)] ))
```

Y se dibuja el grafo de la misma manera que la anterior solo que ahora dejamos espacio en el interior para que se visualice mejor el grafo. Al ser de la misma forma que el anterior se va a mostrar solo las opciones que tenga de forma diferente este gráfico con respecto al anterior.

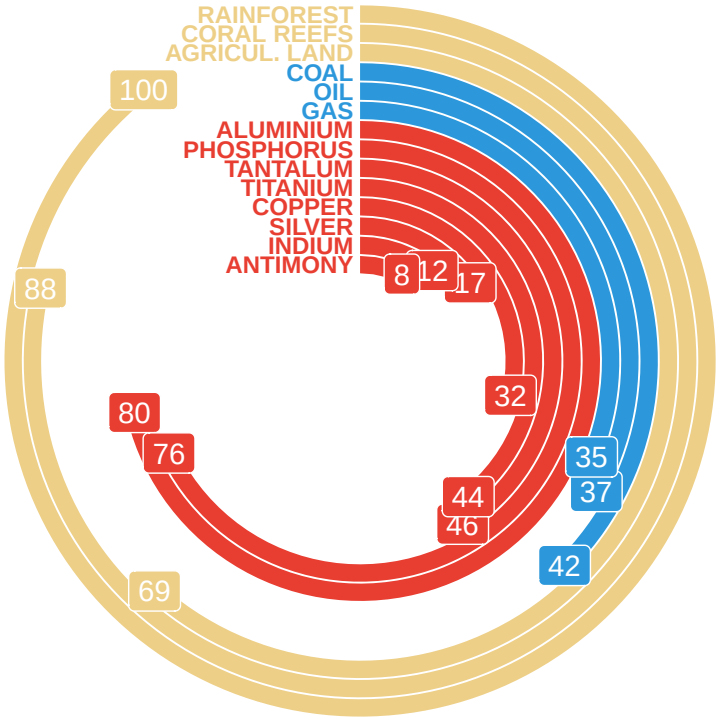
Una de las diferencias es en cuanto al anterior es la forma de crear el texto con el color según los grupos, y las etiquetas que se sitúan al final de las barras siendo cuadrados rellenos con el color del grupo.

```
  geom_text(data = stockCheck, hjust = 1, size = 3,
            aes(x = Name, y = 0, label = Label, colour = factor(Group),
                fontface = "bold")) +
  geom_label(aes(fill = factor(Group)), colour = "white") +
```

Y se cambian los valores del color para asemejarlo al gráfico que se intenta implementar, tiene que ser 4 colores debido a que hay un grupo de valores vacíos para crear el hueco en el medio.

```
  scale_fill_manual(values = c("#edcf87", "#2c98db","#e83d31","#e83d31" )) +
  scale_color_manual(values = c("#edcf87", "#2c98db","#e83d31","#e83d31" )) +
```

Resultando en el siguiente gráfico.



Existe un paquete que proporciona interactividad a los gráficos del paquete ggplot2, y suele servir mejor cuando hay una gran cantidad de datos.
El paquete `ggiraph` ayuda a crear la interactividad con algunas funciones que se unen al gráfico de ggplot2, en este caso se usa `geom-col-interactive` para crear las barras interactivas. Para que se represente bien hay que dar los valores aduacuados a las nuevas variables interactivas `tooltip` y `data-id`. Primero se necesita una gran cantidad de datos, aleatorios en este caso.

```
data <- data.frame(
  id=seq(1,30),
  value=sample( seq(70,100), 30, replace=T)
)
data$individual <- paste0( "Bar. ", data$id, "-", data$value,"%")
```

Y ahora asignar los parámetros adecuados al gráfico de ggplot2 con la función de columnas interactivas.

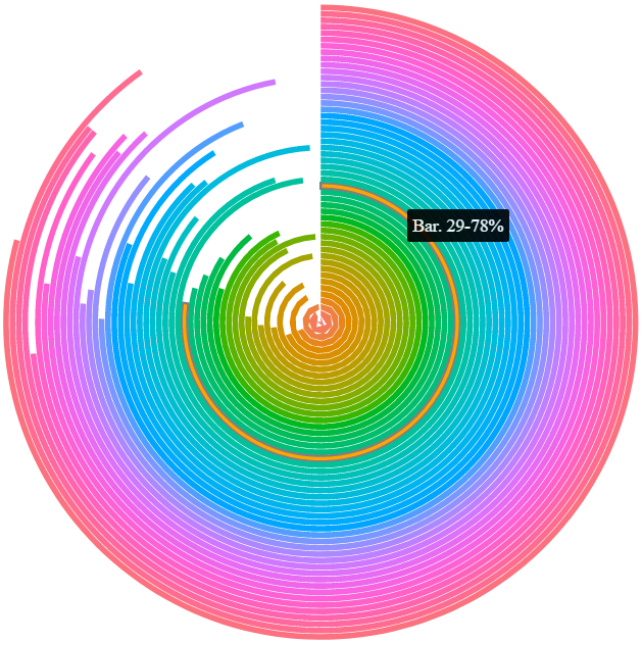
```
library(ggiraph)
p <- ggplot(data, aes( x = individual, y = value, tooltip = individual, label = value,
                                    data_id = individual, fill = individual) ) +
  geom_col_interactive() +
  coord_polar(theta = 'y') +
  xlab("") + ylab("") +
  ylim(c(0,101)) +
  theme_minimal() +
  theme(legend.position = "none",
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
```

```
        axis.line = element_blank(),
        axis.text.y = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks = element_blank())
```

Para la interactividad se necesita crear el objeto girafe (propio del paquete).

```
girafe(ggobj = p)
```

Dando como resultado el siguiente gráfico.



3.2.2. Paquete Plotrix

El paquete plotrix proporciona una serie de plots específicos los cuales ofrecen una fácil personalización (color, posición del texto...).
A lo largo del proyecto se observará dichos grafos usarse en los diferentes tipos de gráficos circulares. En la documentación se encuentran los métodos para dibujar este Diagrama de Barras Radiales `draw.circle` y `draw.arc` que se encargan de dibujar los círculos que se usaran de plantilla para que se dibujen las barras. La orden `draw.arc` necesita conocer en que ángulo comienza las barras y en cual terminan, además de los datos de las barras.

Primero llamar a la libreria.

```
library(plotrix)
```

Luego se crea la función para dibujar el diagrama.

```
circBarPlot <- function(x, labels, colors=rainbow(length(x)), cex.lab=1) {
```

Se establece el área para delimitar donde se dibuja el círculo.

```
  plot(0,xlim=c(-1.2,1.2),ylim=c(-1.2,1.2),type="n",axes=F, xlab=NA, ylab=NA)
```

Se guardan los datos de las barras en una variable.

```
  radii <- seq(1, 0.3, length.out=length(x))
```

Constituir la plantilla para los datos y el ángulo de inicio del diagrama.

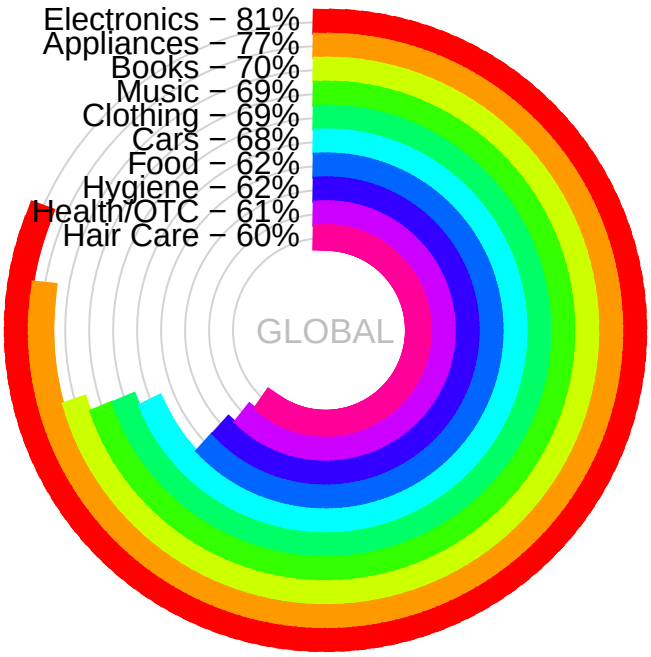
```
  draw.circle(0,0,radii,border="lightgrey")
  angles <- (1/4 - x)*2*pi
```

Por último se dibujan los arcos y la leyenda de cada barra.

```
  draw.arc(0, 0, radii, angles, pi/2, col=colors, lwd=130/length(x), lend=2, n=100)
  ymult <- (par("usr")[4]-par("usr")[3])/
    (par("usr")[2]-par("usr")[1])*par("pin")[1]/par("pin")[2]
  text(x=-0.02, y=radii*ymult, labels=paste(labels," - ", x*100, "%", sep=""),
      pos=2, cex=cex.lab)
```

Después se usara con los datos del primer gráfico.

```
circBarPlot(Percent/100, Category)
# texto en el centro
text(0,0,"GLOBAL",cex=1.1,col="grey")
```



3.2.3. Paquete circlize

El paquete circlize usa la filosofía de descomponer el círculo en figuras geométricas más simples como líneas y puntos implementando grafos circulares a partir de gráficos más simples,"implementa funciones gráficas de bajo nivel para agregar gráficos en las regiones de trazado circular".
Dichas funciones simples son: puntos, líneas, rectángulos, texto, etc. Y para organizar las regiones de trazado circulan existen las siguientes funciones: inicializar, actualizar, trazado, limpiar, etc.

En este apartado se utilizará las funciones de par (parámetros del grafo), inicializar (asigna los sectores al círculo), trazado(crear las regiones de trazado), segmentos(crea segmentos a partir de pares de puntos, en este caso las líneas del grafo), rectángulos (crea las barras), texto y los ejes.

Se vuelven a preparar los datos para este plot debido a como dibuja.

```
color = rainbow(length(Percent))
# circlize genera de dentro hacia afuera por lo que hay que revertir los vectores
Category <- rev(Category)
Percent <- rev(Percent)
color <- rev(color)
library(circlize)
```

Por último dibujamos el grafo
Se instancia el círculo a para que el diagrama empiece a los 90 grados.

```
circos.par("start.degree" = 90, cell.padding = c(0, 0, 0, 0))
```

Se crean los límites del diagrama igualando los 360 grados de la circunferencia al 100 de los valores de las barras.

```
circos.initialize("a", xlim = c(0, 100))
```

La función para dibujar con los parámetros.

```
circos.track(ylim = c(0.5, length(Percent)+0.5), track.height = 0.8,
             bg.border = NA, panel.fun = function(x, y) {
               xlim = CELL_META$xlim
```

Se dibujan los segmentos a partir de pares de puntos (las líneas centrales).

```
               circos.segments(rep(xlim[1], 10), 1:10,
                               rep(xlim[2], 10), 1:10,
                               col = "#CCCCCC")
```

Se crean las barras y la leyenda de cada una.

```
                circos.rect(rep(0, 10), 1:10 - 0.45, Percent, 1:10 + 0.45,
                            col = color, border = "white")
               circos.text(rep(xlim[1], 10), 1:10,
                           paste(Category, " - ", Percent, "%"),
                           facing = "downward", adj = c(1.05, 0.5), cex = 0.8)
               breaks = seq(0, 85, by = 5)
```

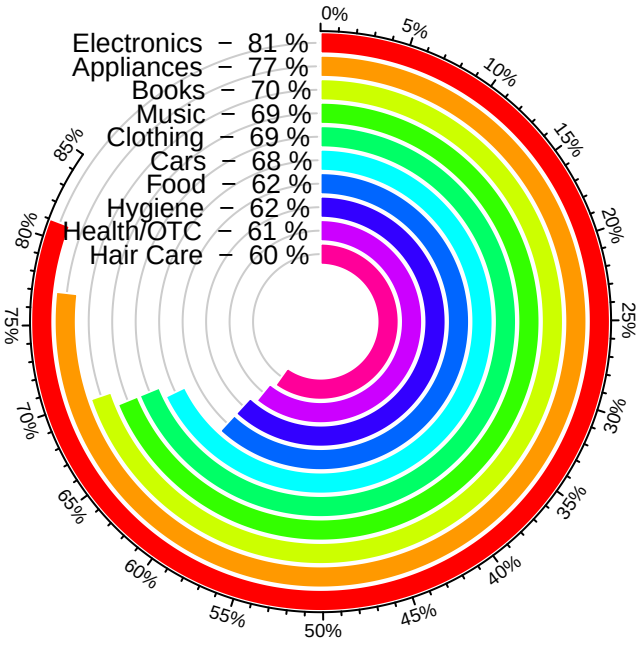
Se pone un circulo exterior para tener una mejor estética.

```
               circos.axis(h = "top", major.at = breaks, labels = paste0(breaks, "%"),
                           labels.cex = 0.6)
             })
```

Y por último para no crear conflictos se limpia el método.

```
circos.clear()
```

Resultando al final es:



3.2.4. Comparaciones

3.3. Diagrama Espiral

Para obtener este grafo habrá que tener unos datos adecuados que suelen implicar tres variables una para ver los grados del círculo, otro para su altura dentro del círculo y por último a veces puede haber otra variable para generar el tamaño de la figura geométrica que se decida (barra, círculo, etc.).

3.3.1. Paquete ggplot2

Se vuelve a ver este paquete debido a su versatilidad, pero ahora se depende mas de los datos que de paquete, el cual usa tres posibles generadores de barras: `rect, segment y title`
Estos datos son el movimiento de una persona durante 5 días, indicando la hora cuando se recogio dichos datos.

```
library(dplyr)
library(readxl)
library(ggplot2)
dat = read_excel("resource/Data1.xlsx")
head(dat, n=5)

## # A tibble: 5 x 4
##   ...1  Date1      Time  `Travel Time`
##   <chr> <chr>      <chr> <chr>
## 1 1     2016-09-04 13:11 34.65
## 2 2     2016-09-04 13:12 34.35
## 3 3     2016-09-04 13:13 33.2
## 4 4     2016-09-04 13:14 33.0166666666667
## 5 5     2016-09-04 13:15 33.25
```

Ahora se tratan los datos para poder visualizarlos, dar a las fechas un formato más manejable.

```
dat$time = with(dat, as.POSIXct(paste(Date1, Time), tz="GMT"))
```

Se obtienen las horas y los días.

```
dat$hour = as.numeric(dat$time) %% (24*60*60) / 3600
dat$day =  as.Date(dat$time)
```

Reformar el tiempo en movimineto pasando a numeros, acortando decimales y generar tramos manejables de datos con rango de 25 mins.

```
names(dat)[grep("Travel",names(dat))] = "TravelTime"
dat$TravelTime = as.numeric(dat$TravelTime)
dat.smry = dat %>%
  mutate(hour.group = cut(hour, breaks=seq(0,24,0.25), labels=seq(0,23.75,0.25),
                          include.lowest=TRUE),
         hour.group = as.numeric(as.character(hour.group))) %>%
  group_by(day, hour.group) %>%
  summarise(meanTT = mean(TravelTime)) %>%
  mutate(spiralTime = as.POSIXct(day) + hour.group*3600)
```

Para el primer grafo se usará el método `rect`
La plantilla con los datos.

```
ggplot(dat.smry, aes(xmin=as.numeric(hour.group), xmax=as.numeric(hour.group) + 0.25,
                     ymin=spiralTime, ymax=spiralTime + meanTT*1500, fill=meanTT)) +
```

Se crean las barras a cada rango de 25 minutos.

```
  geom_rect(color="grey40", size=0.2) +
```

Se ajustan los ejes de las horas y la altura donde empieza cada barra.

```
  scale_x_continuous(limits=c(0,24), breaks=0:23, minor_breaks=0:24,
                     labels=paste0(rep(c(12,1:11),2), rep(c("AM","PM"),each=12))) +
  scale_y_datetime(limits=range(dat.smry$spiralTime) + c(-2*24*3600,3600*19),
                   breaks=seq(min(dat.smry$spiralTime),max(dat.smry$spiralTime),"1 day"),
                   date_labels="%b %e") +
```

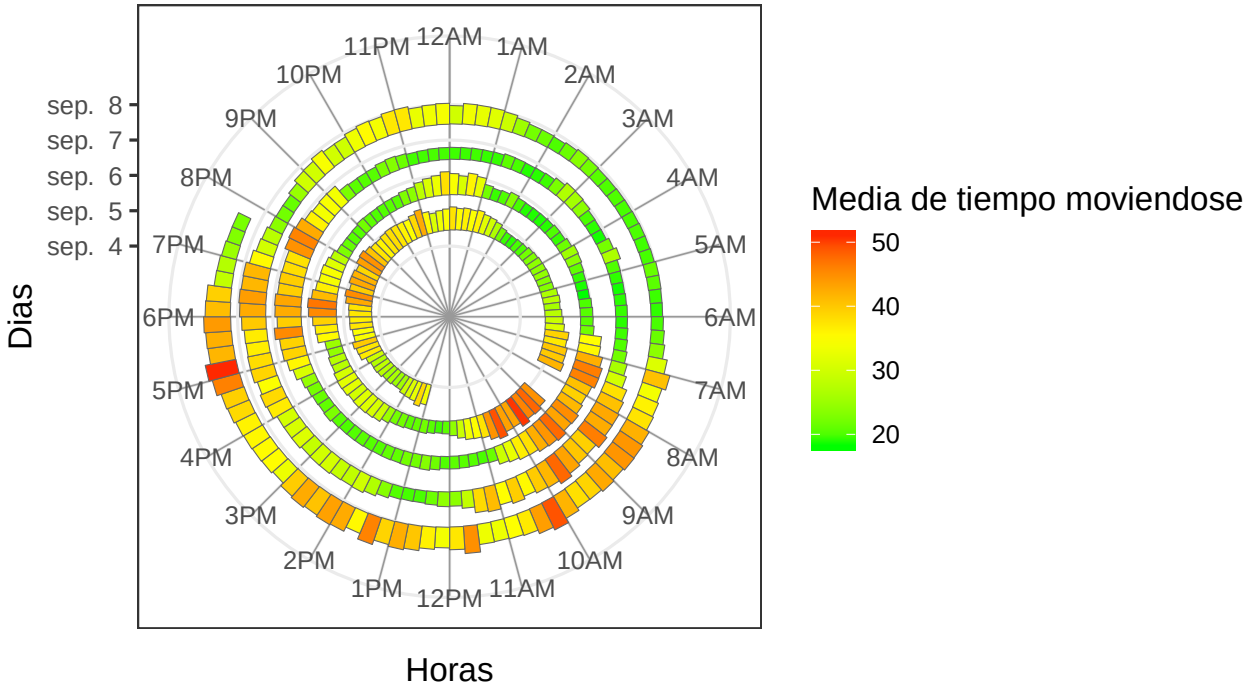
El color verde es para los rangos de poco movimiento, amarillo medio y rojo alto.

```
  scale_fill_gradient2(low="green", mid="yellow", high="red", midpoint=35) +
```

Y tansformar la coordenada x a radial y se ajustan las estéticas.

```
  coord_polar() +
  theme_bw(base_size=13) +
  labs(x="Horas",y="Dias",fill="Media de tiempo moviendose") +
  theme(panel.grid.minor.x=element_line(colour="grey60", size=0.3))
```

Resultando en:



Para el segundo grafo se usará el método **segment**.
Primero se usa la misma plantilla que el ejemplo anterior.

```
ggplot(dat.smry, aes(x=as.numeric(hour.group), xend=as.numeric(hour.group) + 0.25,
                     y=spiralTime, yend=spiralTime, colour=meanTT)) +
```

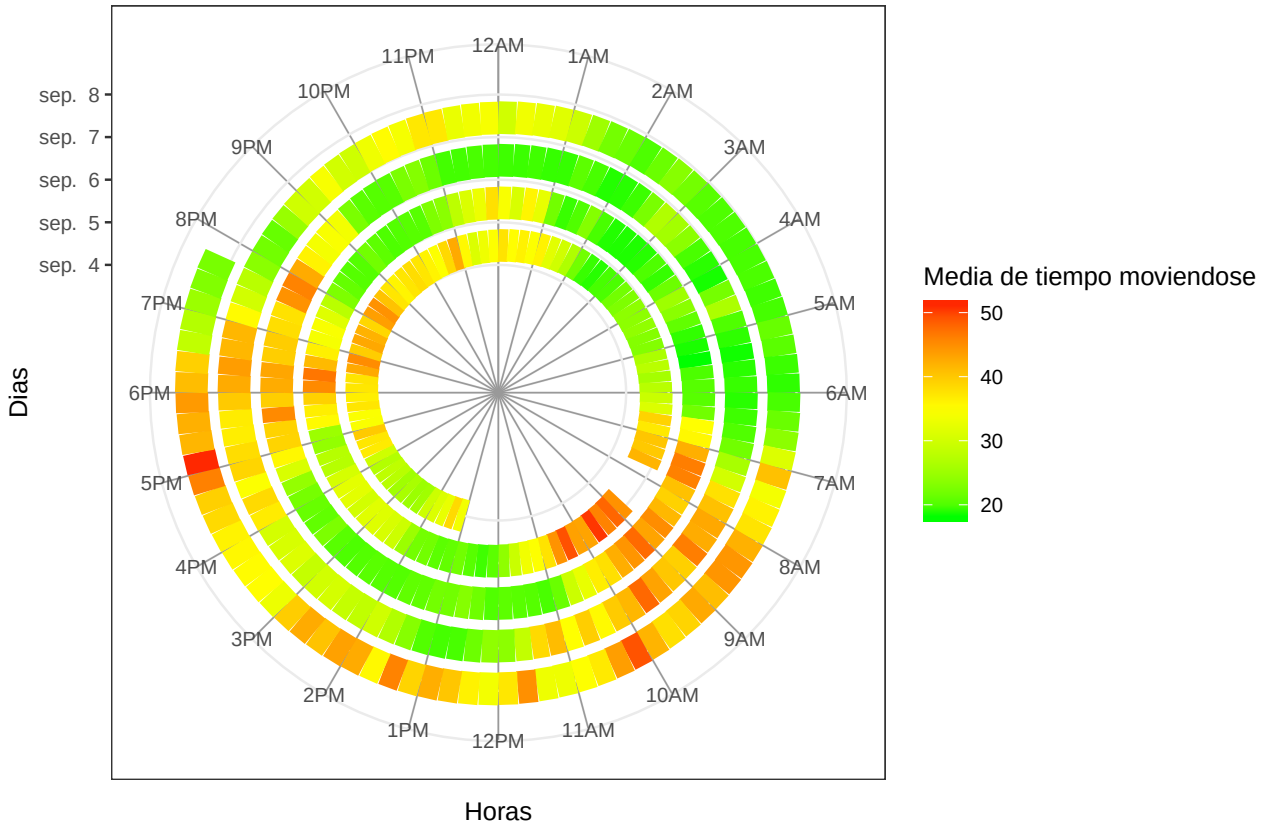
Luego en vez de usar las barras se van a usar segmentos, los cuales dejan una spiral con el mismo tamaño con la única diferencia de los colores.

```
  geom_segment(size=6) +
```

Y después de usar los mismos métodos que antes se obtiene el siguiente gráfico.

```
  scale_x_continuous(limits=c(0,24), breaks=0:23, minor_breaks=0:24,
                     labels=paste0(rep(c(12,1:11),2), rep(c("AM","PM"),each=12))) +
```

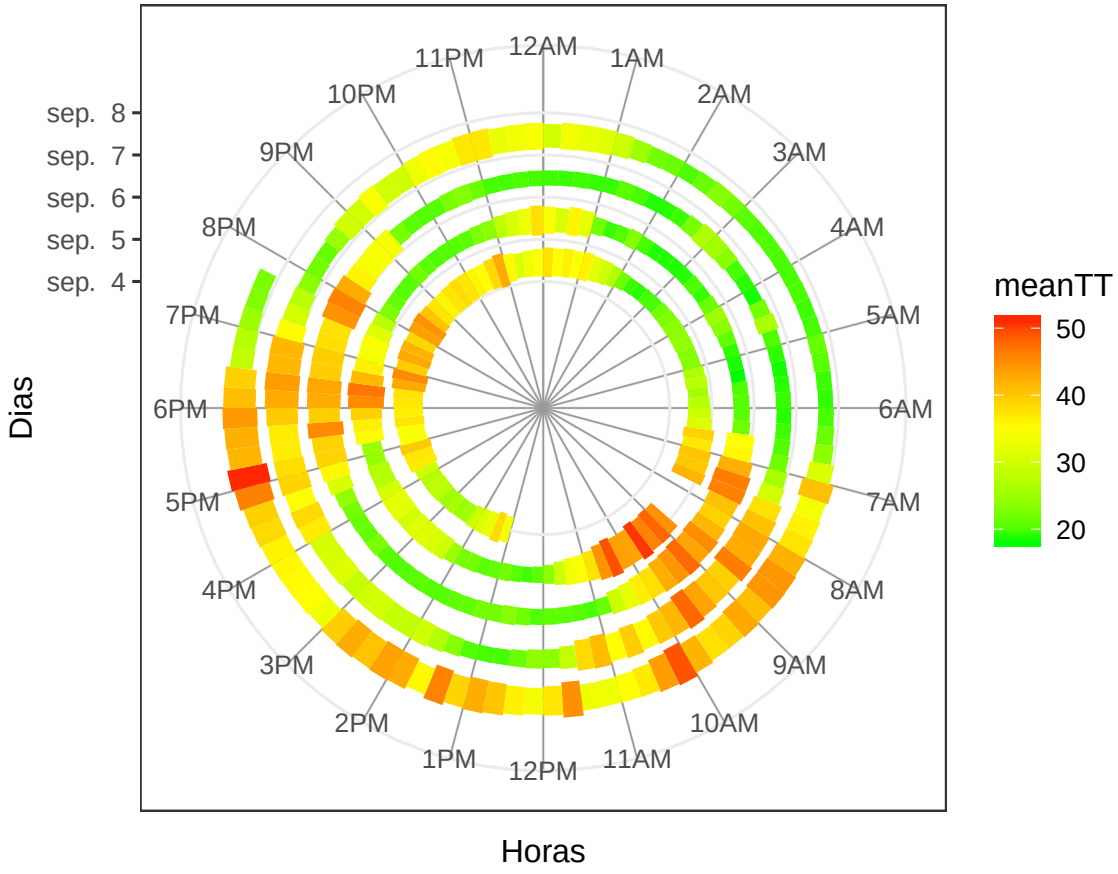
```
scale_y_datetime(limits=range(dat.smry$spiralTime) + c(-3*24*3600,0),
                 breaks=seq(min(dat.smry$spiralTime), max(dat.smry$spiralTime),"1 day"),
                 date_labels="%b %e") +
scale_colour_gradient2(low="green", mid="yellow", high="red", midpoint=35) +
coord_polar() +
theme_bw(base_size=10) +
labs(x="Horas",y="Dias",color="Media de tiempo moviendose") +
theme(panel.grid.minor.x=element_line(colour="grey60", size=0.3))
```



Y para el último grafo se usará el método `title`.
Con este método se puede variar su tamaño para dar una forma similar al primer ejemplo.

```
geom_tile(aes(height=meanTT*1800*0.9)) +
```

Dando el resultado de:



3.3.2. Comparaciones

3.4. Diagrama de Columnas Radiales

Para realizar este tipo de diagramas se afronta de una de las formas que se realizo en Subsección 3.2 creando un diagrama de barras tradicional y transformar el eje x en vez de el eje y.

3.4.1. Paquete ggplot2

Para realizar el plot se usa la misma forma que al crear el diagrama de barras radial, pero en esta ocasión la coordenada a transformar esta vez se la x.
Primero se va a crear los datos para representar en est gráfico.

```
data <- data.frame(
  id=seq(1,20),
  individual=paste( "Colum. ", seq(1,20), sep=""),
  group=c( rep('A', 5), rep('B', 8), rep('C', 5), rep('D', 2)) ,
  value=sample( seq(10,50), 20, replace=T)
)
library(ggplot2)
```

Para implementar este gráfico se escogen los datos, asignando los ejes correspondientes y la figura geométricas de la barra.

```
ggplot(data, aes(x=as.factor(id), y=value)) +
  geom_bar(stat="identity", fill=alpha("blue",0.2)) +
```

Esta vez se puede controlar el espacio del centro mediante la variable `ylim` en lugar de tener que rellenar los datos con valores vacíos para luego no representarlos.

```
  ylim(-10,60) +
```

Se transforma el eje x de lineal a radial y se hace coincidir el principio del gráfico con el principio del círculo (0 grados).

```
  coord_polar(start = 0) +
```

Y por último se ajusta la estética deseada.

```
  theme_minimal() +
  theme(
    axis.text = element_blank(),
    axis.title = element_blank(),
    panel.grid = element_blank(),
    plot.margin = unit(rep(-2,4), "cm")
  )
```

Quedando el gráfico de la siguiente forma.

Pero como se ha observado con anterioridad este paquete deja mucho espacio para la personalización de estos gráficos por lo que se va ha implementar el mismo gráfico que antes, pero con grupos con espacios diferenciados y etiquetas.
Para crear los espacios entre los grupos se va a a nadir al igual que en el diagrama de barras radiales barras con datos vacíos para que dejen el espacio al no ser representados.

```
library(tidyverse)
empty_bar <- 4
```

```
to_add <- data.frame( matrix(NA, empty_bar*nlevels(data$group), ncol(data)) )
colnames(to_add) <- colnames(data)
to_add$group <- rep(levels(data$group), each=empty_bar)
data <- rbind(data, to_add)
data <- data %>% arrange(group)
data$id <- seq(1, nrow(data))
```

Para la parte de las etiquetas habrá que poner el datos que se quiera visualizar en todas las columnas y ajustar su altura y ángulo

```
label_data <- data
number_of_bar <- nrow(label_data)
angle <- 90 - 360 * (label_data$id-0.5) /number_of_bar
label_data$hjust <- ifelse( angle < -90, 1, 0)
label_data$angle <- ifelse(angle < -90, angle+180, angle)
```

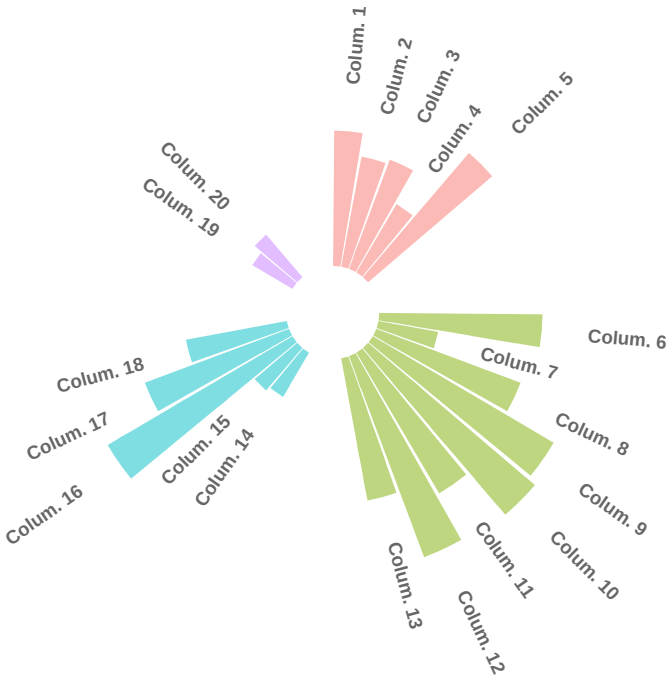
Para ahora crear el gráfico usamos las siguentes funciones.
Lo primero que es distinto será el hecho de que pinta las columnas dependiendo del grupo.

```
ggplot(data, aes(x=as.factor(id), y=value, fill=group)) +
  geom_bar(stat="identity", alpha=0.5) +
```

Por lo demás todo es igual, hasta que se a nade la última función. La cual genera las etiquetas del gráfico ajustadas a la altura y el ángulo.

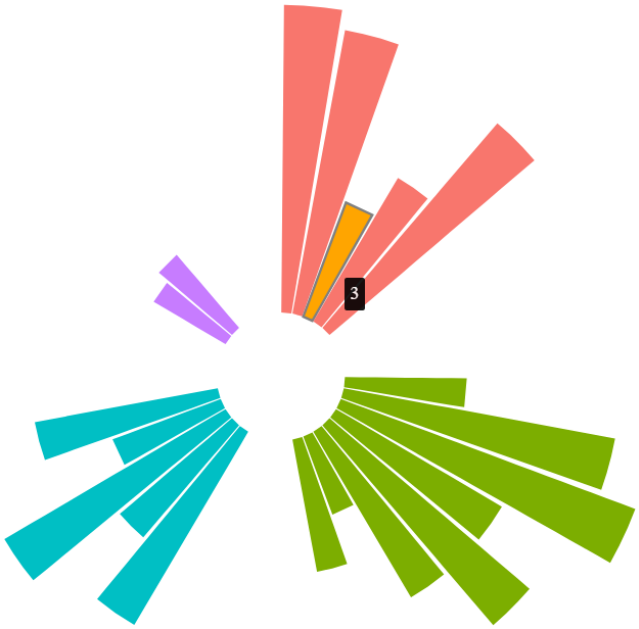
```
  geom_text(data=label_data, aes(x=id, y=value+10, label=individual, hjust=hjust),
      color="black", fontface="bold",alpha=0.6, size=2.5, angle= label_data$angle,
      inherit.aes = FALSE )
```

Resultando en este último gráfico



Existe una forma de realizar este mismo gráfico con interactividad con la ayuda de ggplot2 y ggiraph. Ya se ha explicado en otros digramas como, asi que en este solo se ense na el resultado.

```
library(ggiraph)
p <- ggplot(data, aes(x=as.factor(id), y=value, tooltip=as.factor(id),
                      data_id = as.factor(id), fill = group)) +
  geom_col_interactive() +
  ylim(-10,60) +
  coord_polar(start = 0) +
  theme_minimal() +
  theme(
    axis.text = element_blank(),
    axis.title = element_blank(),
    panel.grid = element_blank(),
    plot.margin = unit(rep(-2,4), "cm")
  )
girafe(ggobj = p)
```



3.4.2. Paquete plotrix

Como se dijo este paquete seía usado para otros gráficos más adelante por sus plots específicos. Y gracias a las funciones `polar plot` y `radial plot` se pude contar con él en este apartado.
Para ello se establecen los datos usados.
Primero la longitud de las columnas con números aleatorios.

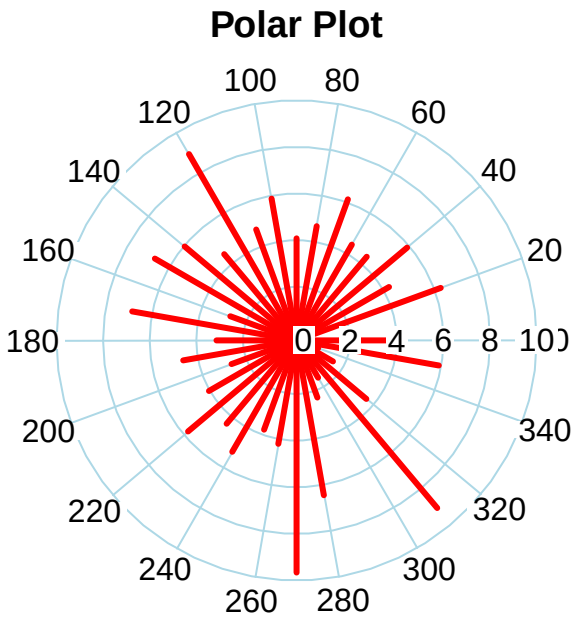
```
long <- c(rnorm(36)*2+5)
```

Luego la posición de cada columna en el círculo, siempre en grados.

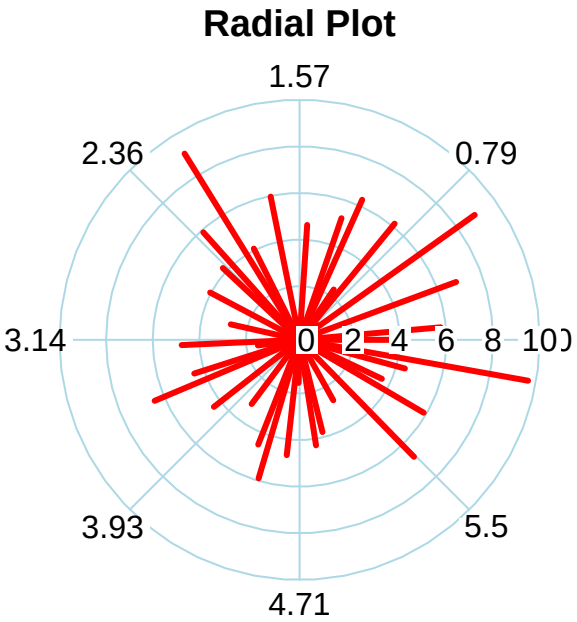
```
pos <- seq(0,350,by=10)
```

Para luego usar las dos funciones, las cuales son capaces de usar los mismos parámetros.

```
library(plotrix)
polar.plot(long, pos,main="Polar Plot",lwd=3,line.col=2,
           rad.col= "lightblue", grid.col= "lightblue",start = 0)
```




```
radial.plot(long, pos,main="Radial Plot",lwd=3,line.col=2,
            rad.col= "lightblue", grid.col= "lightblue", start = 0)
```



Cómo se ve, los gráficos no han salido iguales y es debido a que `polar plot` transforma la posición de los grados a radianes mediante la función `radial plot`, pero con `radial plot` la variable `pos` se interpreta como radianes y no grados.

3.4.3. Paquete cplots

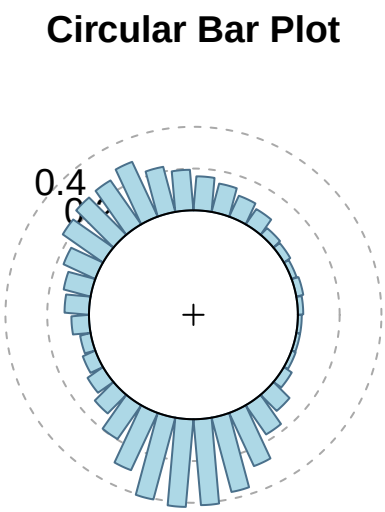
Este paquete tal y como dice su documentación proporciona algunos diagramas circulares provenientes de datos circulares, incluyendo diagramas de barras, de densidad, puntos apilados, etc.

Para reproducir este diagrama con este paquete se debe preparar los datos.

```
library(cplots)
library(circular)
x = c(rvonmises(2000, circular(2*pi/3), 2),
      rvonmises(2000, circular(3*pi/2), 5))
```

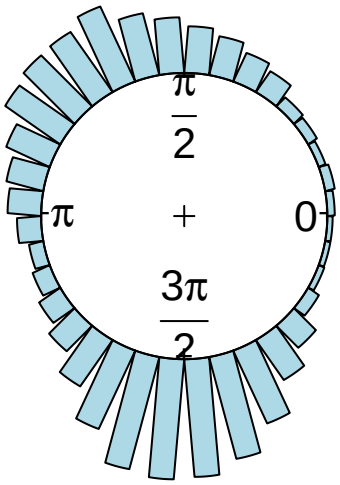
La función **rvonmises** está dentro del paquete circular y sirve para tener datos de forma circular donde se pasa el número de muestreos, la distribución de las muestras indicando en radianes donde se encuentra dentro del círculo y la concentración de las muestras a la distribución pasada.
Y para dibujar el gráfico de columnas radiales se usa **cbarplot**, elcual necesita los datos anteriores y se puede personalizar un poco como el color, la leyenda, etc.

```
cbarplot(x, radius=0.5, nlabels=3, col="lightblue", border="skyblue4")
```



Y si **nlabels** se iguala a 0 visualiza los radianes en el circulo.

```
cbarplot(x, radius=0.5, nlabels=0, col="lightblue")
```



3.4.4. Comparaciones

3.5. Diagrama de Sectores

Para crear este tipo de diagramas casi siempre se necesita de la misma estructura de datos, unas veces son simplemente unos valores, representándolos como un porcentaje del total de la suma de todos los valores, o pueden estar acompa nados de las categorías correspondientes.

3.5.1. R Básico

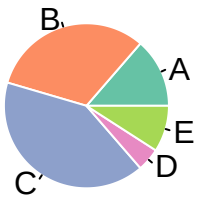
Una de las formar más sencillaspara representar los diagramas de sectores es gracias a la función `pie`. Esta función necesita de los propios valores dichos antes y se pueden a nadir más parámetros para si personalización como las etiquetas, colores y bordes.

```
Porcentaje <- c(3,7,9,1,2)
```

Para mejores colores y una paleta uniforme se opta por el paquete `RColorBrewer`.

```
library(RColorBrewer)
miPaleta <- brewer.pal(5, "Set2")

pie(Porcentaje, labels = c("A","B","C","D","E"), border = "white", col = miPaleta)
```



3.5.2. Paquete ggplot2

Una vez más se encuentra en el proyecto este paquete, por lo que como se ha comprobado antes este paquete es muy versátil para crear los diagramas, ofreciendo muy buena personalización.
Para la realización de estos diagramas circulares la función `coord polar` a resultado ser muy útil y esta vez al igual que en el diagrama de barras radiales se va a transformar la y de forma lineal a radial, pero la diferencia con aquel diagrama es que al dejar el eje x sin asignar cambia del diagrama de barras radiales al de sectores.
Para ello se vuelven a preparar los datos necesarios.

```
library(ggplot2)
data <- data.frame(
  group=LETTERS[1:5],
  value=c(13,7,9,21,2)
)
```

El problema de este diagrama con respecto a otros que usan `ggplot2` es la posicion de las etiquetas ya que solo con la función `geom label` no situa las etiquetas en su correcto lugar, por lo que hay que preparar eltexto para ello.

```
library(tidyverse)
data <- data %>%
  arrange(desc(group)) %>%
  mutate(Porcentaje = value / sum(data$value) *100) %>%
  mutate(ypos = cumsum(Porcentaje)- 0.5*Porcentaje )
```

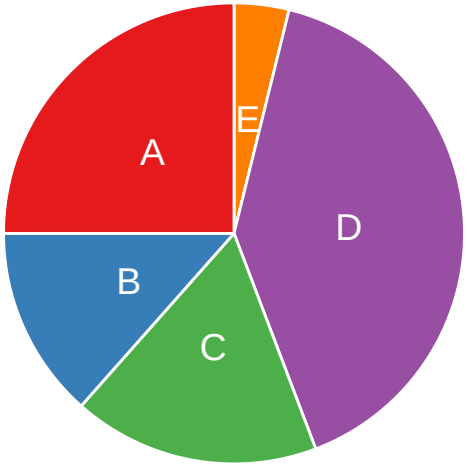
Y ahora se procede a la representación del gráfico pasando como parámetros los datos a usar y asignando al eje x nada (como se dijo antes), al eje y el procentaje y para los colores se obtienen en función del grupo. Se usa la figura geométrica de la barra para representar los datos y se convierte el eje y de lineal a radial.

```
ggplot(data, aes(x="", y=Porcentaje, fill=group)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) +
  theme_void() +
  theme(legend.position="none") +
```

Se coloca el texto en la posición anteriormente calculada y se ajusta la paleta de colores con el paquete usado antes.

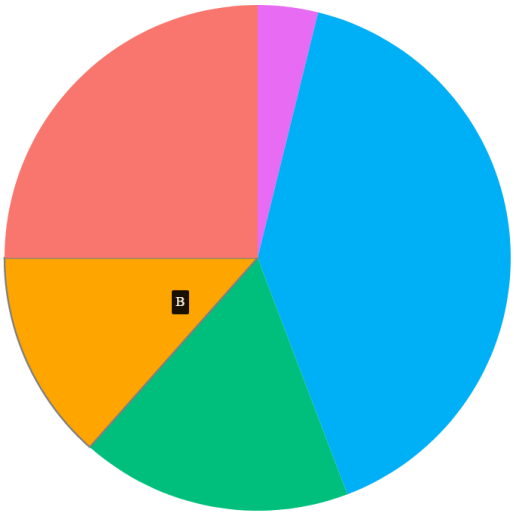
```
  geom_text(aes(y = ypos, label = group), color = "white", size=5) +
  scale_fill_brewer(palette="Set1")
```

Y así resulta el gráfico.



Al igual que muchos de los gráficos que usan ggplot2 se puede crear con el paquete ggiraph los gráficos de ggplot2 interactivos.

```
library(ggiraph)
p <- ggplot(data, aes(x="", y=value, fill=group, tooltip = group, data_id = group)) +
  geom_col_interactive() +
  coord_polar("y", start=0) +
  theme_void() +
  theme(legend.position = "none")
girafe(ggobj = p)
```



3.5.3. Paquete plotrix

Este paquete tiene dos posibles formas de representar los diagramas de sectores una en 2D y otra en 3D la cual no tiene ningún otro paquete que se haya investigado en este proyecto.

Primero se verá la forma de representar el gráfico en 2D. Para ello se usa la función `floating.pie` y `pie.labels`

para las etiquetas. SE obtienen los datos de los valores anteriores al igual que las etiquetas, pero para que el gráfico resulte algo distinto también se va a usar el parámetro `explode`.

Para ello se necesita la libreria y crear un expacio para luego dibujar los sectores.

```
library(plotrix)
plot(0,xlim=c(1.5,5),ylim=c(1,5),type="n",axes=FALSE,xlab="",ylab="", main = "2D")
```

Luego para las etiquetas se calculan las posiciones y angulos donde deben estar, los cuales proporciona ya la función `floating.pie`.

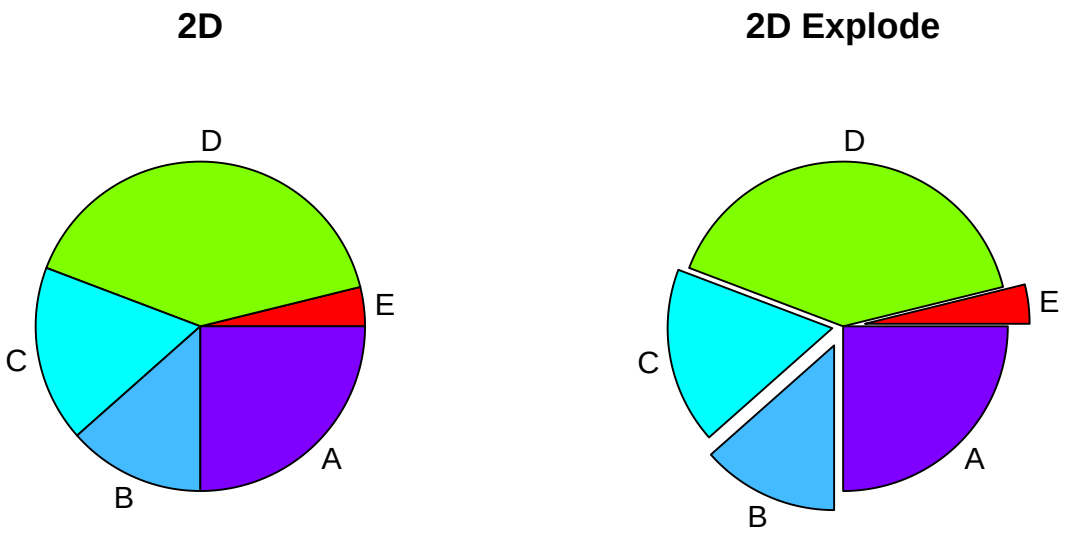
```
angulos <- floating.pie(3.25, 3, data$value, radius = 1.5,
                        col = c("#ff0000","#80ff00","#00ffff","#44bbff","#8000ff"))
```

Y para las etiquetas `pie.labels` necesita dichos angulos, la posición del centro del diagrama y las etiquetas.

```
pie.labels(3.25, 3, angulos, data$group, minangle = 0.2, radius = 1.6)
```

Después se reraliza el mismo trabajo para el parámetro`explode` pero incluyendo el vector con las distancias al centro (tanto en el gráfico como en las etiquetas).

```
pie.labels(3.25, 3, angulos, data$group, minangle = 0.2, radius = 1.6)
plot(0,xlim=c(1.5,5),ylim=c(1,5),type="n",axes=FALSE,xlab="",ylab="", main = "2D Explode")
angulos <- floating.pie(3.25, 3, data$value, radius = 1.5,
                        col = c("#ff0000","#80ff00","#00ffff","#44bbff","#8000ff"),
                        explode = c(0.2,0,0.1,0.2,0))
pie.labels(3.25, 3, angulos, data$group, minangle = 0.2, radius = 1.6,
           explode = c(0.2,0,0.1,0.2,0))
```



Y por último se realiza el mismo gráfico, pero en 3D gracias a la función `pie3D`. Esta forma requiere guardar el resultado en por ejemplo png.

Para ello se establece el nombre del archivo.

```
png(file = "3d_sectores.png")
```

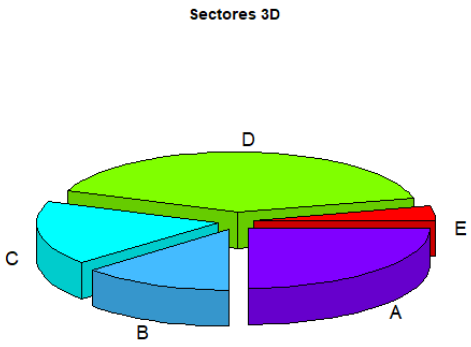
Y crear el gráfico con los valores iguales que antes.

```
pie3D(data$value, labels = data$group, explode = 0.1, main = "Sectores 3D",
      col=c("#ff0000","#80ff00","#00ffff","#44bbff","#8000ff"))
```

Y al final guardarlo resultando en el siguiente gráfico.

```
dev.off()
```

```
## pdf
##   2
```



3.5.4. Paquete rCharts

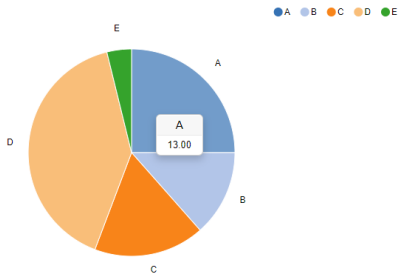
Tanto en este paquete como en el siguiente se va a traer unos diagramas de sectores interactivos, que al trabajo optar por el formato del papel y no página web al final de cada paquete se mostrará la forma de guardar los gráficos ern html para su posterior visualización.
Este paquete sirve para crear y personalizar gráficos interactivos en javascript transformandolos a partir de R. Para representar este gráfico se va a reutilizar los mismos valores que antes e implementarlos con la función `nPlot`.

```
library(rCharts)
p <- nPlot(value~group, data = data, type = 'pieChart')
```

El problema viene a la hora de que se muestre ya que p es una script de javascripts por lo que para mopstrarse con interactividad se necesita incluir en una pagina html, lo cual para este gráfico solo necesita de la etiqueta `<html>` al principio y `</html>` al final, copiando el script del resultado de la siguiente orden.

```
p$print(include_assets=T)
```

Dando como resultado la siguiente imagen que si crea el html sería interactiva.



3.5.5. Paquete plotly

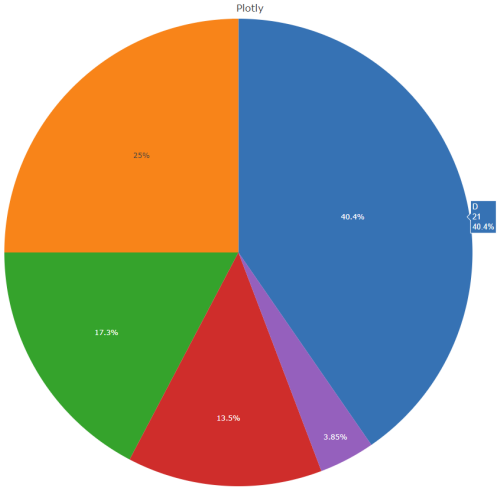
Para este otro paquete el gráfico también se va a realizar interactivo por lo que no se visualiza el gráfico resultante sino una foto de él.
Para realizarlo plotly ofrece la función `plot-ly`, básica para cualquier plot con este paquete, pero al darle el tipo pie reproduce el diagrama deseado. Como hasta ahora se va a seghuir utilizando los mismos datos.

```
library(plotly)
p <- plot_ly(data, labels = data$group, values = data$value, type = 'pie') %>%
  layout(title = 'Plotly',
         xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE))
```

Aunque esta vez la forma de guardarlo en una página web es mucho mas sencilla gracias al paquete `htmlwidgets`, mediante la siguiente orden.

```
library(htmlwidgets)
saveWidget(p, file = "plotly_pie.html")
```

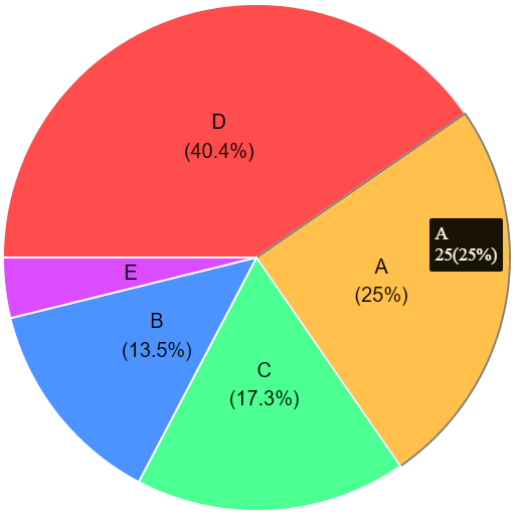
Resultando en el siguiente grafo (no interactivo salvo en el html).



3.5.6. Paquete ggiraphExtra

Otra forma de crear un diagrama de sectores interactivo es mediante este paquete ya que se dedica a crear gráficos interactivos a partir de los gráficos de ggplot2. Por lo que se usan los mismos datos.

```
library(ggiraphExtra)
ggPie(data, aes( pies=group, count=value), interactive = TRUE)
```



3.5.7. Comparaciones

3.6. Diagrama de Rose of Nightingale

Al recopilar las formas para realizar este diagrama se descubre que es la misma manera que si el diagrama constara de gráficos de columnas apiladas, pero dando el formato de unión entre barras que aporta el diagrama de sectores.

3.6.1. Paquete ggplot2

Otra vez en nuestro proyecto se cruza este paquete dislumbrando su versatilidad. El gráfico consistirá de nuevo en crear un diagrama normal (lineal) y transformarlo con la ayuda de `coord polar`, esta vez se realiza un gráfico de barras apiladas y transformar el eje x.
Primero necesita los datos para crear las barras apiladas.

```
data=read.csv("C:/Users/pablo/Documents/tfg/rose/rose.csv", sep = ";")
head(data)
```

```
##        MES  A   B C   D E   F G
## 1  January 10   9 8   7 6   5 4
## 2 February  5 4,5 4 3,5 3 2,5 2
## 3    March  7   6 5   4 3   2 2
## 4    April  9   7 8   7 6   5 4
## 5      May 11   8 4 3,5 3 2,5 2
## 6     June 13   9 5   4 3   2 2
```

Consiste de unos valores aleatorios ordenados por grupo y mes, pero necesita datos agrupados por solo una variabel que esta vez se usan los meses. Gracias a `reshape2` se consigue el agrupamiento por mes.

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
data1=data.frame(t(data))
data2=data1[2:8,]
colnames(data2)=month.name
data2$group=row.names(data2)
```

```
data3=melt(data2,id="group")
head(data3)

##   group variable value
## 1     A  January    10
## 2     B  January     9
## 3     C  January     8
## 4     D  January     7
## 5     E  January     6
## 6     F  January     5
```

Y ahora para crear el gráfico lo único que hay que hacer es crear el diagrama de barras apiladas y aplicarle la transformación, aparte de la estetica deseada.

Para ello se llama a la función `ggplot` con los datos de data3 y asignando el eje x a los mese, el y a los valores y los colores en función del grupo. Y se crea el diagrama de barras tradicional con `geom-bar(stat=ïdentity")`.

```
library(ggplot2)
ggplot(data3,aes(x=variable,y=value,fill=group))+
  geom_bar(stat="identity",width=1,colour="white",size=0.1)+
```

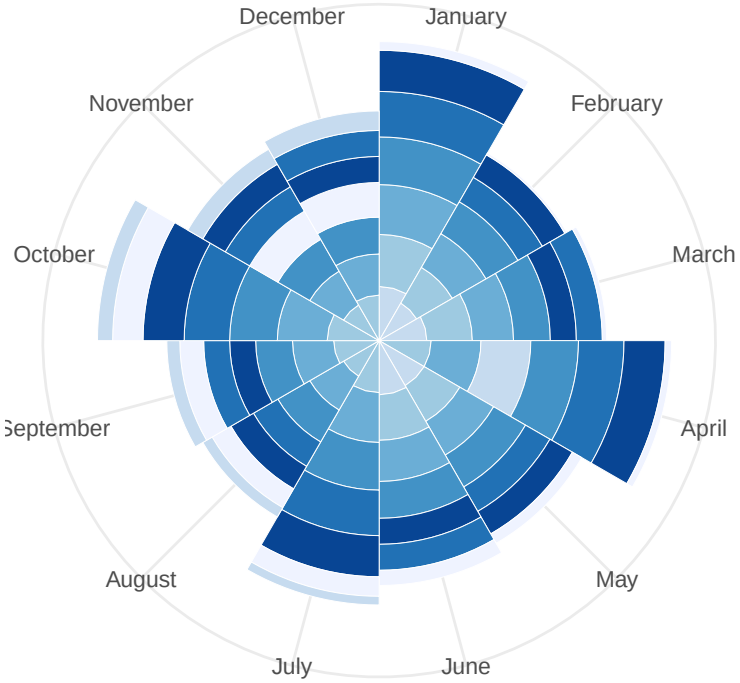
Ahora se transforma la coordenada x y se aplica la escala de colores deseada.

```
  coord_polar('x')+
  scale_fill_brewer(palette="Blues")+
```

Y por último la estetica del grafo deseada (sin leyenda, ejes,fondo blanco, etc).

```
  xlab("")+ylab("") +
  theme_minimal() +
  theme(legend.position = "none",
        axis.line = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank())
```

Dando como resultado el gráfico.



Ahora para ver el gráfico que le da nombre al diagrama se van a obtener los datos de la libreria `HistData` y se va a simular el grafo de Rose of Nightingale.

```
library(HistData)
data(Nightingale)
```

Como en este gráfico solo se muestra la fecha, las muertes y su causa se va a tener que transformar estos datos para que se puedan mostrar al igual que el gráfico que se muestra en [1, pág 113]

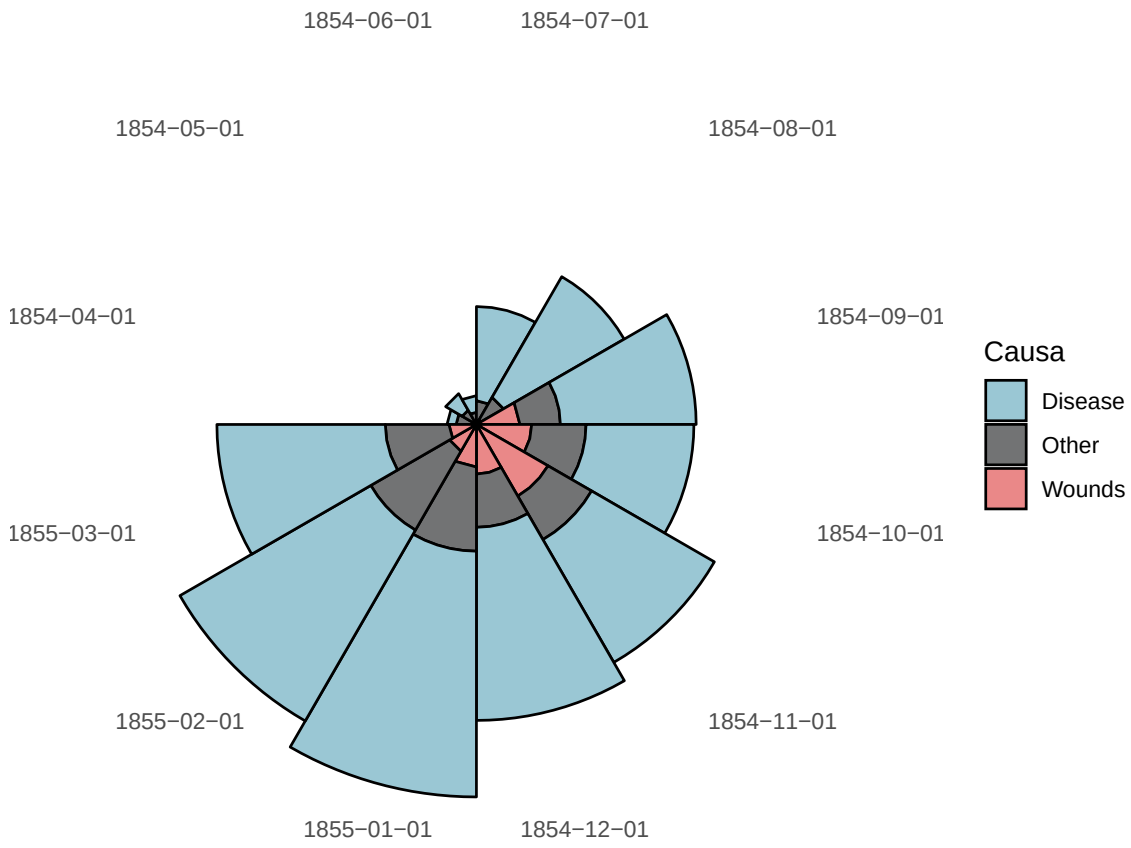
Primero solo se necesita la fecha y las muertes por enfermedad, heridas y otras causas y ordenarlas for la causa de la muerte al igual que en el ejemplo anterior.

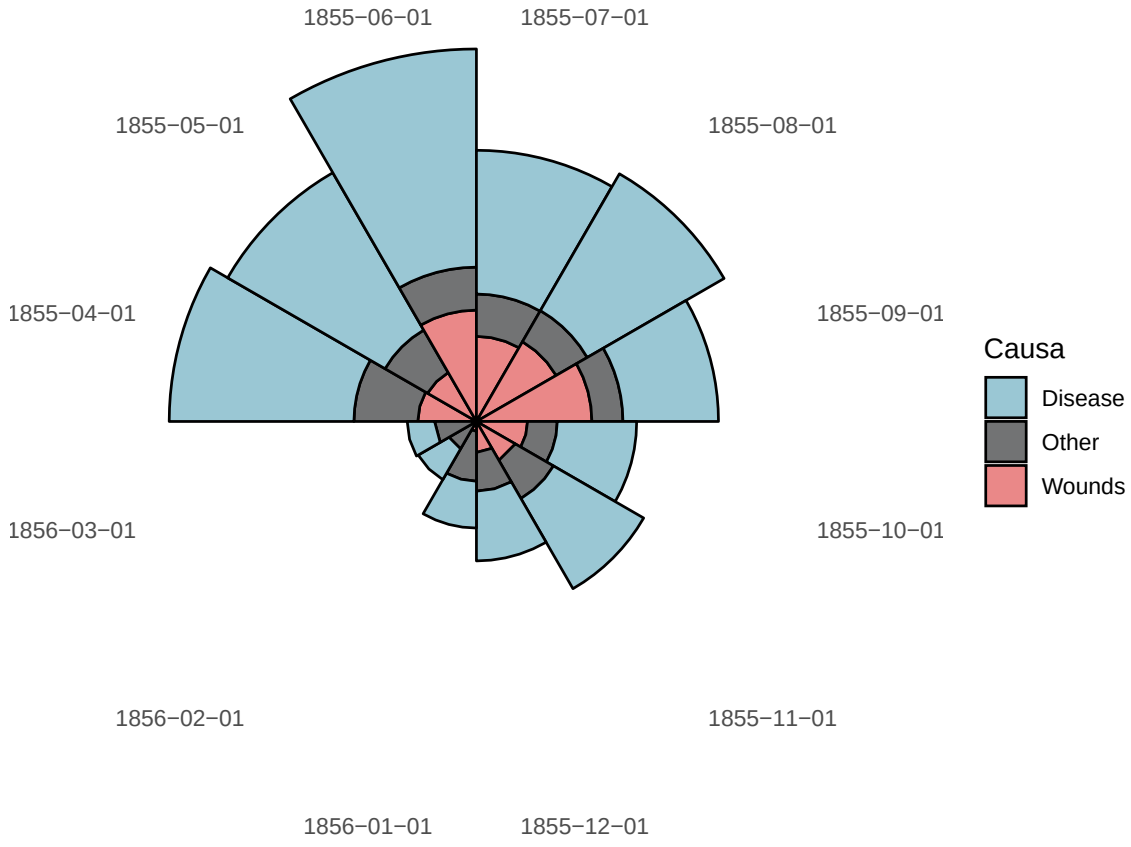
```
require(reshape)
Night<- Nightingale[,c(1,8:10)]
melted <- melt(Night, "Date")
names(melted) <- c("Fecha", "Causa", "Muertes")
melted$Causa <- sub("\\.rate", "", melted$Causa)
```

Luego al igual que el gráfico en el libro separa los datos para crear dos gráficos con la separación a partir de la fecha marzo de 1855.

```
Night <- melted
Night$Month <- format(Night$Fecha, "%b %Y")
Night1 <- subset(Night, Fecha < as.Date("1855-04-01"))
Night2 <- subset(Night, Fecha >= as.Date("1855-04-01"))
```

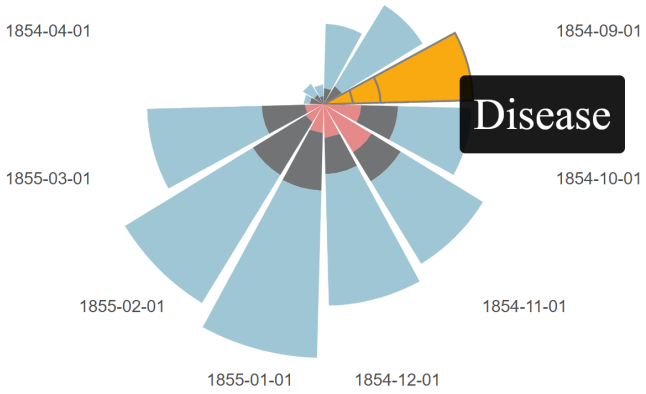
Ahora se dibujan los gráficos uno al lado del otro con la misma fórmula que el ejemplo anterior con el eje x las fechas, el y las muertes y el color en función de las causas.


1854−06−01
1854−07−01
1854−05−01
1854−08−01
1854−04−01
1854−09−01
1855−03−01
1854−10−01
1855−02−01
1854−11−01
1855−01−01
1854−12−01
Causa
Disease
Other
Wounds



Una forma de poder usar la interactividad con diagrama de ggplot2 en con ggiraph y en este caso la función `geom-col-interactive` en vez de `geom-bar`.

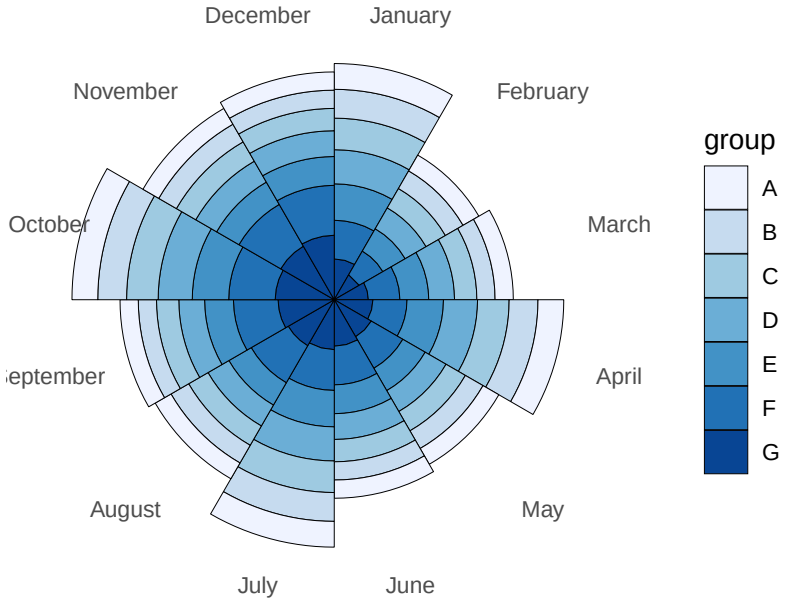
```
library(ggiraph)
p <- ggplot(Night1, aes(x = factor(Fecha), y = Muertes, fill = Causa, tooltip=Causa,
                        data_id =factor(Fecha))) +
  geom_col_interactive() +
  coord_polar(start = 3*pi/2) +
  xlab("") +  ylab("") +
  scale_y_sqrt() +
  scale_fill_manual(values = c("#9ac7d4", "#727374","#ea8888")) +
  theme_minimal() +
  theme(legend.position = "none",
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank())
girafe(ggobj = p)
```



3.6.2. Paquete ggiraphExtra

Este paquete permite crear gráficos interactivos a partir de los gráficos de ggplot2, el propio paquete tiene un dataframe llamado Rose con datos para este diagrama.
Una de las dos funciones que permite realizar esto es `ggRose` y necesita de los datos, la estética, la paleta de color, etc.
Primero se va a representar sin interartividad para el formato del papel, pero se acontinuación se muestra la forma de pasarlo a html para la interactividad.

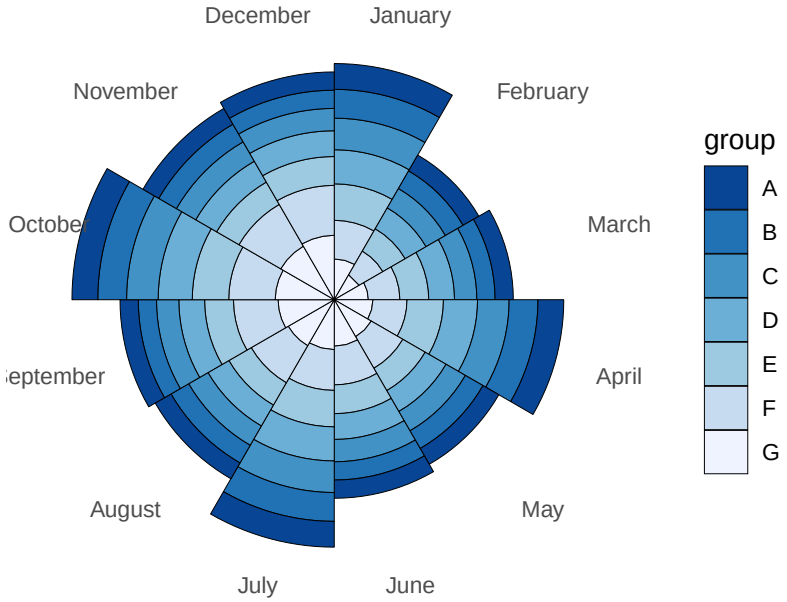
```
library(ggiraphExtra)
ggRose(rose,aes(x=Month,fill=group,y=value),stat="identity",interactive=FALSE
       , palette = "Blues") +
   xlab("") +  ylab("") +
  scale_y_sqrt() +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank())
```




```
p <- ggRose(rose,aes(x=Month,fill=group,y=value),stat="identity",interactive=TRUE
       , palette = "Blues")
library(htmlwidgets)
saveWidget(p, file = "rose_interactive.html")
```

Otro método para realizar el mismo diagrama es `ggBar` y como con el paquete ggplot2 lo que se realiza es de un diagrama de columnas apiladas tradicional transformando la coordenada x de lineal a radial. Como antes con la interactividad desactivada pero con la forma de verlo en html.

```
ggBar(rose,aes(x=Month,fill=group,y=value),stat="identity",polar=TRUE,palette="Blues"
      ,width=1, color="black",size=0.1,reverse=TRUE,interactive=FALSE)+
    xlab("") +  ylab("") +
    scale_y_sqrt() +
    theme_minimal() +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          axis.line = element_blank(),
          axis.text.y = element_blank(),
          axis.ticks = element_blank())
```




```
p <- ggBar(rose,aes(x=Month,fill=group,y=value),stat="identity",polar=TRUE,palette="Blues"
      ,width=1, color="black",size=0.1,reverse=TRUE,interactive=TRUE)
library(htmlwidgets)
saveWidget(p, file = "rose2_interactive.html")
```

3.6.3. Comparaciones

3.7. Diagrama de Cuadricula Circular Ordenada

Explicación.

3.7.1. Paquete circlize

Ejemlpo.

3.7.2. Comparaciones

3.8. Diagrama Rayos de Sol

En este apartado se encuentra una de las formas de representar un diagrama de árbol con formato circulas. Este diagrama se organiza en niveles, donde el nivel interior es la raiz del árbol y a medida que se escala hacia el exterior del círculo se encuentran los hijos hasta llegar a los nodos hoja.
Este diagrama como es una de las formas de representar un árbol con formato circular necesita una estrutura de datos jerárquica para poder implementarse (raiz y hojas ó padres e hijos).

3.8.1. Paquete ggraph

Este paquete surge de la necesidad de crear mejores gráficos de redes que los que ofrece ggplot2, por lo que se creo este paquete como una ampliacion de la API de ggplot2 para que ofrezca una mejor visualización de gráficos de capa por capa.
Para el ejemplo se va a necesitar unos datos jerarquicos, los cuales se obtienen de la base de datos flare siendo apoyada por la libreria `igraph` para transformarla en datos que entienda ggraph.

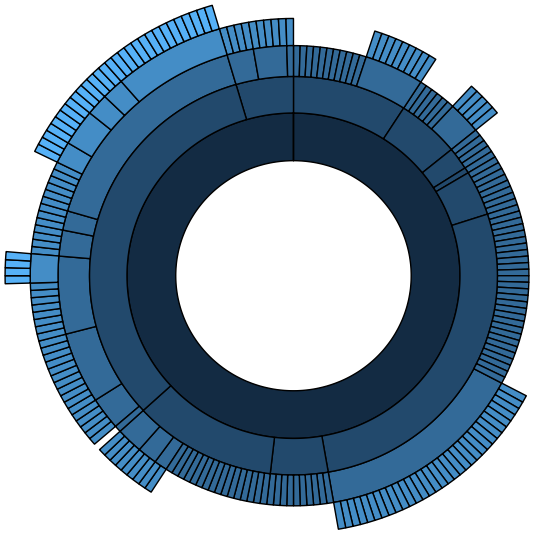
```
library(ggraph)
ramas <- flare$edges
vertices <- flare$vertices
library(igraph)
mygraph <- graph_from_data_frame( ramas, vertices=vertices)
```

Para preparar la plantilla para el gráfico se necesitan los datos, el tipo particion y establecerlo como circular.

```
ggraph(mygraph, 'partition', circular = TRUE) +
```

Luego la función que crea el gráfico se elige **geom-node-arc-bar** ya que dibuja los nodos del grafo como barras la cuales se van a colorear en función de la profundidad. Y para mejor estética se quitan el fondo y la leyenda.

```
  geom_node_arc_bar(aes(fill = depth), size = 0.25) +
  theme_void() +
  theme(legend.position="FALSE")
```

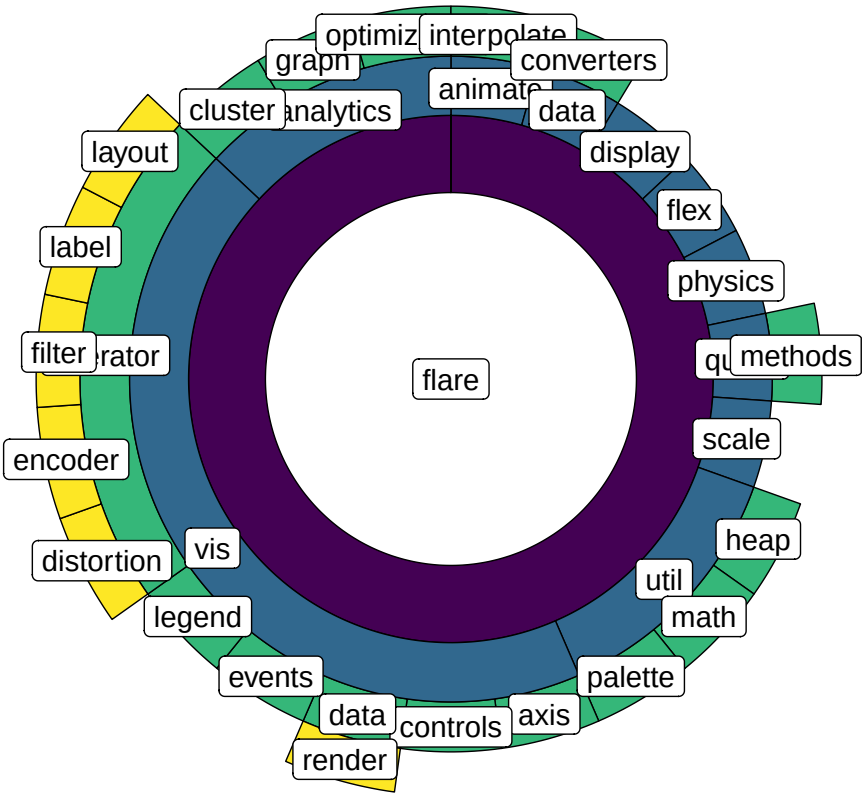


Como se ve en el resultado del gráfico para que se puedan poner etiquetas y entenderse se necesitara quitar algún nivel de profundidad. Y al final recontriur el gráfico de nuevo. con la función de igraph.

```
library(tidyverse)
ramas <- flare$edges %>%
  filter(to %in% from) %>%
  droplevels()
vertices <- flare$vertices %>%
  filter(name %in% c(ramas$from, ramas$to)) %>%
  droplevels()
vertices$size <- runif(nrow(vertices))
mygraph <- graph_from_data_frame( ramas, vertices=vertices)
```

Para el nuevo gráfico solo es distinto al anterior en 2 lineas. (las etiquetas y la paleta de colores)

```
  geom_node_label( aes(label=shortName,)) +
  scale_fill_viridis() +
```



3.8.2. Paquete plotly

Gracias a este paquete se puede realizar un gráfico parecido al del paquete anterior, pero con una interactividad muy bien hecha ya que cuando se clicka en un nodo se recrea el gráfico como si el nodo seleccionado fuera la raiz y asi con los demás nodos, para volver se clicka en el nodo raiz y se vuelve al gráfico donde el nodo antes raiz pasa a hijo y su padre a raiz, asi hasta la raiz real del gráfico.
Para los datos se escogen los sets de datos de un repositorio de github sobre los sabores de café con esta estructura jerarquica.

```
github <- "https://raw.githubusercontent.com/plotly/datasets/"
archivo <- "master/coffee-flavors.csv"
arcom <- "718417069ead87650b90472464c7565dc8c2cb1c/sunburst-coffee-flavors-complete.csv"
url1 <- paste0(github, archivo)
url2 <- paste0(github, arcom)
d1 <- read.csv(url1)
d2 <- read.csv(url2)
head(d1, n=4L)

##                    ids  labels parents
## 1    Enzymatic-Flowery Flowery
## 2     Enzymatic-Fruity  Fruity
## 3      Enzymatic-Herby   Herby
## 4 Sugar Browning-Nutty   Nutty

head(d2, n=4L)

##                     ids         labels parents
## 1               Aromas         Aromas
## 2               Tastes         Tastes
```

```
## 3      Aromas-Enzymatic      Enzymatic  Aromas
## 4 Aromas-Sugar Browning Sugar Browning  Aromas

library(plotly)
```

Para poder representarlos donde el d2 expresa el gráfico completo, pero al no poder verse los nodos más profundos d1 ayuda a representar dichos nodos. Mediante su función principal y el tipo sunburst se a naden los trazos de los dos gráficos.

Como se puede observar para a nadir un trazo de este tipo se necesitan establecer las ids y los padres junto a la máxima profundidad para que se vea mejor.

```
p <- plot_ly() %>%
  add_trace(
    ids = d1$ids,
    labels = d1$labels,
    parents = d1$parents,
    type = 'sunburst',
    maxdepth = 2,
    domain = list(column = 0)
  ) %>%
```

Para el juntar los dos gráficos terminamos el completo con profundidad 3 para que los nodos hojas de este son los nodos raiz del primer gráfico.

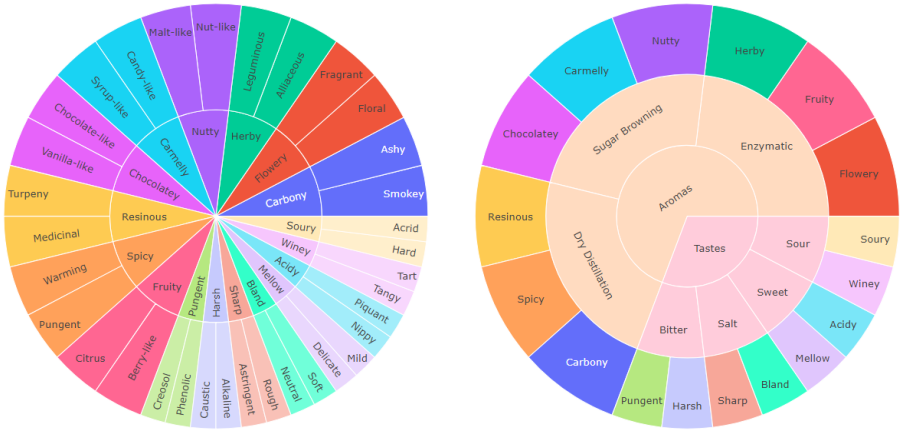
```
  add_trace(
    ids = d2$ids,
    labels = d2$labels,
    parents = d2$parents,
    type = 'sunburst',
    maxdepth = 3,
    domain = list(column = 1)
  ) %>%
```

Para que se visualizen como conectados mediante los colores con `extendsunburstcolors` y situar uno al lado del otro con `grid`.

```
  layout(
    grid = list(columns =2, rows = 1),
    margin = list(l = 0, r = 0, b = 0, t = 0),
    sunburstcolorway = c(
      "#636efa","#EF553B","#00cc96","#ab63fa","#19d3f3",
      "#e763fa", "#FECB52","#FFA15A","#FF6692","#B6E880"
    ),
    extendsunburstcolors = TRUE)
```

Y para que se pueda usar su interactividad la libreria `htmlwidgets` para transformar el gráfico en html.

```
library(htmlwidgets)
saveWidget(p, file = "rayos_interactive.html")
```



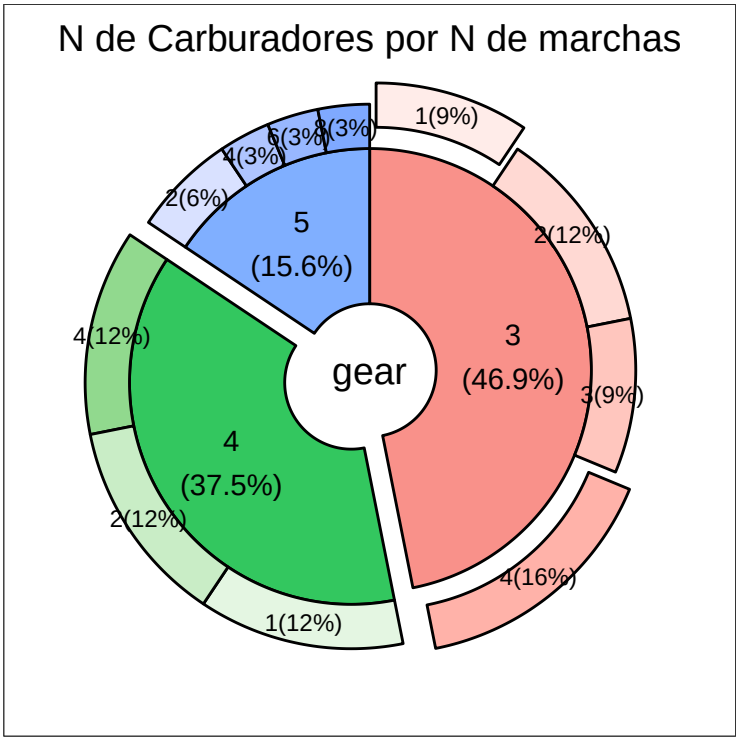
3.8.3. Paquete webr

Este paquete consiste en una serie de funciones que implementan las funciones del libro "Web-based Analysis without R in Your Computer". Dentro del paquete se encuentra la función que se va a usar `pieDonut` que consiste en un digrama de sectores unido a un diagrama de donut. Para poder implementarlo los datos se suelen poner como un data frame con los padres y los hijos.

```
library(dplyr)
df=mtcars %>% group_by(gear,carb) %>% summarize(n=n())
```

Usando pieDonut se puede elegir que datos seran los padres (pies) y cuales los hijos(donut).Para resultar en:

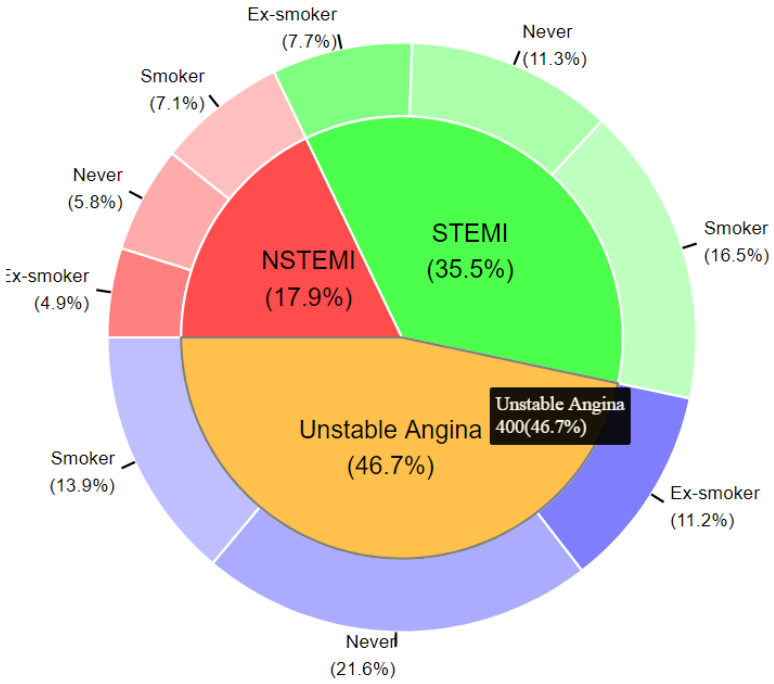
```
library(webr)
library(ggplot2)
PieDonut(df,aes(pies=gear,donuts=carb,count=n),ratioByGroup=FALSE, color = 'black',
         explode = 2, selected = c(1,4), explodeDonut = TRUE, ,labelposition=0,
         title="N de Carburadores por N de marchas")
```



3.8.4. Paquete ggiraphExtra

Otra forma de crear estos pieDonuts es con este paquete, pero en este se puede crearlos interactivos. Se usan datos de las bases de datos moonbook.

```
library(ggiraphExtra)
library(ggplot2)
library(moonBook)
ggPieDonut(acs,aes(pies=Dx,donuts=smoking), interactive = TRUE)
```



3.8.5. Comparaciones

3.9. Diagrama Radar

Para realizar estos digramas se tienen varios paquetes, los cuales siguen la misma estructura de datos de varias variables con un valor numérico.

3.9.1. Paquete fmsb

El nombre de este paquete viene de Functions for Medical Statistics Book with some Demographic Data. Las funciones que tiene el paquete son implementaciones del libro Practices of Medical and Health Data Analysis using R. Para realizar el diagrama primero se necesitan los datos adecuados por lo que se va a optar por representar uno de los gráficos del libro de Manuel Lima [1, pág 151]

```
values <- c(5.3, 5.3, 5.2, 5.6, 6.4, 7.9, 7.5, 7.0, 6.5, 5.6, 5.6, 5.7)
names <- c("Ene", "Feb", "Mar", "Abr", "May", "Jun", "Jul", "Ago", "Sep",
           "Oct", "Nov", "Dec")
data <- as.data.frame(matrix(values, ncol = 12))
colnames(data) <- names
```

Con este paquete se usa la función `radarchart` para dibujar el diagrama radar, pero esta funci+on necesita unos valores máximos y unos mínimos, como el rango del gráfico es de 0 a 10 esos serán los m+aximos y mínimos.

```
data <- rbind(rep(10,12), rep(0,12), data)
```

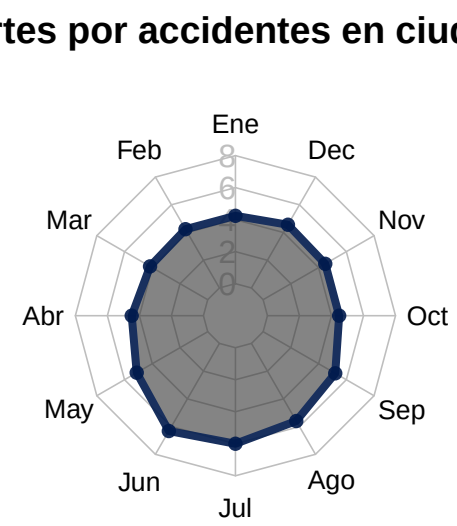
Y ahora se procede a usar la función para dibujar el gráfico, pasando por parámetros los datos, el modo que ense na valores de la cuadrícula.

```
library(fmsb)
radarchart( data  , axistype=1 ,
```

Luego se eligen los colores de los puntos, area, la cuadrícula y las etiquetas.

```
            pcol=rgb(0,0.1,0.3,0.9) , pfcol=rgb(0.2,0.2,0.2,0.6) , plwd=4 ,
            cglcol="grey", cglty=1, axislabcol="grey", caxislabels=seq(0,10,2)
            , cglwd=0.8, vlcex=0.8, title = "Muertes por accidentes en ciudades" )
```

Resultando en el gráfico.



3.9.2. Paquete plotly

Se vuelve a usar este paquete por lo que se sabe que usa la función `plot-ly` para dibujar todo solo le hace falta los datos y el tipo de gráfico que se desea generar.

Primero se escoge el tipo de gráfico.

```
library(plotly)
p <- plot_ly(type = "scatterpolar", fill = "toself") %>%
```

Y con `add-trace` se añade el trazo con los valores usados antes y sus etiquetas.

```
  add_trace( r = values,
             theta = names,
             name = "Ciudades") %>%
```

Con este paquete se puede visualizar más de un diagrama radar para compararlos en el mismo dibujo, lo único necesario es usar más `add-trace`. Para la comparación se han usado los valores de la parte rural del gráfico que se está implementando.

```
values_rural <- c(5.4, 5.3, 5.7, 5.6, 5.8, 6.6, 7.6, 7.8, 6.3, 6.1, 6.2, 5.4)
```

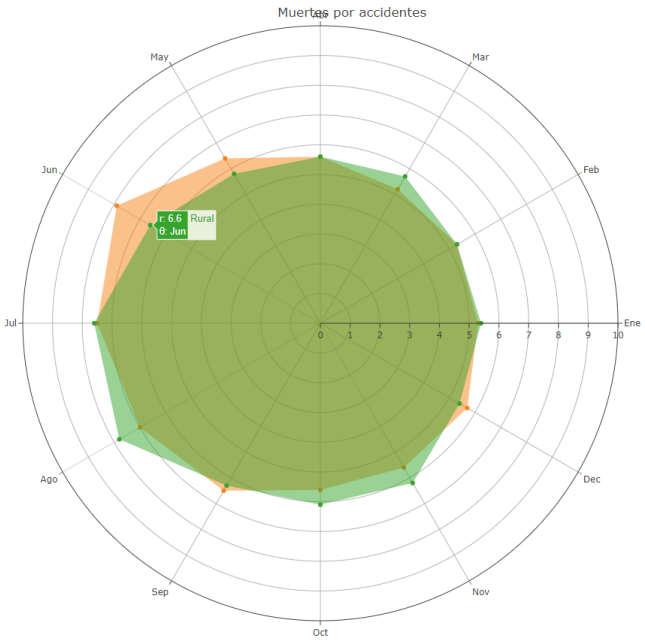
```
  add_trace(r = values_rural,
            theta = names,
            name = "Rural") %>%
```

Y para terminar se le añade el diseño apropiado.

```
  layout( title= "Muertes por accidentes",
    polar = list(
    radialaxis = list(
      visible = T,
      range = c(0, 10))))
```

Debido a que es un diagrama interactivo se usa `htmlwidgets` para usar dicha interactividad, por lo que aqui solo se muestra una foto no el gráfico interactivo.

```
library(htmlwidgets)
saveWidget(p, file="radar_interactive.html")
```

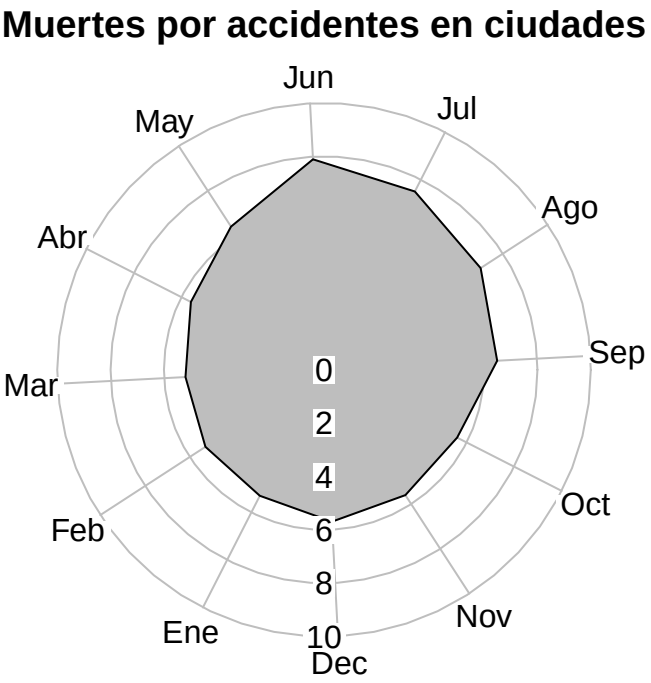


3.9.3. Paquete plotrix

Con este paquete tambien es muy fácil crear estos diagramas radar gracias a la función `radial.plot`, aparte de poder realizar el diagrama al igual que el anterior se pueden comparar en un mismo dibujo dos radares y esta vez con el poder de que el gráfico empiece como el gráfico que se queria y el sentido deseado.
Primero se realiza el mas básico de los radares con los valores anteriores.

Para ello se pasan los datos, se establecen los parámetros, pero lo más importantes es que se establece el tipo del grafo a `rp.type="p"` lo que indica que se va a dibujar un polígono.

```
library(plotrix)
radial.plot(values,labels=names, start = 2.7*pi/2, clockwise = TRUE,rp.type="p",
            main="Muertes por accidentes en ciudades",radial.lim=c(0,10),
            poly.col="grey",show.grid.labels=1)
```



Ahora lo único que se necesita es una matriz con los datos que se quieren que dibujen.

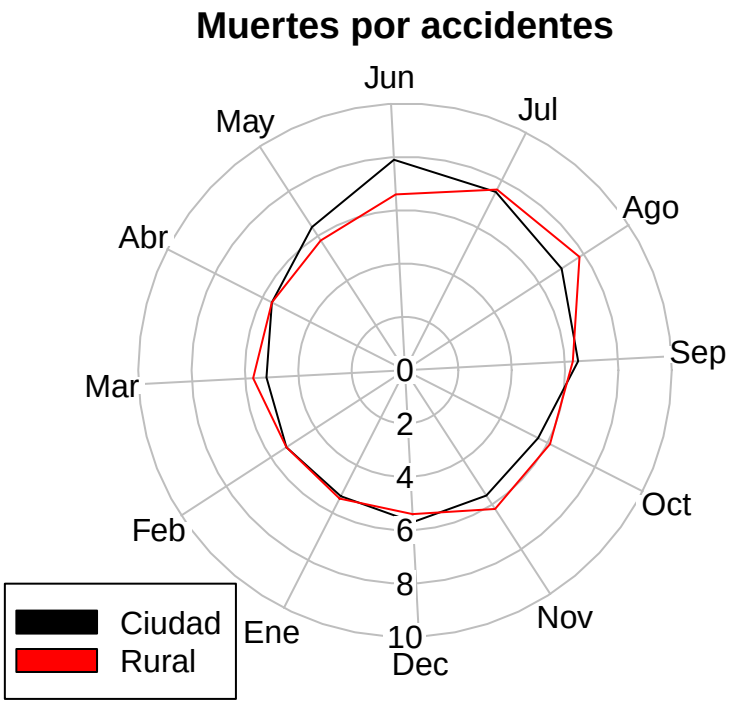
```
data_plotrix <- matrix(rbind(values, values_rural), nrow = 2, ncol = 12)
```

Se vuelve a realizar la misma función solo que sin rellenar el polígono para mejor estética.

```
radial.plot(data_plotrix,labels=names, start = 2.7*pi/2, clockwise = TRUE,rp.type="p",
            main="Muertes por accidentes",radial.lim=c(0,10),show.grid.labels=1)
```

Y para mejor visualización se usa el método `legendg` del propio paquete para la leyenda.

```
legendg(-15,-8,c("Ciudad","Rural"),fill=list(1,2))
```



3.9.4. Paquete ggiraphExtra

Se vuelve a ver este paquete para dar interactividad a los gráficos con ggplot2 y ggiraph, pero como este diagrama no se puede representar con el paquete ggplot2. Con este paquete se usa la función `ggRadar` la cual se ayuda de `ggplot2` para mejorar su aspecto.
Con las dos librerias cargadas.

```
library(ggiraphExtra)
library(ggplot2)
```

Ahora con ggradar y los dos grupos de datos podemos recrear el diagrama y comparar la ciudad con lo rural.

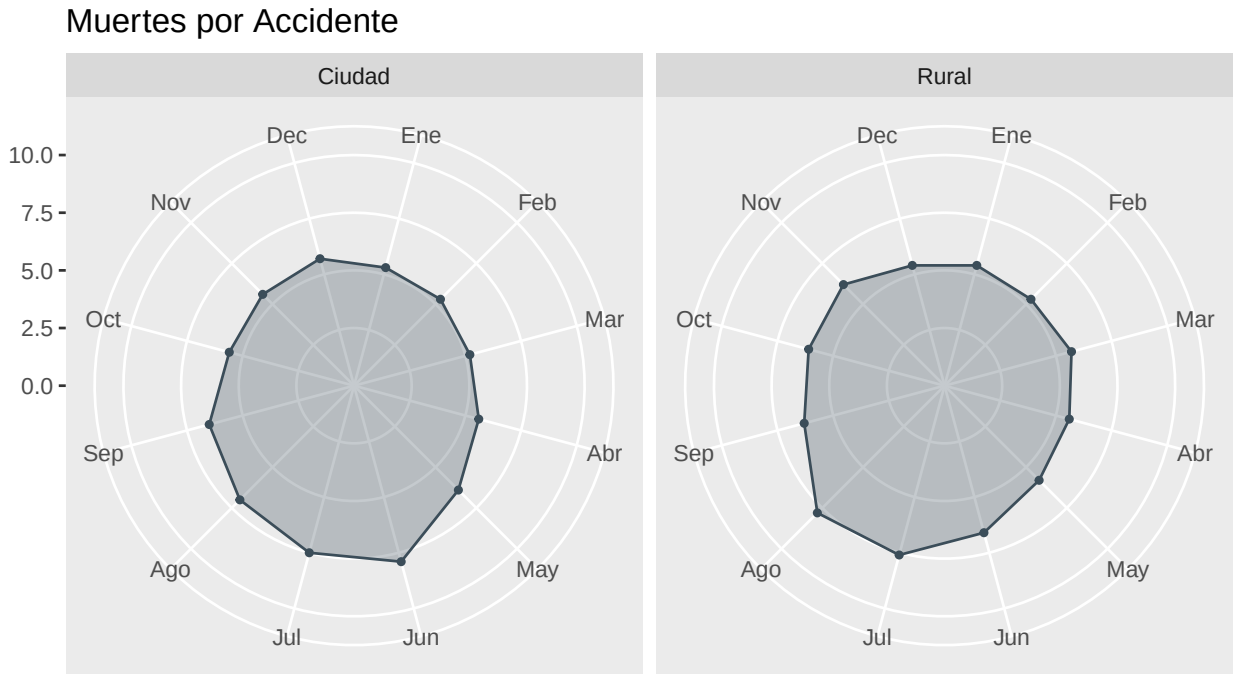
```
p <- ggRadar(data_gg, aes(group = Muerte),
             rescale = FALSE, legend.position = "none",
             size = 1, interactive = FALSE, use.label = TRUE) +
```

Con este método podemos comparar en dos gráfcos separados, pero si no estuviera serián como los anteriores los dos conjuntos de datos en el mismo gráfico.

```
  facet_wrap(~Muerte) +
```

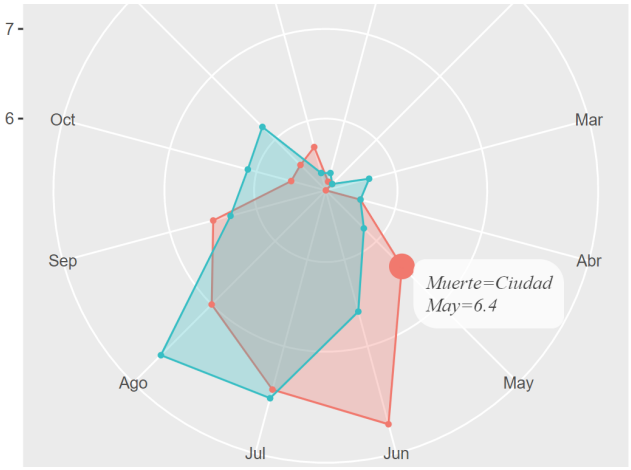
Y por último el rango, el color y título.

```
  ylim(0,10) +
  scale_fill_manual(values = rep(micolor, nrow(data_gg))) +
  scale_color_manual(values = rep(micolor, nrow(data_gg))) +
  ggtitle("Muertes por Accidente")
print(p)
```



Para dar la parte de interactividad hay que quitar la parte de ggplot2 y poder poner `interactive=TRUE`.

```
ggRadar(plot.df, aes(group = Muerte),
        rescale = FALSE, legend.position = "none",
        size = 1, interactive = TRUE, use.label = TRUE)
```



3.9.5. Comparaciones

3.10. Diagrama de Columnas Circulares Separadas

Explicación.

3.10.1. Paquete ggplot2

Ejemlpo.

3.10.2. Paquete circlize

Ejemlpo.

3.10.3. Comparaciones

3.11. Diagrama Flow

Explicación.

3.11.1. Paquete ggplot2

Ejemlpo.

3.11.2. Paquete circlize

Ejemlpo.

3.11.3. Paquete ggiraphExtra

Ejemlpo.

3.11.4. Comparaciones

3.12. Diagrama de Burbujas Circular

Para poder representar este tipo de diagramas se requiere de una serie de datos, los cuales son la posición de los círculos (posición x e y) y el radio de estos. Para poder implementar este diagrama se ha conseguido encontrar dos paquetes que lo permiten, los dos usan una fórmula similar para crear los diagramas usando una función similar para crear el layout que dice donde están los circulos para luego juntarlo con los datos de los radios.

3.12.1. Paquete ggraph-ggforce

Lo primero de todo será llamar a las librerias para luego poder trabajar con ellas.

```
library(ggraph)
library(ggforce)
```

Después se crean las areas de las burbujas de forma aleatoria.

```
areas <- sample(10, 100, TRUE)
```

Ahora gracias a la función `pack-circles` del paquete ggraph se obtienen las posiciones de las burbujas dependiendo del vector de áreas

```
posicion <- pack_circles(areas)
```

Por último a la hora de confeccionar los datos se crea la tabla con x, y y el radio de las burbujas a partir de sus áreas.

```
datos <- data.frame(x = posicion[, 1], y = posicion[, 2], r = sqrt(areas / pi))
```

A la hora de dibujar el gáfico se hace uso de ggplot y la función de ggforce `geom-circle`
Para ello se necesita establecer el espacio para dibujar las burbujas (ggplot) y dibujar las burbujas usando las posiciones y colorear en función del radio.

```
ggplot() +
  geom_circle(aes(x0 = x, y0 = y, r = r, fill = r), data = datos) +
```

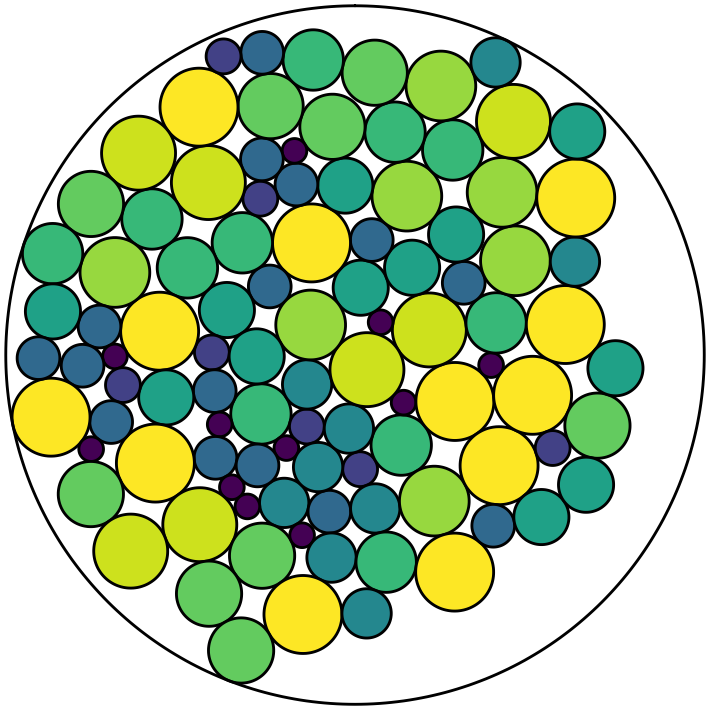
Luego para descatar el gráfico se crea un círculo que encapsule a las burbujas.

```
  geom_circle(aes(x0 = 0, y0 = 0, r = attr(posicion, 'enclosing_radius'))) +
```

Y para la estética se opta por la escala de colores viridis y dejar el gráfico limpio de texto.

```
  scale_fill_viridis() +
  theme_void() +
  theme(legend.position = 'none')
```

Resultando en:



3.12.2. Paquete packcircles

Este paquete ofrece una forma facil de empaquetar círculos dentro de otro para resultar en un diagrama de burbujas circular. Para poder crear las burbujas se necesitan una areas de lo cual se puede utilizar una función del propio paquete para crear el ejemplo. Se establecen las variables necesarias:

```
ncirculos <- 200
limites <- c(-40 , 40)
maxarea <- 40
```

Luego se crean las áreas con una distribución beta con la función `rbeta`.

```
areas <- rbeta(ncirculos, 1, 5) * maxarea
```

Después se necesita crear un layout donde las burbujas no se solapen para que cada burbuja ocupe espacio propio, por lo que ahora se empieza a utilizar la libreria packcircles.

```
library(packcircles)
res <- circleRepelLayout(areas, xlim = limites, ylim = limites)
```

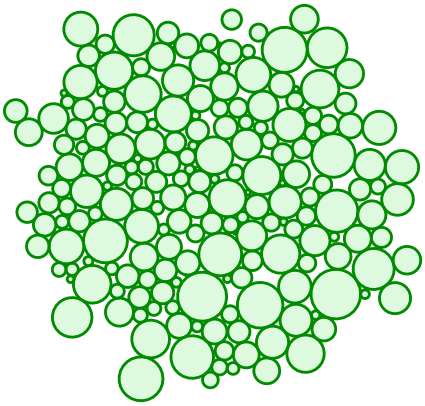
Por último se crea el layout donde se situa a cada burbuja en su lugar, indicando con la variable id (que resulta del siguiente código) en la fila de la matriz que situa las burbujas.

```
datos.gg <- circleLayoutVertices(res$layout, sizetype = "radius")
```

Y gracias a la libreria `ggplot2` se va a proceder a dibujar el resultado con el uso de `geom-poligon`.

```
library(ggplot2)

ggplot(data = datos.gg, aes(x, y, group = id)) +
  geom_polygon(colour="green4", fill="lightgreen", alpha=0.3) +
  coord_equal(xlim=limites, ylim=limites) +
  theme_void()
```



3.12.3. Comparaciones

3.13. Diagrama de Mapa de Árbol Circular

En este apartado se encuentra otra manera de representar diagramas de árbol con formato circular. En el anterior se utilizaba una cuadrícula para representar los nodos y esta vez se usan círculos.
Para crear este diagrama primero hay que conseguir unos datos con una estructura jerárquica la cual indique como esta constituido el árbol, su raiz, las ramas, sus hojas y para este grafo tambien es util un valor que estime el tama no de de los circulos(como la población de un pais).
Para implementar este gráfico se pueden usar 2 paquetes los dos iguales de faciles de usar, ya que lo dificil es crear esas bases de datos.

3.13.1. Paquete ggraph

Como hemos visto en el diagrama de Grid Árbol se va a recurrir a la misma función y misma base de datos solo que ahora cambia el layout de `partition` a `circlepack`, pero por lo demás consiste en lo mismo.
Escoger los datos del propio paquete (`flare`) y gracias al paquete `igraph` transformarlo en unos datos con los que poder trabajar.

```
library(ggraph)
ramas <- flare$edges
vertices <- flare$vertices
library(igraph)
migrafo <- graph_from_data_frame(ramas, vertices = vertices)
```

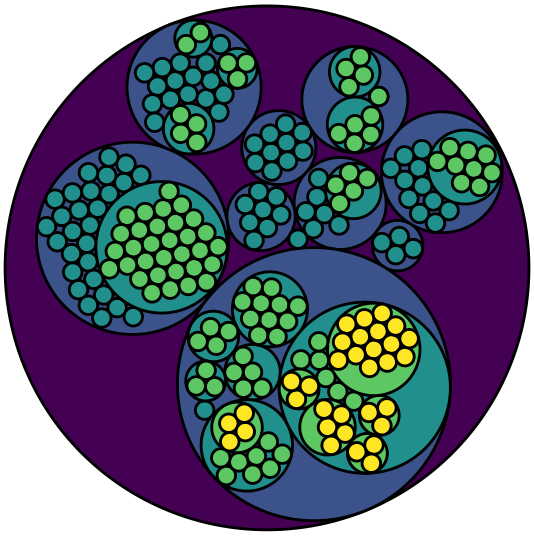
Para dibujar el mapa de árbol circular se necesita llamar a la función principal de paquete y para adjudicar la forma se establece que sean nodos circulares y que el color dependa de la profundidad (nivel de los nodos dentro del árbol).

```
ggraph(migrafo, layout = 'circlepack') +
  geom_node_circle(aes(fill = depth)) +
```

Y para obtener una estetica vistosa se usa la escala de viridis, el fondo blanco y sin leyenda.

```
  scale_fill_viridis() +
  theme_void() +
  theme(legend.position = "none")
```

Resultando en el gráfico deseado.



Aunque con estos datos no se pueden poner unas etiquetas claras para saber que es cada uno de los nodos asi que como en el diagrama grid árbol se procede a eliminar un nivel para mejor visualización.

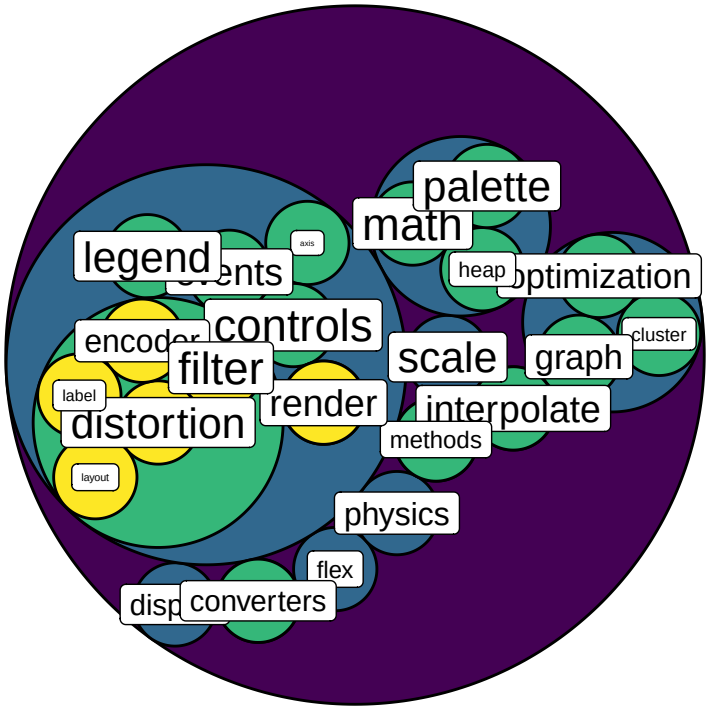
```
library(tidyverse)
ramas <- flare$edges %>%
  filter(to %in% from) %>%
  droplevels()
vertices <- flare$vertices %>%
  filter(name %in% c(ramas$from, ramas$to)) %>%
  droplevels()
vertices$size <- runif(nrow(vertices))
```

Y se necesita volver a construir de nuevo los datos para el gráfico.

```
migrafo <- graph_from_data_frame( ramas, vertices=vertices )
```

Y ahora gracias a este acortamiento de los datos se pueden poner la etiquetas con el a nadido de la orden `geom-node-label`, resultando en la siguiente orden compleja y el gráfico resultante.

```
ggraph(migrafo, layout = 'circlepack') +
  geom_node_circle(aes(fill = depth)) +
  geom_node_label(aes(label = shortName, filter = leaf, size = size)) +
  scale_fill_viridis() +
  theme_void() +
  theme(legend.position = "none")
```



3.13.2. Paquete ciclrpackeR

Este paquete también necesita una estrutura de datos parecida para poder utilizarse, pero aparte de esta complejidad este gráfico nos genera mapas de arboles circulares interactivos asi que al igual que con los diagramas de sectores se mostrará el gráfico, pero también una forma de guardarlo en html para usar la interactividad. Primero necesitamos descargar el paquete del creador.

```
devtools::install_github("jeromefroe/circlepackeR")
library(circlepackeR)
```

Y para los datos por ejemplo se encuentra como se ha dicho en la introducción de este diagrama una base de datos con la poblacion de cada pais.

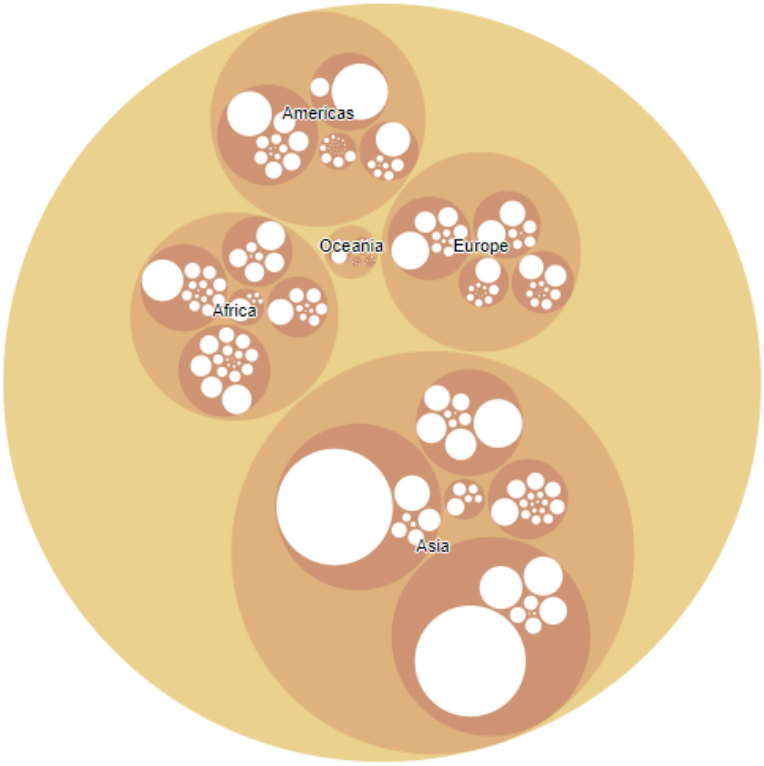
```
url<-"https://raw.githubusercontent.com/"
github <- "holtzy/data_to_viz/master/Example_dataset/"
archivo <- "11_SevCatOneNumNestedOneObsPerGroup.csv"
library(tidyverse)
data <- paste0(url, github, archivo) %>%
  read.table( header=T, sep=";")
```

Luego preparar los datos quitando lineas problemáticas y transformandolos a una estructura en forma jerarquica.

```
data[ which(data$value==-1),"value"] <- 1
colnames(data) <- c("Continente", "Region", "Pais", "Pob")
data <- data %>% filter(Continente!="") %>% droplevels()
data$pathString <- paste("Mundo", data$Continente, data$Region, data$Pais, sep = "/")
library(data.tree)
poblacion <- as.Node(data)
```

Y gracias a su función principal crear el gráfico .

```
p <- circlepackeR(poblacion, size = "Pob", color_min = "hsl(56,80%,80%)"
                  , color_max = "hsl(341,30%,40%)")
```

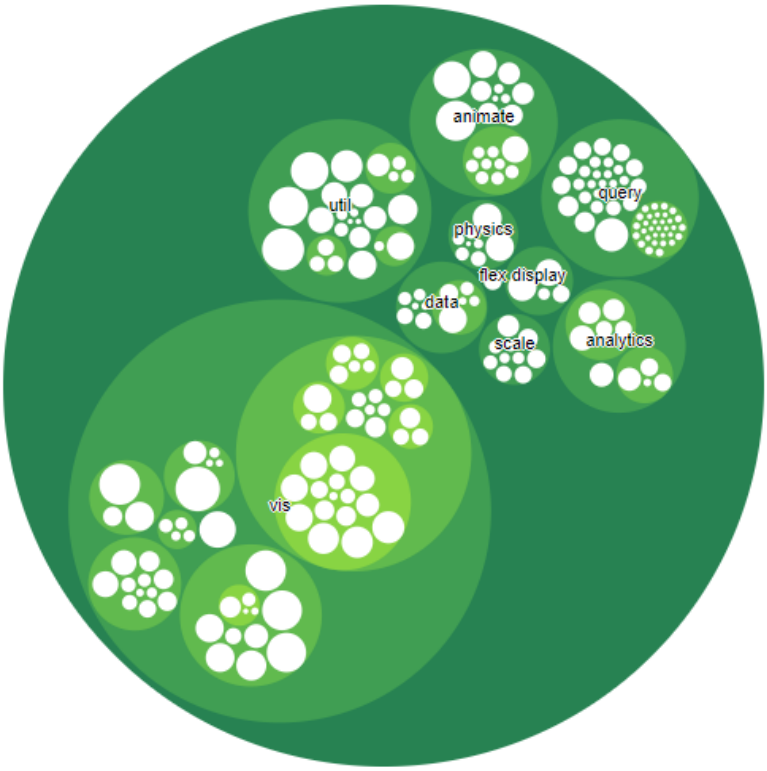


Y para guardalo en html para disfrutar de la interactividad sde necesita de los siguente.

```
library(htmlwidgets)
saveWidget(p, file="tree_interactive.html")
```

Y para poder compararlo con el gráfico del anterior digrama también se escogerá la base de datos flare en formato json de un repositorio de github. Y luego guardandolo en html.

```
url<-"https://gist.githubusercontent.com/"
github <- "mbostock/1093025/raw/05621a578a66fba4d2cbf5a77e2d1bb3a27ac3d4/"
archivo <- "flare.json"
flare <- paste0(url,github,archivo)
p <- circlepackeR(flare, color_min = "hsl(163%, 81%, 22%)",
                  color_max = "hsl(64%, 100%, 50%)")
saveWidget(p, file="tree_flare_interactive.html")
```



3.13.3. Comparaciones

3.14. Diagrama Mapa de Árbol Voronoi

Para realizar este tipo de mapa de árbol solo se ha conseguido encontrar un paquete que realize este diagrama. De forma normal el diagrama voronoi tiene varios paquetes, pero con la forma de arbol parece que es más especifico.

3.14.1. Paquete voronoiTreemap

Este paquete está destinado para el diagrama ya que solo tiene una función de dibujar y unos datos de ejemplo como lo fuerte que es la economia según continente y pais y otra base de datos simulando el precio de los productos en Canada. Priemro se obtienen los datos necesarios. Se cogen del propio paquete, consisten en una variable con los paises, para dividirse el grafo según su peso en la economia y los continentes a los que pertenecen para agruparlos.

```
library(voronoiTreemap)
data(ExampleGDP)
head(ExampleGDP)
```

```
##      h1   h2          h3   color weight codes
## 1 Total Asia       China #f58321  14.84    CN
## 2 Total Asia       Japan #f58321   5.91    JP
## 3 Total Asia       India #f58321   2.83    IN
## 4 Total Asia South Korea #f58321   1.86    KR
## 5 Total Asia      Russia #f58321   1.80    RU
## 6 Total Asia   Indonesia #f58321   1.16    ID
```

Debido a que la función principal, `vt-d3`, necesita datos en json se usa otra de las funciones del paquete para crear el json.

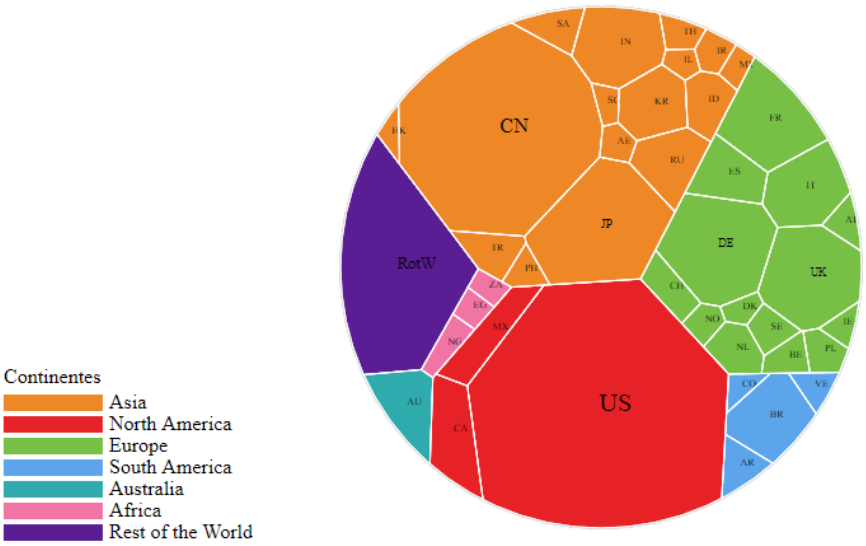
```
gdp_json <- vt_export_json(vt_input_from_df(ExampleGDP))
```

Y por último se emplea la función para crear el diagrama

```
p <- vt_d3(gdp_json, legend = TRUE, legend_title = "Continentes", seed = 1)
```

Debido a que cuenta con interactividad solo se muestra una foto del diagrama obtenido por este método.

```
library(htmlwidgets)
saveWidget(p,file="pol_interactive.html")
```



Para poder realizar los datos que se necesiten exinten dos posibilidades, una seguir la estructura de ExampleGDP, o crear la estructura de nodos gracias a `vt-create-node` y `vt-add-nodes` como en el siguiente ejemplo recreando el gráfico de [1, pág 194].

Se ve como se pueden contruir los nodos mediante las dos funciones.

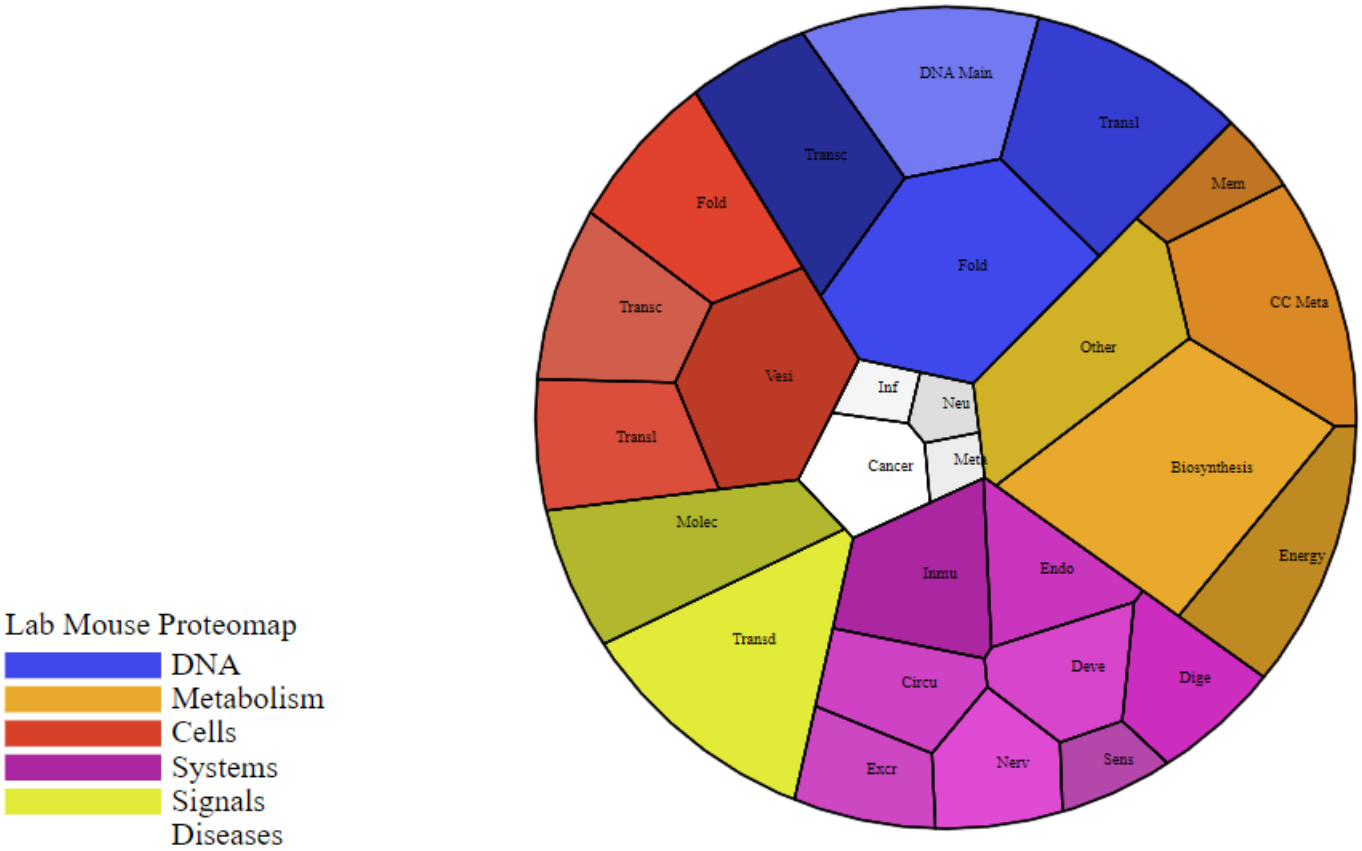
```
n <- vt_create_node("Proteinas")
n <- vt_add_nodes(n, refnode="Proteinas",node_names=c("DNA", "Metabolism", "Cells",
                                                      "Systems", "Signals", "Diseases"),
                  colors = c("#3f48eb","#e8a92c","#d64029","#ab26a0","#e2eb38","#ffffff"))
n <- vt_add_nodes(n, refnode="DNA",node_names=c("DNA Maintenance","Folding", "Translation"
                  , "Transcription"), weights=c(0.7, 1, 0.8, 0.7),
                  codes=c("DNA Main","Fold", "Transl", "Transc")
                  ,colors = c("#7379f0","#3f48eb","#363ed1","#272d96"))
```

Creando todos los nodos nos resulta la siguiente estructura.

```
                                levelName weight         code   color
1  Proteinas                                   NA
2    |--DNA                                     NA              #3f48eb
3    |    |--DNA Maintenance                 0.70     DNA Main #7379f0
4    |    |--Folding                         1.00         Fold #3f48eb
5    |    |--Translation                     0.80       Transl #363ed1
6    |    °--Transcription                   0.70       Transc #272d96
7    |--Metabolism                              NA              #e8a92c
8    |    |--Biosynthesis                    1.20 Biosynthesis #e8a92c
9    |    |--Central Carbon Metabolism       0.70      CC Meta #db8925
10   |    |--Energy Metabolism               0.50       Energy #bf8a21
11   |    |--Membrane transport              0.20          Mem #bf7521
12   |    °--Other Enzymes                   0.70        Other #d1b226
13   |--Cells                                   NA              #d64029
14   |    |--Vesicular Transport             0.70         Vesi #bd3a26
15   |    |--Cytoskeleton                    0.60         Fold #e0422d
16   |    |--Cell Grow and death             0.50       Transl #db4e3b
17   |    °--Cell Comunications              0.50       Transc #d15d4d
18   |--Systems                                 NA              #ab26a0
19   |    |--Inmune System                   0.50         Inmu #ab26a0
20   |    |--Digestive System                0.35         Dige #cc2dbf
21   |    |--Endocrine System                0.35         Endo #c936bd
22   |    |--Circulatory System              0.35        Circu #cf42c3
23   |    |--Developement                    0.35         Deve #d645ca
24   |    |--Nervous System                  0.35         Nerv #de4bd2
25   |    |--Excretory System                0.30         Excr #cc47c1
26   |    °--Sensory System                  0.15         Sens #b347aa
27   |--Signals                                 NA              #e2eb38
28   |    |--Signal Transduction             0.90       Transd #e2eb38
29   |    °--Signals Molecules               0.70        Molec #b1b82e
30   °--Diseases                                NA              #ffffff
31        |--Cancers                         0.30       Cancer #ffffff
32        |--Infectios Diseases              0.10          Inf #f5f5f5
33        |--Metabolic Diseases              0.08         Meta #ededed
34        °--Neurodegerative Diseases        0.10          Neu #dedede
```

Y ofreciendo el siguinte gráfico.

```
data<-vt_export_json(n)
p <- vt_d3(data, label = TRUE,color_border = 'black', seed = 28
      , legend = TRUE, legend_title = "Lab Mouse Proteomap")
saveWidget(p, file = "mouse_interactive.html")
```



3.14.2. Comparaciones

3.15. Diagrama de Mapa de Carreteras

Este diagrama al ser tan especifico solo se ha encontrado un ejemplo con el paquete ggplot2. Por lo que se va a hacer es seguir el ejemplo.
Se va a usar el paquete tigris para obetener las carreteras y el paquete sf para manipularlo.

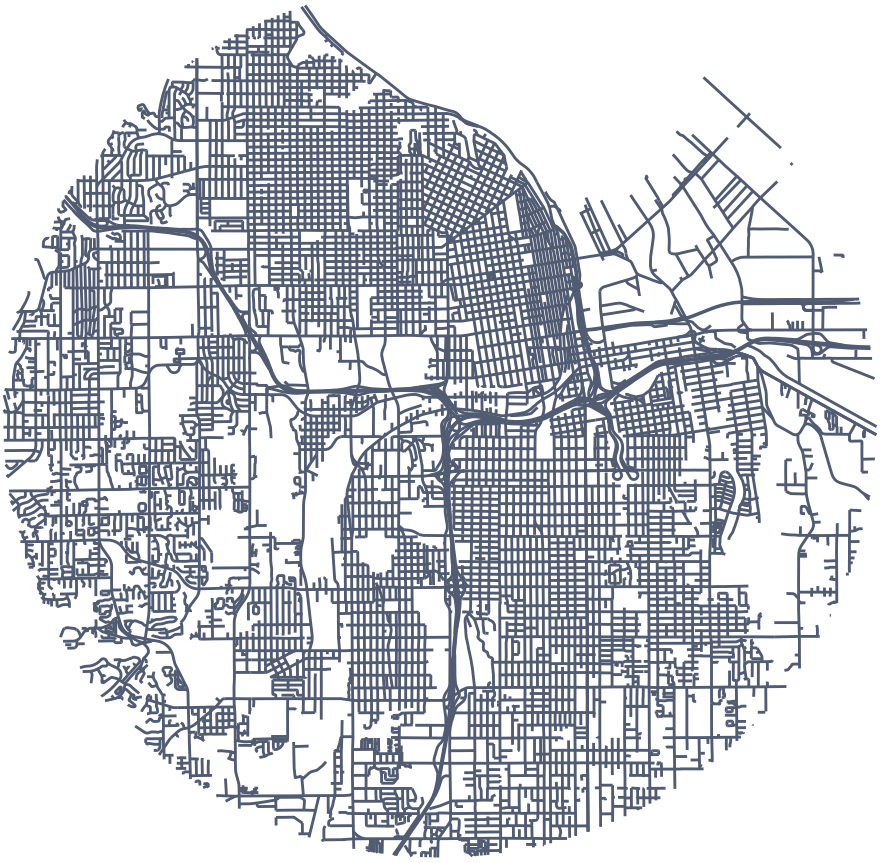
```
library(tigris)
library(sf)
options(tigris_class = "sf")
pierce_roads <- roads(state = "WA", county = "Pierce", year = 2017) %>%
  st_transform(26910)
repos <- "https://opendata.arcgis.com/datasets/"
archivo <- "94d4f87befd84ce5a0c2d3c542c4e219_1.geojson"
url <- paste0(repos, archivo)
tacoma <- st_read(url) %>%
  st_transform(26910)
```

Para guardar las carreteras que están dentro de un circulo se usa `st-intersection` por lo que se guardan esas carreteras para la visualización.

```
tacoma_roads <- st_crop(pierce_roads, st_bbox(tacoma))
cropped_roads <- st_intersection(tacoma_roads,
          st_buffer(st_centroid(st_union(tacoma_roads)), 7000))
```

Y por último gracias a ggplot2 se puede visualizar este conjunto de carreteras formando un círculo.

```
library(ggplot2)
ggplot(cropped_roads) +
  geom_sf(color = "#515b72") +
  theme_void() +
  theme(panel.grid.major = element_line("transparent"))
```



3.16. Dendograma Circular

Explicación.

3.16.1. Paquete ggraph

Ejemlpo.

3.16.2. Paquete circlize

Ejemlpo.

3.16.3. Paquete basicR

Ejemlpo.

3.16.4. Paquete ggplot2

Ejemlpo.

3.16.5. Comparaciones

3.17. Diagrama de Cuerdas

La estructura de datos necesaria para crear este diagrama es muy parecido al digrama de arbol circulas ya que se requiere algún tipo de estructura de origen y destino, Lo que se puede aplicar a matrices de adyacencia por lo que se va a optar por ellas para realizar las estructuras. Para realizar este digrama se han encontrado dos paquetes que ofrecen el digarma de cuerdas, uno de ellos interactivo.

3.17.1. Paquete Circlize

Se vuelve a ver este paquete, pero esta vez en vez de usar las funciones `circos.track, circos.trackPlotRegion` como las principales solo son una ayuda para ayudar a la función `chorDiagram`. Primero se preparan los datos con la estructura de origen y destino. Se ha optado por un set de datos que expone la migración entre paises.

```
repositorio <- "https://raw.githubusercontent.com/"
github <- "holtzy/data_to_viz/master/Example_dataset/"
archivo <-"13_AdjacencyDirectedWeighted.csv"
url <- paste0(repositorio, github, archivo)
data <- read.table(url, header = TRUE)
colnames(data) <- c("Africa", "Este Asia", "Europa", "Ame. Latina",   "Norte Ame.",
                    "Oceania", "Sur Asia", "Sureste Asia", "Union Sovietica",
                    "Oeste Asia")
rownames(data) <- colnames(data)
library(tidyverse)
data_long <- data %>%
  rownames_to_column %>%
  gather(key = 'key', value='value', -rowname)
```

Para iniciar el gráfico como se vio en el anterior uso de este paquetese tiene que crear el circos con unos parametros iniciales para tener una plantilla.

```
library(circlize)
circos.clear()
circos.par(start.degree = 90, gap.degree = 4,
           track.margin = c(-0.1, 0.1), points.overflow.warning = FALSE)
par(mar = rep(0, 4))
```

Para mejor visualización se opta por una paleta de color del paquete viridis.

```
library(viridis)
micolor <- viridis(10, alpha = 1, begin = 0, end = 1, option = "D")
micolor <- micolor[sample(1:10)]
```

Y por último se crea el plot final con la función `chorDiagram` y el texto que le acompa na.

Primero se le pasa la matriz de adyacencia, los colores, que tipo de grafo se elige (en este carro flechas direccionales) y los diferentes valores de la personalización.

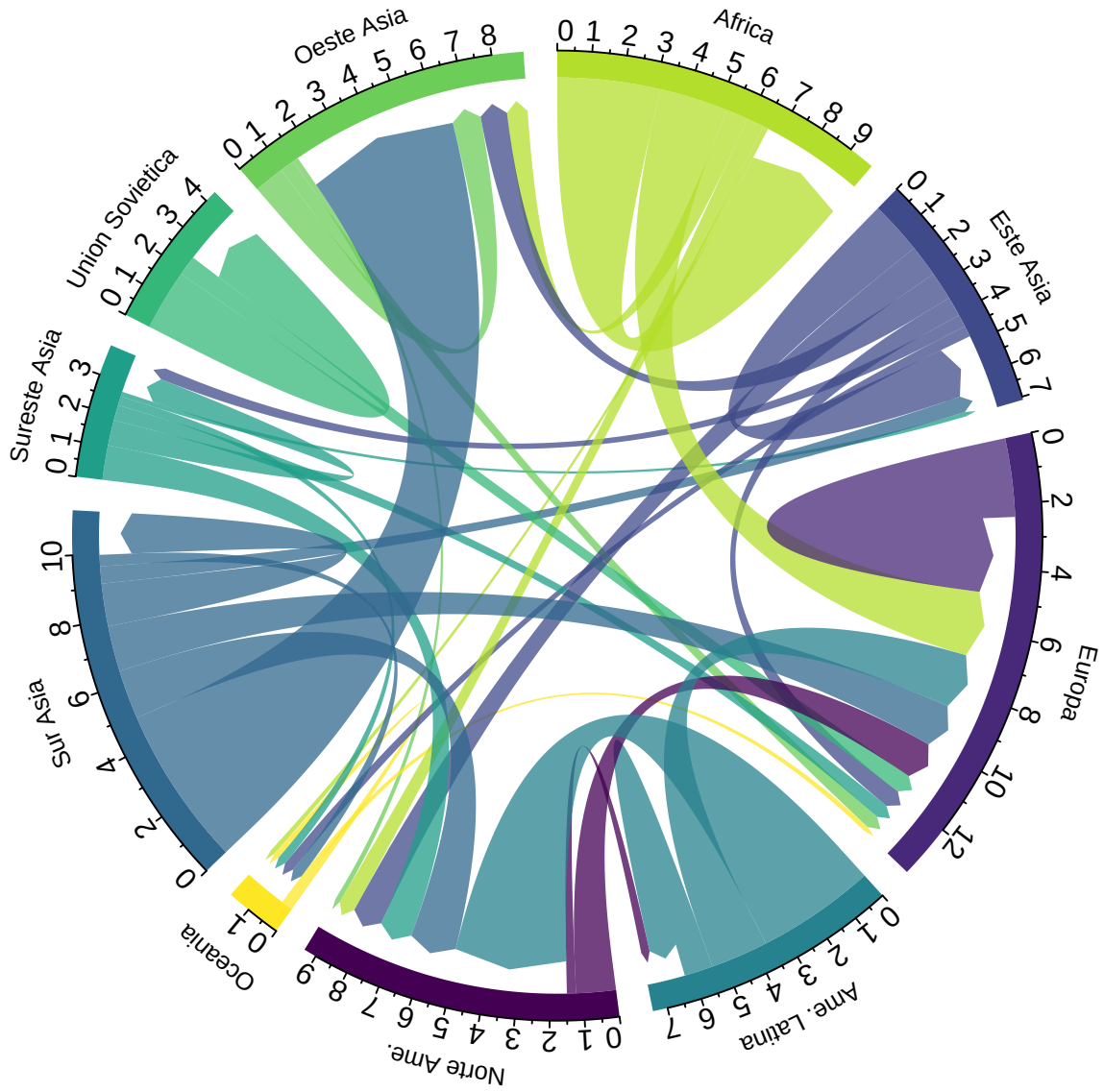
```
chordDiagram(
  x = data_long,
  grid.col = micolor,
  transparency = 0.25,
  directional = 1,
  direction.type = c("arrows", "diffHeight"),
  diffHeight  = -0.04,
  annotationTrack = "grid",
  annotationTrackHeight = c(0.05, 0.1),
```

```
  link.arr.type = "big.arrow",
  link.sort = TRUE,
  link.largest.ontop = TRUE)
```

Luego para personalizar el texto del diagrama se puede usar la función `circos.trackPlotRegion` y dentro de la función `circos.text` y `circos.axis`.

```
circos.trackPlotRegion(
  track.index = 1,
  bg.border = NA,
  panel.fun = function(x, y) {
    xlim = get.cell.meta.data("xlim")
    sector.index = get.cell.meta.data("sector.index")
    circos.text(
      x = mean(xlim),
      y = 3.2,
      labels = sector.index,
      facing = "bending",
      cex = 0.8
    )
    circos.axis(
      h = "top",
      major.at = seq(from = 0, to = xlim[2], by = ifelse(test = xlim[2]>10,
                                                         yes = 2, no = 1)),
      minor.ticks = 1,
      major.tick.percentage = 0.5,
      labels.niceFacing = FALSE)
  }
)
```

Resultando al siguiente gráfico de la migración.



3.17.2. Paquete chorddiag

Este paquete sirve para crear los diagramas de cuerdas usando la libreria de D3 de visualización JavaScript usando el paquete que se usa para guardar los diagramas interactivos `htmlwidgets`. Para instalar este paquete es necesario el siguiente comando debido a que no se encuentra en el repositorio principal de R.

```
devtools::install_github("mattflor/chorddiag")
```

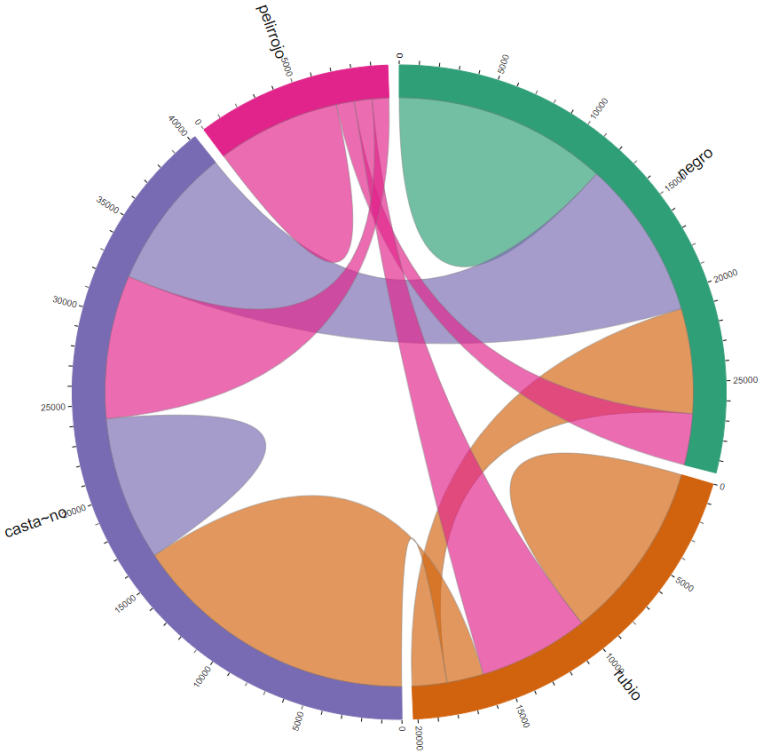
Como con el anterior paquete se necesita la matriz de adyacencia.

```
m <- matrix(c(11975,  5871, 8916, 2868,
              1951, 10048, 2060, 6171,
              8010, 16145, 8090, 8045,
              1013,   990,  940, 6907),
            byrow = TRUE,
```

```
            nrow = 4, ncol = 4)
groupNames <- c("negro", "rubio", "casta~no", "pelirrojo")
row.names(m) <- groupNames
colnames(m) <- groupNames
```

Y usando la función `chorddiag` para realizar el digrama.

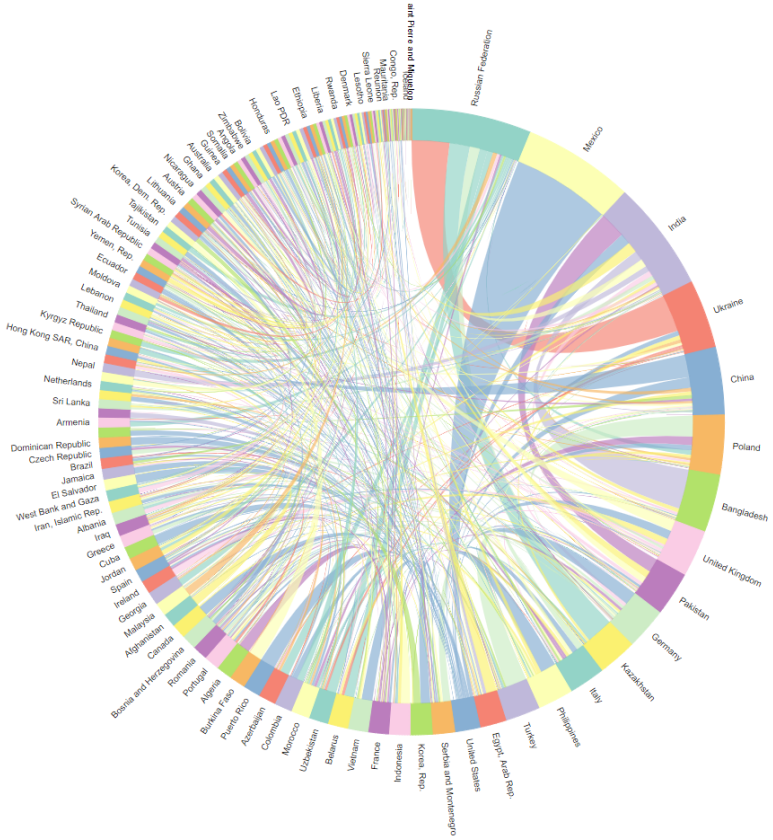
```
library(chorddiag)
p <- chorddiag(m, showGroupnames = T)
library(htmlwidgets)
saveWidget(p, file = "cuerdas_inte.html")
```



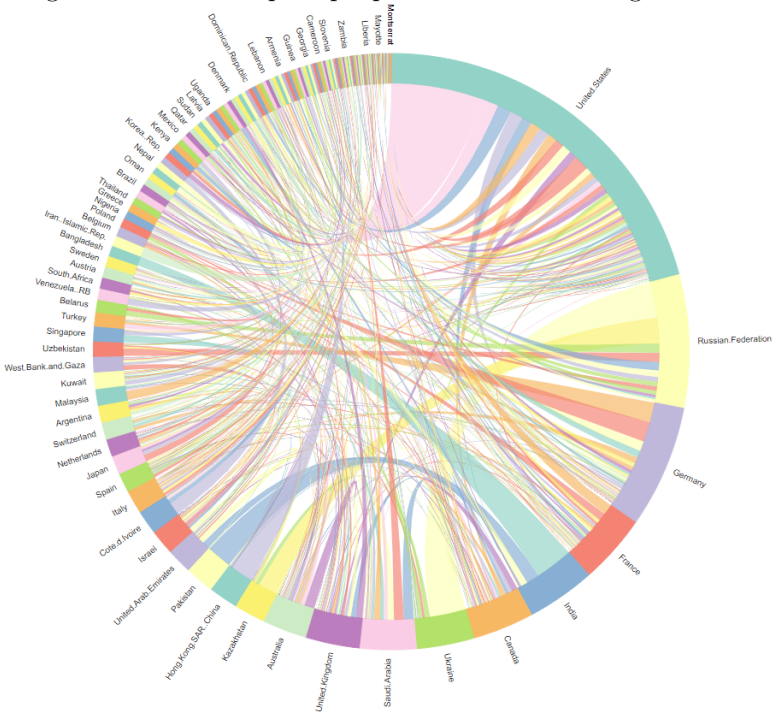
Y para crear un ejemplo más complicado se opta por representar las migraciones entre paises como en el paquete anterior. Preparamos la matriz de adyacencia con los datos del paquete migration.indices.

```
library(migration.indices)
library(RColorBrewer)
data("migration.world")
sort.by.orig <- sort(rowSums(migration.world),
                     decreasing = TRUE, index.return = TRUE)
mig.sorted.by.orig <- migration.world[sort.by.orig$ix, sort.by.orig$ix]
row.names(mig.sorted.by.orig) <- names(sort.by.orig$x)
n <- dim(mig.sorted.by.orig)[1]
groupColors <- rep(brewer.pal(12, "Set3")[c(1:8, 10:12)], length.out = n)
groupNames <- rep("", n)
ix <- c(1:50, seq(52, 100, by = 2),
        seq(105, 150, by = 5),
        160, 180, 226)
groupNames[ix] <- colnames(mig.sorted.by.orig)[ix]
tooltipNames <- colnames(mig.sorted.by.orig)
```

Y el dibujo resultante la función `chorddiag`.



Este gráfico representa de cada pais donde emigra sus habitantes. por lo que ahora simplemente con la traspuesta se puede conseguir saber de cada pais que paises vienen sus inmigrantes.



3.17.3. Comparaciones

3.18. Diagrama de Redes

Explicación.

3.18.1. Paquete qgraph

Ejemlpo.

3.18.2. Paquete igraph

Ejemlpo.

3.18.3. Comparaciones

4. Diagramas no Realizables

- Diagrama de Cuadrícula Desordenado.
- Diagrama de Planos Circulares
- Diagrama de Mapa de Esferas

5. Comparativa

Referencias

[1] M. Lima, *The Book of Circles Visualizing Spheres of Knowledge*. Princeton Architectural Press, NY, 2017.