FAI Item Selection

```r
suppressPackageStartupMessages({
  library("tidyverse")
  library("TAM")
  library("psych")
  library("lordif")
  library("ggridges")
  library("ggpubr")
  library("paletteer")
  library("readxl")
  library("mice")
  library("VIM")
  library("paran")
  library("lavaan")
  library("polycor") # for hetcor()
  library("multilevel")
  library("miceRanger")
  library("papaja")
  theme_set(theme_apa())
  library("outForest")
  library("lavaanExtra")
  # Caricare le librerie necessarie
  library(glmnet) # Per LASSO regression
  library(randomForest) # Per valutare importanza delle variabili
  library(Boruta) # Per selezione robusta delle variabili
  library(missRanger)
  library(caret)
  library(pROC)
  library(lavaan)
  library(semTools)
})

library("here")
```

```
here() starts at /Users/corrado/_repositories/meyer_fai
```

```
# Increase max print
options(max.print = .Machine$integer.max)

source(here("functions", "fai_funs.R"))
```

Import data

```
fai_s <- read_xlsx(
  here("data", "raw", "FAI_TOT_2020_corrected.xlsx"), col_names = TRUE
)
```

Demographic information.

```
demo_info <- recode_demo_info(fai_s)
```

Categorize disease severity.

```
# Function to categorize disease severity
categorize_severity <- function(disease) {
  disease <- tolower(disease) # Normalize text (case insensitive)

  if (grepl("allergia|asma lieve|rinite", disease)) {
    return("Lieve")
  } else if (grepl("diabete|colite ulcerosa|artrite|cardiopatia|insufficienza renale|ipertens
    return("Moderata")
  } else if (grepl("fibrosi cistica|leucemia|epilessia|distrofia|sindrome nefrosica|osteosar
    return("Grave")
  } else if (grepl("tumore|cancro|metastasi|cuore ipoplasico|malattia metabolica|aciduria|en
    return("Critica")
  } else {
    return(NA) # Unclassified
  }
}

# Apply the function to categorize chronic diseases
demo_info <- demo_info %>%
  mutate(severity_level = factor(
    sapply(chronic_disease, categorize_severity),
```

```r
                                  levels = c("Moderata", "Lieve", "Grave", "Critica"),
                                  ordered = TRUE)
  )

# Verify transformation
# table(demo_info$severity_level, demo_info$death_risk)
```

In the original administration, there is no psychological criterion variable available. The only external variable that could serve as a criterion is death risk, based on the assumption that coping is more challenging for a life-threatening disease compared to a non-life-threatening condition.

```r
table(demo_info$death_risk)
```

```

  0   1
264 230
```

Data Wrangling

Extract item responses (columns 51 to 247).

```r
items <- fai_s[, 51:247]
colnames(items) <- paste0("item_", seq_len(ncol(items))) # Rename items
```

Identify and remove items with excessive missing values.

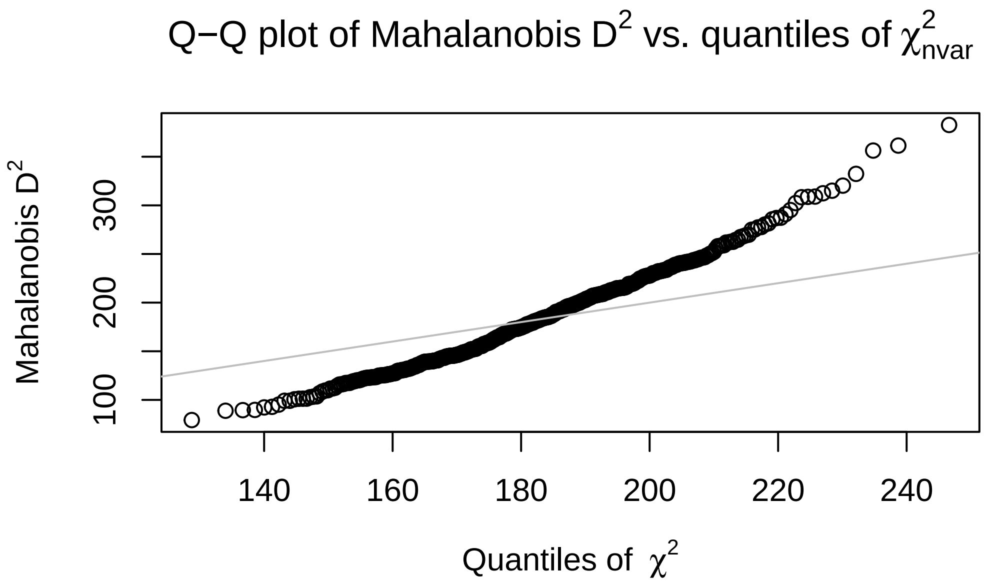
```r
n_nas <- colSums(is.na(items))
bad_items <- names(n_nas[n_nas > 50])  # Adjust threshold as needed
items_filtered <- items %>%
  dplyr::select(-all_of(bad_items))
```

Combine cleaned item data with demographic info.

```r
mydata <- bind_cols(demo_info, items_filtered) %>%
  mutate(subj_id = as.factor(seq_len(nrow(.))))
```

Identify and Remove Problematic Participants

```r
d_num <- mydata %>%
  dplyr::select(starts_with("item_")) %>%
  dplyr::select(where(is.numeric))  # Ensure numeric selection

# Mahalanobis Distance
mahal_out <- careless::mahad(d_num)
```




```r
mahal_cutoff <- boxplot(mahal_out, plot = FALSE)$stats[5]
bad_mahal <- mydata$subj_id[mahal_out > mahal_cutoff]

# Longstring Analysis
longstring_out <- careless::longstring(d_num)
longstring_cutoff <- boxplot(longstring_out, plot = FALSE)$stats[5]
bad_longstring <- mydata$subj_id[longstring_out > longstring_cutoff]

# IRV Analysis
irv_out <- careless::irv(d_num)
irv_cutoff <- boxplot(irv_out, plot = FALSE)$stats[5]
bad_irv <- mydata$subj_id[irv_out > irv_cutoff]

# Person Total Correlation
cm <- colMeans(d_num, na.rm = TRUE)
```

```
person_tot_cor <- apply(d_num, 1, function(x) cor(x, cm, use = "complete.obs"))
person_tot_cutoff <- boxplot(person_tot_cor, plot = FALSE)$stats[1]
bad_person_tot <- mydata$subj_id[person_tot_cor < person_tot_cutoff]

# Combine all flagged participants
bad_ids <- unique(c(bad_mahal, bad_longstring, bad_irv, bad_person_tot))

# Remove problematic participants before imputation
d_clean <- mydata %>%
  dplyr::filter(!subj_id %in% bad_ids)
```

Multiple Imputation

```
# Select numeric item responses for imputation
items_clean <- d_clean %>%
  dplyr::select(starts_with("item_")) %>%
  dplyr::select(where(is.numeric))

# Perform multiple imputation
imp <- missRanger(items_clean, num.trees = 100)

# Ensure imputed values are within the original range and rounded to integers
imp <- imp %>%
  mutate(across(everything(), ~ round(pmax(min(., na.rm = TRUE),
                                           pmin(., max(., na.rm = TRUE))), 0)))

# Extract demographic data excluding items
demo_clean <- d_clean %>%
  dplyr::select(-starts_with("item_"))

# Final cleaned dataset
df <- bind_cols(imp, demo_clean)
```

```
dim(df)
```

```
[1] 453 208
```

Item Selection Based on Clinical Criteria

```
# Define selected items based on clinical criteria
selected_items <- c(
  "i_86", "i_57", "i_5", "i_85", "i_81",
  "i_105", "i_48", "i_133", "i_129", "i_39", "i_103",
  "i_143", "i_79",
  "i_111", "i_34", "i_119", "i_116", "i_23", "i_45", "i_41",
  "i_186", "i_38", "i_128", "i_7", "i_16", "i_29",
  "i_137", "i_96", "i_194"
)

# Convert to match column names in `imp`
selected_items_corrected <- paste0("item_", sub("^i_", "", selected_items))

# Ensure only existing columns are selected
selected_items_corrected <- intersect(selected_items_corrected, colnames(imp))

# Select the matching columns
imp_selected <- imp %>%
  dplyr::select(any_of(selected_items_corrected))

# Validate the selection
if (length(selected_items_corrected) != length(selected_items)) {
  warning("Some selected items were not found in the dataset.")
}
print(names(imp_selected))
```

```
 [1] "item_86"  "item_57"  "item_5"   "item_85"  "item_81"  "item_105"
 [7] "item_48"  "item_133" "item_129" "item_39"  "item_103" "item_143"
[13] "item_79"  "item_111" "item_34"  "item_119" "item_116" "item_23"
[19] "item_45"  "item_41"  "item_186" "item_38"  "item_128" "item_7"
[25] "item_16"  "item_29"  "item_137" "item_96"  "item_194"
```

Define Target Matrix for Factor Structure (29x6)

```
# Initialize a 29x6 matrix filled with zeros
TARGET <- matrix(0, nrow = length(selected_items_corrected), ncol = 6)
```

```r
# Assign factor loadings based on clinical criteria
TARGET[1:5, 1]  <- 1  # F1
TARGET[6:11, 2] <- 1  # F2
TARGET[12:13, 3] <- 1 # F3
TARGET[14:20, 4] <- 1 # F4
TARGET[21:26, 5] <- 1 # F5
TARGET[27:29, 6] <- 1 # F6

# Add row names for clarity
rownames(TARGET) <- selected_items_corrected
colnames(TARGET) <- paste0("F", 1:6)

# Print the target rotation matrix
print(TARGET)
```

```
         F1 F2 F3 F4 F5 F6
item_86   1  0  0  0  0  0
item_57   1  0  0  0  0  0
item_5    1  0  0  0  0  0
item_85   1  0  0  0  0  0
item_81   1  0  0  0  0  0
item_105  0  1  0  0  0  0
item_48   0  1  0  0  0  0
item_133  0  1  0  0  0  0
item_129  0  1  0  0  0  0
item_39   0  1  0  0  0  0
item_103  0  1  0  0  0  0
item_143  0  0  1  0  0  0
item_79   0  0  1  0  0  0
item_111  0  0  0  1  0  0
item_34   0  0  0  1  0  0
item_119  0  0  0  1  0  0
item_116  0  0  0  1  0  0
item_23   0  0  0  1  0  0
item_45   0  0  0  1  0  0
item_41   0  0  0  1  0  0
item_186  0  0  0  0  1  0
item_38   0  0  0  0  1  0
item_128  0  0  0  0  1  0
item_7    0  0  0  0  1  0
item_16   0  0  0  0  1  0
item_29   0  0  0  0  1  0
```

```
item_137  0  0  0  0  0  1
item_96   0  0  0  0  0  1
item_194  0  0  0  0  0  1
```

Define the ESEM Model in Lavaan Syntax

```
model <- '
    efa("efa1")*f1 +
    efa("efa1")*f2 +
    efa("efa1")*f3 +
    efa("efa1")*f4 +
    efa("efa1")*f5 +
    efa("efa1")*f6 =~
       # F1
       item_86 + item_57 + item_5 + item_85 + item_81 +
       # F2
       item_105 + item_48 + item_133 + item_129 + item_39 + item_103 +
       # F3
       item_143 + item_79 +
       # F4
       item_111 + item_34 + item_119 + item_116 + item_23 + item_45 + item_41 +
       # F5
       item_186 + item_38 + item_128 + item_7 + item_16 + item_29 +
       # F6
       item_137 + item_96 + item_194
'
```

Fit the ESEM Model with Target Rotation

```
fit1 <- sem(
  model = model,
  data = imp_selected,
  ordered = TRUE,  # Use ordered estimation if Likert-type items
  rotation = "target",
  rotation.args = list(target = TARGET)
)
```

```r
fit_indices <- fitMeasures(fit1, c(
  "chisq", "df", "pvalue", "cfi", "tli", "rmsea", "srmr",
  "chisq.scaled", "df.scaled", "pvalue.scaled", "cfi.robust", "tli.robust",
  "rmsea.robust", "srmr"
))

# Display selected fit indices (Standard & Robust)
cat("\nModel Fit Indices (Standard & Robust):\n")
```

```
Model Fit Indices (Standard & Robust):
```

```r
fit_indices_df <- data.frame(
  Measure = names(fit_indices),
  Value = round(fit_indices, 3)
)
print(fit_indices_df)
```

```
         Measure   Value
1          chisq 235.901
2             df 247.000
3         pvalue   0.683
4            cfi   1.000
5            tli   1.001
6          rmsea   0.000
7           srmr   0.037
8   chisq.scaled 461.729
9      df.scaled 247.000
10 pvalue.scaled   0.000
11    cfi.robust   0.922
12    tli.robust   0.872
13  rmsea.robust   0.059
14          srmr   0.037
```

```r
# Extract Standardized Factor Loadings
# Get standardized solution
std_solution <- standardizedSolution(fit1)

# Filter only loadings (Lambda matrix)
std_loadings <- std_solution %>%
  filter(op == "=~") %>%
```

```r
  dplyr::select(lhs, rhs, est.std) %>%
  arrange(lhs, desc(abs(est.std)))  # Sorted by factor and magnitude

colnames(std_loadings) <- c("Factor", "Item", "Standardized Loading")

# Display standardized loadings
cat("\nStandardized Factor Loadings:\n")
```

```
Standardized Factor Loadings:
```

```r
print(std_loadings)
```

```
    Factor     Item Standardized.Loading
1       f1  item_81                0.577
2       f1  item_86                0.571
3       f1  item_48                0.511
4       f1 item_105                0.508
5       f1  item_57                0.461
6       f1  item_38                0.358
7       f1  item_39                0.341
8       f1   item_5                0.340
9       f1 item_103                0.312
10      f1 item_194               -0.264
11      f1 item_186                0.221
12      f1 item_137               -0.221
13      f1   item_7                0.177
14      f1  item_41                0.170
15      f1  item_96               -0.168
16      f1  item_85                0.167
17      f1  item_34                0.164
18      f1 item_119               -0.157
19      f1  item_45                0.134
20      f1  item_29                0.123
21      f1  item_23               -0.116
22      f1 item_116                0.077
23      f1 item_129                0.067
24      f1  item_79                0.065
25      f1 item_143                0.065
26      f1 item_111               -0.053
27      f1 item_133                0.031
```

28	f1	item_128	-0.022
29	f1	item_16	0.007
30	f2	item_129	0.908
31	f2	item_133	0.836
32	f2	item_81	0.722
33	f2	item_105	0.594
34	f2	item_48	0.533
35	f2	item_103	0.500
36	f2	item_5	0.475
37	f2	item_39	0.400
38	f2	item_116	0.399
39	f2	item_29	0.344
40	f2	item_111	0.274
41	f2	item_86	0.260
42	f2	item_23	0.239
43	f2	item_137	0.202
44	f2	item_96	0.196
45	f2	item_57	0.191
46	f2	item_34	0.171
47	f2	item_45	0.144
48	f2	item_85	0.120
49	f2	item_79	0.101
50	f2	item_16	-0.093
51	f2	item_119	0.086
52	f2	item_41	-0.066
53	f2	item_7	-0.047
54	f2	item_143	-0.045
55	f2	item_128	-0.037
56	f2	item_194	-0.033
57	f2	item_186	-0.027
58	f2	item_38	0.023
59	f3	item_143	0.928
60	f3	item_79	0.852
61	f3	item_194	0.236
62	f3	item_186	0.226
63	f3	item_85	0.213
64	f3	item_45	-0.200
65	f3	item_86	-0.181
66	f3	item_39	-0.173
67	f3	item_116	-0.152
68	f3	item_5	0.140
69	f3	item_38	0.135
70	f3	item_111	-0.121

71	f3	item_105	0.119
72	f3	item_128	-0.106
73	f3	item_16	-0.102
74	f3	item_48	0.089
75	f3	item_7	-0.077
76	f3	item_23	-0.075
77	f3	item_96	-0.071
78	f3	item_103	-0.062
79	f3	item_41	-0.060
80	f3	item_133	0.054
81	f3	item_34	-0.045
82	f3	item_119	-0.029
83	f3	item_129	0.023
84	f3	item_81	-0.018
85	f3	item_57	-0.012
86	f3	item_29	0.009
87	f3	item_137	0.003
88	f4	item_119	0.836
89	f4	item_111	0.803
90	f4	item_34	0.775
91	f4	item_23	0.705
92	f4	item_41	0.595
93	f4	item_39	0.422
94	f4	item_103	0.378
95	f4	item_116	0.281
96	f4	item_194	0.232
97	f4	item_48	0.229
98	f4	item_85	0.187
99	f4	item_143	-0.182
100	f4	item_38	-0.170
101	f4	item_137	0.155
102	f4	item_96	0.139
103	f4	item_128	0.129
104	f4	item_86	0.105
105	f4	item_29	-0.082
106	f4	item_57	-0.082
107	f4	item_105	0.070
108	f4	item_7	-0.060
109	f4	item_133	0.060
110	f4	item_45	-0.060
111	f4	item_79	-0.055
112	f4	item_129	-0.054
113	f4	item_16	0.033

114	f4	item_81	-0.030
115	f4	item_5	-0.029
116	f4	item_186	-0.021
117	f5	item_7	1.015
118	f5	item_128	0.800
119	f5	item_16	0.664
120	f5	item_186	0.481
121	f5	item_38	0.426
122	f5	item_45	0.397
123	f5	item_23	-0.318
124	f5	item_86	0.307
125	f5	item_85	0.238
126	f5	item_137	0.178
127	f5	item_103	-0.166
128	f5	item_57	0.161
129	f5	item_143	0.161
130	f5	item_79	0.160
131	f5	item_39	0.157
132	f5	item_194	0.138
133	f5	item_81	0.121
134	f5	item_129	0.114
135	f5	item_111	-0.100
136	f5	item_34	-0.096
137	f5	item_119	-0.091
138	f5	item_105	0.091
139	f5	item_96	0.077
140	f5	item_5	-0.064
141	f5	item_41	-0.060
142	f5	item_29	0.057
143	f5	item_133	-0.033
144	f5	item_116	-0.022
145	f5	item_48	-0.021
146	f6	item_137	0.842
147	f6	item_96	0.758
148	f6	item_116	0.492
149	f6	item_194	0.399
150	f6	item_81	-0.286
151	f6	item_103	0.256
152	f6	item_23	0.242
153	f6	item_39	0.223
154	f6	item_38	0.223
155	f6	item_57	-0.221
156	f6	item_86	-0.216

```
157     f6  item_48               -0.199
158     f6  item_34                0.167
159     f6 item_128                0.130
160     f6  item_45                0.130
161     f6 item_143                0.126
162     f6  item_41               -0.120
163     f6 item_186                0.108
164     f6   item_7                0.106
165     f6 item_111               -0.097
166     f6 item_133                0.082
167     f6  item_79                0.056
168     f6  item_16                0.033
169     f6 item_119               -0.017
170     f6  item_85               -0.011
171     f6   item_5                0.010
172     f6 item_129               -0.009
173     f6 item_105                0.005
174     f6  item_29                0.000
```

```r
# Extract Interfactor Correlations

# Extract standardized correlations (Phi matrix)
interfactor_corr <- std_solution %>%
  dplyr::filter(op == "~~" & lhs != rhs) %>%  # Only factor correlations
  dplyr::select(lhs, rhs, est.std) %>%
  arrange(desc(abs(est.std)))  # Sort by magnitude

colnames(interfactor_corr) <- c("Factor 1", "Factor 2", "Standardized Correlation")

# Display interfactor correlations
cat("\nInterfactor Correlations:\n")
```

```
Interfactor Correlations:
```

```r
print(interfactor_corr)
```

```
   Factor.1 Factor.2 Standardized.Correlation
1        f3       f4                    0.697
2        f4       f5                    0.659
3        f1       f6                    0.567
```

4	f3	f5	0.517
5	f5	f6	-0.405
6	f4	f6	-0.394
7	f2	f6	0.349
8	f3	f6	-0.282
9	f1	f5	-0.267
10	f2	f4	-0.212
11	f2	f3	-0.203
12	f1	f3	-0.147
13	f1	f4	-0.121
14	f2	f5	-0.043
15	f1	f2	0.010

Classification Accuracy

```
# Extract manifest variables (indicators) from the fitted model
indicator_names <- lavNames(fit1, type = "ov")

# Select corresponding variables from imputed data
XX <- imp_selected |>
  dplyr::select(all_of(indicator_names))

# Convert to matrix for modeling
X <- as.matrix(XX)
y <- df$death_risk

# Ensure death_risk is a binary factor
df$death_risk <- as.factor(df$death_risk)

# Merge imp_selected with the outcome variable
df_model <- imp_selected %>%
  mutate(death_risk = df$death_risk)

# Split data into training (80%) and testing (20%) sets
set.seed(123)  # For reproducibility
trainIndex <- createDataPartition(df_model$death_risk, p = 0.8, list = FALSE)
train_data <- df_model[trainIndex, ]
test_data <- df_model[-trainIndex, ]

# Fit logistic regression model
logit_model <- glm(death_risk ~ ., data = train_data, family = binomial)
```

```r
# Predict probabilities on test set
pred_probs <- predict(logit_model, test_data, type = "response")

# Convert probabilities to class labels (0.5 threshold)
pred_classes <- factor(ifelse(pred_probs > 0.5, 1, 0), levels = levels(test_data$death_risk))

# Compute classification accuracy
accuracy <- mean(pred_classes == test_data$death_risk)
cat("Classification Accuracy:", round(accuracy, 3), "\n")
```

```
Classification Accuracy: 0.556
```

```r
# Compute AUC (Area Under the Curve)
roc_curve <- roc(test_data$death_risk, pred_probs)
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases
```

```r
auc_value <- auc(roc_curve)
cat("AUC:", round(auc_value, 3), "\n")
```

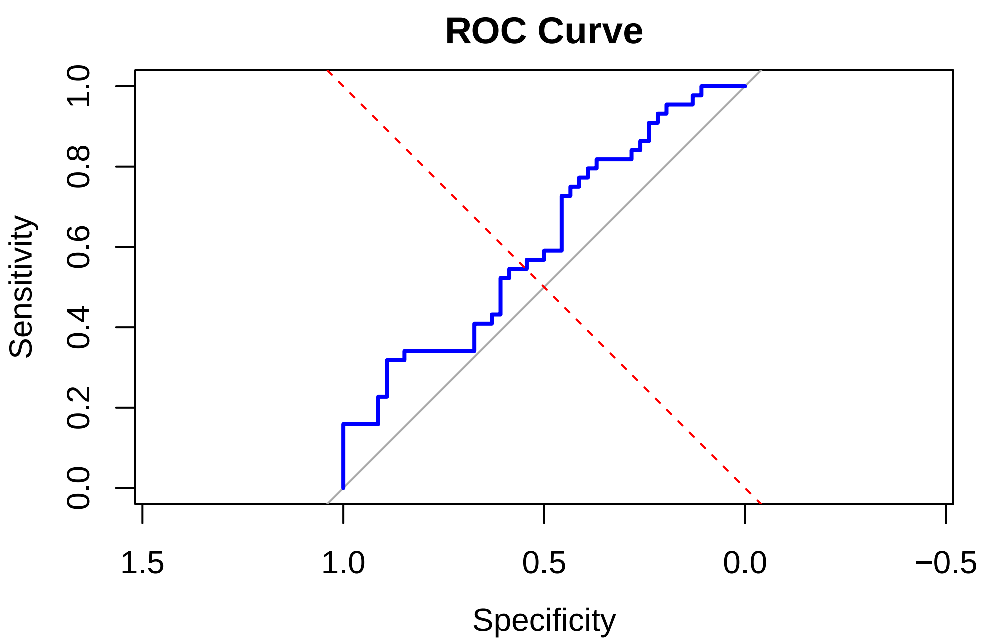
```
AUC: 0.611
```

```r
# Permutation Test: Evaluating Accuracy Against Chance Level

set.seed(123)
null_accuracies <- replicate(1000, {
  shuffled_risk <- sample(train_data$death_risk)  # Shuffle class labels

  # Fit model on shuffled labels
  null_model <- glm(shuffled_risk ~ ., data = train_data, family = binomial)

  # Predict using null model
  null_preds <- predict(null_model, test_data, type = "response")
  null_classes <- factor(ifelse(null_preds > 0.5, 1, 0), levels = levels(test_data$death_ris

  mean(null_classes == test_data$death_risk)  # Compute accuracy on actual test labels
})
```

```
# Compute p-value: Proportion of null models performing as well as or better than the real mo
p_value <- mean(null_accuracies >= accuracy)
cat("P-value for classification above chance:", round(p_value, 4), "\n")
```

```
P-value for classification above chance: 0.332
```

```
# Plot ROC Curve
plot(roc_curve, col = "blue", main = "ROC Curve")
abline(a = 0, b = 1, lty = 2, col = "red")  # Reference line for random classification
```



Remove the worse items

```
selected_items <- c(
  "i_86", "i_57", "i_5", "i_81",
  "i_105", "i_48", "i_133", "i_129", "i_39", "i_103",
  "i_143", "i_79",
  "i_111", "i_34", "i_119", "i_116", "i_23", "i_41",
  "i_186", "i_38", "i_128", "i_7", "i_16",
  "i_137", "i_96", "i_194"
```

```r
)

# Convert to match column names in `imp`
selected_items_corrected <- paste0("item_", sub("^i_", "", selected_items))

# Ensure only existing columns are selected
existing_items <- intersect(selected_items_corrected, colnames(imp))

# Issue a warning if any items are missing
missing_items <- setdiff(selected_items_corrected, existing_items)
if (length(missing_items) > 0) {
  warning("The following items were not found in the dataset: ", paste(missing_items, collaps
}

# Select the matching columns
imp_selected <- imp %>%
  dplyr::select(all_of(existing_items))

# Validate the selection
print(names(imp_selected))
```

```
 [1] "item_86"  "item_57"  "item_5"   "item_81"  "item_105" "item_48"
 [7] "item_133" "item_129" "item_39"  "item_103" "item_143" "item_79"
[13] "item_111" "item_34"  "item_119" "item_116" "item_23"  "item_41"
[19] "item_186" "item_38"  "item_128" "item_7"   "item_16"  "item_137"
[25] "item_96"  "item_194"
```

```r
# Define Target Matrix for Factor Structure (26x6)

# Initialize a 26x6 matrix filled with zeros
TARGET <- matrix(0, nrow = length(existing_items), ncol = 6)

# Assign factor loadings dynamically
factor_assignments <- list(
  F1 = c(1:4),
  F2 = c(5:10),
  F3 = c(11:12),
  F4 = c(13:18),
  F5 = c(19:23),
  F6 = c(24:26)
)
```

```r
# Assign 1s based on factor structure
for (factor in names(factor_assignments)) {
  TARGET[factor_assignments[[factor]], as.numeric(substr(factor, 2, 2))] <- 1
}

# Add row and column names for clarity
rownames(TARGET) <- existing_items
colnames(TARGET) <- paste0("F", 1:6)

# Print the target rotation matrix
print(TARGET)
```

```
         F1 F2 F3 F4 F5 F6
item_86   1  0  0  0  0  0
item_57   1  0  0  0  0  0
item_5    1  0  0  0  0  0
item_81   1  0  0  0  0  0
item_105  0  1  0  0  0  0
item_48   0  1  0  0  0  0
item_133  0  1  0  0  0  0
item_129  0  1  0  0  0  0
item_39   0  1  0  0  0  0
item_103  0  1  0  0  0  0
item_143  0  0  1  0  0  0
item_79   0  0  1  0  0  0
item_111  0  0  0  1  0  0
item_34   0  0  0  1  0  0
item_119  0  0  0  1  0  0
item_116  0  0  0  1  0  0
item_23   0  0  0  1  0  0
item_41   0  0  0  1  0  0
item_186  0  0  0  0  1  0
item_38   0  0  0  0  1  0
item_128  0  0  0  0  1  0
item_7    0  0  0  0  1  0
item_16   0  0  0  0  1  0
item_137  0  0  0  0  0  1
item_96   0  0  0  0  0  1
item_194  0  0  0  0  0  1
```

```r
# Define the ESEM Model in Lavaan Syntax

model <- '
    efa("efa1")*f1 +
    efa("efa1")*f2 +
    efa("efa1")*f3 +
    efa("efa1")*f4 +
    efa("efa1")*f5 +
    efa("efa1")*f6 =~
       # F1
       item_86 + item_57 + item_5 + item_81 +
       # F2
       item_105 + item_48 + item_133 + item_129 + item_39 + item_103 +
       # F3
       item_143 + item_79 +
       # F4
       item_111 + item_34 + item_119 + item_116 + item_23 + item_41 +
       # F5
       item_186 + item_38 + item_128 + item_7 + item_16 +
       # F6
       item_137 + item_96 + item_194
'

# Fit the ESEM Model with Target Rotation
fit2 <- sem(
  model = model,
  data = imp_selected,
  ordered = TRUE,  # Use ordered estimation if Likert-type items
  rotation = "target",
  rotation.args = list(target = TARGET)
)
```

```r
fit2_indices <- fitMeasures(fit2, c(
  "chisq", "df", "pvalue", "cfi", "tli", "rmsea", "srmr",
  "chisq.scaled", "df.scaled", "pvalue.scaled", "cfi.robust", "tli.robust",
  "rmsea.robust", "srmr"
))

# Display selected fit indices (Standard & Robust)
cat("\nModel Fit Indices (Standard & Robust):\n")
```

```
Model Fit Indices (Standard & Robust):
```

```r
fit2_indices_df <- data.frame(
  Measure = names(fit2_indices),
  Value = round(fit2_indices, 3)
)
print(fit2_indices_df)
```

```
          Measure   Value
1           chisq 182.195
2              df 184.000
3          pvalue   0.524
4             cfi   1.000
5             tli   1.000
6           rmsea   0.000
7            srmr   0.035
8    chisq.scaled 391.429
9       df.scaled 184.000
10 pvalue.scaled   0.000
11    cfi.robust   0.930
12    tli.robust   0.877
13  rmsea.robust   0.063
14           srmr   0.035
```

```r
# Extract Standardized Factor Loadings
# Get standardized solution
std_solution <- standardizedSolution(fit2)

# Filter only loadings (Lambda matrix)
std_loadings <- std_solution %>%
  filter(op == "=~") %>%
  dplyr::select(lhs, rhs, est.std) %>%
  arrange(lhs, desc(abs(est.std)))  # Sorted by factor and magnitude

colnames(std_loadings) <- c("Factor", "Item", "Standardized Loading")

# Display standardized loadings
cat("\nStandardized Factor Loadings:\n")
```

```
Standardized Factor Loadings:
```

```r
print(std_loadings)
```

```
   Factor     Item Standardized.Loading
1       f1  item_81                0.562
2       f1  item_86                0.553
3       f1 item_105                0.485
4       f1  item_48                0.474
5       f1  item_57                0.436
6       f1   item_5                0.336
7       f1  item_39                0.327
8       f1  item_38                0.322
9       f1 item_103                0.316
10      f1 item_194               -0.314
11      f1 item_119               -0.255
12      f1 item_137               -0.196
13      f1 item_186                0.157
14      f1   item_7                0.147
15      f1 item_111               -0.139
16      f1  item_23               -0.134
17      f1  item_96               -0.127
18      f1  item_41                0.117
19      f1  item_34                0.106
20      f1 item_116                0.106
21      f1 item_129                0.082
22      f1  item_79               -0.064
23      f1 item_143               -0.059
24      f1 item_128               -0.050
25      f1 item_133                0.026
26      f1  item_16               -0.003
27      f2 item_129                0.890
28      f2 item_133                0.848
29      f2  item_81                0.738
30      f2 item_105                0.632
31      f2  item_48                0.559
32      f2 item_103                0.502
33      f2   item_5                0.475
34      f2  item_39                0.416
35      f2 item_116                0.382
36      f2 item_111                0.298
37      f2  item_86                0.283
38      f2  item_57                0.213
39      f2  item_23                0.195
```

40	f2	item_137	0.192
41	f2	item_96	0.186
42	f2	item_34	0.153
43	f2	item_79	0.123
44	f2	item_119	0.111
45	f2	item_16	-0.107
46	f2	item_38	0.079
47	f2	item_41	-0.077
48	f2	item_7	-0.036
49	f2	item_128	-0.032
50	f2	item_194	-0.032
51	f2	item_186	0.018
52	f2	item_143	-0.002
53	f3	item_143	0.952
54	f3	item_79	0.787
55	f3	item_186	0.227
56	f3	item_194	0.206
57	f3	item_86	-0.176
58	f3	item_116	-0.154
59	f3	item_38	0.143
60	f3	item_105	0.132
61	f3	item_128	-0.129
62	f3	item_39	-0.122
63	f3	item_5	0.122
64	f3	item_16	-0.112
65	f3	item_111	-0.112
66	f3	item_48	0.111
67	f3	item_23	-0.109
68	f3	item_41	-0.084
69	f3	item_7	-0.080
70	f3	item_34	-0.067
71	f3	item_103	-0.066
72	f3	item_133	0.053
73	f3	item_57	-0.051
74	f3	item_96	-0.050
75	f3	item_137	0.018
76	f3	item_119	-0.018
77	f3	item_81	-0.018
78	f3	item_129	0.010
79	f4	item_119	0.837
80	f4	item_34	0.792
81	f4	item_111	0.792
82	f4	item_23	0.750

83	f4	item_41	0.614
84	f4	item_39	0.376
85	f4	item_103	0.357
86	f4	item_116	0.254
87	f4	item_38	-0.250
88	f4	item_194	0.228
89	f4	item_48	0.226
90	f4	item_7	-0.172
91	f4	item_57	-0.138
92	f4	item_143	-0.135
93	f4	item_137	0.116
94	f4	item_96	0.103
95	f4	item_186	-0.077
96	f4	item_129	-0.069
97	f4	item_81	-0.061
98	f4	item_133	0.039
99	f4	item_86	0.033
100	f4	item_16	-0.029
101	f4	item_5	-0.023
102	f4	item_79	0.022
103	f4	item_128	0.021
104	f4	item_105	0.020
105	f5	item_7	1.084
106	f5	item_128	0.906
107	f5	item_16	0.723
108	f5	item_186	0.504
109	f5	item_38	0.455
110	f5	item_86	0.324
111	f5	item_23	-0.323
112	f5	item_57	0.199
113	f5	item_137	0.198
114	f5	item_103	-0.175
115	f5	item_194	0.173
116	f5	item_79	0.123
117	f5	item_111	-0.122
118	f5	item_39	0.121
119	f5	item_119	-0.106
120	f5	item_143	0.102
121	f5	item_34	-0.101
122	f5	item_96	0.092
123	f5	item_5	-0.088
124	f5	item_48	-0.085
125	f5	item_133	-0.077

```
126     f5 item_105                0.073
127     f5  item_81                0.067
128     f5 item_129                0.059
129     f5  item_41               -0.049
130     f5 item_116               -0.028
131     f6 item_137                0.822
132     f6  item_96                0.733
133     f6 item_116                0.488
134     f6 item_194                0.403
135     f6  item_81               -0.275
136     f6  item_23                0.268
137     f6 item_103                0.251
138     f6  item_39                0.207
139     f6  item_86               -0.206
140     f6  item_48               -0.205
141     f6  item_38                0.202
142     f6  item_57               -0.200
143     f6  item_34                0.193
144     f6 item_128                0.157
145     f6   item_7                0.129
146     f6 item_143                0.126
147     f6 item_111               -0.107
148     f6 item_186                0.101
149     f6  item_41               -0.093
150     f6  item_79                0.076
151     f6 item_133                0.071
152     f6  item_16                0.060
153     f6 item_119               -0.028
154     f6   item_5                0.019
155     f6 item_105               -0.008
156     f6 item_129               -0.005
```

```r
# Extract Interfactor Correlations

# Extract standardized correlations (Phi matrix)
interfactor_corr <- std_solution %>%
  dplyr::filter(op == "~~" & lhs != rhs) %>%  # Only factor correlations
  dplyr::select(lhs, rhs, est.std) %>%
  arrange(desc(abs(est.std)))  # Sort by magnitude

colnames(interfactor_corr) <- c("Factor 1", "Factor 2", "Standardized Correlation")
```

```r
# Display interfactor correlations
cat("\nInterfactor Correlations:\n")
```

```
Interfactor Correlations:
```

```r
print(interfactor_corr)
```

```
   Factor.1 Factor.2 Standardized.Correlation
1        f4       f5                    0.710
2        f3       f4                    0.678
3        f3       f5                    0.551
4        f1       f6                    0.497
5        f5       f6                   -0.426
6        f4       f6                   -0.365
7        f2       f6                    0.345
8        f3       f6                   -0.237
9        f2       f3                   -0.202
10       f1       f5                   -0.187
11       f2       f4                   -0.171
12       f1       f2                   -0.049
13       f2       f5                   -0.021
14       f1       f4                    0.016
15       f1       f3                    0.015
```

Combine the first two factors

Also, add additional items to maximize classification accuracy. These are the items that maximally contribute to accurate classification: item_22 item_23 item_26 item_40 item_99 item_121 item_125 item_139 item_157 item_159 item_164 item_187.

```r
selected_items <- c(
  "i_86", "i_57", "i_5", "i_81",
  "i_105", "i_48", "i_133", "i_129", "i_39", "i_103",
  "i_143", "i_79",
  "i_111", "i_34", "i_119", "i_23", "i_41", "i_139",
  "i_186", "i_38", "i_128", "i_7", "i_16", "i_125",
  "i_137", "i_96", "i_194", "i_159"
)
```

```r
# Convert to match column names in `imp`
selected_items_corrected <- paste0("item_", sub("^i_", "", selected_items))

# Ensure only existing columns are selected
existing_items <- intersect(selected_items_corrected, colnames(imp))

# Issue a warning if any items are missing
missing_items <- setdiff(selected_items_corrected, existing_items)
if (length(missing_items) > 0) {
  warning("The following items were not found in the dataset: ", paste(missing_items, collaps
}

# Select the matching columns
imp_selected <- imp %>%
  dplyr::select(all_of(existing_items))

# Validate the selection
print(names(imp_selected))
```

```
 [1] "item_86"  "item_57"  "item_5"   "item_81"  "item_105" "item_48"
 [7] "item_133" "item_129" "item_39"  "item_103" "item_143" "item_79"
[13] "item_111" "item_34"  "item_119" "item_23"  "item_41"  "item_139"
[19] "item_186" "item_38"  "item_128" "item_7"   "item_16"  "item_125"
[25] "item_137" "item_96"  "item_194" "item_159"
```

```r
# Define Target Matrix for Factor Structure

# Determine the number of rows dynamically
num_items <- length(existing_items)
TARGET <- matrix(0, nrow = num_items, ncol = 5)

# Assign factor loadings dynamically
factor_assignments <- list(
  F1 = 1:10,    # Caratteristiche bambino e richieste caregiving
  F2 = 11:12,   # Percezione cura
  F3 = 13:18,   # Fattori intrapsichici
  F4 = 19:24,   # Coping
  F5 = 25:28    # Iperprotezione
)

# Assign 1s based on factor structure
```

```r
for (factor in names(factor_assignments)) {
  TARGET[factor_assignments[[factor]], as.numeric(substr(factor, 2, 2))] <- 1
}

# Add row and column names for clarity
rownames(TARGET) <- existing_items
colnames(TARGET) <- paste0("F", 1:5)

# Print the target rotation matrix
print(TARGET)
```

```
         F1 F2 F3 F4 F5
item_86   1  0  0  0  0
item_57   1  0  0  0  0
item_5    1  0  0  0  0
item_81   1  0  0  0  0
item_105  1  0  0  0  0
item_48   1  0  0  0  0
item_133  1  0  0  0  0
item_129  1  0  0  0  0
item_39   1  0  0  0  0
item_103  1  0  0  0  0
item_143  0  1  0  0  0
item_79   0  1  0  0  0
item_111  0  0  1  0  0
item_34   0  0  1  0  0
item_119  0  0  1  0  0
item_23   0  0  1  0  0
item_41   0  0  1  0  0
item_139  0  0  1  0  0
item_186  0  0  0  1  0
item_38   0  0  0  1  0
item_128  0  0  0  1  0
item_7    0  0  0  1  0
item_16   0  0  0  1  0
item_125  0  0  0  1  0
item_137  0  0  0  0  1
item_96   0  0  0  0  1
item_194  0  0  0  0  1
item_159  0  0  0  0  1
```

```r
# Define the ESEM Model in Lavaan Syntax
model <- '
    efa("efa1")*f1 +
    efa("efa1")*f2 +
    efa("efa1")*f3 +
    efa("efa1")*f4 +
    efa("efa1")*f5 =~
       # F1
       item_86 + item_57 + item_5 + item_81 +
       item_105 + item_48 + item_133 + item_129 + item_39 + item_103 +
       # F2
       item_143 + item_79 +
       # F3
       item_111 + item_34 + item_119 + item_23 + item_41 + item_139 +
       # F4
       item_186 + item_38 + item_128 + item_7 + item_16 + item_125 +
       # F4
       item_137 + item_96 + item_194 + item_159
'

# Fit the ESEM Model with Target Rotation
fit3 <- sem(
  model = model,
  data = imp_selected,
  ordered = TRUE,  # Use ordered estimation if Likert-type items
  rotation = "target",
  rotation.args = list(target = TARGET)
)
```

```r
fit3_indices <- fitMeasures(fit3, c(
  "chisq", "df", "pvalue", "cfi", "tli", "rmsea", "srmr",
  "chisq.scaled", "df.scaled", "pvalue.scaled", "cfi.robust", "tli.robust",
  "rmsea.robust", "srmr"
))

# Display selected fit indices (Standard & Robust)
cat("\nModel Fit Indices (Standard & Robust):\n")
```

```
Model Fit Indices (Standard & Robust):
```

```r
fit3_indices_df <- data.frame(
  Measure = names(fit3_indices),
  Value = round(fit3_indices, 3)
)
print(fit3_indices_df)
```

```
         Measure   Value
1          chisq 342.662
2             df 248.000
3         pvalue   0.000
4            cfi   0.994
5            tli   0.990
6          rmsea   0.029
7           srmr   0.044
8   chisq.scaled 648.584
9      df.scaled 248.000
10 pvalue.scaled   0.000
11    cfi.robust   0.864
12    tli.robust   0.793
13  rmsea.robust   0.081
14          srmr   0.044
```

```r
# Extract Standardized Factor Loadings
# Get standardized solution
std_solution <- standardizedSolution(fit3)

# Filter only loadings (Lambda matrix)
std_loadings <- std_solution %>%
  filter(op == "=~") %>%
  dplyr::select(lhs, rhs, est.std) %>%
  arrange(lhs, desc(abs(est.std)))  # Sorted by factor and magnitude

colnames(std_loadings) <- c("Factor", "Item", "Standardized Loading")

# Display standardized loadings
cat("\nStandardized Factor Loadings:\n")
```

```
Standardized Factor Loadings:
```

```r
print(std_loadings)
```

```
   Factor     Item Standardized.Loading
1      f1  item_81                0.973
2      f1 item_105                0.830
3      f1 item_129                0.805
4      f1  item_48                0.750
5      f1 item_133                0.730
6      f1  item_39                0.705
7      f1 item_103                0.693
8      f1  item_86                0.640
9      f1   item_5                0.577
10     f1  item_57                0.433
11     f1  item_34                0.349
12     f1  item_38                0.314
13     f1   item_7                0.304
14     f1 item_143               -0.295
15     f1  item_41                0.265
16     f1  item_96                0.251
17     f1  item_23                0.211
18     f1 item_194               -0.204
19     f1 item_128                0.193
20     f1 item_111                0.188
21     f1 item_159                0.184
22     f1 item_137                0.172
23     f1  item_79               -0.166
24     f1 item_186                0.160
25     f1 item_139                0.116
26     f1  item_16                0.083
27     f1 item_125                0.072
28     f1 item_119               -0.034
29     f2 item_143                0.746
30     f2  item_79                0.641
31     f2 item_125                0.317
32     f2 item_129               -0.314
33     f2 item_133               -0.279
34     f2 item_186                0.255
35     f2 item_194                0.244
36     f2 item_137                0.239
37     f2 item_139                0.192
38     f2  item_81               -0.185
39     f2  item_38                0.170
```

40	f2	item_96	0.168
41	f2	item_41	0.164
42	f2	item_111	-0.152
43	f2	item_34	0.107
44	f2	item_159	0.096
45	f2	item_86	-0.079
46	f2	item_5	0.056
47	f2	item_128	-0.052
48	f2	item_57	-0.050
49	f2	item_105	0.049
50	f2	item_16	-0.047
51	f2	item_23	0.026
52	f2	item_48	0.016
53	f2	item_119	-0.015
54	f2	item_39	0.013
55	f2	item_103	0.011
56	f2	item_7	-0.006
57	f3	item_41	0.959
58	f3	item_119	0.787
59	f3	item_111	0.768
60	f3	item_139	0.767
61	f3	item_34	0.657
62	f3	item_23	0.609
63	f3	item_48	0.359
64	f3	item_39	0.344
65	f3	item_38	-0.299
66	f3	item_79	0.227
67	f3	item_103	0.214
68	f3	item_159	-0.184
69	f3	item_7	-0.161
70	f3	item_133	0.149
71	f3	item_194	0.117
72	f3	item_81	0.111
73	f3	item_105	0.093
74	f3	item_186	-0.092
75	f3	item_143	0.088
76	f3	item_57	-0.086
77	f3	item_129	0.075
78	f3	item_16	-0.063
79	f3	item_137	-0.054
80	f3	item_5	0.051
81	f3	item_125	-0.049
82	f3	item_86	0.047

83	f3	item_128	-0.019
84	f3	item_96	-0.011
85	f4	item_7	1.058
86	f4	item_128	0.878
87	f4	item_125	0.742
88	f4	item_16	0.698
89	f4	item_186	0.670
90	f4	item_38	0.548
91	f4	item_86	0.427
92	f4	item_23	-0.343
93	f4	item_143	0.317
94	f4	item_79	0.305
95	f4	item_57	0.291
96	f4	item_105	0.259
97	f4	item_41	-0.256
98	f4	item_81	0.254
99	f4	item_139	-0.232
100	f4	item_96	-0.155
101	f4	item_129	0.145
102	f4	item_194	0.138
103	f4	item_48	0.114
104	f4	item_39	0.106
105	f4	item_159	-0.104
106	f4	item_111	-0.080
107	f4	item_137	-0.066
108	f4	item_34	-0.061
109	f4	item_119	-0.059
110	f4	item_103	-0.032
111	f4	item_5	0.017
112	f4	item_133	-0.010
113	f5	item_137	0.665
114	f5	item_159	0.586
115	f5	item_96	0.463
116	f5	item_194	0.455
117	f5	item_139	-0.430
118	f5	item_143	0.364
119	f5	item_41	-0.364
120	f5	item_79	0.337
121	f5	item_133	0.325
122	f5	item_86	-0.248
123	f5	item_23	0.219
124	f5	item_81	-0.197
125	f5	item_129	0.194

```
126     f5 item_119                0.180
127     f5  item_57               -0.174
128     f5 item_111                0.140
129     f5   item_7               -0.125
130     f5  item_48               -0.119
131     f5 item_103                0.107
132     f5  item_34                0.107
133     f5  item_16               -0.099
134     f5  item_39                0.077
135     f5 item_128               -0.056
136     f5   item_5                0.049
137     f5  item_38                0.044
138     f5 item_186                0.032
139     f5 item_105                0.030
140     f5 item_125                0.015
```

```r
# Extract Interfactor Correlations

# Extract standardized correlations (Phi matrix)
interfactor_corr <- std_solution %>%
  dplyr::filter(op == "~~" & lhs != rhs) %>%  # Only factor correlations
  dplyr::select(lhs, rhs, est.std) %>%
  arrange(desc(abs(est.std)))  # Sort by magnitude

colnames(interfactor_corr) <- c("Factor 1", "Factor 2", "Standardized Correlation")

# Display interfactor correlations
cat("\nInterfactor Correlations:\n")
```

```

Interfactor Correlations:
```

```r
print(interfactor_corr)
```

```
   Factor.1 Factor.2 Standardized.Correlation
1        f3       f4                    0.751
2        f1       f3                   -0.451
3        f1       f4                   -0.424
4        f1       f5                    0.335
5        f1       f2                    0.235
6        f2       f5                   -0.225
```

7	f4	f5	0.128
8	f2	f4	-0.071
9	f3	f5	0.031
10	f2	f3	-0.010

Classification Accuracy

```
# Get observed variable names from the fitted model
indicator_names <- lavNames(fit3, type = "ov")

# Select corresponding variables from imputed dataset
XX <- imp_selected |>
  dplyr::select(all_of(indicator_names))

# Convert to matrix for modeling
X <- as.matrix(XX)
y <- df$death_risk  # Outcome variable

# Ensure death_risk is a binary factor
df$death_risk <- factor(df$death_risk)

# Merge imputed data with the criterion variable
df_model <- imp_selected %>%
  mutate(death_risk = df$death_risk)

# Validate the factor levels
if (length(levels(df_model$death_risk)) != 2) {
  stop("Error: The death_risk variable must be binary.")
}

# Train-Test Split
set.seed(123)  # For reproducibility
trainIndex <- createDataPartition(df_model$death_risk, p = 0.8, list = FALSE)
train_data <- df_model[trainIndex, ]
test_data <- df_model[-trainIndex, ]
```

```
# Fit Logistic Regression Model
logit_model <- glm(death_risk ~ ., data = train_data, family = binomial)

# Predict probabilities on test set
pred_probs <- predict(logit_model, test_data, type = "response")
```

```r
# Convert probabilities to class labels using dynamic factor levels
threshold <- 0.5
pred_classes <- factor(ifelse(pred_probs > threshold, levels(df$death_risk)[2], levels(df$dea
                       levels = levels(df$death_risk))

# Compute classification accuracy
accuracy <- mean(pred_classes == test_data$death_risk, na.rm = TRUE)
cat("Classification Accuracy:", round(accuracy, 3), "\n")
```

```
Classification Accuracy: 0.622
```

```r
# Compute AUC (Area Under the Curve)
roc_curve <- roc(test_data$death_risk, pred_probs)
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases
```

```r
auc_value <- auc(roc_curve)
cat("AUC:", round(auc_value, 3), "\n")
```

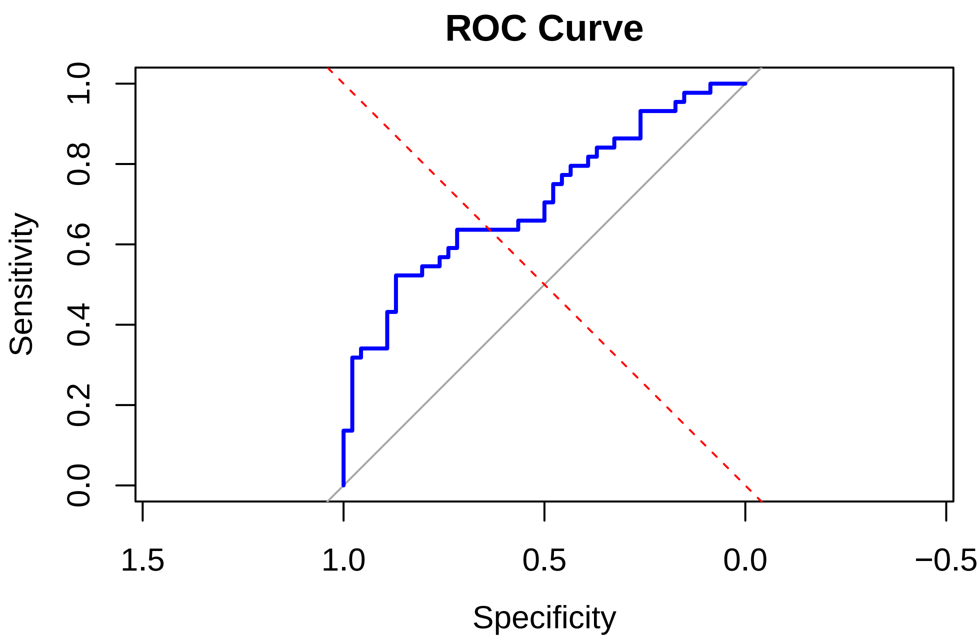
```
AUC: 0.71
```

```r
# Permutation Test: Evaluating Accuracy Against Chance Level
set.seed(123)
null_accuracies <- replicate(1000, {
  shuffled_risk <- sample(train_data$death_risk)  # Shuffle class labels

  # Fit model on shuffled labels
  null_model <- glm(shuffled_risk ~ ., data = train_data, family = binomial)

  # Predict using null model
  null_preds <- predict(null_model, test_data, type = "response")
  null_classes <- factor(ifelse(null_preds > threshold, levels(df$death_risk)[2], levels(df$d
                         levels = levels(df$death_risk))

  mean(null_classes == test_data$death_risk, na.rm = TRUE)  # Compute accuracy on actual tes
})
```

```r
# Compute p-value: Proportion of null models performing as well as or better than the real mo
p_value <- mean(null_accuracies >= accuracy)
cat("P-value for classification above chance:", round(p_value, 4), "\n")
```

```
P-value for classification above chance: 0.123
```

```r
# Plot ROC Curve
plot(roc_curve, col = "blue", main = "ROC Curve")
abline(a = 0, b = 1, lty = 2, col = "red")  # Reference line for random classification
```



Interpreting the Results

Classification Accuracy:

Measures the percentage of correct classifications in the test set. If accuracy > 0.70, the model has reasonably good predictive power.

AUC (Area Under the Curve):

AUC = 0.5: Model performs at chance level (random guessing). AUC > 0.7: Good discrimination between positive/negative cases. AUC 1: Excellent classification performance.

Permutation Test (p-value):

Tests whether the classifier performs significantly better than chance. If $p < 0.05$: The model significantly outperforms random classification. If p 0.5: The model is not better than random guessing.