Modelos Selecionados

```
constants = yaml.load_file("constants.yaml")

categorical_columns = constants$categorical_columns

daily_columns = constants$daily_columns

numerical_columns = constants$numerical_columns

solutos_columns = constants$solutos_columns

all_columns = c(numerical_columns, categorical_columns)

original_columns = constants$original_columns

derived_columns = setdiff(daily_columns, original_columns)
```

```
df = readRDS('./data/processed_data_mean_v2.rds') %>%
    mutate_at(categorical_columns, list(~factor(.)))

df_names = readxl::read_excel('./data/Nomes das variaveis.xlsx') %>%
  mutate(variavel = tolower(variavel),
         nome = coalesce(nome, variavel)) %>%
  select(nome, variavel)
```

```
selected_columns_yaml = yaml.load_file("selected_columns.yaml")

selected_numerical_columns = selected_columns_yaml$selected_numerical_columns

selected_categorical_columns = selected_columns_yaml$selected_categorical_columns

drop_scr = TRUE

target = "ira"

to_camila = FALSE
```

```
if (drop_scr){
  selected_numerical_columns = setdiff(selected_numerical_columns, 'scr')
}
```

```
gilberto_diagnostics = function(fit.model){
  X <- model.matrix(fit.model)
  n <- nrow(X)
  p <- ncol(X)
  w <- fit.model$weights
  W <- diag(w)
  H <- solve(t(X)%*%W%*%X)
```

```
  H <- sqrt(W)%*%X%*%H%*%t(X)%*%sqrt(W)
  h <- diag(H)
  ts <- resid(fit.model,type="pearson")/sqrt(1-h)
  td <- resid(fit.model,type="deviance")/sqrt(1-h)
  di <- (h/(1-h))*(ts^2)
  a <- max(td)
  b <- min(td)

  df1 = data.frame(fitted = fitted(fit.model), h = h, indice = df$numero)

  p1 = ggplot(df1) +
    geom_point(aes(x = fitted, y = h)) +
    geom_text(data = df1 %>% top_n(h, n = 3),
              aes(x = fitted, y = h, label = indice),
              hjust = -0.1, vjust = 0) +
    labs(x = "Valor Ajustado", y = "Medida h") +
    theme_bw()

  df2 = data.frame(di = di, index = seq(1, length(di)), indice = df$numero)

  p2 = ggplot(df2) +
    geom_point(aes(x = index, y = di)) +
    geom_text(data = df2 %>% top_n(di, n = 3),
              aes(x = index, y = di, label = indice),
              hjust = -0.1, vjust = 0) +
    labs(x = "Índice", y = "Distância de Cook") +
    theme_bw()

  df3 = data.frame(td = td, index = seq(1, length(td)), indice = df$numero)

  p3 = ggplot(df3) +
    geom_point(aes(x = index, y = td)) +
    geom_text(data = df3 %>% filter(td < -2 | td > 2),
              aes(x = index, y = td, label = indice),
              hjust = -0.1, vjust = 0) +
    labs(x = "Índice", y = "Resíduo Componente do Desvio") +
    geom_hline(yintercept = 2, linetype = 'dotted') +
    geom_hline(yintercept = -2, linetype = 'dotted') +
    theme_bw()

  df4 = data.frame(fitted = fitted(fit.model), td = td)

  p4 = ggplot(df4) +
    geom_point(aes(x = fitted, y = td)) +
    labs(x = "Valor Ajustado", y = "Resíduo Componente do Desvio") +
    theme_bw()

  print(p1)
  print(p2)
  print(p3)
  print(p4)
}
```

```r
envelope_simulado = function(model, tol = 1e-22){
  X <- model.matrix(model)
  n <- nrow(X)
  p <- ncol(X)
  w <- model$weights
  W <- diag(w)
  H <- solve(t(X)%*%W%*%X, tol = tol)
  H <- sqrt(W)%*%X%*%H%*%t(X)%*%sqrt(W)
  h <- diag(H)
  td <- resid(model,type="deviance") / sqrt(1 - h)
  e <- matrix(0, n, 100)
  #
  for(i in 1:100){
    dif <- runif(n) - fitted(model)
    dif[dif >= 0 ] <- 0
    dif[dif < 0] <- 1
    nresp <- dif
    fit <- glm(nresp ~ X, family = binomial)
    w <- fit$weights
    W <- diag(w)
    H <- solve(t(X)%*%W%*%X, tol = tol)
    H <- sqrt(W)%*%X%*%H%*%t(X)%*%sqrt(W)
    h <- diag(H)
    e[,i] <- sort(resid(fit, type = "deviance") / sqrt(1 - h))
  }
  #
  e1 <- numeric(n)
  e2 <- numeric(n)
  #
  for(i in 1:n){
    eo <- sort(e[i,])
    e1[i] <- (eo[2] + eo[3]) / 2
    e2[i] <- (eo[97] + eo[98]) / 2
  }

  med <- apply(e, 1, mean)

  p = ggplot(data.frame(td = td,
                        e1 = e1,
                        e2 = e2,
                        med = med)) +
    stat_qq(aes(sample = td)) +
    stat_qq(aes(sample = e1),
            geom = 'line') +
    stat_qq(aes(sample = e2),
            geom = 'line') +
    stat_qq(aes(sample = med),
            geom = 'line',
            linetype = 'dotted') +
    labs(x = "Percentil da N(0,1)", y = "Componente do Desvio") +
    theme_bw()

  print(p)
```

```
}
```

```
fit_model = function(df,
                     numerical_features,
                     categorical_features,
                     target,
                     include_interaction = FALSE){
  factor_formula = paste(sprintf("factor(%s)",
                                 categorical_features),
                         collapse = " + ")
  numerical_formula = paste(numerical_features, collapse = " + ")
  interaction_formula = ""

  if('causa_ira' %in% c(categorical_features,
                        numerical_features) & include_interaction){
    solutos_features = intersect(numerical_features, solutos_columns)
    if(length(solutos_features > 0)){
      interaction_formula = paste(paste0("factor(causa_ira) : ",
                                         solutos_features),
                                  collapse = " + ")
    }
  }

  formulas_vetor = c(factor_formula, numerical_formula, interaction_formula)

  formula_string = sprintf("%s ~ %s", target,
                           paste(formulas_vetor[formulas_vetor!=""], collapse = " + "))
  formula = formula_string %>% as.formula

  model <- glm(formula, data = df, family = "binomial")
  return(list(model = model, formula = formula))
}

fit_multiple_models = function(df, numerical_features, categorical_features,
                               target, include_interaction=FALSE, trace=0){
  full_model <- fit_model(df, numerical_features, categorical_features,
                          target, include_interaction=include_interaction)

  null_model = glm(ira ~ 1, data = df,
                   family = "binomial")

  backwards = step(full_model, trace = trace)

  forwards = step(null_model,
                  scope = list(lower = formula(null_model),
                               upper = formula(full_model)),
                  direction = "forward", trace = trace)

  stepwise = step(null_model,
                  list(lower = formula(null_model),
                       upper = formula(full_model)),
                  direction = "both", trace = trace)

  features_union = c(all.vars(backwards$formula[-2]),
```

```
                    all.vars(forwards$formula[-2]),
                    all.vars(stepwise$formula[-2])) %>%
    unique

  features_intersection = intersect(intersect(all.vars(backwards$formula[-2]),
                                              all.vars(forwards$formula[-2])),
                                    all.vars(stepwise$formula[-2]))

  union_model = fit_model(df, features_union, c(),
                          target, include_interaction = include_interaction)
  intersection_model = fit_model(df, features_intersection, c(),
                                 target, include_interaction = include_interaction)

  return(list("Full model" = full_model,
              "Backwards" = backwards,
              "Forwards" = forwards,
              "Stepwise" = stepwise,
              "Union model" = union_model,
              "Intersection model" = intersection_model))

}

model_summary = function(df, model){
  columns = all.vars(model$formula[-2])

  options(xtable.comment = FALSE)

  model %>%
    tidy() %>%
    rename(Termo = term,
           Estimativa = estimate,
           `Erro padrão` = std.error,
           Estatística = statistic,
           `p-valor` = p.value) %>%
    niceFormatting(caption = "Resumo do modelo") %>%
    print

  model %>%
    glance() %>%
    select(AIC, deviance, df.residual, nobs) %>%
    rename(`Graus de liberdade do resíduo` = df.residual,
           `Número de observações` = nobs) %>%
    niceFormatting(caption = "Detalhes do modelo") %>%
    print

  confint(model) %>%
    exp %>%
    as_tibble(rownames = 'Variável') %>%
    mutate(Estimativa = exp(model$coefficients)) %>%
    select(Variável, Estimativa, everything()) %>%
    niceFormatting(caption = "Intervalo de confiança para a razão de chances") %>%
    print
```

```
  if (length(coef(model)) - 1 > 1){
    car::vif(model) %>%
      as_tibble(rownames = 'Variável') %>%
      rename(Valor = value) %>%
      niceFormatting(caption = "VIF") %>%
      print
  }

  df$prob = predict(model, type = "response")
  g = roc(ira ~ prob, data = df, auc=TRUE)
  confusion = coords(g, x = "best", best.method="youden",
                     ret = c("threshold",
                             "sensitivity", "specificity",
                             "tp", "tn", "fn", "fp"))

  p = ggroc(g, legacy.axes = TRUE)+
    geom_point(aes(x = 1 - confusion$specificity, y=confusion$sensitivity),
               colour="blue", size=5) +
    geom_abline(slope = 1, intercept=0, linetype="longdash") +
    labs(subtitle = paste("AUC:", round(g$auc, 3)),
         x = '1 - Especificidade', y = 'Sensibilidade')

  return(g)
}

model_diagnostics = function(df, model, run_envelope = FALSE){

  gilberto_diagnostics(model)

  if (run_envelope) envelope_simulado(model)

  probabilities <- predict(model, type = "response")
  columns = intersect(all.vars(model$formula[-2]), numerical_columns)

  df_2 = df %>%
    dplyr::select(all_of(columns)) %>%
    mutate(logit = log(probabilities/(1-probabilities))) %>%
    gather(key = "predictors", value = "predictor.value", -logit)

  p1 = ggplot(df_2, aes(logit, predictor.value))+
    geom_point(size = 0.5, alpha = 0.5) +
    geom_smooth(method = "loess") +
    theme_bw() +
    facet_wrap(~ predictors, scales = "free_y", ncol = 2) +
    labs(y = 'Valor do preditor', x = 'Logaritmo da razão de chances')

  model.data <- augment(model) %>%
    mutate(index = 1:n())

  p2 = ggplot(model.data, aes(index, .std.resid)) +
    geom_point(aes(color = ira), alpha = .5) +
    theme_bw()
```

```
  print(p1)
}


niceFormatting = function(df, caption=""){
  df %>%
    kbl(booktabs = T, longtable = T, caption = caption, digits = 3) %>%
    kable_styling(latex_options = c("striped", "HOLD_position", "repeat_header"))
}

model_selection = function(df, numerical_columns, categorical_columns,
                           target, trace = 0, include_interaction = FALSE){

  fitted_models = fit_multiple_models(df,
                                      numerical_columns,
                                      categorical_columns,
                                      target,
                                      include_interaction = include_interaction,
                                      trace = trace)

  for (model in names(fitted_models)){
    cat("### ", model, "\n")

    model_summary(df, fitted_models[[model]])

    if (!model %in% c('Full model', 'Union model')) {
      model_diagnostics(df, fitted_models[[model]], run_envelope = TRUE)
    } else {
    model_diagnostics(df, fitted_models[[model]])
    }

    cat("\n")
    cat("\n")
    cat("\\newpage")
  }
  return(fitted_models)
}

transpose_df <- function(df) {
  t_df <- data.table::transpose(df)
  colnames(t_df) <- rownames(df)
  rownames(t_df) <- colnames(df)
  t_df <- t_df %>%
    tibble::rownames_to_column(.data = .) %>%
    tibble::as_tibble(.)
  return(t_df)
}

loocv = function(df, formula){
  train.control <- trainControl(method = "LOOCV",
                                summaryFunction = twoClassSummary,
                                classProbs=T,
                                savePredictions = T)
```

```
  df_cv = df %>%
    mutate(ira = if_else(ira == 0, "no", "yes"))

  model <- train(formula,
                 data = df_cv,
                 method = "glm",
                 family = "binomial",
                 trControl = train.control,
                 metric = "Sens",
                 maximize = TRUE)

  g = roc(obs ~ yes, data = model$pred, auc=TRUE)

  youden = coords(g, x = 'best', best.method="youden",
                  ret = c("threshold", "specificity", "sensitivity"))

  confusion = coords(g,
                     ret = c("threshold",
                             "sensitivity", "specificity",
                             "tp", "tn", "fn", "fp"))

  youden_index = confusion %>%
    mutate(index = row_number()) %>%
    filter(threshold == youden$threshold) %>%
    .$index

  confusion %>%
    rename(`Ponto de corte` = threshold,
           Especificidade = specificity,
           Sensibilidade = sensitivity,
           `Verdadeiro positivo` = tp,
           `Verdadeiro negativo` = tn,
           `Falso positivo` = fp,
           `Falso negativo` = fn) %>%
    niceFormatting(caption = sprintf("Youden LOOCV threshold = %.3f", youden$threshold)) %>%
    row_spec(youden_index, bold = T, color = "#D7261E") %>%
    kable_styling(font_size = 7) %>%
    column_spec(1:7, width = "2cm") %>%
    print

  p = ggroc(g, legacy.axes = TRUE)+
    geom_point(aes(x = 1 - youden$specificity,
                   y = youden$sensitivity),
               colour="blue",
               alpha = 0.1,
               size=3) +
    geom_abline(slope = 1, intercept=0, linetype="longdash") +
    labs(subtitle = paste("AUC:", round(g$auc, 3)),
         x = '1 - Especificidade', y = 'Sensibilidade')

  print(p)
  return(g)
}
```

```r
get_name = function(column) df_names %>% filter(variavel == column) %>% .$nome
```

```r
df = df %>%
  mutate(row_number = row_number()) %>%
  filter(row_number != 35)

df_summary = tibble()
```

```r
variaveis = c('sofa', 'una', 'scr_basal')

cat(paste("# IRA ~ ", paste(lapply(variaveis, get_name), collapse = " + "), "\n \n"))
```

IRA ~ Índice SOFA + Sódio urinário + Creatinina sérica basal

```r
model_fit = fit_model(df, variaveis, categorical_features = c(), target = target)

g = model_summary(df, model_fit$model)
```

Table 1: Resumo do modelo

Termo	Estimativa	Erro padrão	Estatística	p-valor
(Intercept)	-3.638	1.510	-2.409	0.016
sofa	0.778	0.252	3.090	0.002
una	-0.009	0.006	-1.486	0.137
scr_basal	3.508	1.714	2.047	0.041

Table 2: Detalhes do modelo

AIC	deviance	Graus de liberdade do resíduo	Número de observações
55.499	47.499	47	51

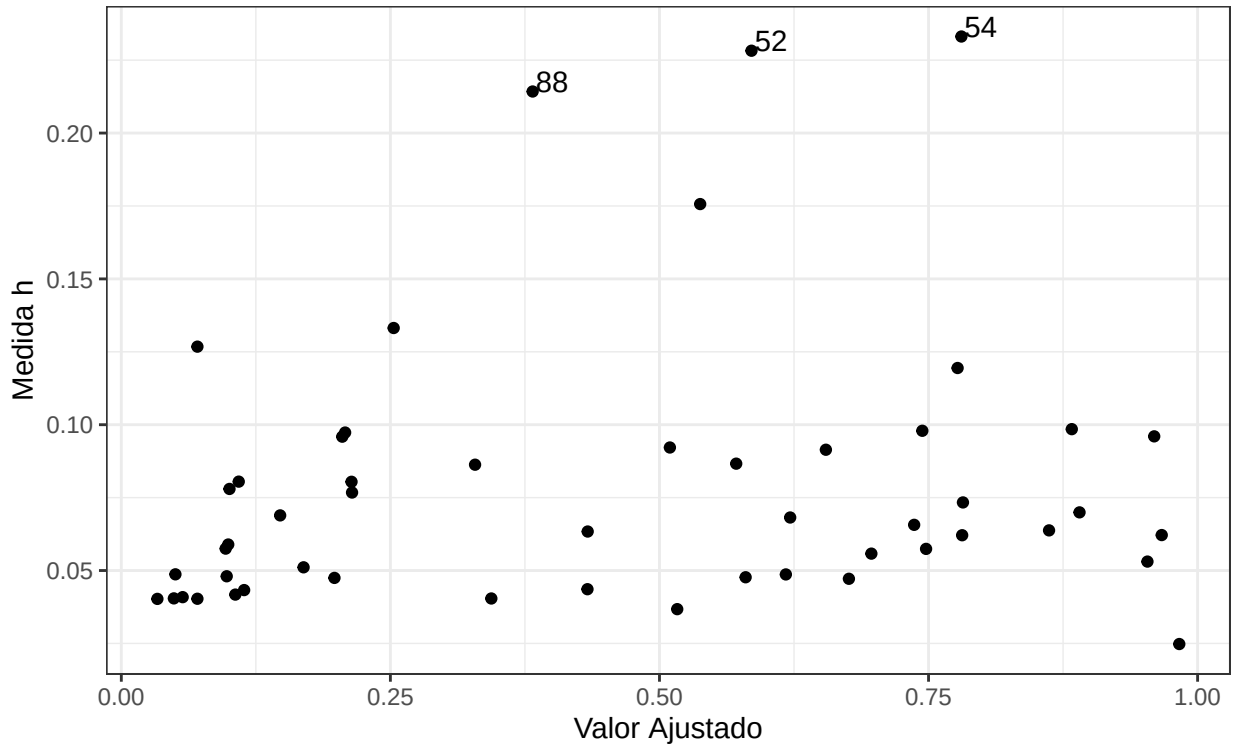
Table 3: Intervalo de confiança para a razão de chances

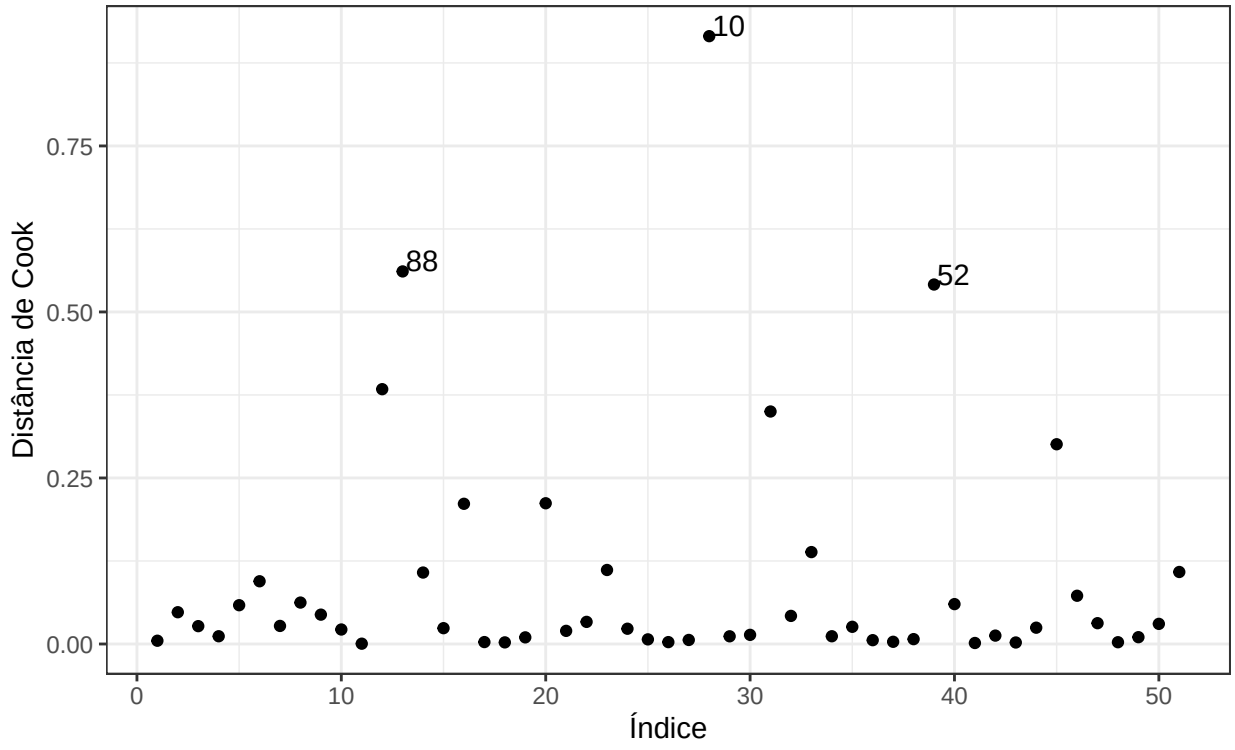
Variável	Estimativa	2.5 %	97.5 %
(Intercept)	0.026	0.001	0.373
sofa	2.177	1.407	3.856
una	0.991	0.979	1.002
scr_basal	33.397	1.663	1567.448

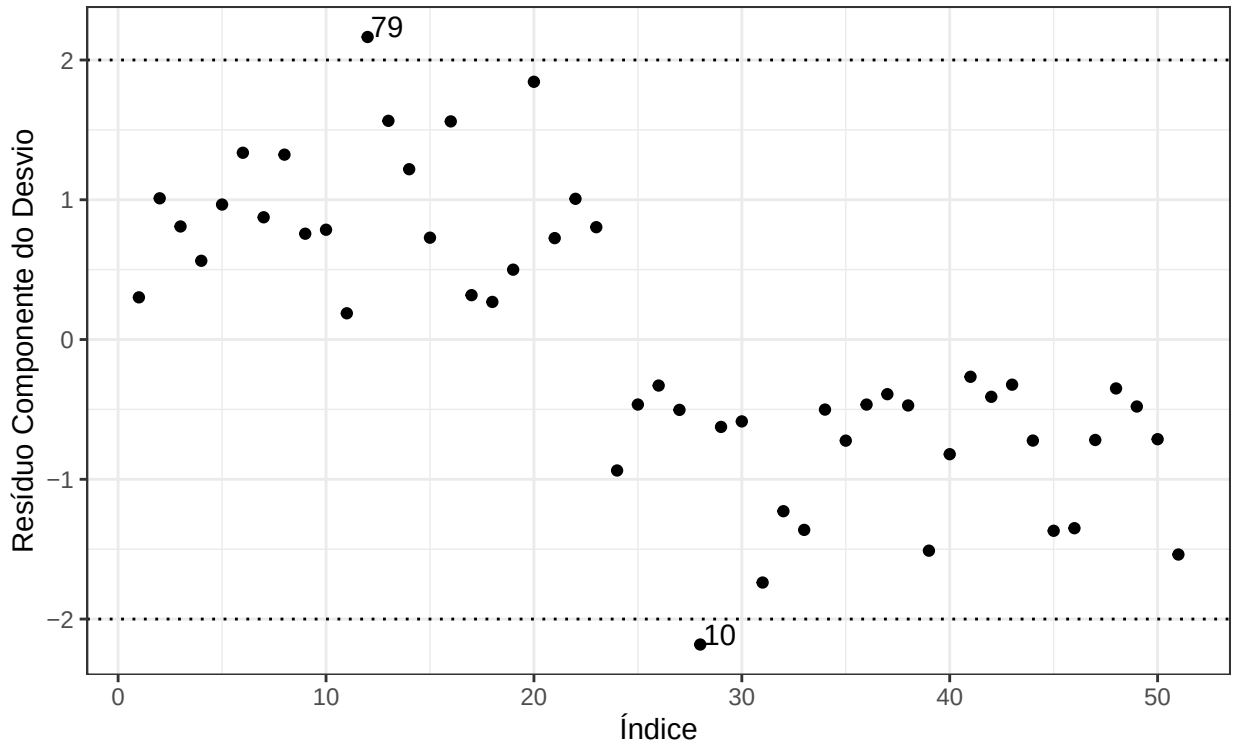
Table 4: VIF

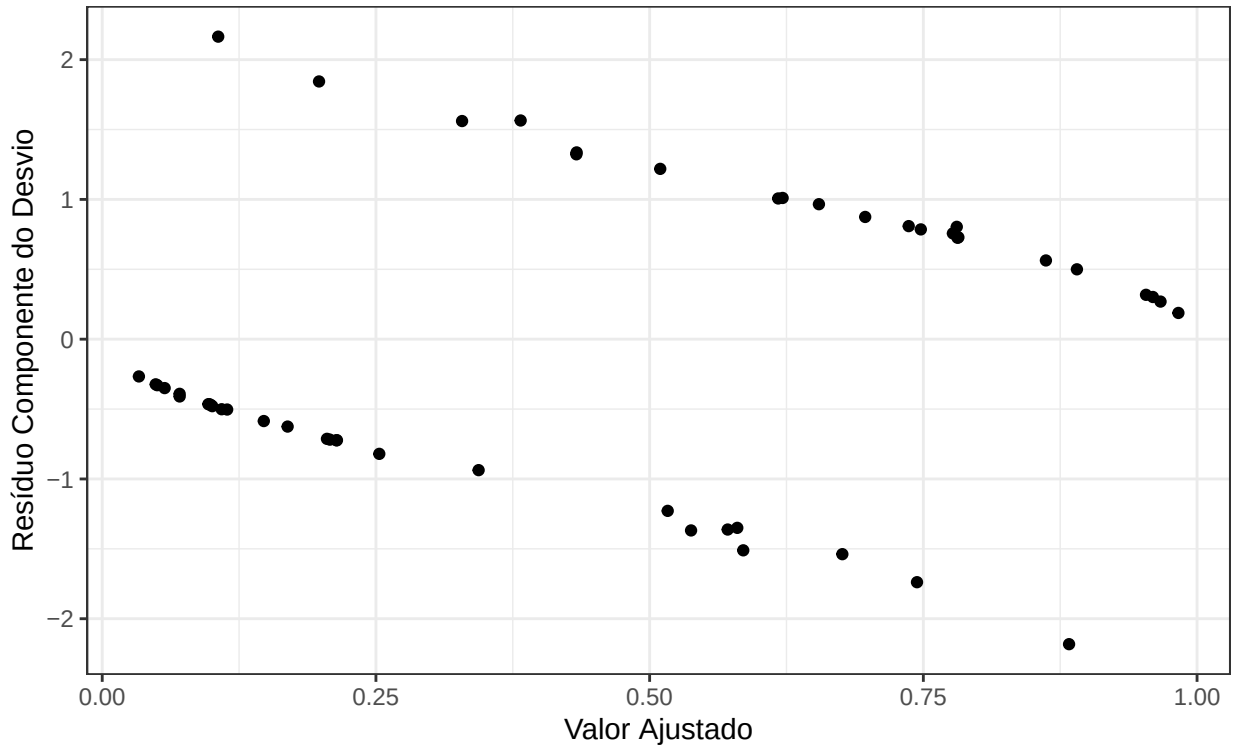
Variável	Valor
sofa	1.069
una	1.024
scr_basal	1.091

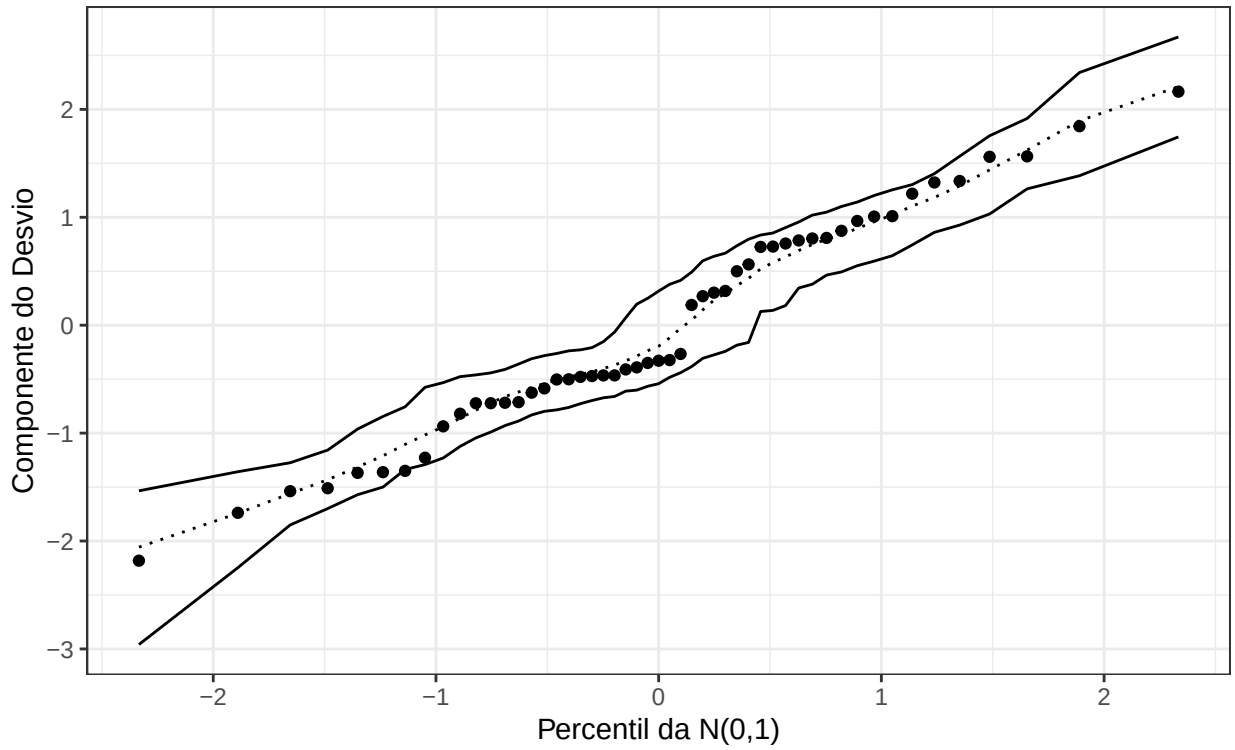
```r
if(!to_camila) model_diagnostics(df, model_fit$model, run_envelope = TRUE)
```

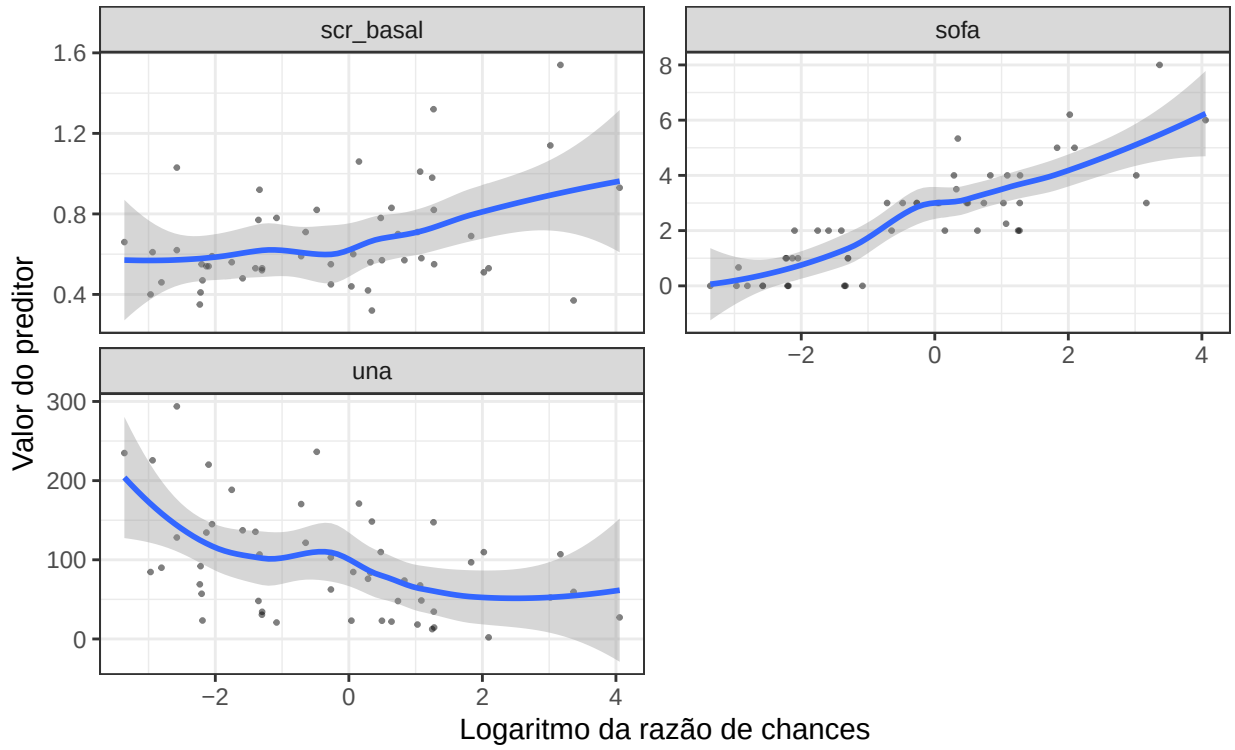





79
10
Resíduo Componente do Desvio
Índice
2
1
0
−1
−2
0
10
20
30
40
50


Resíduo Componente do Desvio
Valor Ajustado
2
1
0
−1
−2
0.00
0.25
0.50
0.75
1.00


Componente do Desvio
Percentil da N(0,1)

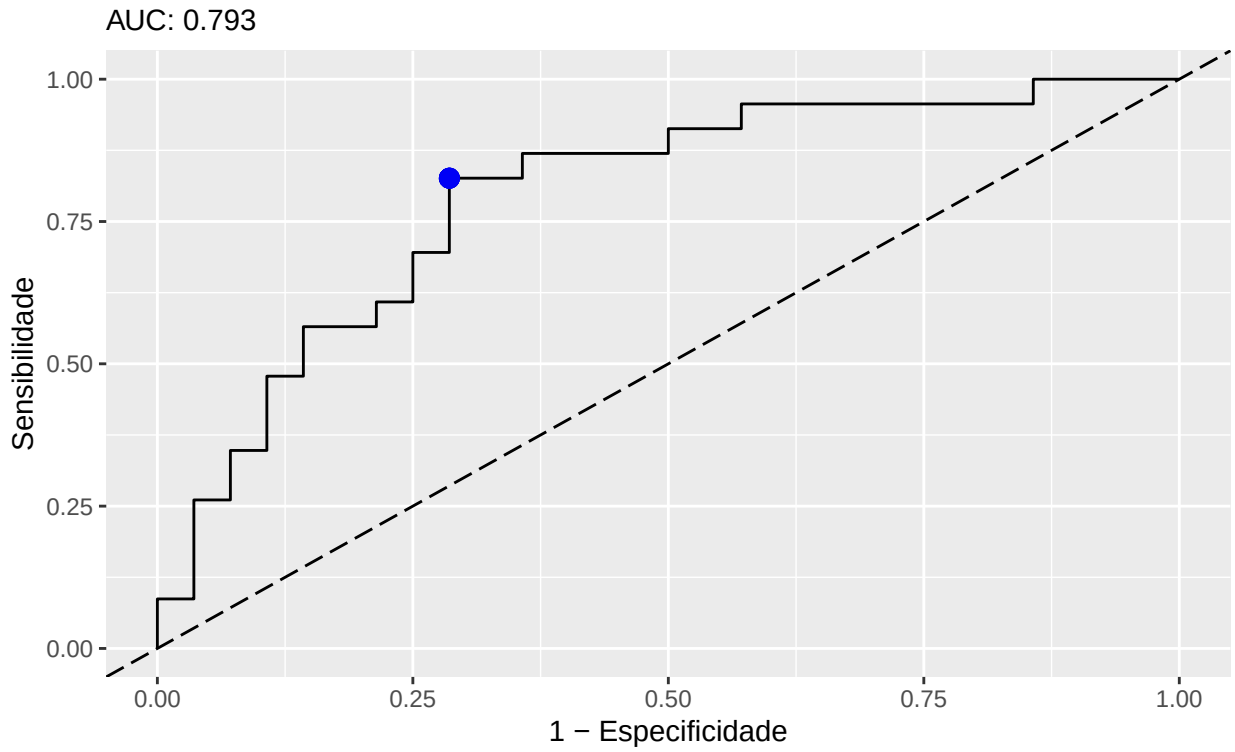

scr_basal
sofa
una
Valor do preditor
Logaritmo da razão de chances

```
cat("\\newpage")
```

```
g_cv = loocv(df, model_fit$formula)
```

Table 5: Youden LOOCV threshold = 0.377

Ponto de corte	Sensibilidade	Especificidade	Verdadeiro positivo	Verdadeiro negativo	Falso negativo	Falso positivo
-Inf	1.000	0.000	23	0	0	28
0.043	1.000	0.036	23	1	0	27
0.052	1.000	0.071	23	2	0	26
0.056	1.000	0.107	23	3	0	25
0.064	1.000	0.143	23	4	0	24
0.071	0.957	0.143	22	4	1	24
0.078	0.957	0.179	22	5	1	23
0.092	0.957	0.214	22	6	1	22
0.103	0.957	0.250	22	7	1	21
0.104	0.957	0.286	22	8	1	20
0.107	0.957	0.321	22	9	1	19
0.114	0.957	0.357	22	10	1	18
0.119	0.957	0.393	22	11	1	17
0.139	0.957	0.429	22	12	1	16
0.158	0.913	0.429	21	12	2	16
0.169	0.913	0.464	21	13	2	15
0.201	0.913	0.500	21	14	2	14
0.225	0.870	0.500	20	14	3	14
0.229	0.870	0.536	20	15	3	13
0.232	0.870	0.571	20	16	3	12
0.232	0.870	0.607	20	17	3	11
0.249	0.870	0.643	20	18	3	10
0.279	0.826	0.643	19	18	4	10
0.325	0.826	0.679	19	19	4	9
0.377	**0.826**	**0.714**	**19**	**20**	**4**	**8**
0.401	0.783	0.714	18	20	5	8
0.434	0.739	0.714	17	20	6	8
0.498	0.696	0.714	16	20	7	8
0.565	0.696	0.750	16	21	7	7
0.596	0.652	0.750	15	21	8	7
0.604	0.609	0.750	14	21	9	7
0.615	0.609	0.786	14	22	9	6
0.623	0.565	0.786	13	22	10	6
0.640	0.565	0.821	13	23	10	5
0.666	0.565	0.857	13	24	10	4
0.694	0.522	0.857	12	24	11	4
0.710	0.478	0.857	11	24	12	4
0.714	0.478	0.893	11	25	12	3
0.725	0.435	0.893	10	25	13	3
0.739	0.391	0.893	9	25	14	3
0.748	0.348	0.893	8	25	15	3
0.757	0.348	0.929	8	26	15	2
0.766	0.304	0.929	7	26	16	2
0.795	0.261	0.929	6	26	17	2
0.838	0.261	0.964	6	27	17	1
0.867	0.217	0.964	5	27	18	1
0.916	0.174	0.964	4	27	19	1
0.953	0.130	0.964	3	27	20	1
0.956	0.087	0.964	2	27	21	1
0.961	0.087	1.000	2	28	21	0
0.973	0.043	1.000	1	28	22	0
Inf	0.000	1.000	0	28	23	0




```
df_summary = model_fit$model %>%
  glance %>%
  mutate(`Variáveis` =  paste(lapply(variaveis, get_name), collapse = ", "),
         AUC = g$auc %>% as.numeric,
         `CV AUC` = g_cv$auc %>% as.numeric) %>%
  bind_rows(df_summary)
```

```r
variaveis = c('sofa', 'su', 'scr_basal')

cat(paste("# IRA ~ ", paste(lapply(variaveis, get_name), collapse = " + "), "\n \n"))
```

IRA ~ Índice SOFA + Uréia sérica + Creatinina sérica basal

```r
model_fit = fit_model(df, variaveis, categorical_features = c(), target = target)

g = model_summary(df, model_fit$model)
```

Table 6: Resumo do modelo

Termo	Estimativa	Erro padrão	Estatística	p-valor
(Intercept)	-5.313	1.691	-3.141	0.002
sofa	0.788	0.259	3.046	0.002
su	0.031	0.019	1.604	0.109
scr_basal	3.280	1.748	1.877	0.061

Table 7: Detalhes do modelo

AIC	deviance	Graus de liberdade do resíduo	Número de observações
54.837	46.837	47	51

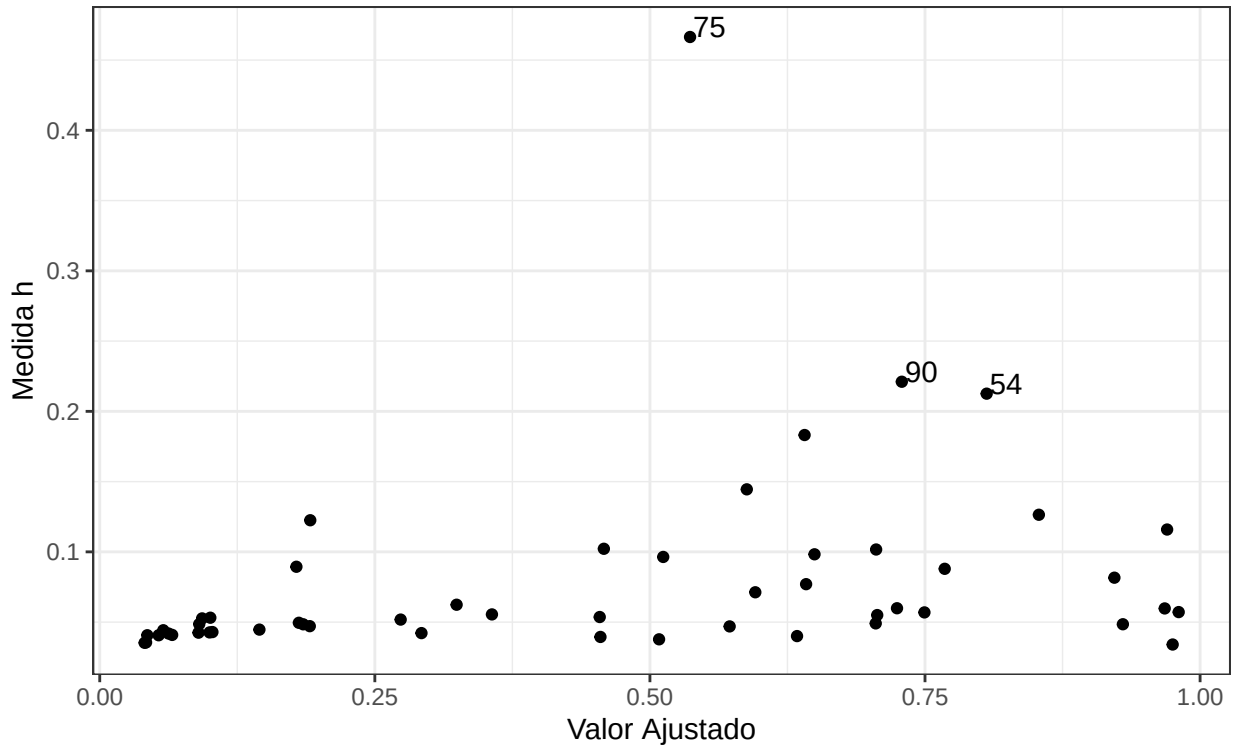
Table 8: Intervalo de confiança para a razão de chances

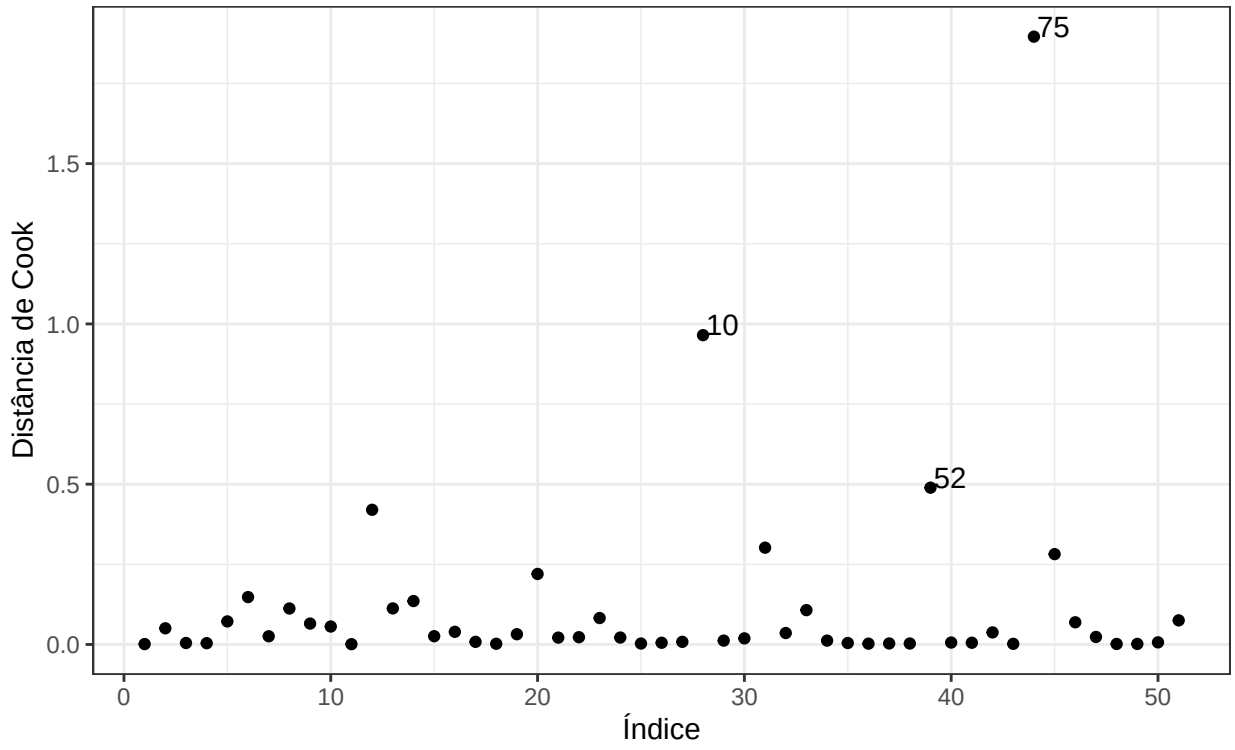
Variável	Estimativa	2.5 %	97.5 %
(Intercept)	0.005	0.000	0.087
sofa	2.199	1.406	3.961
su	1.031	0.997	1.077
scr_basal	26.585	1.215	1314.570

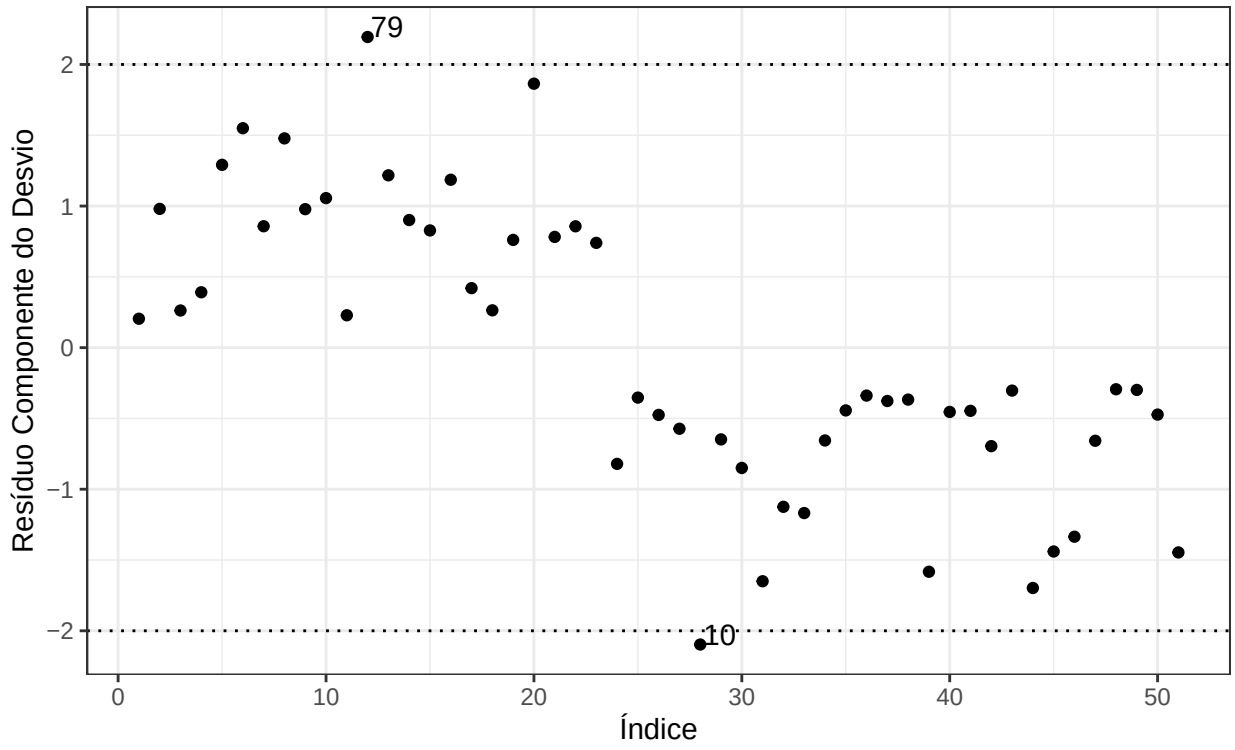
Table 9: VIF

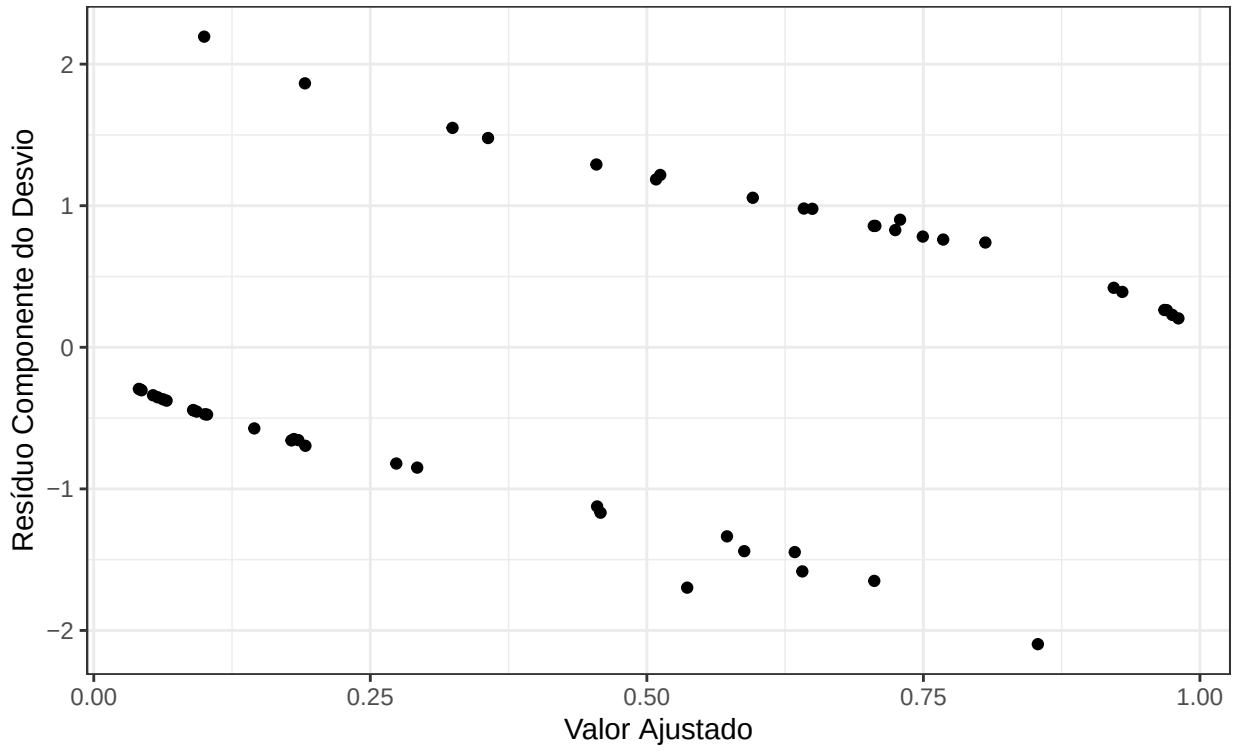
Variável	Valor
sofa	1.101
su	1.007
scr_basal	1.107

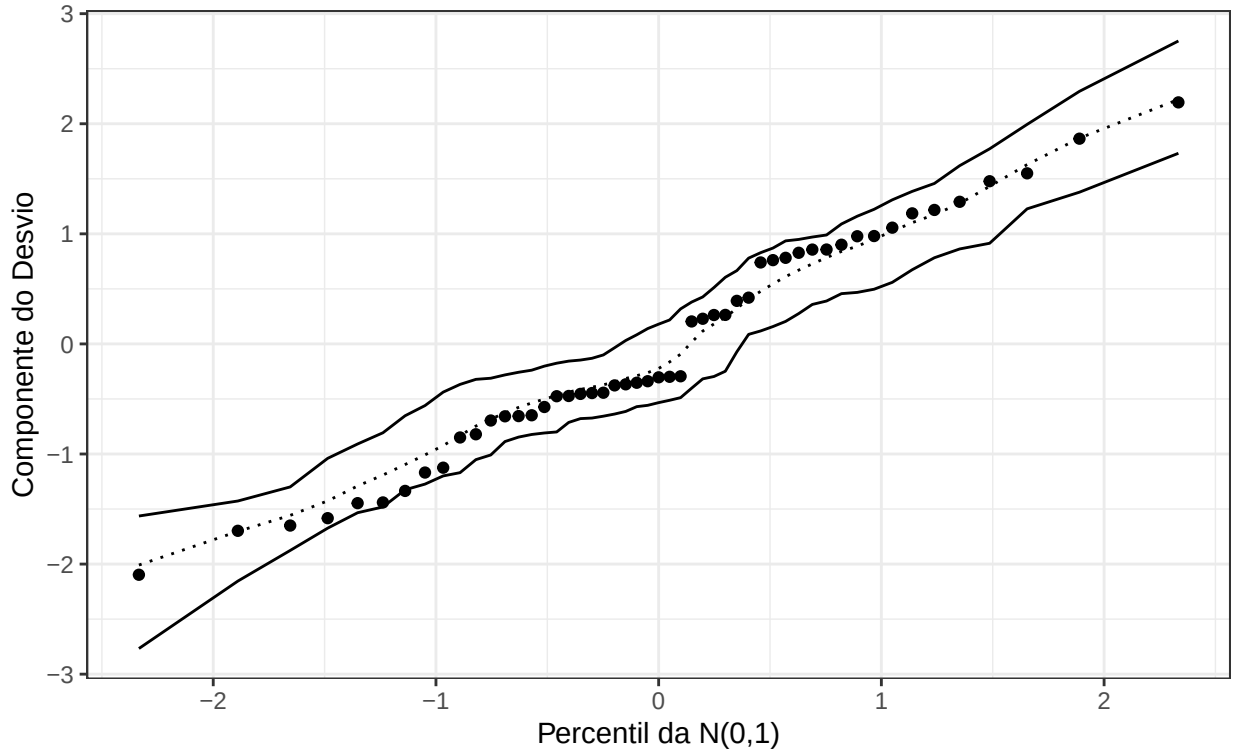
```r
if(!to_camila) model_diagnostics(df, model_fit$model, run_envelope = TRUE)
```

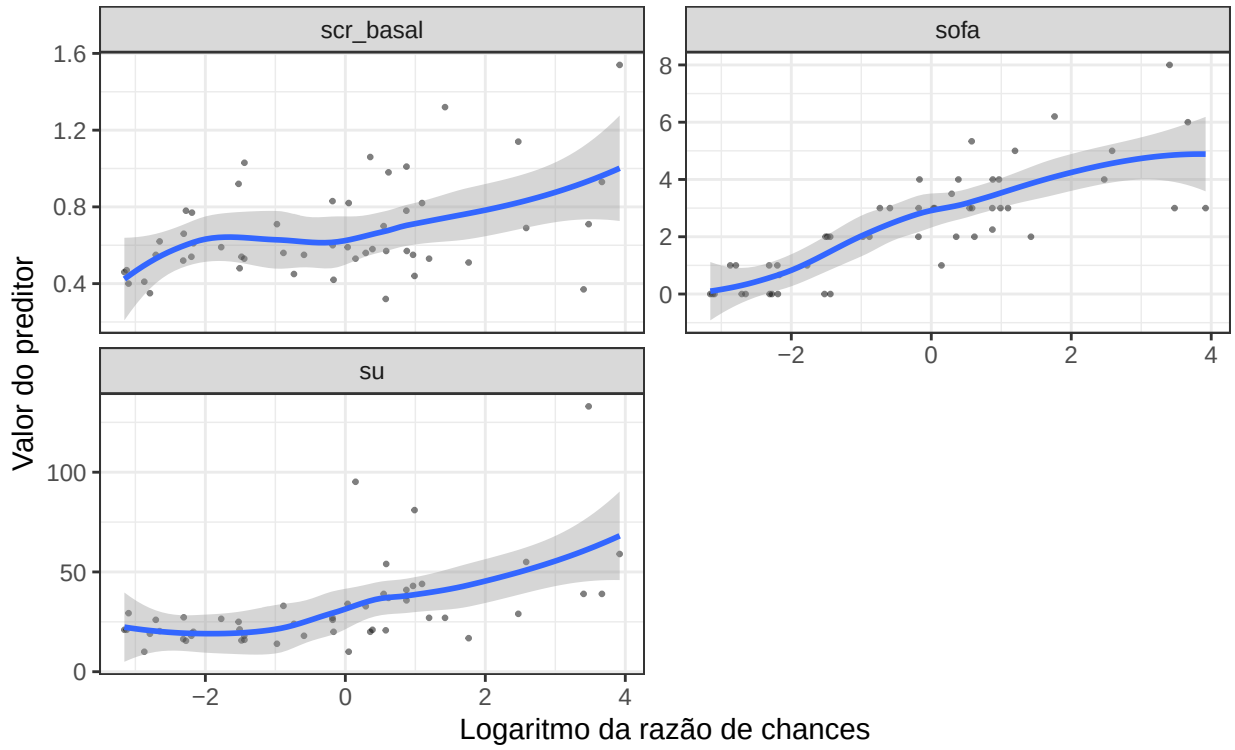

75
90
54
0.4
0.3
0.2
0.1
Medida h
0.00
0.25
0.50
0.75
1.00
Valor Ajustado


75
10
52
1.5
1.0
0.5
0.0
Distância de Cook
0
10
20
30
40
50
Índice


79
10
Resíduo Componente do Desvio
Índice
2
1
0
−1
−2
0
10
20
30
40
50


Resíduo Componente do Desvio
Valor Ajustado
2
1
0
−1
−2
0.00
0.25
0.50
0.75
1.00

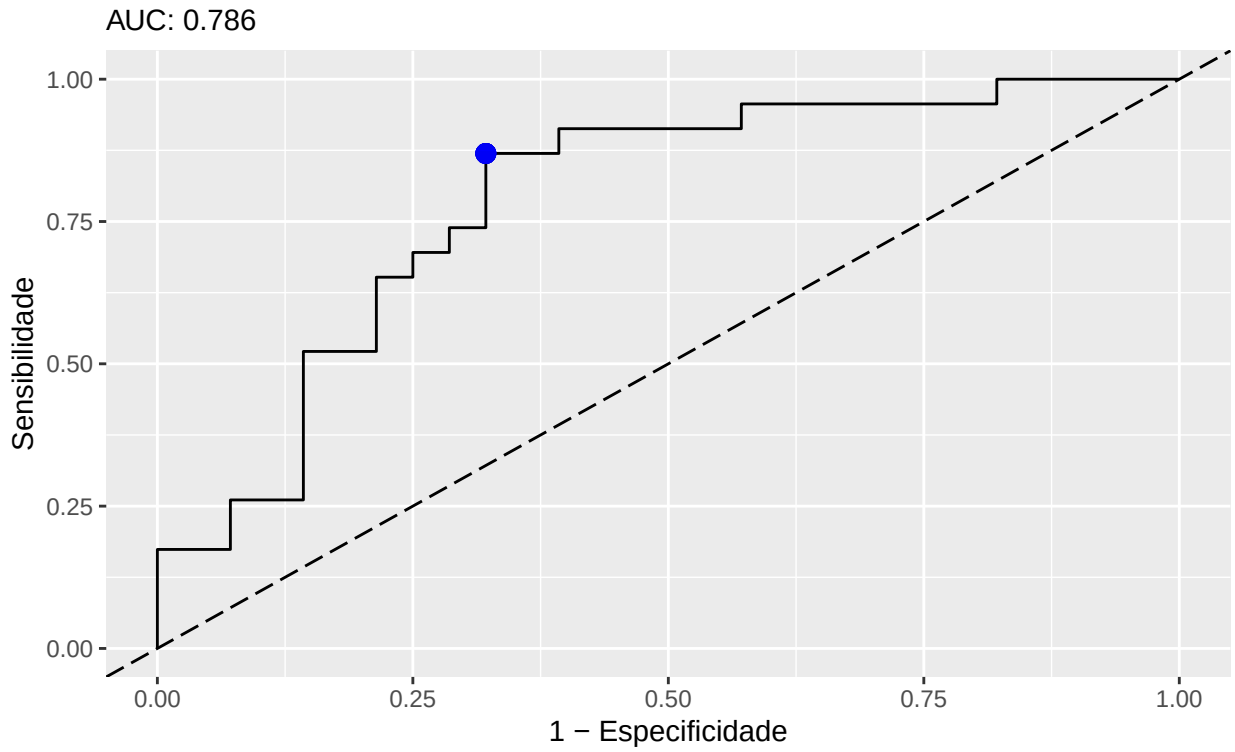





```
g_cv = loocv(df, model_fit$formula)
```

Table 10: Youden LOOCV threshold = 0.312

Ponto de corte	Sensibilidade	Especificidade	Verdadeiro positivo	Verdadeiro negativo	Falso negativo	Falso positivo
-Inf	1.000	0.000	23	0	0	28
0.043	1.000	0.036	23	1	0	27
0.044	1.000	0.071	23	2	0	26
0.050	1.000	0.107	23	3	0	25
0.058	1.000	0.143	23	4	0	24
0.061	1.000	0.179	23	5	0	23
0.063	0.957	0.179	22	5	1	23
0.067	0.957	0.214	22	6	1	22
0.081	0.957	0.250	22	7	1	21
0.094	0.957	0.286	22	8	1	20
0.097	0.957	0.321	22	9	1	19
0.102	0.957	0.357	22	10	1	18
0.107	0.957	0.393	22	11	1	17
0.129	0.957	0.429	22	12	1	16
0.151	0.913	0.429	21	12	2	16
0.171	0.913	0.464	21	13	2	15
0.192	0.913	0.500	21	14	2	14
0.195	0.913	0.536	21	15	2	13
0.208	0.913	0.571	21	16	2	12
0.250	0.913	0.607	21	17	2	11
0.284	0.870	0.607	20	17	3	11
0.297	0.870	0.643	20	18	3	10
0.312	**0.870**	**0.679**	**20**	**19**	**3**	**9**
0.371	0.826	0.679	19	19	4	9
0.442	0.783	0.679	18	19	5	9
0.467	0.739	0.679	17	19	6	9
0.481	0.739	0.714	17	20	6	8
0.500	0.696	0.714	16	20	7	8
0.538	0.696	0.750	16	21	7	7
0.583	0.652	0.750	15	21	8	7
0.606	0.652	0.786	15	22	8	6
0.612	0.609	0.786	14	22	9	6
0.631	0.565	0.786	13	22	10	6
0.655	0.522	0.786	12	22	11	6
0.675	0.522	0.821	12	23	11	5
0.689	0.522	0.857	12	24	11	4
0.690	0.478	0.857	11	24	12	4
0.699	0.435	0.857	10	24	13	4
0.721	0.391	0.857	9	24	14	4
0.740	0.348	0.857	8	24	15	4
0.748	0.304	0.857	7	24	16	4
0.763	0.261	0.857	6	24	17	4
0.781	0.261	0.893	6	25	17	3
0.850	0.261	0.929	6	26	17	2
0.921	0.217	0.929	5	26	18	2
0.935	0.174	0.929	4	26	19	2
0.946	0.174	0.964	4	27	19	1
0.957	0.174	1.000	4	28	19	0
0.966	0.130	1.000	3	28	20	0
0.970	0.087	1.000	2	28	21	0
0.977	0.043	1.000	1	28	22	0
Inf	0.000	1.000	0	28	23	0




```
df_summary = model_fit$model %>%
  glance %>%
  mutate(`Variáveis` =  paste(lapply(variaveis, get_name), collapse = ", "),
         AUC = g$auc %>% as.numeric,
         `CV AUC` = g_cv$auc %>% as.numeric) %>%
  bind_rows(df_summary)
```

Resumo

```
df_summary %>%
  select(`Variáveis`, everything(),
         -null.deviance, -df.null,
         -nobs, -df.residual, -logLik) %>%
  arrange(AIC) %>%
  niceFormatting(caption = "Comparação de modelos")
```

Table 11: Comparação de modelos

Variáveis	AIC	BIC	deviance	AUC	CV AUC
Índice SOFA, Uréia sérica, Creatinina sérica basal	54.837	62.564	46.837	0.865	0.786
Índice SOFA, Sódio urinário, Creatinina sérica basal	55.499	63.227	47.499	0.857	0.793