Network Analysis

Shara He, Adanya Johnson

6/7/2021

Setting Up the Environment

You will need the following packages installed:
`brainGraph`, `igraph`, `tidyverse`, `ggplot2`, `grid`, `gridExtra`, `knitr`, `kableExtra`

Preparing Data

Load in the required data.

```
# Read in data
load("~/Desktop/School/GatesLab/NetworkAnalysis/FullMatrices.Rdata")

# Preview all data
# head(outReg)
```

Extract Submatrices for each Person

The R data `outReg` is a list of 40 lists; each list corresponds to an individual and contains a 52 by 52 regression matrix. We want to analyze the first 26 variables for each individual, and we will extract a 26 by 26 regression matrix for each person. The matrices are saved in two different lists. The first list `subdf` contains the original 26 by 26 regression matrix for each individual person. The second list `subdf_pos` contains a positive regression matrix, where every negative value is converted into zero.

```
submatrix = list()
for (x in 1:length(outReg)) {
  submatrix = append(submatrix, list(outReg[[x]][["regression_matrix"]][1:26, 1:26]))
}

# Convert all submatrices into double type
feature_names = c("energetic", "enthusiastic", "content", "irritable", "restless",
                  "worried", "guilty",  "afraid", "anhedonia", "angry","hopeless",
                  "down", "positive", "fatigue", "tension", "concentrate", "accepted",
                  "threatened", "ruminate", "avoid_act", "reassure", "procrast",
                  "hours", "difficult", "unsatisfy", "avoid_people")
subdf = list()
subdf_pos = list()

for (x in 1:length(outReg)) {
  subdf[[x]] = matrix(
    unlist(submatrix[x]),
    ncol = 26,
```

```
    byrow = FALSE,
    dimnames = list(feature_names, feature_names)
  )
  subdf_pos[[x]] = pmax(subdf[[x]], 0)
}
```

Extract Directed Graphs for each Person

Here, we convert both the original and positive matrices for each individual into directed graphs. The original and positive graphs are stored in two separate lists.

```
dir_graph = list()
dir_graph_pos = list()

for (x in 1:length(outReg)) {
  dir_graph[[x]] = graph.adjacency(subdf[[x]],
                                   mode = "directed",
                                   weighted = TRUE)
  dir_graph_pos[[x]] = graph.adjacency(subdf_pos[[x]],
                                       mode = "directed",
                                       weighted = TRUE)
}

dir_graph[[19]]
```

```
## IGRAPH dddcb24 DNW- 26 114 --
## + attr: name (v/c), weight (e/n)
## + edges from dddcb24 (vertex names):
##  [1] energetic   ->enthusiastic energetic   ->down
##  [3] energetic   ->fatigue      energetic   ->tension
##  [5] energetic   ->ruminate     energetic   ->avoid_act
##  [7] enthusiastic->energetic    enthusiastic->positive
##  [9] enthusiastic->tension      enthusiastic->procrast
## [11] content     ->irritable    content     ->restless
## [13] content     ->worried      content     ->guilty
## [15] content     ->angry        content     ->down
## + ... omitted several edges
```

```
dir_graph_pos[[19]]
```

```
## IGRAPH db83915 DNW- 26 73 --
## + attr: name (v/c), weight (e/n)
## + edges from db83915 (vertex names):
##  [1] energetic   ->enthusiastic enthusiastic->energetic
##  [3] enthusiastic->positive     content     ->positive
##  [5] irritable   ->worried      irritable   ->angry
##  [7] irritable   ->threatened   irritable   ->ruminate
##  [9] restless    ->worried      restless    ->angry
## [11] restless    ->threatened   restless    ->avoid_people
## [13] worried     ->irritable    worried     ->restless
## [15] worried     ->guilty       worried     ->afraid
## + ... omitted several edges
```

Metrics for each Person

Now, using the directed graphs, we will calculate the following metrics for each person: Density; Overall Weight; Edge Weight; Global Efficiency; Degree of Each Variable; Strength of Each Variable; Betweenness Centrality; Community Detection.

Note, Global Efficiency, Betweenness Centrality, and Community Detection require the positive directed graphs rather than the original graph.

```
density = list()
overall_weight = list()
edge_weights = list()
global_efficiency = list()
variable_degrees = list()
variable_strengths = list()
betweenness = list()
cluster = list()


for (x in 1:length(outReg)) {
  density[[x]] = edge_density(dir_graph[[x]], loops = FALSE)
  overall_weight[[x]] = sum(strength(dir_graph[[x]]))
  edge_weights[[x]] = edge_attr(dir_graph_pos[[x]], "weight")

  global_efficiency[[x]] = efficiency(dir_graph_pos[[x]],
                                      type = c("global"),
                                      weights = edge_weights[[x]])
  variable_degrees[[x]] = degree(dir_graph[[x]])
  variable_strengths[[x]] = strength(dir_graph[[x]])

  betweenness[[x]] = estimate_betweenness(dir_graph_pos[[x]],
                                          cutoff = -1, weights = edge_weights[[x]])
  cluster[[x]] = cluster_walktrap(
    dir_graph_pos[[x]],
    weights = E(dir_graph_pos[[x]])$edge_weights,
    steps = 4
  )
}
print(paste0("Density of the first person's graph: ", round(density[[1]], 4)))
```

```
## [1] "Density of the first person's graph: 0.1077"
```

```
print(paste0(
  "Overall Weight of the first person's graph: ",
  round(overall_weight[[1]], 4)
))
```

```
## [1] "Overall Weight of the first person's graph: 10.6119"
```

```
print(paste0(
  "Global Efficiency of the first person's graph: ",
  round(global_efficiency[[1]], 4)
))
```

```
## [1] "Global Efficiency of the first person's graph: 12.5183"
```

```r
print("Degree of each variable in the first person's graph: ")
```

```
## [1] "Degree of each variable in the first person's graph: "
```

```r
print(variable_degrees[[1]])
```

```
##    energetic enthusiastic      content    irritable     restless      worried
##            4            4            3            8            5            9
##       guilty       afraid    anhedonia        angry     hopeless         down
##            1            9           12            7           11            8
##     positive      fatigue      tension  concentrate     accepted   threatened
##            7            0            7            3            1            3
##     ruminate    avoid_act     reassure     procrast        hours    difficult
##           13            3            1            2            2            8
##    unsatisfy avoid_people
##            1            8
```

```r
print("Strengths of each variable in the first person's graph: ")
```

```
## [1] "Strengths of each variable in the first person's graph: "
```

```r
print(variable_strengths[[1]])
```

```
##    energetic enthusiastic      content    irritable     restless      worried
##   0.76229416   0.96384765   0.47966325   0.83037658   0.57779129   1.20858363
##       guilty       afraid    anhedonia        angry     hopeless         down
##   0.02643344  -0.01065439   0.82426251   0.99845363   0.89364307   0.74160674
##     positive      fatigue      tension  concentrate     accepted   threatened
##   0.95930381   0.00000000   0.76755289   0.18317240   0.14779364   0.16860533
##     ruminate    avoid_act     reassure     procrast        hours    difficult
##   0.79906030   0.47042696   0.04843503  -0.06829517  -1.55552557  -0.56623424
##    unsatisfy avoid_people
##   0.20728066   0.75406944
```

```r
print("Betweenness Centrality of each variable in the first person's graph: ")
```

```
## [1] "Betweenness Centrality of each variable in the first person's graph: "
```

```r
print(betweenness[[1]])
```

```
##    energetic enthusiastic      content    irritable     restless      worried
##            0            5            0           40            1            0
##       guilty       afraid    anhedonia        angry     hopeless         down
##            0           10           49           26           85           37
##     positive      fatigue      tension  concentrate     accepted   threatened
##           46            0            2            0            0            0
##     ruminate    avoid_act     reassure     procrast        hours    difficult
##          111            0            0            0            0            0
##    unsatisfy avoid_people
##            0           51
```

```r
print("Clustering Walktrap of each variable in the first person's graph: ")
```

```
## [1] "Clustering Walktrap of each variable in the first person's graph: "
```

```r
print(cluster[[1]])
```

```
## IGRAPH clustering walktrap, groups: 11, mod: 0.27
## + groups:
```

```
##   $`1`
##   [1] "concentrate" "difficult"
##
##   $`2`
##   [1] "angry"        "down"         "ruminate"     "avoid_act"    "avoid_people"
##
##   $`3`
##   [1] "content"  "positive" "accepted"
##
##   $`4`
##   + ... omitted several groups/vertices
```

Computing Metric Summary Statistics

Here, we calculate the mean, standard deviation, and range for Density, Overall Weight, Global Efficiency, Degree of Each Variable, Strength of Each Variable, and Betweenness Centrality. The summary statistics for Density, Overall Weight, and Global Efficiency are stored within a table for accessible viewing.

```
density_mean = mean(unlist(density))
density_sd = sd(unlist(density))
density_range = range(unlist(density))

overall_weight_mean = mean(unlist(overall_weight))
overall_weight_sd = sd(unlist(overall_weight))
overall_weight_range = range(unlist(overall_weight))

global_efficiency_mean = mean(unlist(global_efficiency))
global_efficiency_sd = sd(unlist(global_efficiency))
global_efficiency_range = range(unlist(global_efficiency))

variable_degrees_mat = do.call(rbind, variable_degrees)
variable_degrees_mean = apply(variable_degrees_mat, 2, mean)
variable_degrees_sd = apply(variable_degrees_mat, 2, sd)
variable_degrees_range = apply(variable_degrees_mat, 2, range)

variable_strengths_mat = do.call(rbind, variable_strengths)
variable_strengths_mean = apply(variable_strengths_mat, 2, mean)
variable_strengths_sd = apply(variable_strengths_mat, 2, sd)
variable_strengths_range = apply(variable_strengths_mat, 2, range)

betweenness_mat = do.call(rbind, betweenness)
betweenness_mean = apply(betweenness_mat, 2, mean)
betweenness_sd = apply(betweenness_mat, 2, sd)
betweenness_range = apply(betweenness_mat, 2, range)

# Store all summary stats into a DataFrame
net_stat = data.frame(
  "Density" = c(density_mean, density_sd, density_range),
  "Overall Weight" = c(overall_weight_mean, overall_weight_sd, overall_weight_range),
  "Global Efficiency" = c(
    global_efficiency_mean,
    global_efficiency_sd,
    global_efficiency_range
```

```
  )
)
row.names(net_stat) = c("Mean", "Standard Deviation", "Minimum", "Maximum")

kt = knitr::kable(
  x = net_stat,
  row.names = TRUE,
  col.names = c("Density", "Overvall Weight", "Global Efficiency")
)
kt
```

	Density	Overvall Weight	Global Efficiency
Mean	0.1651923	10.90231	24.557177
Standard Deviation	0.0643016	10.72964	33.724137
Minimum	0.0461538	-32.41157	1.450913
Maximum	0.3230769	33.45451	187.580449

```
# kable_styling(kt, latex_options = "striped", full_width = F)
```

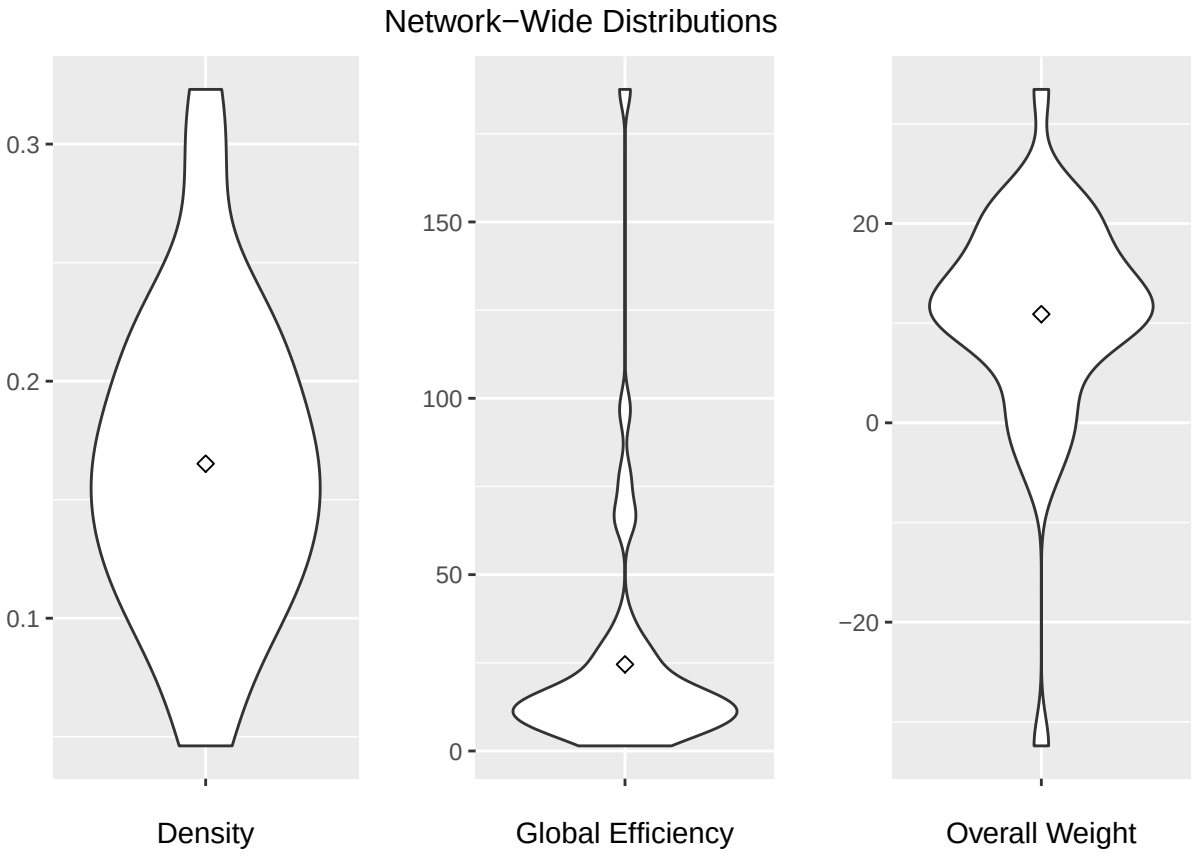
Plots and Distribution

Distribution across each network-wide metric using data

```
# Combine network-wide metrics into a dataframe
network_df = data.frame(
  "Density" = unlist(density),
  "Overall Weight" = unlist(overall_weight),
  "Global Efficiency" = unlist(global_efficiency)
)

# Violin Plots
density_plot = ggplot(network_df, aes(y = Density, x = "")) +
  geom_violin() +
  stat_summary(
    fun = mean,
    geom = "point",
    shape = 23,
    size = 2
  ) +
  xlab("Density") +
  ylab("")
efficiency_plot = ggplot(network_df, aes(y = Global.Efficiency, x = "")) +
  geom_violin() +
  stat_summary(
    fun = mean,
    geom = "point",
    shape = 23,
    size = 2
  ) +
  xlab("Global Efficiency") +
```

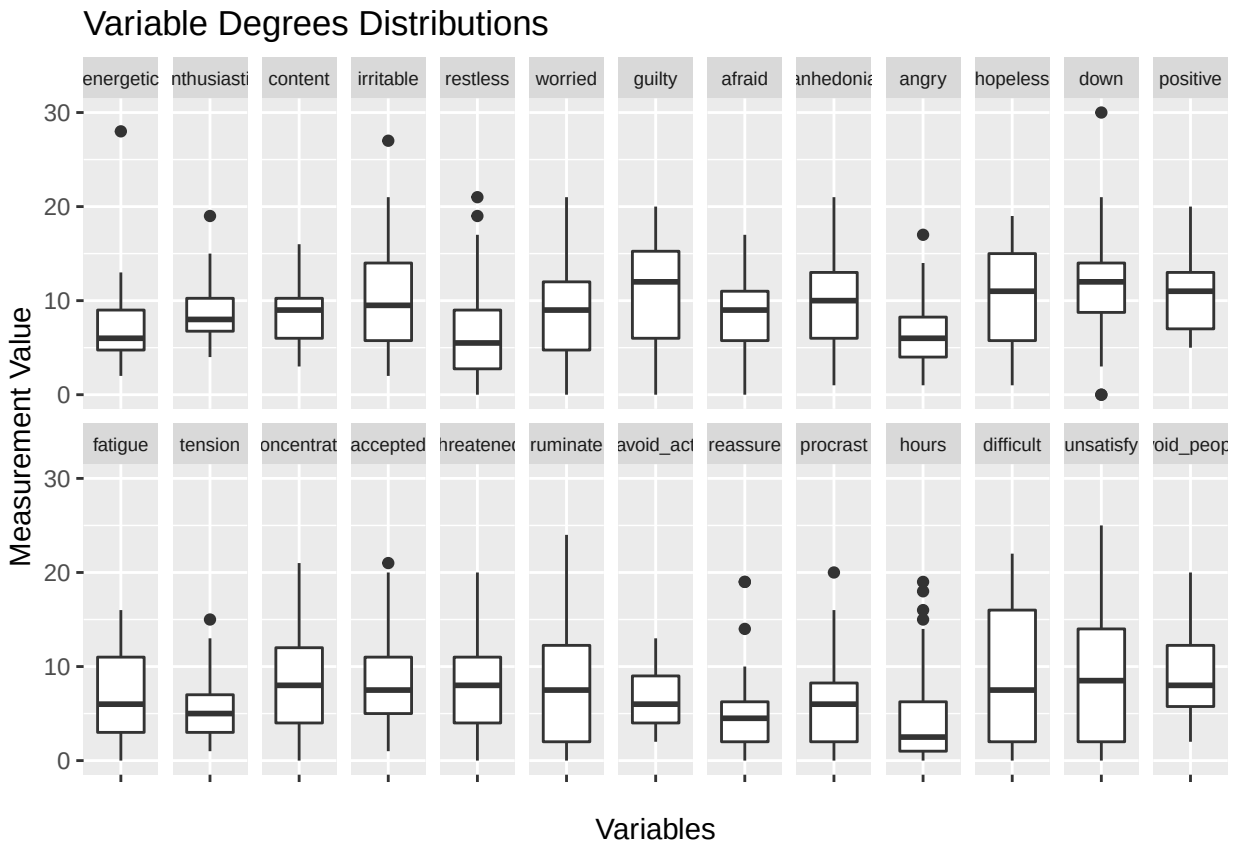
```
  ylab("")
overall_weight_plot = ggplot(network_df, aes(y = Overall.Weight, x = "")) +
  geom_violin() +
  stat_summary(
    fun = mean,
    geom = "point",
    shape = 23,
    size = 2
  ) +
  xlab("Overall Weight") +
  ylab("")
grid.arrange(
  density_plot,
  efficiency_plot,
  overall_weight_plot,
  ncol = 3,
  top = textGrob("Network-Wide Distributions")
)
```



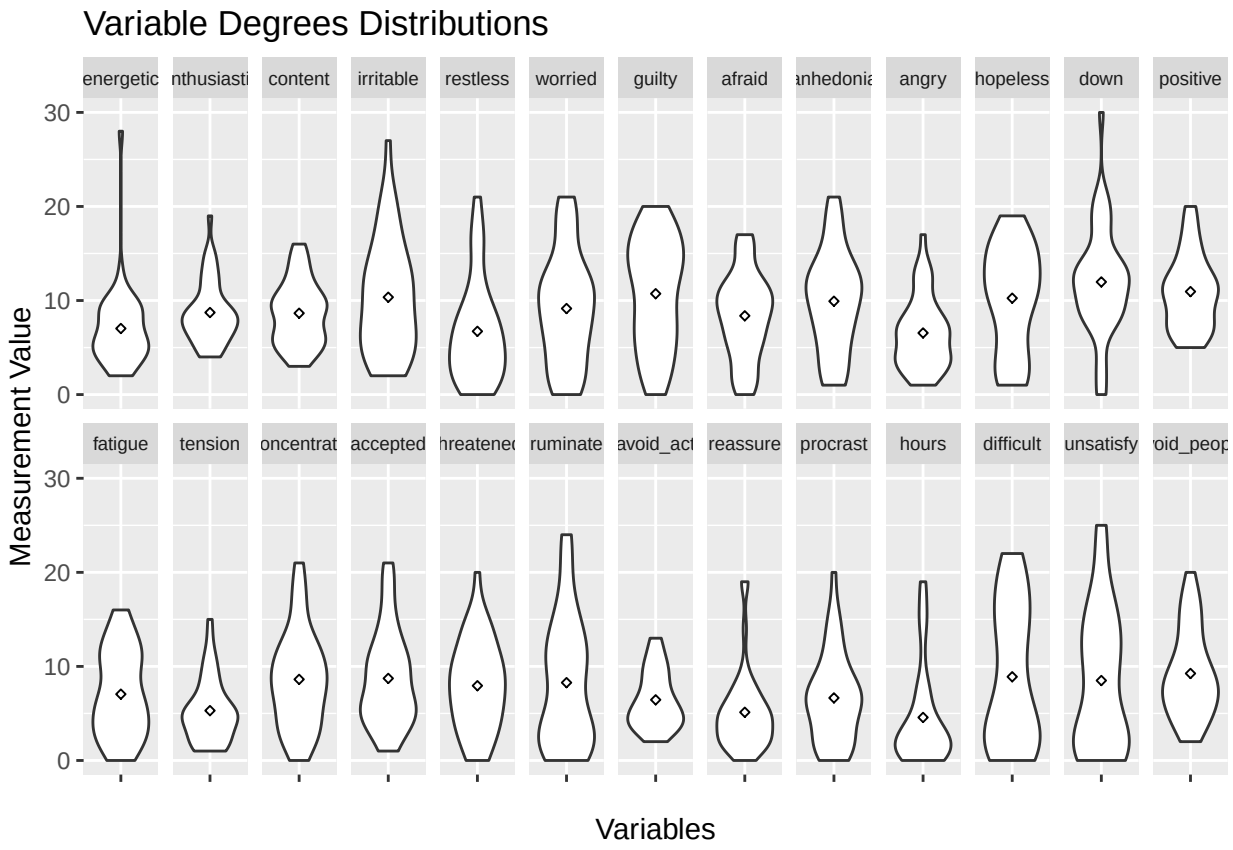
The above plot illustrates that the density, global efficiency, and overall weight measurements are set at different scales and show different distributions. Density shows a relatively symmetrical distribution, whereas Global Efficiency shows a right-skewed distribution and Overall Weight shows a left-skewed distribution.

Plot distributions across each variable-level metric

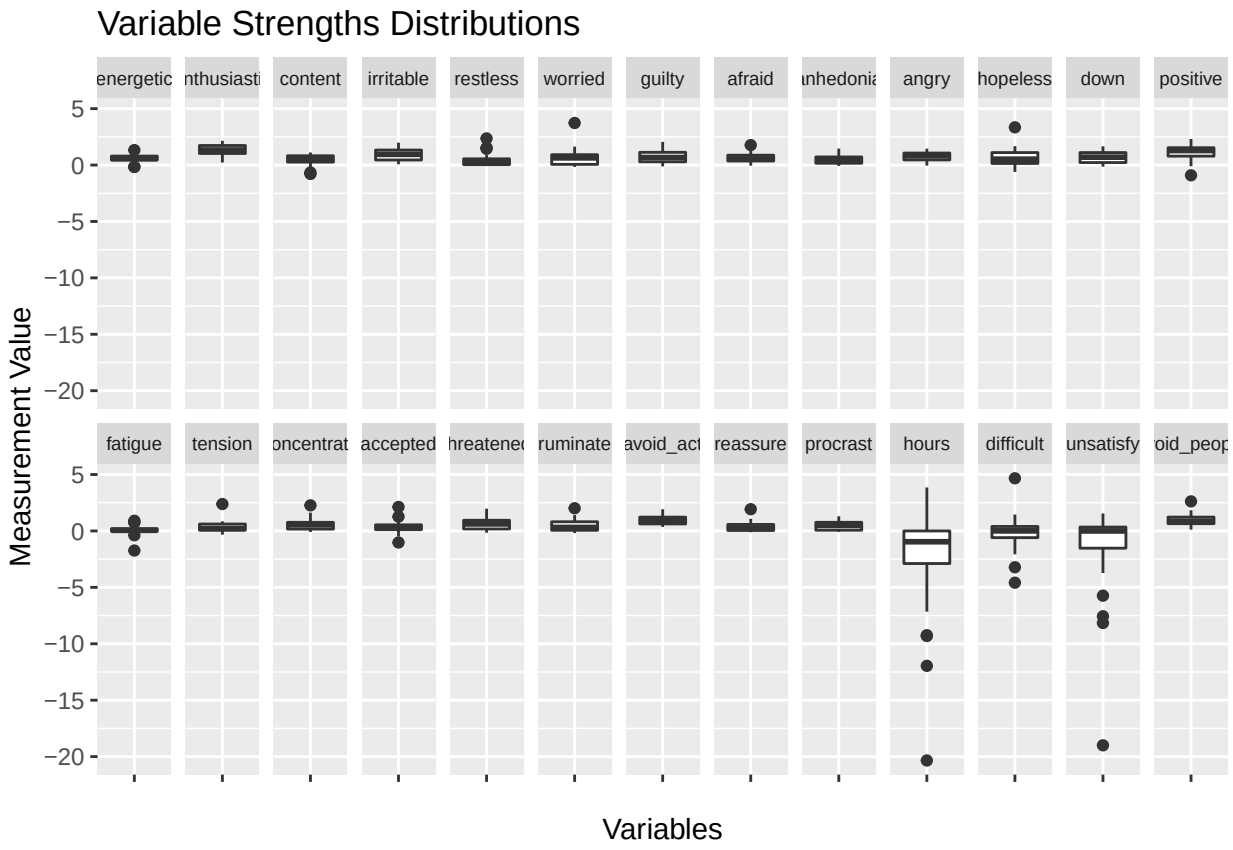
```
# Variable Degrees Boxplot
ggplot(stack(data.frame(variable_degrees_mat)), aes(x = "" , y = values)) +
  geom_boxplot() +
  facet_wrap( ~ ind, ncol = 13) +
  ggtitle("Variable Degrees Distributions") +
  xlab("Variables") +
  ylab("Measurement Value") +
  theme(strip.text = element_text(size = 7))
```



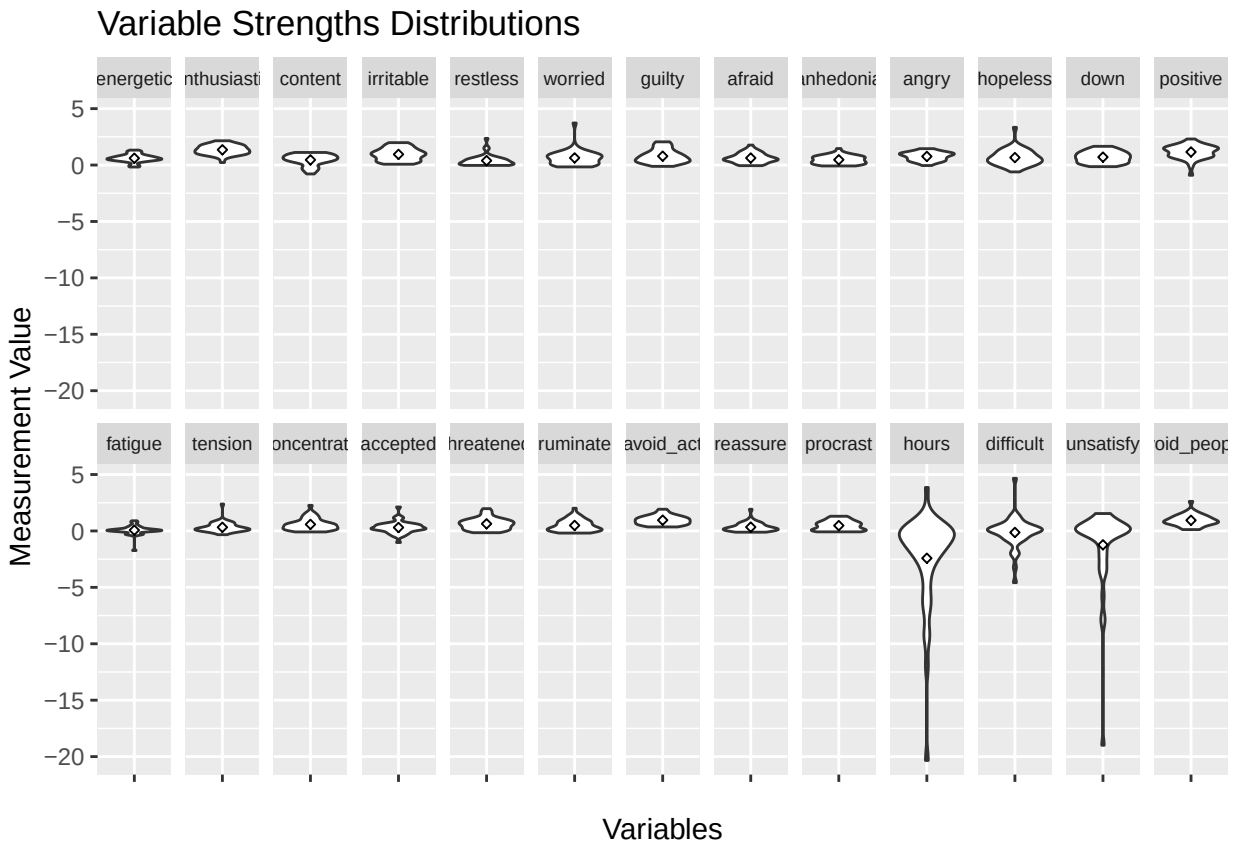

```
# Variable Degrees Violin Plot
ggplot(stack(data.frame(variable_degrees_mat)), aes(x = "" , y = values)) +
  geom_violin() +
  facet_wrap( ~ ind, ncol = 13) +
  ggtitle("Variable Degrees Distributions") +
  xlab("Variables") +
  ylab("Measurement Value") +
  theme(strip.text = element_text(size = 7)) +
  stat_summary(
    fun = mean,
    geom = "point",
    shape = 23,
    size = 1
  )
```



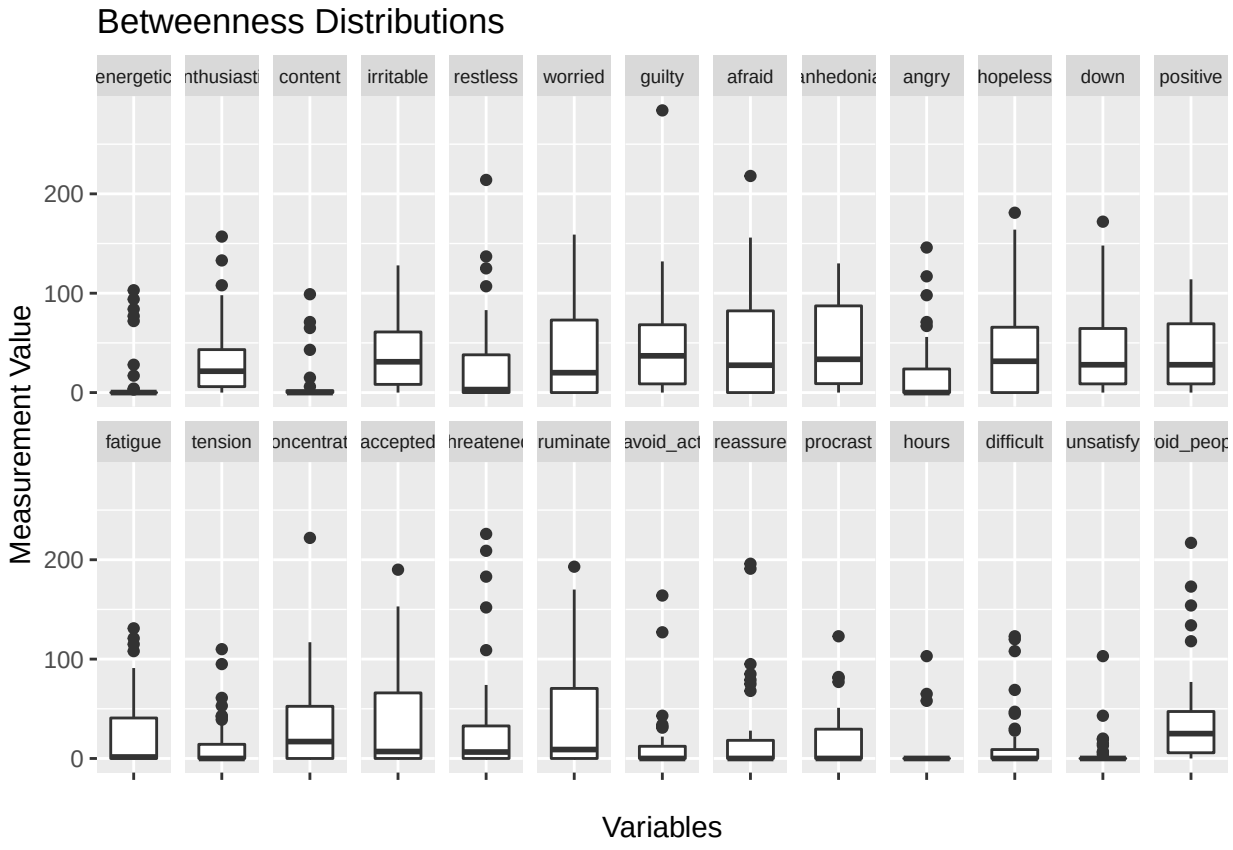

```
# Variable Strengths Boxplot
ggplot(stack(data.frame(variable_strengths_mat)), aes(x = "" , y = values)) +
  geom_boxplot() +
  facet_wrap( ~ ind, ncol = 13) +
  ggtitle("Variable Strengths Distributions") +
  xlab("Variables") +
  ylab("Measurement Value") +
  theme(strip.text = element_text(size = 7))
```



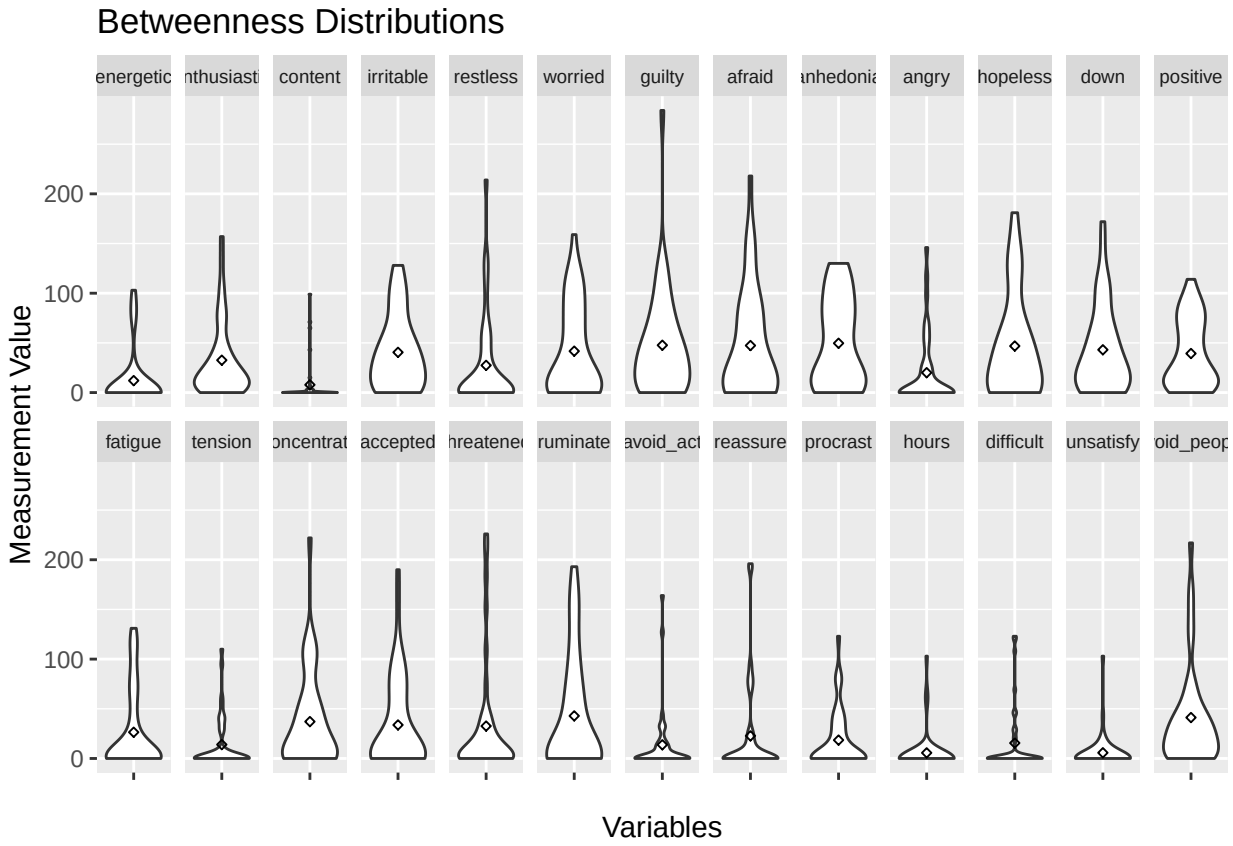

```
# Variable Strengths Violin Plot
ggplot(stack(data.frame(variable_strengths_mat)), aes(x = "" , y = values)) +
  geom_violin() +
  facet_wrap( ~ ind, ncol = 13) +
  ggtitle("Variable Strengths Distributions") +
  xlab("Variables") +
  ylab("Measurement Value") +
  # ylim(-5,5) +
  theme(strip.text = element_text(size = 7)) +
  stat_summary(
    fun = mean,
    geom = "point",
    shape = 23,
    size = 1
  )
```




```r
# Betweenness Boxplot
ggplot(stack(data.frame(betweenness_mat)), aes(x = "" , y = values)) +
  geom_boxplot() +
  facet_wrap( ~ ind, ncol = 13) +
  ggtitle("Betweenness Distributions") +
  xlab("Variables") +
  ylab("Measurement Value") +
  theme(strip.text = element_text(size = 7))
```




```
# Betweenness Violin Plot
ggplot(stack(data.frame(betweenness_mat)), aes(x = "" , y = values)) +
  geom_violin() +
  facet_wrap( ~ ind, ncol = 13) +
  ggtitle("Betweenness Distributions") +
  xlab("Variables") +
  ylab("Measurement Value") +
  theme(strip.text = element_text(size = 7)) +
  stat_summary(
    fun = mean,
    geom = "point",
    shape = 23,
    size = 1
  )
```



The above violin plots help to visualize the skewness in some of the variable distributions. Specifically, many of the variables show a right-skewed distribution for betweenness. However, these distributions are not visibly apparent when we graph only the average measurement value, such as in the comparison plot below.

Plot the average measurement value for variable-level metrics

```
var_level_df = data.frame(variable = feature_names,
                          variable_degrees_mean,
                          variable_strengths_mean,
                          betweenness_mean)
colnames(var_level_df) =
  c("Variable",
    "Variable Degrees",
    "Variable Strengths",
    "Betweenness")
ggplot(data.frame(
  stack(var_level_df),
  variable = c(feature_names, feature_names, feature_names)
),
aes(x = values, y = variable)) +
  geom_point() +
  geom_line(group = 1, orientation = "y") +
  facet_wrap(~ ind, scales = "free") +
  ggtitle("Average Comparisons on Variable Level Metrics") +
  ylab("Variables") +
  xlab("Average Measurement Value")
```

```
## Warning in stack.data.frame(var_level_df): non-vector columns will be ignored
```