PS5

Pengjia "Hari" Cui

1. factors and ordering.

```r
income<-ordered(c("Mid","High","Low"))
income
```

```
[1] Mid  High Low
Levels: High < Low < Mid
```

```r
as.numeric(income)
```

```
[1] 3 1 2
```

The results of the code highlight how R handles ordered factors and their numeric representation. When the ordered() function is used without explicitly defining the order of levels, R assigns them based on alphabetical order. In this case, "High" is considered the lowest level, followed by "Low," and "Mid" as the highest. This can be seen in the output where the levels are displayed as High < Low < Mid—which doesn't align with a natural income hierarchy.

When the factor is converted to numeric values using as.numeric(income), R assigns numbers based on this ordering. Since "High" is the lowest in the assigned levels, it gets 1, "Low" gets 2, and "Mid" gets 3. This means the numeric representation does not reflect a typical income ranking where "Low" would be 1, "Mid" would be 2, and "High" would be 3.

2. Ordered categorical responses.

a

```r
library(MASS)
library(caret)
```

```
Warning: package 'caret' was built under R version 4.4.2

Loading required package: ggplot2

Loading required package: lattice
```

```r
library(stargazer)
```

```
Please cite as:

 Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.

 R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

```r
drury <- read.csv("drury_jpr_data.csv")

drury$result <- ordered(drury$result, levels = c(1, 2, 3))

drury$log_gnprat <- log(drury$gnprat)


model1 <- polr(result ~ log_gnprat + trade + tarcst + cost + coop,
               data = drury, Hess = TRUE)


model2 <- polr(result ~ log_gnprat + trade + tarcst,
               data = drury, Hess = TRUE)


stargazer(model1, model2, type = "text",
          title = "Ordered Logit Models of Sanction Success",
          dep.var.labels = "Result (Ordered Outcome)",
          covariate.labels = c("log(GNP Ratio)",
                               "Trade",
```

```r
                              "Target GNP Cost",
                              "Sender Cost",
                              "Cooperative Relationship"),
          omit.stat = c("LL", "ser", "f"))
```

```
Ordered Logit Models of Sanction Success
=====================================================
                          Dependent variable:
                     ----------------------------
                     Result (Ordered Outcome)
                         (1)           (2)
-----------------------------------------------------
log(GNP Ratio)          0.027         0.180*
                       (0.130)       (0.105)

Trade                   0.010         0.004
                       (0.007)       (0.006)

Target GNP Cost         0.002         0.001*
                       (0.002)       (0.001)

Sender Cost            -0.633*
                       (0.362)

Cooperative Relationship -0.566**
                       (0.251)

-----------------------------------------------------
Observations             79            79
=====================================================
Note:                 *p<0.1; **p<0.05; ***p<0.01
```

```r
cat("Model 1 AIC:", AIC(model1), "\n")
```

```
Model 1 AIC: 167.3797
```

```r
cat("Model 2 AIC:", AIC(model2), "\n")
```

```
Model 2 AIC: 173.7126
```

```r
cat("Model 1 BIC:", BIC(model1), "\n")
```

```
Model 1 BIC: 183.9658
```

```r
cat("Model 2 BIC:", BIC(model2), "\n")
```

```
Model 2 BIC: 185.5598
```

```r
pred1_in <- predict(model1, newdata = drury, type = "class")
pred2_in <- predict(model2, newdata = drury, type = "class")

cat("\nIn-sample performance:\n")
```

```
In-sample performance:
```

```r
conf_m1_in <- confusionMatrix(pred1_in, drury$result)
conf_m2_in <- confusionMatrix(pred2_in, drury$result)
conf_m1_in
```

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3
         1 13  8  1
         2 13 19 15
         3  1  5  4

Overall Statistics

               Accuracy : 0.4557
                 95% CI : (0.3431, 0.5717)
    No Information Rate : 0.4051
    P-Value [Acc > NIR] : 0.2106

                  Kappa : 0.1385

 Mcnemar's Test P-Value : 0.1027
```

```
Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity            0.4815   0.5938  0.20000
Specificity            0.8269   0.4043  0.89831
Pos Pred Value         0.5909   0.4043  0.40000
Neg Pred Value         0.7544   0.5938  0.76812
Prevalence             0.3418   0.4051  0.25316
Detection Rate         0.1646   0.2405  0.05063
Detection Prevalence   0.2785   0.5949  0.12658
Balanced Accuracy      0.6542   0.4990  0.54915
```

```
conf_m2_in
```

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3
         1 10  7  2
         2 17 24 16
         3  0  1  2

Overall Statistics

               Accuracy : 0.4557
                 95% CI : (0.3431, 0.5717)
    No Information Rate : 0.4051
    P-Value [Acc > NIR] : 0.2105584

                  Kappa : 0.1163

 Mcnemar's Test P-Value : 0.0002258

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity            0.3704   0.7500  0.10000
Specificity            0.8269   0.2979  0.98305
Pos Pred Value         0.5263   0.4211  0.66667
Neg Pred Value         0.7167   0.6364  0.76316
Prevalence             0.3418   0.4051  0.25316
Detection Rate         0.1266   0.3038  0.02532
```

```
Detection Prevalence   0.2405   0.7215  0.03797
Balanced Accuracy      0.5986   0.5239  0.54153
```

```r
set.seed(123)
train_idx <- createDataPartition(drury$result, p = 0.7, list = FALSE)
train_data <- drury[train_idx, ]
test_data  <- drury[-train_idx, ]

m1_train <- polr(result ~ log_gnprat + trade + tarcst + cost + coop,
                 data = train_data, Hess = TRUE)
m2_train <- polr(result ~ log_gnprat + trade + tarcst,
                 data = train_data, Hess = TRUE)

pred1_out <- predict(m1_train, newdata = test_data, type = "class")
pred2_out <- predict(m2_train, newdata = test_data, type = "class")


cat("\nOut-of-sample performance:\n")
```

```
Out-of-sample performance:
```

```r
conf_m1_out <- confusionMatrix(pred1_out, test_data$result)
conf_m2_out <- confusionMatrix(pred2_out, test_data$result)
conf_m1_out
```

```
Confusion Matrix and Statistics

          Reference
Prediction 1 2 3
         1 5 2 1
         2 3 6 5
         3 0 1 0

Overall Statistics

               Accuracy : 0.4783
                 95% CI : (0.2682, 0.6941)
    No Information Rate : 0.3913
    P-Value [Acc > NIR] : 0.2582
```

```
                  Kappa : 0.1712

 Mcnemar's Test P-Value : 0.2762

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity            0.6250   0.6667  0.00000
Specificity            0.8000   0.4286  0.94118
Pos Pred Value         0.6250   0.4286  0.00000
Neg Pred Value         0.8000   0.6667  0.72727
Prevalence             0.3478   0.3913  0.26087
Detection Rate         0.2174   0.2609  0.00000
Detection Prevalence   0.3478   0.6087  0.04348
Balanced Accuracy      0.7125   0.5476  0.47059
```

```
conf_m2_out
```

```
Confusion Matrix and Statistics

          Reference
Prediction 1 2 3
         1 1 0 0
         2 7 9 6
         3 0 0 0

Overall Statistics

               Accuracy : 0.4348
                 95% CI : (0.2319, 0.6551)
    No Information Rate : 0.3913
    P-Value [Acc > NIR] : 0.4099

                  Kappa : 0.0743

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity           0.12500  1.00000   0.0000
Specificity           1.00000  0.07143   1.0000
```

```
Pos Pred Value         1.00000  0.40909     NaN
Neg Pred Value         0.68182  1.00000  0.7391
Prevalence             0.34783  0.39130  0.2609
Detection Rate         0.04348  0.39130  0.0000
Detection Prevalence   0.04348  0.95652  0.0000
Balanced Accuracy      0.56250  0.53571  0.5000
```

b

Quantity of interest:

How does the probability of each result category change as log_gnprat moves from its minimum to maximum, holding other covariates at typical values (e.g., their medians)?

```r
typical_trade  <- median(drury$trade,   na.rm = TRUE)
typical_tarcst <- median(drury$tarcst,  na.rm = TRUE)
typical_cost   <- median(drury$cost,    na.rm = TRUE)
typical_coop   <- as.integer(round(mean(drury$coop, na.rm = TRUE)))

lgnp_seq <- seq(
  from = min(drury$log_gnprat, na.rm = TRUE),
  to   = max(drury$log_gnprat, na.rm = TRUE),
  length.out = 50
)

scenario_df <- data.frame(
  log_gnprat = lgnp_seq,
  trade      = typical_trade,
  tarcst     = typical_tarcst,
  cost       = typical_cost,
  coop       = typical_coop
)
```

```r
coefs <- coef(model1)
zeta  <- model1$zeta
V     <- vcov(model1)
est   <- c(coefs, zeta)
```

```r
set.seed(3407)
R <- 1000
sim_draws <- mvrnorm(n = R, mu = est, Sigma = V)
```

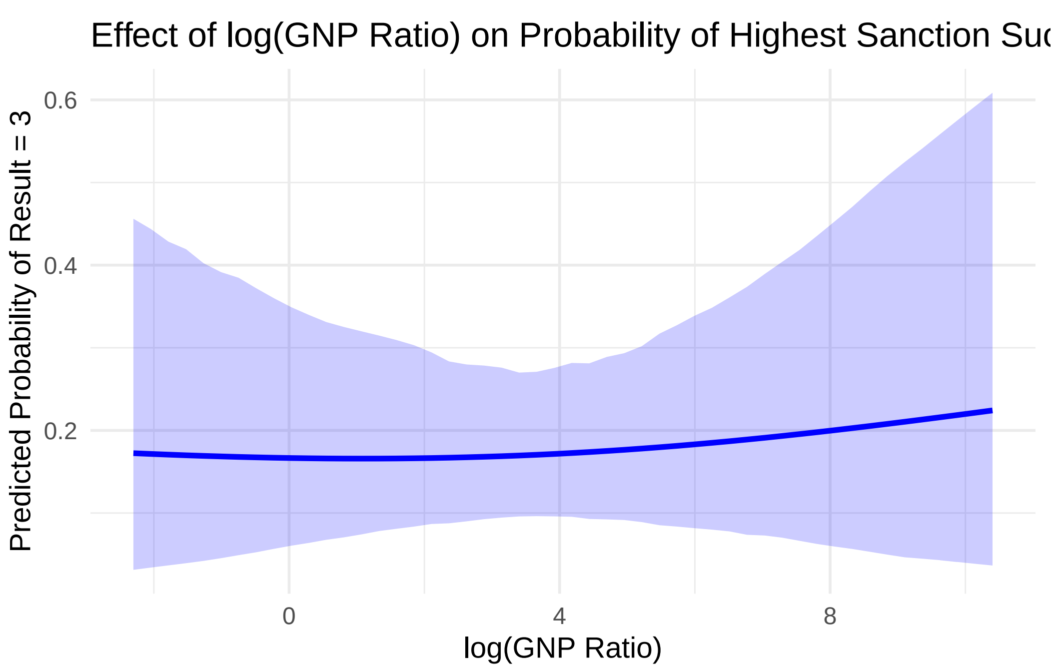
```
inv_logit <- function(z) 1 / (1 + exp(-z))
predict_cat_probs <- function(beta_vec, zeta_vec, x_row) {
  xb <- sum(beta_vec * x_row)
  p1 <- inv_logit(zeta_vec[1] - xb)
  p2 <- inv_logit(zeta_vec[2] - xb) - p1
  p3 <- 1 - inv_logit(zeta_vec[2] - xb)
  c(p1 = p1, p2 = p2, p3 = p3)
}
```

```
out_res <- data.frame(
  log_gnprat = lgnp_seq,
  p1_mean = NA, p1_lo = NA, p1_hi = NA,
  p2_mean = NA, p2_lo = NA, p2_hi = NA,
  p3_mean = NA, p3_lo = NA, p3_hi = NA
)

xvars <- names(coefs)

for (i in seq_len(nrow(scenario_df))) {
  x_row <- as.numeric(scenario_df[i, xvars])

  p_store <- matrix(NA, nrow = R, ncol = 3)

  for (r in seq_len(R)) {
    betas_r <- sim_draws[r, 1:5]
    zetas_r <- sim_draws[r, 6:7]

    p_store[r, ] <- predict_cat_probs(betas_r, zetas_r, x_row)
  }

  # Summaries for each category
  out_res$p1_mean[i] <- mean(p_store[,1])
  out_res$p1_lo[i]   <- quantile(p_store[,1], 0.025)
  out_res$p1_hi[i]   <- quantile(p_store[,1], 0.975)

  out_res$p2_mean[i] <- mean(p_store[,2])
  out_res$p2_lo[i]   <- quantile(p_store[,2], 0.025)
  out_res$p2_hi[i]   <- quantile(p_store[,2], 0.975)

  out_res$p3_mean[i] <- mean(p_store[,3])
  out_res$p3_lo[i]   <- quantile(p_store[,3], 0.025)
  out_res$p3_hi[i]   <- quantile(p_store[,3], 0.975)
```

```
}
```

```
library(ggplot2)

ggplot(out_res, aes(x = log_gnprat, y = p3_mean)) +
  geom_line(color = "blue", size = 1) +
  geom_ribbon(aes(ymin = p3_lo, ymax = p3_hi), alpha = 0.2, fill = "blue") +
  labs(
    x = "log(GNP Ratio)",
    y = "Predicted Probability of Result = 3",
    title = "Effect of log(GNP Ratio) on Probability of Highest Sanction Success"
  ) +
  theme_minimal()
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



c

```r
drury$result_numeric <- as.numeric(drury$result)

ols_model <- lm(result_numeric ~ log_gnprat + trade + tarcst + cost + coop,
                data = drury)

summary(ols_model)
```

```

Call:
lm(formula = result_numeric ~ log_gnprat + trade + tarcst + cost +
    coop, data = drury)

Residuals:
     Min       1Q   Median       3Q      Max
-1.57706 -0.58439 -0.02409  0.58596  1.39265

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.4948471  0.3728098   6.692 3.86e-09 ***
log_gnprat   0.0026725  0.0467623   0.057   0.9546
trade        0.0036223  0.0023254   1.558   0.1236
tarcst       0.0002465  0.0001437   1.715   0.0905 .
cost        -0.2321608  0.1294035  -1.794   0.0769 .
coop        -0.1734534  0.0853216  -2.033   0.0457 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7198 on 73 degrees of freedom
  (37 observations deleted due to missingness)
Multiple R-squared:  0.1845,    Adjusted R-squared:  0.1286
F-statistic: 3.302 on 5 and 73 DF,  p-value: 0.009606
```

I'd say it's better.

d

```r
library(nnet)
```

```
Warning: package 'nnet' was built under R version 4.4.2
```

```r
drury$result_unordered <- factor(drury$result, ordered = FALSE)
model_mn <- multinom(
  result_unordered ~ log_gnprat + trade + tarcst + cost + coop,
  data = drury
)
```

```
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 77.864803
iter  20 value 74.982955
final  value 74.974858
converged
```

```r
summary(model_mn)
```

```
Call:
multinom(formula = result_unordered ~ log_gnprat + trade + tarcst +
    cost + coop, data = drury)

Coefficients:
  (Intercept)   log_gnprat       trade      tarcst        cost       coop
2   0.2653256  0.229892449 0.002650732 0.001547503 -0.07159357 -0.4968503
3   2.2737814 -0.002035756 0.014949120 0.003104967 -1.08830270 -0.7948363

Std. Errors:
  (Intercept) log_gnprat       trade      tarcst      cost     coop
2    1.309996  0.1646072 0.008393752 0.002018321 0.4487488 0.302353
3    1.494338  0.1859333 0.009706515 0.002440650 0.5772072 0.405684

Residual Deviance: 149.9497
AIC: 173.9497
```

```r
library(brant)
```

```
Warning: package 'brant' was built under R version 4.4.2
```

```r
model_ordinal <- polr(
  result ~ log_gnprat + trade + tarcst + cost + coop,
  data = drury,
```

```
  Hess = TRUE
)

brant_test <- brant(model_ordinal)
```

```
--------------------------------------------
Test for    X2  df  probability
--------------------------------------------
Omnibus     3.33    5   0.65
log_gnprat  2.85    1   0.09
trade       0.53    1   0.47
tarcst      0   1   0.96
cost        1.42    1   0.23
coop        0.04    1   0.84
--------------------------------------------

H0: Parallel Regression Assumption holds
```

```
brant_test
```

```
                    X2 df probability
Omnibus    3.325547874  5  0.64993264
log_gnprat 2.853318809  1  0.09118552
trade      0.532049973  1  0.46574628
tarcst     0.003048354  1  0.95596965
cost       1.419103011  1  0.23355110
coop       0.041041594  1  0.83945766
```

The ordered logit specification appears valid under this assumption. There is no strong evidence that one would need a more flexible approach.

e

```
drury$result_mnl <- factor(drury$result, ordered = FALSE)

mnl_model <- multinom(
  result_mnl ~ log_gnprat + trade + tarcst + cost + coop,
  data = drury
)
```

```
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 77.864803
iter  20 value 74.982955
final  value 74.974858
converged
```

```r
summary(mnl_model)
```

```
Call:
multinom(formula = result_mnl ~ log_gnprat + trade + tarcst +
    cost + coop, data = drury)

Coefficients:
  (Intercept)   log_gnprat       trade      tarcst        cost       coop
2   0.2653256  0.229892449 0.002650732 0.001547503 -0.07159357 -0.4968503
3   2.2737814 -0.002035756 0.014949120 0.003104967 -1.08830270 -0.7948363

Std. Errors:
  (Intercept) log_gnprat       trade      tarcst      cost      coop
2    1.309996  0.1646072 0.008393752 0.002018321 0.4487488 0.302353
3    1.494338  0.1859333 0.009706515 0.002440650 0.5772072 0.405684

Residual Deviance: 149.9497
AIC: 173.9497
```

```r
typical_trade  <- median(drury$trade,   na.rm = TRUE)
typical_tarcst <- median(drury$tarcst,  na.rm = TRUE)
typical_cost   <- median(drury$cost,    na.rm = TRUE)
typical_coop   <- round(mean(drury$coop, na.rm = TRUE))

log_gnprat_seq <- seq(
  from = min(drury$log_gnprat, na.rm = TRUE),
  to   = max(drury$log_gnprat, na.rm = TRUE),
  length.out = 50
)

scenario_df <- data.frame(
  log_gnprat = log_gnprat_seq,
  trade      = typical_trade,
  tarcst     = typical_tarcst,
```

```r
  cost        = typical_cost,
  coop        = typical_coop
)
```

```r
set.seed(3407)
R <- 200
N <- nrow(drury)

boot_preds <- array(NA, dim = c(R, nrow(scenario_df), 3))

for (r in seq_len(R)) {
  idx <- sample.int(N, size = N, replace = TRUE)
  mnl_boot <- multinom(
    result_mnl ~ log_gnprat + trade + tarcst + cost + coop,
    data = drury[idx, ]
  )
  p_mat <- predict(mnl_boot, newdata = scenario_df, type = "probs")
  boot_preds[r, , ] <- p_mat
}
```

```
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 72.246248
iter  20 value 70.110174
iter  20 value 70.110173
iter  20 value 70.110173
final  value 70.110173
converged
# weights:  21 (12 variable)
initial  value 84.593146
iter  10 value 66.331810
iter  20 value 63.892287
final  value 63.883600
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 81.008774
iter  20 value 77.689049
final  value 77.684457
converged
# weights:  21 (12 variable)
initial  value 90.086208
```

```
iter  10 value 76.627749
iter  20 value 74.455582
final  value 74.420368
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 76.739549
iter  20 value 73.625092
final  value 73.624529
converged
# weights:  21 (12 variable)
initial  value 93.382045
iter  10 value 78.910533
iter  20 value 76.684247
iter  20 value 76.684247
iter  20 value 76.684247
final  value 76.684247
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 78.994585
iter  20 value 76.062935
final  value 76.062495
converged
# weights:  21 (12 variable)
initial  value 83.494534
iter  10 value 67.322159
iter  20 value 63.829223
final  value 63.823954
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 71.897118
iter  20 value 66.469284
iter  30 value 65.971523
iter  40 value 65.971199
iter  40 value 65.971199
iter  40 value 65.971199
final  value 65.971199
converged
# weights:  21 (12 variable)
initial  value 93.382045
iter  10 value 84.008318
```

```
iter  20 value 80.160724
final  value 80.160487
converged
# weights:  21 (12 variable)
initial  value 80.198697
iter  10 value 67.921000
final  value 67.094569
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 72.600554
iter  20 value 66.614589
final  value 66.614585
converged
# weights:  21 (12 variable)
initial  value 81.297309
iter  10 value 65.463541
iter  20 value 60.674484
final  value 60.558378
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 73.169154
iter  20 value 72.203017
final  value 72.203009
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 77.026439
final  value 75.343217
converged
# weights:  21 (12 variable)
initial  value 82.395922
iter  10 value 65.776593
final  value 63.406531
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 79.073418
iter  20 value 77.098440
final  value 77.095316
converged
# weights:  21 (12 variable)
```

```
initial  value 92.283432
iter  10 value 72.901583
iter  20 value 69.483464
final  value 69.483357
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 76.460360
iter  20 value 75.165253
final  value 75.165229
converged
# weights:  21 (12 variable)
initial  value 80.198697
iter  10 value 66.873402
iter  20 value 64.538642
iter  20 value 64.538642
iter  20 value 64.538642
final  value 64.538642
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 78.877627
final  value 77.297774
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 81.240765
iter  20 value 78.844798
final  value 78.841846
converged
# weights:  21 (12 variable)
initial  value 76.902860
iter  10 value 65.406279
iter  20 value 57.582328
final  value 57.582296
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 74.683043
iter  20 value 72.741167
final  value 72.740996
converged
# weights:  21 (12 variable)
```

```
initial  value 83.494534
iter  10 value 67.976914
iter  20 value 64.600829
final  value 64.540109
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 79.984846
iter  20 value 77.441589
final  value 77.430152
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 69.985211
final  value 67.234029
converged
# weights:  21 (12 variable)
initial  value 94.480657
iter  10 value 83.151874
iter  20 value 77.355997
final  value 77.349903
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 75.270330
iter  20 value 71.729374
final  value 71.726784
converged
# weights:  21 (12 variable)
initial  value 93.382045
iter  10 value 81.567642
iter  20 value 73.058494
final  value 73.045577
converged
# weights:  21 (12 variable)
initial  value 84.593146
iter  10 value 72.357477
iter  20 value 69.412579
final  value 69.331003
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 75.502424
```

```
iter  20 value 72.735563
iter  20 value 72.735563
iter  20 value 72.735563
final  value 72.735563
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 75.934016
iter  20 value 71.892377
iter  20 value 71.892377
iter  20 value 71.892377
final  value 71.892377
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 76.272000
iter  20 value 72.011372
final  value 72.010643
converged
# weights:  21 (12 variable)
initial  value 93.382045
iter  10 value 83.393039
final  value 79.253589
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 66.530525
iter  20 value 61.244557
iter  30 value 61.227537
iter  30 value 61.227537
iter  30 value 61.227537
final  value 61.227537
converged
# weights:  21 (12 variable)
initial  value 78.001472
iter  10 value 65.393512
iter  20 value 61.878837
final  value 61.878354
converged
# weights:  21 (12 variable)
initial  value 93.382045
iter  10 value 75.367435
iter  20 value 72.247683
```

```
iter  20 value 72.247682
iter  20 value 72.247682
final  value 72.247682
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 75.174365
iter  20 value 72.832693
final  value 72.832674
converged
# weights:  21 (12 variable)
initial  value 81.297309
iter  10 value 64.159826
iter  20 value 61.491628
final  value 61.488309
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 72.994286
iter  20 value 69.244809
final  value 69.225449
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 77.183136
final  value 75.535998
converged
# weights:  21 (12 variable)
initial  value 76.902860
iter  10 value 59.867318
iter  20 value 52.915578
iter  30 value 52.859321
final  value 52.859307
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 67.169873
iter  20 value 62.375326
final  value 62.351206
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 75.369409
```

```
final  value 73.955411
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 82.602924
iter  20 value 79.229249
final  value 79.229058
converged
# weights:  21 (12 variable)
initial  value 96.677881
iter  10 value 79.167833
iter  20 value 74.883328
iter  20 value 74.883328
iter  20 value 74.883328
final  value 74.883328
converged
# weights:  21 (12 variable)
initial  value 83.494534
iter  10 value 66.348988
iter  20 value 61.090580
final  value 61.090575
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 77.208312
iter  20 value 71.702122
iter  20 value 71.702122
iter  20 value 71.702122
final  value 71.702122
converged
# weights:  21 (12 variable)
initial  value 82.395922
iter  10 value 71.711572
iter  20 value 67.267047
final  value 67.247133
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 78.784860
iter  20 value 75.638826
final  value 75.638602
converged
# weights:  21 (12 variable)
```

```
initial  value 93.382045
iter  10 value 78.780596
final  value 76.271083
converged
# weights:  21 (12 variable)
initial  value 79.100085
iter  10 value 66.130110
iter  20 value 60.512860
iter  30 value 60.482983
iter  40 value 60.482435
iter  40 value 60.482435
iter  40 value 60.482435
final  value 60.482435
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 72.225800
iter  20 value 65.201285
final  value 64.961849
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 78.681257
iter  20 value 72.022774
final  value 71.911897
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 77.576702
iter  20 value 73.117646
final  value 73.078333
converged
# weights:  21 (12 variable)
initial  value 85.691759
iter  10 value 76.387060
iter  20 value 68.144232
iter  30 value 68.055693
iter  40 value 68.052350
final  value 68.052347
converged
# weights:  21 (12 variable)
initial  value 95.579269
iter  10 value 74.529320
```

```
iter  20 value 69.002687
final  value 69.001927
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 79.455098
iter  20 value 75.459658
final  value 75.453249
converged
# weights:  21 (12 variable)
initial  value 101.072331
iter  10 value 83.161207
iter  20 value 74.874788
iter  30 value 74.720730
final  value 74.720465
converged
# weights:  21 (12 variable)
initial  value 96.677881
iter  10 value 85.795803
iter  20 value 83.207630
final  value 83.207622
converged
# weights:  21 (12 variable)
initial  value 78.001472
iter  10 value 59.236164
iter  20 value 56.868005
iter  20 value 56.868004
iter  20 value 56.868004
final  value 56.868004
converged
# weights:  21 (12 variable)
initial  value 84.593146
iter  10 value 68.399292
iter  20 value 66.689686
final  value 66.689684
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 73.657123
iter  20 value 72.883248
final  value 72.883142
converged
# weights:  21 (12 variable)
```

```
initial  value 90.086208
iter  10 value 78.326752
iter  20 value 75.861647
iter  20 value 75.861647
iter  20 value 75.861647
final  value 75.861647
converged
# weights:  21 (12 variable)
initial  value 93.382045
iter  10 value 77.395032
iter  20 value 74.875817
final  value 74.875804
converged
# weights:  21 (12 variable)
initial  value 75.804248
iter  10 value 66.171809
iter  20 value 62.025263
iter  30 value 61.975133
iter  30 value 61.975133
iter  30 value 61.975133
final  value 61.975133
converged
# weights:  21 (12 variable)
initial  value 85.691759
iter  10 value 75.548766
iter  20 value 70.485105
final  value 70.480130
converged
# weights:  21 (12 variable)
initial  value 78.001472
iter  10 value 66.890683
iter  20 value 62.011239
final  value 62.011236
converged
# weights:  21 (12 variable)
initial  value 95.579269
iter  10 value 74.530523
iter  20 value 70.396556
iter  20 value 70.396556
iter  20 value 70.396556
final  value 70.396556
converged
# weights:  21 (12 variable)
```

```
initial  value 95.579269
iter  10 value 79.158502
iter  20 value 75.357571
iter  20 value 75.357571
iter  20 value 75.357571
final  value 75.357571
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 71.162477
iter  20 value 65.955188
final  value 65.954403
converged
# weights:  21 (12 variable)
initial  value 85.691759
iter  10 value 68.117829
iter  20 value 60.535481
iter  30 value 60.488473
iter  40 value 60.488402
iter  40 value 60.488402
iter  40 value 60.488402
final  value 60.488402
converged
# weights:  21 (12 variable)
initial  value 83.494534
iter  10 value 69.677640
iter  20 value 64.889213
iter  20 value 64.889212
iter  20 value 64.889212
final  value 64.889212
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 75.998373
final  value 73.516267
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 75.917722
iter  20 value 72.126510
final  value 72.126509
converged
# weights:  21 (12 variable)
```

```
initial  value 85.691759
iter  10 value 65.273767
iter  20 value 61.311268
final  value 61.300865
converged
# weights:  21 (12 variable)
initial  value 81.297309
iter  10 value 69.382251
iter  20 value 66.400077
iter  20 value 66.400076
iter  20 value 66.400076
final  value 66.400076
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 70.481347
iter  20 value 68.177553
final  value 68.174816
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 76.526103
iter  20 value 73.429654
iter  20 value 73.429653
iter  20 value 73.429653
final  value 73.429653
converged
# weights:  21 (12 variable)
initial  value 76.902860
iter  10 value 67.582364
iter  20 value 65.285884
final  value 65.283062
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 80.897392
iter  20 value 78.184606
final  value 78.184121
converged
# weights:  21 (12 variable)
initial  value 83.494534
iter  10 value 69.896427
final  value 68.159633
```

```
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 78.316176
iter  20 value 71.871794
final  value 71.847586
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 74.940644
iter  20 value 71.836915
final  value 71.836895
converged
# weights:  21 (12 variable)
initial  value 98.875106
iter  10 value 92.111139
iter  20 value 89.319602
final  value 89.297396
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 75.263436
iter  20 value 69.193140
final  value 69.193138
converged
# weights:  21 (12 variable)
initial  value 81.297309
iter  10 value 66.970015
iter  20 value 64.018729
final  value 64.018559
converged
# weights:  21 (12 variable)
initial  value 76.902860
iter  10 value 66.211605
final  value 65.482085
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 71.073224
iter  20 value 67.759614
final  value 67.634573
converged
# weights:  21 (12 variable)
```

```
initial  value 96.677881
iter  10 value 87.757894
iter  20 value 85.026137
final  value 85.026126
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 77.340359
final  value 75.827486
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 76.029175
final  value 73.722600
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 75.304808
final  value 72.578233
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 80.775700
iter  20 value 74.050919
iter  30 value 73.970624
final  value 73.970568
converged
# weights:  21 (12 variable)
initial  value 84.593146
iter  10 value 71.176382
iter  20 value 69.795191
final  value 69.795058
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 71.032452
final  value 69.272382
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 76.715554
final  value 74.668630
converged
```

```
# weights:  21 (12 variable)
initial  value 93.382045
iter  10 value 74.879666
iter  20 value 71.021963
final  value 71.011132
converged
# weights:  21 (12 variable)
initial  value 99.973718
iter  10 value 87.252822
final  value 82.658446
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 78.464691
iter  20 value 70.588301
final  value 70.586398
converged
# weights:  21 (12 variable)
initial  value 83.494534
iter  10 value 69.670770
iter  20 value 60.168941
iter  30 value 60.060211
final  value 60.060184
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 70.349047
iter  20 value 62.623875
final  value 62.623836
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 75.463458
iter  20 value 74.242911
final  value 74.242707
converged
# weights:  21 (12 variable)
initial  value 78.001472
iter  10 value 62.373855
iter  20 value 59.033693
iter  20 value 59.033693
iter  20 value 59.033693
final  value 59.033693
```

```
converged
# weights:  21 (12 variable)
initial  value 85.691759
iter  10 value 77.117309
iter  20 value 75.491739
final  value 75.491733
converged
# weights:  21 (12 variable)
initial  value 83.494534
iter  10 value 67.411377
iter  20 value 63.212988
final  value 63.159853
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 80.302260
final  value 79.045649
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 78.497270
iter  20 value 75.979477
final  value 75.979457
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 74.357867
iter  20 value 72.279655
final  value 72.250113
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 77.349211
iter  20 value 71.272796
iter  30 value 70.640487
final  value 70.639468
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 71.758878
iter  20 value 68.421746
final  value 68.421742
converged
```

```
# weights:  21 (12 variable)
initial  value 84.593146
iter  10 value 66.536986
iter  20 value 61.690174
final  value 61.676957
converged
# weights:  21 (12 variable)
initial  value 81.297309
iter  10 value 64.349017
iter  20 value 63.356870
final  value 63.356869
converged
# weights:  21 (12 variable)
initial  value 81.297309
iter  10 value 62.070108
iter  20 value 60.405790
final  value 60.405790
converged
# weights:  21 (12 variable)
initial  value 82.395922
iter  10 value 68.960710
iter  20 value 64.097378
iter  30 value 63.573771
iter  40 value 63.568026
final  value 63.568024
converged
# weights:  21 (12 variable)
initial  value 97.776494
iter  10 value 90.491269
iter  20 value 85.110816
final  value 85.100661
converged
# weights:  21 (12 variable)
initial  value 84.593146
iter  10 value 75.575815
iter  20 value 71.064690
final  value 71.064536
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 60.407736
iter  20 value 56.877619
final  value 56.877617
```

```
converged
# weights:  21 (12 variable)
initial  value 82.395922
iter  10 value 72.148541
iter  20 value 66.538543
final  value 66.498886
converged
# weights:  21 (12 variable)
initial  value 103.269555
iter  10 value 94.210394
final  value 91.686411
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 74.983375
iter  20 value 67.862921
iter  30 value 67.676778
final  value 67.676252
converged
# weights:  21 (12 variable)
initial  value 81.297309
iter  10 value 56.497878
iter  20 value 54.887566
iter  30 value 54.869160
final  value 54.869157
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 73.808654
iter  20 value 67.141291
iter  30 value 66.588559
final  value 66.588265
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 73.370299
iter  20 value 71.390646
final  value 71.386833
converged
# weights:  21 (12 variable)
initial  value 74.705636
iter  10 value 67.293027
final  value 66.362572
```

```
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 80.783980
final  value 78.716877
converged
# weights:  21 (12 variable)
initial  value 94.480657
iter  10 value 84.671719
final  value 84.524290
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 78.680773
iter  20 value 71.999479
final  value 71.946641
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 77.306578
iter  20 value 74.945723
final  value 74.945332
converged
# weights:  21 (12 variable)
initial  value 96.677881
iter  10 value 83.061585
iter  20 value 76.934382
final  value 76.932972
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 70.032660
final  value 68.043329
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 67.744165
iter  20 value 63.779552
iter  20 value 63.779552
iter  20 value 63.779552
final  value 63.779552
converged
# weights:  21 (12 variable)
```

```
initial  value 85.691759
iter  10 value 75.515780
iter  20 value 72.838028
final  value 72.831594
converged
# weights:  21 (12 variable)
initial  value 81.297309
iter  10 value 66.479942
iter  20 value 61.150260
iter  20 value 61.150259
iter  20 value 61.150259
final  value 61.150259
converged
# weights:  21 (12 variable)
initial  value 84.593146
iter  10 value 68.226643
iter  20 value 62.107381
final  value 62.100428
converged
# weights:  21 (12 variable)
initial  value 81.297309
iter  10 value 66.536222
iter  20 value 62.831047
final  value 62.828902
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 79.993490
iter  20 value 77.830470
final  value 77.830437
converged
# weights:  21 (12 variable)
initial  value 75.804248
iter  10 value 66.576377
final  value 63.900561
converged
# weights:  21 (12 variable)
initial  value 83.494534
iter  10 value 58.733558
iter  20 value 55.637527
final  value 55.637491
converged
# weights:  21 (12 variable)
```

```
initial  value 84.593146
iter  10 value 75.970205
final  value 74.882734
converged
# weights:  21 (12 variable)
initial  value 98.875106
iter  10 value 83.384669
iter  20 value 80.429398
final  value 80.177150
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 74.604108
final  value 71.718429
converged
# weights:  21 (12 variable)
initial  value 75.804248
iter  10 value 62.887494
final  value 61.649561
converged
# weights:  21 (12 variable)
initial  value 97.776494
iter  10 value 85.807032
iter  20 value 81.919087
final  value 81.918202
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 63.132969
iter  20 value 61.389330
iter  30 value 61.385655
final  value 61.385625
converged
# weights:  21 (12 variable)
initial  value 82.395922
iter  10 value 68.904145
iter  20 value 61.918045
final  value 61.918034
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 75.906975
iter  20 value 73.426198
```

```
iter  20 value 73.426198
iter  20 value 73.426198
final  value 73.426198
converged
# weights:  21 (12 variable)
initial  value 95.579269
iter  10 value 83.684627
iter  20 value 82.878382
final  value 82.877824
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 75.175143
iter  20 value 70.374976
final  value 70.374942
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 79.002668
iter  20 value 74.608508
iter  20 value 74.608508
iter  20 value 74.608508
final  value 74.608508
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 80.814649
iter  20 value 78.408646
final  value 78.407345
converged
# weights:  21 (12 variable)
initial  value 78.001472
iter  10 value 67.307915
iter  20 value 59.099206
iter  30 value 58.511843
iter  40 value 58.494767
final  value 58.494759
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 77.495102
iter  20 value 71.627490
iter  20 value 71.627490
```

```
iter  20 value 71.627490
final  value 71.627490
converged
# weights:  21 (12 variable)
initial  value 80.198697
iter  10 value 63.349456
iter  20 value 59.932048
iter  20 value 59.932048
iter  20 value 59.932048
final  value 59.932048
converged
# weights:  21 (12 variable)
initial  value 82.395922
iter  10 value 67.044381
iter  20 value 63.492991
final  value 63.490762
converged
# weights:  21 (12 variable)
initial  value 80.198697
iter  10 value 68.117789
iter  20 value 63.880572
iter  20 value 63.880571
iter  20 value 63.880571
final  value 63.880571
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 68.657170
iter  20 value 66.942388
iter  20 value 66.942388
iter  20 value 66.942388
final  value 66.942388
converged
# weights:  21 (12 variable)
initial  value 95.579269
iter  10 value 81.418571
iter  20 value 77.977705
final  value 77.977679
converged
# weights:  21 (12 variable)
initial  value 84.593146
iter  10 value 73.276612
iter  20 value 72.228039
```

```
final  value 72.226773
converged
# weights:  21 (12 variable)
initial  value 94.480657
iter  10 value 76.245626
iter  20 value 73.623885
iter  20 value 73.623884
iter  20 value 73.623884
final  value 73.623884
converged
# weights:  21 (12 variable)
initial  value 83.494534
iter  10 value 72.373659
iter  20 value 67.652172
final  value 67.651898
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 69.337179
iter  20 value 66.501380
final  value 66.496908
converged
# weights:  21 (12 variable)
initial  value 81.297309
iter  10 value 62.758234
iter  20 value 54.781895
final  value 54.772350
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 76.330657
iter  20 value 71.608182
iter  30 value 71.319196
iter  30 value 71.319196
iter  30 value 71.319196
final  value 71.319196
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 76.744660
iter  20 value 73.875046
final  value 73.868622
converged
```

```
# weights:  21 (12 variable)
initial  value 83.494534
iter  10 value 74.969865
iter  20 value 72.570259
final  value 72.520819
converged
# weights:  21 (12 variable)
initial  value 83.494534
iter  10 value 66.514895
iter  20 value 63.017437
final  value 63.003006
converged
# weights:  21 (12 variable)
initial  value 76.902860
iter  10 value 67.206620
iter  20 value 65.845936
final  value 65.845065
converged
# weights:  21 (12 variable)
initial  value 94.480657
iter  10 value 86.074959
final  value 82.225479
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 77.986908
iter  20 value 74.098124
final  value 74.025387
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 80.361372
final  value 76.159497
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 69.155452
iter  20 value 65.695154
final  value 65.689332
converged
# weights:  21 (12 variable)
initial  value 85.691759
iter  10 value 73.307654
```

```
iter  20 value 66.856637
final  value 66.856631
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 78.399632
final  value 76.940969
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 74.983863
iter  20 value 71.882681
final  value 71.882571
converged
# weights:  21 (12 variable)
initial  value 85.691759
iter  10 value 73.427326
iter  20 value 64.892350
final  value 64.892180
converged
# weights:  21 (12 variable)
initial  value 97.776494
iter  10 value 90.104372
final  value 86.440008
converged
# weights:  21 (12 variable)
initial  value 92.283432
iter  10 value 83.488616
iter  20 value 78.015724
final  value 77.989975
converged
# weights:  21 (12 variable)
initial  value 82.395922
iter  10 value 69.179581
iter  20 value 63.758684
final  value 63.751958
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 76.950718
iter  20 value 75.415387
final  value 75.414509
converged
```

```
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 66.970627
iter  20 value 65.230991
final  value 65.230981
converged
# weights:  21 (12 variable)
initial  value 99.973718
iter  10 value 87.452336
iter  20 value 82.269907
final  value 82.269746
converged
# weights:  21 (12 variable)
initial  value 84.593146
iter  10 value 74.173524
iter  20 value 72.138844
final  value 72.108646
converged
# weights:  21 (12 variable)
initial  value 85.691759
iter  10 value 74.619718
final  value 71.494089
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 75.949656
final  value 74.547638
converged
# weights:  21 (12 variable)
initial  value 88.987595
iter  10 value 72.359036
iter  20 value 68.247981
final  value 68.247965
converged
# weights:  21 (12 variable)
initial  value 81.297309
iter  10 value 64.830497
iter  20 value 61.875016
final  value 61.874978
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 78.226425
```

```
iter  20 value 71.045298
final  value 71.006163
converged
# weights:  21 (12 variable)
initial  value 97.776494
iter  10 value 71.548854
iter  20 value 63.899400
iter  30 value 63.848942
final  value 63.848940
converged
# weights:  21 (12 variable)
initial  value 84.593146
iter  10 value 74.966913
final  value 71.066387
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 69.697168
iter  20 value 67.781957
final  value 67.780618
converged
# weights:  21 (12 variable)
initial  value 87.888983
iter  10 value 82.404133
final  value 80.725125
converged
# weights:  21 (12 variable)
initial  value 90.086208
iter  10 value 77.973913
iter  20 value 72.819462
final  value 72.811837
converged
# weights:  21 (12 variable)
initial  value 86.790371
iter  10 value 76.117373
iter  20 value 73.565811
final  value 73.565810
converged
# weights:  21 (12 variable)
initial  value 91.184820
iter  10 value 76.843503
iter  20 value 71.971268
final  value 71.962459
```

```
converged
# weights:  21 (12 variable)
initial  value 80.198697
iter  10 value 56.330705
iter  20 value 54.476991
final  value 54.476979
converged
# weights:  21 (12 variable)
initial  value 80.198697
iter  10 value 73.162077
iter  20 value 70.041802
final  value 70.019893
converged
# weights:  21 (12 variable)
initial  value 82.395922
iter  10 value 70.923905
iter  20 value 66.395219
iter  30 value 66.374529
final  value 66.374474
converged
# weights:  21 (12 variable)
initial  value 82.395922
iter  10 value 63.428532
iter  20 value 60.783459
final  value 60.781762
converged
```

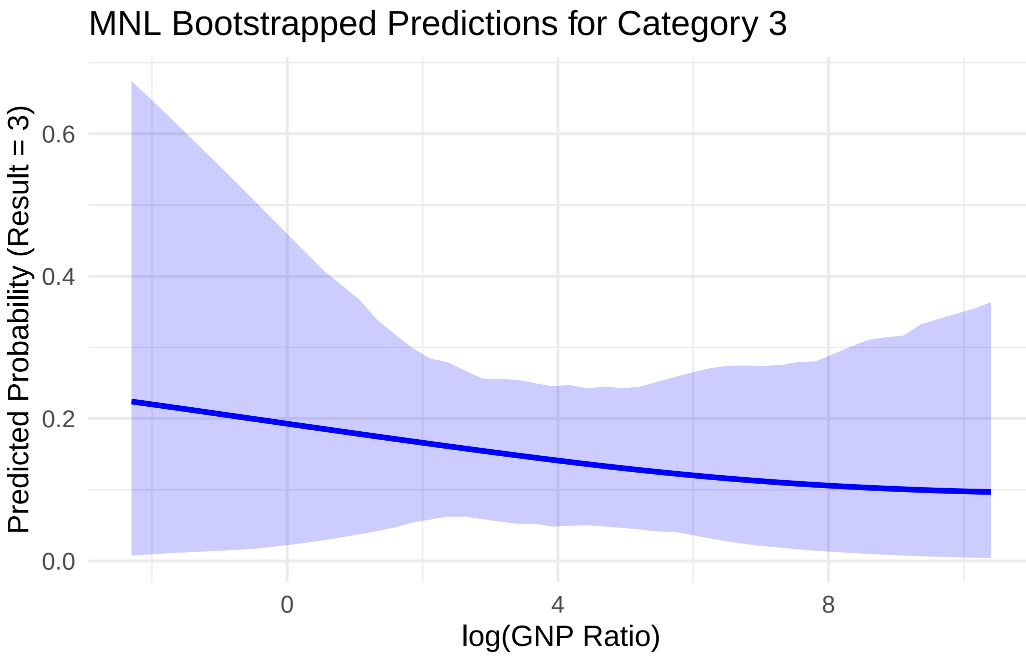
```r
prob_mean <- apply(boot_preds, c(2,3), mean)
prob_lo   <- apply(boot_preds, c(2,3), quantile, 0.025)
prob_hi   <- apply(boot_preds, c(2,3), quantile, 0.975)

library(dplyr)
```

```
Attaching package: 'dplyr'

The following object is masked from 'package:MASS':

    select

The following objects are masked from 'package:stats':
```

```
    filter, lag


The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
plot_df <- scenario_df %>%
  mutate(

    p1 = prob_mean[,1],
    p1_lo = prob_lo[,1],
    p1_hi = prob_hi[,1],

    p2 = prob_mean[,2],
    p2_lo = prob_lo[,2],
    p2_hi = prob_hi[,2],

    p3 = prob_mean[,3],
    p3_lo = prob_lo[,3],
    p3_hi = prob_hi[,3]
  )
```

```r
ggplot(plot_df, aes(x = log_gnprat, y = p3)) +
  geom_line(color = "blue", size = 1) +
  geom_ribbon(aes(ymin = p3_lo, ymax = p3_hi), alpha = 0.2, fill = "blue") +
  labs(
    x = "log(GNP Ratio)",
    y = "Predicted Probability (Result = 3)",
    title = "MNL Bootstrapped Predictions for Category 3"
  ) +
  theme_minimal()
```



The quantity of interest is how the predicted probabilities of each outcome (1, 2, or 3) shift as log_gnprat varies from its minimum to maximum, while other covariates remain at typical values. The multinomial logit assigns unique intercepts and slopes for each outcome category instead of assuming parallel regressions. This can produce more flexible probability curves across the range of log_gnprat. Bootstrap-based confidence ribbons illustrate sampling variability in these predictions, with wider ribbons indicating greater uncertainty.