Análises de Estímulos Musicais

Frederico Pedrosa

2025-12-03

Contents

Desenho do Estudo

O estudo avaliou 200 participantes (100 músicos e 100 não-músicos, 18-40 anos) em uma tarefa de reconhecimento emocional. Foram utilizados 116 estímulos musicais inéditos, compostos para evocar quatro emoções distintas (Alegria, Medo/Raiva, Tristeza e Serenidade), representantes dos 4 quadrantes do Modelo Circumplexo. A tarefa consistiu em identificar a emoção percebida em cada estímulo (Acurácia/Congruência) e avaliar valência e alerta (SAM). Variáveis de controle incluíram BDI, BAI e questionário musical.

Resumo

Unidimensionalidade Hierárquica A Análise Fatorial Confirmatória (CFA) indicou que um Fator Geral de Habilidade (G_Ability) explica a vasta maioria da variância das emoções específicas (Felicidade, Medo/Raiva, Tristeza), com confiabilidade excelente (= 0.92).

Estabilidade Clínica Modelos Mistos e Random Forest confirmaram que o desempenho no teste não é influenciado por sintomatologia ansiosa (BAI) ou depressiva (BDI). A variância explicada pelo Random Forest foi negativa (-14%), comprovando a ausência de vieses não-lineares associados à psicopatologia.

Independência de Treino: Não houve diferença significativa entre músicos e não-músicos, nem influência da escolaridade ou idade.

Preditação: A única variável preditora significativa foi o Sexo (p = .023) no modelo linear; nó raiz na árvore de decisão), com mulheres apresentando uma leve vantagem no reconhecimento emocional.

1 Set Up

```
## Warning: package 'tidyr' was built under R version 4.3.2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v purrr     1.0.2
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.5.1     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
## Loading required package: MASS
##
##
## Attaching package: 'MASS'
##
##
## The following object is masked from 'package:dplyr':
```

```
##
##     select
##
##
## Loading required package: msm

## Warning: package 'msm' was built under R version 4.3.3

## Loading required package: polycor

## Warning: package 'lavaan' was built under R version 4.3.3

## This is lavaan 0.6-18
## lavaan is FREE software! Please report any bugs.
##
## ###############################################################################
## This is semTools 0.5-6
## All users of R (or SEM) are invited to submit functions or ideas for functions.
## ###############################################################################
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Warning: package 'sjPlot' was built under R version 4.3.3

## Warning: package 'rpart.plot' was built under R version 4.3.2

## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.

## Warning: package 'ggeffects' was built under R version 4.3.3

## [conflicted] Will prefer dplyr::select over any other package.
## New names:
```

1.1 Arrumando o data

```r
# 1. Definir os números sequenciais de 1 a 116, formatados com zeros à esquerda (001, 002, ..., 116)
stimuli_numbers <- sprintf("%03d", 1:116)

# 2. Renomear o grupo de Emoção Percebida (colunas 27 a 142)
names(data)[27:142] <- paste0("Emotion_St_", stimuli_numbers)

# 3. Renomear o grupo de Congruência (colunas 144 a 259)
names(data)[144:259] <- paste0("Congruence_St_", stimuli_numbers)
```

```r
# 4. Renomear o grupo de Arousal (colunas 267 a 382)
names(data)[267:382] <- paste0("Arousal_St_", stimuli_numbers)

# 5. Renomear o grupo de Valence (colunas 384 a 499)
names(data)[384:499] <- paste0("Valence_St_", stimuli_numbers)

# 6. (Recomendado) Remover as colunas vazias se estiverem em branco
data <- data[, -c(26, 143, 266, 383)] # Remove as colunas pelos índices

# 7. Verifique o resultado
head(names(data))
```

```
## [1] "Participants" "Group"        "Order"        "Education"    "Sex"
## [6] "Age"
```

1.2 Transformar em formato longo

```r
# 1. Pivotar as respostas de EMOÇÃO PERCEBIDA (com o prefixo correto)
emotion_long <- data %>%
  select(Participants, Group, starts_with("Emotion_St_")) %>%
  pivot_longer(
    cols = -c(Participants, Group),
    names_to = "Stimulus_ID",
    names_prefix = "Emotion_St_", # <--- AJUSTE AQUI
    values_to = "Perceived_Emotion"
  )

# 2. Pivotar as respostas de CONGRUÊNCIA
congruence_long <- data %>%
  select(Participants, starts_with("Congruence_St_")) %>%
  pivot_longer(
    cols = -Participants,
    names_to = "Stimulus_ID",
    names_prefix = "Congruence_St_",
    values_to = "Congruence"
  )

# 3. Pivotar as respostas de AROUSAL
arousal_long <- data %>%
  select(Participants, starts_with("Arousal_St_")) %>%
  pivot_longer(
    cols = -Participants,
    names_to = "Stimulus_ID",
    names_prefix = "Arousal_St_",
    values_to = "Arousal"
  )

# 4. Pivotar as respostas de VALÊNCIA
valence_long <- data %>%
  select(Participants, starts_with("Valence_St_")) %>%
  pivot_longer(
```

```
    cols = -Participants,
    names_to = "Stimulus_ID",
    names_prefix = "Valence_St_",
    values_to = "Valence"
  )

# 5. Juntar tudo em um único dataframe longo
data_long <- emotion_long %>%
  left_join(congruence_long, by = c("Participants", "Stimulus_ID")) %>%
  left_join(arousal_long, by = c("Participants", "Stimulus_ID")) %>%
  left_join(valence_long, by = c("Participants", "Stimulus_ID")) %>%
  mutate(Stimulus_ID = as.numeric(Stimulus_ID)) # Agora isso vai funcionar!

# 6. Adicionar a coluna com a EMOÇÃO PRETENDIDA para cada estímulo
data_long <- data_long %>%
  mutate(
    Intended_Emotion = case_when(
      Stimulus_ID <= 29 ~ "Happiness",
      Stimulus_ID <= 58 ~ "Fear/Anger",
      Stimulus_ID <= 87 ~ "Serenity",
      TRUE              ~ "Sadness"
    )
  )

# 7. Verifique o resultado (agora deve estar correto)
head(data_long)
```

```
## # A tibble: 6 x 8
##   Participants Group Stimulus_ID Perceived_Emotion Congruence Arousal Valence
##          <dbl> <dbl>       <dbl>             <dbl>      <dbl>   <dbl>   <dbl>
## 1            1     1           1                 2          0       6       1
## 2            1     1           2                 4          1       6       6
## 3            1     1           3                 2          1       5       2
## 4            1     1           4                 2          0       6       1
## 5            1     1           5                 2          1       6       1
## 6            1     1           6                 2          1       9       1
## # i 1 more variable: Intended_Emotion <chr>
```

2 Seleção de melhores estímulos por TRI de 2pl

2.1 Felicidade

```
# --- Criar o Dataframe Largo Completo ---
all_responses_wide <- data_long %>%
  dplyr::select(Participants, Stimulus_ID, Congruence) %>%

  pivot_wider(
    names_from = Stimulus_ID,
    values_from = Congruence,
    names_prefix = "St_"
```

```
  )

# --- TRI para "Felicidade" usando o dataframe completo ---

# 1. Definir os nomes das colunas de "Felicidade" (Estímulos 1 a 29)
happiness_cols <- paste0("St_", 1:29)

# 2. Selecionar apenas essas colunas para a análise
happiness_response_matrix <- all_responses_wide %>%
  dplyr::select(all_of(happiness_cols))

# 3. Rodar o modelo TRI 2PL
model_happiness <- ltm(happiness_response_matrix ~ z1, IRT.param = TRUE)

# --- PASSO 3: Analisar e Exibir os Resultados ---

# Extrair e organizar os coeficientes
coef_happiness <- as.data.frame(coef(model_happiness))

results_happiness <- coef_happiness %>%
  rename(Difficulty = Dffclt, Discrimination = Dscrmn) %>%
  rownames_to_column(var = "Stimulus_ID") %>%
  mutate(Stimulus_ID = as.numeric(str_replace(Stimulus_ID, "St_", ""))) %>%
  arrange(desc(Discrimination)) # Ordena pelos itens mais discriminativos

# Mostrar os resultados
print("--- Parâmetros dos Itens para Felicidade (ordenado por Discriminação) ---")
```

```
## [1] "--- Parâmetros dos Itens para Felicidade (ordenado por Discriminação) ---"
```

```
print(results_happiness)
```

```
##    Stimulus_ID  Difficulty Discrimination
## 1            6  -1.4864795     1.97321170
## 2           13  -1.4390486     1.62780637
## 3           15  -0.7850875     1.61261607
## 4           14  -1.5444820     1.24666211
## 5           19  -2.3341774     1.17551433
## 6            3  -1.3953428     1.15476762
## 7           21  -1.7300940     1.11982663
## 8           18  -1.1370254     0.92507298
## 9           27  -2.3296849     0.91557787
## 10          20  -0.6732575     0.79282418
## 11          10  -2.5453331     0.77420340
## 12           5   0.2624575     0.77016712
## 13          24  -2.8618081     0.60743266
## 14           2   0.3400622     0.49758069
## 15           7  -4.8457126     0.44715743
## 16          22   1.7563206     0.43405679
## 17          25  -0.2454918     0.42648237
## 18           9  -1.6290668     0.42421642
## 19          26  -1.5767939     0.37717711
```

```
## 20           1  -1.3485105     0.37522992
## 21          12   4.0028668     0.25873010
## 22          28  -5.7378382     0.15266446
## 23          29  -6.8311064     0.12802599
## 24           8 -14.5629655     0.05498106
## 25          16  -1.8571480    -0.05392699
## 26          17  -3.1608488    -0.06996723
## 27           4   2.0265218    -0.07924950
## 28          11   0.5532685    -0.21970908
## 29          23   0.7854505    -0.51982463
```

2.2 Medo/Raiva

```
# 1. Definir os nomes das colunas de "Fear/Anger" (Estímulos 30 a 58)
fearanger_cols <- paste0("St_", 30:58)

# 2. Selecionar apenas essas colunas para a análise
fearanger_response_matrix <- all_responses_wide %>%
  dplyr::select(all_of(fearanger_cols))

# 3. Rodar o modelo TRI 2PL
model_fearanger <- ltm(fearanger_response_matrix ~ z1, IRT.param = TRUE)

# 4. Extrair, organizar e exibir os resultados
coef_fearanger <- as.data.frame(coef(model_fearanger))

results_fearanger <- coef_fearanger %>%
  rename(Difficulty = Dffclt, Discrimination = Dscrmn) %>%
  rownames_to_column(var = "Stimulus_ID") %>%
  mutate(Stimulus_ID = as.numeric(str_replace(Stimulus_ID, "St_", ""))) %>%
  arrange(desc(Discrimination))

# Mostrar os resultados
print("--- Parâmetros dos Itens para Fear/Anger (ordenado por Discriminação) ---")
```

```
## [1] "--- Parâmetros dos Itens para Fear/Anger (ordenado por Discriminação) ---"
```

```
print(results_fearanger)
```

```
##    Stimulus_ID   Difficulty Discrimination
## 1           35  -1.56588683     2.15400521
## 2           41  -0.93465868     2.06358014
## 3           43  -0.92178019     1.81796774
## 4           36  -1.34067678     1.72281479
## 5           31  -1.68259958     1.38036402
## 6           51  -1.31576162     1.10007384
## 7           45  -2.66841750     1.08082208
## 8           33  -1.08833360     1.01713834
## 9           42  -1.21795512     0.93457410
## 10          54  -0.79563718     0.87008335
## 11          40  -1.46732992     0.85908109
```

```
## 12          53  -1.99095159     0.77871578
## 13          57   2.96945612     0.75832293
## 14          52  -0.36638633     0.73804812
## 15          47   0.40914230     0.64502206
## 16          50  -0.07400229     0.57968455
## 17          44  -0.98770247     0.52678415
## 18          30   0.75490908     0.41756336
## 19          56  -2.49007737     0.31916801
## 20          48   0.07343733     0.28309333
## 21          34  -3.68136058     0.26750968
## 22          49  -3.48493764     0.23938082
## 23          55  -2.78270206     0.22502327
## 24          46  -4.71126480     0.14141340
## 25          39  -5.80304962     0.12639114
## 26          58 -61.20045989    -0.02265357
## 27          37  -3.55912805    -0.17513003
## 28          32   0.56727286    -0.17832202
## 29          38  -2.27871295    -0.20754759
```

2.3 Serenidade

Apenas 2 estímulos foram considerados adequados

```
# --- Análise TRI para "Serenity" ---

# 1. Definir os nomes das colunas de "Serenity" (Estímulos 59 a 87)
serenity_cols <- paste0("St_", 59:87)

# 2. Selecionar apenas essas colunas para a análise
serenity_response_matrix <- all_responses_wide %>%
  dplyr::select(all_of(serenity_cols))

# 3. Rodar o modelo TRI 2PL
model_serenity <- ltm(serenity_response_matrix ~ z1, IRT.param = TRUE)

# 4. Extrair, organizar e exibir os resultados
coef_serenity <- as.data.frame(coef(model_serenity))

results_serenity <- coef_serenity %>%
  rename(Difficulty = Dffclt, Discrimination = Dscrmn) %>%
  rownames_to_column(var = "Stimulus_ID") %>%
  mutate(Stimulus_ID = as.numeric(str_replace(Stimulus_ID, "St_", ""))) %>%
  arrange(desc(Discrimination))

# Mostrar os resultados
print("--- Parâmetros dos Itens para Serenity (ordenado por Discriminação) ---")
```

```
## [1] "--- Parâmetros dos Itens para Serenity (ordenado por Discriminação) ---"
```

```
print(results_serenity)
```

```
##    Stimulus_ID  Difficulty Discrimination
```

```
## 1           68  3.04412384     1.28158515
## 2           74  3.64134246     0.70753413
## 3           70  1.49298331     0.38321480
## 4           85  0.47192170     0.12748102
## 5           59 26.36841419     0.03773165
## 6           77  5.42669538    -0.12681753
## 7           80  3.66770092    -0.17604989
## 8           62  1.26554994    -0.25896380
## 9           69  1.83159391    -0.28408637
## 10          76  1.20495665    -0.30863042
## 11          87 -1.39730426    -0.36147804
## 12          79 -0.05041367    -0.42827091
## 13          82  0.28199824    -0.61157247
## 14          83  0.03238754    -0.62994503
## 15          64 -0.03203581    -0.75299896
## 16          75  0.66074822    -0.84784840
## 17          67  1.41082123    -0.98782894
## 18          63  1.23594914    -1.00603088
## 19          78  1.21954920    -1.06023142
## 20          65  1.33669676    -1.06438960
## 21          81  0.66063357    -1.11264415
## 22          60  1.88560001    -1.14800319
## 23          72  1.17869982    -1.15277908
## 24          73  1.46855066    -1.24461058
## 25          86  1.52892833    -1.29279953
## 26          71  1.52419927    -1.49457399
## 27          66  0.54691522    -1.51400027
## 28          61  1.14749538    -1.69011792
## 29          84  0.92307193    -1.82454908
```

2.4 Tristeza

```r
# 1. Definir os nomes das colunas de "Sadness" (Estímulos 88 a 116)
sadness_cols <- paste0("St_", 88:116)

# 2. Selecionar apenas essas colunas para a análise
sadness_response_matrix <- all_responses_wide %>%
  dplyr::select(all_of(sadness_cols))

# 3. Rodar o modelo TRI 2PL
model_sadness <- ltm(sadness_response_matrix ~ z1, IRT.param = TRUE)

# 4. Extrair, organizar e exibir os resultados
coef_sadness <- as.data.frame(coef(model_sadness))

results_sadness <- coef_sadness %>%
  rename(Difficulty = Dffclt, Discrimination = Dscrmn) %>%
  rownames_to_column(var = "Stimulus_ID") %>%
  mutate(Stimulus_ID = as.numeric(str_replace(Stimulus_ID, "St_", ""))) %>%
  arrange(desc(Discrimination))

# Mostrar os resultados
```

```r
print("--- Parâmetros dos Itens para Sadness (ordenado por Discriminação) ---")
```

```
## [1] "--- Parâmetros dos Itens para Sadness (ordenado por Discriminação) ---"
```

```r
print(results_sadness)
```

```
##    Stimulus_ID    Difficulty Discrimination
## 1           95  2.376989e-01    1.442444044
## 2          100 -1.396261e+00    1.189556904
## 3          101 -1.930251e+00    1.120197203
## 4          115 -5.929896e-01    1.101948023
## 5          111 -7.702453e-01    1.039857932
## 6           96 -1.155017e+00    0.868476174
## 7           90 -5.449360e-01    0.852128196
## 8          116  1.160149e+00    0.779811172
## 9           92  1.065258e+00    0.775811621
## 10         114 -1.438963e-01    0.763680048
## 11          88 -1.931341e+00    0.745000462
## 12         110 -1.870469e+00    0.674045911
## 13         108 -1.425989e-01    0.595477650
## 14         103  3.604645e-01    0.537683742
## 15          89  1.763832e+00    0.524438513
## 16         106  2.012291e-03    0.490563353
## 17          94 -1.216200e+00    0.418634802
## 18          97 -3.016340e+00    0.320394333
## 19          91 -2.230140e+00    0.282775967
## 20         112 -1.598377e+00    0.230597710
## 21         102  2.756249e-04    0.161907316
## 22         109 -1.007528e+01    0.059317895
## 23          93 -1.701081e+01    0.041648870
## 24         113 -9.101814e+01    0.008030253
## 25          99  4.680243e+01    0.002567058
## 26         105 -2.324536e+00   -0.034443556
## 27         107  1.352346e+00   -0.044403749
## 28          98  1.440165e+00   -0.212268187
## 29         104 -1.625883e+00   -0.488180532
```

3 Criar data frame com os melhores itens de cada categoria

```r
# Criando um dataframe com os melhores itens para usar em nosso código
best_items_df <- bind_rows(
  results_happiness %>% head(10) %>% mutate(Factor = "Happiness"),
  results_fearanger %>% head(10) %>% mutate(Factor = "FearAnger"),
  results_sadness %>% head(10) %>% mutate(Factor = "Sadness")
)

# Extrair apenas os IDs dos estímulos selecionados
selected_stimuli_ids <- best_items_df$Stimulus_ID

# Ver a lista final de 30 itens
print(selected_stimuli_ids)
```

```
##  [1]   6  13  15  14  19   3  21  18  27  20  35  41  43  36  31  51  45  33  42
## [20]  54  95 100 101 115 111  96  90 116  92 114
```

```
#### Passo 2: Preparar os Dados para a CFA
# Filtrar o dataframe largo para conter apenas as colunas dos melhores itens
cfa_data_final <- all_responses_wide %>%
  dplyr::select(all_of(paste0("St_", selected_stimuli_ids)))

# Declarar as variáveis como categóricas (ordenadas) para o lavaan usar o estimador WLSMV
cfa_data_final[] <- lapply(cfa_data_final, ordered)
```

4 Análise Fatorial Confirmatória das três categorias com estímulos adequados

4.1 Modelo com 3 dimensões

Já que os estimúlos de serenidade tem poucos itns com discriminação adequada

```
conflicts_prefer(dplyr::filter)
```

```
## [conflicted] Will prefer dplyr::filter over any other package.
```

```
# Extrair os IDs para cada fator para facilitar a escrita
ids_happy <- best_items_df %>% filter(Factor == "Happiness") %>% pull(Stimulus_ID)
ids_fear <- best_items_df %>% filter(Factor == "FearAnger") %>% pull(Stimulus_ID)
ids_sad <- best_items_df %>% filter(Factor == "Sadness") %>% pull(Stimulus_ID)

# Criar a string do modelo manualmente
cfa_model_final_string <- '
  Happiness =~ St_6 + St_13 + St_15 + St_14 + St_19 + St_3 + St_21
  + St_18 + St_27 + St_20
  FearAnger =~ St_35 + St_41 + St_43 + St_36 + St_31 + St_51 + St_45
  + St_33 + St_42 + St_54
  Sadness   =~ St_95 + St_100 + St_101 + St_115 + St_111 + St_96
  + St_90 + St_116 + St_92 + St_114

'

# Rodar a Análise Fatorial Confirmatória
fit_cfa_final <- cfa(
  model = cfa_model_final_string,
  data = cfa_data_final,
  estimator = "wlsmv",
  ordered = T
)

# --- ANÁLISE DOS RESULTADOS ---

# 1. Verificar os Índices de Ajuste do Modelo
print("--- Índices de Ajuste do Modelo ---")
```

```
## [1] "--- Índices de Ajuste do Modelo ---"
```

```
fitmeasures(fit_cfa_final, c("chisq", "df", "pvalue", "cfi", "tli", "rmsea", "srmr"))
```

```
##   chisq      df  pvalue     cfi     tli   rmsea    srmr
## 427.268 402.000   0.185   0.989   0.988   0.018   0.123
```

```
# 2. Ver o resumo completo com as cargas fatoriais padronizadas
print("--- Resumo Completo do Modelo ---")
```

```
## [1] "--- Resumo Completo do Modelo ---"
```

```
summary(fit_cfa_final, standardized = TRUE, fit.measures = TRUE)
```

```
## lavaan 0.6-18 ended normally after 36 iterations
##
##   Estimator                                         DWLS
##   Optimization method                           NLMINB
##   Number of model parameters                        63
##
##   Number of observations                           200
##
## Model Test User Model:
##                                               Standard      Scaled
##   Test Statistic                               427.268     479.472
##   Degrees of freedom                               402         402
##   P-value (Chi-square)                           0.185       0.005
##   Scaling correction factor                                  1.610
##   Shift parameter                                          214.041
##     simple second-order correction
##
## Model Test Baseline Model:
##
##   Test statistic                              2781.698    1281.194
##   Degrees of freedom                               435         435
##   P-value                                        0.000       0.000
##   Scaling correction factor                                  2.773
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.989       0.908
##   Tucker-Lewis Index (TLI)                       0.988       0.901
##
##   Robust Comparative Fit Index (CFI)                            NA
##   Robust Tucker-Lewis Index (TLI)                               NA
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.018       0.031
##   90 Percent confidence interval - lower         0.000       0.018
##   90 Percent confidence interval - upper         0.031       0.041
##   P-value H_0: RMSEA <= 0.050                    1.000       0.999
```

```
##   P-value H_0: RMSEA >= 0.080                    0.000       0.000
##
##   Robust RMSEA                                                  NA
##   90 Percent confidence interval - lower                        NA
##   90 Percent confidence interval - upper                        NA
##   P-value H_0: Robust RMSEA <= 0.050                            NA
##   P-value H_0: Robust RMSEA >= 0.080                            NA
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.123       0.123
##
## Parameter Estimates:
##
##   Parameterization                                Delta
##   Standard errors                             Robust.sem
##   Information                                   Expected
##   Information saturated (h1) model          Unstructured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   Happiness =~
##     St_6              1.000                               0.639    0.639
##     St_13             1.012    0.142    7.139    0.000    0.647    0.647
##     St_15             0.923    0.178    5.178    0.000    0.590    0.590
##     St_14             0.859    0.172    4.994    0.000    0.549    0.549
##     St_19             1.155    0.190    6.074    0.000    0.738    0.738
##     St_3              0.725    0.187    3.868    0.000    0.463    0.463
##     St_21             0.942    0.159    5.923    0.000    0.602    0.602
##     St_18             0.733    0.154    4.763    0.000    0.468    0.468
##     St_27             0.811    0.176    4.600    0.000    0.518    0.518
##     St_20             0.795    0.159    4.997    0.000    0.508    0.508
##   FearAnger =~
##     St_35             1.000                               0.788    0.788
##     St_41             0.995    0.128    7.794    0.000    0.784    0.784
##     St_43             0.848    0.132    6.419    0.000    0.668    0.668
##     St_36             0.881    0.123    7.136    0.000    0.694    0.694
##     St_31             0.777    0.137    5.693    0.000    0.613    0.613
##     St_51             0.788    0.122    6.459    0.000    0.621    0.621
##     St_45             0.815    0.178    4.575    0.000    0.642    0.642
##     St_33             0.620    0.126    4.919    0.000    0.488    0.488
##     St_42             0.594    0.121    4.897    0.000    0.468    0.468
##     St_54             0.536    0.129    4.166    0.000    0.422    0.422
##   Sadness =~
##     St_95             1.000                               0.462    0.462
##     St_100            1.468    0.321    4.571    0.000    0.678    0.678
##     St_101            1.296    0.304    4.268    0.000    0.599    0.599
##     St_115            1.279    0.315    4.064    0.000    0.591    0.591
##     St_111            0.830    0.225    3.696    0.000    0.384    0.384
##     St_96             1.006    0.267    3.772    0.000    0.465    0.465
##     St_90             1.290    0.299    4.308    0.000    0.596    0.596
##     St_116            0.661    0.216    3.052    0.002    0.305    0.305
##     St_92             0.818    0.211    3.881    0.000    0.378    0.378
##     St_114            0.944    0.257    3.670    0.000    0.436    0.436
```

```
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   Happiness ~~
##     FearAnger         0.437    0.098    4.461    0.000    0.869    0.869
##     Sadness           0.250    0.069    3.651    0.000    0.849    0.849
##   FearAnger ~~
##     Sadness           0.312    0.072    4.320    0.000    0.858    0.858
##
## Thresholds:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     St_6|t1          -1.126    0.113   -9.995    0.000   -1.126   -1.126
##     St_13|t1         -0.994    0.107   -9.311    0.000   -0.994   -0.994
##     St_15|t1         -0.539    0.094   -5.751    0.000   -0.539   -0.539
##     St_14|t1         -0.915    0.104   -8.819    0.000   -0.915   -0.915
##     St_19|t1         -1.311    0.123  -10.657    0.000   -1.311   -1.311
##     St_3|t1          -0.789    0.100   -7.918    0.000   -0.789   -0.789
##     St_21|t1         -0.954    0.105   -9.068    0.000   -0.954   -0.954
##     St_18|t1         -0.553    0.094   -5.889    0.000   -0.553   -0.553
##     St_27|t1         -1.103    0.112   -9.887    0.000   -1.103   -1.103
##     St_20|t1         -0.292    0.090   -3.240    0.001   -0.292   -0.292
##     St_35|t1         -1.227    0.118  -10.398    0.000   -1.227   -1.227
##     St_41|t1         -0.722    0.098   -7.387    0.000   -0.722   -0.722
##     St_43|t1         -0.674    0.097   -6.983    0.000   -0.674   -0.674
##     St_36|t1         -0.954    0.105   -9.068    0.000   -0.954   -0.954
##     St_31|t1         -1.058    0.110   -9.662    0.000   -1.058   -1.058
##     St_51|t1         -0.722    0.098   -7.387    0.000   -0.722   -0.722
##     St_45|t1         -1.405    0.129  -10.862    0.000   -1.405   -1.405
##     St_33|t1         -0.568    0.094   -6.027    0.000   -0.568   -0.568
##     St_42|t1         -0.598    0.095   -6.301    0.000   -0.598   -0.598
##     St_54|t1         -0.372    0.091   -4.081    0.000   -0.372   -0.372
##     St_95|t1          0.151    0.089    1.692    0.091    0.151    0.151
##     St_100|t1        -0.806    0.100   -8.049    0.000   -0.806   -0.806
##     St_101|t1        -1.058    0.110   -9.662    0.000   -1.058   -1.058
##     St_115|t1        -0.332    0.091   -3.661    0.000   -0.332   -0.332
##     St_111|t1        -0.412    0.092   -4.501    0.000   -0.412   -0.412
##     St_96|t1         -0.539    0.094   -5.751    0.000   -0.539   -0.539
##     St_90|t1         -0.253    0.090   -2.818    0.005   -0.253   -0.253
##     St_116|t1         0.496    0.093    5.336    0.000    0.496    0.496
##     St_92|t1          0.454    0.092    4.919    0.000    0.454    0.454
##     St_114|t1        -0.063    0.089   -0.705    0.481   -0.063   -0.063
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .St_6              0.592                               0.592    0.592
##    .St_13             0.582                               0.582    0.582
##    .St_15             0.652                               0.652    0.652
##    .St_14             0.699                               0.699    0.699
##    .St_19             0.455                               0.455    0.455
##    .St_3              0.786                               0.786    0.786
##    .St_21             0.638                               0.638    0.638
##    .St_18             0.781                               0.781    0.781
##    .St_27             0.732                               0.732    0.732
##    .St_20             0.742                               0.742    0.742
```

```
##    .St_35             0.379                                  0.379    0.379
##    .St_41             0.385                                  0.385    0.385
##    .St_43             0.554                                  0.554    0.554
##    .St_36             0.518                                  0.518    0.518
##    .St_31             0.625                                  0.625    0.625
##    .St_51             0.614                                  0.614    0.614
##    .St_45             0.588                                  0.588    0.588
##    .St_33             0.762                                  0.762    0.762
##    .St_42             0.781                                  0.781    0.781
##    .St_54             0.822                                  0.822    0.822
##    .St_95             0.787                                  0.787    0.787
##    .St_100            0.540                                  0.540    0.540
##    .St_101            0.641                                  0.641    0.641
##    .St_115            0.651                                  0.651    0.651
##    .St_111            0.853                                  0.853    0.853
##    .St_96             0.784                                  0.784    0.784
##    .St_90             0.645                                  0.645    0.645
##    .St_116            0.907                                  0.907    0.907
##    .St_92             0.857                                  0.857    0.857
##    .St_114            0.810                                  0.810    0.810
##     Happiness         0.408    0.122    3.347    0.001       1.000    1.000
##     FearAnger         0.621    0.145    4.272    0.000       1.000    1.000
##     Sadness           0.213    0.079    2.685    0.007       1.000    1.000
```

```r
# 3. Calcular a Fidedignidade (Confiabilidade Composta - Ômega)
print("--- Fidedignidade (Confiabilidade Composta) ---")
```

```
## [1] "--- Fidedignidade (Confiabilidade Composta) ---"
```

```r
compRelSEM(fit_cfa_final)
```

```
## Happiness FearAnger   Sadness
##     0.691     0.745     0.604
```

4.2 Modelo Unidimensional

Como os três fatores apresentam alta correlação e confiabilidade baixa envestigou-se o modelo unifatorial

A estrutura unidimensional se constitui de um fator em que todos os itens carregam de forma positiva consistentemente com os dados semânticos e da PANAS.

```r
cfa_GFactor <- '
  # Fator Geral (G)
  G =~ St_6 + St_13 + St_15 + St_14 + St_19 + St_3 + St_21 + St_18 + St_27
  + St_20 + St_35 + St_41 + St_43 + St_36 + St_31 + St_51 + St_45 + St_33
  + St_42 + St_54 + St_95 + St_100 + St_101 + St_115 + St_111 + St_96 + St_90
  + St_116 + St_92 + St_114
'

# Rodar a CFA Bifatorial
fit_GFactorr <- cfa(
  model = cfa_GFactor,
```

```r
  data = cfa_data_final,
  estimator = "wlsmv",
  ordered = TRUE,
  orthogonal = TRUE  # Este argumento é fundamental para o modelo bifatorial
)

summary(fit_GFactorr, standardized = TRUE, fit.measures = TRUE)
```

```
## lavaan 0.6-18 ended normally after 27 iterations
##
##   Estimator                                         DWLS
##   Optimization method                           NLMINB
##   Number of model parameters                        60
##
##   Number of observations                           200
##
## Model Test User Model:
##                                              Standard      Scaled
##   Test Statistic                              439.055     487.390
##   Degrees of freedom                              405         405
##   P-value (Chi-square)                          0.117       0.003
##   Scaling correction factor                                 1.618
##   Shift parameter                                         216.068
##     simple second-order correction
##
## Model Test Baseline Model:
##
##   Test statistic                             2781.698    1281.194
##   Degrees of freedom                              435         435
##   P-value                                       0.000       0.000
##   Scaling correction factor                                 2.773
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                   0.985       0.903
##   Tucker-Lewis Index (TLI)                      0.984       0.895
##
##   Robust Comparative Fit Index (CFI)                           NA
##   Robust Tucker-Lewis Index (TLI)                              NA
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                         0.021       0.032
##   90 Percent confidence interval - lower        0.000       0.020
##   90 Percent confidence interval - upper        0.033       0.042
##   P-value H_0: RMSEA <= 0.050                   1.000       0.999
##   P-value H_0: RMSEA >= 0.080                   0.000       0.000
##
##   Robust RMSEA                                                 NA
##   90 Percent confidence interval - lower                       NA
##   90 Percent confidence interval - upper                       NA
##   P-value H_0: Robust RMSEA <= 0.050                           NA
##   P-value H_0: Robust RMSEA >= 0.080                           NA
```

```
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.124       0.124
##
## Parameter Estimates:
##
##   Parameterization                               Delta
##   Standard errors                             Robust.sem
##   Information                                   Expected
##   Information saturated (h1) model          Unstructured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   G =~
##     St_6              1.000                               0.603    0.603
##     St_13             1.016    0.146    6.979    0.000    0.612    0.612
##     St_15             0.926    0.181    5.115    0.000    0.558    0.558
##     St_14             0.858    0.175    4.893    0.000    0.517    0.517
##     St_19             1.163    0.194    5.994    0.000    0.701    0.701
##     St_3              0.726    0.190    3.815    0.000    0.438    0.438
##     St_21             0.948    0.162    5.838    0.000    0.572    0.572
##     St_18             0.737    0.157    4.701    0.000    0.444    0.444
##     St_27             0.820    0.179    4.570    0.000    0.494    0.494
##     St_20             0.805    0.161    4.998    0.000    0.486    0.486
##     St_35             1.262    0.225    5.600    0.000    0.761    0.761
##     St_41             1.253    0.209    6.005    0.000    0.756    0.756
##     St_43             1.067    0.199    5.358    0.000    0.643    0.643
##     St_36             1.113    0.177    6.280    0.000    0.671    0.671
##     St_31             0.982    0.210    4.674    0.000    0.592    0.592
##     St_51             0.998    0.165    6.040    0.000    0.602    0.602
##     St_45             1.033    0.227    4.547    0.000    0.623    0.623
##     St_33             0.776    0.195    3.978    0.000    0.468    0.468
##     St_42             0.751    0.194    3.870    0.000    0.453    0.453
##     St_54             0.677    0.184    3.672    0.000    0.408    0.408
##     St_95             0.710    0.158    4.504    0.000    0.428    0.428
##     St_100            1.053    0.192    5.473    0.000    0.635    0.635
##     St_101            0.932    0.212    4.399    0.000    0.562    0.562
##     St_115            0.917    0.214    4.280    0.000    0.553    0.553
##     St_111            0.588    0.158    3.724    0.000    0.354    0.354
##     St_96             0.718    0.173    4.150    0.000    0.433    0.433
##     St_90             0.930    0.181    5.144    0.000    0.561    0.561
##     St_116            0.468    0.150    3.119    0.002    0.282    0.282
##     St_92             0.591    0.161    3.665    0.000    0.356    0.356
##     St_114            0.681    0.165    4.134    0.000    0.411    0.411
##
## Thresholds:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     St_6|t1          -1.126    0.113   -9.995    0.000   -1.126   -1.126
##     St_13|t1         -0.994    0.107   -9.311    0.000   -0.994   -0.994
##     St_15|t1         -0.539    0.094   -5.751    0.000   -0.539   -0.539
##     St_14|t1         -0.915    0.104   -8.819    0.000   -0.915   -0.915
##     St_19|t1         -1.311    0.123  -10.657    0.000   -1.311   -1.311
##     St_3|t1          -0.789    0.100   -7.918    0.000   -0.789   -0.789
```

```
##     St_21|t1         -0.954    0.105   -9.068    0.000   -0.954   -0.954
##     St_18|t1         -0.553    0.094   -5.889    0.000   -0.553   -0.553
##     St_27|t1         -1.103    0.112   -9.887    0.000   -1.103   -1.103
##     St_20|t1         -0.292    0.090   -3.240    0.001   -0.292   -0.292
##     St_35|t1         -1.227    0.118  -10.398    0.000   -1.227   -1.227
##     St_41|t1         -0.722    0.098   -7.387    0.000   -0.722   -0.722
##     St_43|t1         -0.674    0.097   -6.983    0.000   -0.674   -0.674
##     St_36|t1         -0.954    0.105   -9.068    0.000   -0.954   -0.954
##     St_31|t1         -1.058    0.110   -9.662    0.000   -1.058   -1.058
##     St_51|t1         -0.722    0.098   -7.387    0.000   -0.722   -0.722
##     St_45|t1         -1.405    0.129  -10.862    0.000   -1.405   -1.405
##     St_33|t1         -0.568    0.094   -6.027    0.000   -0.568   -0.568
##     St_42|t1         -0.598    0.095   -6.301    0.000   -0.598   -0.598
##     St_54|t1         -0.372    0.091   -4.081    0.000   -0.372   -0.372
##     St_95|t1          0.151    0.089    1.692    0.091    0.151    0.151
##     St_100|t1        -0.806    0.100   -8.049    0.000   -0.806   -0.806
##     St_101|t1        -1.058    0.110   -9.662    0.000   -1.058   -1.058
##     St_115|t1        -0.332    0.091   -3.661    0.000   -0.332   -0.332
##     St_111|t1        -0.412    0.092   -4.501    0.000   -0.412   -0.412
##     St_96|t1         -0.539    0.094   -5.751    0.000   -0.539   -0.539
##     St_90|t1         -0.253    0.090   -2.818    0.005   -0.253   -0.253
##     St_116|t1         0.496    0.093    5.336    0.000    0.496    0.496
##     St_92|t1          0.454    0.092    4.919    0.000    0.454    0.454
##     St_114|t1        -0.063    0.089   -0.705    0.481   -0.063   -0.063
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .St_6              0.636                               0.636    0.636
##    .St_13             0.625                               0.625    0.625
##    .St_15             0.688                               0.688    0.688
##    .St_14             0.733                               0.733    0.733
##    .St_19             0.508                               0.508    0.508
##    .St_3              0.808                               0.808    0.808
##    .St_21             0.673                               0.673    0.673
##    .St_18             0.803                               0.803    0.803
##    .St_27             0.755                               0.755    0.755
##    .St_20             0.764                               0.764    0.764
##    .St_35             0.421                               0.421    0.421
##    .St_41             0.429                               0.429    0.429
##    .St_43             0.586                               0.586    0.586
##    .St_36             0.550                               0.550    0.550
##    .St_31             0.650                               0.650    0.650
##    .St_51             0.638                               0.638    0.638
##    .St_45             0.612                               0.612    0.612
##    .St_33             0.781                               0.781    0.781
##    .St_42             0.795                               0.795    0.795
##    .St_54             0.833                               0.833    0.833
##    .St_95             0.817                               0.817    0.817
##    .St_100            0.597                               0.597    0.597
##    .St_101            0.684                               0.684    0.684
##    .St_115            0.694                               0.694    0.694
##    .St_111            0.874                               0.874    0.874
##    .St_96             0.813                               0.813    0.813
##    .St_90             0.686                               0.686    0.686
```

```
##    .St_116           0.920                                 0.920    0.920
##    .St_92            0.873                                 0.873    0.873
##    .St_114           0.831                                 0.831    0.831
##     G                0.364    0.115    3.168    0.002       1.000    1.000
```

```r
fitmeasures(fit_GFactorr, c("chisq", "df", "pvalue", "cfi", "tli", "rmsea", "srmr"))
```

```
##   chisq      df  pvalue     cfi     tli   rmsea    srmr
## 439.055 405.000   0.117   0.985   0.984   0.021   0.124
```

```r
semTools::compRelSEM(fit_GFactorr)
```

```
##     G
## 0.864
```

4.3 Modelo Bifatorial

A estrutura bifatorial foi testada mas o modelo "quebra" porque o Fator Geral (G) "suga" quase toda a variância, deixando os fatores específicos (Happiness, Fear, Sadness) com variância zero ou negativa (os chamados Heywood Cases). É por isso que os Omegas específicos deram quase zero (0.064, 0.006).

```r
cfa_bifactor_string <- '
  # Fator Geral (G)
  G =~ St_6 + St_13 + St_15 + St_14 + St_19 + St_3 + St_21 + St_18 + St_27
  + St_20 + St_35 + St_41 + St_43 + St_36 + St_31 + St_51 + St_45 + St_33
  + St_42 + St_54 + St_95 + St_100 + St_101 + St_115 + St_111 + St_96 + St_90
  + St_116 + St_92 + St_114

  # Fatores de Grupo Específicos
  Happiness =~ St_6 + St_13 + St_15 + St_14 + St_19 + St_3 + St_21 + St_18
  + St_27 + St_20
  FearAnger =~ St_35 + St_41 + St_43 + St_36 + St_31 + St_51 + St_45 + St_33
  + St_42 + St_54
  Sadness   =~ St_95 + St_100 + St_101 + St_115 + St_111 + St_96 + St_90
  + St_116 + St_92 + St_114
'

# Rodar a CFA Bifatorial
fit_cfa_bifactor <- cfa(
  model = cfa_bifactor_string,
  data = cfa_data_final,
  estimator = "wlsmv",
  ordered = TRUE,
  orthogonal = TRUE
)
```

```
## Warning: lavaan->lav_lavaan_step11_estoptim():
##    Model estimation FAILED! Returning starting values.
```

O modelo não convergiu, provavelmente por ser muito "dispendioso" para o tanto de dados que temos

```r
compRelSEM(fit_cfa_bifactor)
```

```
##         G Happiness FearAnger   Sadness
##     0.832     0.064     0.006     0.159
```

4.4 Fator de Segunda Ordem

Rouba toda a variância dos fatores de primeira

```r
# Definição do Modelo de Segunda Ordem
cfa_second_order_string <- '

  Happiness =~ St_6 + St_13 + St_15 + St_14 + St_19 + St_3 + St_21
  + St_18 + St_27 + St_20
  FearAnger =~ St_35 + St_41 + St_43 + St_36 + St_31 + St_51 + St_45
  + St_33 + St_42 + St_54
  Sadness   =~ St_95 + St_100 + St_101 + St_115 + St_111 + St_96
  + St_90 + St_116 + St_92 + St_114

  # Fator de Segunda Ordem
  G_Ability =~ Happiness + FearAnger + Sadness
'

# Rodar a Análise
fit_second_order <- cfa(
  model = cfa_second_order_string,
  data = cfa_data_final,
  estimator = "wlsmv",
  ordered = TRUE
)

# Ver resultados
summary(fit_second_order, standardized = TRUE, fit.measures = TRUE)
```

```
## lavaan 0.6-18 ended normally after 41 iterations
##
##   Estimator                                         DWLS
##   Optimization method                           NLMINB
##   Number of model parameters                        63
##
##   Number of observations                           200
##
## Model Test User Model:
##                                              Standard      Scaled
##   Test Statistic                               427.268     479.472
##   Degrees of freedom                               402         402
##   P-value (Chi-square)                           0.185       0.005
##   Scaling correction factor                                  1.610
##   Shift parameter                                          214.041
##     simple second-order correction
##
## Model Test Baseline Model:
```

```
##
##   Test statistic                                2781.698    1281.194
##   Degrees of freedom                                435         435
##   P-value                                         0.000       0.000
##   Scaling correction factor                                   2.773
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                     0.989       0.908
##   Tucker-Lewis Index (TLI)                        0.988       0.901
##
##   Robust Comparative Fit Index (CFI)                             NA
##   Robust Tucker-Lewis Index (TLI)                                NA
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                           0.018       0.031
##   90 Percent confidence interval - lower          0.000       0.018
##   90 Percent confidence interval - upper          0.031       0.041
##   P-value H_0: RMSEA <= 0.050                     1.000       0.999
##   P-value H_0: RMSEA >= 0.080                     0.000       0.000
##
##   Robust RMSEA                                                   NA
##   90 Percent confidence interval - lower                         NA
##   90 Percent confidence interval - upper                         NA
##   P-value H_0: Robust RMSEA <= 0.050                             NA
##   P-value H_0: Robust RMSEA >= 0.080                             NA
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                            0.123       0.123
##
## Parameter Estimates:
##
##   Parameterization                                Delta
##   Standard errors                             Robust.sem
##   Information                                   Expected
##   Information saturated (h1) model          Unstructured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   Happiness =~
##     St_6              1.000                               0.639    0.639
##     St_13             1.012    0.142    7.139    0.000    0.647    0.647
##     St_15             0.923    0.178    5.178    0.000    0.590    0.590
##     St_14             0.859    0.172    4.994    0.000    0.549    0.549
##     St_19             1.155    0.190    6.074    0.000    0.738    0.738
##     St_3              0.725    0.187    3.868    0.000    0.463    0.463
##     St_21             0.942    0.159    5.923    0.000    0.602    0.602
##     St_18             0.733    0.154    4.763    0.000    0.468    0.468
##     St_27             0.811    0.176    4.600    0.000    0.518    0.518
##     St_20             0.795    0.159    4.997    0.000    0.508    0.508
##   FearAnger =~
##     St_35             1.000                               0.788    0.788
```

```
##     St_41             0.995    0.128    7.794    0.000    0.784    0.784
##     St_43             0.848    0.132    6.419    0.000    0.668    0.668
##     St_36             0.881    0.123    7.136    0.000    0.694    0.694
##     St_31             0.777    0.137    5.693    0.000    0.613    0.613
##     St_51             0.788    0.122    6.459    0.000    0.621    0.621
##     St_45             0.815    0.178    4.575    0.000    0.642    0.642
##     St_33             0.620    0.126    4.919    0.000    0.488    0.488
##     St_42             0.594    0.121    4.897    0.000    0.468    0.468
##     St_54             0.536    0.129    4.166    0.000    0.422    0.422
##   Sadness =~
##     St_95             1.000                               0.462    0.462
##     St_100            1.468    0.321    4.571    0.000    0.678    0.678
##     St_101            1.296    0.304    4.268    0.000    0.599    0.599
##     St_115            1.279    0.315    4.064    0.000    0.591    0.591
##     St_111            0.830    0.225    3.696    0.000    0.384    0.384
##     St_96             1.006    0.267    3.772    0.000    0.465    0.465
##     St_90             1.290    0.299    4.308    0.000    0.596    0.596
##     St_116            0.661    0.216    3.052    0.002    0.305    0.305
##     St_92             0.818    0.211    3.881    0.000    0.378    0.378
##     St_114            0.944    0.257    3.670    0.000    0.436    0.436
##   G_Ability =~
##     Happiness         1.000                               0.927    0.927
##     FearAnger         1.247    0.245    5.093    0.000    0.937    0.937
##     Sadness           0.714    0.165    4.339    0.000    0.915    0.915
##
## Thresholds:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     St_6|t1         -1.126    0.113   -9.995    0.000   -1.126   -1.126
##     St_13|t1        -0.994    0.107   -9.311    0.000   -0.994   -0.994
##     St_15|t1        -0.539    0.094   -5.751    0.000   -0.539   -0.539
##     St_14|t1        -0.915    0.104   -8.819    0.000   -0.915   -0.915
##     St_19|t1        -1.311    0.123  -10.657    0.000   -1.311   -1.311
##     St_3|t1         -0.789    0.100   -7.918    0.000   -0.789   -0.789
##     St_21|t1        -0.954    0.105   -9.068    0.000   -0.954   -0.954
##     St_18|t1        -0.553    0.094   -5.889    0.000   -0.553   -0.553
##     St_27|t1        -1.103    0.112   -9.887    0.000   -1.103   -1.103
##     St_20|t1        -0.292    0.090   -3.240    0.001   -0.292   -0.292
##     St_35|t1        -1.227    0.118  -10.398    0.000   -1.227   -1.227
##     St_41|t1        -0.722    0.098   -7.387    0.000   -0.722   -0.722
##     St_43|t1        -0.674    0.097   -6.983    0.000   -0.674   -0.674
##     St_36|t1        -0.954    0.105   -9.068    0.000   -0.954   -0.954
##     St_31|t1        -1.058    0.110   -9.662    0.000   -1.058   -1.058
##     St_51|t1        -0.722    0.098   -7.387    0.000   -0.722   -0.722
##     St_45|t1        -1.405    0.129  -10.862    0.000   -1.405   -1.405
##     St_33|t1        -0.568    0.094   -6.027    0.000   -0.568   -0.568
##     St_42|t1        -0.598    0.095   -6.301    0.000   -0.598   -0.598
##     St_54|t1        -0.372    0.091   -4.081    0.000   -0.372   -0.372
##     St_95|t1         0.151    0.089    1.692    0.091    0.151    0.151
##     St_100|t1       -0.806    0.100   -8.049    0.000   -0.806   -0.806
##     St_101|t1       -1.058    0.110   -9.662    0.000   -1.058   -1.058
##     St_115|t1       -0.332    0.091   -3.661    0.000   -0.332   -0.332
##     St_111|t1       -0.412    0.092   -4.501    0.000   -0.412   -0.412
##     St_96|t1        -0.539    0.094   -5.751    0.000   -0.539   -0.539
##     St_90|t1        -0.253    0.090   -2.818    0.005   -0.253   -0.253
```

```
##     St_116|t1         0.496    0.093    5.336    0.000    0.496    0.496
##     St_92|t1          0.454    0.092    4.919    0.000    0.454    0.454
##     St_114|t1        -0.063    0.089   -0.705    0.481   -0.063   -0.063
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .St_6              0.592                               0.592    0.592
##    .St_13             0.582                               0.582    0.582
##    .St_15             0.652                               0.652    0.652
##    .St_14             0.699                               0.699    0.699
##    .St_19             0.455                               0.455    0.455
##    .St_3              0.786                               0.786    0.786
##    .St_21             0.638                               0.638    0.638
##    .St_18             0.781                               0.781    0.781
##    .St_27             0.732                               0.732    0.732
##    .St_20             0.742                               0.742    0.742
##    .St_35             0.379                               0.379    0.379
##    .St_41             0.385                               0.385    0.385
##    .St_43             0.554                               0.554    0.554
##    .St_36             0.518                               0.518    0.518
##    .St_31             0.625                               0.625    0.625
##    .St_51             0.614                               0.614    0.614
##    .St_45             0.588                               0.588    0.588
##    .St_33             0.762                               0.762    0.762
##    .St_42             0.781                               0.781    0.781
##    .St_54             0.822                               0.822    0.822
##    .St_95             0.787                               0.787    0.787
##    .St_100            0.540                               0.540    0.540
##    .St_101            0.642                               0.642    0.642
##    .St_115            0.651                               0.651    0.651
##    .St_111            0.853                               0.853    0.853
##    .St_96             0.784                               0.784    0.784
##    .St_90             0.645                               0.645    0.645
##    .St_116            0.907                               0.907    0.907
##    .St_92             0.857                               0.857    0.857
##    .St_114            0.810                               0.810    0.810
##    .Happiness         0.057    0.044    1.300    0.194    0.141    0.141
##    .FearAnger         0.076    0.063    1.201    0.230    0.122    0.122
##    .Sadness           0.035    0.025    1.400    0.162    0.162    0.162
##     G_Ability         0.351    0.115    3.045    0.002    1.000    1.000
```

```
fitmeasures(fit_second_order, c("chisq", "df", "pvalue", "cfi", "tli",
                                "rmsea", "srmr"))
```

```
##   chisq      df  pvalue     cfi     tli   rmsea    srmr
## 427.268 402.000   0.185   0.989   0.988   0.018   0.123
```

```
# Calcular confiabilidade (Omega) para o modelo de segunda ordem
compRelSEM(fit_second_order)
```

```
## Happiness FearAnger   Sadness
##     0.691     0.745     0.604
```

4.5 Aqui considerar só a confiabilidade do Fator Geral

A confiabilidade composta foi calculada através do coeficiente Ômega de McDonald, considerando a natureza ordinal dos dados e a estrutura de segunda ordem. O Fator Geral (G_Ability) apresentou um índice de confiabilidade excelente (= 0.923), demonstrando alta precisão na mensuração da habilidade global de reconhecimento emocional.

Os fatores de primeira ordem também apresentaram índices adequados (Felicidade: 0.885; Medo/Raiva: 0.918; Tristeza: 0.757). Contudo, dada a elevada carga do fator geral sobre estes domínios específicos, esses índices refletem, em grande parte, a confiabilidade da variância comum (G) compartilhada pelos itens.

```
omega_G <- compRelSEM(fit_second_order, higher = "G_Ability", ord.scale = FALSE)
print(omega_G)
```

```
## Happiness FearAnger   Sadness G_Ability
##     0.885     0.918     0.757     0.923
```

5 Correlação com BDI e BAI

##Extração dos escores fatoriais do fator de segunda ordem

```
# Extrair os escores do modelo de segunda ordem
fator_scores <- lavPredict(fit_second_order)

# Vamos ver o que ele gerou (deve ter Happiness, FearAnger, Sadness e G_Ability)
head(fator_scores)
```

```
##       Happiness   FearAnger     Sadness   G_Ability
## [1,]  0.3202796  0.24999054  0.33235856  0.29813719
## [2,]  0.7571243  1.03572786  0.67675823  0.79598642
## [3,]  0.5265121  0.47999902  0.38090877  0.44899681
## [4,] -0.1236982 -0.66342665  0.04321414 -0.21920733
## [5,] -0.2616402 -0.05918781  0.06718569 -0.07457912
## [6,] -0.5375110 -0.56319338 -0.18294817 -0.40343230
```

```
# Transformar em um dataframe para facilitar
fator_scores_df <- as.data.frame(fator_scores)

# O que nos interessa é apenas a coluna "G_Ability"
# Vamos adicionar essa coluna ao seu dataframe original 'data'
# IMPORTANTE: Certifique-se de que a ordem das linhas não mudou (se não houve exclusão de dados)
data$G_Score <- fator_scores_df$G_Ability

# Verifique se deu certo
head(data %>% select(Participants, BAI, BDI, G_Score))
```

```
## # A tibble: 6 x 4
##   Participants   BAI   BDI G_Score
##          <dbl> <dbl> <dbl>   <dbl>
## 1            1     3     4  0.298
## 2            2     2     1  0.796
```

```
## 3            3    29    10  0.449
## 4            4     6     6 -0.219
## 5            5     5     7 -0.0746
## 6            6     5     8 -0.403
```

5.1 Correlações

A base de dados só tem os escores da soma de cada instrumento.

```r
# --- Correlação com BAI (Ansiedade) ---
cor_bai <- cor.test(data$G_Score, data$BAI, method = "spearman")
```

```
## Warning in cor.test.default(data$G_Score, data$BAI, method = "spearman"):
## Cannot compute exact p-value with ties
```

```r
print("--- Correlação entre Fator G e BAI (Spearman) ---")
```

```
## [1] "--- Correlação entre Fator G e BAI (Spearman) ---"
```

```r
print(cor_bai)
```

```
##
##  Spearman's rank correlation rho
##
## data:  data$G_Score and data$BAI
## S = 1274144, p-value = 0.5327
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##        rho
## 0.04436792
```

```r
# --- Correlação com BDI (Depressão) ---
cor_bdi <- cor.test(data$G_Score, data$BDI, method = "spearman")
```

```
## Warning in cor.test.default(data$G_Score, data$BDI, method = "spearman"):
## Cannot compute exact p-value with ties
```

```r
print("--- Correlação entre Fator G e BDI (Spearman) ---")
```

```
## [1] "--- Correlação entre Fator G e BDI (Spearman) ---"
```

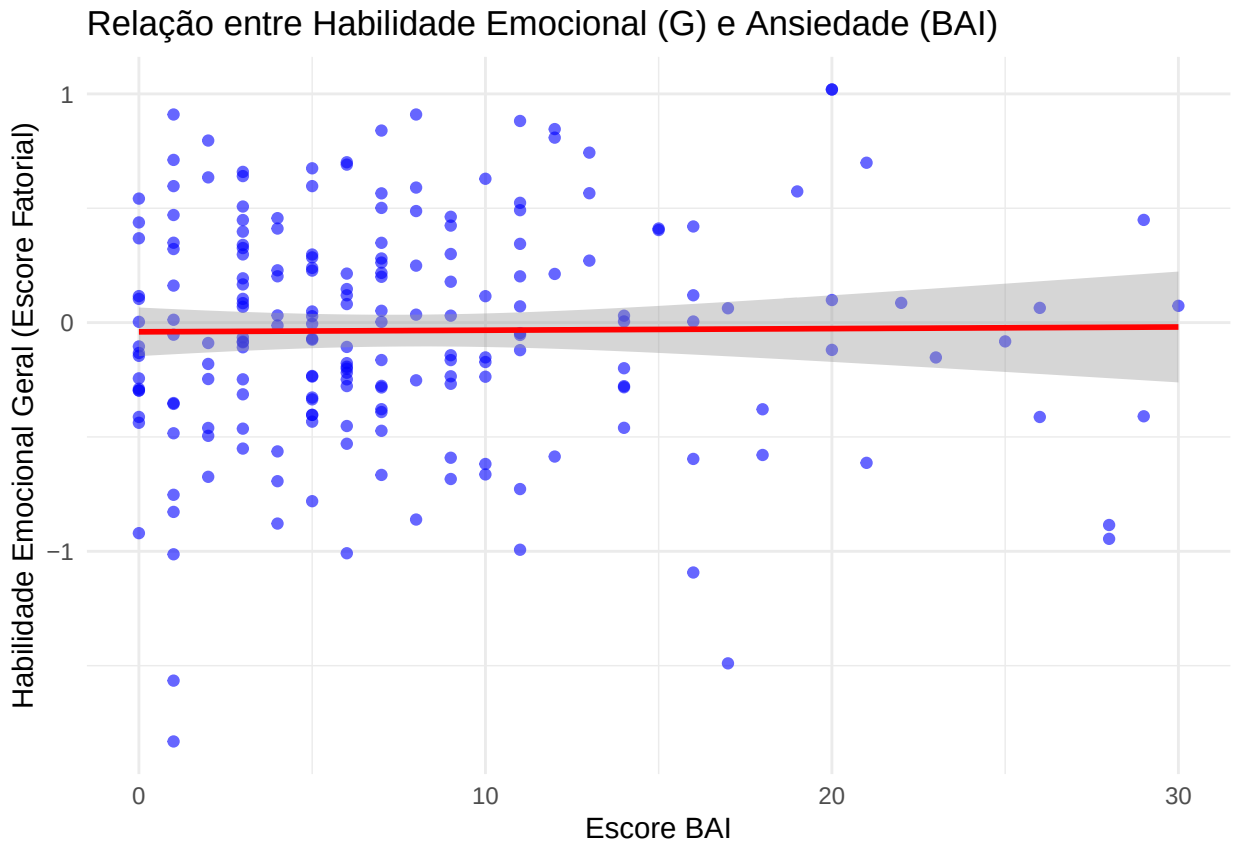
```r
print(cor_bdi)
```

```
##
##  Spearman's rank correlation rho
##
## data:  data$G_Score and data$BDI
## S = 1306453, p-value = 0.7772
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##        rho
## 0.02013555
```

5.1.1 Visualização

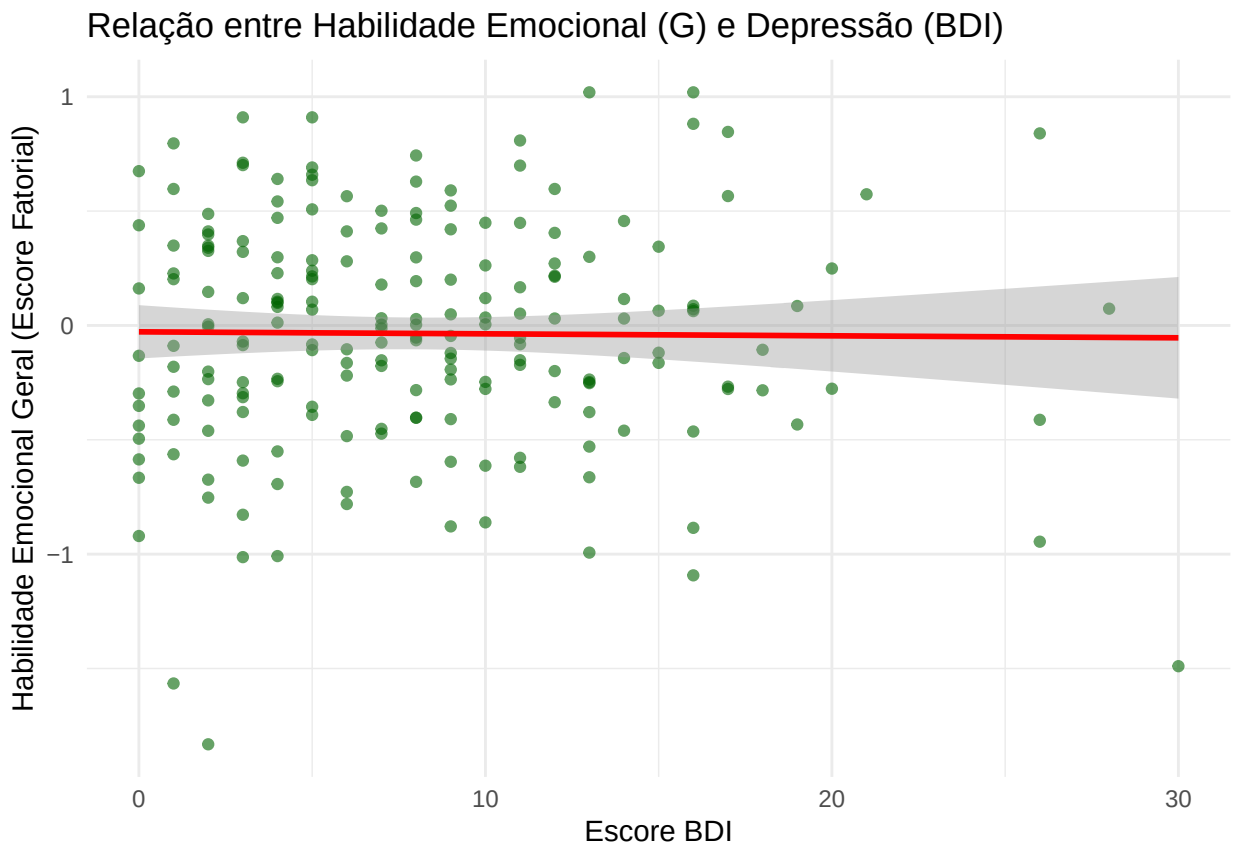
```
library(ggplot2)

# Gráfico G-Factor vs BAI
ggplot(data, aes(x = BAI, y = G_Score)) +
  geom_point(alpha = 0.6, color = "blue") +
  geom_smooth(method = "lm", color = "red") +
  labs(title = "Relação entre Habilidade Emocional (G) e Ansiedade (BAI)",
       x = "Escore BAI", y = "Habilidade Emocional Geral (Escore Fatorial)") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# Gráfico G-Factor vs BDI
ggplot(data, aes(x = BDI, y = G_Score)) +
  geom_point(alpha = 0.6, color = "darkgreen") +
  geom_smooth(method = "lm", color = "red") +
  labs(title = "Relação entre Habilidade Emocional (G) e Depressão (BDI)",
       x = "Escore BDI", y = "Habilidade Emocional Geral (Escore Fatorial)") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



6 Modelos mistos

6.1 Arrumar o data

```
# 1. Recuperar a lista dos 30 melhores itens
# selected_stimuli_ids <- c(6, 13, 15, ..., 114) # Os IDs que o TRI selecionou

# 2. Filtrar o data_long para ter apenas esses itens
# E garantir que as colunas BAI e BDI estão lá
data_mixed <- data_long %>%
  filter(Stimulus_ID %in% selected_stimuli_ids) %>%
  # Garantir que a variável de resposta é numérica (0 ou 1)
  mutate(Congruence = as.numeric(Congruence)) %>%
  # Juntar com os dados clínicos (se já não estiverem no data_long)
  left_join(data %>% select(Participants, BAI, BDI), by = "Participants")

# Verifique se deu certo
names(data_mixed)
```

```
##  [1] "Participants"      "Group"             "Stimulus_ID"
##  [4] "Perceived_Emotion" "Congruence"        "Arousal"
##  [7] "Valence"           "Intended_Emotion"  "BAI"
## [10] "BDI"
```

6.2 Modelo misto para ansiedade

A chance de acerto para Medo/Raiva (intercepto) é alta e significativa. Os participantes são bons em identificar Medo e Raiva.

Sadness (-1.32, $p < .001$): o coeficiente é negativo e altamente significativo, além do Odds Ratio ser 0.266. É muito mais difícil reconhecer Tristeza do que Medo/Raiva (a chance de acerto cai para cerca de 26% da chance original).

Existe uma tendência muito leve (não significativa estatisticamente a 5%, mas < 10%) de que, conforme a ansiedade aumenta, o reconhecimento de tristeza melhora ligeiramente (ou a dificuldade diminui).

Habilidade Emocional (Fator G) é independente do estado clínico atual do paciente. Ou seja, uma pessoa deprimida não "desaprende" a reconhecer emoções no seu teste. Isso sugere que o teste mede um traço estável (habilidade cognitiva) e não um estado passageiro.

```
# --- MODELO PARA ANSIEDADE (BAI) ---
model_bai <- glmer(Congruence ~ BAI * Intended_Emotion +
                     (1 | Participants) +  # Efeito aleatório do Participante (Intercepto)
                     (1 | Stimulus_ID),    # Efeito aleatório do Item (Dificuldade)
                   data = data_mixed,
                   family = binomial,
                   control = glmerControl(optimizer = "bobyqa")) # Ajuda a convergir

# Ver resultados
summary(model_bai)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: Congruence ~ BAI * Intended_Emotion + (1 | Participants) + (1 |
##     Stimulus_ID)
##    Data: data_mixed
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   6003.7   6057.2  -2993.8   5987.7     5992
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.3419 -0.5235  0.3447  0.5400  3.2880
##
## Random effects:
##  Groups       Name        Variance Std.Dev.
##  Participants (Intercept) 0.9897   0.9948
##  Stimulus_ID  (Intercept) 0.5178   0.7196
## Number of obs: 6000, groups:  Participants, 200; Stimulus_ID, 30
##
## Fixed effects:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   1.652203   0.268791   6.147 7.91e-10 ***
## BAI                          -0.004034   0.013758  -0.293 0.769350
## Intended_EmotionHappiness     0.035425   0.347126   0.102 0.918716
## Intended_EmotionSadness      -1.325805   0.344483  -3.849 0.000119 ***
## BAI:Intended_EmotionHappiness  0.002971   0.012305   0.241 0.809224
```

```
## BAI:Intended_EmotionSadness    0.019802   0.011754   1.685 0.092035 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) BAI    Int_EH Int_ES BAI:I_EH
## BAI         -0.397
## Intndd_EmtH -0.645  0.122
## Intndd_EmtS -0.653  0.126  0.503
## BAI:Intn_EH  0.176 -0.438 -0.278 -0.138
## BAI:Intn_ES  0.191 -0.472 -0.143 -0.265  0.513
```

```
anova(model_bai, type = "III")
```

```
## Warning in anova.merMod(model_bai, type = "III"): additional arguments ignored:
## 'type'
```

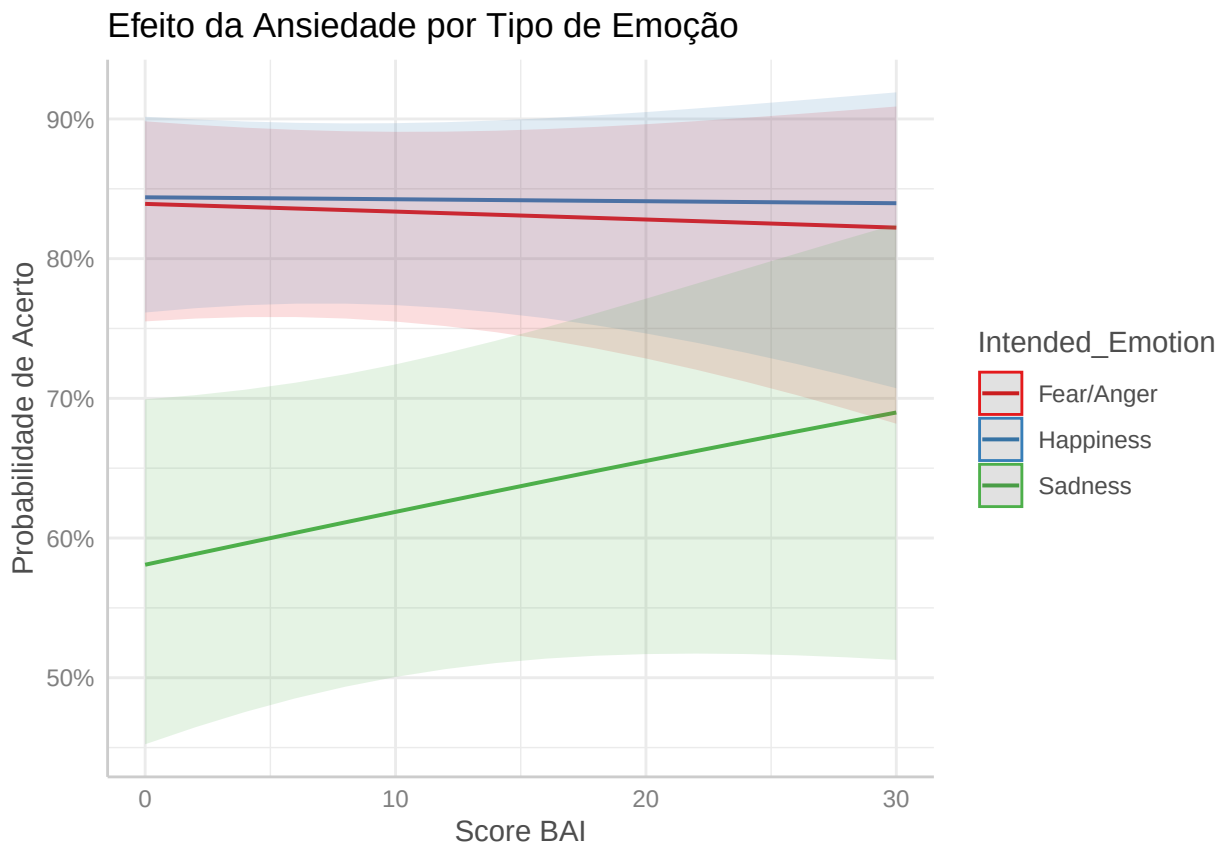
```
## Analysis of Variance Table
##                       npar  Sum Sq Mean Sq F value
## BAI                      1  0.1668  0.1668  0.1668
## Intended_Emotion         2 17.5513  8.7757  8.7757
## BAI:Intended_Emotion     2  3.4086  1.7043  1.7043
```

```
exp(fixef(model_bai))
```

```
##                   (Intercept)                           BAI
##                      5.218464                      0.995974
##     Intended_EmotionHappiness       Intended_EmotionSadness
##                      1.036060                      0.265589
## BAI:Intended_EmotionHappiness   BAI:Intended_EmotionSadness
##                      1.002975                      1.020000
```

```
# Gráfico da interação BAI * Emoção
plot(ggpredict(model_bai, terms = c("BAI", "Intended_Emotion"))) +
  labs(y = "Probabilidade de Acerto", x = "Score BAI", title = "Efeito da Ansiedade por Tipo de Emoção")
```

```
## Data were 'prettified'. Consider using `terms="BAI [all]"` to get smooth
##   plots.
```



6.3 Modelo misto para depressão

```
# --- MODELO PARA DEPRESSÃO (BDI) ---
model_bdi <- glmer(Congruence ~ BDI * Intended_Emotion +
                     (1 | Participants) +
                     (1 | Stimulus_ID),
                   data = data_mixed,
                   family = binomial,
                   control = glmerControl(optimizer = "bobyqa"))

summary(model_bdi)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: Congruence ~ BDI * Intended_Emotion + (1 | Participants) + (1 |
##     Stimulus_ID)
##    Data: data_mixed
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   6001.0   6054.6  -2992.5   5985.0     5992
##
```

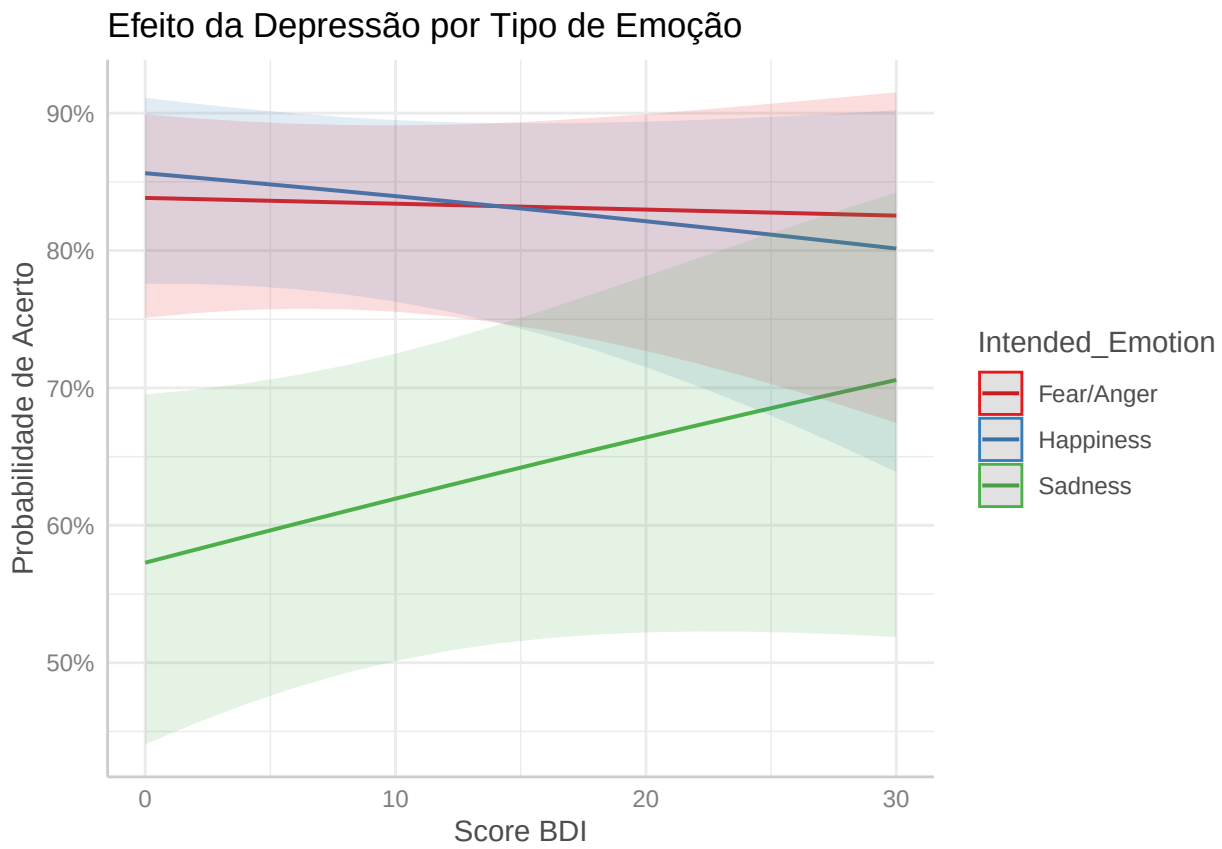
```
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.3648 -0.5271  0.3450  0.5399  3.1777
##
## Random effects:
##  Groups       Name        Variance Std.Dev.
##  Participants (Intercept) 0.9914   0.9957
##  Stimulus_ID  (Intercept) 0.5185   0.7201
## Number of obs: 6000, groups:  Participants, 200; Stimulus_ID, 30
##
## Fixed effects:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    1.645649   0.276570   5.950 2.68e-09 ***
## BDI                           -0.003076   0.015536  -0.198 0.843076
## Intended_EmotionHappiness      0.139228   0.352455   0.395 0.692826
## Intended_EmotionSadness       -1.352482   0.349392  -3.871 0.000108 ***
## BDI:Intended_EmotionHappiness -0.009909   0.013949  -0.710 0.477491
## BDI:Intended_EmotionSadness    0.022469   0.013414   1.675 0.093915 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) BDI    Int_EH Int_ES BDI:I_EH
## BDI         -0.451
## Intndd_EmtH -0.635  0.142
## Intndd_EmtS -0.645  0.148  0.503
## BDI:Intn_EH  0.201 -0.445 -0.322 -0.159
## BDI:Intn_ES  0.217 -0.477 -0.164 -0.308  0.515
```

```
exp(fixef(model_bdi))
```

```
##                   (Intercept)                           BDI
##                     5.1843750                     0.9969292
##     Intended_EmotionHappiness       Intended_EmotionSadness
##                     1.1493856                     0.2585977
## BDI:Intended_EmotionHappiness   BDI:Intended_EmotionSadness
##                     0.9901401                     1.0227238
```

```
plot(ggpredict(model_bdi, terms = c("BDI", "Intended_Emotion"))) +
  labs(y = "Probabilidade de Acerto", x = "Score BDI", title = "Efeito da Depressão por Tipo de Emoção")
```

```
## Data were 'prettified'. Consider using `terms="BDI [all]"` to get smooth
##   plots.
```



Não existe diferença de reconhecimento na emoção entre grupo clínico e não clínico

```
# Vamos buscar as colunas de interpretação no dataframe original 'data'
# e juntar ao 'data_mixed' usando a coluna Participants como chave
data_mixed <- data_mixed %>%
  left_join(data %>% select(Participants, BAI_Interpretation, BDI_Interpretation),
            by = "Participants")

# Verifique se agora elas apareceram
names(data_mixed)
```

```
##  [1] "Participants"       "Group"              "Stimulus_ID"
##  [4] "Perceived_Emotion"  "Congruence"         "Arousal"
##  [7] "Valence"            "Intended_Emotion"   "BAI"
## [10] "BDI"                "BAI_Interpretation" "BDI_Interpretation"
```

```
# Criar grupos binários (Ex: Moderado/Grave = 1, Mínimo/Leve = 0)
# Ajuste os números 3 e 4 baseados na sua legenda do Excel (verifique seus dados)
data_mixed <- data_mixed %>%
  mutate(
    Grupo_Depressao = ifelse(BDI_Interpretation >= 3, "Clinico", "Controle"),
    Grupo_Ansiedade = ifelse(BAI_Interpretation >= 3, "Clinico", "Controle")
  )

# Modelo para Grupo de Depressão
model_grupo_bdi <- glmer(Congruence ~ Grupo_Depressao * Intended_Emotion +
```

```r
                          (1 | Participants) + (1 | Stimulus_ID),
                          data = data_mixed, family = binomial,
                          control = glmerControl(optimizer = "bobyqa"))

summary(model_grupo_bdi)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula:
## Congruence ~ Grupo_Depressao * Intended_Emotion + (1 | Participants) +
##     (1 | Stimulus_ID)
##    Data: data_mixed
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   5999.6   6053.2  -2991.8   5983.6     5992
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.3548 -0.5184  0.3452  0.5387  3.2430
##
## Random effects:
##  Groups       Name        Variance Std.Dev.
##  Participants (Intercept) 0.9854   0.9927
##  Stimulus_ID  (Intercept) 0.5183   0.7199
## Number of obs: 6000, groups:  Participants, 200; Stimulus_ID, 30
##
## Fixed effects:
##                                                Estimate Std. Error z value
## (Intercept)                                      1.2388     0.5121   2.419
## Grupo_DepressaoControle                          0.3976     0.4684   0.849
## Intended_EmotionHappiness                       -0.2563     0.5164  -0.496
## Intended_EmotionSadness                         -0.4479     0.5165  -0.867
## Grupo_DepressaoControle:Intended_EmotionHappiness   0.3300     0.4133   0.798
## Grupo_DepressaoControle:Intended_EmotionSadness    -0.7529     0.4122  -1.826
##                                                Pr(>|z|)
## (Intercept)                                      0.0156 *
## Grupo_DepressaoControle                          0.3959
## Intended_EmotionHappiness                        0.6196
## Intended_EmotionSadness                          0.3858
## Grupo_DepressaoControle:Intended_EmotionHappiness   0.4246
## Grupo_DepressaoControle:Intended_EmotionSadness     0.0678 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) Grp_DC Int_EH Int_ES G_DC:I_EH
## Grp_DprssCn -0.876
## Intndd_EmtH -0.514  0.348
## Intndd_EmtS -0.517  0.350  0.509
## Grp_DC:I_EH  0.398 -0.454 -0.763 -0.393
## Grp_DC:I_ES  0.401 -0.460 -0.394 -0.766  0.515
```

7 Variáveis SD

```
# 1. Limpar tentativas anteriores de join incorreto (remover sufixos .x e .y se existirem)
# Isso garante que estamos trabalhando com o dataframe limpo antes de tentar de novo
data_mixed <- data_mixed %>%
  select(!matches("\\.y$")) %>%  # Remove colunas terminadas em .y
  rename_with(~ str_remove(., "\\.x$"), matches("\\.x$")) # Remove o .x das originais se houver

# 2. Selecionar APENAS as variáveis NOVAS (Corrigido: removido o 'Group')
demograficos_novos <- data %>%
  dplyr::select(Participants, Sex, Age, `Years of study`, `Years of musical education`)
  # Note que removi 'Group' desta lista acima

# 3. Juntar ao data_mixed
data_mixed <- data_mixed %>%
  left_join(demograficos_novos, by = "Participants")

# 4. Verificar se funcionou (veja se 'Group' e 'Sex' aparecem corretamente)
print(names(data_mixed))
```

```
##  [1] "Participants"              "Group"
##  [3] "Stimulus_ID"               "Perceived_Emotion"
##  [5] "Congruence"                "Arousal"
##  [7] "Valence"                   "Intended_Emotion"
##  [9] "BAI"                       "BDI"
## [11] "BAI_Interpretation"        "BDI_Interpretation"
## [13] "Grupo_Depressao"           "Grupo_Ansiedade"
## [15] "Sex"                       "Age"
## [17] "Years of study"            "Years of musical education"
```

```
# 5. AGORA sim, criar os fatores
data_mixed <- data_mixed %>%
  mutate(
    Group_Factor = factor(Group, levels = c(2, 1), labels = c("Nao-Musico", "Musico")),
    Sex_Factor = factor(Sex, levels = c(1, 2), labels = c("Feminino", "Masculino"))
  )

# Verifique o resultado final
head(data_mixed %>% select(Participants, Group_Factor, Sex_Factor))
```

```
## # A tibble: 6 x 3
##   Participants Group_Factor Sex_Factor
##          <dbl> <fct>        <fct>
## 1            1 Musico       Feminino
## 2            1 Musico       Feminino
## 3            1 Musico       Feminino
## 4            1 Musico       Feminino
## 5            1 Musico       Feminino
## 6            1 Musico       Feminino
```

7.1 Músicos X Não-Músicos

Não há diferença

```
model_musicos <- glmer(Congruence ~ Group_Factor * Intended_Emotion +
                         (1 | Participants) +
                         (1 | Stimulus_ID),
                       data = data_mixed,
                       family = binomial,
                       control = glmerControl(optimizer = "bobyqa"))

summary(model_musicos)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: Congruence ~ Group_Factor * Intended_Emotion + (1 | Participants) +
##     (1 | Stimulus_ID)
##    Data: data_mixed
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   6005.3   6058.9  -2994.6   5989.3     5992
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.2201 -0.5282  0.3461  0.5412  3.3065
##
## Random effects:
##  Groups       Name        Variance Std.Dev.
##  Participants (Intercept) 0.9883   0.9941
##  Stimulus_ID  (Intercept) 0.5177   0.7195
## Number of obs: 6000, groups:  Participants, 200; Stimulus_ID, 30
##
## Fixed effects:
##                                             Estimate Std. Error z value
## (Intercept)                                  1.61230    0.26338   6.122
## Group_FactorMusico                           0.01613    0.18555   0.087
## Intended_EmotionHappiness                    0.03250    0.34361   0.095
## Intended_EmotionSadness                     -1.26341    0.34139  -3.701
## Group_FactorMusico:Intended_EmotionHappiness 0.05261    0.16892   0.311
## Group_FactorMusico:Intended_EmotionSadness   0.18139    0.15904   1.141
##                                             Pr(>|z|)
## (Intercept)                                 9.27e-10 ***
## Group_FactorMusico                          0.930739
## Intended_EmotionHappiness                   0.924649
## Intended_EmotionSadness                     0.000215 ***
## Group_FactorMusico:Intended_EmotionHappiness 0.755444
## Group_FactorMusico:Intended_EmotionSadness   0.254061
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) Grp_FM Int_EH Int_ES G_FM:I_EH
```

```
## Grp_FctrMsc -0.352
## Intndd_EmtH -0.651  0.110
## Intndd_EmtS -0.660  0.114  0.503
## Grp_FM:I_EH  0.157 -0.447 -0.244 -0.121
## Grp_FM:I_ES  0.174 -0.490 -0.128 -0.233  0.522
```

7.2 Anos de estudo musical

Não há diferença

```
model_anos_musica <- glmer(Congruence ~ `Years of musical education` * Intended_Emotion +
                             (1 | Participants) +
                             (1 | Stimulus_ID),
                           data = data_mixed,
                           family = binomial,
                           control = glmerControl(optimizer = "bobyqa"))

summary(model_anos_musica)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: Congruence ~ `Years of musical education` * Intended_Emotion +
##     (1 | Participants) + (1 | Stimulus_ID)
##    Data: data_mixed
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   6002.2   6055.7  -2993.1   5986.2     5992
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.3051 -0.5278  0.3468  0.5401  3.3066
##
## Random effects:
##  Groups       Name        Variance Std.Dev.
##  Participants (Intercept) 0.9884   0.9942
##  Stimulus_ID  (Intercept) 0.5178   0.7196
## Number of obs: 6000, groups:  Participants, 200; Stimulus_ID, 30
##
## Fixed effects:
##                                                  Estimate Std. Error
## (Intercept)                                      1.651310   0.257833
## `Years of musical education`                    -0.004997   0.012262
## Intended_EmotionHappiness                        0.150657   0.340644
## Intended_EmotionSadness                         -1.223641   0.338304
## `Years of musical education`:Intended_EmotionHappiness -0.014511   0.010989
## `Years of musical education`:Intended_EmotionSadness    0.008304   0.010538
##                                                 z value Pr(>|z|)
## (Intercept)                                       6.405 1.51e-10 ***
## `Years of musical education`                     -0.408 0.683625
## Intended_EmotionHappiness                         0.442 0.658292
## Intended_EmotionSadness                          -3.617 0.000298 ***
```

```
## `Years of musical education`:Intended_EmotionHappiness  -1.320 0.186685
## `Years of musical education`:Intended_EmotionSadness     0.788 0.430717
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) `Yomed` Int_EH Int_ES `Yome`:I_EH
## `Yromedctn` -0.292
## Intndd_EmtH -0.658  0.089
## Intndd_EmtS -0.667  0.093   0.502
## `Yome`:I_EH  0.131 -0.450  -0.204 -0.100
## `Yome`:I_ES  0.142 -0.482  -0.104 -0.191  0.524
```

7.3 Sexo e Idade

Homens têm desempenho pior que Mulheres. Homens têm cerca de 31% menos chance de acertar do que as mulheres.

```
model_demo <- glmer(Congruence ~ Sex_Factor + Age + `Years of study` +
                      (1 | Participants) +
                      (1 | Stimulus_ID),
                    data = data_mixed,
                    family = binomial,
                    control = glmerControl(optimizer = "bobyqa"))

summary(model_demo)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula:
## Congruence ~ Sex_Factor + Age + `Years of study` + (1 | Participants) +
##     (1 | Stimulus_ID)
##    Data: data_mixed
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   6011.2   6051.4  -2999.6   5999.2     5994
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.5010 -0.5292  0.3449  0.5391  3.1556
##
## Random effects:
##  Groups       Name        Variance Std.Dev.
##  Participants (Intercept) 0.9537   0.9766
##  Stimulus_ID  (Intercept) 0.8425   0.9179
## Number of obs: 6000, groups:  Participants, 200; Stimulus_ID, 30
##
## Fixed effects:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          1.276572   0.455542   2.802  0.00507 **
## Sex_FactorMasculino -0.369128   0.155128  -2.379  0.01734 *
```

```
## Age                  0.001519   0.014682   0.103  0.91762
## `Years of study`     0.006727   0.023734   0.283  0.77684
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) Sx_FcM Age
## Sx_FctrMscl -0.102
## Age         -0.494 -0.109
## `Yrsofstdy` -0.459  0.038 -0.438
```

7.4 Músicos têm maior Habilidade Geral (G)?

```
# 1. Trazer o G_Score do dataframe original 'data' para o 'data_mixed'
data_mixed <- data_mixed %>%
  left_join(data %>% select(Participants, G_Score), by = "Participants")

# 2. Modelo Linear (LM) com o Escore Geral
# Usamos distinct() para ter apenas uma linha por participante (já que o G_Score é igual para todas as
dados_por_pessoa <- data_mixed %>%
  distinct(Participants, .keep_all = TRUE)

model_global <- lm(G_Score ~ Group_Factor + Sex_Factor + Age + `Years of study`,
                   data = dados_por_pessoa)

summary(model_global)
```

```
##
## Call:
## lm(formula = G_Score ~ Group_Factor + Sex_Factor + Age + `Years of study`,
##     data = dados_por_pessoa)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.75197 -0.33133  0.02374  0.34354  1.09979
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.077330   0.195434  -0.396   0.6928
## Group_FactorMusico   0.018266   0.070105   0.261   0.7947
## Sex_FactorMasculino -0.161993   0.070539  -2.297   0.0227 *
## Age                  0.002084   0.006651   0.313   0.7544
## `Years of study`     0.003366   0.010823   0.311   0.7561
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4951 on 195 degrees of freedom
## Multiple R-squared:  0.02774,    Adjusted R-squared:  0.007801
## F-statistic: 1.391 on 4 and 195 DF,  p-value: 0.2384
```

7.5 Não linearidade

7.5.1 Árvore de decisão

É possível que as divisões sejam overfitting, por isso vale testar a poda, abaixo.

```
# 1. Selecionar apenas as variáveis que fazem sentido para prever o G_Score
# (Usando o dataframe 'dados_por_pessoa' que já tem o G_Score e 1 linha por sujeito)
dados_tree <- dados_por_pessoa %>%
  select(
    G_Score,
    Sex_Factor,
    Age,
    `Years of study`,
    Group_Factor,        # Músico vs Não
    `Years of musical education`,
    BAI,
    BDI
  )

# 2. Rodar a Árvore de Regressão
# cp = 0.01 é o parâmetro de complexidade padrão.
# Se a árvore não crescer, tentaremos baixar isso só pra ver se existe algum sinal fraco.
arvore <- rpart(G_Score ~ .,
                data = dados_tree,
                method = "anova")

# 3. Visualizar
print("--- Resumo da Árvore ---")
```

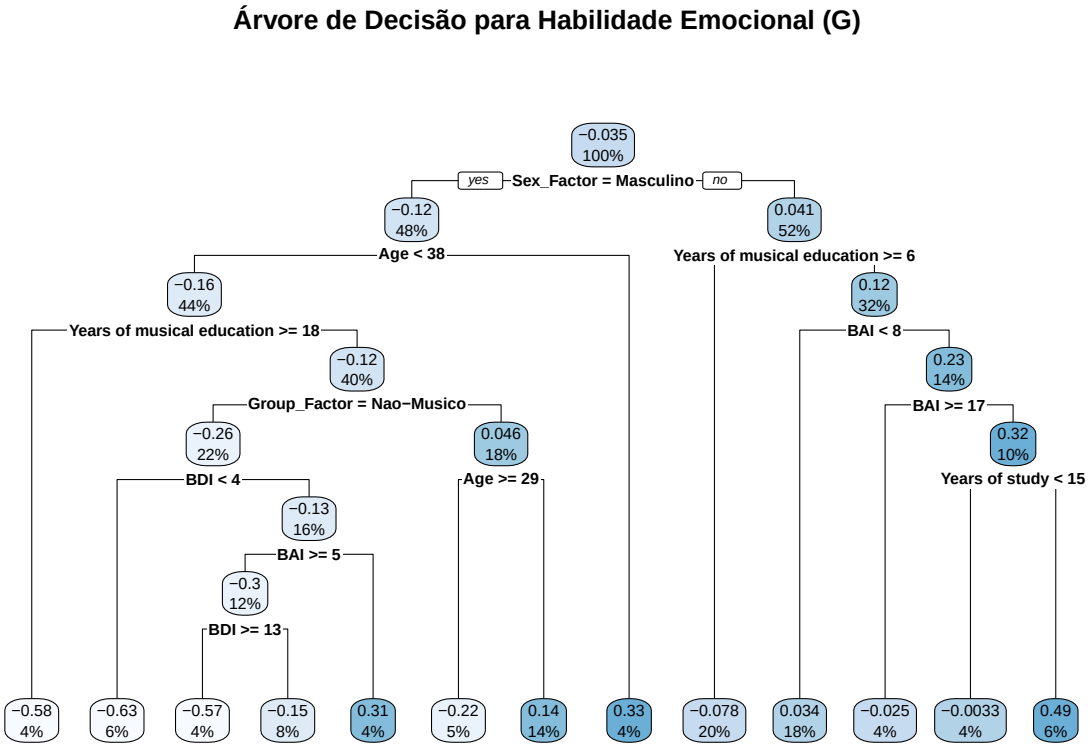
```
## [1] "--- Resumo da Árvore ---"
```

```
printcp(arvore) # Mostra se valeu a pena dividir os dados
```

```
##
## Regression tree:
## rpart(formula = G_Score ~ ., data = dados_tree, method = "anova")
##
## Variables actually used in tree construction:
## [1] Age                         BAI
## [3] BDI                         Group_Factor
## [5] Sex_Factor                  Years of musical education
## [7] Years of study
##
## Root node error: 49.156/200 = 0.24578
##
## n= 200
##
##         CP nsplit rel error xerror    xstd
## 1 0.036511      0   1.00000 1.0075 0.11313
## 2 0.019232      6   0.78094 1.1229 0.12284
## 3 0.018716      7   0.76170 1.1845 0.11962
## 4 0.018617      8   0.74299 1.1871 0.11952
```

```
## 5 0.015555      9   0.72437 1.2098 0.11965
## 6 0.010000     12   0.67770 1.2549 0.12129
```

```r
# Plotar (Se houver algo para plotar)
rpart.plot(arvore, main = "Árvore de Decisão para Habilidade Emocional (G)")
```



7.5.2 Árvore com poda

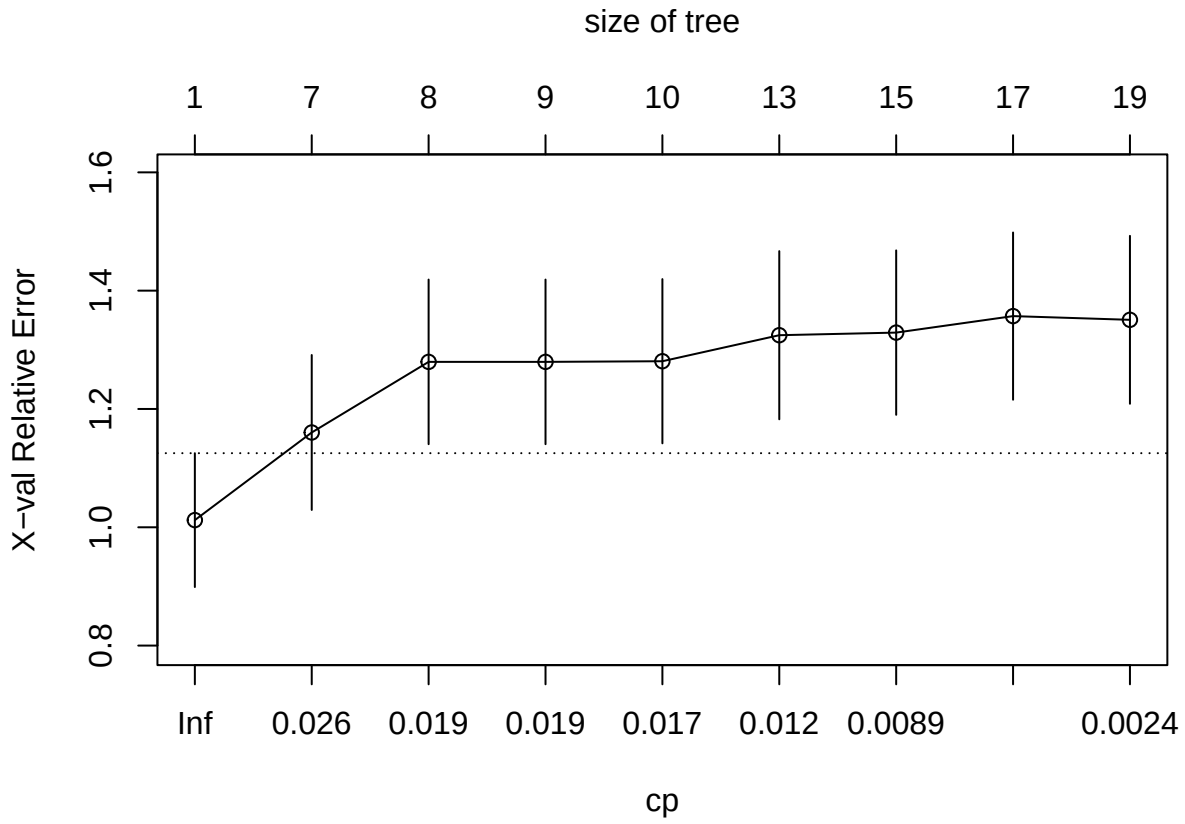
A única variável preditora foi sexo, de fato.

```r
# 1. Rodar a árvore "Grande" (igual você já fez, mas garantindo que ela cresça)
# Definimos um cp bem baixo (0.001) para ela crescer bastante antes de podar
arvore_completa <- rpart(G_Score ~ .,
                         data = dados_tree,
                         method = "anova",
                         control = rpart.control(cp = 0.001))

# 2. Visualizar a tabela de complexidade (Isso mostra onde o erro é menor)
printcp(arvore_completa)
```

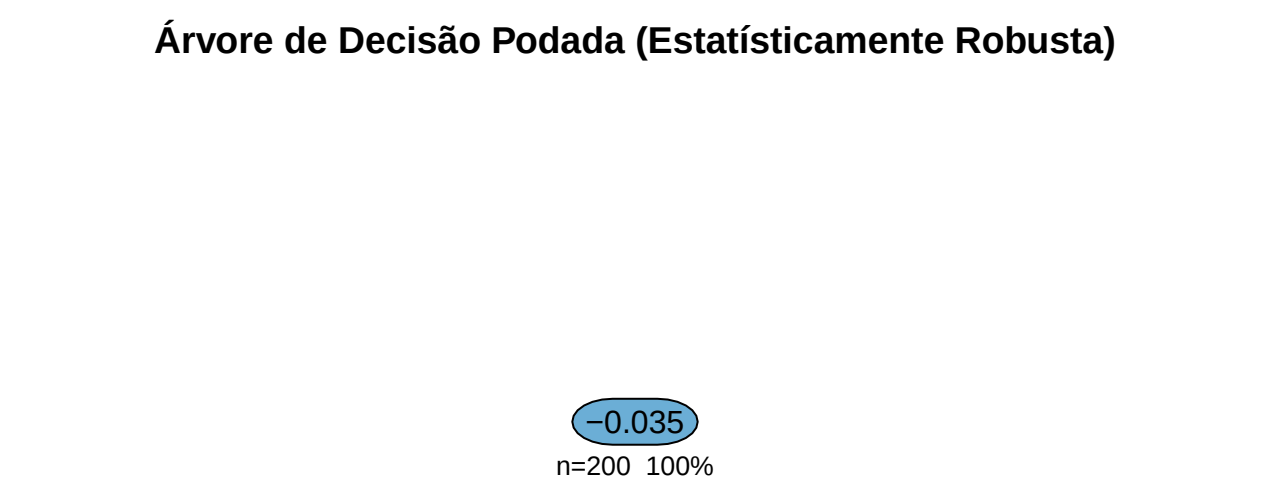
```
##
## Regression tree:
## rpart(formula = G_Score ~ ., data = dados_tree, method = "anova",
##     control = rpart.control(cp = 0.001))
```

```
##
## Variables actually used in tree construction:
## [1] Age                          BAI
## [3] BDI                          Group_Factor
## [5] Sex_Factor                   Years of musical education
## [7] Years of study
##
## Root node error: 49.156/200 = 0.24578
##
## n= 200
##
##          CP nsplit rel error xerror    xstd
## 1 0.0365108      0   1.00000 1.0121 0.11315
## 2 0.0192319      6   0.78094 1.1602 0.13097
## 3 0.0187164      7   0.76170 1.2796 0.13917
## 4 0.0186172      8   0.74299 1.2796 0.13917
## 5 0.0155553      9   0.72437 1.2807 0.13882
## 6 0.0095788     12   0.67770 1.3247 0.14214
## 7 0.0081812     14   0.65855 1.3291 0.13899
## 8 0.0055303     16   0.64218 1.3570 0.14137
## 9 0.0010000     18   0.63112 1.3507 0.14196
```

```
plotcp(arvore_completa) # Gráfico do erro vs tamanho da árvore
```



```r
# 3. Identificar automaticamente o melhor CP (aquele com menor erro de validação)
best_cp_index <- which.min(arvore_completa$cptable[, "xerror"])
best_cp <- arvore_completa$cptable[best_cp_index, "CP"]

# 4. Podar a árvore usando esse CP ideal
arvore_podada <- prune(arvore_completa, cp = best_cp)

# 5. Plotar a Árvore Podada (A versão "Honesta")
rpart.plot(arvore_podada,
           main = "Árvore de Decisão Podada (Estatísticamente Robusta)",
           type = 3,
           extra = 101,
           under = TRUE,
           fallen.leaves = TRUE,
           box.palette = "BuOr")
```




```r
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16)
## Platform: x86_64-apple-darwin20 (64-bit)
## Running under: macOS Monterey 12.7.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRlapack.dylib;  LAPACK
```

```
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Sao_Paulo
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] ggeffects_1.7.0      randomForest_4.7-1.1 rpart.plot_3.1.2
##  [4] rpart_4.1.23         sjPlot_2.8.16        lmerTest_3.1-3
##  [7] lme4_1.1-35.5        Matrix_1.6-1.1       semTools_0.5-6
## [10] lavaan_0.6-18        ltm_1.2-0            polycor_0.8-1
## [13] msm_1.8.2            MASS_7.3-60          conflicted_1.2.0
## [16] lubridate_1.9.3      forcats_1.0.0        stringr_1.5.1
## [19] dplyr_1.1.4          purrr_1.0.2          readr_2.1.5
## [22] tibble_3.2.1         ggplot2_3.5.1        tidyverse_2.0.0
## [25] tidyr_1.3.1          readxl_1.4.3
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.1    sjlabelled_1.2.0    farver_2.1.2
##  [4] fastmap_1.2.0       TH.data_1.1-2       sjstats_0.19.0
##  [7] digest_0.6.36       estimability_1.5.1  timechange_0.3.0
## [10] lifecycle_1.0.4     survival_3.7-0      magrittr_2.0.3
## [13] compiler_4.3.1      rlang_1.1.6         tools_4.3.1
## [16] utf8_1.2.6          yaml_2.3.10         knitr_1.48
## [19] labeling_0.4.3      mnormt_2.1.1        multcomp_1.4-26
## [22] expm_0.999-9        withr_3.0.2         numDeriv_2016.8-1.1
## [25] datawizard_0.10.0   grid_4.3.1          stats4_4.3.1
## [28] fansi_1.0.6         xtable_1.8-4        colorspace_2.1-1
## [31] emmeans_1.10.2      scales_1.3.0        insight_0.20.2
## [34] cli_3.6.5           mvtnorm_1.2-5       rmarkdown_2.27
## [37] generics_0.1.3      rstudioapi_0.16.0   performance_0.12.2
## [40] tzdb_0.4.0          minqa_1.2.7         cachem_1.1.0
## [43] splines_4.3.1       parallel_4.3.1      cellranger_1.1.0
## [46] vctrs_0.6.5         boot_1.3-30         sandwich_3.1-0
## [49] hms_1.1.3           glue_1.8.0          nloptr_2.1.1
## [52] admisc_0.35         codetools_0.2-20    stringi_1.8.4
## [55] gtable_0.3.5        quadprog_1.5-8      munsell_0.5.1
## [58] pillar_1.9.0        htmltools_0.5.8.1   R6_2.5.1
## [61] evaluate_0.24.0     pbivnorm_0.6.0      lattice_0.22-6
## [64] haven_2.5.4         highr_0.11          snakecase_0.11.1
## [67] memoise_2.0.1       Rcpp_1.0.14         coda_0.19-4.1
## [70] nlme_3.1-165        mgcv_1.9-1          xfun_0.52
## [73] zoo_1.8-12          sjmisc_2.8.10       pkgconfig_2.0.3
```