Parameter recovery using summary statistics

Adriana F. Chávez De la Peña

2024-12-27

Single simulation

We start by **specifying our simulation environment**.

```
# True hierarchical means
drift_mean <- 0
bound_mean <- 1.5
nondt_mean <- 0.4

# Design settings
nParticipants <- 50
nTrials <- 150
```

We create a function to **sample individual parameters** from hierarchical distributions.

```
# Load library to sample from truncated normals
library(truncnorm)

# Create function
getIndPar <- function(n_participants, dmean, bmean, nmean){
  d <- rtruncnorm(n = n_participants, mean = dmean, sd = 1)
  b <- rtruncnorm(n = n_participants, a = 0, mean = bmean, sd = 0.5)
  n <- rtruncnorm(n = n_participants, a = 0, mean = nmean, sd = 0.15)
  return(data.frame("bound" = b, "drift" = d, "nondt" = n))
}

# Create a set of individual parameters
indPar <- getIndPar(n_participants = nParticipants, dmean = drift_mean,
                    bmean = bound_mean, nmean = nondt_mean)
head(indPar)
```

```
##       bound      drift     nondt
## 1 2.2822559  0.3889554 0.2403297
## 2 2.0151772  2.1401645 0.3051538
## 3 0.5501789 -0.2683785 0.6964738
## 4 1.1869556 -0.6032781 0.6484686
## 5 1.4422403  0.9629682 0.4946537
## 6 0.8212650  1.6479620 0.3313863
```

We then build a function to **simulate sample statistics based on the probabilistic EZ DDM system of equations**

```
# Write function
simSumStats <- function(indiv_pars, n_trials){
  v <- indiv_pars$drift
  a <- indiv_pars$bound
  t <- indiv_pars$nondt
  N <- n_trials
```

```
  nP <- length(v)

  y <- exp(-a * v)
  # Forward EZ equations
  PredAccuracyRate <- 1 / (1 + y)  # Equation 1
  PredMean <- t + ((a / (2 * v)) * ((1 - y) / (1 + y)))  # Equation 2
  PredVariance <- (a / (2 * v^3)) * (( 1 - 2 * a * v * y - exp(-a*2*v)) / ((y + 1)^2))  # Equation 3
  # Samplers
  ObservedAccuracyTotal <- rbinom(nP, size = N, prob = PredAccuracyRate)
  # Random sampler for ObservedMean
  ObservedMean <- rnorm(nP, mean = PredMean, sd = sqrt(PredVariance / N))
  # Random sampler for ObservedVariance
  ObservedVariance <- rnorm(nP, mean = PredVariance, sd = sqrt((2 * (PredVariance^2)) / (N - 1)))

  return(data.frame("A" = ObservedAccuracyTotal,
                    "Mrt" = ObservedMean,
                    "Vrt" = ObservedVariance))
}

# Simulate summary statistics per participant, based on individual parameters
getStats <- simSumStats(indiv_pars = indPar, n_trials = nTrials)
head(getStats)
```

```
##     A       Mrt         Vrt
## 1 105 1.4626804 0.915215292
## 2 146 0.7492467 0.087573360
## 3  72 0.7727313 0.003324808
## 4  52 0.9712224 0.070905637
## 5 123 0.9211493 0.147941206
## 6 123 0.4844872 0.013819967
```

We write a general **JAGS model** to estimate the hierarchical mean parameters from the summary statistics simulated.

```
write("
    model {
      bound_mean ~ dnorm(1.5, pow(0.5, -2))T(0.10,)
      nondt_mean ~ dnorm(0.4, pow(0.15, -2))T(0.05,)
      drift_mean ~ dnorm(0, pow(1, -2))

      bound_sdev ~ dunif(0.1, 0.5)
      nondt_sdev ~ dunif(0.05, 0.5)
      drift_sdev ~ dunif(0.1, 0.5)

      # Sampling model
      for (p in 1:n_Participants) {
        bound[p] ~ dnorm(bound_mean, pow(bound_sdev, -2))T(0.10,)
        nondt[p] ~ dnorm(nondt_mean, pow(nondt_sdev, -2))T(0.05,)
        drift[p] ~ dnorm(drift_mean, pow(drift_sdev, -2))

        # Forward equations from EZ Diffusion
        ey[p] = exp(-bound[p] * drift[p])
        Pc[p] = 1 / (1 + ey[p])
        PRT[p] = 2 * pow(drift[p], 3) / bound[p] * pow(ey[p] + 1, 2) / (2 * -bound[p] * drift[p] * ey[p] -
        MDT[p] = (bound[p] / (2 * drift[p])) * (1 - ey[p]) / (1 + ey[p])
        MRT[p] = MDT[p] + nondt[p]
```

```
        # Loss functions using MRT, PRT, and Pc
        correct[p] ~ dbin(Pc[p], n_trials)
        meanRT[p] ~ dnorm(MRT[p], PRT[p] * n_trials)
        varRT[p] ~ dnorm(1 / PRT[p], 0.5 * (n_trials - 1) * PRT[p] * PRT[p])
      }
    }
", file = "./model.bug")
```

We specify our **JAGS settings**

```
# Define MCMC chains
n.chains <- 4
n.burnin <- 500
n.iter <- 2500
n.thin <- 1

# Set up arbitrary initial values
myinits <- rep(list(list()), n.chains)
  for(i in 1:n.chains){
    myinits[[i]] <- list(drift = rnorm(nParticipants,0,0.5))
  }

# Pass data to JAGS
data_toJAGS <- list(n_Participants = nParticipants,
                    n_trials = nTrials,
                    correct = getStats$A,
                    meanRT = getStats$Mrt,
                    varRT = getStats$Vrt)
```

Run JAGS

```
library(R2jags)

samples <- jags(data=data_toJAGS,
                parameters.to.save=c("drift_mean", "bound_mean", "nondt_mean"),
                model="./model.bug",
                n.chains=n.chains, n.iter=n.iter,
                n.burnin=n.burnin, n.thin=n.thin,
                DIC=T, inits=myinits)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 150
##    Unobserved stochastic nodes: 156
##    Total graph size: 1576
##
## Initializing model
```

```
# Load hierarchical mean parameters
drift <- samples$BUGSoutput$sims.list$drift_mean
bound <- samples$BUGSoutput$sims.list$bound_mean
nondt <- samples$BUGSoutput$sims.list$nondt_mean
```

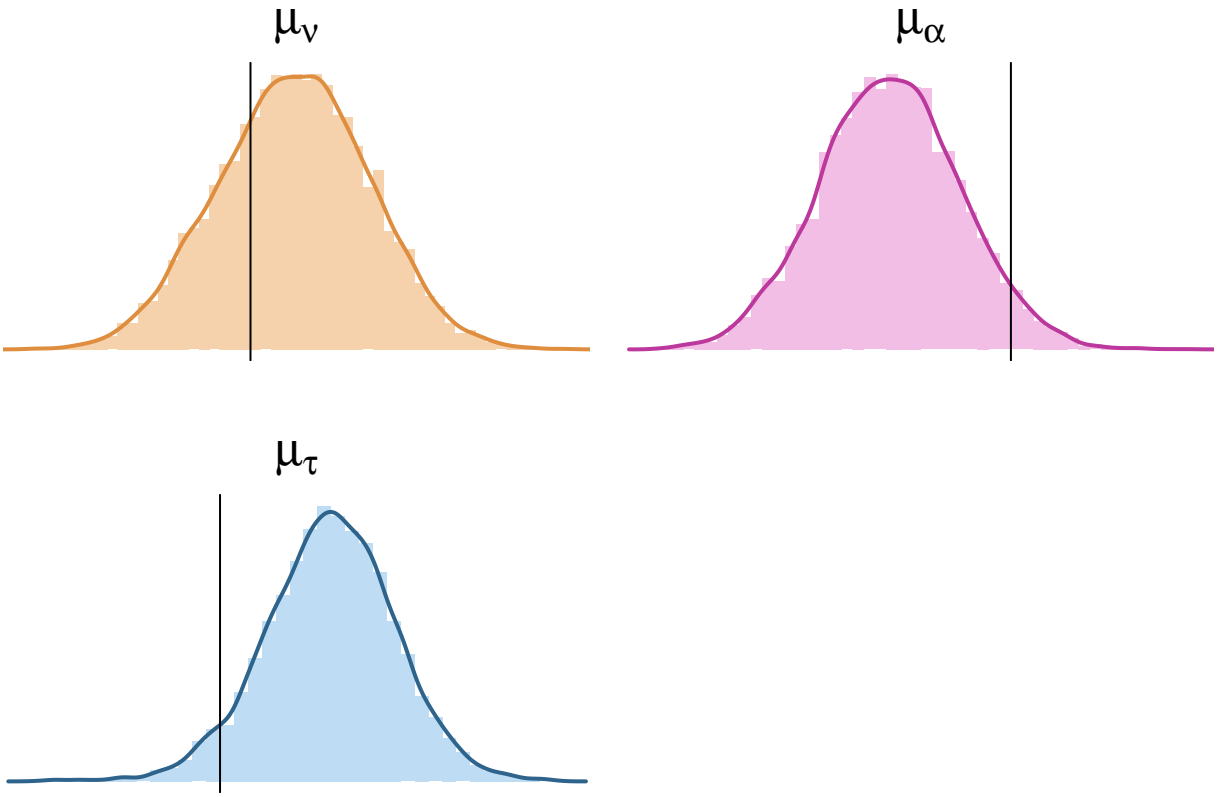
We explore the **posterior distributions** obtained per hierarchical mean parameter.

```
par(mfrow=c(2,2), mai=c(0.2,0.1,0.5,0.1))# oma=c(2.5,1.75,upper_margin,right_margin))
hist(drift, ann=F, axes=F, col=drift_color1, border = NA, freq = FALSE, breaks = 50)
```

```
lines(density(drift), col=drift_color2, lwd=2)
mtext(expression(paste(mu[nu])),3, line=0.4, cex=1.5)
abline(v=drift_mean)

hist(bound, ann=F, axes=F, col=bound_color1, border = NA, freq = FALSE, breaks = 50)
lines(density(bound), col=bound_color2, lwd=2)
mtext(expression(paste(mu[alpha])),3, line=0.4, cex=1.5)
abline(v=bound_mean)

hist(nondt, ann=F, axes=F, col=nondt_color1, border = NA, freq = FALSE, breaks = 50)
lines(density(nondt), col=nondt_color2, lwd=2)
mtext(expression(paste(mu[tau])),3, line=0.4, cex=1.5)
abline(v=nondt_mean)
```



Overall, neither of these posterior distributions is fully centered around the true value (marked by the vertical lines), but they do overlap with them.

Let's evaluate the convergence of these posterior chains by computing the $\hat{R}$ statistic. We usually say convergence is good if $\hat{R} < 1.05$

```
# Load a function to compute Rhat statistic
source("https://raw.githubusercontent.com/Adrifelcha/EZ-project/refs/heads/main/code/functions/rhat.R")
# Compute Rhats using our function
rhats <- apply(samples$BUGSoutput$sims.array,3,getRhat)

# Verify none of the Rhats are larger than 1.05
rule <- 1.05
bad.Rhat <- which(rhats>rule)
test.rhat <- length(bad.Rhat) > 0
  if(test.rhat){
          par(mfrow=c(1,1))
          which.are.bad.Rhats <- names(bad.Rhat)
```

```
        hist(rhats, breaks = 50)
        abline(v=rule, col="red", lty=2)
        legend("top",paste("Rhat > ",rule," | ",
                           (round(nrow(bad.Rhat)/(length(as.vector(rhats))),5))*100,
                           "% of chains | ", length(which.are.bad.Rhats), " chains", sep=""), lty=2, col=
        table(which.are.bad.Rhats)
  }else{      paste("No Rhat greater than ", rule, sep="")       }
```

```
## [1] "No Rhat greater than 1.05"
```

Run a complete simulation study

The results shown before correspond to a single simulation. Let's explore what happens if we repeat this exercise many times, by simulating `nDatasets`.

```
nDatasets <- 100
```

To make this endeavour easier, I've created a lttle script called `load_functions.R` that calls the necessary custom functions to run the simmulation study.

```
setwd("./scripts/")
source("./load_functions.R")
```

And now, we use our custom function `runSims()` to conduct a **full simulation study** where we generate `nDatasets` with the same size (`nParticipants` and `nTrials`). Each dataset has its own set of hierarchical mean parameters, sampled from the prior distribution. Then, for each dataset we fit the same JAGS model to obtain hierarchical mean estimates.

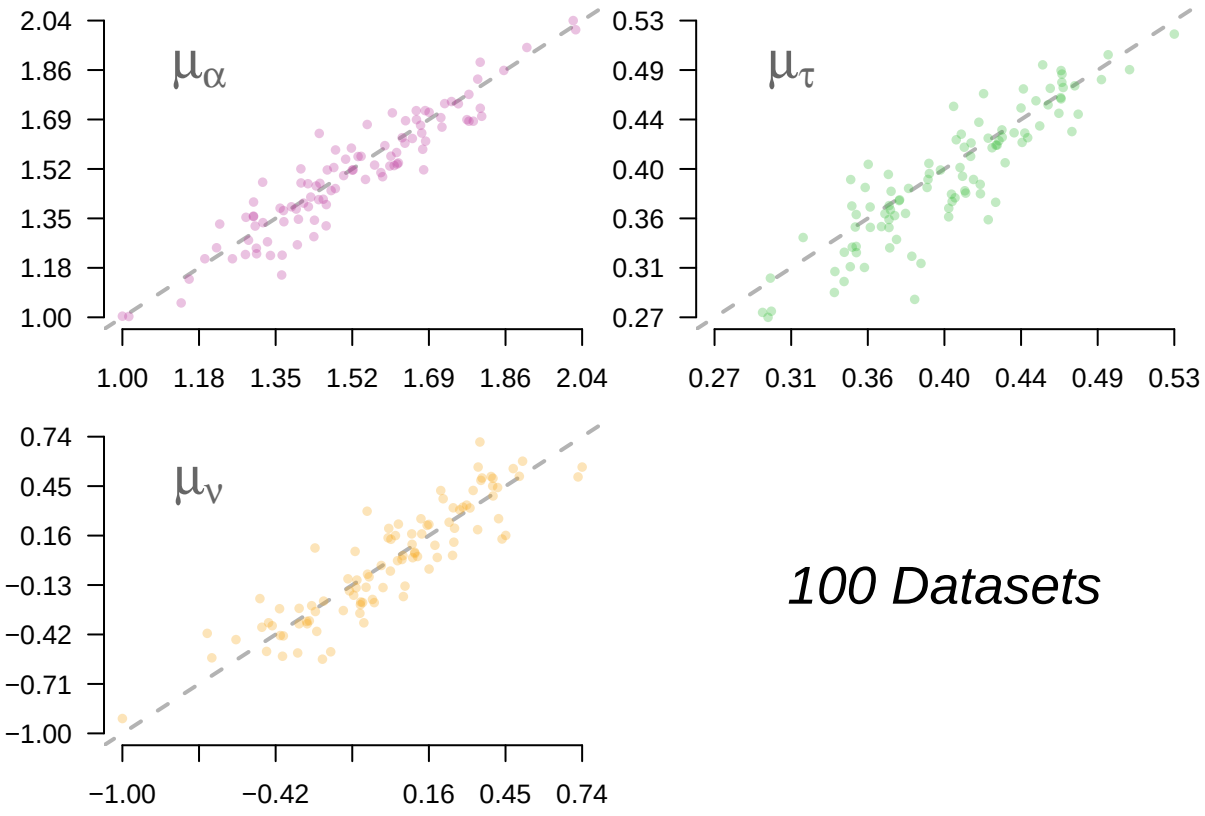
We are interested in exploring the adequacy of these hierarchical mean estimations over the course of many iterations.

```
# Run simulation study
sim_study <- runSims(nParticipants = nParticipants, nTrials = nTrials,
                     nDatasets = nDatasets,
                     modelType = "hierarchical",   # Hierarchical DDM model
                     modelFile="./model.bug",      # JAGS model
                     # Pass JAGS settings
                     n.chains = n.chains, n.burnin= n.burnin,
                     n.iter= n.iter, n.thin= n.thin,
                     # Check Rhat statistics
                     rhatCheck=TRUE,
                     # Check only hierarchical mean parameters
                     track_allParameters = FALSE,
                     # Repeat iterations with Rhat > 1.05
                     redo_if_bad_rhat=FALSE,
                     # Folder to store results
                     output.folder = "./results/")
```

```
## This simulation had been run before.
## Loading stored results: COMPLETE!
##  Running this simulation study took  8.366667 minutes.
## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~!!!
## R-hat check (Hierarchical mean parameters):
## No Rhat greater than 1.05
## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~!!!
```

We end by using the custom function `plot_recovery()` to display the parameter recovery performance of our model across all datasets.

```
plot_recovery(sim_study,plotType = 1)
plot(2,3, ann=F, type="n", axes=F)
text(2,3,paste(sim_study$settings$nDatasets, "Datasets"), f=3, cex=2)
```



100 Datasets

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