Modèle Quantal Response Equilibrium

SCAIA Matteo - HACHEM REDA Riwa - GILLET Louison

2025-05-24

Introduction

Ce document présente une estimation individuelle du modèle **Quantal Response Equilibrium (QRE)** basé sur les données collectées. Nous estimons pour chaque participant le paramètre λ, mesurant la sensibilité aux gains. L'ajustement du modèle est évalué par la log-vraisemblance et le score de Brier.

Chargement des bibliothèques et données

```
library(readxl)
library(ggplot2)
library(tidyr)
library(dplyr)
library(xtable)
library(factoextra)

donnees <- read_excel('../data/Exp_rience.xlsx')
donneesQ = donnees[1:63,7:21]
```

```
questions <- c("Q00_16-20 Arad & Rubinstein", "Q00_Mode du groupe", "Q00_ChatGPT",
               "Q00_Prime de 20","Q01_12-20" ,"Q02_Alaoui & Penta", "Q03_Alaoui & Penta
               "Q04_Goeree moderate", "Q05_Goeree extreme", "Q06_Cycle A", "Q07_Cycle C"
               "Q08_Pan A", "Q09_Bear C", "Q10_long pan A", "Q11_long pan C")


actions_par_question <- list(
  "Q00_16-20 Arad & Rubinstein" = 16:20,
  "Q00_Mode du groupe" = 16:20,
  "Q00_ChatGPT" = 16:20,
  "Q00_Prime de 20" = 16:20,
  "Q01_12-20" = c(12,14,16,18,20),
  "Q02_Alaoui & Penta" = c(12,14,16,18,20),
  "Q03_Alaoui & Penta plus" = c(12,14,16,18,20),
  "Q04_Goeree moderate" = c(14,12,18,16,20),
  "Q05_Goeree extreme" = c(18,16,14,12,20),
  "Q06_Cycle A" = c(12, 14, 16, 18, 20),
  "Q07_Cycle C" = c(12, 14, 16, 18, 20),
  "Q08_Pan A" = c(12, 14, 16, 18, 20),
  "Q09_Bear C" = c(12, 14, 16, 18, 20),
  "Q10_long pan A" = c(12, 14, 16, 18, 20),
  "Q11_long pan C" = c(12, 14, 16, 18, 20)
)

bonus_par_question <- list(
```

```
  "Q00_16-20 Arad & Rubinstein" = list(gauche = 10, egal = 0),
  "Q00_Mode du groupe" = list(gauche = 10, egal = 0),
  "Q00_ChatGPT" = list(gauche = 10, egal = 0),
  "Q00_Prime de 20" = list(gauche = 20, egal = 0),
  "Q01_12-20" = list(gauche = 20, egal = 0),
  "Q02_Alaoui & Penta" = list(gauche = 20, egal = 10),
  "Q03_Alaoui & Penta plus" = list(gauche = 40, egal = 10),
  "Q04_Goeree moderate" = list(gauche = 20, egal = 0),
  "Q05_Goeree extreme" = list(gauche = 20, egal = 0),
  "Q06_Cycle A" = list(gauche = 20, egal = 0),
  "Q07_Cycle C" = list(gauche = 20, egal = 0),
  "Q08_Pan A" = list(gauche = 20, egal = 0),
  "Q09_Bear C" = list(gauche = 20, egal = 0),
  "Q10_long pan A" = list(gauche = 20, egal = 0),
  "Q11_long pan C" = list(gauche = 20, egal = 0)
)
```

Fonctions

```
logit_qre_response <- function(payoffs, lambda) {
  exp_payoff <- exp(lambda * payoffs)
  exp_payoff / sum(exp_payoff)
}

qre_strategy <- function(lambda, actions, bonus_gauche, bonus_egal) {
  expected_payoff <- sapply(seq_along(actions), function(a_idx) {
    a <- actions[a_idx]
    if (a_idx < length(actions)) {
      bonus_pos <- a_idx + 1
      bonus_left <- bonus_gauche
    } else {
      bonus_left <- 0
    }
    bonus_same <- bonus_egal
    gain <- a + bonus_left + bonus_same
    return(gain)
  })

  probs <- logit_qre_response(expected_payoff, lambda)
  return(probs)
}
```

Log-vraisemblance et estimation

```
log_likelihood_qre <- function(lambda, reponses, questions, actions_map) {
  total_loglik <- 0

  for (j in seq_along(reponses)) {
    r <- reponses[j]
    q <- questions[j]
    actions <- actions_map[[q]]
    bonus <- bonus_par_question[[q]]

    qre_probs <- qre_strategy(lambda, actions, bonus_gauche = bonus$gauche, bonus_egal =

    if (!is.na(r)) {
      idx <- which(actions == r)
      if (length(idx) == 1) {
        p_r <- qre_probs[idx]
      } else {
        p_r <- 0
      }
      if (p_r > 0 && !is.na(p_r)) {
        total_loglik <- total_loglik + log(p_r)
      } else {
        total_loglik <- total_loglik - 1e6
      }
    }
  }

  return(total_loglik)
}
```

Estimation individuelle

```
resultats_qre <- data.frame()

for (i in 1:(nrow(donnees)-1)) {
  individu <- donnees[i, ]

  reponses <- sapply(questions, function(q) {
    rep <- individu[[q]]
    if (!is.na(rep) && grepl(":", rep)) {
      val <- as.numeric(trimws(strsplit(rep, ":")[[1]][2]))
    } else {
```

```
    val <- as.numeric(trimws(rep))
  }
  return(val)
})

tryCatch({
  opt <- optim(
    par = c(lambda = 1),
    fn = function(par) -log_likelihood_qre(par[1], reponses, questions, actions_par_q
    method = "L-BFGS-B",
    lower = c(0.01), upper = c(20)
  )

  lambda_estime <- round(opt$par[1], 3)
  max_vraisemblance <- round(-opt$value, 3)

  # Score de Brier
  brier_total <- 0
  nb_valides <- 0

  for (j in seq_along(reponses)) {
    r <- reponses[j]
    q <- questions[j]
    actions <- actions_par_question[[q]]
    bonus <- bonus_par_question[[q]]

    qre_probs <- qre_strategy(lambda_estime, actions, bonus_gauche = bonus$gauche, bon

    if (!is.na(r)) {
      y_true <- rep(0, length(actions))
      idx <- which(actions == r)
      if (length(idx) == 1) {
        y_true[idx] <- 1
        brier_score <- sum((qre_probs - y_true)^2)
        brier_total <- brier_total + brier_score
        nb_valides <- nb_valides + 1
      }
    }
  }

  brier_moyen <- ifelse(nb_valides > 0, brier_total / nb_valides, NA)

  resultats_qre <- rbind(resultats_qre, data.frame(
    id = i,
```

```
      lambda = lambda_estime,
      VraisemblanceMax = max_vraisemblance,
      BrierScore = round(brier_moyen, 4)
    ))

  }, error = function(e) {
    resultats_qre <- rbind(resultats_qre, data.frame(
      id = i,
      lambda = NA,
      VraisemblanceMax = NA,
      BrierScore = NA
    ))
    cat("Erreur pour l'individu", i, ":", e$message, "\n")
  })
}
print(resultats_qre)
```

```
##          id lambda VraisemblanceMax BrierScore
## lambda    1  1.495           -4.312     0.1501
## lambda1   2  0.359          -16.690     0.6370
## lambda2   3  0.036          -23.806     0.7981
## lambda3   4  0.231          -20.010     0.7500
## lambda4   5  0.196          -20.648     0.7524
## lambda5   6  0.010          -22.613     0.8026
## lambda6   7  0.039          -23.769     0.7950
## lambda7   8  0.801          -11.084     0.4595
## lambda8   9  0.274          -19.254     0.7111
## lambda9  10  0.054          -23.492     0.7801
## lambda10 11  0.010          -24.914     0.8198
## lambda11 12  0.207          -20.447     0.7541
## lambda12 13  0.010          -24.194     0.8018
## lambda13 14  0.185          -20.838     0.6874
## lambda14 15  0.010          -24.414     0.8066
## lambda15 16  0.099          -19.577     0.7834
## lambda16 17  0.603          -13.861     0.5900
## lambda17 18  1.046           -8.347     0.3291
## lambda18 19  0.463          -14.107     0.5679
## lambda19 20  0.036          -23.806     0.7931
## lambda20 21  0.167          -21.190     0.6985
## lambda21 22  0.010          -24.924     0.8202
## lambda22 23  0.022          -24.007     0.7956
## lambda23 24  0.010          -25.064     0.8231
## lambda24 25  0.136          -21.794     0.7727
## lambda25 26  0.118          -22.175     0.7337
```

```
## lambda26 27  0.327          -18.353     0.6994
## lambda27 28  0.046          -20.486     0.7968
## lambda28 29  0.231          -20.010     0.7312
## lambda29 30  0.185          -20.838     0.6746
## lambda30 31  0.010          -24.214     0.8010
## lambda31 32  0.010          -25.974     0.8464
## lambda32 33  0.010          -25.374     0.8310
## lambda33 34  0.010          -25.974     0.8464
## lambda34 35  0.011          -24.104     0.7990
## lambda35 36  0.024          -23.984     0.7995
## lambda36 37  0.089          -22.790     0.7698
## lambda37 38  0.046          -23.642     0.7869
## lambda38 39  0.185          -20.838     0.7651
## lambda39 40  0.010          -24.424     0.8068
## lambda40 41  0.010          -24.154     0.8010
## lambda41 42  0.010          -24.264     0.8033
## lambda42 43  0.010          -24.494     0.8092
## lambda43 44  0.077          -23.038     0.7880
## lambda44 45  0.041          -23.729     0.7895
## lambda45 46  0.089          -22.790     0.7196
## lambda46 47  0.010          -24.134     0.7999
## lambda47 48  0.039          -23.769     0.7977
## lambda48 49  0.159          -21.353     0.7072
## lambda49 50  0.070          -23.184     0.7805
## lambda50 51  0.010          -24.184     0.8003
## lambda51 52  0.010          -24.874     0.8190
## lambda52 53  0.143          -21.655     0.7541
## lambda53 54  0.167          -21.190     0.7632
## lambda54 55  0.143          -21.655     0.7724
## lambda55 56  0.151          -21.508     0.6589
## lambda56 57  0.010          -24.124     0.7996
## lambda57 58  0.603          -13.861     0.5841
## lambda58 59  0.196          -20.648     0.7657
## lambda59 60  0.077          -23.038     0.7612
## lambda60 61  0.136          -21.794     0.6836
## lambda61 62  0.010          -25.884     0.8440
## lambda62 63  1.575           -4.552     0.1435
```

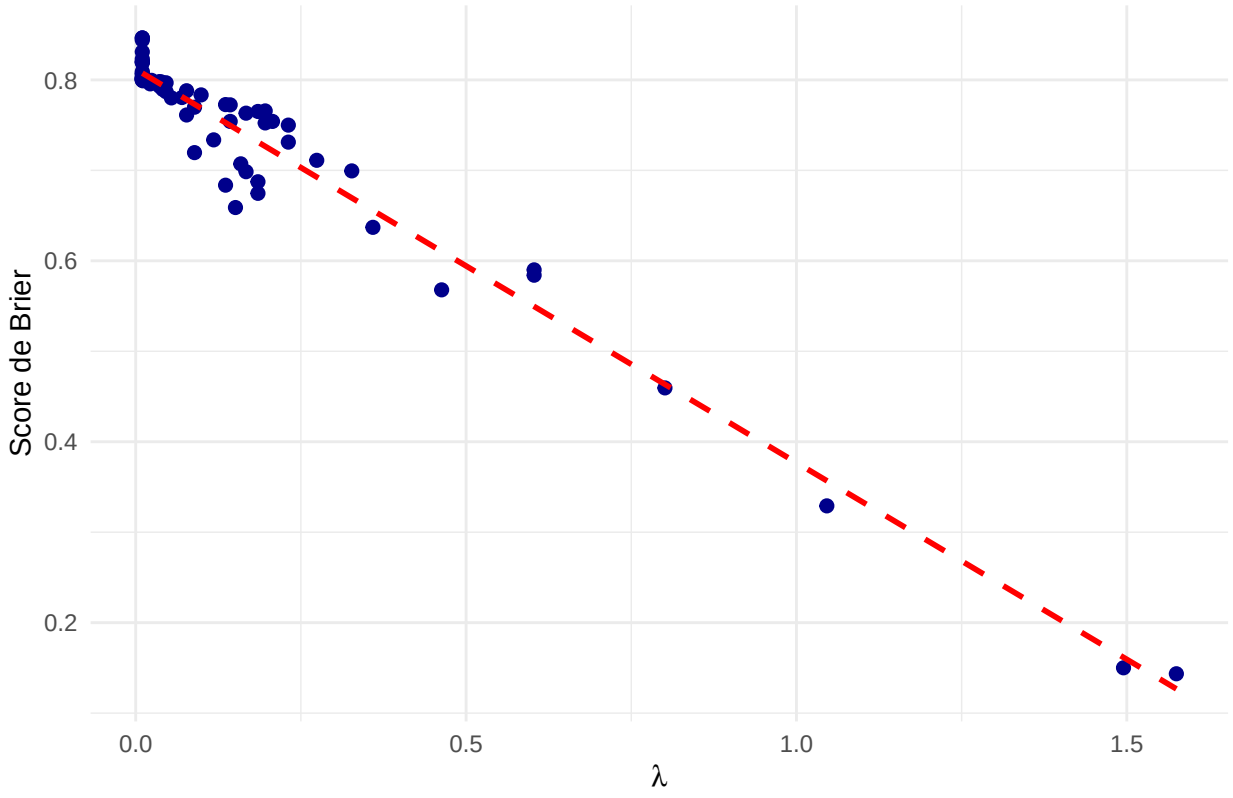
Visualisation

```
resultats_long <- resultats_qre %>%
  mutate(VraisemblanceMax_scaled = -VraisemblanceMax / 50) %>%
  pivot_longer(cols = c(lambda, VraisemblanceMax_scaled, BrierScore),
```

```
                names_to = "parametre", values_to = "valeur") %>%
  mutate(parametre = recode(parametre,
                            "lambda" = "Lambda ( )",
                            "VraisemblanceMax_scaled" = "-Log-vraisemblance",
                            "BrierScore" = "Score de Brier"))

ggplot(resultats_qre, aes(x = lambda, y = BrierScore)) +
  geom_point(color = "darkblue", size = 2) +
  geom_smooth(method = "lm", se = FALSE, color = "red", linetype = "dashed") +
  theme_minimal() +
  labs(
    title = "",
    x = expression(lambda),
    y = "Score de Brier"
  )
```

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



Estimation par question

```
resultats_par_question_qre <- data.frame()
```

```r
for (q in questions) {
  actions <- actions_par_question[[q]]
  bonus <- bonus_par_question[[q]]

  reponses <- sapply(donnees[[q]], function(rep) {
    if (!is.na(rep) && grepl(":", rep)) {
      as.numeric(trimws(strsplit(rep, ":")[[1]][2]))
    } else {
      as.numeric(trimws(rep))
    }
  })

  tryCatch({
    opt <- optim(
      par = c(lambda = 1),
      fn = function(par) {
        lambda <- par[1]
        total_loglik <- 0
        for (j in seq_along(reponses)) {
          r <- reponses[j]
          if (is.na(r) || !(r %in% actions)) next
          probs <- qre_strategy(lambda, actions, bonus$gauche, bonus$egal)
          idx <- which(actions == r)
          p_r <- probs[idx]
          total_loglik <- total_loglik + ifelse(p_r > 0, log(p_r), -1e6)
        }
        return(-total_loglik)
      },
      method = "L-BFGS-B", lower = 0.01, upper = 20
    )

    lambda_estime <- round(opt$par[1], 3)
    loglik <- round(-opt$value, 3)

    # Score de Brier
    brier_total <- 0
    n_valid <- 0

    for (j in seq_along(reponses)) {
      r <- reponses[j]
      if (is.na(r) || !(r %in% actions)) next
      probs <- qre_strategy(lambda_estime, actions, bonus$gauche, bonus$egal)
      y_true <- rep(0, length(actions))
      idx <- which(actions == r)
```

```
      y_true[idx] <- 1
      brier_total <- brier_total + sum((probs - y_true)^2)
      n_valid <- n_valid + 1
    }

    brier_moyen <- if (n_valid > 0) round(brier_total / n_valid, 4) else NA

    resultats_par_question_qre <- rbind(resultats_par_question_qre, data.frame(
      Question = q,
      lambda = lambda_estime,
      LogVraisemblance = loglik,
      Brier = brier_moyen
    ))

  }, error = function(e) {
    resultats_par_question_qre <- rbind(resultats_par_question_qre, data.frame(
      Question = q,
      lambda = NA,
      LogVraisemblance = NA,
      Brier = NA
    ))
    cat("Erreur pour la question", q, ":", e$message, "\n")
  })
}

# Affichage final
print(resultats_par_question_qre)
```

```
##                             Question lambda LogVraisemblance  Brier
## lambda   Q00_16-20 Arad & Rubinstein  0.102          -97.316 0.7814
## lambda1           Q00_Mode du groupe  0.065         -100.282 0.7924
## lambda2                 Q00_ChatGPT  0.080          -99.778 0.7883
## lambda3              Q00_Prime de 20  0.042          -99.382 0.7885
## lambda4                   Q01_12-20  0.032         -100.282 0.7924
## lambda5           Q02_Alaoui & Penta  0.082          -95.961 0.7577
## lambda6      Q03_Alaoui & Penta plus  0.016         -100.149 0.7928
## lambda7          Q04_Goeree moderate  0.026         -100.633 0.7944
## lambda8           Q05_Goeree extreme  0.044          -99.442 0.7836
## lambda9                  Q06_Cycle A  0.010         -102.038 0.8038
## lambda10                 Q07_Cycle C  0.010         -101.478 0.8004
## lambda11                   Q08_Pan A  0.028         -100.524 0.7938
## lambda12                  Q09_Bear C  0.010         -101.398 0.7999
## lambda13              Q10_long pan A  0.034          -93.830 0.7903
## lambda14              Q11_long pan C  0.019          -94.565 0.7973
```

Génération des graphiques

```
plots <- list()

for (q in 1:15) {
  actions <- actions_par_question[[q]]
  bonus <- bonus_par_question[[q]]

  reponses <- sapply(donneesQ[[q]], function(rep) {
    if (!is.na(rep) && grepl(":", rep)) {
      as.numeric(trimws(strsplit(rep, ":")[[1]][2]))
    } else {
      as.numeric(trimws(rep))
    }
  })

  tryCatch({
    opt <- optim(
      par = c(lambda = 1),
      fn = function(par) {
        lambda <- par[1]
        total_loglik <- 0
        for (j in seq_along(reponses)) {
          r <- reponses[j]
          if (is.na(r) || !(r %in% actions)) next
          probs <- qre_strategy(lambda, actions, bonus$gauche, bonus$egal)
          idx <- which(actions == r)
          p_r <- probs[idx]
          total_loglik <- total_loglik + ifelse(p_r > 0, log(p_r), -1e6)
        }
        return(-total_loglik)
      },
      method = "L-BFGS-B", lower = 0.01, upper = 20
    )

    lambda_estime <- opt$par[1]
    strategie_qre <- qre_strategy(lambda_estime, actions, bonus$gauche, bonus$egal)

    df_pred <- data.frame(
      Action = factor(actions, levels = actions),
      Valeur = strategie_qre,
      Type = "Prévu"
    )
```

```
    Q  <- donneesQ[[q]]
    Qr <- table(Q) / length(Q)
    df_obs <- data.frame(
      Action = factor(actions, levels = actions),
      Valeur = as.numeric(Qr),
      Type = "Observé"
    )

    df_combined <- rbind(df_pred, df_obs)

    p <- ggplot(df_combined, aes(x = Action, y = Valeur, fill = Type)) +
      geom_bar(stat = "identity", position = "dodge") +
      labs(title = paste("Question", q+1),
           x = "Action possible", y = "Fréquence") +
      scale_fill_manual(values = c("Observé" = "black", "Prévu" = "lightgrey")) +
      theme_minimal(base_size = 10) +
      theme(axis.text.x = element_text(angle = 45, hjust = 1),
            legend.position = "none")

    plots[[length(plots) + 1]] <- p

  }, error = function(e) {
    cat("Erreur pour la question", q, ":", e$message, "\n")
  })
}


# Suppression de l'ancien PDF
pdf_path <- "comparaison_QRE_5x3.pdf"
if (file.exists(pdf_path)) file.remove(pdf_path)
```

```
## [1] TRUE
```

```
# Export en PDF
library(ggpubr)
pdf(pdf_path, width = 20, height = 12)
ggarrange(plotlist = plots, ncol = 5, nrow = 3)
dev.off()
```

```
## pdf
##   2
```