Social Project

2025-01-10

```
library(igraph)
```

```
##
## Caricamento pacchetto: 'igraph'

## I seguenti oggetti sono mascherati da 'package:stats':
##
##     decompose, spectrum

## Il seguente oggetto è mascherato da 'package:base':
##
##     union
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::%--%()      masks igraph::%--%()
## x dplyr::as_data_frame() masks tibble::as_data_frame(), igraph::as_data_frame()
## x purrr::compose()       masks igraph::compose()
## x tidyr::crossing()      masks igraph::crossing()
## x dplyr::filter()        masks stats::filter()
## x dplyr::lag()           masks stats::lag()
## x purrr::simplify()      masks igraph::simplify()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(ggplot2)
theme_set(theme_minimal())
```

Data

Reading music data

```
ms_data = read.csv("final_cleaned_lyrics_dataset.csv")

adj_mat = read.csv("final_adjacency matrix.csv", row.names = 1)
adj_mat = as.matrix(t(adj_mat))
```

Taking a look of the structure

```
str(ms_data)
```

```
## 'data.frame':    221160 obs. of  5 variables:
##  $ user_id      : chr  "4353e884c1" "4353e884c1" "4353e884c1" "4353e884c1" ...
##  $ disorder     : chr  "anxiety" "anxiety" "anxiety" "anxiety" ...
##  $ artist       : chr  "Echo & the Bunnymen" "Turin Brakes" "Radiohead" "Peace" ...
##  $ title        : chr  "The Killing Moon" "Red Moon" "Sail To The Moon" "Under the Moon" ...
##  $ cleaned_lyric: chr  "blue moon saw soon youll take arm late beg cancel though know must killing t
```

```
str(adj_mat)
```

```
##  int [1:18990, 1:4295] 0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:18990] "X..." "X.Weird.Al..Yankovic" "X.NOT" "X.WAGGOT" ...
##   ..$ : chr [1:4295] "0001e573dc" "00021bdb1d" "0003c5dea3" "00180e338b" ...
```

Distribution of people

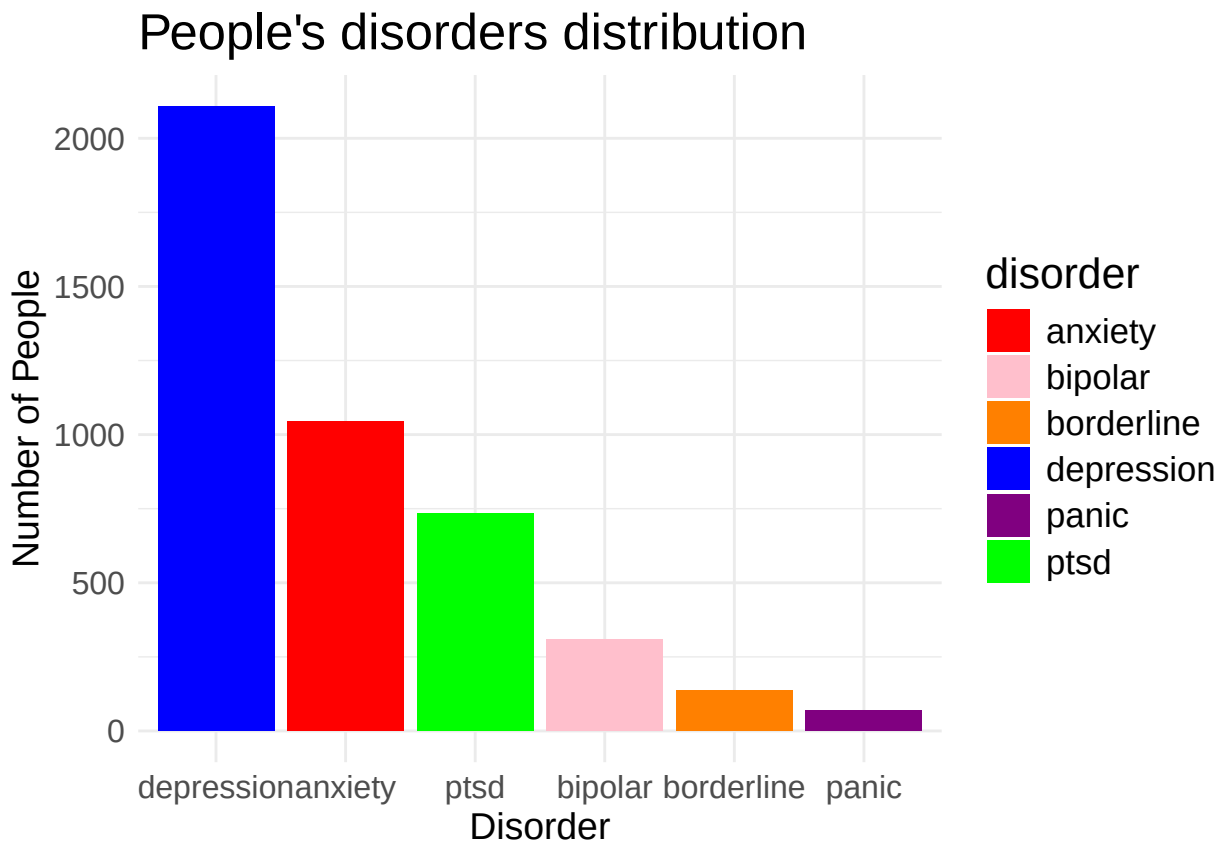
Visualizing the distribution of people

```
distr = ms_data %>%
  select(user_id, disorder)%>%
  distinct()%>%
  count(disorder)%>%
  arrange(desc(n))
distr
```

```
##     disorder    n
## 1 depression 2108
## 2    anxiety 1043
## 3       ptsd  734
## 4    bipolar  308
## 5 borderline  136
## 6      panic   68
```

```
disorder_colors = c(
  "anxiety" = rgb(1, 0, 0),       # Red
  "depression" = rgb(0, 0, 1),    # Blue
  "ptsd" = rgb(0, 1, 0),          # Green
  "borderline" = rgb(1, 0.5, 0),  # Orange
  "panic" = rgb(0.5, 0, 0.5),     # Purple
  "bipolar" = rgb(1, 0.75, 0.8)   # Pink
)
```

```
plt = distr %>% ggplot(aes(x = fct_reorder(disorder, n, .desc = TRUE), y = n, fill = disorder))+
  geom_bar(stat = "identity")+
  scale_fill_manual(values = disorder_colors) +
  ggtitle("People's disorders distribution")+
  labs(x = "Disorder", y = "Number of People")+
  theme(
    text = element_text(size = 16),
    axis.text.x = element_text(size = 12),
    axis.text.y = element_text(size = 12),
    axis.title = element_text(size = 14)
  )

ggsave("Images/Network/Disorders_distribution_plot.png", plot = plt, width = 10, height = 6)

plt
```



Network Analysis

Creating the network object

```
net = graph_from_incidence_matrix(adj_mat, directed = F)
```

```
## Warning: `graph_from_incidence_matrix()` was deprecated in igraph 1.6.0.
```

```
## i Please use ‘graph_from_biadjacency_matrix()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
str(net)
```

```
## Class 'igraph'  hidden list of 10
##  $ : num 23285
##  $ : logi FALSE
##  $ : num [1:93964] 19833 21005 21603 23062 19140 ...
##  $ : num [1:93964] 0 0 0 0 1 1 1 1 1 1 ...
##  $ : NULL
##  $ : NULL
##  $ : NULL
##  $ : NULL
##  $ :List of 4
##   ..$ : num [1:3] 1 0 1
##   ..$ : Named list()
##   ..$ :List of 2
##   .. ..$ type: logi [1:23285] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   .. ..$ name: chr [1:23285] "X..." "X.Weird.Al..Yankovic" "X.NOT" "X.WAGGOT" ...
##   ..$ : Named list()
##  $ :<environment: 0x0000020d4b2b4f90>
```

```
table(V(net)$type)
```

```
##
## FALSE  TRUE
## 18990  4295
```

```
rm(adj_mat)
```

Adding the disorders as nodes characteristics

```
V(net)$disorder = ifelse(
  V(net)$type,
  ms_data$disorder[match(V(net)$name, ms_data$user_id)],
  NA)
```

```
str(net)
```

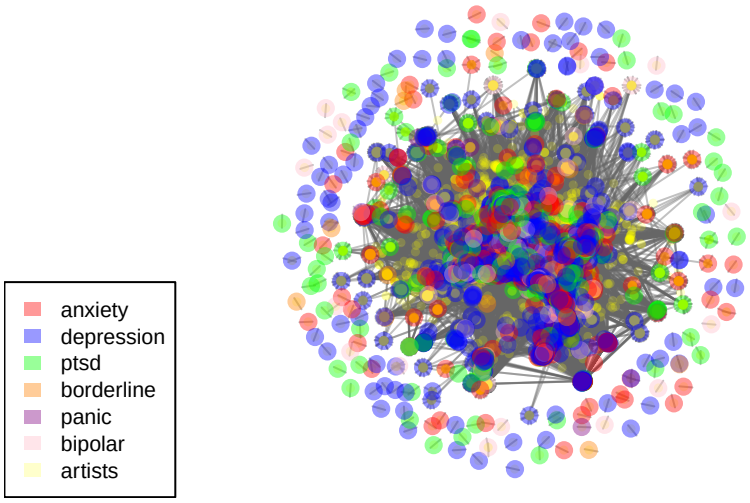
```
## Class 'igraph'  hidden list of 10
##  $ : num 23285
##  $ : logi FALSE
##  $ : num [1:93964] 19833 21005 21603 23062 19140 ...
##  $ : num [1:93964] 0 0 0 0 1 1 1 1 1 1 ...
##  $ : NULL
##  $ : NULL
##  $ : NULL
##  $ : NULL
```

```
##  $ :List of 4
##   ..$ : num [1:3] 1 0 1
##   ..$ : Named list()
##   ..$ :List of 3
##   .. ..$ type    : logi [1:23285] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   .. ..$ name    : chr [1:23285] "X..." "X.Weird.Al..Yankovic" "X.NOT" "X.WAGGOT" ...
##   .. ..$ disorder: chr [1:23285] NA NA NA NA ...
##   ..$ : Named list()
##  $ :<environment: 0x0000020d4b2b4f90>
```

First look

Let's take a first look to the network

```
# Assigning colors
disorder_colors = c(
  "anxiety" = rgb(1, 0, 0, alpha = 0.4),        # Red with transparency
  "depression" = rgb(0, 0, 1, alpha = 0.4),     # Blue with transparency
  "ptsd" = rgb(0, 1, 0, alpha = 0.4),           # Green with transparency
  "borderline" = rgb(1, 0.5, 0, alpha = 0.4),   # Orange with transparency
  "panic" = rgb(0.5, 0, 0.5, alpha = 0.4),      # Purple with transparency
  "bipolar" = rgb(1, 0.75, 0.8, alpha = 0.4)    # Pink with transparency
)

V(net)$color = ifelse(V(net)$type,
                      disorder_colors[V(net)$disorder],
                      rgb(1, 1, 0, alpha = 0.2) # Yellow with transparency
)

# Actually plotting
plot(net,
     vertex.label = NA,
     vertex.size=(1+V(net)$type)*4,
     vertex.frame.color = NA,
     edge.color = rgb(0.4,0.4,0.4, alpha = 0.4))

legend("bottomleft",
       legend = c(names(disorder_colors), "artists"),
       fill = c(disorder_colors, rgb(1, 1, 0, alpha = 0.2)),
       border = NA,
       cex = 0.7)
```



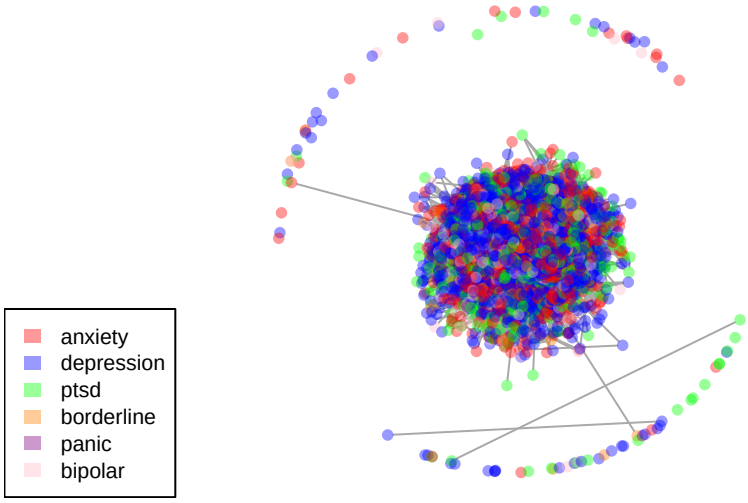
Quite a mess.. But still we can see that we have some users with no authors in common with the others, while the majority of the network is well connected. Also people with different type of diseases are well mixed. So from a first look it seems that the connection between a person and an author doesn't depend on the type of disorder he has.

Projection over the users

Let's try with the projections

```
net_bp = bipartite_projection(net)
net_pj = net_bp$proj2
l_bp = layout_with_kk(net_pj)
```

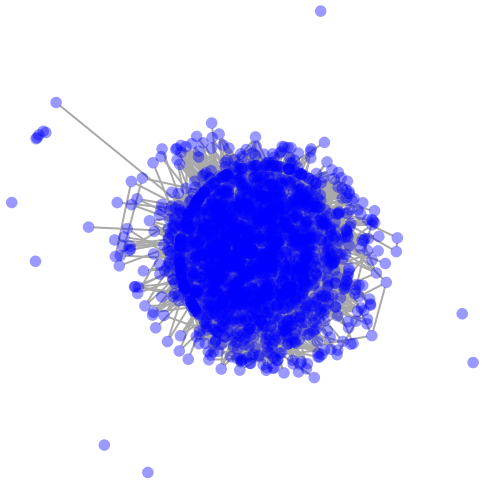
```
plot(net_pj,
     vertex.label = NA,
     vertex.size = 5,
     layout = l_bp,
     vertex.frame.color = NA)

legend("bottomleft",
       legend = names(disorder_colors),
       fill = disorder_colors,
       border = NA,
       cex = 0.7)
```



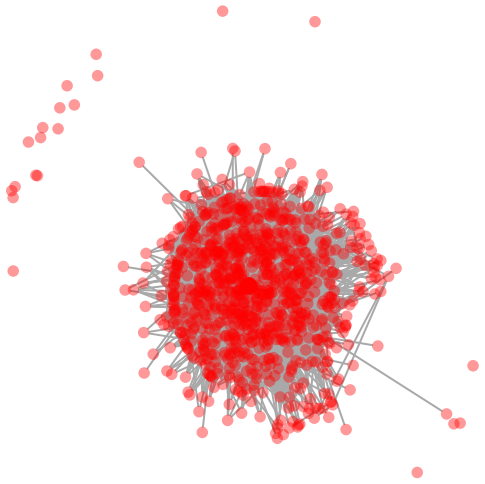
This is quite messy too, but also here we can see that users are well mixed

Let's take a look to the network of people with different diseases

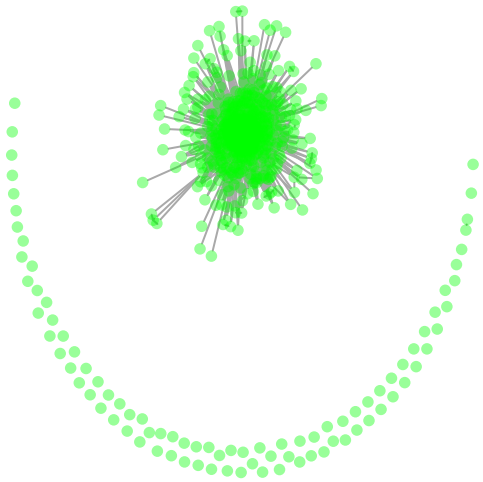
```
net_bp_dep = induced_subgraph(net_pj, vids = V(net_pj)[V(net_pj)$disorder == "depression"])

l_dep = layout_with_kk(net_bp_dep)

plot(net_bp_dep,
     vertex.label = NA,
     vertex.size = 5,
     layout = l_dep,
     vertex.frame.color = NA)
title("Depression Network")
```




```
net_bp_anx = induced_subgraph(net_pj, vids = V(net_pj)[V(net_pj)$disorder == "anxiety"])

l_anx = layout_with_kk(net_bp_anx)

plot(net_bp_anx,
     vertex.label = NA,
     vertex.size = 5,
     layout = l_anx,
     vertex.frame.color = NA)
title("Anxiety Network")
```

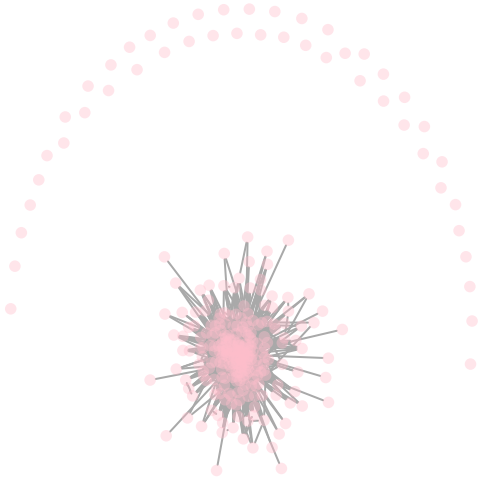



```
net_bp_ptsd = induced_subgraph(net_pj, vids = V(net_pj)[V(net_pj)$disorder == "ptsd"])

l_ptsd = layout_with_fr(net_bp_ptsd)

plot(net_bp_ptsd,
     vertex.label = NA,
     vertex.size = 5,
     layout = l_ptsd,
     vertex.frame.color = NA)
title("Ptsd Network")
```

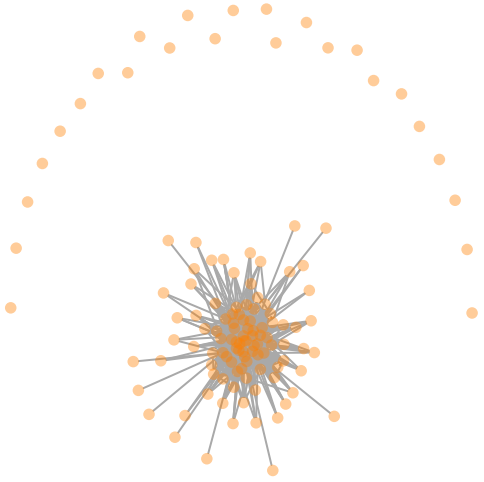



```
net_bp_bi = induced_subgraph(net_pj, vids = V(net_pj)[V(net_pj)$disorder == "bipolar"])

l_bi = layout_with_fr(net_bp_bi)

plot(net_bp_bi,
     vertex.label = NA,
     vertex.size = 5,
     layout = l_bi,
     vertex.frame.color = NA)
title("Bipolar Network")
```

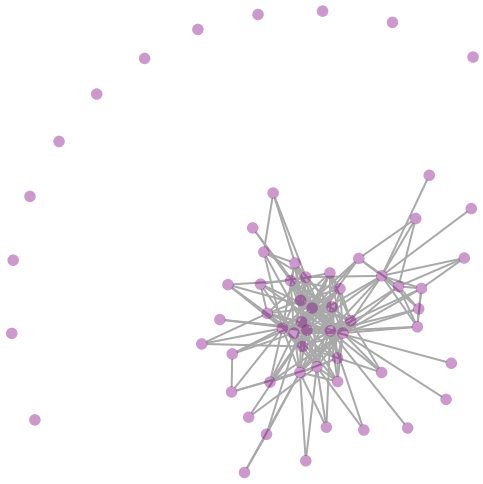
Bipolar Network



```
net_bp_bor = induced_subgraph(net_pj, vids = V(net_pj)[V(net_pj)$disorder == "borderline"])

l_bor = layout_with_fr(net_bp_bor)

plot(net_bp_bor,
     vertex.label = NA,
     vertex.size = 5,
     layout = l_bor,
     vertex.frame.color = NA)
title("Borderline Network")
```




```
net_bp_pan = induced_subgraph(net_pj, vids = V(net_pj)[V(net_pj)$disorder == "panic"])

l_pan = layout_with_fr(net_bp_pan)

plot(net_bp_pan,
     vertex.label = NA,
     vertex.size = 5,
     layout = l_pan,
     vertex.frame.color = NA)
title("Panic Network")
```



We can see that the majority of people appertaining to the same group are well connected.

Clusters

Let's take a look to a cluster of people from different groups to see also how they connect to each others

Firstly we create the clusters

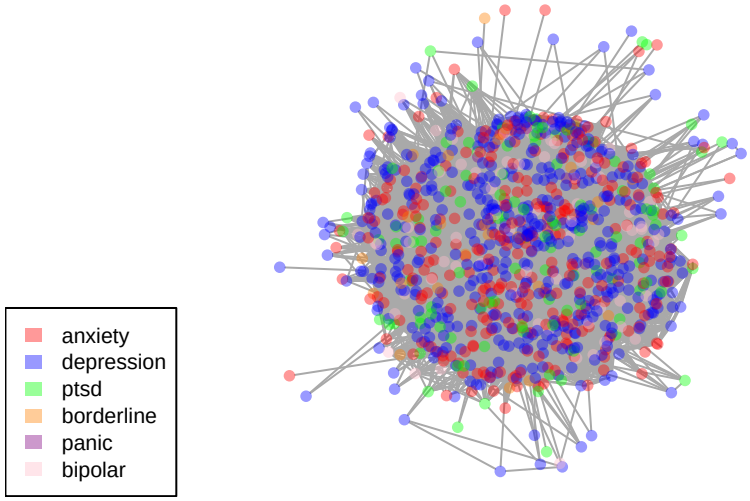
```
set.seed(1)
clusters = cluster_louvain(net_pj)
cluster_sizes = table(membership(clusters))
cluster_sizes[cluster_sizes>1]
```

```
##
##    1    2    3    4    5   10   35   89   94  105
##  984  818 1059  851  418    2    2    4    2    2
```

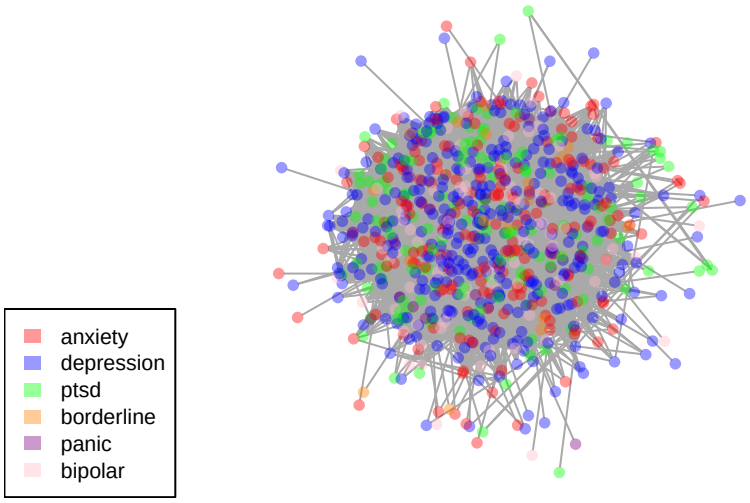
We can see that there are 5 main clusters. Let's see how they are composed.

```
for(i in 1:5){
  subgraph = induced_subgraph(net_pj, which(membership(clusters) == i))

  plot(subgraph,
       vertex.label = NA,
       vertex.size = 5,
```

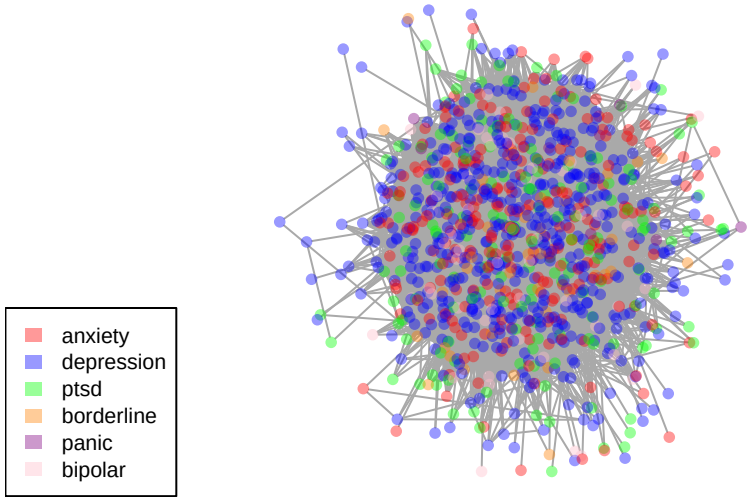
```
       layout = layout_with_kk,
       vertex.frame.color = NA)

  title(paste("Cluster", i, "(", cluster_sizes[i], "nodes)"))

  legend("bottomleft",
         legend = names(disorder_colors),
         fill = disorder_colors,
         border = NA,
         cex = 0.7)

  print(paste("Composition cluster", i))
  print(table(V(subgraph)$disorder))
}
```



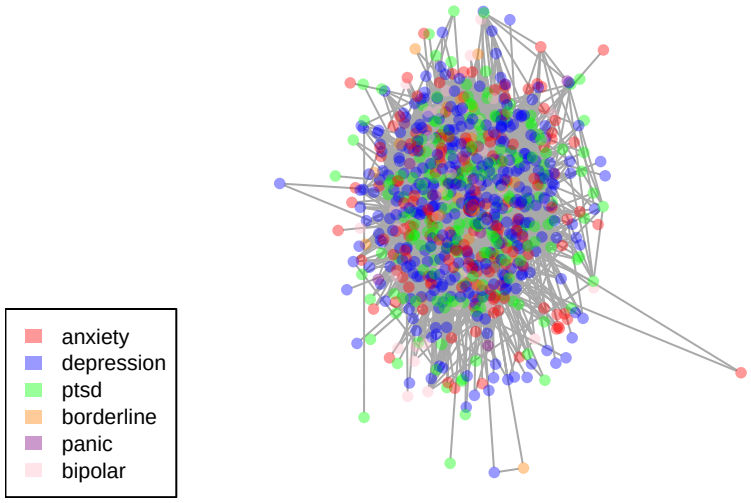

```
## [1] "Composition cluster 1"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##        287         67         25        487          9        109
```



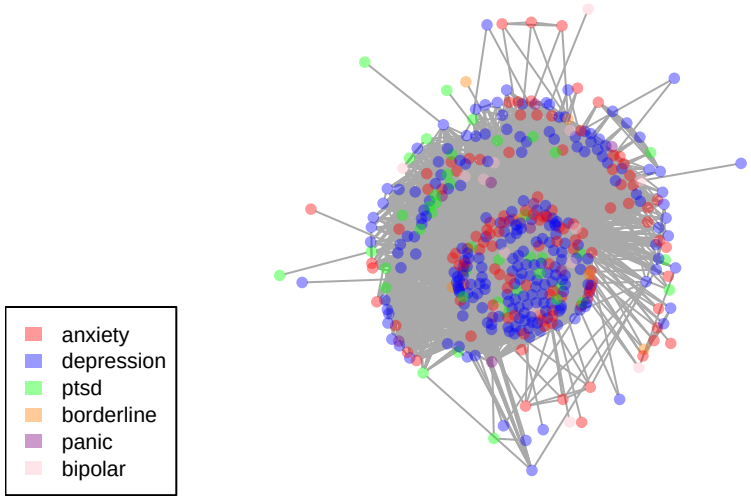

```
## [1] "Composition cluster 2"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##        192         84         21        358         14        149
```




```
## [1] "Composition cluster 3"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##        244         59         41        543         10        162
```




```
## [1] "Composition cluster 4"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##        175         52         26        366         18        214
```




```
## [1] "Composition cluster 5"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##        114         14          6        231         11         42
```

To understand better what's going on with the edges I will assign a black color to edges that are connecting two nodes with the same disorder, and leave the gray one otherwise

```
link_type = table(V(net_pj)$disorder[match(ends(net_pj, E(net_pj))[,1], V(net_pj)$name)] == V(net_pj)$di
link_type
```

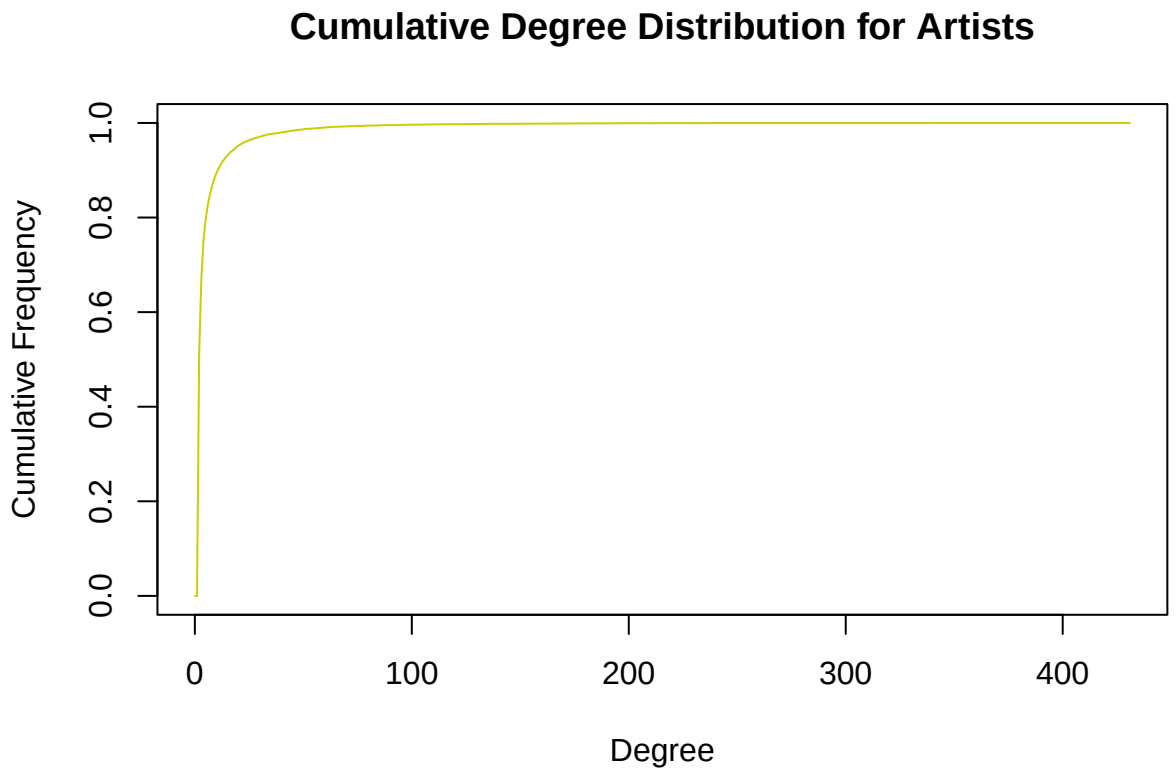
```
##
##  FALSE   TRUE
## 582774 288097
```

The last string gives FALSE if the edge is connecting two nodes with different $disorders, and TRUE otherwise. Than is summing up the numbers in a table. So in the dataset, the number of links between people with different disorders (FALSE) are 582774, and the others (TRUE) are 288097 These are a lot of edges considering that we were starting from 93964 in the bipartite network.

Let's first explain why

Degree distribution for artists nodes

```
degree_artists = degree(net, v = V(net)[!type])

deg_dist_art = degree_distribution(net, cumulative=T, v = V(net)[!type])
plot(x=0:max(degree_artists),
     y=1-deg_dist_art,
     col="yellow3",
     type = "l",
     xlab="Degree",
     ylab="Cumulative Frequency",
     main = "Cumulative Degree Distribution for Artists")
```




```
mean(degree_artists)
```

```
## [1] 4.948078
```

```
degree_artists %>% table()
```

```
## .
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 9726 3041 1518  843  587  426  342  280  228  182  164  129  118   94  107   71
##   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
##   73   82   54   68   52   41   33   38   28   37   25   25   26   27   23   24
##   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48
```

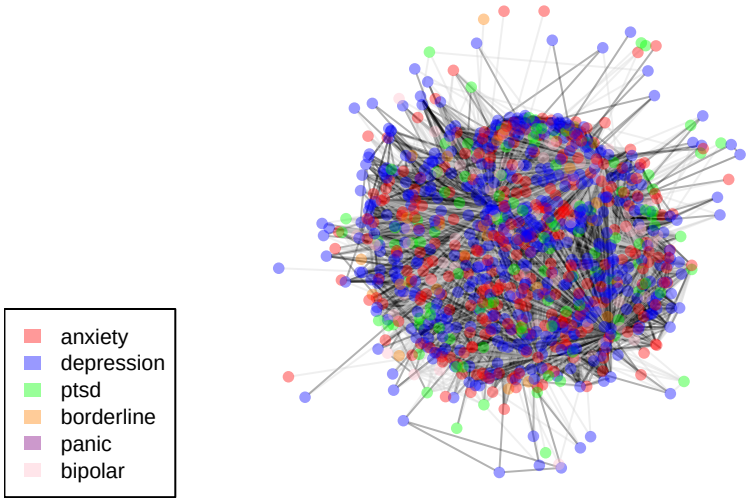
```
##   19   14   13   10   14   13   19   16   10   13   15   11   15   14    3   15
##   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64
##   11   12    5    6    5   10    5    7    7    8    6    6    8    6    5    2
##   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79   80
##    3    2    4    4    9    1    3    1    4    2    2    2    2    5    3    4
##   81   82   83   84   85   86   87   88   90   91   92   93   94   95   96   97
##    1    2    3    1    2    5    1    2    1    1    1    2    3    2    4    1
##   98   99  100  103  104  105  107  109  110  112  113  114  115  117  118  120
##    1    1    1    2    1    4    1    2    1    2    2    1    2    1    1    1
##  123  124  126  128  129  130  133  134  136  137  144  147  149  150  154  155
##    1    1    2    1    1    2    1    1    2    1    1    1    1    2    2    2
##  161  165  166  167  173  175  176  179  181  183  185  192  193  199  207  211
##    1    1    2    1    1    1    2    1    1    1    1    1    1    1    1    1
##  216  221  226  237  278  335  407  431
##    1    1    1    1    1    1    1    1
```

We can see that the average degree for artist nodes is almost 5, and only this is a factor that makes the edge count higher when we project the network over the users. On top of that, the distribution of this network has a long tail, with most artists nodes having low degrees but a few nodes having very high degrees. In other words, most artists are "niche" ones, but there are a considerable number of them that are shared between the majority of the people in the network. The second type of artists nodes are called hubs. When we project a network with hubs over the users, the edges count explodes.

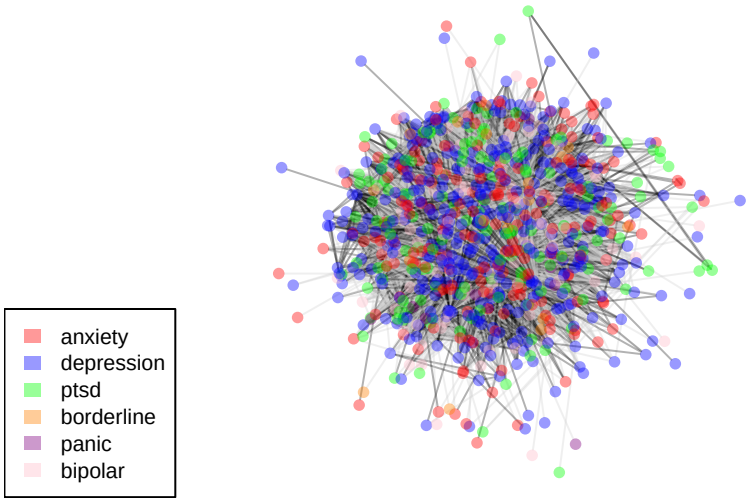
But let's go back to the visualization of the network with the distinction between edges

```
E(net_pj)$color = ifelse(V(net_pj)$disorder[match(ends(net_pj, E(net_pj))[,1], V(net_pj)$name)] == V(ne
                         rgb(0, 0, 0, alpha = 0.3),          # Black with transparency
                         rgb(0.8, 0.8, 0.8, alpha = 0.3)     # Really Light Gray with transparency
  )
```

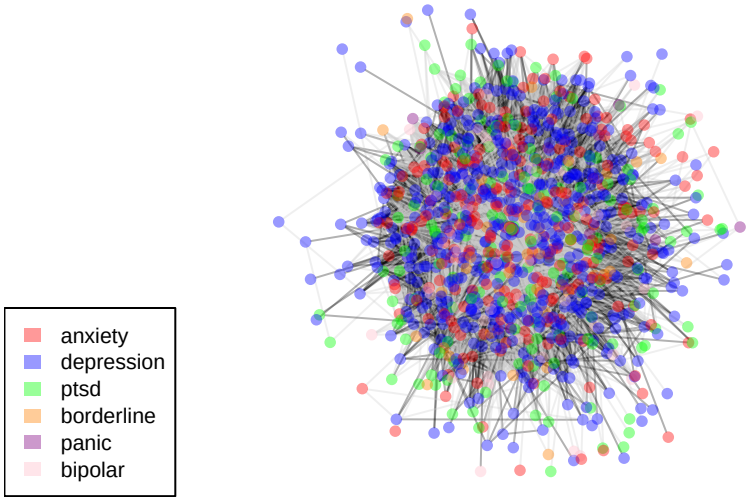
```
for(i in 1:5){
  subgraph = induced_subgraph(net_pj, which(membership(clusters) == i))

  plot(subgraph,
       vertex.label = NA,
       vertex.size = 5,
       layout = layout_with_kk,
       vertex.frame.color = NA)

  title(paste("Cluster", i, "(", cluster_sizes[i], "nodes)"))

  legend("bottomleft",
         legend = names(disorder_colors),
         fill = disorder_colors,
         border = NA,
         cex = 0.7)

  print(paste("Composition cluster", i))
  print(table(E(subgraph)$color))
  density_score = (2 * ecount(subgraph)) / (vcount(subgraph) * (vcount(subgraph) - 1))
  print(paste("Density score cluster", i, ":", density_score))
}
```



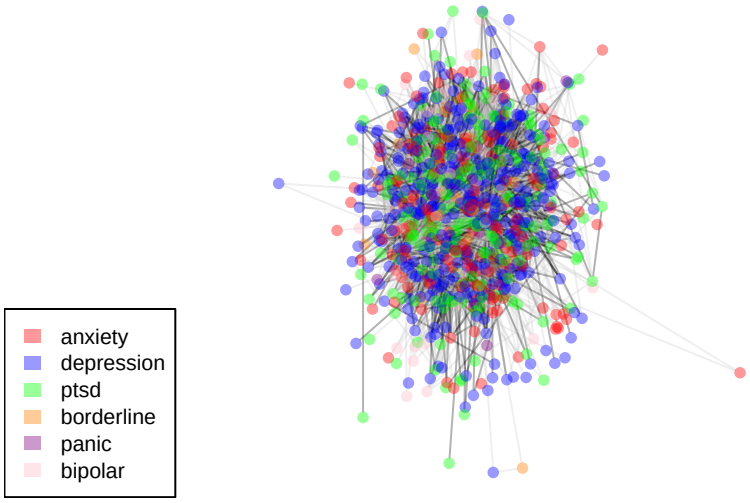

```
## [1] "Composition cluster 1"
##
## #0000004D #CCCCCC4D
##     42256     77593
## [1] "Density score cluster 1 : 0.247808269028774"
```



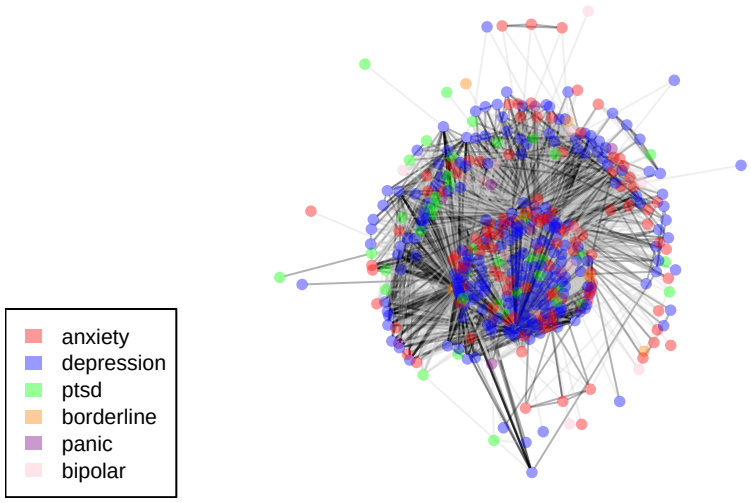

```
## [1] "Composition cluster 2"
##
## #0000004D #CCCCCC4D
##     16203     39098
## [1] "Density score cluster 2 : 0.165496045224792"
```




```
## [1] "Composition cluster 3"
##
## #0000004D #CCCCCC4D
##     33191     60455
## [1] "Density score cluster 3 : 0.167162015740498"
```




```
## [1] "Composition cluster 4"
##
## #0000004D #CCCCCC4D
##     11914     30724
## [1] "Density score cluster 4 : 0.117890371189604"
```




```
## [1] "Composition cluster 5"
##
## #0000004D #CCCCCC4D
##     11483     15735
## [1] "Density score cluster 5 : 0.312301355088178"
```

What if we try to dig deeper? We can cluster a cluster and see if we discover some strange patterns

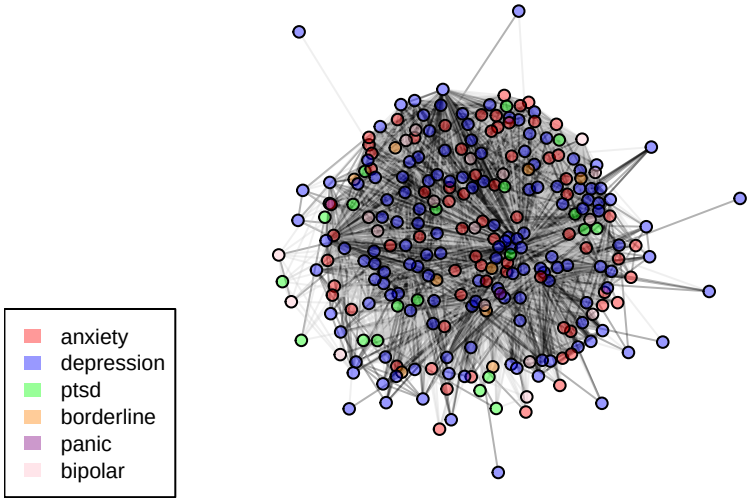
```
set.seed(1)
subgraph = induced_subgraph(net_pj, which(membership(clusters) == 1))
sub_clusters = cluster_leading_eigen(subgraph)
cluster_sizes = table(membership(sub_clusters))
cluster_sizes
```

```
##
##   1   2   3   4   5   6
## 262 280  70  74 198 100
```

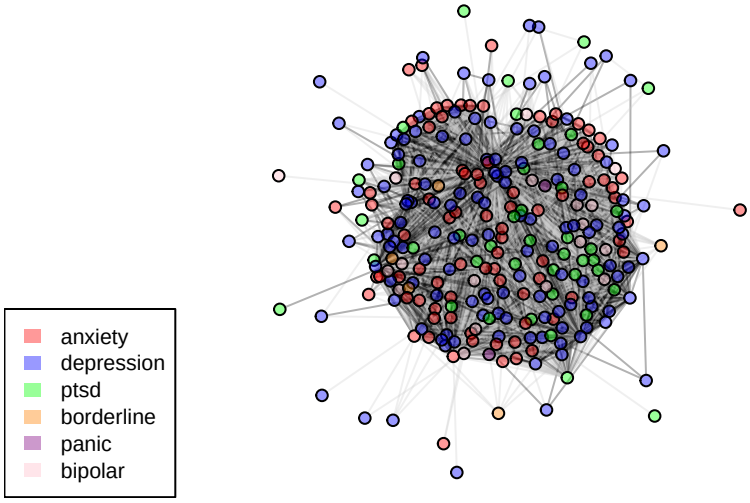
```
# Lowering vertex transparency

disorder_colors = c(
  "anxiety" = rgb(1, 0, 0, alpha = 0.4),        # Red with transparency
  "depression" = rgb(0, 0, 1, alpha = 0.4),     # Blue with transparency
  "ptsd" = rgb(0, 1, 0, alpha = 0.4),           # Green with transparency
```

```
  "borderline" = rgb(1, 0.5, 0, alpha = 0.4),  # Orange with transparency
  "panic" = rgb(0.5, 0, 0.5, alpha = 0.4),     # Purple with transparency
  "bipolar" = rgb(1, 0.75, 0.8, alpha = 0.4)   # Pink with transparency
)

V(subgraph)$color = disorder_colors[V(subgraph)$disorder]
```

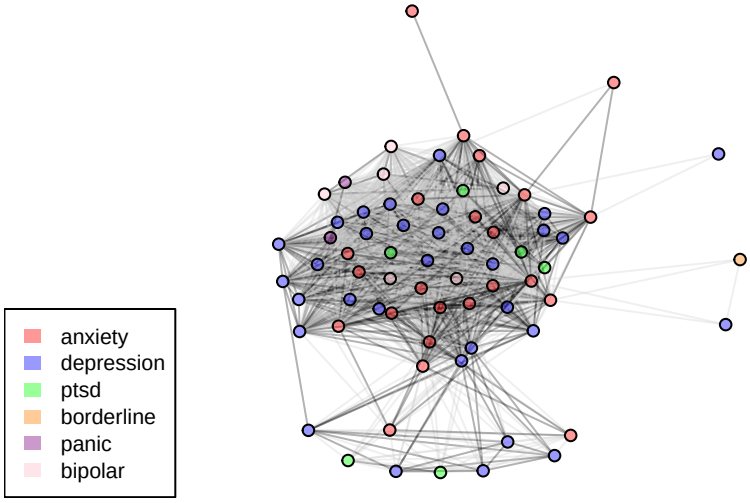
```
for(i in 1:6){
  subsubgraph = induced_subgraph(subgraph, which(membership(sub_clusters) == i))

  plot(subsubgraph,
       vertex.label = NA,
       vertex.size = 5,
       layout = layout_with_kk)

  title(paste("Subcluster", i, "(", cluster_sizes[i], "nodes)"))

  legend("bottomleft",
         legend = names(disorder_colors),
         fill = disorder_colors,
         border = NA,
         cex = 0.7)

  print(paste("Composition cluster", i))
  print(table(V(subsubgraph)$disorder))
  print(paste("Same/diffrent type edge in cluster", i))
  print(table(E(subsubgraph)$color))

  density_score = (2 * ecount(subsubgraph)) / (vcount(subsubgraph) * (vcount(subsubgraph) - 1))
  print(paste("Density score cluster", i, ":", density_score))
}
```



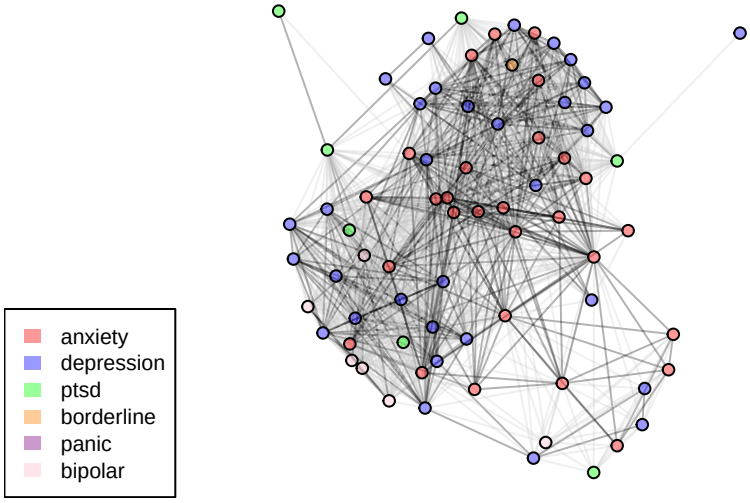

```
## [1] "Composition cluster 1"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##         70         18          9        144          1         20
## [1] "Same/diffrent type edge in cluster 1"
##
## #0000004D #CCCCCC4D
##      4244      6802
## [1] "Density score cluster 1 : 0.323067473896639"
```



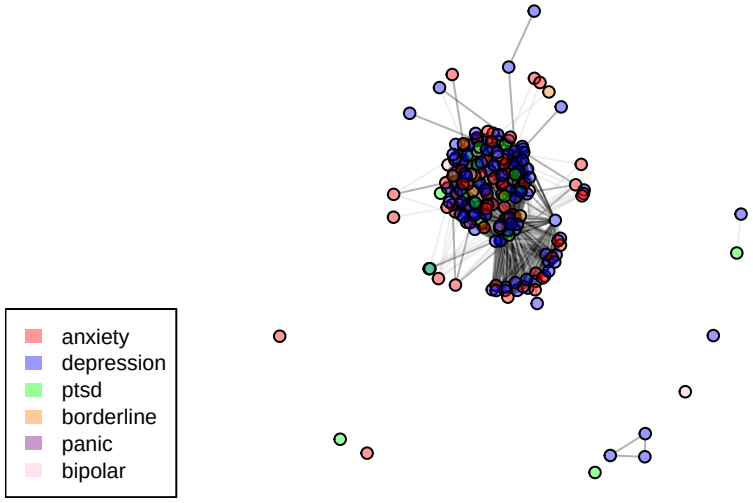

```
## [1] "Composition cluster 2"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##         85         18          5        129          2         41
## [1] "Same/diffrent type edge in cluster 2"
##
## #0000004D #CCCCCC4D
##      3464      6870
## [1] "Density score cluster 2 : 0.264567332309268"
```



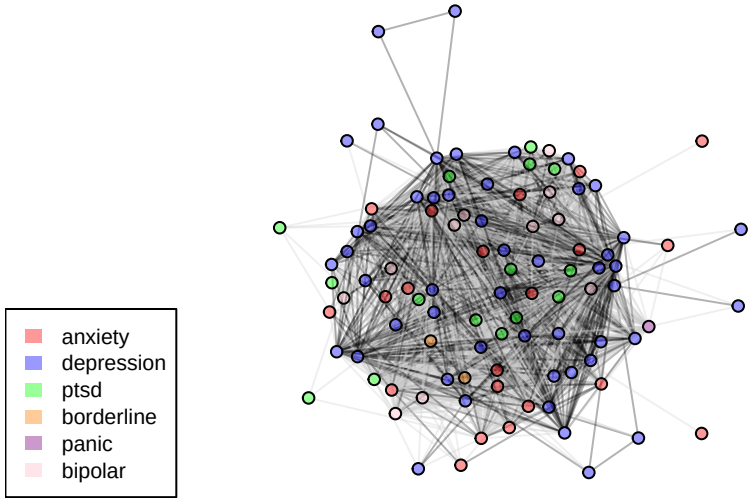

```
## [1] "Composition cluster 3"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##         23          6          1         32          2          6
## [1] "Same/diffrent type edge in cluster 3"
##
## #0000004D #CCCCCC4D
##       526      1105
## [1] "Density score cluster 3 : 0.67536231884058"
```




```
## [1] "Composition cluster 4"
##
##    anxiety    bipolar borderline depression       ptsd
##         28          6          1         32          7
## [1] "Same/diffrent type edge in cluster 4"
##
## #0000004D #CCCCCC4D
##       423       819
## [1] "Density score cluster 4 : 0.459829692706405"
```




```
## [1] "Composition cluster 5"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##         60          8          7        100          3         20
## [1] "Same/diffrent type edge in cluster 5"
##
## #0000004D #CCCCCC4D
##      3680      6848
## [1] "Density score cluster 5 : 0.539814387530124"
```




```
## [1] "Composition cluster 6"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##         21         11          2         50          1         15
## [1] "Same/diffrent type edge in cluster 6"
##
## #0000004D #CCCCCC4D
##       737      1508
## [1] "Density score cluster 6 : 0.453535353535354"
```

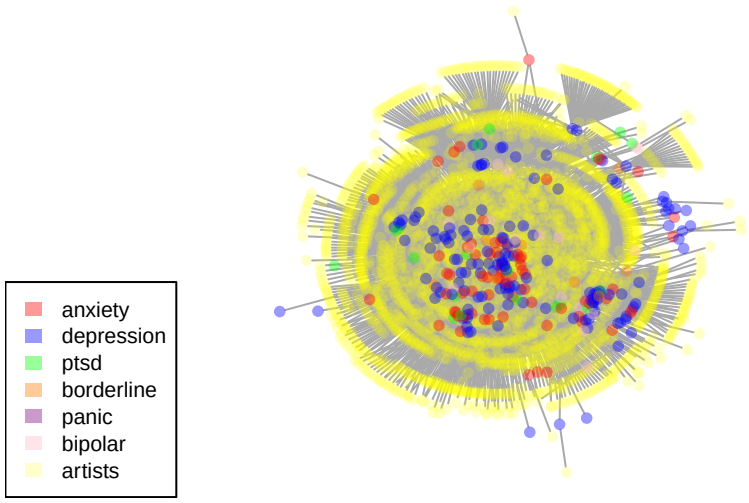
Also this really small groups are well mixed, most of them with high density score, one of which being even over 65%. That's probably due to the presence of hubs that we highlighted earlier

Back to the bipartite

Let's try to Visualize the clusters reintroducing artists nodes

```
for(i in 1:6){
  subsubgraph = induced_subgraph(subgraph, which(membership(sub_clusters) == i))
  cluster_nodes = V(subsubgraph)$name
  connecting_nodes = unique(unlist(neighborhood(net, order = 1, nodes = cluster_nodes)))

  full_subgraph = induced_subgraph(net, vids = connecting_nodes)
```

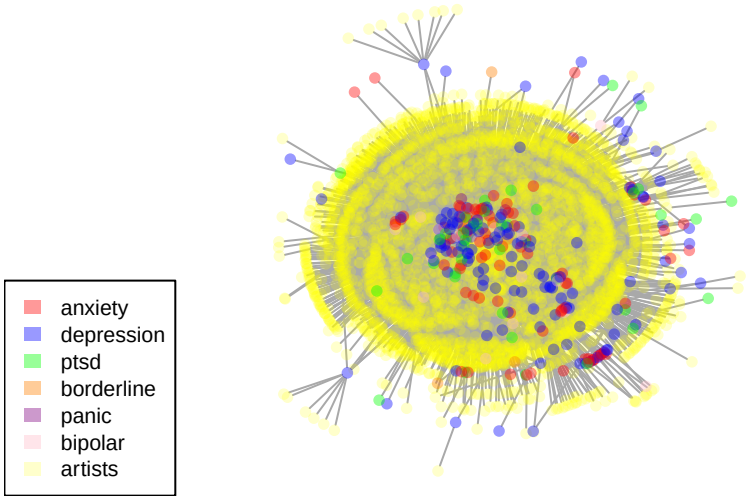
```
  plot(full_subgraph,
       vertex.label = NA,
       vertex.size = 5,
       vertex.frame.color = NA,
       layout = layout_with_kk)

  title(paste("Subcluster", i))

  legend("bottomleft",
         legend = c(names(disorder_colors), "artists"),
         fill = c(disorder_colors, rgb(1, 1, 0, alpha = 0.2)),
         border = NA,
         cex = 0.7)

  degree_artists = degree(full_subgraph, v = V(full_subgraph)[!type])

  print(paste("Artists degrees of cluster", i))
  print(table(degree_artists))
}
```




```
## [1] "Artists degrees of cluster 1"
## degree_artists
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 1457  274  112   89   48   35   25   21   13   14   17    7    6    4   12   10
##   17   18   19   20   21   22   23   24   26   27   28   29   31   32   33   34
```

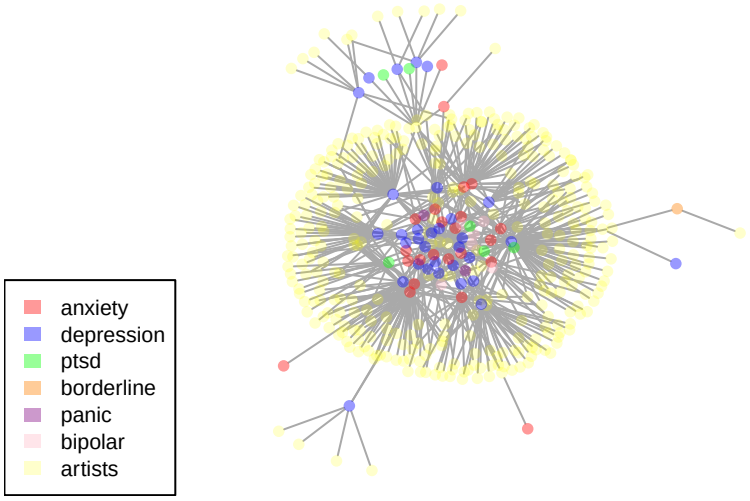
```
##    8    5    6    3    2    3    3    3    3    1    1    1    2    2    1    2
##   37   39   41   44   51   53   78   81   83
##    1    1    1    1    1    2    1    1    1
```

Subcluster 2



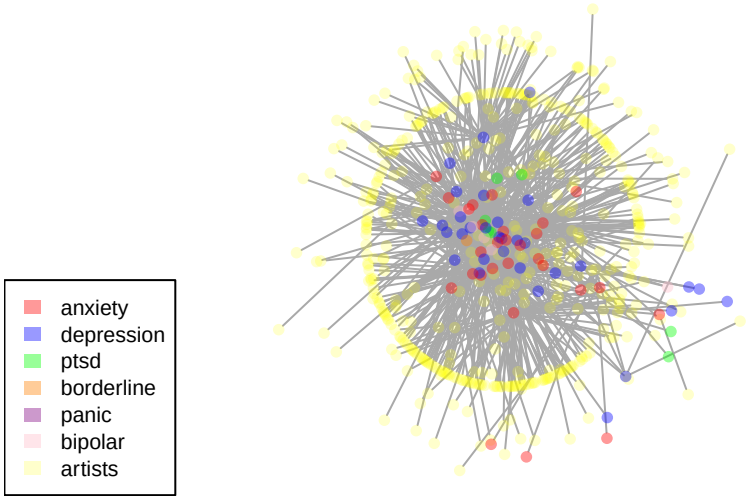

```
## [1] "Artists degrees of cluster 2"
## degree_artists
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2025  446  188  123   63   58   40   34   19   18   16   14   12   14   10    7
##   17   18   19   20   21   23   24   25   26   27   28   29   30   32   33   35
##    6    7    6    6    1    2    2    2    3    6    2    1    4    2    3    1
##   36   38   39   40   43   44   45   46   48   51
##    4    2    1    1    1    1    1    1    2    1
```

Subcluster 3



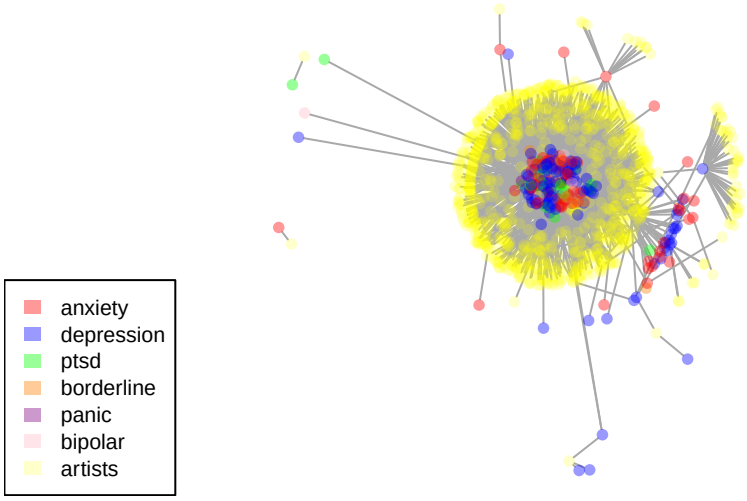

```
## [1] "Artists degrees of cluster 3"
## degree_artists
##   1   2   3   4   5   6   7  13  56
## 276  45  15   9   4   3   2   2   1
```

Subcluster 4

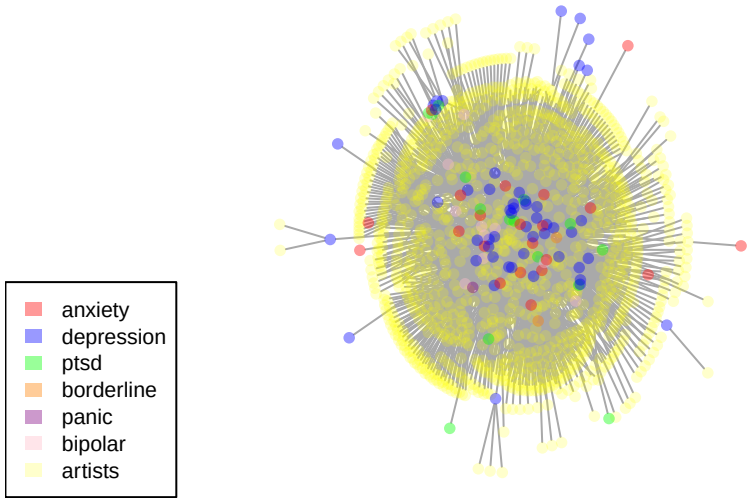



```
## [1] "Artists degrees of cluster 4"
## degree_artists
##   1   2   3   4   5   6   7   8  11  13  33  35
## 392  40  15   6   1   3   1   5   2   1   1   1
```

Subcluster 5




```
## [1] "Artists degrees of cluster 5"
## degree_artists
##   1   2   3   4   5   6   7   8   9  11  12  14  17  42 140
## 680 112  39  25  14   7   6   1   1   1   2   1   2   1   1
```



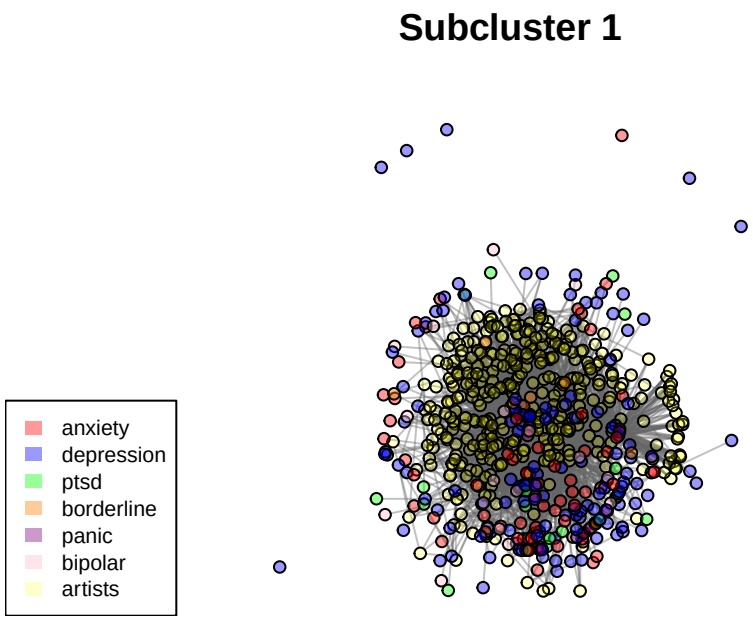

```
## [1] "Artists degrees of cluster 6"
## degree_artists
##   1   2   3   4   5   6   7   8   9  10  11  12  13  15  17  18  19  20  22  23
## 858 148  54  33  14  13   7   7   5   4   4   2   1   4   1   1   1   1   1   3
##  26  30  39
##   2   1   2
```

These plots highlight more the fact that a big chunk of artist are niche one. Let's try to highlight hubs instead, removing all artists that are listened by < 4 users

Hubs hilight

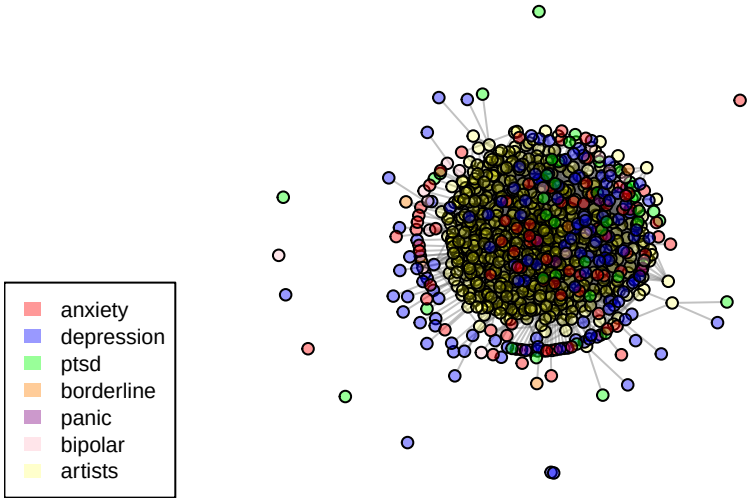
```
for(i in 1:6){
  subsubgraph = induced_subgraph(subgraph, which(membership(sub_clusters) == i))
  cluster_nodes = V(subsubgraph)$name
  connecting_nodes = unique(unlist(neighborhood(net, order = 1, nodes = cluster_nodes)))

  full_subgraph = induced_subgraph(net, vids = connecting_nodes)

  degree_artists = degree(full_subgraph, v = V(full_subgraph)[!type])
  users = V(full_subgraph)[type]
  artists = V(full_subgraph)[!type]
  high_degree_artists = artists[degree_artists > 3]
```

```
  high_degree_full_subgraph = induced_subgraph(full_subgraph, vids = c(users, high_degree_artists))

  plot(high_degree_full_subgraph,
       vertex.label = NA,
       vertex.size = 5,
       edge.color = rgb(0.4,0.4,0.4, alpha = 0.4),
       layout = layout_with_kk)

  title(paste("Subcluster", i))

  legend("bottomleft",
         legend = c(names(disorder_colors), "artists"),
         fill = c(disorder_colors, rgb(1, 1, 0, alpha = 0.2)),
         border = NA,
         cex = 0.7)

  high_degree_artists = degree(high_degree_full_subgraph, v = V(high_degree_full_subgraph)[!type])

  print(paste("Artists degrees of cluster", i))
  print(table(high_degree_artists))
}
```

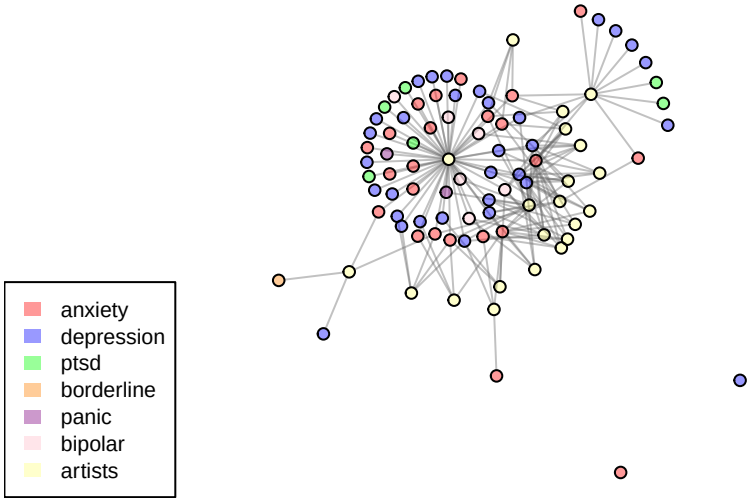



```
## [1] "Artists degrees of cluster 1"
## high_degree_artists
##  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 26 27 28 29 31
```

```
## 89 48 35 25 21 13 14 17  7  6  4 12 10  8  5  6  3  2  3  3  3  3  1  1  1  2
## 32 33 34 37 39 41 44 51 53 78 81 83
##  2  1  2  1  1  1  1  1  2  1  1  1
```

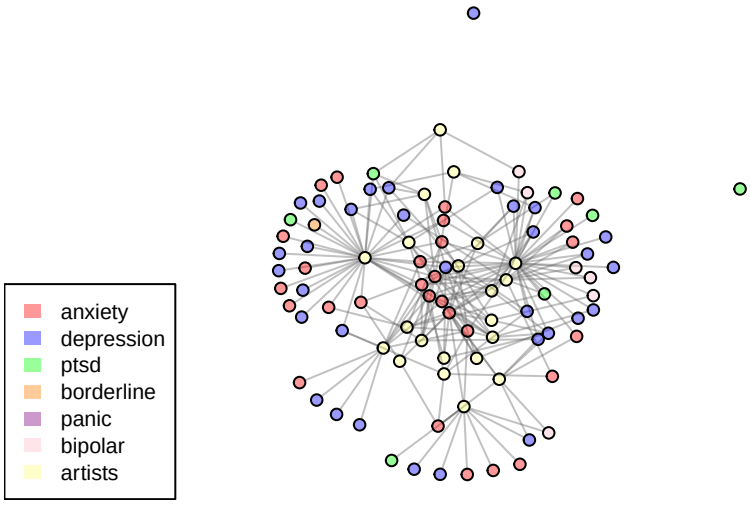



```
## [1] "Artists degrees of cluster 2"
## high_degree_artists
##   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  23  24
## 123  63  58  40  34  19  18  16  14  12  14  10   7   6   7   6   6   1   2   2
##  25  26  27  28  29  30  32  33  35  36  38  39  40  43  44  45  46  48  51
##   2   3   6   2   1   4   2   3   1   4   2   1   1   1   1   1   1   2   1
```

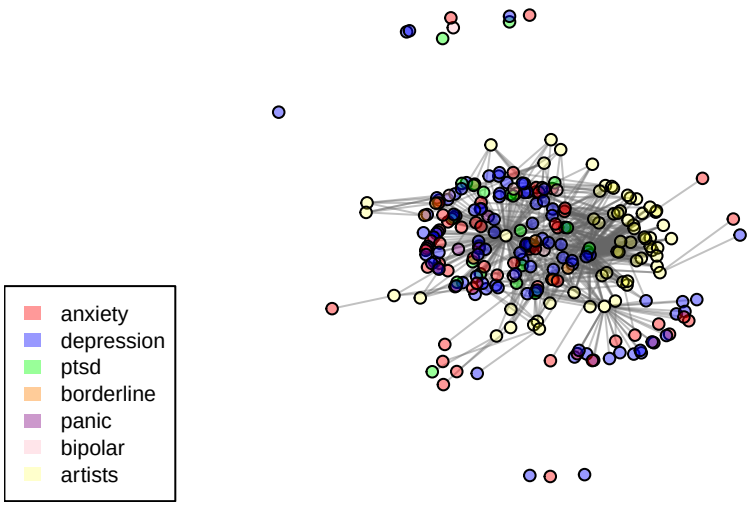



```
## [1] "Artists degrees of cluster 3"
## high_degree_artists
##  4  5  6  7 13 56
##  9  4  3  2  2  1
```

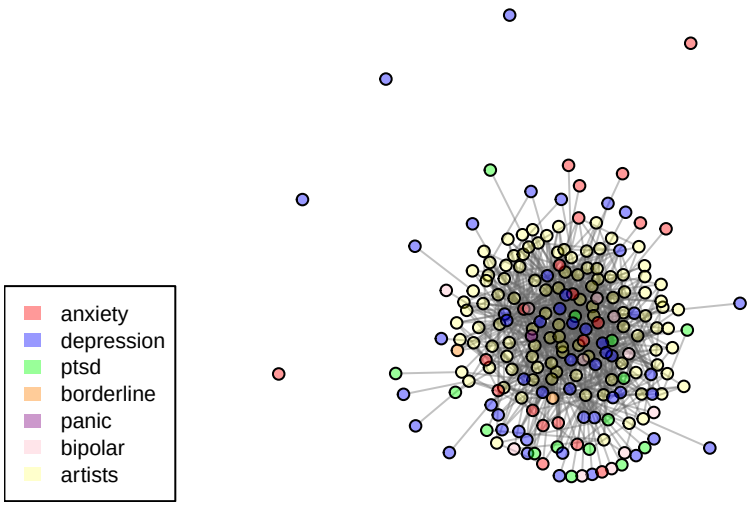
Subcluster 4




```
## [1] "Artists degrees of cluster 4"
## high_degree_artists
##  4  5  6  7  8 11 13 33 35
##  6  1  3  1  5  2  1  1  1
```




```
## [1] "Artists degrees of cluster 5"
## high_degree_artists
##   4   5   6   7   8   9  11  12  14  17  42 140
##  25  14   7   6   1   1   1   2   1   2   1   1
```



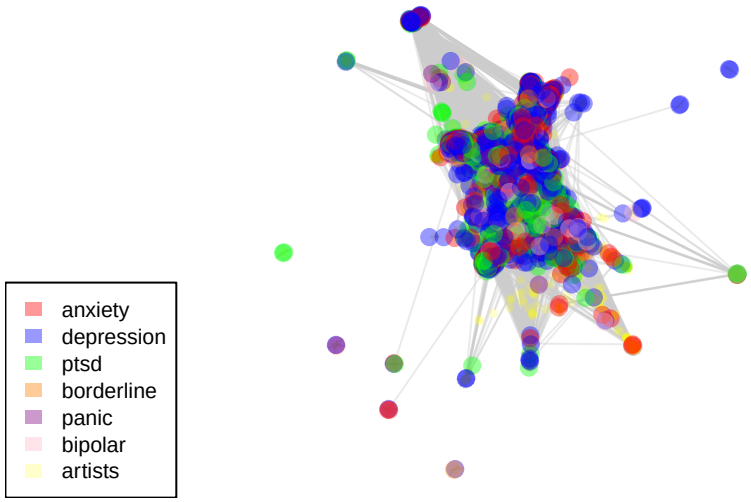

```
## [1] "Artists degrees of cluster 6"
## high_degree_artists
##  4  5  6  7  8  9 10 11 12 13 15 17 18 19 20 22 23 26 30 39
## 33 14 13  7  7  5  4  4  2  1  4  1  1  1  1  1  3  2  1  2
```

More network plotting

Now that we know that in our data the majority of the artists are listened by only one person (so they are not useful for connecting edges) we can try to plot again the network without this nodes I'm removing also disconnected user nodes for visualization purposes

```
degree_artists = degree(net, v = V(net)[!type])
artists = V(net)[!type]
artists_to_keep = artists[degree_artists > 1]

net2 = induced_subgraph(net, vids = c(artists_to_keep, V(net)[type]))

degree_users = degree(net2, v = V(net2)[type])
users = V(net2)[type]
users_to_keep = users[degree_users > 0]

net2 = induced_subgraph(net2, vids = c(V(net2)[!type], users_to_keep))

plot(net2,
```

```
     vertex.label = NA,
     vertex.size=(1+V(net2)$type)*4,
     vertex.frame.color = NA,
     edge.color = rgb(0.8,0.8,0.8, alpha = 0.4))

legend("bottomleft",
       legend = c(names(disorder_colors), "artists"),
       fill = c(disorder_colors, rgb(1, 1, 0, alpha = 0.2)),
       border = NA,
       cex = 0.7)
```



Not helpful at all, so let's try to cluster starting from the bipartite network without the last removals

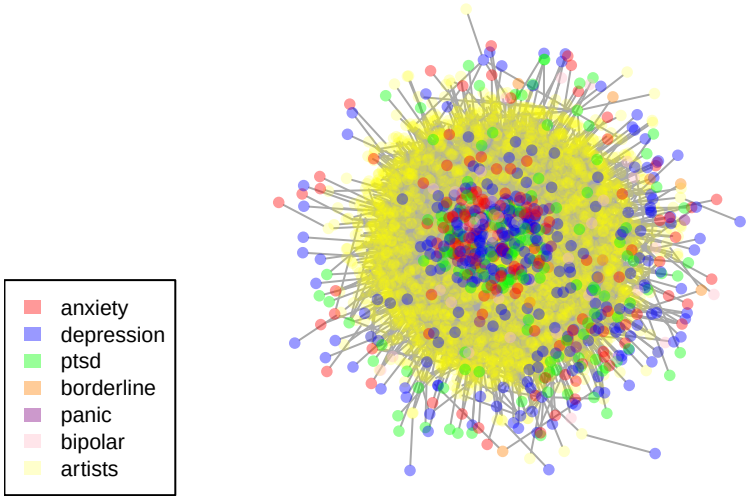
CLustering on the bipartite network

```
set.seed(0)
clusters = cluster_louvain(net2)
cluster_sizes = table(membership(clusters))
cluster_sizes
```

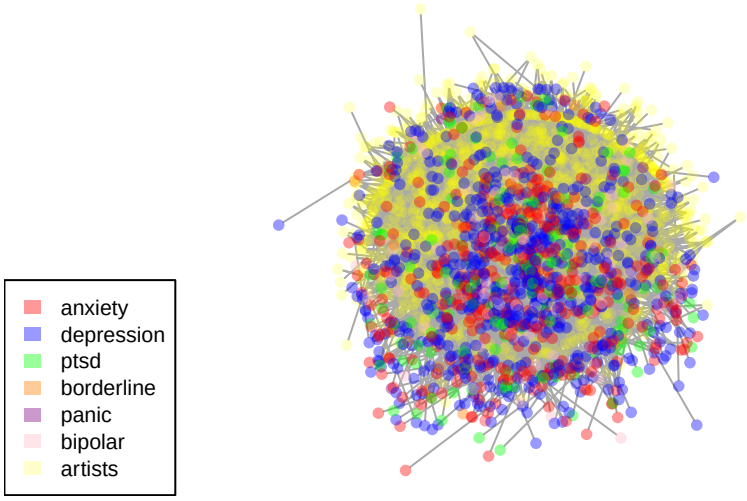
```
##
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2866 2815 1983 1694  287  479 1356  486  130  668  151  412   11    5    3   15
```

```
##   17   18   19   20   21   22   23   24   25   26   27   28
##    4    5    4    3    3    3    3    3    7    3    4    3
```

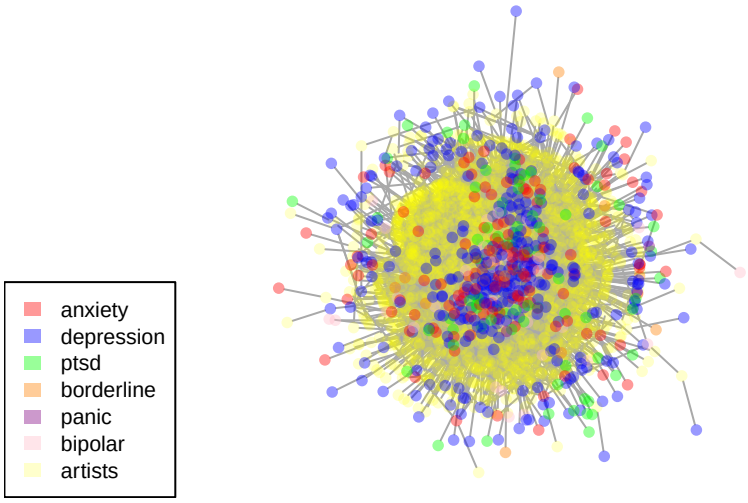
```r
for(i in 1:10){
  subgraph = induced_subgraph(net2, which(membership(clusters) == i))

  plot(subgraph,
       vertex.label = NA,
       vertex.size = 5,
       layout = layout_with_kk,
       vertex.frame.color = c(NA, "grey15")[V(net2)$type+1])

  title(paste("Cluster", i, "(", cluster_sizes[i], "nodes)"))

  legend("bottomleft",
         legend = c(names(disorder_colors), "artists"),
         fill = c(disorder_colors, rgb(1, 1, 0, alpha = 0.2)),
         border = NA,
         cex = 0.7)

  print(paste("Composition cluster", i))
  print(table(V(subgraph)$disorder))

  degree_artists = degree(subgraph, v = V(subgraph)[!type])

  print(paste("Artists degrees of cluster", i))
  print(table(degree_artists))
}
```



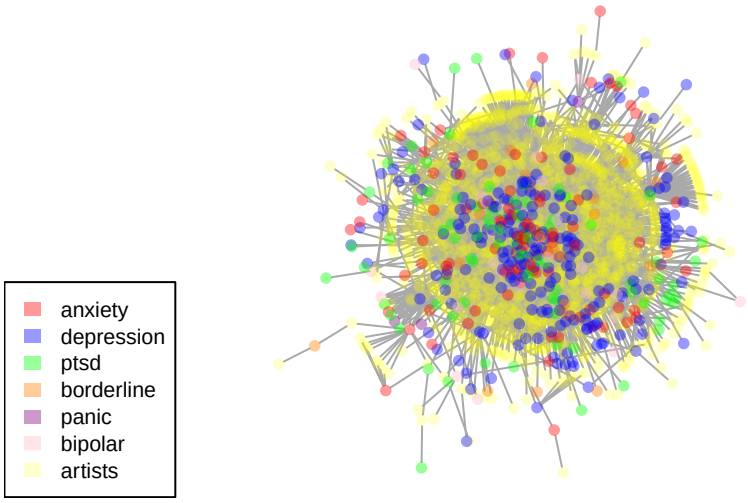

```
## [1] "Composition cluster 1"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##        152         41         18        284         14        158
## [1] "Artists degrees of cluster 1"
## degree_artists
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 186 657 349 209 136 122  79  65  53  36  22  40  39  20  19  23  20   9  11  13
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  37  38  39  44  46
##   9   7   7   7   8   5   1   2   2   5   3   7   4   3   2   4   2   2   1   2
##  47  50  54  55  59  63
##   2   1   1   1   1   2
```



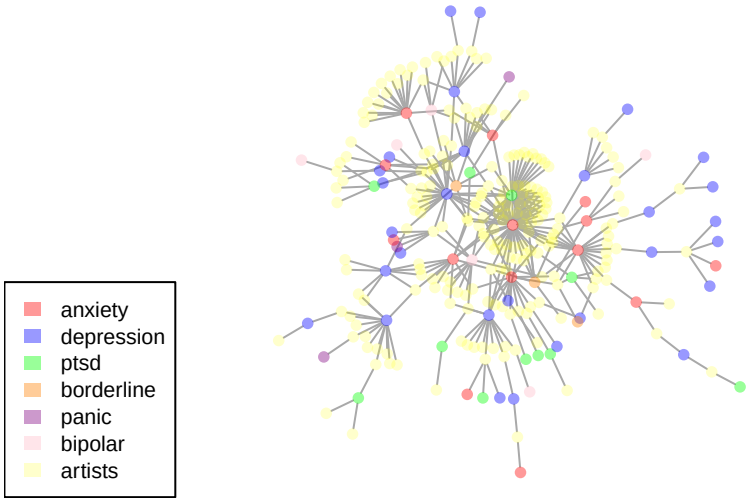

```
## [1] "Composition cluster 2"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##        395        101         33        664         18        170
## [1] "Artists degrees of cluster 2"
## degree_artists
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##  43 332 187 119 104  75  55  48  43  30  27  25  28  23  17  10  10  16  10  16
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##  12  12   6   4   7  13   5   6   6   3   6   1   7   3   3   4   6   5   6   7
##  42  43  44  45  46  48  49  50  51  53  54  55  56  58  60  61  63  64  65  66
##   1   4   7   6   5   2   1   2   6   1   4   1   1   2   1   2   2   5   1   1
##  67  69  71  72  74  75  76  78  79  81  83  87  89  92  95  98 101 102 107 110
##   1   2   2   1   2   1   1   1   2   1   1   1   1   1   1   1   2   1   1   1
## 112 113 116 121 122 126 128 137 140 141 146 185 250 287
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```



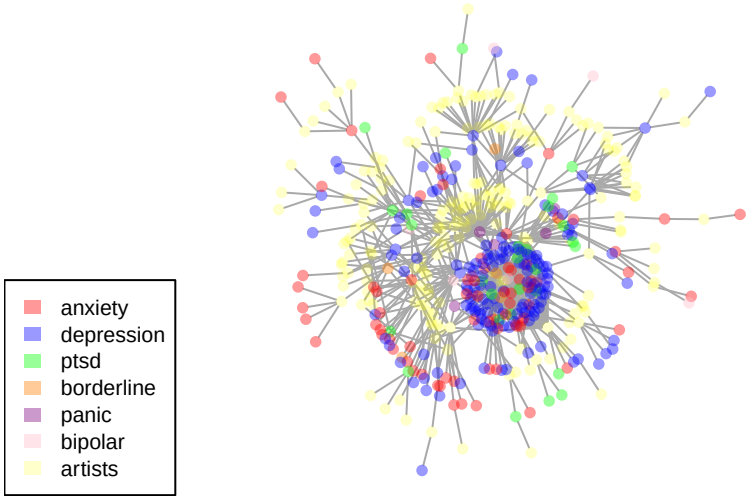

```
## [1] "Composition cluster 3"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##        133         41         19        319          3         82
## [1] "Artists degrees of cluster 3"
## degree_artists
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 287 419 193 109  69  49  36  31  19  23  18   9  11   7   9   2   9   7   7   7
##  21  22  23  24  25  26  27  28  29  30  32  33  34  36  38  39  40  41  42  43
##   7   1   2   6   4   2   2   6   3   2   1   4   1   1   1   2   1   1   1   1
##  45  46  48  55  58  59  60  66  70  72  74  78
##   3   1   1   2   1   1   1   2   1   1   1   1
```



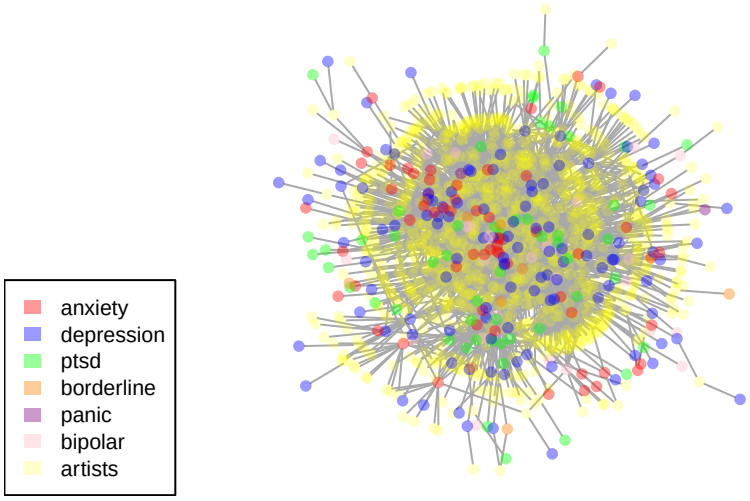

```
## [1] "Composition cluster 4"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##         94         27         22        208          6         87
## [1] "Artists degrees of cluster 4"
## degree_artists
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 260 408 191  87  56  58  33  24  23  14  12  13   5   3   4   6   5   7   4   5
##  21  22  23  24  25  26  27  28  29  30  31  35  37  39  40  46  55  56  58  69
##   3   3   2   2   2   1   1   2   1   1   3   1   1   1   1   2   1   1   1   1
##  71
##   1
```



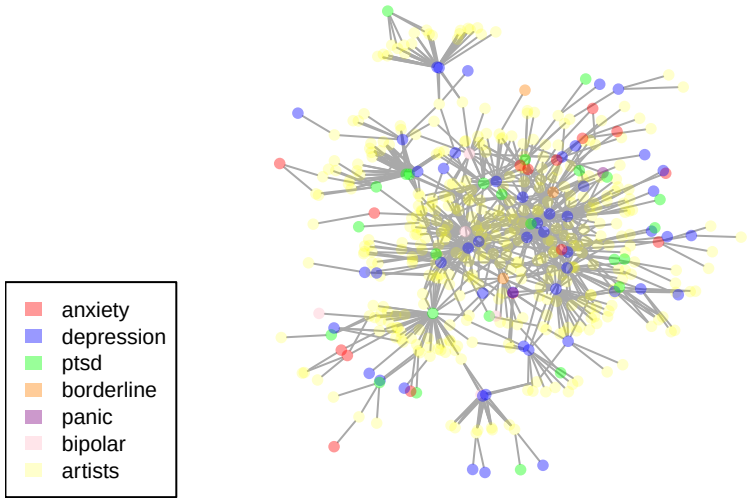

```
## [1] "Composition cluster 5"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##         14          6          3         29          3         11
## [1] "Artists degrees of cluster 5"
## degree_artists
##   1   2   3   4   5   6
## 103  88  17   9   3   1
```



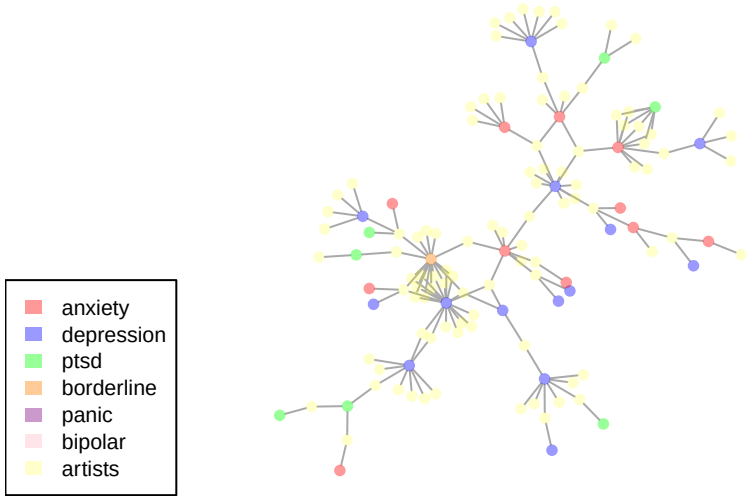

```
## [1] "Composition cluster 6"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##         72          7          5        149          8         28
## [1] "Artists degrees of cluster 6"
## degree_artists
##   1   2   3   4   5   6   7   8  10  12  14  22  28  30  34  47 127
##  93  69  20   6   5   6   1   1   1   1   1   1   1   1   1   1   1
```



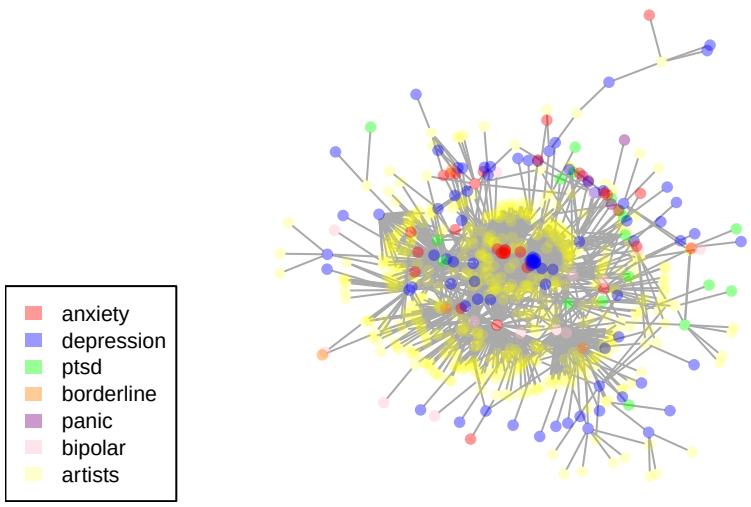

```
## [1] "Composition cluster 7"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##         61         26          9        115          3         54
## [1] "Artists degrees of cluster 7"
## degree_artists
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  19  20  25
## 455 320 138  57  41  20  18   6   6   2   3   5   7   1   2   2   1   1   1   1
##  30
##   1
```




```
## [1] "Composition cluster 8"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##         16          5          3         51          1         25
## [1] "Artists degrees of cluster 8"
## degree_artists
##   1   2   3   4   5   6   7   8   9
## 154 125  64  17  13   4   1   5   2
```




```
## [1] "Composition cluster 9"
##
##    anxiety borderline depression       ptsd
##         11          1         14          7
## [1] "Artists degrees of cluster 9"
## degree_artists
##  1  2  3  4
## 57 31  6  3
```

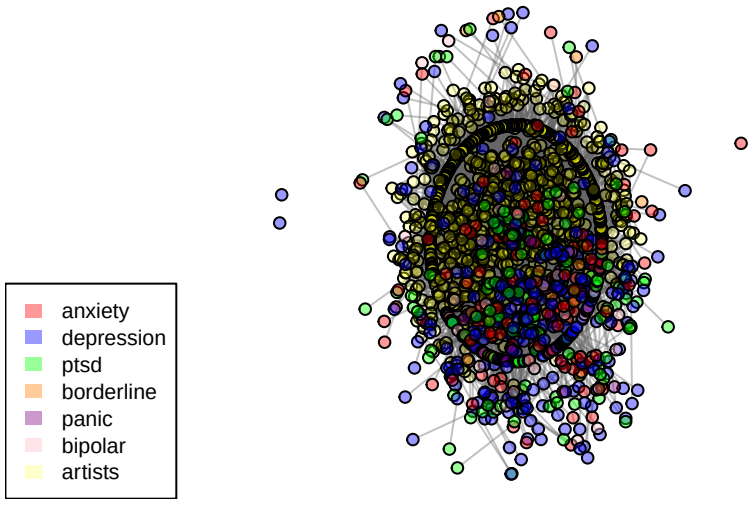



```
## [1] "Composition cluster 10"
##
##    anxiety    bipolar borderline depression      panic       ptsd
##         33         14          4         83          3         19
## [1] "Artists degrees of cluster 10"
## degree_artists
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 150 148  40  20   4   2   1   2  17  18  10   3  29  25  24   7   5   2   1   1
##  24  26  28
##   1   1   1
```

We can observe two main shapes for the plots, that are due to the presence or not of hubs in the cluster. If we do the same thing as before (removing low degree artists) here is what happens:

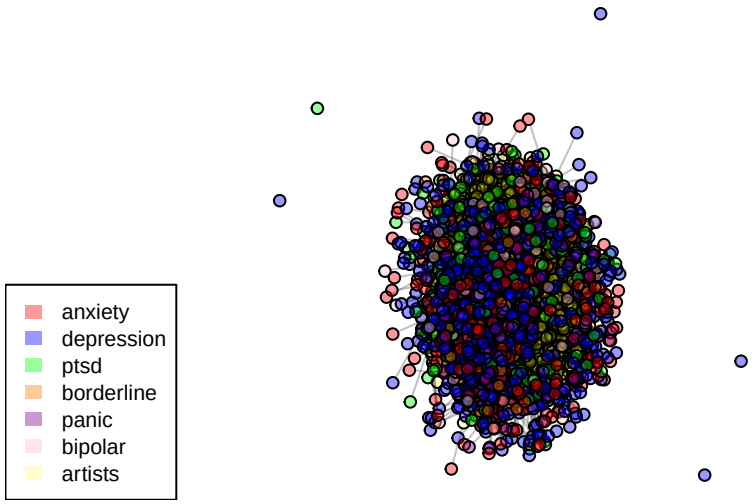
```
for(i in 1:10){
  subgraph = induced_subgraph(net2, which(membership(clusters) == i))

  degree_artists = degree(subgraph, v = V(subgraph)[!type])
  users = V(subgraph)[type]
  artists = V(subgraph)[!type]
  high_degree_artists = artists[degree_artists > 3]

  high_degree_subgraph = induced_subgraph(subgraph, vids = c(users, high_degree_artists))

  plot(high_degree_subgraph,
```

```
      vertex.label = NA,
      vertex.size = 5,
      edge.color = rgb(0.4,0.4,0.4, alpha = 0.4),
      layout = layout_with_kk)

  title(paste("Subcluster", i))

  legend("bottomleft",
         legend = c(names(disorder_colors), "artists"),
         fill = c(disorder_colors, rgb(1, 1, 0, alpha = 0.2)),
         border = NA,
         cex = 0.7)

  high_degree_artists = degree(high_degree_subgraph, v = V(high_degree_subgraph)[!type])

  print(paste("Artists degrees of cluster", i))
  print(table(high_degree_artists))
}
```

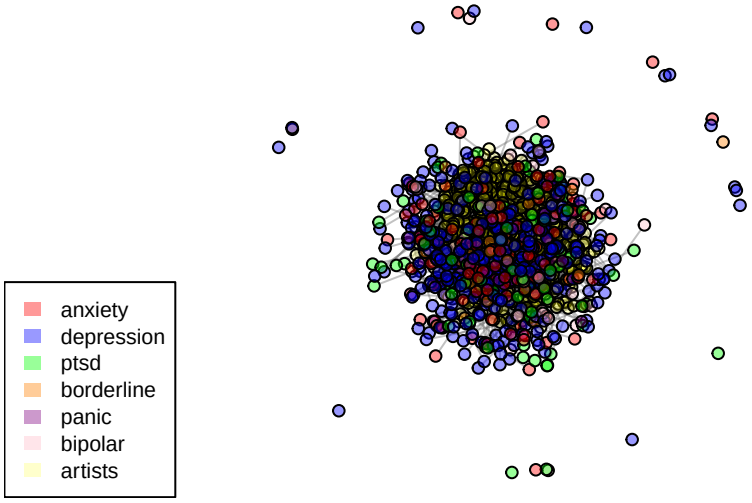



```
## [1] "Artists degrees of cluster 1"
## high_degree_artists
##   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23
## 209 136 122  79  65  53  36  22  40  39  20  19  23  20   9  11  13   9   7   7
##  24  25  26  27  28  29  30  31  32  33  34  35  37  38  39  44  46  47  50  54
##   7   8   5   1   2   2   5   3   7   4   3   2   4   2   2   1   2   2   1   1
```

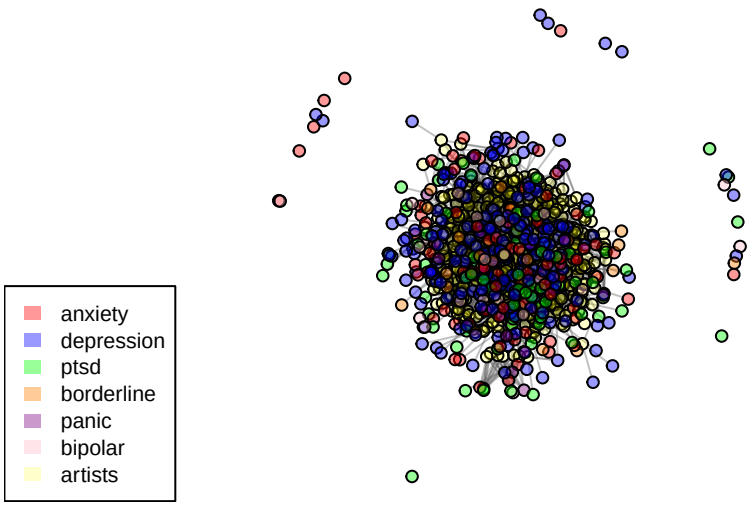
```
##  55  59  63
##   1   1   2
```



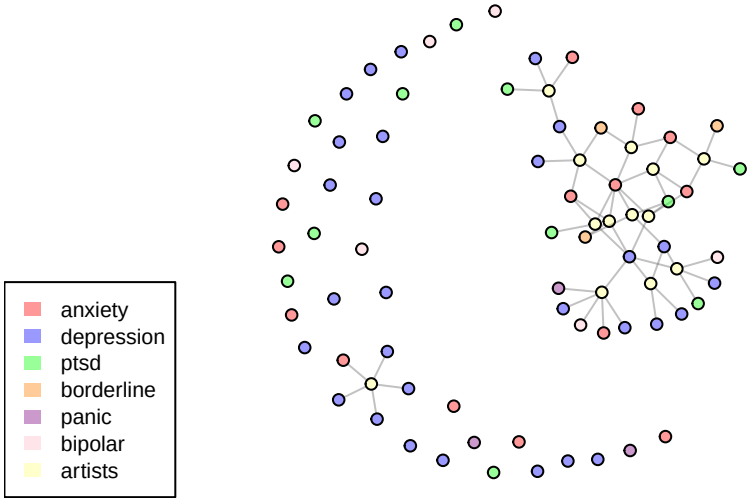

```
## [1] "Artists degrees of cluster 2"
## high_degree_artists
##   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23
## 119 104  75  55  48  43  30  27  25  28  23  17  10  10  16  10  16  12  12   6
##  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  42  43  44
##   4   7  13   5   6   6   3   6   1   7   3   3   4   6   5   6   7   1   4   7
##  45  46  48  49  50  51  53  54  55  56  58  60  61  63  64  65  66  67  69  71
##   6   5   2   1   2   6   1   4   1   1   2   1   2   2   5   1   1   1   2   2
##  72  74  75  76  78  79  81  83  87  89  92  95  98 101 102 107 110 112 113 116
##   1   2   1   1   1   2   1   1   1   1   1   1   1   2   1   1   1   1   1   1
## 121 122 126 128 137 140 141 146 185 250 287
##   1   1   1   1   1   1   1   1   1   1   1
```



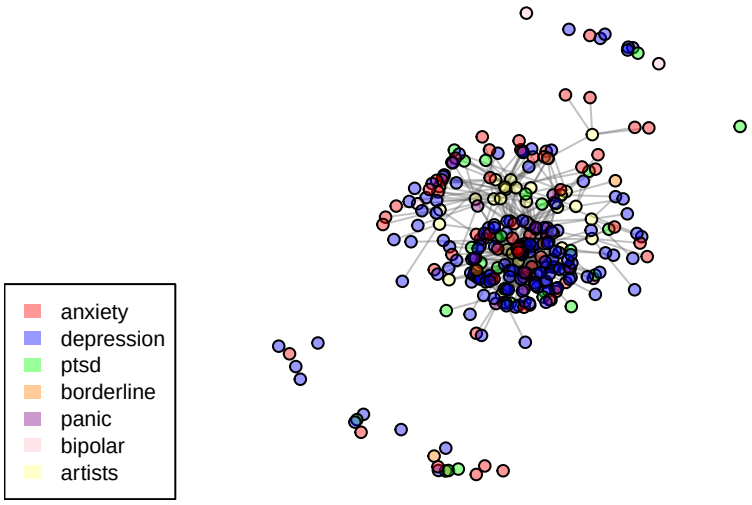

```
## [1] "Artists degrees of cluster 3"
## high_degree_artists
##   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23
## 109  69  49  36  31  19  23  18   9  11   7   9   2   9   7   7   7   7   1   2
##  24  25  26  27  28  29  30  32  33  34  36  38  39  40  41  42  43  45  46  48
##   6   4   2   2   6   3   2   1   4   1   1   1   2   1   1   1   1   3   1   1
##  55  58  59  60  66  70  72  74  78
##   2   1   1   1   2   1   1   1   1
```



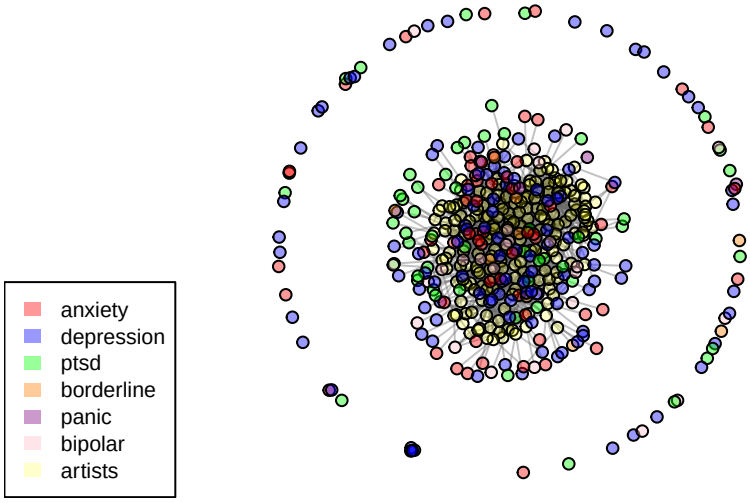

```
## [1] "Artists degrees of cluster 4"
## high_degree_artists
##  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
## 87 56 58 33 24 23 14 12 13  5  3  4  6  5  7  4  5  3  3  2  2  2  1  1  2  1
## 30 31 35 37 39 40 46 55 56 58 69 71
##  1  3  1  1  1  1  2  1  1  1  1  1
```



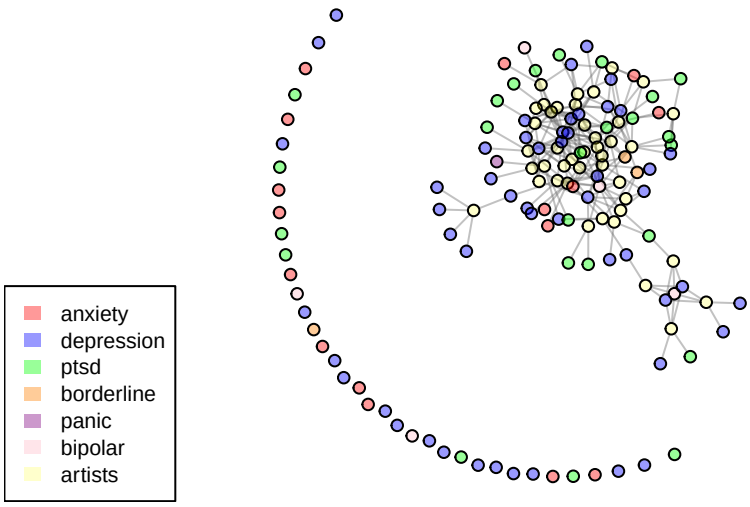

```
## [1] "Artists degrees of cluster 5"
## high_degree_artists
## 4 5 6
## 9 3 1
```



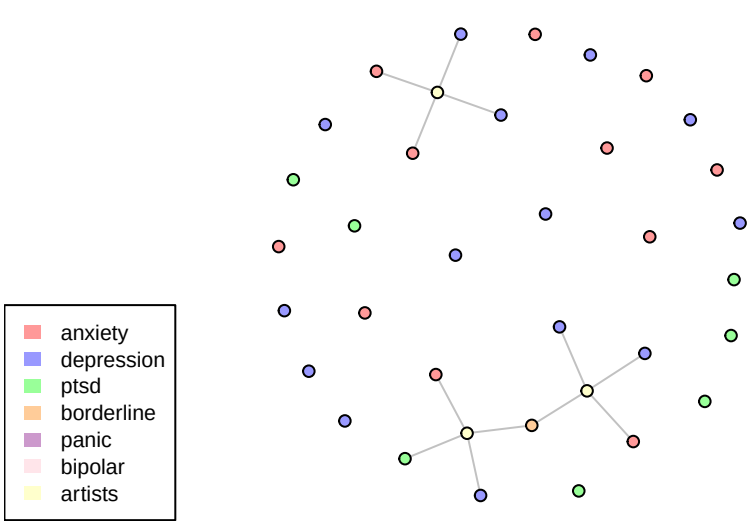

```
## [1] "Artists degrees of cluster 6"
## high_degree_artists
##   4   5   6   7   8  10  12  14  22  28  30  34  47 127
##   6   5   6   1   1   1   1   1   1   1   1   1   1   1
```



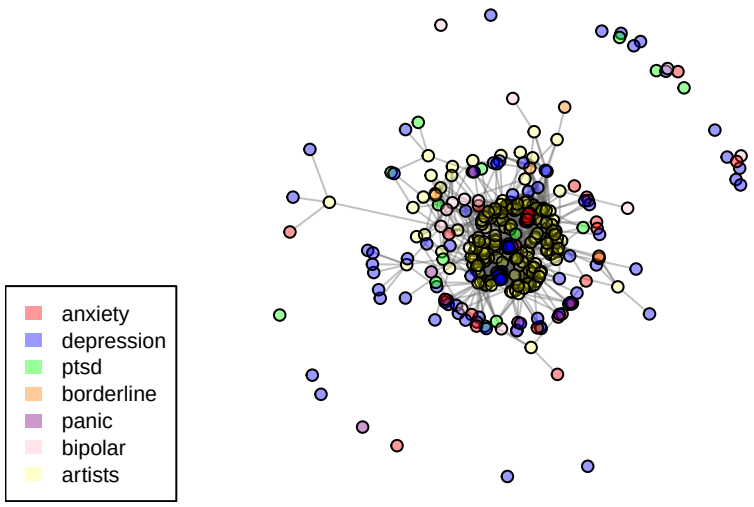

```
## [1] "Artists degrees of cluster 7"
## high_degree_artists
##  4  5  6  7  8  9 10 11 12 13 14 15 16 17 19 20 25 30
## 57 41 20 18  6  6  2  3  5  7  1  2  2  1  1  1  1  1
```




```
## [1] "Artists degrees of cluster 8"
## high_degree_artists
##  4  5  6  7  8  9
## 17 13  4  1  5  2
```




```
## [1] "Artists degrees of cluster 9"
## high_degree_artists
## 4
## 3
```




```
## [1] "Artists degrees of cluster 10"
## high_degree_artists
##  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 24 26 28
## 20  4  2  1  2 17 18 10  3 29 25 24  7  5  2  1  1  1  1  1
```

Communities that where clustered around hubs remain mostly intact and maintain the shape of one giant component. On the other hand, smaller communities without high influential points lost a good percentage of people from the main network.

Lyrics analysis

Sentiment analysis

So far we learned that people with the same disorders are not spread around the same artists. But same artists can produce different types of songs, with different topics and different emotions attached to them Let's try to dig into this aspect

```
#library(syuzhet)
#sentiment = get_nrc_sentiment(ms_data$cleaned_lyric)
```

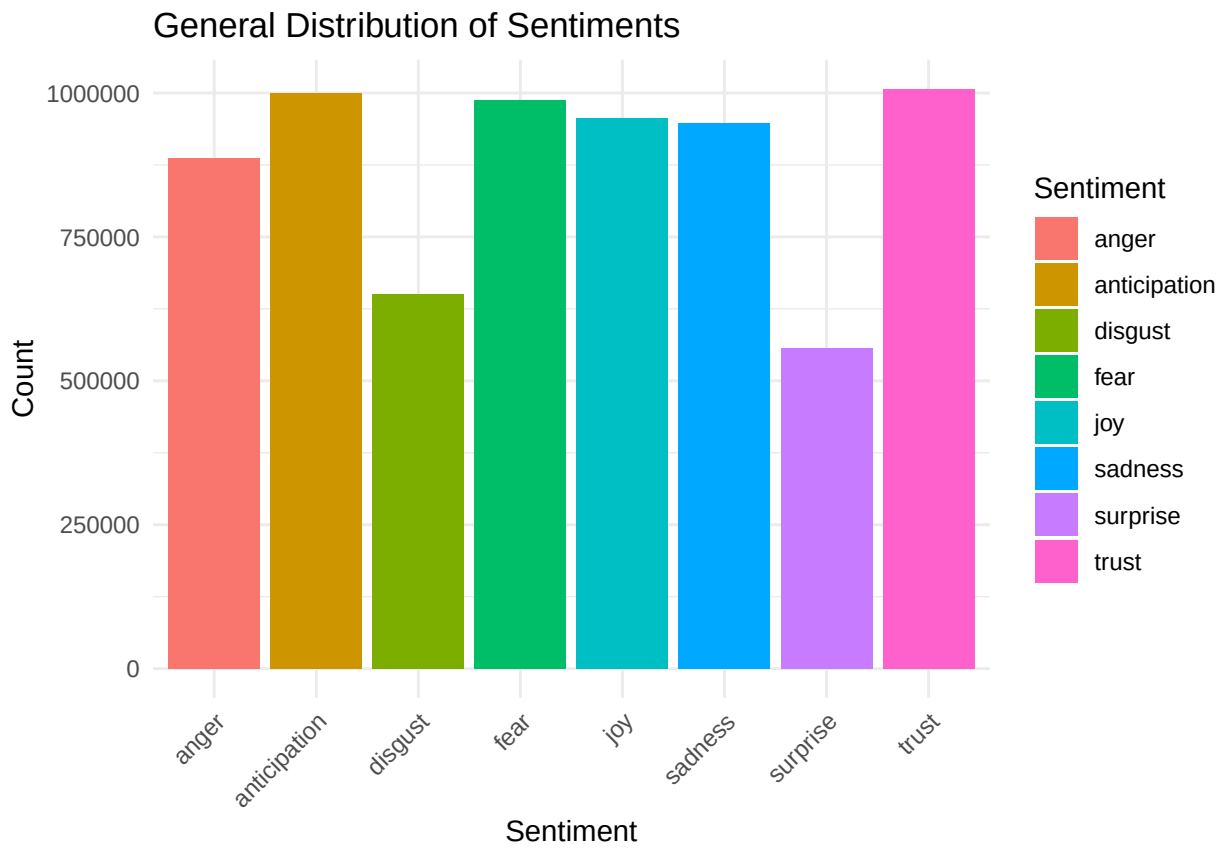
```
#head(sentiment)
```

```r
#ms_sent_data = cbind(ms_data, sentiment)
#write.csv(ms_sent_data, "final_cleaned_lyrics_with_sentiment.csv", row.names = FALSE)
```

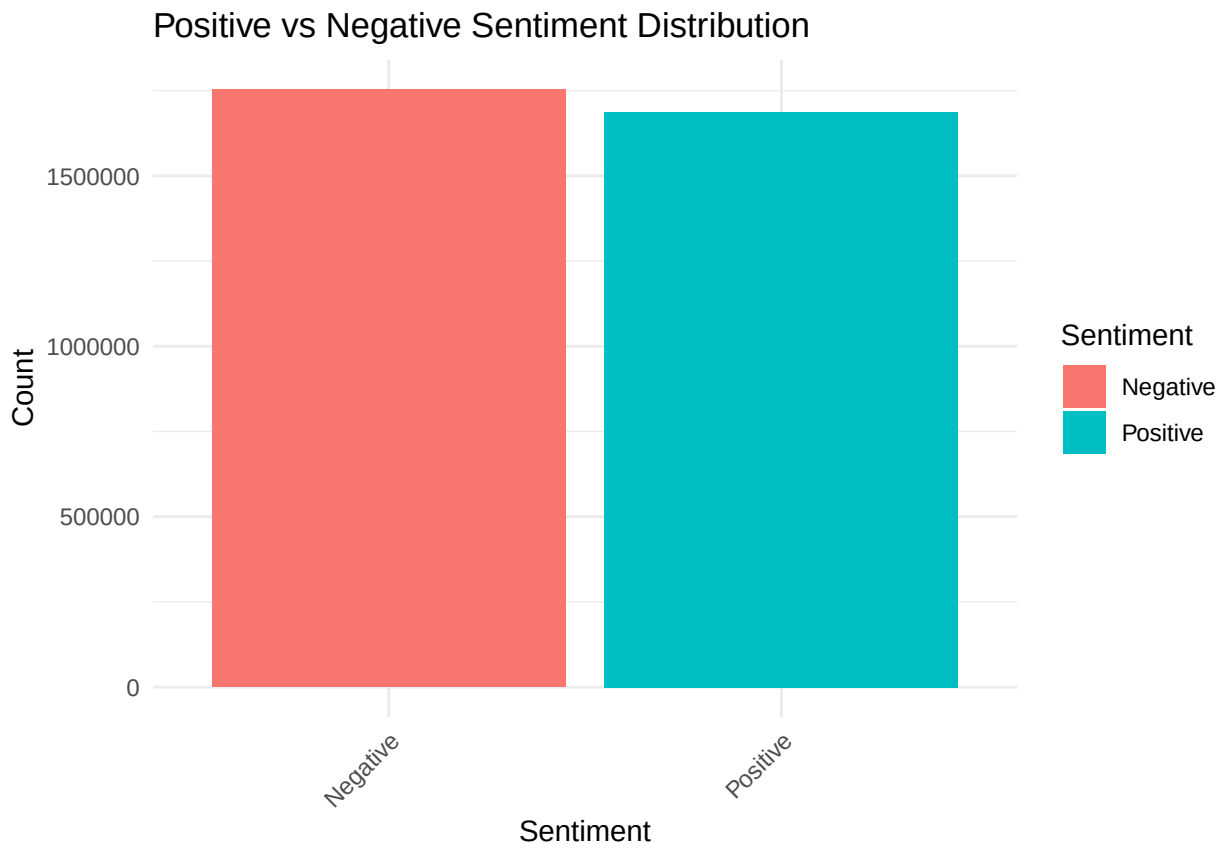
```r
ms_sent_data = read.csv("final_cleaned_lyrics_with_sentiment.csv")
head(ms_sent_data)
```

```
##      user_id disorder              artist                                title
## 1 4353e884c1  anxiety Echo & the Bunnymen                   The Killing Moon
## 2 4353e884c1  anxiety        Turin Brakes                          Red Moon
## 3 4353e884c1  anxiety           Radiohead                   Sail To The Moon
## 4 4353e884c1  anxiety               Peace                     Under the Moon
## 5 4353e884c1  anxiety             Karen O The Moon Song - Studio Version Duet
## 6 4353e884c1  anxiety       Joni Mitchell                  Moon At The Window
##
## 1
## 2                                                    x2 sometimes letting go easier dead friend ca
## 3
## 4
## 5
## 6 take cheerful resignation heart humility thats take cheerful person told nobody harder could nobody
##   anger anticipation disgust fear joy sadness surprise trust negative positive
## 1     2            3       0    2   0       5        0     0        7        1
## 2     2            1       3    3   2       4        1     2        4        3
## 3     1            1       0    2   1       0        0     2        3        3
## 4     4            2       4    3   3       3        0     4        7        3
## 5     2            3       2    1   3       3        0     2        2        5
## 6     4            2       2    6   4       5        5     3        8        6
```

```r
# Summarize the general distribution of sentiments
sentiment_totals = colSums(ms_sent_data[, c("anger", "anticipation", "disgust", "fear", "joy",
                                            "sadness", "surprise", "trust")])

# Create a data frame for plotting
sentiment_df = data.frame(
  Sentiment = names(sentiment_totals),
  Count = as.numeric(sentiment_totals)
)

# Plot for general distribution of sentiments
ggplot(sentiment_df, aes(x = Sentiment, y = Count, fill = Sentiment)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(
    title = "General Distribution of Sentiments",
    x = "Sentiment",
    y = "Count"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Summarize the positive/negative sentiments
pos_neg_totals = colSums(ms_sent_data[, c("positive", "negative")])

# Create a data frame for positive/negative plotting
pos_neg_df = data.frame(
  Sentiment = c("Positive", "Negative"),
  Count = as.numeric(pos_neg_totals)
)

# Plot for positive/negative sentiment distribution
ggplot(pos_neg_df, aes(x = Sentiment, y = Count, fill = Sentiment)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(
    title = "Positive vs Negative Sentiment Distribution",
    x = "Sentiment",
    y = "Count"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



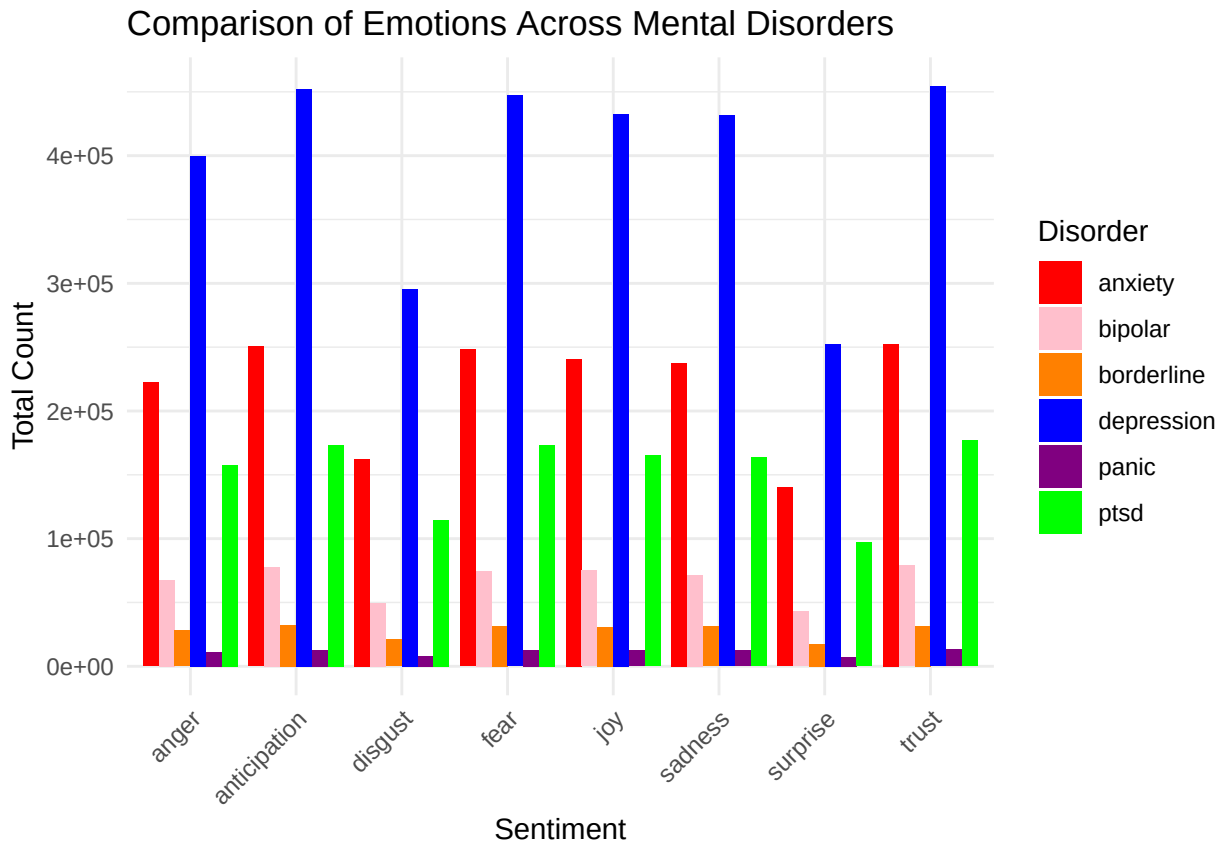

```
disorder_colors = c(
  "anxiety" = rgb(1, 0, 0),        # Red
  "depression" = rgb(0, 0, 1),     # Blue
  "ptsd" = rgb(0, 1, 0),           # Green
  "borderline" = rgb(1, 0.5, 0),   # Orange
  "panic" = rgb(0.5, 0, 0.5),      # Purple
  "bipolar" = rgb(1, 0.75, 0.8)    # Pink
)
```

```
# Select relevant columns (disorder and sentiment scores)
relevant_columns = c("disorder", "anger", "anticipation", "disgust", "fear",
                     "joy", "sadness", "surprise", "trust")

# Aggregate sentiment scores by disorder
disorder_sentiments = ms_sent_data[, relevant_columns] %>%
  group_by(disorder) %>%
  summarise(across(everything(), sum, na.rm = TRUE)) %>%
  pivot_longer(cols = -disorder, names_to = "Sentiment", values_to = "Count")
```

```
## Warning: There was 1 warning in `summarise()`.
## i In argument: `across(everything(), sum, na.rm = TRUE)`.
## i In group 1: `disorder = "anxiety"`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
```

```
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

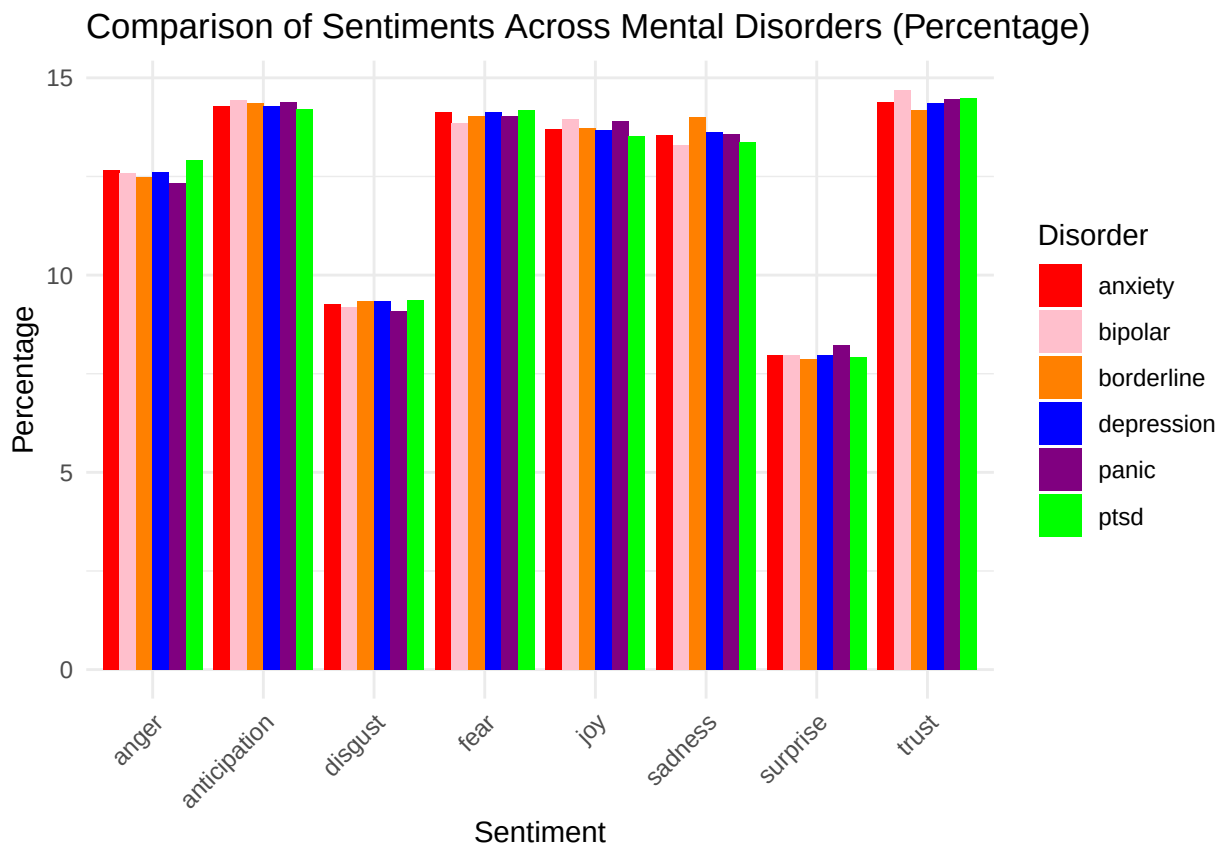
```
# Plot emotions comparison between disorders
ggplot(disorder_sentiments, aes(x = Sentiment, y = Count, fill = disorder)) +
  scale_fill_manual(values = disorder_colors) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(
    title = "Comparison of Emotions Across Mental Disorders",
    x = "Sentiment",
    y = "Total Count",
    fill = "Disorder"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



This is not really helpful because the plot pretty much highlight always the general distribution of users between disorders. Let's try to work with percentages

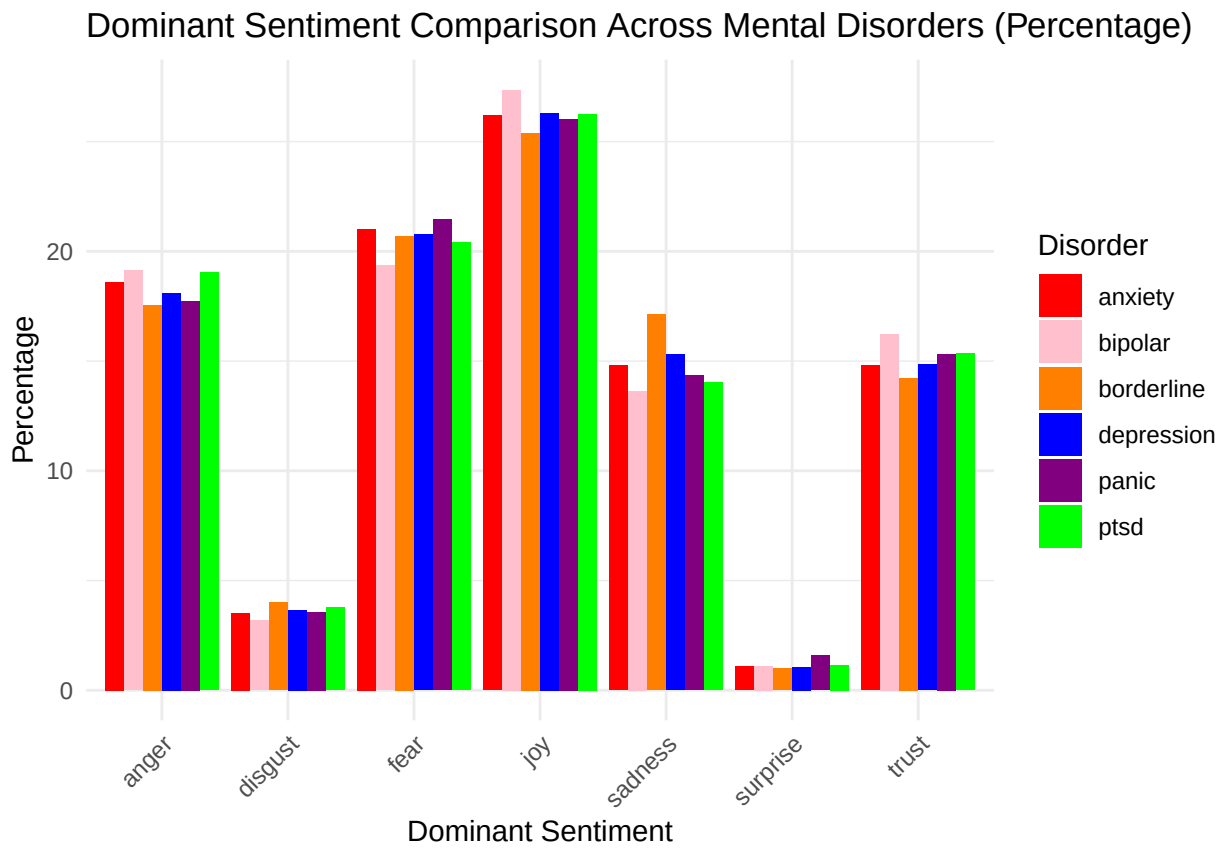
```
# Select relevant columns (disorder and sentiment scores)
relevant_columns = c("disorder", "anger", "anticipation", "disgust", "fear",
                     "joy", "sadness", "surprise", "trust")
```

```r
# Aggregate sentiment scores by disorder
disorder_sentiments = ms_sent_data[, relevant_columns] %>%
  group_by(disorder) %>%
  summarise(across(everything(), sum, na.rm = TRUE)) %>%
  mutate(Total = rowSums(across(where(is.numeric)))) %>%
  pivot_longer(cols = -c(disorder, Total), names_to = "Sentiment", values_to = "Count") %>%
  mutate(Percentage = (Count / Total) * 100)

# Plot emotions comparison as percentages
ggplot(disorder_sentiments, aes(x = Sentiment, y = Percentage, fill = disorder)) +
  scale_fill_manual(values = disorder_colors) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(
    title = "Comparison of Sentiments Across Mental Disorders (Percentage)",
    x = "Sentiment",
    y = "Percentage",
    fill = "Disorder"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Too similar, doesn't give any messages. That because every song is categorized as multiple sentiments, giving a score that is proportional to how much that sentiment is present in the song. Let's try to take only the highest sentiment for each song

```r
# Select relevant columns (disorder and sentiment scores)
relevant_columns = c("disorder", "anger", "disgust", "fear",
                     "joy", "sadness", "surprise", "trust")

# Identify the highest sentiment for each row and assign +1
disorder_sentiments = ms_sent_data[, relevant_columns] %>%
  mutate(dominant_sentiment = apply(.[, -1], 1, function(x) names(x)[which.max(x)])) %>%
  group_by(disorder, dominant_sentiment) %>%
  summarise(count = n(), .groups = "drop") %>%
  # Sum all counts per disorder
  group_by(disorder) %>%
  mutate(total_count = sum(count)) %>%
  # Calculate percentage for each sentiment
  mutate(percentage = (count / total_count) * 100)

# Plot the results with custom colors
ggplot(disorder_sentiments, aes(x = dominant_sentiment, y = percentage, fill = disorder)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = disorder_colors) +  # Use custom colors
  theme_minimal() +
  labs(
    title = "Dominant Sentiment Comparison Across Mental Disorders (Percentage)",
    x = "Dominant Sentiment",
    y = "Percentage",
    fill = "Disorder"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Topic Analysis

The topic modelling has been done in Python using the Top2Vec library. Here we have the dataset that includes the topic, the list of more common words for each topic and the topic name that I assigned based on the more common words.

```
ms_topic = read.csv("final_cleaned_lyrics_with_topics.csv")
head(ms_topic)
```

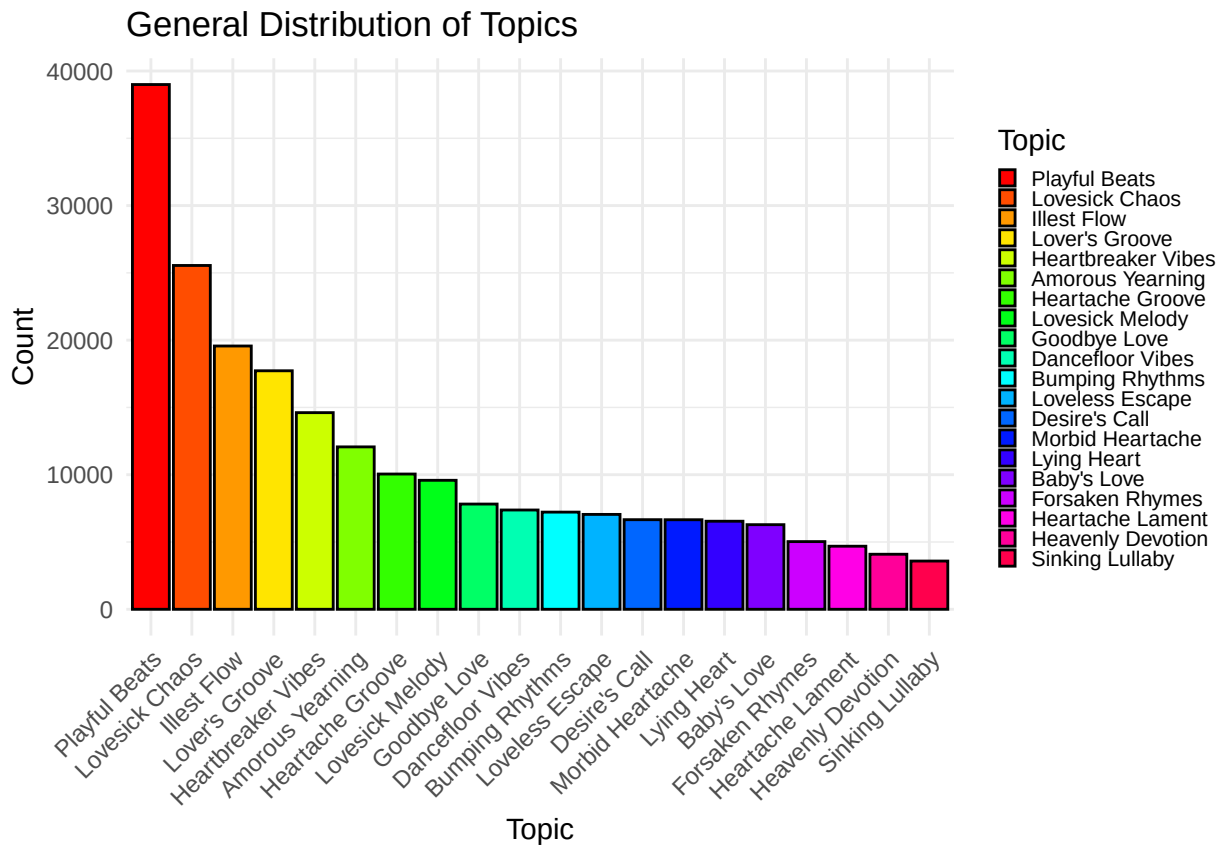
```
##      user_id disorder song_id              artist
## 1 4353e884c1  anxiety       0 Echo & the Bunnymen
## 2 4353e884c1  anxiety       1        Turin Brakes
## 3 4353e884c1  anxiety       2           Radiohead
## 4 4353e884c1  anxiety       3              Peace
## 5 4353e884c1  anxiety       4            Karen O
## 6 4353e884c1  anxiety       5       Joni Mitchell
##                                  title
## 1                    The Killing Moon
## 2                            Red Moon
## 3                    Sail To The Moon
## 4                      Under the Moon
## 5 The Moon Song - Studio Version Duet
## 6                   Moon At The Window
##
```

```
## 1
## 2                                                    x2 sometimes letting go easier dead friend ca
## 3
## 4
## 5
## 6 take cheerful resignation heart humility thats take cheerful person told nobody harder could nobody
##   topic       topic_name
## 1     8 Morbid Heartache
## 2     6  Lovesick Melody
## 3    19  Sinking Lullaby
## 4     1   Lovesick Chaos
## 5     6  Lovesick Melody
## 6     1   Lovesick Chaos
##
## 1              scream commotion morbid slither dyin forsaken gruesome loveless sadistic bleed wretche
## 2 fallin raindrop runaway moonlit yearning lovesick leavin kaleidoscope loveless gloom wrld hazy twir
## 3                                                    rhyming raindrop oar undertow reeling poem qu
## 4                commotion reeling lovesick heartbreaker runaway heartache leavin brokenhearted fall
## 5 fallin raindrop runaway moonlit yearning lovesick leavin kaleidoscope loveless gloom wrld hazy twir
## 6                commotion reeling lovesick heartbreaker runaway heartache leavin brokenhearted fall
```

```
disorder_colors = c(
  "anxiety" = rgb(1, 0, 0),       # Red
  "depression" = rgb(0, 0, 1),    # Blue
  "ptsd" = rgb(0, 1, 0),          # Green
  "borderline" = rgb(1, 0.5, 0),  # Orange
  "panic" = rgb(0.5, 0, 0.5),     # Purple
  "bipolar" = rgb(1, 0.75, 0.8)   # Pink
)
```

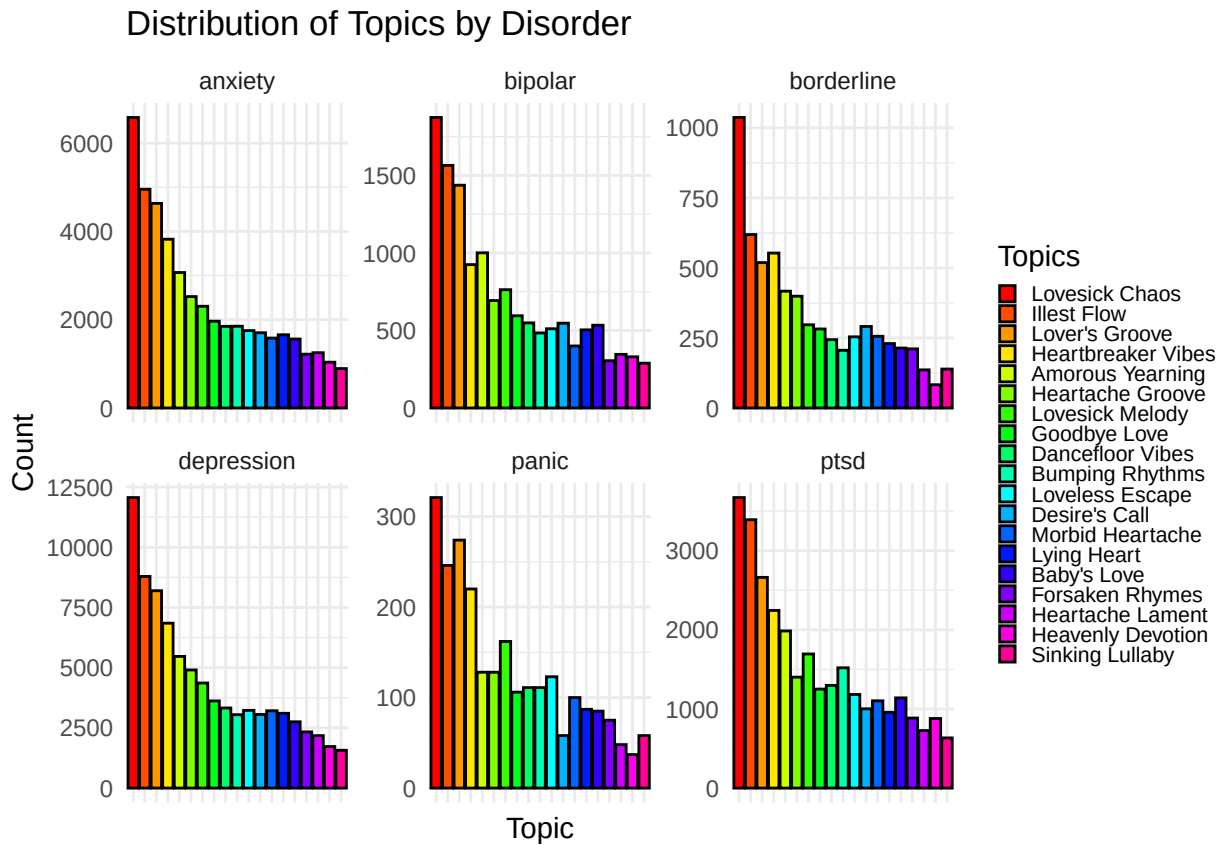
```
# Summarize the count for each topic
topic_count = ms_topic %>%
  count(topic_name) %>%
  arrange(desc(n))

# Reorder the topic_name based on the count
ms_topic$topic_name = factor(ms_topic$topic_name, levels = topic_count$topic_name)

# General Distribution of Topics
plt = ggplot(ms_topic, aes(x = factor(topic_name))) +
  geom_bar(aes(fill = factor(topic_name)), color = "black") +
  scale_fill_manual(values = rainbow(20)) +  # Rainbow colors for topics
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = "Topic", y = "Count", title = "General Distribution of Topics") +
  guides(
    fill = guide_legend(
      title = "Topic",           # Title of the legend
      title.position = "top",    # Position of the title
      label.position = "right",  # Position of the labels
      label.theme = element_text(size = 8),  # Label size
      keyheight = unit(0.5, "lines"),  # Size of the legend keys
      keywidth = unit(0.5, "lines")   # Width of the legend keys
    )
  )
```

```
# Save the plot
ggsave("Images/Songs_Topic/general_distribution_plot.png", plot = plt, width = 10, height = 6)

# Display the plot
plt
```



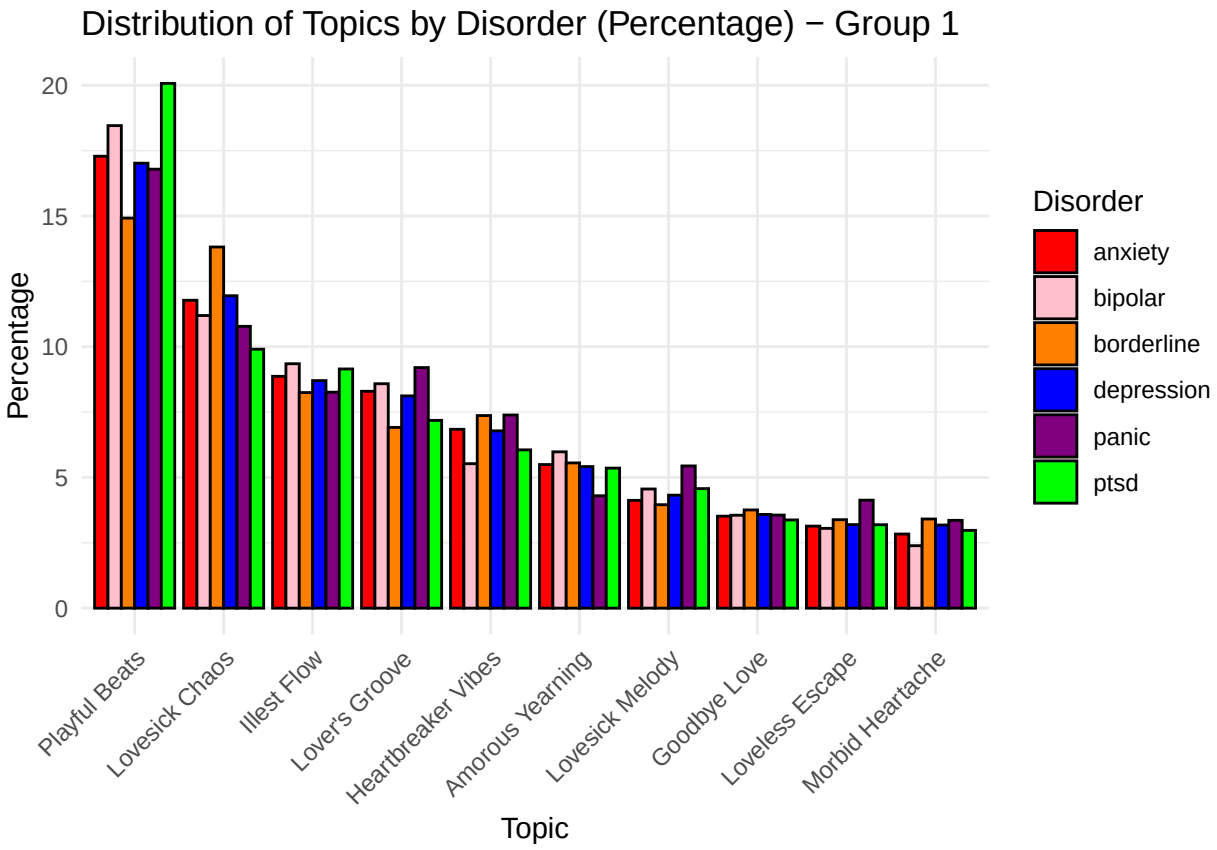

```
# Filter out "Playful Beats" from the dataset
ms_topic_filtered = ms_topic %>%
  filter(topic_name != "Playful Beats")

# Combined Distribution of Topics by Disorder using facets (after filtering)
plt = ggplot(ms_topic_filtered, aes(x = factor(topic_name), fill = topic_name)) +
  geom_bar(position = "dodge", color = "black") +
  scale_fill_manual(values = rainbow(20)) +
  labs(x = "Topic", y = "Count", title = "Distribution of Topics by Disorder") +
  theme_minimal() +
  theme(axis.text.x = element_blank()) +
  facet_wrap(~ disorder, scales = "free_y") +  # Create facets by disorder
  guides(
    fill = guide_legend(
      title = "Topics",           # Title of the legend
      title.position = "top",       # Position of the title
      label.position = "right",     # Position of the labels
      label.theme = element_text(size = 8),  # Label size
      keyheight = unit(0.5, "lines"),  # Size of the legend keys
```

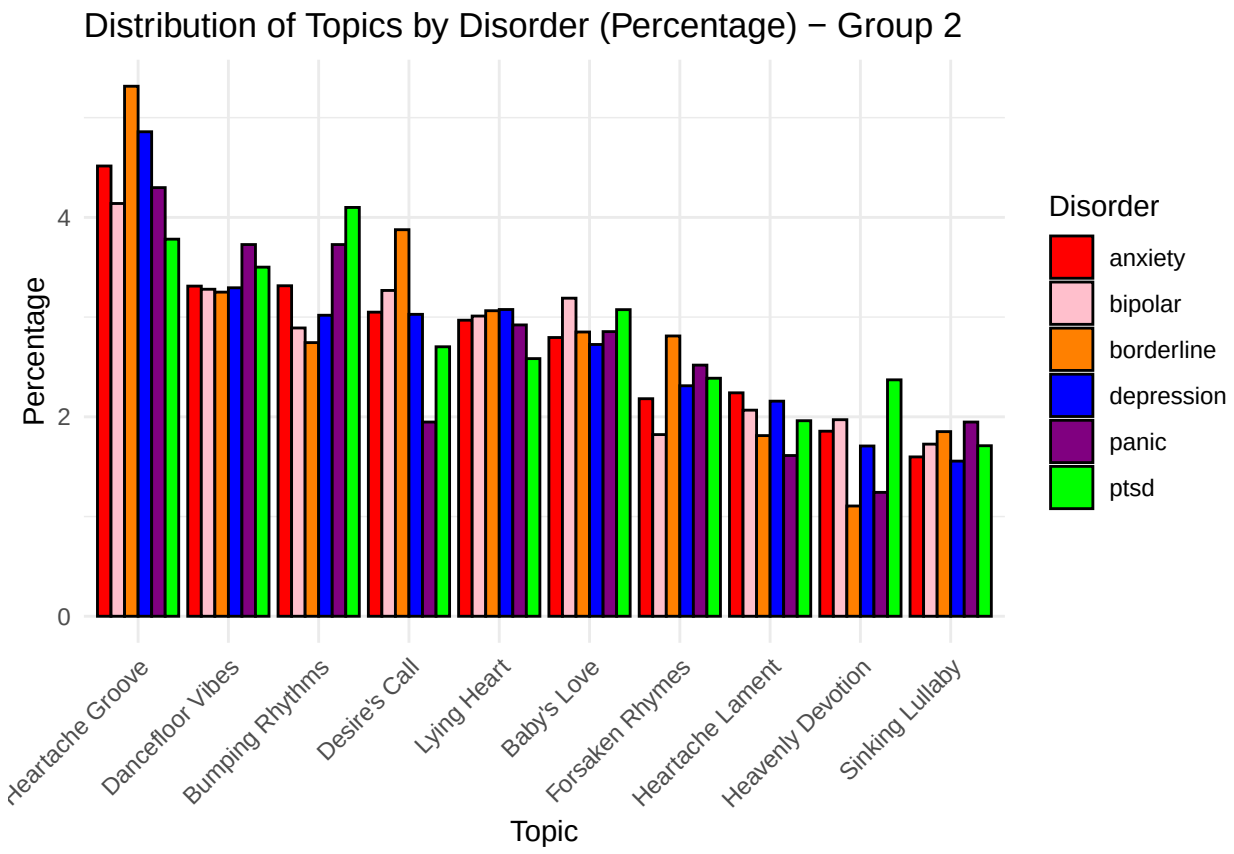
```
      keywidth = unit(0.5, "lines")    # Width of the legend keys
    )
  )

ggsave("Images/Songs_Topic/disorders_distribution_plot.png", plot = plt, width = 10, height = 6)

plt
```




```
# Summarize the counts by disorder and topic, then calculate the percentage
topic_distribution = ms_topic %>%
  group_by(disorder, topic, topic_name) %>%
  summarise(Count = n(), .groups = 'drop') %>%
  group_by(disorder) %>%
  mutate(Total = sum(Count)) %>%
  ungroup() %>%
  mutate(Percentage = (Count / Total) * 100,
         Topic_Group = ifelse(topic <= 9, 1, 2))

# For loop to create and save plots for each group
for (group_num in 1:2) {
  # Filter data based on Topic_Group
  topic_group_data = topic_distribution %>% filter(Topic_Group == group_num)

  # Create plot
  p = ggplot(topic_group_data, aes(x = factor(topic_name), y = Percentage, fill = disorder)) +
```

```
    scale_fill_manual(values = disorder_colors) +
    geom_bar(stat = "identity", position = "dodge", color = "black") +
    theme_minimal() +
    labs(
      title = paste("Distribution of Topics by Disorder (Percentage) - Group", group_num),
      x = "Topic",
      y = "Percentage",
      fill = "Disorder"
    ) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
  print(p)

  # Save the plot
  ggsave(paste0("Images/Songs_Topic/topic_distribution_group_", group_num, ".png"), plot = p, width = 8
}
```






```
rm(list = ls())
```

Twitts Analysis

Now I will repeat pretty much the same steps with some small adjustments

Sentiment analysis

```
#tw_data = read.csv("cleaned_tweets_dataset.csv")
```

```
#head(tw_data)
```

```
# library(syuzhet)
# sentiment = get_nrc_sentiment(tw_data$cleaned_text) # this took 12hrs
```

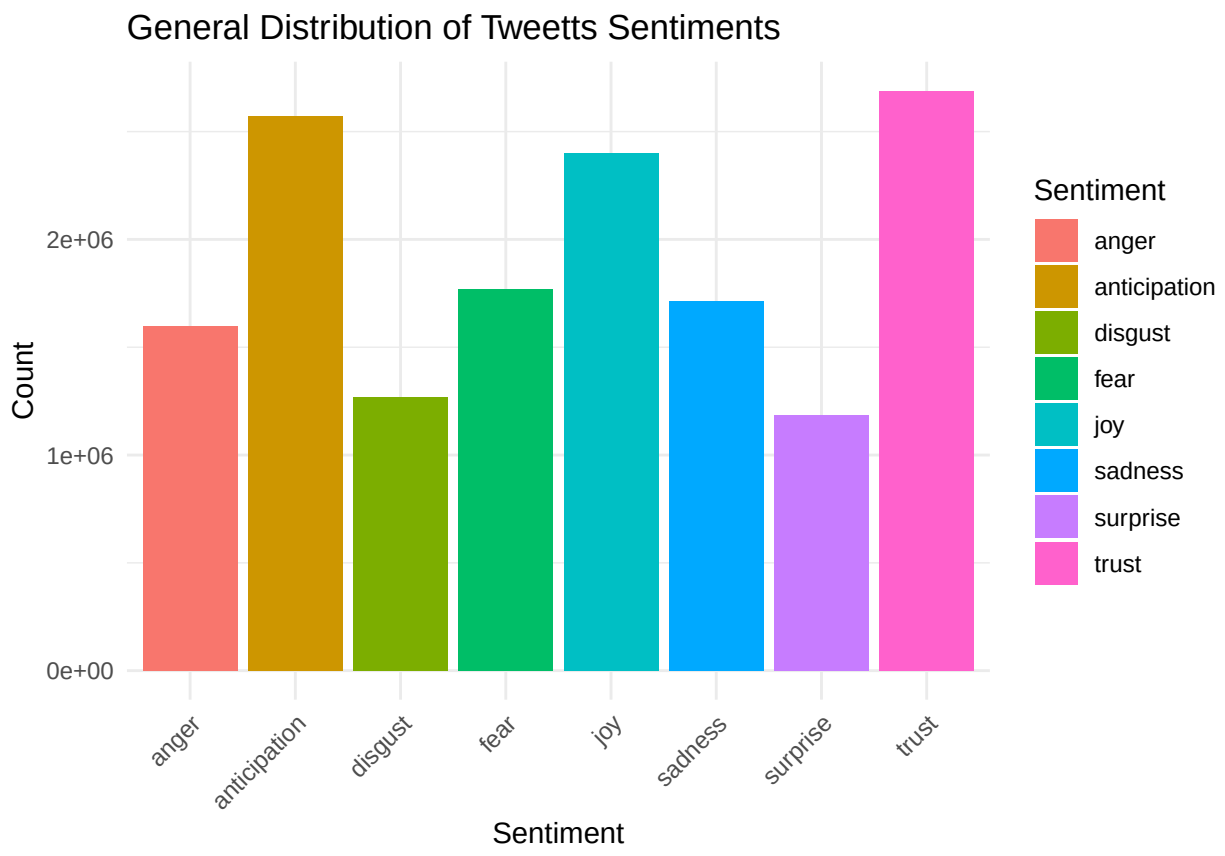
```
#head(sentiment)
```

```
#tw_sent_data = cbind(tw_data, sentiment)
#write.csv(tw_sent_data, "cleaned_tweets_with_sentiment.csv", row.names = FALSE)
```

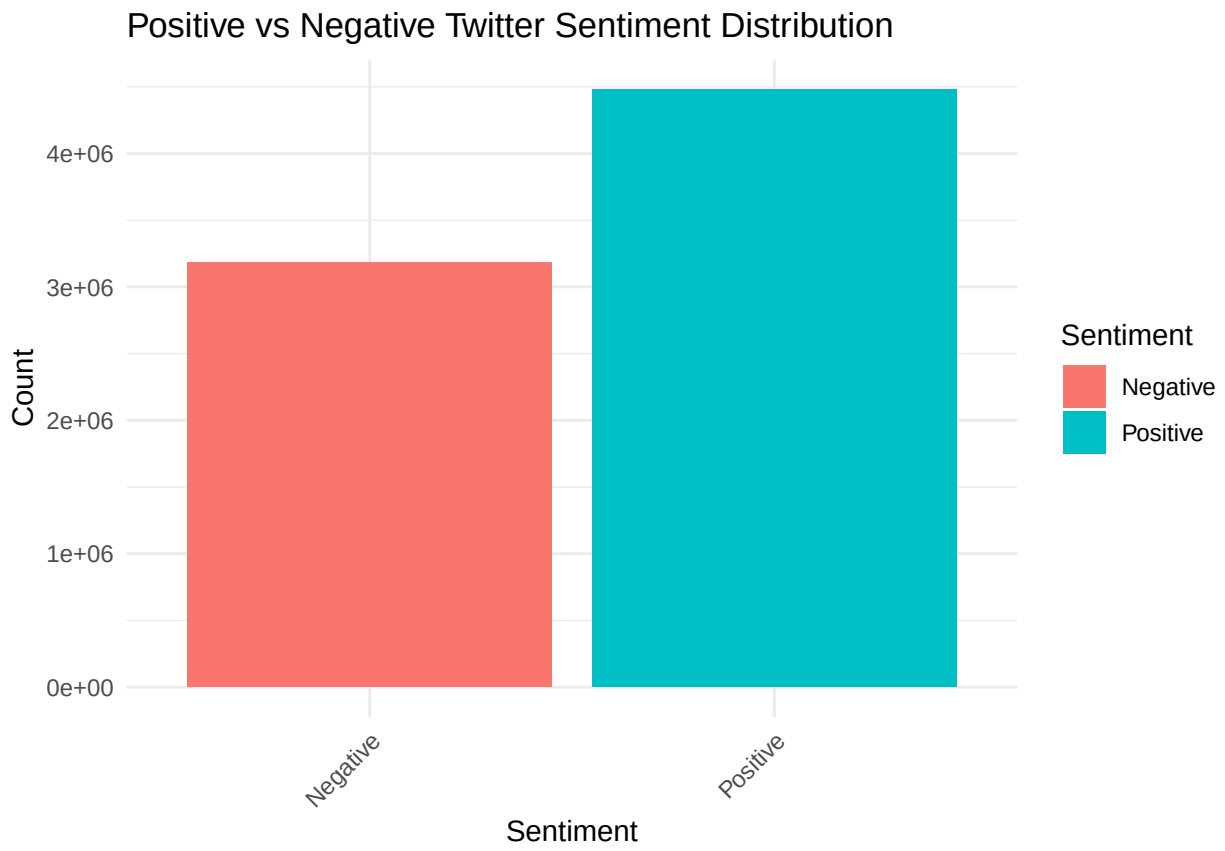
```r
tw_sent_data = read.csv("cleaned_tweets_with_sentiment.csv")
head(tw_sent_data)
```

```
##      user_id disorder
## 1 3f21058fc8  anxiety
## 2 3f21058fc8  anxiety
## 3 3f21058fc8  anxiety
## 4 3f21058fc8  anxiety
## 5 3f21058fc8  anxiety
## 6 3f21058fc8  anxiety
##
## 1                          ternangis baca text amp dengar call reply dieorang im glad followed nasihat
## 2 learn smthg valuable today honest condition clientsall replied thoughtful supportive word considera
## 3                                                                                                ohhl
## 4
## 5                                                 couldnt sleep figured need stimulus win anxiety new
## 6
##   anger anticipation disgust fear joy sadness surprise trust negative positive
## 1     0            2       0    0   3       0        0     3        0        3
## 2     1            0       1    1   1       1        0     4        0        7
## 3     0            1       0    0   2       0        0     1        0        3
## 4     0            0       0    0   0       0        0     0        0        0
## 5     3            3       1    2   2       2        1     3        3        3
## 6     0            1       0    0   2       0        0     2        0        2
```

```r
# Summarize the general distribution of sentiments
sentiment_totals = colSums(tw_sent_data[, c("anger", "anticipation", "disgust", "fear", "joy",
                                            "sadness", "surprise", "trust")])

# Create a data frame for plotting
sentiment_df = data.frame(
  Sentiment = names(sentiment_totals),
  Count = as.numeric(sentiment_totals)
)

# Plot for general distribution of sentiments
ggplot(sentiment_df, aes(x = Sentiment, y = Count, fill = Sentiment)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(
    title = "General Distribution of Tweetts Sentiments",
    x = "Sentiment",
    y = "Count"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



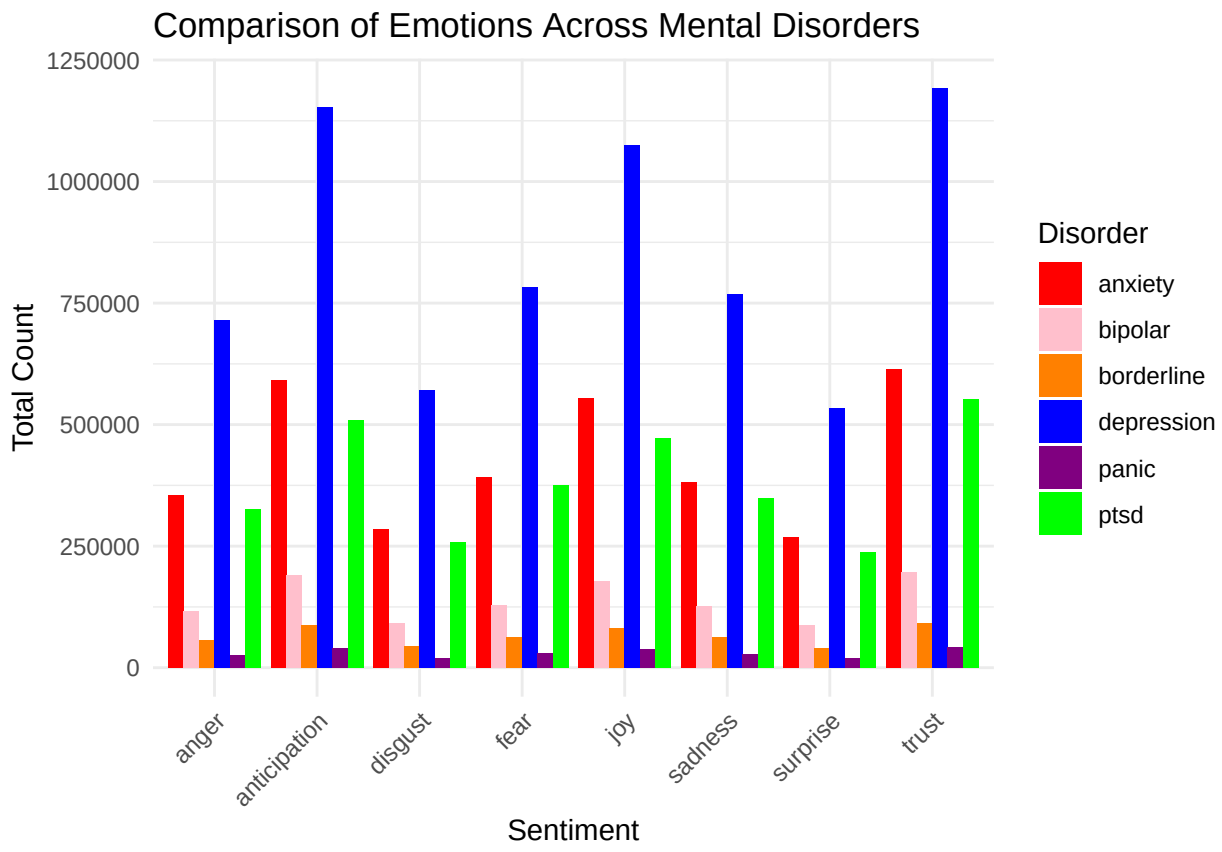

```r
# Summarize the positive/negative sentiments
pos_neg_totals = colSums(tw_sent_data[, c("positive", "negative")])

# Create a data frame for positive/negative plotting
pos_neg_df = data.frame(
  Sentiment = c("Positive", "Negative"),
  Count = as.numeric(pos_neg_totals)
)

# Plot for positive/negative sentiment distribution
ggplot(pos_neg_df, aes(x = Sentiment, y = Count, fill = Sentiment)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(
    title = "Positive vs Negative Twitter Sentiment Distribution",
    x = "Sentiment",
    y = "Count"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
disorder_colors = c(
  "anxiety" = rgb(1, 0, 0),        # Red
  "depression" = rgb(0, 0, 1),     # Blue
  "ptsd" = rgb(0, 1, 0),           # Green
  "borderline" = rgb(1, 0.5, 0),   # Orange
  "panic" = rgb(0.5, 0, 0.5),      # Purple
  "bipolar" = rgb(1, 0.75, 0.8)    # Pink
)
```

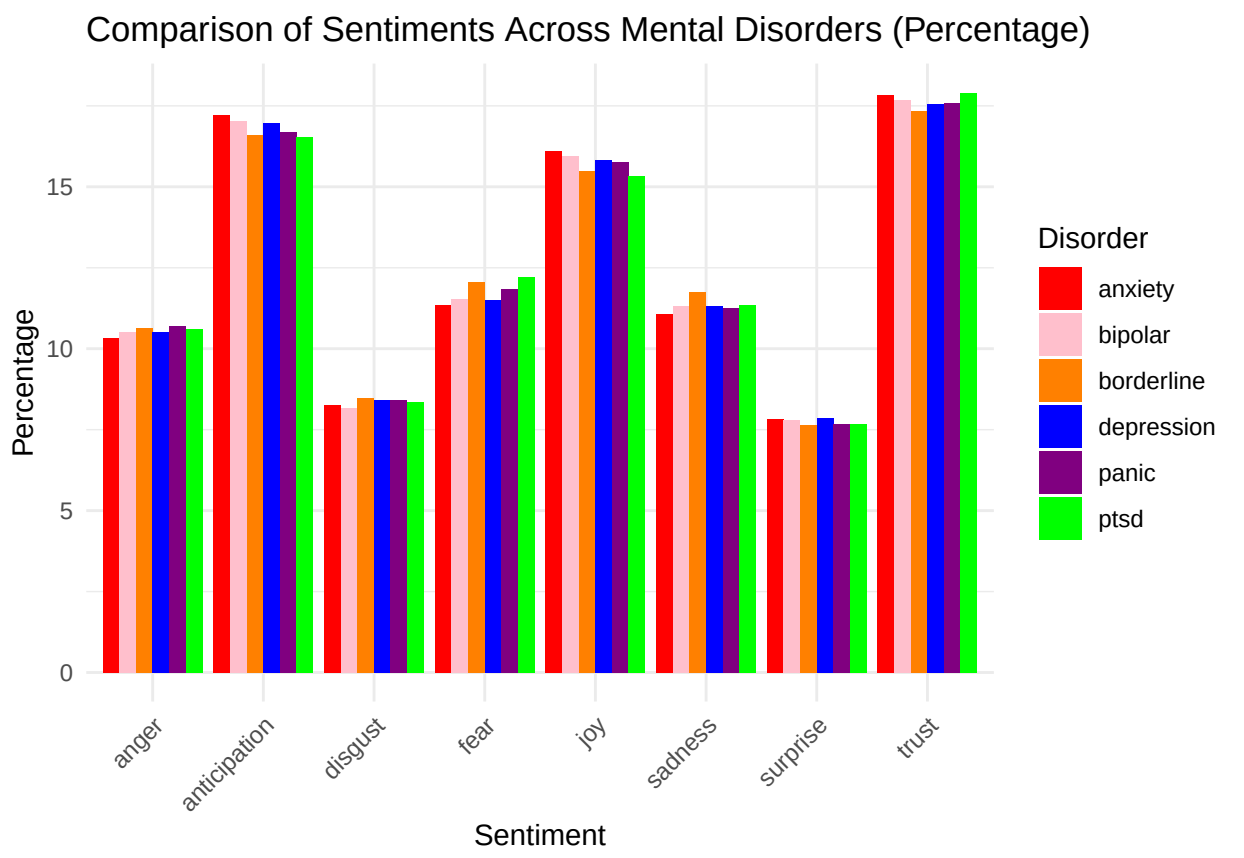
```
# Select relevant columns (disorder and sentiment scores)
relevant_columns = c("disorder", "anger", "anticipation", "disgust", "fear",
                     "joy", "sadness", "surprise", "trust")

# Aggregate sentiment scores by disorder
disorder_sentiments = tw_sent_data[, relevant_columns] %>%
  group_by(disorder) %>%
  summarise(across(everything(), sum, na.rm = TRUE)) %>%
  pivot_longer(cols = -disorder, names_to = "Sentiment", values_to = "Count")

# Plot emotions comparison between disorders
ggplot(disorder_sentiments, aes(x = Sentiment, y = Count, fill = disorder)) +
  scale_fill_manual(values = disorder_colors) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(
    title = "Comparison of Emotions Across Mental Disorders",
```

```
    x = "Sentiment",
    y = "Total Count",
    fill = "Disorder"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



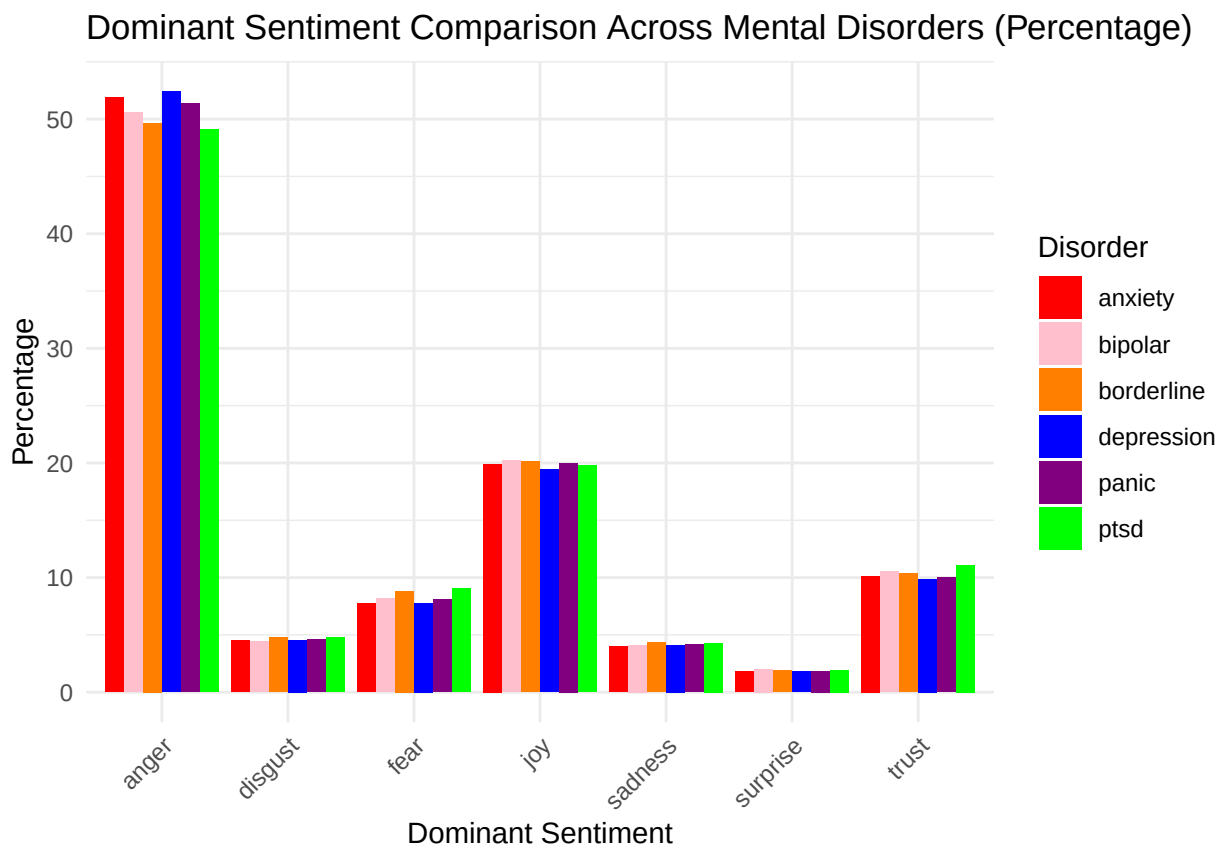

```
# Select relevant columns (disorder and sentiment scores)
relevant_columns = c("disorder", "anger", "anticipation", "disgust", "fear",
                     "joy", "sadness", "surprise", "trust")

# Aggregate sentiment scores by disorder
disorder_sentiments = tw_sent_data[, relevant_columns] %>%
  group_by(disorder) %>%
  summarise(across(everything(), sum, na.rm = TRUE)) %>%
  mutate(Total = rowSums(across(where(is.numeric)))) %>%
  pivot_longer(cols = -c(disorder, Total), names_to = "Sentiment", values_to = "Count") %>%
  mutate(Percentage = (Count / Total) * 100)

# Plot emotions comparison as percentages
ggplot(disorder_sentiments, aes(x = Sentiment, y = Percentage, fill = disorder)) +
  scale_fill_manual(values = disorder_colors) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(
    title = "Comparison of Sentiments Across Mental Disorders (Percentage)",
```

```
    x = "Sentiment",
    y = "Percentage",
    fill = "Disorder"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Select relevant columns (disorder and sentiment scores)
relevant_columns = c("disorder", "anger", "disgust", "fear",
                     "joy", "sadness", "surprise", "trust")

# Identify the highest sentiment for each row and assign +1
disorder_sentiments = tw_sent_data[, relevant_columns] %>%
  mutate(dominant_sentiment = apply(.[, -1], 1, function(x) names(x)[which.max(x)])) %>%
  group_by(disorder, dominant_sentiment) %>%
  summarise(count = n(), .groups = "drop") %>%
  # Sum all counts per disorder
  group_by(disorder) %>%
  mutate(total_count = sum(count)) %>%
  # Calculate percentage for each sentiment
  mutate(percentage = (count / total_count) * 100)

# Plot the results with custom colors
ggplot(disorder_sentiments, aes(x = dominant_sentiment, y = percentage, fill = disorder)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = disorder_colors) +  # Use custom colors
```

```
  theme_minimal() +
  labs(
    title = "Dominant Sentiment Comparison Across Mental Disorders (Percentage)",
    x = "Dominant Sentiment",
    y = "Percentage",
    fill = "Disorder"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
rm(list = ls())
```

Topic Analysis

As for the lyrics, the topic modelling for tweets has been done in Python using the Top2Vec library. Here we have the dataset that includes the topic, the list of more common words for each topic and the topic name that I assigned based on the more common words. The only difference in this case is that (due to machine limitations) the analysis focus only on a random sample of tweets selected from those of each user, in proportion with the total number of tweets he wrote.

```
tw_topic = read.csv("final_twitter_with_topics.csv")
head(tw_topic)
```

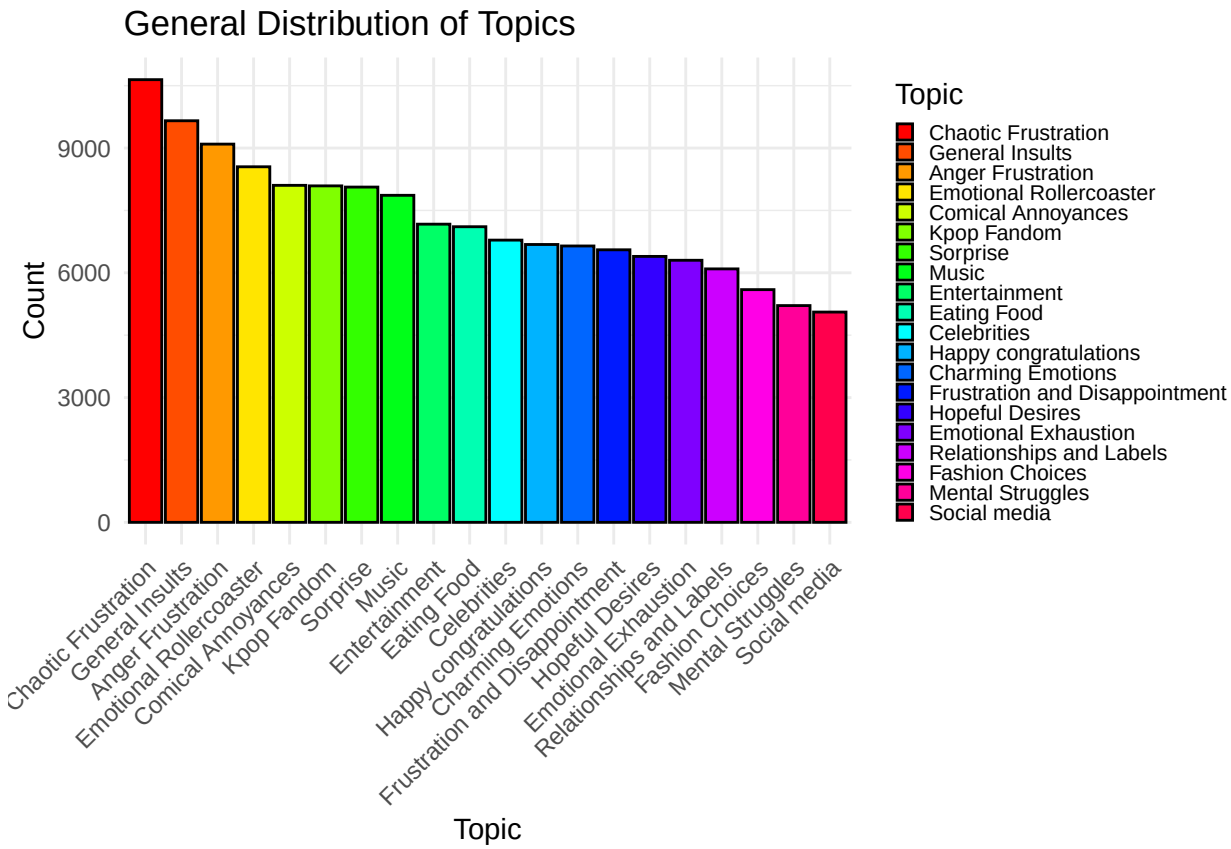
```
##      user_id disorder
```

```
## 1 0001e573dc  anxiety
## 2 0001e573dc  anxiety
## 3 0001e573dc  anxiety
## 4 0001e573dc  anxiety
## 5 0001e573dc  anxiety
## 6 0001e573dc  anxiety
##                                                                                                                    cleaned
## 1                                                                                        love important announcement twi
## 2                                                                   right mind decides business three year ffs life want living
## 3                                                                                         good good luck english lit paper
## 4                                                                                                          really light bl
## 5 okay least know dont labour government wake morning cry harrys new album absolute banger mention pl
## 6                                                                                            hope x btw loved video yeste
##   topic                    topic_name
## 1    19                  Social media
## 2    13 Frustration and Disappointment
## 3    11          Happy congratulations
## 4    17                Fashion Choices
## 5     7                         Music
## 6    11          Happy congratulations
##
## 1                        tweeting twitter tweeted tweet unfollow retweet hashtag unfollowed insta ins
## 2                                                                                       fuck crap barely
## 3 happy merry happiest congratulation wishing congrats happier friday glad yay celebrating celebrate
## 4                                                                                             wig haircut lip make
## 5                                                                     song rap spotify tune hiphop music
## 6 happy merry happiest congratulation wishing congrats happier friday glad yay celebrating celebrate
```

```r
disorder_colors = c(
  "anxiety" = rgb(1, 0, 0),       # Red
  "depression" = rgb(0, 0, 1),    # Blue
  "ptsd" = rgb(0, 1, 0),          # Green
  "borderline" = rgb(1, 0.5, 0),  # Orange
  "panic" = rgb(0.5, 0, 0.5),     # Purple
  "bipolar" = rgb(1, 0.75, 0.8)   # Pink
)
```

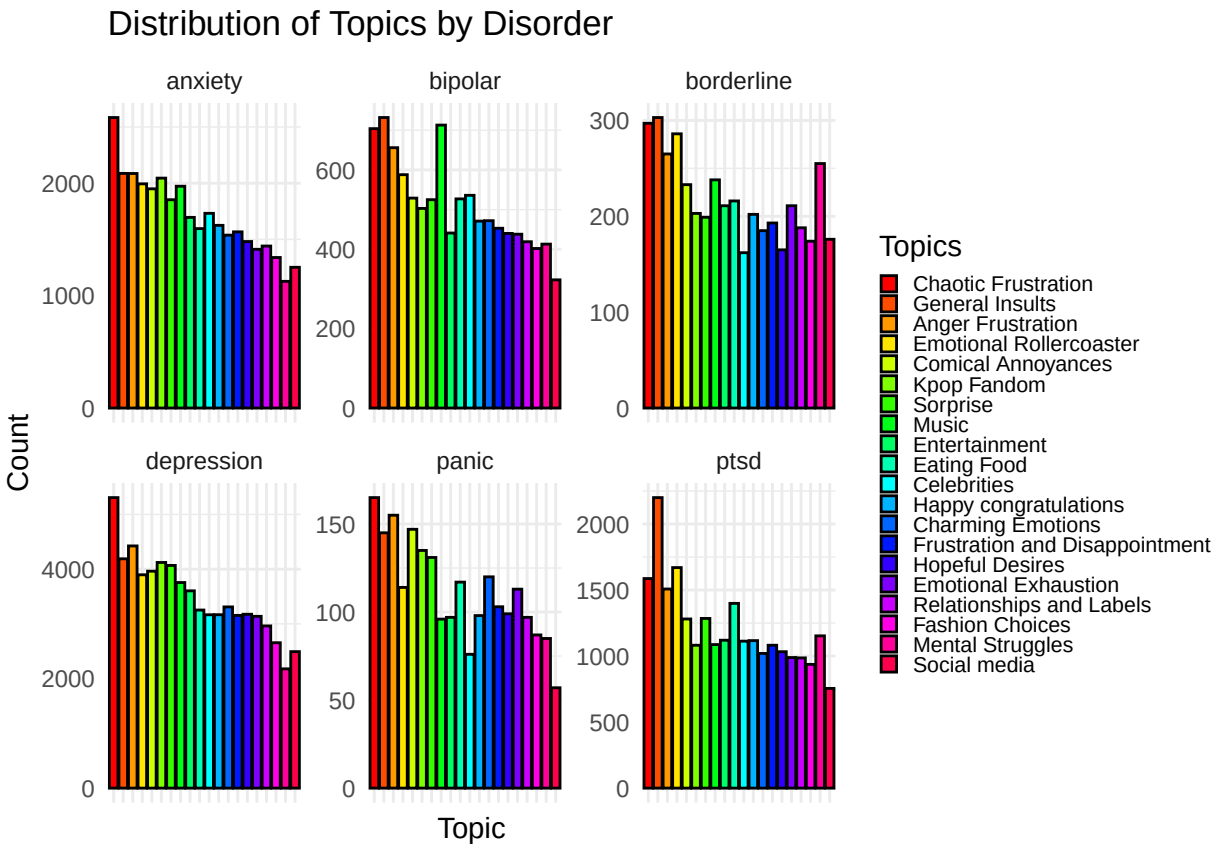
```r
# Summarize the count for each topic
topic_count = tw_topic %>%
  count(topic_name) %>%
  arrange(desc(n))

# Reorder the topic_name based on the count
tw_topic$topic_name = factor(tw_topic$topic_name, levels = topic_count$topic_name)

# General Distribution of Topics
plt = ggplot(tw_topic, aes(x = factor(topic_name))) +
  geom_bar(aes(fill = factor(topic_name)), color = "black") +
  scale_fill_manual(values = rainbow(20)) +  # Rainbow colors for topics
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = "Topic", y = "Count", title = "General Distribution of Topics") +
  guides(
    fill = guide_legend(
      title = "Topic",           # Title of the legend
```

```
      title.position = "top",     # Position of the title
      label.position = "right",  # Position of the labels
      label.theme = element_text(size = 8),  # Label size
      keyheight = unit(0.5, "lines"),  # Size of the legend keys
      keywidth = unit(0.5, "lines")   # Width of the legend keys
    )
  )

# Save the plot
ggsave("Images/Twitter_Topic/general_distribution_plot.png", plot = plt, width = 10, height = 6)

# Display the plot
plt
```



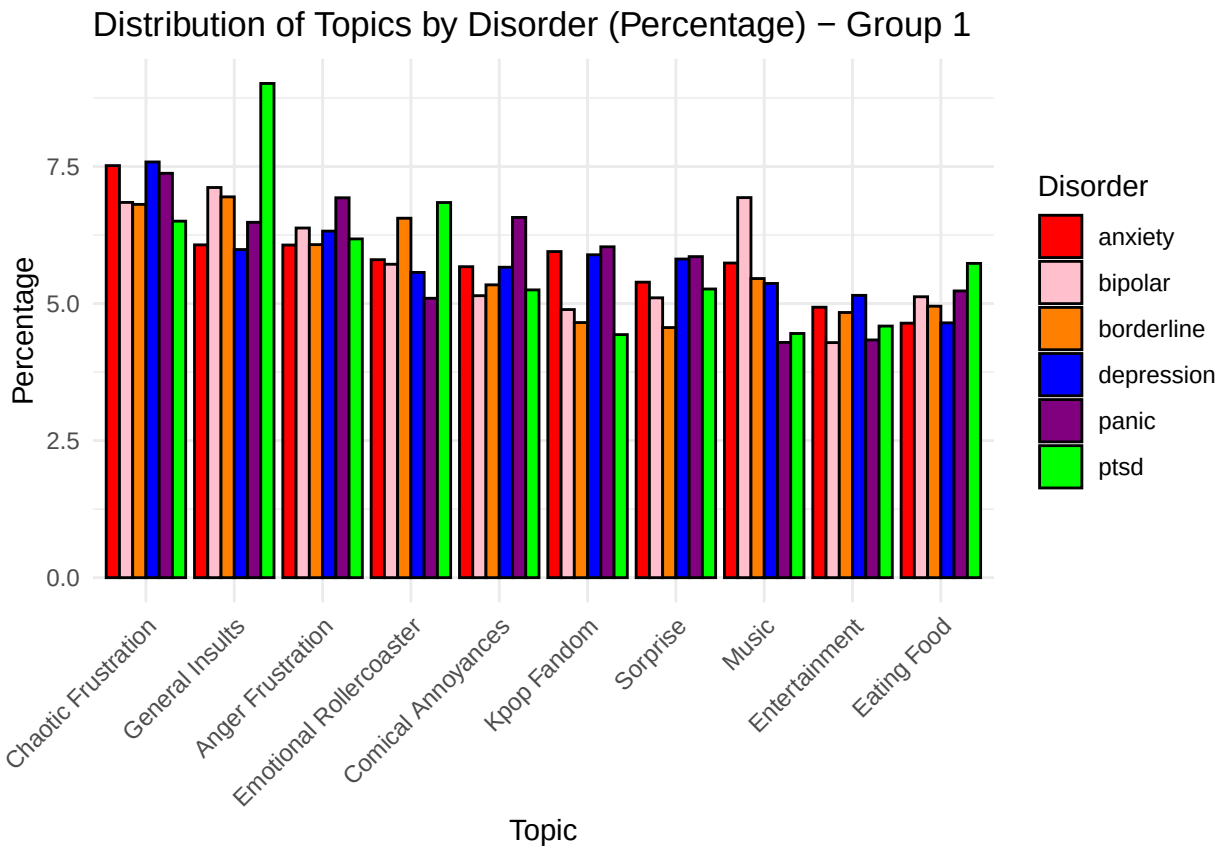

```
# Filter out "Playful Beats" from the dataset
tw_topic_filtered = tw_topic %>%
  filter(topic_name != "Playful Beats")

# Combined Distribution of Topics by Disorder using facets (after filtering)
plt = ggplot(tw_topic_filtered, aes(x = factor(topic_name), fill = topic_name)) +
  geom_bar(position = "dodge", color = "black") +
  scale_fill_manual(values = rainbow(20)) +
  labs(x = "Topic", y = "Count", title = "Distribution of Topics by Disorder") +
  theme_minimal() +
  theme(axis.text.x = element_blank()) +
```

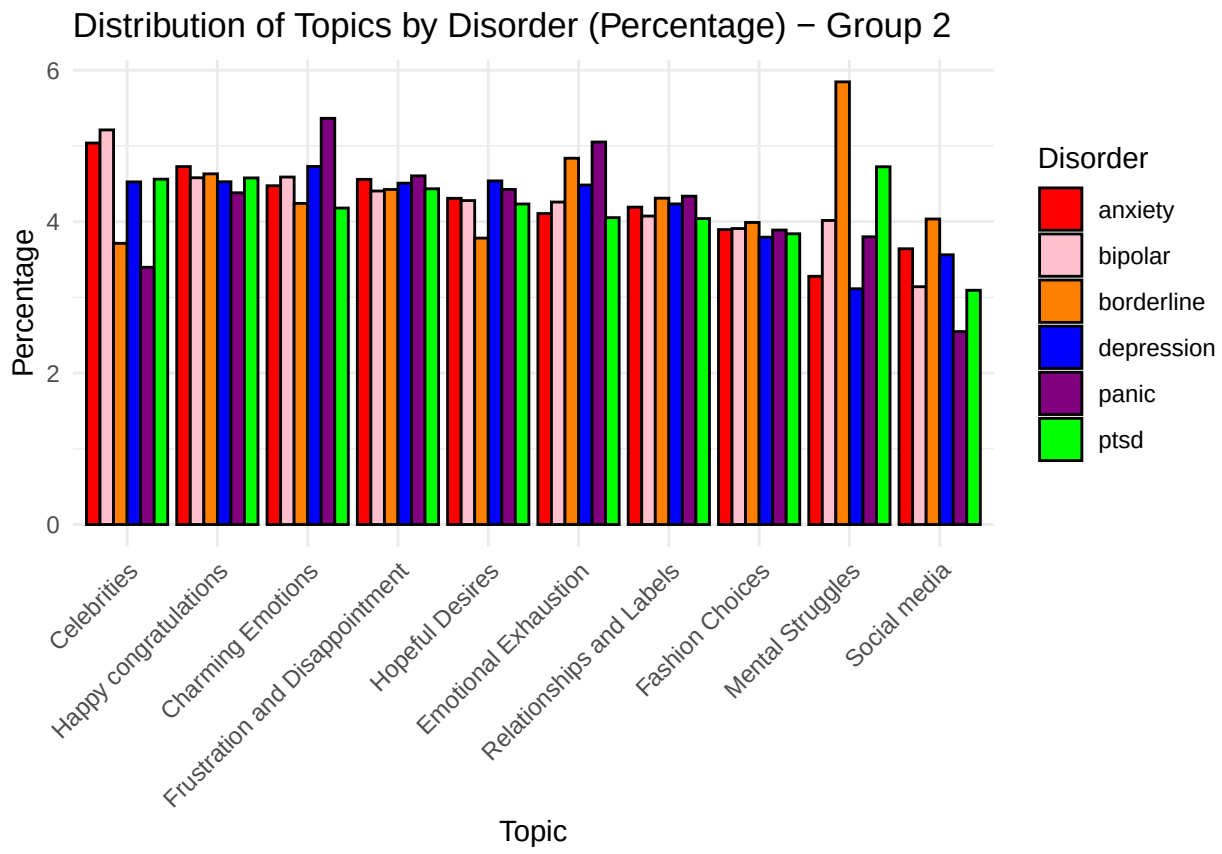
```r
  facet_wrap(~ disorder, scales = "free_y") +  # Create facets by disorder
  guides(
    fill = guide_legend(
      title = "Topics",           # Title of the legend
      title.position = "top",       # Position of the title
      label.position = "right",     # Position of the labels
      label.theme = element_text(size = 8),  # Label size
      keyheight = unit(0.5, "lines"),  # Size of the legend keys
      keywidth = unit(0.5, "lines")    # Width of the legend keys
    )
  )

ggsave("Images/Twitter_topic/disorders_distribution_plot.png", plot = plt, width = 10, height = 6)

plt
```




```r
# Summarize the counts by disorder and topic, then calculate the percentage
topic_distribution = tw_topic %>%
  group_by(disorder, topic, topic_name) %>%
  summarise(Count = n(), .groups = 'drop') %>%
  group_by(disorder) %>%
  mutate(Total = sum(Count)) %>%
  ungroup() %>%
  mutate(Percentage = (Count / Total) * 100,
         Topic_Group = ifelse(topic <= 9, 1, 2))
```

```
# For loop to create and save plots for each group
for (group_num in 1:2) {
  # Filter data based on Topic_Group
  topic_group_data = topic_distribution %>% filter(Topic_Group == group_num)

  # Create plot
  p = ggplot(topic_group_data, aes(x = factor(topic_name), y = Percentage, fill = disorder)) +
    scale_fill_manual(values = disorder_colors) +
    geom_bar(stat = "identity", position = "dodge", color = "black") +
    theme_minimal() +
    labs(
      title = paste("Distribution of Topics by Disorder (Percentage) - Group", group_num),
      x = "Topic",
      y = "Percentage",
      fill = "Disorder"
    ) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
  print(p)

  # Save the plot
  ggsave(paste0("Images/Twitter_topic/topic_distribution_group_", group_num, ".png"), plot = p, width =
}
```





Final Analysis: is the mood of these persons influenced by the music they listen to?

Now I will explore the significance of the disorders, sentiments of the songs and their topics on the tweets sentiment.

```
ms_sent_data = read.csv("final_cleaned_lyrics_with_sentiment.csv")
ms_topic = read.csv("final_cleaned_lyrics_with_topics.csv")
tw_sent_data = read.csv("cleaned_tweets_with_sentiment.csv")
```

Firstly I will take for each person the more common sentiment among all the songs that he listen to, choosed between anger, fear, joy, sadness and surprise

```
dominant_sentiment_ms_data = ms_sent_data %>%
  group_by(user_id, disorder) %>%
  summarise(
    anger = sum(anger),
    fear = sum(fear),
    joy = sum(joy),
    sadness = sum(sadness),
    .groups = 'drop'
  )
```

```
dominant_sentiment_ms_data = dominant_sentiment_ms_data %>%
  rowwise() %>%
  mutate(dominant_song_sentiment = names(cur_data())[which.max(c(anger, fear, joy, sadness)) + 2]) %>%
  select(user_id, disorder, dominant_song_sentiment)
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `dominant_song_sentiment = names(cur_data())[which.max(c(anger,
##   fear, joy, sadness)) + 2]`.
## i In row 1.
## Caused by warning:
## ! `cur_data()` was deprecated in dplyr 1.1.0.
## i Please use `pick()` instead.
```

```
dominant_sentiment_ms_data
```

```
## # A tibble: 4,397 x 3
## # Rowwise:
##    user_id    disorder   dominant_song_sentiment
##    <chr>      <chr>      <chr>
##  1 0001e573dc anxiety    sadness
##  2 00021bdb1d anxiety    sadness
##  3 00021bdb1d depression sadness
##  4 0003c5dea3 panic      fear
##  5 00180e338b ptsd       fear
##  6 001fec300a ptsd       fear
##  7 0028a9ef7c anxiety    sadness
##  8 004fcce5bd depression fear
##  9 0052e8ca6f depression joy
## 10 0067dd24ff depression joy
## # i 4,387 more rows
```

Than I will do the same thing with the songs_topic

```
dominant_ms_topics = ms_topic %>%
  group_by(user_id, disorder, topic_name) %>%
  summarise(topic_count = n(), .groups = 'drop') %>%
  group_by(user_id, disorder) %>%
  slice_max(topic_count, n = 1) %>%
  select(user_id, disorder, dominant_song_topic = topic_name)

dominant_ms_topics
```

```
## # A tibble: 5,813 x 3
## # Groups:   user_id, disorder [4,397]
##    user_id    disorder   dominant_song_topic
##    <chr>      <chr>      <chr>
##  1 0001e573dc anxiety    Playful Beats
##  2 00021bdb1d anxiety    Heartache Groove
##  3 00021bdb1d depression Heartache Groove
##  4 0003c5dea3 panic      Lovesick Chaos
##  5 0003c5dea3 panic      Lovesick Melody
##  6 00180e338b ptsd       Playful Beats
```

```
##  7 001fec300a ptsd       Playful Beats
##  8 0028a9ef7c anxiety    Goodbye Love
##  9 004fcce5bd depression Lovesick Chaos
## 10 004fcce5bd depression Lying Heart
## # i 5,803 more rows
```

And finally I'm taking the general positive/negative sentiment of the tweets of each person and use the highest value as the reference for the general mood of a person

```
dominant_sentiment_tw_data = tw_sent_data %>%
  group_by(user_id, disorder) %>%
  summarise(
    positive = sum(positive),
    negative = sum(negative),
    .groups = 'drop'
  )

dominant_sentiment_tw_data = dominant_sentiment_tw_data %>%
  rowwise() %>%
  mutate(dominant_twit_sentiment = names(cur_data())[which.max(c(positive, negative)) + 2]) %>%
  select(user_id, disorder, dominant_twit_sentiment)

dominant_sentiment_tw_data
```

```
## # A tibble: 4,407 x 3
## # Rowwise:
##    user_id    disorder   dominant_twit_sentiment
##    <chr>      <chr>      <chr>
##  1 0001e573dc anxiety    positive
##  2 00021bdb1d anxiety    positive
##  3 00021bdb1d depression positive
##  4 0003c5dea3 panic      positive
##  5 00180e338b ptsd       negative
##  6 001fec300a ptsd       positive
##  7 0028a9ef7c anxiety    positive
##  8 004fcce5bd depression positive
##  9 0052e8ca6f depression positive
## 10 0067dd24ff depression positive
## # i 4,397 more rows
```

```
# Joining everything in one dataset
big_final = dominant_sentiment_ms_data %>%
  inner_join(dominant_ms_topics, by = c("user_id", "disorder")) %>%
  inner_join(dominant_sentiment_tw_data, by = c("user_id", "disorder"))

big_final$dominant_twit_sentiment = as.factor(big_final$dominant_twit_sentiment)
head(big_final)
```

```
## # A tibble: 6 x 5
## # Rowwise:
##   user_id    disorder   dominant_song_sentiment dominant_song_topic
##   <chr>      <chr>      <chr>                   <chr>
## 1 0001e573dc anxiety    sadness                 Playful Beats
```

```
## 2 00021bdb1d anxiety    sadness                Heartache Groove
## 3 00021bdb1d depression sadness                Heartache Groove
## 4 0003c5dea3 panic      fear                   Lovesick Chaos
## 5 0003c5dea3 panic      fear                   Lovesick Melody
## 6 00180e338b ptsd       fear                   Playful Beats
## # i 1 more variable: dominant_twit_sentiment <fct>
```

Then I fitted some logistic regressions models to see which variables has an influence on the mood of a person

```
big_final$dominant_twit_sentiment = relevel(big_final$dominant_twit_sentiment, ref = "negative")

# Fit logistic regression model
model = glm(dominant_twit_sentiment ~ dominant_song_topic,
            data = big_final, family = binomial)

# Summary of the model
summary(model)
```

```
##
## Call:
## glm(formula = dominant_twit_sentiment ~ dominant_song_topic,
##     family = binomial, data = big_final)
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        1.931150   0.171343  11.271   <2e-16 ***
## dominant_song_topicBaby's Love     1.287726   0.537756   2.395   0.0166 *
## dominant_song_topicBumping Rhythms 0.320142   0.313747   1.020   0.3075
## dominant_song_topicDancefloor Vibes -0.130174  0.306738  -0.424   0.6713
## dominant_song_topicDesire's Call   -0.009337   0.353393  -0.026   0.9789
## dominant_song_topicForsaken Rhymes 0.052148   0.324345   0.161   0.8723
## dominant_song_topicGoodbye Love    0.474976   0.358419   1.325   0.1851
## dominant_song_topicHeartache Groove 0.109506  0.280154   0.391   0.6959
## dominant_song_topicHeartache Lament -0.344185  0.403976  -0.852   0.3942
## dominant_song_topicHeartbreaker Vibes 0.176535  0.229616   0.769   0.4420
## dominant_song_topicHeavenly Devotion 0.659117  0.456645   1.443   0.1489
## dominant_song_topicIllest Flow     -0.238597   0.221569  -1.077   0.2815
## dominant_song_topicLoveless Escape 0.170111   0.308830   0.551   0.5818
## dominant_song_topicLover's Groove  0.036500   0.216549   0.169   0.8661
## dominant_song_topicLovesick Chaos  0.212430   0.197250   1.077   0.2815
## dominant_song_topicLovesick Melody 0.197082   0.258186   0.763   0.4453
## dominant_song_topicLying Heart     0.026595   0.352784   0.075   0.9399
## dominant_song_topicMorbid Heartache -0.055307  0.300091  -0.184   0.8538
## dominant_song_topicPlayful Beats   -0.281097   0.188507  -1.491   0.1359
## dominant_song_topicSinking Lullaby 0.536950   0.496294   1.082   0.2793
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4350.0  on 5810  degrees of freedom
## Residual deviance: 4307.9  on 5791  degrees of freedom
## AIC: 4347.9
```

```
##
## Number of Fisher Scoring iterations: 5
```

Here wee see that one topic apart, songs topic are not statistically significant. However..

```
big_final$dominant_twit_sentiment = relevel(big_final$dominant_twit_sentiment, ref = "positive")

# Fit logistic regression model
model = glm(dominant_twit_sentiment ~ disorder * dominant_song_sentiment ,
            data = big_final, family = binomial)

# Summary of the model
summary(model)
```

```
##
## Call:
## glm(formula = dominant_twit_sentiment ~ disorder * dominant_song_sentiment,
##     family = binomial, data = big_final)
##
## Coefficients:
##                                                 Estimate Std. Error z value
## (Intercept)                                    -1.176321   0.199120  -5.908
## disorderbipolar                                 0.164720   0.391517   0.421
## disorderborderline                             -0.002334   0.605452  -0.004
## disorderdepression                             -0.171752   0.246261  -0.697
## disorderpanic                                  -1.462736   1.054073  -1.388
## disorderptsd                                   -0.452919   0.315361  -1.436
## dominant_song_sentimentfear                    -1.084879   0.263615  -4.115
## dominant_song_sentimentjoy                     -0.967415   0.238967  -4.048
## dominant_song_sentimentsadness                 -0.827021   0.280683  -2.946
## disorderbipolar:dominant_song_sentimentfear    -0.278426   0.565471  -0.492
## disorderborderline:dominant_song_sentimentfear -0.399054   0.867757  -0.460
## disorderdepression:dominant_song_sentimentfear  0.602726   0.319162   1.888
## disorderpanic:dominant_song_sentimentfear       1.731506   1.232762   1.405
## disorderptsd:dominant_song_sentimentfear        0.574053   0.411355   1.396
## disorderbipolar:dominant_song_sentimentjoy     -0.764753   0.517201  -1.479
## disorderborderline:dominant_song_sentimentjoy   0.305520   0.707037   0.432
## disorderdepression:dominant_song_sentimentjoy   0.176341   0.294675   0.598
## disorderpanic:dominant_song_sentimentjoy        0.991512   1.219118   0.813
## disorderptsd:dominant_song_sentimentjoy         0.396511   0.382283   1.037
## disorderbipolar:dominant_song_sentimentsadness  0.006040   0.537667   0.011
## disorderborderline:dominant_song_sentimentsadness -0.191549   0.826731  -0.232
## disorderdepression:dominant_song_sentimentsadness  0.325669   0.340047   0.958
## disorderpanic:dominant_song_sentimentsadness       2.485249   1.268281   1.960
## disorderptsd:dominant_song_sentimentsadness        0.859019   0.418359   2.053
##                                                Pr(>|z|)
## (Intercept)                                    3.47e-09 ***
## disorderbipolar                                 0.67396
## disorderborderline                              0.99692
## disorderdepression                              0.48553
## disorderpanic                                   0.16523
## disorderptsd                                    0.15095
## dominant_song_sentimentfear                    3.87e-05 ***
```

```
## dominant_song_sentimentjoy                          5.16e-05 ***
## dominant_song_sentimentsadness                       0.00321 **
## disorderbipolar:dominant_song_sentimentfear          0.62245
## disorderborderline:dominant_song_sentimentfear       0.64561
## disorderdepression:dominant_song_sentimentfear       0.05896 .
## disorderpanic:dominant_song_sentimentfear            0.16015
## disorderptsd:dominant_song_sentimentfear             0.16286
## disorderbipolar:dominant_song_sentimentjoy           0.13924
## disorderborderline:dominant_song_sentimentjoy        0.66566
## disorderdepression:dominant_song_sentimentjoy        0.54956
## disorderpanic:dominant_song_sentimentjoy             0.41604
## disorderptsd:dominant_song_sentimentjoy              0.29963
## disorderbipolar:dominant_song_sentimentsadness       0.99104
## disorderborderline:dominant_song_sentimentsadness    0.81678
## disorderdepression:dominant_song_sentimentsadness    0.33821
## disorderpanic:dominant_song_sentimentsadness         0.05005 .
## disorderptsd:dominant_song_sentimentsadness          0.04004 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4350.0  on 5810  degrees of freedom
## Residual deviance: 4277.8  on 5787  degrees of freedom
## AIC: 4325.8
##
## Number of Fisher Scoring iterations: 5
```

From this model, we can see that the dominant_song_sentiment has a significant impact on the dominant_twit_sentiment, with "fear," "joy," and "sadness" being associated with a reduced likelihood of positive dominant_twit_sentiment compared to "anger." The interaction between dominant_song_sentiment and disorder appears less consistently significant, but there is evidence that sadness combined with certain disorders (e.g., panic and PTSD) may influence the likelihood of positive dominant_twit_sentiment.

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Caricamento pacchetto: 'pROC'
```

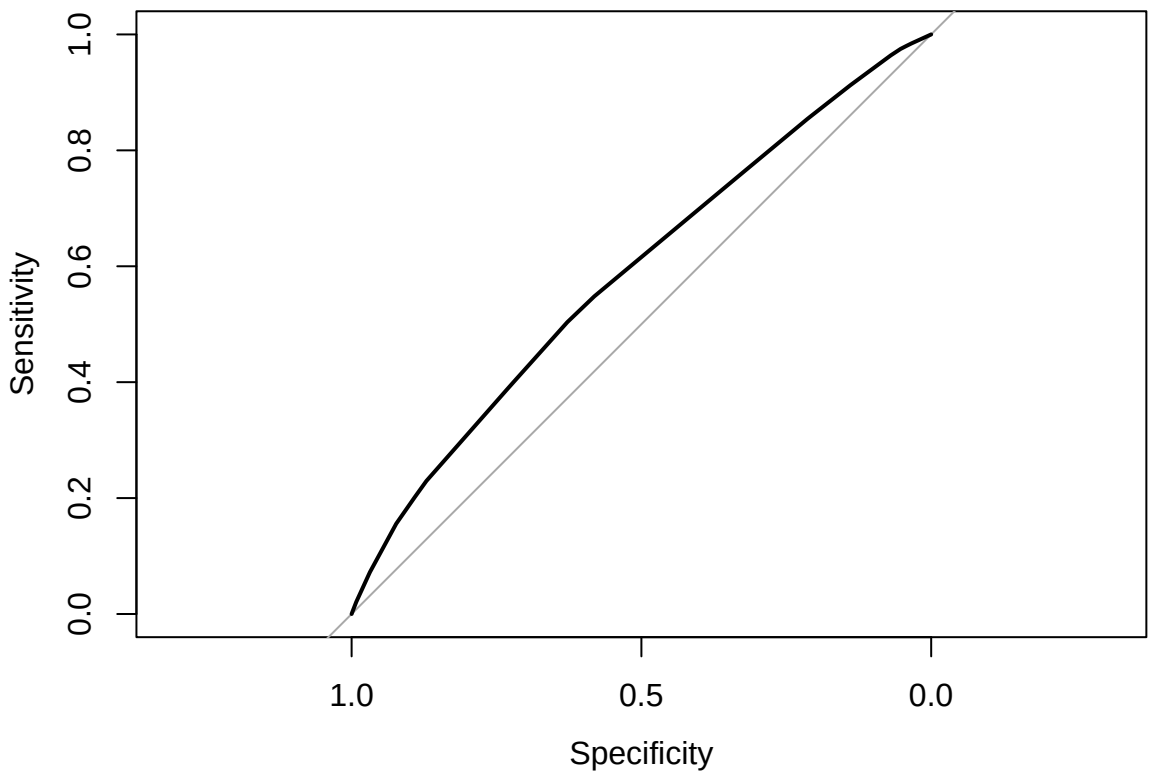
```
## I seguenti oggetti sono mascherati da 'package:stats':
##
##     cov, smooth, var
```

```
roc_curve = roc(big_final$dominant_twit_sentiment, fitted(model))
```

```
## Setting levels: control = positive, case = negative
```

```
## Setting direction: controls < cases
```

```r
plot(roc_curve)
```




```r
auc(roc_curve)
```

```
## Area under the curve: 0.5874
```

The Roc curve shows that the general model is not a good predictor of the mood of a person

What if we try to analyze only the positive vs negative of songs?

```r
dominant_sentiment_ms_data = ms_sent_data %>%
  group_by(user_id, disorder) %>%
  summarise(
    positive = sum(positive),
    negative = sum(negative),
    .groups = 'drop'
  )

dominant_sentiment_ms_data = dominant_sentiment_ms_data %>%
  rowwise() %>%
  mutate(dominant_song_sentiment = names(cur_data())[which.max(c(positive, negative)) + 2]) %>%
  select(user_id, disorder, dominant_song_sentiment)

dominant_sentiment_ms_data
```

```
## # A tibble: 4,397 x 3
## # Rowwise:
##    user_id    disorder   dominant_song_sentiment
##    <chr>      <chr>      <chr>
##  1 0001e573dc anxiety    positive
##  2 00021bdb1d anxiety    negative
##  3 00021bdb1d depression negative
##  4 0003c5dea3 panic      negative
##  5 00180e338b ptsd       negative
##  6 001fec300a ptsd       negative
##  7 0028a9ef7c anxiety    negative
##  8 004fcce5bd depression negative
##  9 0052e8ca6f depression positive
## 10 0067dd24ff depression positive
## # i 4,387 more rows
```

```
# Joining everything in one dataset
big_final = dominant_sentiment_ms_data %>%
  inner_join(dominant_ms_topics, by = c("user_id", "disorder")) %>%
  inner_join(dominant_sentiment_tw_data, by = c("user_id", "disorder"))

big_final$dominant_twit_sentiment = as.factor(big_final$dominant_twit_sentiment)
head(big_final)
```

```
## # A tibble: 6 x 5
## # Rowwise:
##   user_id    disorder   dominant_song_sentiment dominant_song_topic
##   <chr>      <chr>      <chr>                   <chr>
## 1 0001e573dc anxiety    positive                Playful Beats
## 2 00021bdb1d anxiety    negative                Heartache Groove
## 3 00021bdb1d depression negative                Heartache Groove
## 4 0003c5dea3 panic      negative                Lovesick Chaos
## 5 0003c5dea3 panic      negative                Lovesick Melody
## 6 00180e338b ptsd       negative                Playful Beats
## # i 1 more variable: dominant_twit_sentiment <fct>
```

```
big_final$dominant_twit_sentiment = relevel(big_final$dominant_twit_sentiment, ref = "positive")

# Fit logistic regression model
model = glm(dominant_twit_sentiment ~ disorder * dominant_song_sentiment ,
            data = big_final, family = binomial)

# Summary of the model
summary(model)
```

```
##
## Call:
## glm(formula = dominant_twit_sentiment ~ disorder * dominant_song_sentiment,
##     family = binomial, data = big_final)
##
## Coefficients:
##                                                 Estimate Std. Error z value
## (Intercept)                                    -1.848189   0.114043 -16.206
```

```
## disorderbipolar                                        -0.149907   0.245606  -0.610
## disorderborderline                                     -0.324034   0.369831  -0.876
## disorderdepression                                      0.073761   0.137966   0.535
## disorderpanic                                           0.183182   0.427656   0.428
## disorderptsd                                           -0.007649   0.173236  -0.044
## dominant_song_sentimentpositive                        -0.336950   0.167449  -2.012
## disorderbipolar:dominant_song_sentimentpositive         0.043043   0.365111   0.118
## disorderborderline:dominant_song_sentimentpositive      0.663347   0.498297   1.331
## disorderdepression:dominant_song_sentimentpositive      0.091360   0.201916   0.452
## disorderpanic:dominant_song_sentimentpositive          -0.770630   0.743022  -1.037
## disorderptsd:dominant_song_sentimentpositive            0.106021   0.259007   0.409
##                                                       Pr(>|z|)
## (Intercept)                                             <2e-16 ***
## disorderbipolar                                         0.5416
## disorderborderline                                      0.3809
## disorderdepression                                      0.5929
## disorderpanic                                           0.6684
## disorderptsd                                            0.9648
## dominant_song_sentimentpositive                         0.0442 *
## disorderbipolar:dominant_song_sentimentpositive         0.9062
## disorderborderline:dominant_song_sentimentpositive      0.1831
## disorderdepression:dominant_song_sentimentpositive      0.6509
## disorderpanic:dominant_song_sentimentpositive           0.2997
## disorderptsd:dominant_song_sentimentpositive            0.6823
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4350.0  on 5810  degrees of freedom
## Residual deviance: 4332.6  on 5799  degrees of freedom
## AIC: 4356.6
##
## Number of Fisher Scoring iterations: 5
```

Here we see again that the disorder is not significant for the mood of the person, and nieter the intercation terms are.

However we see that the song sentiment positive is significant, with a negative effect

```
big_final$dominant_twit_sentiment = relevel(big_final$dominant_twit_sentiment, ref = "positive")

# Fit logistic regression model
model = glm(dominant_twit_sentiment ~ dominant_song_sentiment ,
            data = big_final, family = binomial)

# Summary of the model
summary(model)
```

```
##
## Call:
## glm(formula = dominant_twit_sentiment ~ dominant_song_sentiment,
##     family = binomial, data = big_final)
##
```

```
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     -1.82982    0.05448 -33.588  < 2e-16 ***
## dominant_song_sentimentpositive -0.26137    0.07996  -3.269  0.00108 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4350.0  on 5810  degrees of freedom
## Residual deviance: 4339.3  on 5809  degrees of freedom
## AIC: 4343.3
##
## Number of Fisher Scoring iterations: 4
```

This means that people that are listening to positive music are less likely to have positive mood. This could be interpreted as people who are in a bad mood prefer positive songs.

So it seems that the correlation that I'm seeking is the inverse of what I was expecting. From this analysis it seems that music can't influence people mood with a direct correlation, but the music that a person listen is chosed ad an inverse function of the mood. If we try to reverse the regression:

```
big_final$dominant_song_sentiment = as.factor(big_final$dominant_song_sentiment)
big_final$dominant_song_sentiment = relevel(big_final$dominant_song_sentiment, ref = "negative")

# Fit logistic regression model
model = glm(dominant_song_sentiment ~ disorder * dominant_twit_sentiment ,
            data = big_final, family = binomial)

# Summary of the model
summary(model)
```

```
##
## Call:
## glm(formula = dominant_song_sentiment ~ disorder * dominant_twit_sentiment,
##     family = binomial, data = big_final)
##
## Coefficients:
##                                                  Estimate Std. Error z value
## (Intercept)                                       0.15238    0.05736   2.657
## disorderbipolar                                  -0.09209    0.11941  -0.771
## disorderborderline                               -0.19109    0.17060  -1.120
## disorderdepression                               -0.04475    0.07035  -0.636
## disorderpanic                                     0.10790    0.22616   0.477
## disorderptsd                                     -0.20913    0.08956  -2.335
## dominant_twit_sentimentnegative                  -0.33695    0.16745  -2.012
## disorderbipolar:dominant_twit_sentimentnegative   0.04304    0.36511   0.118
## disorderborderline:dominant_twit_sentimentnegative  0.66335    0.49829   1.331
## disorderdepression:dominant_twit_sentimentnegative  0.09136    0.20192   0.452
## disorderpanic:dominant_twit_sentimentnegative    -0.77063    0.74258  -1.038
## disorderptsd:dominant_twit_sentimentnegative      0.10602    0.25900   0.409
##                                                  Pr(>|z|)
## (Intercept)                                       0.00789 **
## disorderbipolar                                   0.44059
```

```
## disorderborderline                                      0.26267
## disorderdepression                                      0.52470
## disorderpanic                                           0.63328
## disorderptsd                                            0.01954 *
## dominant_twit_sentimentnegative                         0.04419 *
## disorderbipolar:dominant_twit_sentimentnegative         0.90615
## disorderborderline:dominant_twit_sentimentnegative      0.18311
## disorderdepression:dominant_twit_sentimentnegative      0.65093
## disorderpanic:dominant_twit_sentimentnegative           0.29938
## disorderptsd:dominant_twit_sentimentnegative            0.68229
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8051.6  on 5810  degrees of freedom
## Residual deviance: 8031.0  on 5799  degrees of freedom
## AIC: 8055
##
## Number of Fisher Scoring iterations: 3
```

From this we see that individuals with negative mood are generally more likely to listen to positive songs compared to those with positive mood. PTSD shows a significant main effect, indicating that these individuals are also less likely to listen to negative songs overall. However, also in this case, there is no significant evidence that the relationship between mood and songs sentiments varies based on specific disorders.

Let's see what happens if we fit the positive/negative mood over the emotion's scores of the songs.

```
sum_sentiment_ms_data = ms_sent_data %>%
  group_by(user_id, disorder) %>%
  summarise(
    anger = sum(anger),
    fear = sum(fear),
    joy = sum(joy),
    sadness = sum(sadness),
    .groups = 'drop'
  )

sum_sentiment_ms_data
```

```
## # A tibble: 4,397 x 6
##    user_id    disorder   anger  fear   joy sadness
##    <chr>      <chr>      <int> <int> <int>   <int>
##  1 0001e573dc anxiety       11     9    10      13
##  2 00021bdb1d anxiety        9    10     4      12
##  3 00021bdb1d depression     9    10     4      12
##  4 0003c5dea3 panic         97   120    94     103
##  5 00180e338b ptsd        3566  3575  3070    3104
##  6 001fec300a ptsd           4     7     5       5
##  7 0028a9ef7c anxiety        3     3     0       4
##  8 004fcce5bd depression    23    30    23      30
##  9 0052e8ca6f depression    43    29    64      36
## 10 0067dd24ff depression    46    32    66      41
## # i 4,387 more rows
```

```r
count_ms_topics = ms_topic %>%
  # Count the occurrences of each topic per user and disorder
  count(user_id, disorder, topic_name) %>%
  # Reshape to wide format
  pivot_wider(names_from = topic_name, values_from = n, values_fill = 0)
count_ms_topics
```

```
## # A tibble: 4,397 x 22
##    user_id    disorder   `Playful Beats` `Heartache Groove` `Amorous Yearning`
##    <chr>      <chr>                <int>              <int>               <int>
##  1 0001e573dc anxiety                  2                  0                   0
##  2 00021bdb1d anxiety                  0                  1                   0
##  3 00021bdb1d depression               0                  1                   0
##  4 0003c5dea3 panic                    1                  1                   2
##  5 00180e338b ptsd                   194                  7                  11
##  6 001fec300a ptsd                     1                  0                   0
##  7 0028a9ef7c anxiety                  0                  0                   0
##  8 004fcce5bd depression               0                  0                   0
##  9 0052e8ca6f depression               2                  0                   2
## 10 0067dd24ff depression               2                  0                   2
## # i 4,387 more rows
## # i 17 more variables: `Baby's Love` <int>, `Dancefloor Vibes` <int>,
## #   `Desire's Call` <int>, `Forsaken Rhymes` <int>, `Heartache Lament` <int>,
## #   `Heartbreaker Vibes` <int>, `Heavenly Devotion` <int>, `Illest Flow` <int>,
## #   `Lover's Groove` <int>, `Lovesick Chaos` <int>, `Lovesick Melody` <int>,
## #   `Morbid Heartache` <int>, `Sinking Lullaby` <int>, `Bumping Rhythms` <int>,
## #   `Goodbye Love` <int>, `Loveless Escape` <int>, `Lying Heart` <int>
```

```r
dominant_sentiment_tw_data = tw_sent_data %>%
  group_by(user_id, disorder) %>%
  summarise(
    positive = sum(positive),
    negative = sum(negative),
    .groups = 'drop'
  )

dominant_sentiment_tw_data = dominant_sentiment_tw_data %>%
  rowwise() %>%
  mutate(dominant_twit_sentiment = names(cur_data())[which.max(c(positive, negative)) + 2]) %>%
  select(user_id, disorder, dominant_twit_sentiment)

dominant_sentiment_tw_data
```

```
## # A tibble: 4,407 x 3
## # Rowwise:
##    user_id    disorder   dominant_twit_sentiment
##    <chr>      <chr>      <chr>
##  1 0001e573dc anxiety    positive
##  2 00021bdb1d anxiety    positive
##  3 00021bdb1d depression positive
##  4 0003c5dea3 panic      positive
##  5 00180e338b ptsd       negative
```

```
##  6 001fec300a ptsd       positive
##  7 0028a9ef7c anxiety    positive
##  8 004fcce5bd depression positive
##  9 0052e8ca6f depression positive
## 10 0067dd24ff depression positive
## # i 4,397 more rows
```

```
big_final = sum_sentiment_ms_data %>%
  inner_join(count_ms_topics, by = c("user_id", "disorder")) %>%
  inner_join(dominant_sentiment_tw_data, by = c("user_id", "disorder"))

big_final$dominant_twit_sentiment = as.factor(big_final$dominant_twit_sentiment)
big_final
```

```
## # A tibble: 4,395 x 27
##    user_id disorder anger  fear   joy sadness `Playful Beats` `Heartache Groove`
##    <chr>   <chr>    <int> <int> <int>   <int>           <int>               <int>
##  1 0001e5~ anxiety     11     9    10      13               2                   0
##  2 00021b~ anxiety      9    10     4      12               0                   1
##  3 00021b~ depress~     9    10     4      12               0                   1
##  4 0003c5~ panic       97   120    94     103               1                   1
##  5 00180e~ ptsd      3566  3575  3070    3104             194                   7
##  6 001fec~ ptsd         4     7     5       5               1                   0
##  7 0028a9~ anxiety      3     3     0       4               0                   0
##  8 004fcc~ depress~    23    30    23      30               0                   0
##  9 0052e8~ depress~    43    29    64      36               2                   0
## 10 0067dd~ depress~    46    32    66      41               2                   0
## # i 4,385 more rows
## # i 19 more variables: `Amorous Yearning` <int>, `Baby's Love` <int>,
## #   `Dancefloor Vibes` <int>, `Desire's Call` <int>, `Forsaken Rhymes` <int>,
## #   `Heartache Lament` <int>, `Heartbreaker Vibes` <int>,
## #   `Heavenly Devotion` <int>, `Illest Flow` <int>, `Lover's Groove` <int>,
## #   `Lovesick Chaos` <int>, `Lovesick Melody` <int>, `Morbid Heartache` <int>,
## #   `Sinking Lullaby` <int>, `Bumping Rhythms` <int>, `Goodbye Love` <int>, ...
```

```
big_final$dominant_twit_sentiment = relevel(big_final$dominant_twit_sentiment, ref = "positive")
model = glm(dominant_twit_sentiment ~ .-user_id, data = big_final, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = dominant_twit_sentiment ~ . - user_id, family = binomial,
##     data = big_final)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.9587138  0.1004928 -19.491  < 2e-16 ***
## disorderbipolar     -0.1357383  0.2093600  -0.648 0.516760
## disorderborderline  -0.0722794  0.2936576  -0.246 0.805578
## disorderdepression   0.0972430  0.1163059   0.836 0.403100
## disorderpanic        0.0442059  0.3914595   0.113 0.910089
## disorderptsd         0.0518611  0.1488865   0.348 0.727595
## anger               -0.0017211  0.0025097  -0.686 0.492841
```

```
## fear                 -0.0003217  0.0025331  -0.127 0.898943
## joy                  -0.0045449  0.0013631  -3.334 0.000855 ***
## sadness               0.0057688  0.0025397   2.271 0.023118 *
## ‘Playful Beats‘       0.0129076  0.0125706   1.027 0.304510
## ‘Heartache Groove‘   -0.0198596  0.0247706  -0.802 0.422703
## ‘Amorous Yearning‘    0.0384669  0.0212800   1.808 0.070659 .
## ‘Baby’s Love‘        -0.0211689  0.0331318  -0.639 0.522868
## ‘Dancefloor Vibes‘   -0.0377747  0.0289201  -1.306 0.191494
## ‘Desire’s Call‘       0.0425499  0.0341009   1.248 0.212118
## ‘Forsaken Rhymes‘    -0.0107172  0.0218201  -0.491 0.623311
## ‘Heartache Lament‘    0.0324422  0.0409614   0.792 0.428350
## ‘Heartbreaker Vibes‘ -0.0142539  0.0183471  -0.777 0.437218
## ‘Heavenly Devotion‘  -0.0391903  0.0363989  -1.077 0.281619
## ‘Illest Flow‘         0.0193646  0.0136875   1.415 0.157138
## ‘Lover’s Groove‘      0.0198824  0.0161284   1.233 0.217665
## ‘Lovesick Chaos‘     -0.0204372  0.0146165  -1.398 0.162046
## ‘Lovesick Melody‘    -0.0200115  0.0254194  -0.787 0.431135
## ‘Morbid Heartache‘    0.0063423  0.0230588   0.275 0.783277
## ‘Sinking Lullaby‘    -0.0521609  0.0484047  -1.078 0.281212
## ‘Bumping Rhythms‘     0.0045819  0.0281751   0.163 0.870815
## ‘Goodbye Love‘        0.0304248  0.0298305   1.020 0.307765
## ‘Loveless Escape‘    -0.0698175  0.0311848  -2.239 0.025167 *
## ‘Lying Heart‘         0.0452927  0.0356327   1.271 0.203693
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3341.4  on 4394  degrees of freedom
## Residual deviance: 3241.4  on 4365  degrees of freedom
## AIC: 3301.4
##
## Number of Fisher Scoring iterations: 6
```

```
big_final = sum_sentiment_ms_data %>%
  inner_join(dominant_sentiment_tw_data, by = c("user_id", "disorder"))

big_final$dominant_twit_sentiment = as.factor(big_final$dominant_twit_sentiment)
big_final
```

```
## # A tibble: 4,395 x 7
##    user_id    disorder   anger  fear   joy sadness dominant_twit_sentiment
##    <chr>      <chr>      <int> <int> <int>   <int> <fct>
##  1 0001e573dc anxiety       11     9    10      13 positive
##  2 00021bdb1d anxiety        9    10     4      12 positive
##  3 00021bdb1d depression     9    10     4      12 positive
##  4 0003c5dea3 panic         97   120    94     103 positive
##  5 00180e338b ptsd        3566  3575  3070    3104 negative
##  6 001fec300a ptsd           4     7     5       5 positive
##  7 0028a9ef7c anxiety        3     3     0       4 positive
##  8 004fcce5bd depression    23    30    23      30 positive
##  9 0052e8ca6f depression    43    29    64      36 positive
## 10 0067dd24ff depression    46    32    66      41 positive
## # i 4,385 more rows
```

```r
big_final$dominant_twit_sentiment = relevel(big_final$dominant_twit_sentiment, ref = "positive")
model = glm(dominant_twit_sentiment ~ .-user_id, data = big_final, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = dominant_twit_sentiment ~ . - user_id, family = binomial,
##     data = big_final)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.9588052  0.0991325 -19.759  < 2e-16 ***
## disorderbipolar    -0.1325697  0.2084949  -0.636  0.52488
## disorderborderline -0.1193296  0.2917795  -0.409  0.68256
## disorderdepression  0.0884118  0.1155103   0.765  0.44403
## disorderpanic      -0.0233838  0.3897054  -0.060  0.95215
## disorderptsd        0.0209651  0.1478029   0.142  0.88720
## anger               0.0050942  0.0011679   4.362 1.29e-05 ***
## fear               -0.0050354  0.0018904  -2.664  0.00773 **
## joy                -0.0038208  0.0007261  -5.262 1.43e-07 ***
## sadness             0.0041351  0.0015332   2.697  0.00700 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3341.4  on 4394  degrees of freedom
## Residual deviance: 3271.7  on 4385  degrees of freedom
## AIC: 3291.7
##
## Number of Fisher Scoring iterations: 5
```

From this model the inverse correlation that was observed before is still present, with a statistical significance much higher. It can be seen that Joy (being the best emotion of the group) has a negative estimate, meaning that people that listen to songs with high joy scores are less likely to have a positive mood. The inverse can be observe for sadness, leading to a higher likelihood of being positive. Also emotion like fear and anger are statistically significant, with anger having the same effect of sadness while fear is much similar to joy.