Modèle Noisy Introspection

SCAIA Matteo - HACHEM REDA Riwa - GILLET Louison

2025-05-24

Introduction

Ce document présente une estimation individuelle des paramètres du modèle de Noisy Introspection basé sur les données collectées. Nous estimons pour chaque individu les paramètres μ (erreur de réponse) et τ (distribution du niveau de réflexion) par la méthode de maximum vraisemblance.

Chargement des bibliothèques et données

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

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")

# Listes d'actions et bonus
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 du modèle

```
logit_response <- function(payoffs, mu) {
  exp_payoff <- exp(payoffs / mu)
  exp_payoff / sum(exp_payoff)
}

ni_strategy <- function(mu, k, actions, bonus_gauche, bonus_egal) {
  probs <- rep(1 / length(actions), length(actions))  # Niveau 0
  for (i in 1:k) {
    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 <- probs[bonus_pos] * bonus_gauche
      } else {
        bonus_left <- 0
      }
      bonus_same <- probs[a_idx] * bonus_egal
      gain <- a + bonus_left + bonus_same
      return(gain)
    })
    probs <- logit_response(expected_payoff, mu)
  }
  return(probs)
}
```

Score de Brier

```
calcul_brier_score <- function(probs, reponse_obs, actions) {
  idx_obs <- which(actions == reponse_obs)
  brier <- sum((probs - ifelse(seq_along(actions) == idx_obs, 1, 0))^2)
  return(brier)
}

brier_score_total <- function(mu, s, reponses, questions, actions_map, k_max = 4) {
  total_brier <- 0
  poisson_weights <- dpois(0:k_max, lambda = s)
  poisson_weights <- poisson_weights / sum(poisson_weights)
  for (j in seq_along(reponses)) {
    r <- reponses[j]
    q <- questions[j]
    actions <- actions_map[[q]]
    bonus <- bonus_par_question[[q]]
    if (is.na(r) || !(r %in% actions)) next
    ni_levels <- lapply(0:k_max, function(k) {
      ni_strategy(mu, k, actions, bonus$gauche, bonus$egal)
    })
    mix_prob <- Reduce(`+`, Map(`*`, poisson_weights, ni_levels))
    total_brier <- total_brier + calcul_brier_score(mix_prob, r, actions)
  }
  return(total_brier/15)
}
```

Estimation des paramètres

```
log_likelihood <- function(mu, s, reponses, questions, actions_map, k_max = 4) {
  total_loglik <- 0
  poisson_weights <- dpois(0:k_max, lambda = s)
  poisson_weights <- poisson_weights / sum(poisson_weights)

  for (j in seq_along(reponses)) {
    r <- reponses[j]
    q <- questions[j]
    actions <- actions_map[[q]]
    bonus <- bonus_par_question[[q]]
    ni_levels <- lapply(0:k_max, function(k) {
      ni_strategy(mu, k, actions, bonus_gauche = bonus$gauche, bonus_egal = bonus$egal)
    })
```

```
    if (!is.na(r)) {
      idx <- which(actions == r)
      if (length(idx) == 1) {
        p_r <- sum(vapply(0:k_max, function(k) poisson_weights[k + 1] * ni_levels[[k +
      } 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)
}
```

Boucle d'estimation par individu

```
resultats <- 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(mu = 1.862, s = 1.393),
      fn = function(par) -log_likelihood(par[1], par[2], reponses, questions, actions_pa
      method = "L-BFGS-B",
      lower = c(0.001, 0.001), upper = c(1000, 1000)
    )
```

```
    mu_estime <- round(opt$par[1],3)
    s_estime <- round(opt$par[2],3)
    max_vraisemblance <- round(-opt$value,3)
    brier_score <- round(brier_score_total(mu_estime, s_estime, reponses, questions, ac

    resultats <- rbind(resultats, data.frame(id = i, mu = mu_estime, s = s_estime,
                                             VraisemblanceMax = max_vraisemblance, Brier

  }, error = function(e) {
    resultats <- rbind(resultats, data.frame(id = i, mu = NA, s = NA, VraisemblanceMax =
    cat("Erreur pour l'individu", i, ":", e$message, "\n")
  })
}

print(resultats)
```

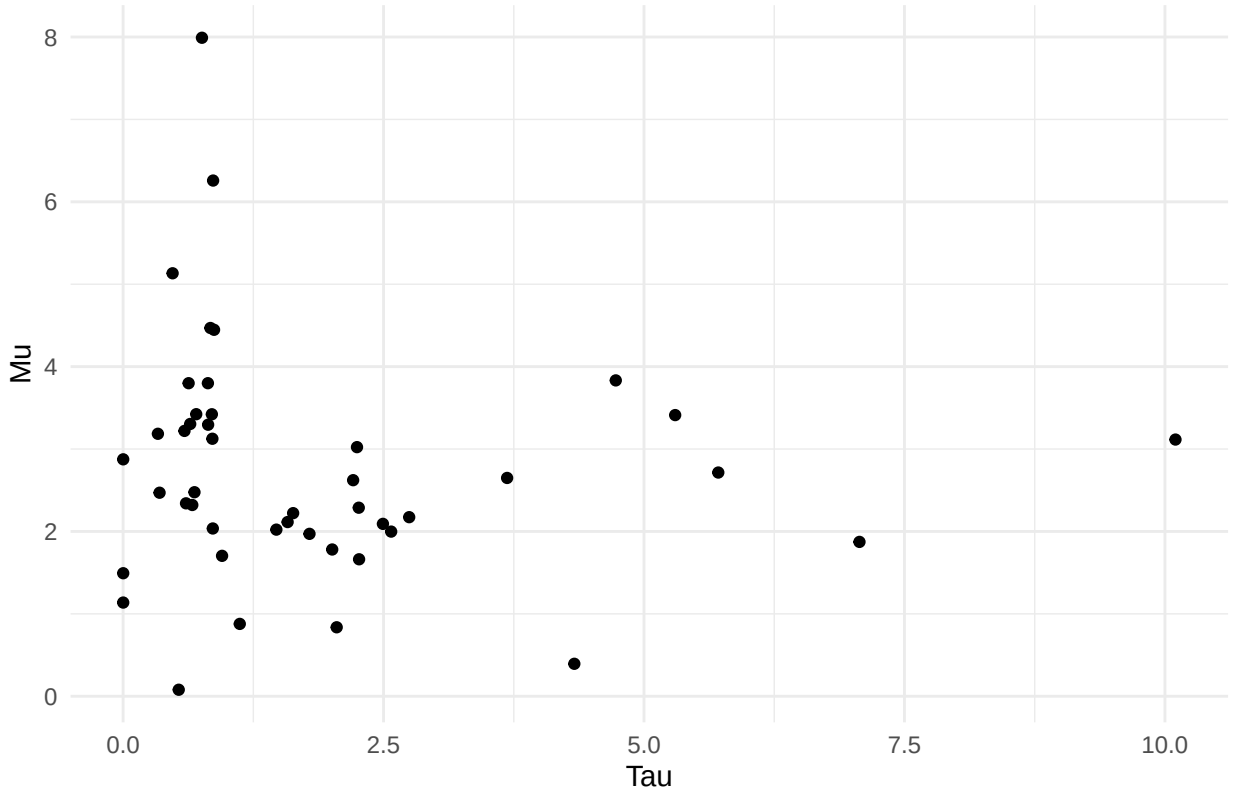
```
##       id       mu       s VraisemblanceMax Brier
## mu     1    3.125   0.857          -16.475 0.612
## mu1    2    3.412   5.299          -19.066 0.666
## mu2    3 1000.000   1.017          -24.168 0.801
## mu3    4    1.493   0.001           -7.511 0.259
## mu4    5    0.079   0.534          -11.336 0.390
## mu5    6 1000.000 102.129          -22.548 0.747
## mu6    7    0.394   4.331          -18.612 0.688
## mu7    8    2.476   0.685          -17.583 0.676
## mu8    9    0.878   1.120          -14.164 0.517
## mu9   10    2.622   2.208          -21.896 0.755
## mu10 11   13.982   0.970          -23.858 0.793
## mu11 12    1.781   2.007          -20.598 0.717
## mu12 13 1000.000 767.217          -24.158 0.800
## mu13 14    1.704   0.950          -17.308 0.622
## mu14 15    4.447   0.873          -21.065 0.736
## mu15 16    2.715   5.713          -17.982 0.624
## mu16 17    2.321   0.664          -17.979 0.686
## mu17 18    3.220   0.589          -17.026 0.615
## mu18 19    2.114   1.579          -17.177 0.601
## mu19 20    2.649   3.686          -25.610 0.848
## mu20 21    3.423   0.702          -21.189 0.744
## mu21 22 1000.000   1.012          -24.143 0.800
## mu22 23    6.258   0.864          -22.833 0.771
## mu23 24  169.726   1.015          -24.140 0.800
## mu24 25    0.837   2.050          -16.697 0.601
## mu25 26    3.799   0.814          -21.522 0.743
```

```
## mu26 27    2.036   0.861          -18.403 0.627
## mu27 28 1000.000 245.422          -20.942 0.694
## mu28 29    2.288   2.262          -20.712 0.738
## mu29 30    4.788 425.568          -19.156 0.657
## mu30 31    5.133   0.475          -22.442 0.751
## mu31 32   17.082   0.955          -23.973 0.797
## mu32 33    7.991   0.757          -23.480 0.790
## mu33 34   17.082   0.955          -23.973 0.797
## mu34 35    2.222   1.632          -20.260 0.693
## mu35 36 1000.000   1.018          -24.174 0.801
## mu36 37    2.092   2.494          -20.000 0.715
## mu37 38    2.022   1.471          -19.697 0.645
## mu38 39    1.137   0.001           -7.058 0.118
## mu39 40    4.469   0.838          -21.111 0.735
## mu40 41 1000.000   1.017          -24.170 0.801
## mu41 42    3.115  10.101          -22.650 0.771
## mu42 43    5.180 741.254          -22.557 0.760
## mu43 44 1000.000 768.479          -24.154 0.800
## mu44 45    3.023   2.245          -21.336 0.723
## mu45 46    2.341   0.604          -18.336 0.628
## mu46 47 1000.000   0.001          -24.144 0.800
## mu47 48    1.873   7.068          -18.809 0.675
## mu48 49    2.875   0.001          -17.053 0.612
## mu49 50    1.662   2.265          -20.440 0.693
## mu50 51    3.799   0.629          -21.558 0.730
## mu51 52 1000.000   1.018          -24.146 0.800
## mu52 53  119.116   0.991          -24.139 0.800
## mu53 54    2.173   2.746          -19.902 0.703
## mu54 55    1.998   2.573          -20.351 0.683
## mu55 56    3.422   0.852          -18.844 0.680
## mu56 57    3.833   4.729          -22.232 0.753
## mu57 58    2.470   0.350          -15.496 0.582
## mu58 59    1.971   1.789          -18.170 0.629
## mu59 60    3.185   0.334          -19.251 0.675
## mu60 61    3.304   0.644          -17.582 0.632
## mu61 62   14.598   0.948          -23.908 0.796
## mu62 63    3.295   0.816          -18.190 0.667
```

Visualisation des résultats

```
resultats_filtrés <- resultats[resultats$mu >= 0 & resultats$mu <= 11 &
                                 resultats$s >= 0 & resultats$s <= 11, ]
```

```r
ggplot(resultats_filtrés, aes(x = s, y = mu)) +
  geom_point() +
  theme_minimal() +
  labs(title = "", x = "Tau", y = "Mu")
```



Estimation par question

```r
log_likelihood_question <- function(mu, s, reponses, actions, bonus, k_max = 4) {
  poisson_weights <- dpois(0:k_max, lambda = s)
  poisson_weights <- poisson_weights / sum(poisson_weights)
  total_loglik <- 0

  for (j in seq_along(reponses)) {
    r <- reponses[j]
    if (is.na(r) || !(r %in% actions)) next
    ni_levels <- lapply(0:k_max, function(k) ni_strategy(mu, k, actions, bonus$gauche,
    idx <- which(actions == r)
    p_r <- sum(vapply(0:k_max, function(k) poisson_weights[k + 1] * ni_levels[[k + 1]][
    if (p_r > 0 && !is.na(p_r)) {
      total_loglik <- total_loglik + log(p_r)
    } else {
      total_loglik <- total_loglik - 1e6
```

```
    }
  }
  return(total_loglik)
}

brier_score_question <- function(mu, s, reponses, actions, bonus, k_max = 4) {
  poisson_weights <- dpois(0:k_max, lambda = s)
  poisson_weights <- poisson_weights / sum(poisson_weights)
  total_brier <- 0
  valid_n <- 0

  for (j in seq_along(reponses)) {
    r <- reponses[j]
    if (is.na(r) || !(r %in% actions)) next
    ni_levels <- lapply(0:k_max, function(k) ni_strategy(mu, k, actions, bonus$gauche, 
    mix_prob <- Reduce(`+`, Map(`*`, poisson_weights, ni_levels))
    total_brier <- total_brier + calcul_brier_score(mix_prob, r, actions)
    valid_n <- valid_n + 1
  }
  return(if (valid_n > 0) total_brier / valid_n else NA)
}

# Résultats par question
resultats_par_question <- data.frame()

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(mu = 1.5, s = 1.5),
      fn = function(par) -log_likelihood_question(par[1], par[2], reponses, actions, bo
      method = "L-BFGS-B", lower = c(0.001, 0.001), upper = c(1000, 1000)
    )

    mu_estime <- round(opt$par[1], 3)
```

```
    s_estime <- round(opt$par[2], 3)
    loglik <- round(-opt$value, 3)
    brier <- round(brier_score_question(mu_estime, s_estime, reponses, actions, bonus),

    resultats_par_question <- rbind(resultats_par_question,
                                    data.frame(Question = q, mu = mu_estime, s = s_estim
                                               LogVraisemblance = loglik, Brier = brier)

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


print(resultats_par_question)
```

```
##                          Question     mu       s LogVraisemblance Brier
## mu   Q00_16-20 Arad & Rubinstein  1.388   1.659          -94.150 0.752
## mu1           Q00_Mode du groupe  1.527   3.579          -99.028 0.785
## mu2                  Q00_ChatGPT  1.535   2.457          -95.668 0.766
## mu3              Q00_Prime de 20  4.405 735.341          -99.514 0.789
## mu4                   Q01_12-20  8.032   0.001          -99.094 0.784
## mu5           Q02_Alaoui & Penta  5.950   0.625          -95.023 0.747
## mu6      Q03_Alaoui & Penta plus 18.986 618.629         -100.804 0.796
## mu7          Q04_Goeree moderate  6.357  64.195          -98.977 0.779
## mu8           Q05_Goeree extreme  3.917   0.643          -91.294 0.725
## mu9                  Q06_Cycle A  7.815   0.336          -98.934 0.787
## mu10                 Q07_Cycle C  9.014   8.463          -99.743 0.790
## mu11                   Q08_Pan A  5.970   2.500          -97.698 0.777
## mu12                  Q09_Bear C  7.990   0.650          -99.029 0.785
## mu13              Q10_long pan A  4.322   0.511          -86.888 0.746
## mu14              Q11_long pan C 13.914  45.699          -94.450 0.797
```

Génération des graphiques

```
library(ggplot2)
library(ggpubr)
library(readxl)

# Liste pour stocker les graphiques
```

```r
plots <- list()

# Boucle sur les 15 questions
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(mu = 1.5, s = 1.5),
      fn = function(par) -log_likelihood_question(par[1], par[2], reponses, actions, bo
      method = "L-BFGS-B", lower = c(0.01, 0.01), upper = c(30, 30)
    )

    mu <- opt$par[1]
    s <- opt$par[2]
    k_max <- 4

    poids_k <- dpois(0:k_max, lambda = s)
    poids_k <- poids_k / sum(poids_k)

    strats_k <- lapply(0:k_max, function(k) {
      ni_strategy(mu, k, actions, bonus$gauche, bonus$egal)
    })

    strategie_ni <- Reduce(`+`, Map(`*`, poids_k, strats_k))

    df_pred <- data.frame(
      Action = factor(actions, levels = actions),
      Valeur = strategie_ni,
      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 lors du traitement de la question", q, ":", e$message, "\n")
  })
}

# Exporter tous les graphiques sur une seule page PDF (5 colonnes × 3 lignes)
pdf("comparaison_superposee_5x3.pdf", width = 20, height = 12)
ggarrange(plotlist = plots, ncol = 5)
```

```
## $`1`
```

```
##
## $`2`
```

```
##
## $`3`
```

```
##
## attr(,"class")
## [1] "list"      "ggarrange"
```

```
dev.off()
```

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