```
#install.packages("metafor")
#install.packages("tidyverse")
#install.packages("readxl")
#install.packages("dplyr")
#install.packages("RoBMA")



library(metafor)
library(tidyverse)
library(readxl)
library(dplyr)
#library(RoBMA)

xlsx_file   <- "
MA_GAISTEM_ModAnalysis_Reduced_CBNP_last2_KvsS_20260525_red_res.xlsx"

dat1 <- read_excel(xlsx_file)

# ---------------------------
# Random Effects MA
# ---------------------------

dat1$yi <- as.numeric(gsub(",", ".", dat1$yi))
dat1$vi <- as.numeric(gsub(",", ".", dat1$vi))

dat1$intn <- as.numeric(dat1$intn)
dat1$cn <- as.numeric(dat1$cn)

summary(dat1$yi)
```

```
summary(dat1$vi)

dat1 <- dat1[!is.na(dat1$yi) & !is.na(dat1$vi), ]

cat("Number of valid studies:", nrow(dat1), "\n")

overallresult <- rma(yi, vi, data=dat1)

overallresult

predict(overallresult)

tau2_overall <- as.numeric(overallresult$tau2)
vi_bar <- mean(dat1$vi, na.rm = TRUE)
cat("tau^2 =", round(tau2_overall, 4), " mean vi =", round(vi_bar,4), "\n")

forest.rma(overallresult, slab = dat1$Study, header="Study")

##check for influence
#influence analysis
influence(overallresult)

####### Hedges & Pigott approx power for 2-group moderator ########

power_mod_2groups <- function(k1, k2, tau2, vi_bar, delta = 0.4, alpha = 0.05) {
 V_diff <- tau2 * (1/k1 + 1/k2) + vi_bar * (1/k1 + 1/k2)
 se_diff <- sqrt(V_diff)
 z_crit <- qnorm(1 - alpha/2)
 ncp <- delta / se_diff
```

```
 power <- 1 - pnorm(z_crit, mean = ncp, sd = 1) + pnorm(-z_crit, mean = ncp, sd = 1)
 return(list(power = power, se_diff = se_diff, V_diff = V_diff))
}

######## Power summaries for moderators (Hedges & Pigott approx) ########

moderators <- c("domain","ed_level","GAI_system",

"GAI_type","thinking_skill","adapted_system","learning_outcome","interaction_duration",
"ICAP_level_avg", "ICAP_level_AuS")

power_results <- data.frame(moderator = character(),
                level1 = character(), k1 = integer(),
                level2 = character(), k2 = integer(),
                tau2 = numeric(), vi_bar = numeric(),
                power_delta_0.2 = numeric(), power_delta_0.4 = numeric(),
power_delta_0.6 = numeric(),
                stringsAsFactors = FALSE)

for (modname in moderators) {
 if (! modname %in% names(dat1)) next
 counts <- dat1 %>% count(!!sym(modname), name = "k")

 if (nrow(counts) < 2) {
  next
 }
  counts <- counts[order(-counts$k),]
  lvl1 <- as.character(counts[[1,1]])
  k1 <- counts[[1,"k"]]
```

```
  lvl2 <- as.character(counts[[2,1]])
  k2 <- counts[[2,"k"]]

  p20 <- power_mod_2groups(k1 = k1, k2 = k2, tau2 = tau2_overall, vi_bar = vi_bar, delta
= 0.2)$power
  p40 <- power_mod_2groups(k1 = k1, k2 = k2, tau2 = tau2_overall, vi_bar = vi_bar, delta
= 0.4)$power
  p60 <- power_mod_2groups(k1 = k1, k2 = k2, tau2 = tau2_overall, vi_bar = vi_bar, delta
= 0.6)$power

  power_results <- rbind(power_results,
                         data.frame(moderator = modname,
                                    level1 = lvl1, k1 = k1,
                                    level2 = lvl2, k2 = k2,
                                    tau2 = tau2_overall, vi_bar = vi_bar,
                                    power_delta_0.2 = p20, power_delta_0.4 = p40, power_delta_0.6 =
p60,
                                    stringsAsFactors = FALSE))
 }

 power_results <- power_results %>% mutate(across(starts_with("power_delta"),
~round(.x,3)))
 print(power_results)
 write.csv(power_results, "ModeratorPowerSummary_update.csv", row.names = FALSE)

 ############ Moderator Analysis #########

 #####domain######

 mod.domainq <- rma(yi, vi, mods = ~ factor(domain), data=dat1)
```

```
mod.domainq


Qdomain_collapsed_string <- paste(mod.domainq[["QMdf"]])
Qdomain_type1 <- data.frame(CollapsedQMdf = Qdomain_collapsed_string)
Qdomain_type2 <- round(mod.domainq$QM,2)
Qdomain_type3 <- round(mod.domainq$QMp,3)


QdomainQ <- paste(
  "Qb(",Qdomain_collapsed_string,") =",
  Qdomain_type2,
  ", p =",
  Qdomain_type3,
  collapse = " "
)


cat(QdomainQ, "\n")
domainQtest <- data.frame(Text = QdomainQ)
write.table(domainQtest, file = "QdomainQ.txt", row.names = FALSE, col.names =
FALSE, quote = FALSE)


mod.domain <- rma(yi, vi, mods = ~ factor(domain)-1, data=dat1)
mod.domain


mod.domain_table <-coef(summary(mod.domain))
domainSumParticipants <- dat1 %>%
  group_by(domain) %>%
  summarise(nexp = sum(intn, na.rm = TRUE),
            nctrl = sum(cn, na.rm = TRUE))
domainNumComp <- dat1 %>%
```

```
  count(domain)
 domainNumComp <- rename(domainNumComp, kcomparisons = n)
 domain_type_table.final<- cbind(domainSumParticipants,domainNumComp[c(2)],
mod.domain_table)
 write.csv(domain_type_table.final, " Mod.domainResult_update.csv",sep = ";")

 ###### educational level ######

 mod.edlevelq <- rma(yi, vi, mods = ~ factor(ed_level), data=dat1)
 mod.edlevelq

 Qedlevel_collapsed_string <- paste(mod.edlevelq[["QMdf"]])
 Qedlevel_type1 <- data.frame(CollapsedQMdf = Qedlevel_collapsed_string)
 Qedlevel_type2 <- round(mod.edlevelq$QM,2)
 Qedlevel_type3 <- round(mod.edlevelq$QMp,3)

 QedlevelQ <- paste(
  "Qb(",Qedlevel_collapsed_string,") =",
  Qedlevel_type2,
  ", p =",
  Qedlevel_type3,
  collapse = " "
 )

 cat(QedlevelQ, "\n")
 edlevelQtest <- data.frame(Text = QedlevelQ)
 write.table(edlevelQtest, file = "QedlevelQ_updated.txt", row.names = FALSE,
col.names = FALSE, quote = FALSE)
```

```
mod.edlevel <- rma(yi, vi, mods = ~ factor(ed_level)-1, data=dat1)
mod.edlevel

mod.edlevel_table <-coef(summary(mod.edlevel))
edlevelSumParticipants <- dat1 %>%
  group_by(ed_level) %>%
  summarise(nexp = sum(intn, na.rm = TRUE),
            nctrl = sum(cn, na.rm = TRUE))
edlevelNumComp <- dat1 %>%
  count(ed_level)
edlevelNumComp <- rename(edlevelNumComp, kcomparisons = n)
edlevel_type_table.final<- cbind(edlevelSumParticipants,edlevelNumComp[c(2)],
mod.edlevel_table)
write.csv(edlevel_type_table.final, " /3. Runde/Mod.edlevelResult_oLin.csv",sep = ";")

##### GAI system ######

mod.GAI_systemq <- rma(yi, vi, mods = ~ factor(GAI_system), data=dat1)
mod.GAI_systemq


QGAI_system_collapsed_string <- paste(mod.GAI_systemq[["QMdf"]])
QGAI_system_type1 <- data.frame(CollapsedQMdf = QGAI_system_collapsed_string)
QGAI_system_type2 <- round(mod.GAI_systemq$QM,2)
QGAI_system_type3 <- round(mod.GAI_systemq$QMp,3)

QGAI_systemQ <- paste(
  "Qb(",QGAI_system_collapsed_string,") =",
```

```
  QGAI_system_type2,
  ", p =",
  QGAI_system_type3,
  collapse = " "
 )

 cat(QGAI_systemQ, "\n")
 GAI_systemQtest <- data.frame(Text = QGAI_systemQ)
 write.table(GAI_systemQtest, file = " /3. Runde/QGAI_system_udpated.txt", row.names
= FALSE, col.names = FALSE, quote = FALSE)


 mod.GAI_system <- rma(yi, vi, mods = ~ factor(GAI_system)-1, data=dat1)
 mod.GAI_system


 mod.GAI_system_table <-coef(summary(mod.GAI_system))
 GAI_systemSumParticipants <- dat1 %>%
  group_by(GAI_system) %>%
  summarise(nexp = sum(intn, na.rm = TRUE),
        nctrl = sum(cn, na.rm = TRUE))
 GAI_systemNumComp <- dat1 %>%
  count(GAI_system)
 GAI_systemNumComp <- rename(GAI_systemNumComp, kcomparisons = n)
 GAI_system_type_table.final<-
cbind(GAI_systemSumParticipants,GAI_systemNumComp[c(2)],
mod.GAI_system_table)
 write.csv(GAI_system_type_table.final, " /3.
Runde/Mod.GAI_systemResult_updated.csv",sep = ";")
```

```
###### Type of GAI ####

mod.GAI_typeq <- rma(yi, vi, mods = ~ factor(GAI_type), data=dat1)
mod.GAI_typeq

QGAI_type_collapsed_string <- paste(mod.GAI_typeq[["QMdf"]])
QGAI_type_type1 <- data.frame(CollapsedQMdf = QGAI_type_collapsed_string)
QGAI_type_type2 <- round(mod.GAI_typeq$QM,2)
QGAI_type_type3 <- round(mod.GAI_typeq$QMp,3)

QGAI_typeQ <- paste(
  "Qb(",QGAI_type_collapsed_string,") =",
  QGAI_type_type2,
  ", p =",
  QGAI_type_type3,
  collapse = " "
)

cat(QGAI_typeQ, "\n")
GAI_typeQtest <- data.frame(Text = QGAI_typeQ)
write.table(GAI_typeQtest, file = " /3. Runde/QGAI_typeQ_updated.txt", row.names =
FALSE, col.names = FALSE, quote = FALSE)

mod.GAI_type <- rma(yi, vi, mods = ~ factor(GAI_type)-1, data=dat1)
mod.GAI_type

mod.GAI_type_table <-coef(summary(mod.GAI_type))

GAI_typeSumParticipants <- dat1 %>%
```

```
  group_by(GAI_type) %>%
  summarise(nexp = sum(intn, na.rm = TRUE),
            nctrl = sum(cn, na.rm = TRUE))
 GAI_typeNumComp <- dat1 %>%
  count(GAI_type)
 GAI_typeNumComp <- rename(GAI_typeNumComp, kcomparisons = n)
 GAI_type_type_table.final<- cbind(GAI_typeSumParticipants,GAI_typeNumComp[c(2)],
mod.GAI_type_table)
 write.csv(GAI_type_type_table.final, " /3. Runde/Mod.GAI_typeResult_oLin.csv")


 ####Type of thinking skill####

 mod.thinking_skillq <- rma(yi, vi, mods = ~ factor(thinking_skill), data=dat1)
 mod.thinking_skillq


 Qthinking_skill_collapsed_string <- paste(mod.thinking_skillq[["QMdf"]])
 Qthinking_skill_type1 <- data.frame(CollapsedQMdf =
Qthinking_skill_collapsed_string)
 Qthinking_skill_type2 <- round(mod.thinking_skillq$QM,2)
 Qthinking_skill_type3 <- round(mod.thinking_skillq$QMp,3)

 Qthinking_skillQ <- paste(
  "Qb(",Qthinking_skill_collapsed_string,") =",
  Qthinking_skill_type2,
  ", p =",
  Qthinking_skill_type3,
  collapse = " "
```

```
)

cat(Qthinking_skillQ, "\n")

thinking_skillQtest <- data.frame(Text = Qthinking_skillQ)

write.table(thinking_skillQtest, file = " /3. Runde/Qthinking_skillQ_updated.txt",
row.names = FALSE, col.names = FALSE, quote = FALSE)


mod.thinking_skill <- rma(yi, vi, mods = ~ factor(thinking_skill)-1, data=dat1)

mod.thinking_skill


mod.thinking_skill_table <-coef(summary(mod.thinking_skill))


thinking_skillSumParticipants <- dat1 %>%

  group_by(thinking_skill) %>%

  summarise(nexp = sum(intn, na.rm = TRUE),

        nctrl = sum(cn, na.rm = TRUE))

thinking_skillNumComp <- dat1 %>%

  count(thinking_skill)

thinking_skillNumComp <- rename(thinking_skillNumComp, kcomparisons = n)

thinking_skill_type_table.final<-
cbind(thinking_skillSumParticipants,thinking_skillNumComp[c(2)],
mod.thinking_skill_table)

write.csv(thinking_skill_type_table.final, " /3.
Runde/Mod.thinking_skillResult_updated.csv")



#### Adapted system ####


mod.adapted_systemq <- rma(yi, vi, mods = ~ factor(adapted_system), data=dat1)

mod.adapted_systemq
```

```
 Qadapted_system_collapsed_string <- paste(mod.adapted_systemq[["QMdf"]])
 Qadapted_system_type1 <- data.frame(CollapsedQMdf =
Qadapted_system_collapsed_string)
 Qadapted_system_type2 <- round(mod.adapted_systemq$QM,2)
 Qadapted_system_type3 <- round(mod.adapted_systemq$QMp,3)

 Qadapted_systemQ <- paste(
  "Qb(",Qadapted_system_collapsed_string,") =",
  Qadapted_system_type2,
  ", p =",
  Qadapted_system_type3,
  collapse = " "
 )

 cat(Qadapted_systemQ, "\n")
 adapted_systemQtest <- data.frame(Text = Qadapted_systemQ)
 write.table(adapted_systemQtest, file = " /3. Runde/Qadapted_systemQ_updated.txt",
row.names = FALSE, col.names = FALSE, quote = FALSE)


 mod.adapted_system <- rma(yi, vi, mods = ~ factor(adapted_system)-1, data=dat1)
 mod.adapted_system

 mod.adapted_system_table <-coef(summary(mod.adapted_system))

 adapted_systemSumParticipants <- dat1 %>%
  group_by(adapted_system) %>%
```

```
  summarise(nexp = sum(intn, na.rm = TRUE),
            nctrl = sum(cn, na.rm = TRUE))
 adapted_systemNumComp <- dat1 %>%
  count(adapted_system)
 adapted_systemNumComp <- rename(adapted_systemNumComp, kcomparisons = n)
 adapted_system_type_table.final<-
cbind(adapted_systemSumParticipants,adapted_systemNumComp[c(2)],
mod.adapted_system_table)
 write.csv(adapted_system_type_table.final, " /3.
Runde/Mod.adapted_systemResult_updated.csv")

 #### Learning outcome####

 mod.learning_outcomeq <- rma(yi, vi, mods = ~ factor(learning_outcome), data=dat1)
 mod.learning_outcomeq

 Qlearning_outcome_collapsed_string <- paste(mod.learning_outcomeq[["QMdf"]])
 Qlearning_outcome_type1 <- data.frame(CollapsedQMdf =
Qlearning_outcome_collapsed_string)
 Qlearning_outcome_type2 <- round(mod.learning_outcomeq$QM,2)
 Qlearning_outcome_type3 <- round(mod.learning_outcomeq$QMp,3)

 Qlearning_outcomeQ <- paste(
  "Qb(",Qlearning_outcome_collapsed_string,") =",
  Qlearning_outcome_type2,
  ", p =",
  Qlearning_outcome_type3,
  collapse = " "
 )
```

```
 cat(Qlearning_outcomeQ, "\n")

 learning_outcomeQtest <- data.frame(Text = Qlearning_outcomeQ)

 write.table(learning_outcomeQtest, file = " /3.
Runde/Qlearning_outcomeQ_updated.txt", row.names = FALSE, col.names = FALSE,
quote = FALSE)


 mod.learning_outcome <- rma(yi, vi, mods = ~ factor(learning_outcome)-1, data=dat1)

 mod.learning_outcome


 mod.learning_outcome_table <-coef(summary(mod.learning_outcome))


 learning_outcomeSumParticipants <- dat1 %>%

  group_by(learning_outcome) %>%

  summarise(nexp = sum(intn, na.rm = TRUE),

        nctrl = sum(cn, na.rm = TRUE))

 learning_outcomeNumComp <- dat1 %>%

  count(learning_outcome)

 learning_outcomeNumComp <- rename(learning_outcomeNumComp, kcomparisons =
n)

 learning_outcome_type_table.final<-
cbind(learning_outcomeSumParticipants,learning_outcomeNumComp[c(2)],
mod.learning_outcome_table)

 write.csv(learning_outcome_type_table.final, " /3.
Runde/Mod.learning_outcomeResult_updated.csv")


 ##### Interaction duration #####


 dat1_idur <- dat1 %>%

  filter(!is.na(yi), !is.na(vi), !is.na(interaction_duration)) %>%
```

```
  mutate(interaction_duration = droplevels(factor(interaction_duration)))


 mod.interaction_durationq <- rma(yi, vi, mods = ~ interaction_duration, data =
dat1_idur)
 mod.interaction_durationq



 Qinteraction_duration_df <- paste0(mod.interaction_durationq$QMdf, collapse = ", ")
 Qinteraction_durationQ  <- paste0(
  "Qb(", Qinteraction_duration_df, ") = ",
  round(mod.interaction_durationq$QM, 2),
  ", p = ",
  round(mod.interaction_durationq$QMp, 3)
 )

 cat(Qinteraction_durationQ, "\n")
 write.table(
  data.frame(Text = Qinteraction_durationQ),
  file = "Qinteraction_durationQ_updated.txt",
  row.names = FALSE, col.names = FALSE, quote = FALSE
 )



 mod.interaction_duration <- rma(yi, vi, mods = ~ interaction_duration - 1, data =
dat1_idur)
 mod.interaction_duration



 interaction_durationSumParticipants <- dat1_idur %>%
```

```
  group_by(interaction_duration) %>%
  summarise(
    nexp  = sum(intn, na.rm = TRUE),
    nctrl = sum(cn,   na.rm = TRUE),
    .groups = "drop"
  )

interaction_durationNumComp <- dat1_idur %>%
  count(interaction_duration, name = "kcomparisons")



mod_tab <- coef(summary(mod.interaction_duration)) %>%
  as.data.frame() %>%
  rownames_to_column("term") %>%
  mutate(
    interaction_duration = gsub("^interaction_duration", "", term),
    interaction_duration = gsub("^factor\\(interaction_duration\\)", "",
interaction_duration)
  ) %>%
  select(-term)

interaction_duration_type_table.final <- interaction_durationSumParticipants %>%
  left_join(interaction_durationNumComp, by = "interaction_duration") %>%
  left_join(mod_tab, by = "interaction_duration")

write.csv(interaction_duration_type_table.final," /3.
Runde/Mod.interaction_durationResult_updated.csv",row.names = FALSE)
```

```
###### ICAP Level avg ####


mod.ICAP_level_avgq <- rma(yi, vi, mods = ~ factor(ICAP_level_avg), data=dat1)

mod.ICAP_level_avgq


QICAP_level_avg_collapsed_string <- paste(mod.ICAP_level_avgq[["QMdf"]])

QICAP_level_avg_type1 <- data.frame(CollapsedQMdf =
QICAP_level_avg_collapsed_string)

QICAP_level_avg_type2 <- round(mod.ICAP_level_avgq$QM,2)

QICAP_level_avg_type3 <- round(mod.ICAP_level_avgq$QMp,3)


QICAP_level_avgQ <- paste(

 "Qb(",QICAP_level_avg_collapsed_string,") =",

 QICAP_level_avg_type2,

 ", p =",

 QICAP_level_avg_type3,

 collapse = " "

)


cat(QICAP_level_avgQ, "\n")

ICAP_level_avgQtest <- data.frame(Text = QICAP_level_avgQ)

write.table(ICAP_level_avgQtest, file = " QICAP_level_avgQ_updated.txt", row.names =
FALSE, col.names = FALSE, quote = FALSE)


mod.ICAP_level_avg <- rma(yi, vi, mods = ~ factor(ICAP_level_avg)-1, data=dat1)

mod.ICAP_level_avg


mod.ICAP_level_avg_table <-coef(summary(mod.ICAP_level_avg))
```

```
ICAP_level_avgSumParticipants <- dat1 %>%
  group_by(ICAP_level_avg) %>%
  summarise(nexp = sum(intn, na.rm = TRUE),
            nctrl = sum(cn, na.rm = TRUE))
ICAP_level_avgNumComp <- dat1 %>%
  count(ICAP_level_avg)
ICAP_level_avgNumComp <- rename(ICAP_level_avgNumComp, kcomparisons = n)
ICAP_level_avg_type_table.final<-
cbind(ICAP_level_avgSumParticipants,ICAP_level_avgNumComp[c(2)],
mod.ICAP_level_avg_table)
write.csv(ICAP_level_avg_type_table.final, " /3.
Runde/Mod.ICAP_level_avgResult_updated.csv")



###### ICAP_level_any ####


mod.ICAP_level_anyq <- rma(yi, vi, mods = ~ factor(ICAP_level_any), data=dat1)
mod.ICAP_level_anyq


QICAP_level_any_collapsed_string <- paste(mod.ICAP_level_anyq[["QMdf"]])
QICAP_level_any_type1 <- data.frame(CollapsedQMdf =
QICAP_level_any_collapsed_string)
QICAP_level_any_type2 <- round(mod.ICAP_level_anyq$QM,2)
QICAP_level_any_type3 <- round(mod.ICAP_level_anyq$QMp,3)


QICAP_level_anyQ <- paste(
  "Qb(",QICAP_level_any_collapsed_string,") =",
  QICAP_level_any_type2,
  ", p =",
  QICAP_level_any_type3,
```

```
  collapse = " "
 )


 cat(QICAP_level_anyQ, "\n")
 ICAP_level_anyQtest <- data.frame(Text = QICAP_level_anyQ)
 write.table(ICAP_level_anyQtest, file = " /3. Runde/QICAP_level_anyQ_oLin.txt",
row.names = FALSE, col.names = FALSE, quote = FALSE)


 mod.ICAP_level_any <- rma(yi, vi, mods = ~ factor(ICAP_level_any)-1, data=dat1)
 mod.ICAP_level_any


 mod.ICAP_level_any_table <-coef(summary(mod.ICAP_level_any))


 ICAP_level_anySumParticipants <- dat1 %>%
  group_by(ICAP_level_any) %>%
  summarise(nexp = sum(intn, na.rm = TRUE),
         nctrl = sum(cn, na.rm = TRUE))
 ICAP_level_anyNumComp <- dat1 %>%
  count(ICAP_level_any)
 ICAP_level_anyNumComp <- rename(ICAP_level_anyNumComp, kcomparisons = n)
 ICAP_level_any_type_table.final<-
cbind(ICAP_level_anySumParticipants,ICAP_level_anyNumComp[c(2)],
mod.ICAP_level_any_table)
 write.csv(ICAP_level_any_type_table.final, " /3.
Runde/Mod.ICAP_level_avgResult_oLin.csv")


 ##### ICAP level AuS ######


 mod.ICAP_level_AuSq <- rma(yi, vi, mods = ~ factor(ICAP_level_AuS), data=dat1)
 mod.ICAP_level_AuSq
```

```
 QICAP_level_AuS_collapsed_string <- paste(mod.ICAP_level_AuSq[["QMdf"]])
 QICAP_level_AuS_type1 <- data.frame(CollapsedQMdf =
QICAP_level_AuS_collapsed_string)
 QICAP_level_AuS_type2 <- round(mod.ICAP_level_AuSq$QM,2)
 QICAP_level_AuS_type3 <- round(mod.ICAP_level_AuSq$QMp,3)

 QICAP_level_AuSQ <- paste(
  "Qb(",QICAP_level_AuS_collapsed_string,") =",
  QICAP_level_AuS_type2,
  ", p =",
  QICAP_level_AuS_type3,
  collapse = " "
 )

 cat(QICAP_level_AuSQ, "\n")
 ICAP_level_AuSQtest <- data.frame(Text = QICAP_level_AuSQ)
 write.table(ICAP_level_AuSQtest, file = " /3. Runde/QICAP_level_AuSQ_updated.txt",
row.names = FALSE, col.names = FALSE, quote = FALSE)


 mod.ICAP_level_AuS <- rma(yi, vi, mods = ~ factor(ICAP_level_AuS)-1, data=dat1)
 mod.ICAP_level_AuS


 mod.ICAP_level_AuS_table <-coef(summary(mod.ICAP_level_AuS))
 ICAP_level_AuSSumParticipants <- dat1 %>%
  group_by(ICAP_level_AuS) %>%
```

```
  summarise(nexp = sum(intn, na.rm = TRUE),
            nctrl = sum(cn, na.rm = TRUE))
 ICAP_level_AuSNumComp <- dat1 %>%
  count(ICAP_level_AuS)
 ICAP_level_AuSNumComp <- rename(ICAP_level_AuSNumComp, kcomparisons = n)
 ICAP_level_AuS_type_table.final<-
cbind(ICAP_level_AuSSumParticipants,ICAP_level_AuSNumComp[c(2)],
mod.ICAP_level_AuS_table)
 write.csv(ICAP_level_AuS_type_table.final, " /3.
Runde/Mod.ICAP_level_AuSResult2_updated.csv",sep = ";")



############################## Meta-Regression ########################

library(readxl)
library(dplyr)
library(metafor)



dat <- read_excel("MA_GAISTEM_ModAnalysis_Reg_CNA.xlsx",
                  sheet = "Tabelle1")


names(dat1)



dat1 <- dat1 %>%
 mutate(
  OUT = factor(learning_outcome),
  #DUR = factor(DUR),
```

```
   ICAP = factor(ICAP_level_AuS)
  )

dat_mod <- dat1 %>%
  filter(!is.na(OUT),
         OUT != "",
         !is.na(ICAP))

table(dat_mod$OUT)
table(dat_mod$ICAP)



res_out_icap_main <- rma(
  yi   = yi,
  vi   = vi,
  mods = ~ OUT + ICAP,
  data = dat_mod,
  method = "REML"
)

summary(res_out_icap_main)



res_out_icap_int <- rma(
  yi   = yi,
  vi   = vi,
  mods = ~ OUT * ICAP,   # = OUT + ICAP + OUT:ICAP
  data = dat_mod,
  method = "REML"
```

```
)

summary(res_out_icap_int)

anova(res_out_icap_main, res_out_icap_int, refit=TRUE)

############################ CNA ##############################

library(readxl)
library(dplyr)
library(cna)

#dat <- read_excel("MA_GAISTEM_ModAnalysis_Reg_CNA.xlsx",
#                  sheet = "Tabelle1")
dat<-dat1

nrow(dat)

names(dat)

unique(dat$OUT)
unique(dat$ICAP)

dat_cna <- dat %>%
  mutate(
    E = case_when(
      yi < 0.1  ~ 0L,
      yi <= 0.6 ~ 1L,
      yi > 0.6  ~ 2L,
```

```
  TRUE     ~ NA_integer_
 ),

 OUT_clean  = trimws(as.character(OUT)),
 #DUR_clean  = trimws(as.character(DUR)),
 ICAP_clean = trimws(as.character(ICAP)),

 O = case_when(
  OUT_clean == "K" ~ 0L,
  OUT_clean == "S" ~ 1L,
  TRUE          ~ NA_integer_
 ),

 #D = case_when(
 #  grepl("^<", DUR_clean) ~ 0L,
 #  grepl("^>", DUR_clean) ~ 1L,
 #  TRUE               ~ NA_integer_
 #),

 I = case_when(
  ICAP_clean == "0"  ~ 0L,
  ICAP_clean == "1A" ~ 1L,
  ICAP_clean == "1S" ~ 2L,
  TRUE           ~ NA_integer_
 )
) %>%

select(O, E, I) %>% #, D)
```

```
 filter(!is.na(O), !is.na(I), !is.na(E)) #!is.na(D),

sum(!is.na(dat$yi))



dim(dat_cna)
str(dat_cna)
table(dat_cna$O, dat_cna$E) #, dat_cna$D



cna_dat_cs <- dat_cna %>%
 mutate(
  E = if_else(E == 2L, 1L, 0L)
 ) %>%
 as.data.frame()

dim(cna_dat_cs)
table(cna_dat_cs$O, cna_dat_cs$E) #cna_dat_cs$D,

ct <- configTable(cna_dat_cs, type = "mv")
ct

res_cna_cs <- cna(ct,
            outcome = "E=1",
            con = 0.8,
            cov = 0.8)

res_cna_cs
```

```
dat_nec <- cna_dat_cs %>%
 mutate(
   K = if_else(O == 0L, 1L, 0L)
 )



table(dat_nec$K, dat_nec$E)


E1 <- dat_nec$E == 1
K1 <- dat_nec$K == 1


cons_nec <- sum(E1 & K1) / sum(E1)



cov_nec  <- sum(E1 & K1) / sum(K1)


cons_nec
cov_nec



# ── ICAP-Notwendigkeitsanalyse ──────────────────────────────────────────


# Überblickstabelle
table(cna_dat_cs$I, cna_dat_cs$E)


# Referenzvektor: große Effekte
E1 <- cna_dat_cs$E == 1L


# Konsistenz & Coverage für jedes ICAP-Level
```

```
icap_nec <- lapply(list(
  list(lv = 0L, label = "kein (0)"),
  list(lv = 1L, label = "1A"),
  list(lv = 2L, label = "1S")
), function(x) {
  cond <- cna_dat_cs$I == x$lv
  data.frame(
    ICAP_Level  = x$label,
    n_gesamt    = sum(cond),
    n_E1        = sum(E1 & cond),
    cons_nec    = round(sum(E1 & cond) / sum(E1),   4),
    cov_nec     = round(sum(E1 & cond) / sum(cond), 4)
  )
}) %>% bind_rows()

print(icap_nec)

# ── Zusatz: "Irgendeines ICAP-Levels" (I > 0) als Bedingung ──────────────
any_icap <- cna_dat_cs$I > 0L
cons_any <- sum(E1 & any_icap) / sum(E1)
cov_any  <- sum(E1 & any_icap) / sum(any_icap)

cat("\nI > 0 (beliebiges ICAP-Level):\n")
cat(sprintf("  cons_nec = %.4f\n  cov_nec  = %.4f\n", cons_any, cov_any))

# Fit intercept-only model for reference line
res <- rma(yi, vi, data = dat, method = "REML")

# Funnel plot
```

```
funnel(res,
     xlab  = "Hedge's g",
     ylab  = "Standard Error",
     main  = "Funnel Plot",
     level = c(90, 95, 99),   # pseudo-confidence contours
     shade = c("white", "gray75", "gray60"),
     legend = TRUE)

# Egger's test for funnel asymmetry
regtest(res)

# ── 1. Trim-and-Fill ──────────────────────────────────────────────
taf <- trimfill(res)
summary(taf)

# Funnel plot with imputed studies shown as open circles
funnel(taf,
     xlab   = "Effect Size (yi)",
     main   = "Funnel Plot: Trim-and-Fill",
     legend = TRUE)

# ── 2. Rank Correlation Test (Begg's Test) ────────────────────────
ranktest(res)

# ── 3. Rosenthal's Fail-Safe N ────────────────────────────────────
fsn(yi, vi, data = dat, type = "Rosenthal")
```

```
#RoBMA 4.0
library(rjags)
library(RoBMA)
library(metafor)
library(readxl)
library(ggplot2)
library(dplyr)
library(tibble)
library(janitor)

xlsx_file   <- "
MA_GAISTEM_ModAnalysis_Reduced_CBNP_last2_KvsS_20260525_red_res2.xlsx"

dat_raw <- read_excel(xlsx_file)

# Spaltennamen bereinigen
dat <- dat_raw %>%
 clean_names() %>%
 rename(study = study,
       yi    = yi,
       vi    = vi,
       cn    = cn) %>%
 mutate(
   # Standardfehler aus Varianz berechnen (für RoBMA)
   sei    = sqrt(vi),

   # duration: non-breaking spaces bereinigen, dann faktorisieren
   duration = gsub("\u00a0", " ", duration),
```

```
  duration = trimws(duration),

  # Alle kategorialen Moderatoren als Faktor
  domain    = as.factor(domain),
  ed_level  = as.factor(ed_level),
  system    = as.factor(system),
  adapted   = as.factor(adapted),
  outcome   = as.factor(outcome),
  type      = as.factor(type),
  thinking  = as.factor(thinking),
  duration  = as.factor(duration),
  isar_level = factor(isar_level,
                      levels = c("0", "1A", "1S"),
                      labels = c("R", "A", "S"))
 )

# Datensatz-Überblick
cat("=== Datensatz-Überblick ===\n")
cat("k  =", nrow(dat), "Studien\n")
cat("yi : M =", round(mean(dat$yi),  3),
  "| SD =", round(sd(dat$yi), 3),
  "| Range [", round(min(dat$yi), 3), ",", round(max(dat$yi), 3), "]\n")
cat("vi : M =", round(mean(dat$vi),  4),
  "| Range [", round(min(dat$vi), 4), ",", round(max(dat$vi), 4), "]\n")
cat("sei: M =", round(mean(dat$sei), 3),
  "| Range [", round(min(dat$sei), 3), ",", round(max(dat$sei), 3), "]\n\n")

cat("=== Moderatoren ===\n")
cat("domain:    ", table(dat$domain),    "\n")
```

```
cat("ed_level:   ", table(dat$ed_level, useNA = "ifany"), "\n")
cat("system:     ", table(dat$system),     "\n")
cat("adapted:    ", table(dat$adapted),    "\n")
cat("outcome:    ", table(dat$outcome),    "\n")
cat("type:       ", table(dat$type),       "\n")
cat("thinking:   ", table(dat$thinking),   "\n")
cat("duration:   ", table(dat$duration, useNA = "ifany"), "\n")
cat("isar_level: ", table(dat$isar_level), "\n\n")

# Fehlende Werte prüfen
cat("=== Fehlende Werte ===\n")
print(colSums(is.na(dat)))



################## 2. Klassische REML Meta-Analyse als Vergleich

fit_re <- rma(
  yi   = yi,
  vi   = vi,
  data = dat,
  method = "REML",
  slab   = study
)

pi_re <- predict(fit_re)

cat("\n=== REFERENZ: Klassische RE-Meta-Analyse ===\n")
cat(sprintf("k              = %d\n",          fit_re$k))
```

```
cat(sprintf("Gesamteffekt g    = %.3f (SE = %.3f)\n", fit_re$beta, fit_re$se))
cat(sprintf("95%% KI           = [%.3f, %.3f]\n", fit_re$ci.lb,  fit_re$ci.ub))
cat(sprintf("95%% Prädikt.-PI  = [%.3f, %.3f]\n", pi_re$pi.lb,   pi_re$pi.ub))
cat(sprintf("I²               = %.2f%%\n",       fit_re$I2))
cat(sprintf("τ  = %.4f,  τ²   = %.4f\n",         sqrt(fit_re$tau2), fit_re$tau2))
cat(sprintf("Q(%d) = %.2f, p = %.4f\n",
        fit_re$k - 1, fit_re$QE, fit_re$QEp))

# Referenz-Forest-Plot
forest(fit_re,
     main   = "Forest Plot – GAISTEM (RE-Modell, Referenz)",
     xlab   = "Hedges' g",
     header = "Studie",
     col    = "darkblue")


###################################
##################### 3. Priors definieren
###################################


# ---- 3a. Effekt-Prior ----
prior_mu <- prior(
  distribution = "cauchy",
  parameters   = list(location = 0, scale = 0.707)
)

# ---- 3b. Heterogenitäts-Prior (an I² > 96% angepasst) ----
prior_tau <- prior(
```

```
 distribution = "normal",
 parameters   = list(mean = 0, sd = 0.5),
 truncation   = list(lower = 0, upper = Inf)
)

# ---- 3c. Publication-Bias-Modelle ----



# Diagnostik
robma_fns <- ls("package:RoBMA")
pb_fns    <- robma_fns[grepl("weight|PET|PEESE|bias", robma_fns, ignore.case = TRUE)]
cat("\nVerfügbare RoBMA PB-Funktionen:\n")
cat(paste(" •", pb_fns, collapse = "\n"), "\n")



##############################################
############# 4. RoBMA Analyse #################
##############################################

cat("k =", nrow(dat), "| PSMA-Ensemble | 4 × 10.000 Posterior-Samples\n\n")

fit_robma <- RoBMA(
 yi  = dat$yi,
 vi  = dat$vi,

 measure           = "SMD",
 model_type        = "PSMA",
 prior_effect      = prior_mu,
 prior_heterogeneity = prior_tau,
```

```
  chains   = 4,
  sample   = 10000,
  burnin   = 2000,
  thin     = 1,
  parallel = TRUE,
  seed     = 42
)
##############################################
######################### 5. Konvergenz ##############
##############################################

# ---- 5a. Trace-Plots ----
par(mfrow = c(1, 2))
plot(fit_robma, type = "trace", parameter = "mu",
    main = "Trace: μ (Gesamteffekt)")
plot(fit_robma, type = "trace", parameter = "tau",
    main = "Trace: τ (Heterogenität)")
par(mfrow = c(1, 1))

# ---- 5b. Autokorrelations-Plots ----
par(mfrow = c(1, 2))
plot(fit_robma, type = "autocorrelation", parameter = "mu",
    main = "Autokorrelation: μ")
plot(fit_robma, type = "autocorrelation", parameter = "tau",
    main = "Autokorrelation: τ")
par(mfrow = c(1, 1))

# ---- 5c. Numerische Konvergenz (R-hat, ESS) ----
```

```
cat("\nNumerische Konvergenzkennwerte (aus summary):\n")
conv_summary <- summary(fit_robma)
print(conv_summary)



################################################################################
#### 
################### 6. Ergebnisse
##############################################
################################################################################

cat("\n\n=== Hauptergebnisse RoBMA ===\n")
cat(strrep("=", 65), "\n")

print(summary(fit_robma))

cat("\n--- Bayesianische Evidenz je Modell-Komponente ---\n")
comp_summary <- summary(fit_robma, type = "components")
print(comp_summary)

cat("\n--- Posterior-Modellgewichte ---\n")
print(summary(fit_robma, type = "models"))

post <- summary(fit_robma)$estimates

cat("\n--- Posterior-Schätzer ---\n")
cat(sprintf("μ (Gesamteffekt) : %.3f  [%.3f, %.3f]  95%% KrI\n",
        post["mu",  "Mean"], post["mu",  "0.025"], post["mu",  "0.975"]))
cat(sprintf("τ (Heterogenität): %.3f  [%.3f, %.3f]  95%% KrI\n",
```

```
      post["tau", "Mean"], post["tau", "0.025"], post["tau", "0.975"]))


########################################################################
############# 7. Plots  ###################################
########################################################################

plot(fit_robma, type = "forest",
   main = "RoBMA Forest Plot – GAISTEM (BMA)")

plot(fit_robma, type = "posterior", parameter = "mu",
   main = "Posterior: Gesamteffekt μ")

plot(fit_robma, type = "posterior", parameter = "tau",
   main = "Posterior: Heterogenität τ")

plot(fit_robma, type = "weightfunction",
   main = "Modell-gemittelte Selektionsfunktion (Publication Bias)")

plot(fit_robma, type = "prior_vs_posterior", parameter = "mu", prior = TRUE,
   main = "Prior vs. Posterior: Effektgröße μ", xlim = c(-1.5, 1.5))

############################################################
############# 8. Sensitivitätsanalyse ######################
############################################################

# S1: Diffuserer τ-Prior – HN(0, 1.0) statt HN(0, 0.5)
fit_s1 <- RoBMA(
  yi = dat$yi, vi = dat$vi,
```

```
  measure           = "SMD",
  prior_heterogeneity = prior("normal", list(mean = 0, sd = 1.0),
                              truncation = list(lower = 0, upper = Inf)),
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 101
)

# S2: Normalverteilungs-Prior für μ
fit_s2 <- RoBMA(
  yi = dat$yi, vi = dat$vi,
  measure           = "SMD",
  model_type          = "PSMA",
  prior_effect      = prior("normal", list(mean = 0, sd = 0.5)),
  prior_heterogeneity = prior("normal", list(mean = 0, sd = 0.5),
                              truncation = list(lower = 0, upper = Inf)),
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 202
)

# S3: Nur PET-PEESE (ohne Gewichtsfunktionen)
fit_s3 <- RoBMA(
  yi = dat$yi, vi = dat$vi,
  measure           = "SMD",
  model_type          = "PP",
  prior_effect      = prior("cauchy", list(location = 0, scale = 0.707)),
  prior_heterogeneity = prior("normal", list(mean = 0, sd = 0.5),
                              truncation = list(lower = 0, upper = Inf)),
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 303
```

```
)

get_estimates <- function(fit, label) {
  p <- summary(fit)$estimates
  data.frame(
    Modell   = label,
    mu_mean  = round(p["mu",  "Mean"],  3),
    mu_lb    = round(p["mu",  "0.025"], 3),
    mu_ub    = round(p["mu",  "0.975"], 3),
    tau_mean = round(p["tau", "Mean"],  3),
    tau_lb   = round(p["tau", "0.025"], 3),
    tau_ub   = round(p["tau", "0.975"], 3)
  )
}

sens_table <- bind_rows(
  data.frame(
    Modell   = "RE klassisch (Referenz)",
    mu_mean  = round(fit_re$beta,      3),
    mu_lb    = round(fit_re$ci.lb,     3),
    mu_ub    = round(fit_re$ci.ub,     3),
    tau_mean = round(sqrt(fit_re$tau2), 3),
    tau_lb   = NA_real_, tau_ub = NA_real_
  ),
  get_estimates(fit_robma, "RoBMA PSMA: HN(0.5) + Cauchy(0.707) [Haupt]"),
  get_estimates(fit_s1,    "RoBMA PSMA: HN(1.0) + Cauchy(0.707) [S1]"),
  get_estimates(fit_s2,    "RoBMA PSMA: HN(0.5) + Normal(0,0.5) [S2]"),
  get_estimates(fit_s3,    "RoBMA PP:   HN(0.5) + Cauchy(0.707) [S3]")
)
```

```
cat("\n--- Sensitivitäts-Vergleich ---\n")
print(sens_table, row.names = FALSE)


################################################################
########################## 9. Moderatorenanalyse RoBMA  ##########
################################################################

# ---- 91. Learning outcome (K vs. S) ----
fit_mod_outcome <- RoBMA(
  yi = dat$yi, vi = dat$vi,
  measure          = "SMD",
  model_type       = "PSMA",
  prior_effect     = prior_mu,
  prior_heterogeneity = prior_tau,
  mods             = ~ outcome,
  data             = dat,
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 42
)
cat("\n=== Moderator: outcome (K vs. S) ===\n")
print(summary(fit_mod_outcome))
print(summary(fit_mod_outcome, type = "components"))


# ---- 9b. ISAR Level (R vs. A vs S) ----

fit_mod_isar <- RoBMA(
```

```
  yi = dat$yi, vi = dat$vi,
  measure          = "SMD",
  model_type       = "PSMA",
  prior_effect     = prior_mu,
  prior_heterogeneity = prior_tau,
  mods             = ~ isar_level,   # Faktor direkt, 2 Kontraste
  data             = dat,
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 45
)
cat("\n=== Moderator: isar_level (0 / 1A / 1S) ===\n")
print(summary(fit_mod_isar))
print(summary(fit_mod_isar, type = "components"))

# ---- 9c. Thinking skills ----
# thinking (LOT vs. HOT):
fit_mod_thinking <- RoBMA(
  yi = dat$yi, vi = dat$vi,
  measure = "SMD",
  model_type = "PSMA",
  prior_effect = prior_mu,
  prior_heterogeneity = prior_tau,
  mods = ~ thinking,
  data = dat,
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 43
)
cat("\n=== Moderator: GAI system (LOT=0 vs. HOT=1) ===\n")
print(summary(fit_mod_thinking))
```

```
print(summary(fit_mod_thinking, type = "components"))


# ---- 9d. GAI system ----
# system (GPT vs. OTH):
fit_mod_system <- RoBMA(
  yi = dat$yi, vi = dat$vi,
  measure = "SMD",
  model_type = "PSMA",
  prior_effect = prior_mu,
  prior_heterogeneity = prior_tau,
  mods = ~ system,
  data = dat,
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 44
)
cat("\n=== Moderator: GAI system (OTH=0 vs. GPT=1) ===\n")
print(summary(fit_mod_system))
print(summary(fit_mod_system, type = "components"))



# ---- 9e. domain (OTH=0 vs. IT=1) ----

fit_mod_domain <- RoBMA(
  yi = dat$yi, vi = dat$vi,
  measure          = "SMD",
  model_type       = "PSMA",
  prior_effect     = prior_mu,
  prior_heterogeneity = prior_tau,
  mods             = ~ domain,
```

```
  data             = dat,
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 50
)
cat("\n=== Moderator: domain (OTH=0 vs. IT=1) ===\n")
print(summary(fit_mod_domain))
print(summary(fit_mod_domain, type = "components"))



# ---- 9f. ed_level (K12=0 vs. Uni=1) ----
#  1 Studie entfernt (n=58)

fit_mod_edlevel <- RoBMA(
  yi = dat$yi, vi = dat$vi,
  measure          = "SMD",
  model_type       = "PSMA",
  prior_effect     = prior_mu,
  prior_heterogeneity = prior_tau,
  mods             = ~ ed_level,
  data             = dat,
  subset           = !is.na(dat$ed_level),   # 1 NA ausschließen
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 51
)
cat("\n=== Moderator: ed_level (K12=0 vs. Uni=1, k=58) ===\n")
print(summary(fit_mod_edlevel))
print(summary(fit_mod_edlevel, type = "components"))
```

```
# ---- 9g. GAI system (OTH=0 vs. GPT=1) ----

fit_mod_system <- RoBMA(
  yi = dat$yi, vi = dat$vi,
  measure          = "SMD",
  model_type       = "PSMA",
  prior_effect     = prior_mu,
  prior_heterogeneity = prior_tau,
  mods             = ~ system,
  data             = dat,
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 52
)
cat("\n=== Moderator: system (OTH=0 vs. GPT=1) ===\n")
print(summary(fit_mod_system))
print(summary(fit_mod_system, type = "components"))



# ---- 9h. adapted system (nicht-adaptiert=0 vs. adaptiert=1) ----

fit_mod_adapted <- RoBMA(
  yi = dat$yi, vi = dat$vi,
  measure          = "SMD",
  model_type       = "PSMA",
  prior_effect     = prior_mu,
  prior_heterogeneity = prior_tau,
  mods             = ~ adapted,
  data             = dat,
  chains = 4, sample = 10000, burnin = 2000,
```

```
  parallel = TRUE, seed = 53
)
cat("\n=== Moderator: adapted (0=nicht, 1=adaptiert) ===\n")
print(summary(fit_mod_adapted))
print(summary(fit_mod_adapted, type = "components"))


# ---- 9i. GAI type (HL=0 vs. HH=1) ----

fit_mod_type <- RoBMA(
  yi = dat$yi, vi = dat$vi,
  measure          = "SMD",
  model_type       = "PSMA",
  prior_effect     = prior_mu,
  prior_heterogeneity = prior_tau,
  mods             = ~ type,
  data             = dat,
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 54
)
cat("\n=== Moderator: type (HL=0 vs. HH=1) ===\n")
print(summary(fit_mod_type))
print(summary(fit_mod_type, type = "components"))


# ---- 9j. thinking (LOT=0 vs. HOT=1) ----

fit_mod_thinking <- RoBMA(
  yi = dat$yi, vi = dat$vi,
```

```
  measure          = "SMD",
  model_type       = "PSMA",
  prior_effect     = prior_mu,
  prior_heterogeneity = prior_tau,
  mods             = ~ thinking,
  data             = dat,
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 55
)
cat("\n=== Moderator: thinking (LOT=0 vs. HOT=1) ===\n")
print(summary(fit_mod_thinking))
print(summary(fit_mod_thinking, type = "components"))


# ---- 9k. interaction duration (< 4 w=0 vs. > 4 w=1) ----
# 11 Studien entfernt (k=48)
# Ergebnis mit Vorsicht interpretieren

fit_mod_duration <- RoBMA(
  yi = dat$yi, vi = dat$vi,
  measure          = "SMD",
  model_type       = "PSMA",
  prior_effect     = prior_mu,
  prior_heterogeneity = prior_tau,
  mods             = ~ duration,
  data             = dat,
  subset           = !is.na(dat$duration),
  chains = 4, sample = 10000, burnin = 2000,
  parallel = TRUE, seed = 56
```

```
)
cat("\n=== Moderator: duration (< 4w=0 vs. > 4w=1, k=48) ===\n")
print(summary(fit_mod_duration))
print(summary(fit_mod_duration, type = "components"))



#################################################################
######### 10. Plotting OUTCOME and ISAR Moderator ###############
#################################################################

# marginal means
mm_outcome <- marginal_means(fit_mod_outcome)
mm_isar    <- marginal_means(fit_mod_isar)

summary(mm_outcome)
summary(mm_isar)

# ---- Learning outcome ----

# RStudio preview
par(mar = c(4, 4, 3, 4))
plot(mm_outcome, parameter = "outcome", lwd = 2,
    xlim = c(-2, 2), xlab = "Hedges' g",
    main = "Model-averaged marginal means by learning outcome")

# PNG export
png("RoBMA_outcome.png", width = 4.5, height = 4.0, units = "in", res = 300)
par(mar = c(4, 4, 3, 4))                  # must repeat inside device
plot(mm_outcome, parameter = "outcome", lwd = 2,
```

```
    xlim = c(-2, 2), xlab = "Hedges' g",
    main = "Model-averaged marginal means by learning outcome")
dev.off()

# ---- ISAR level ----

# RStudio preview
par(mar = c(4, 4, 3, 4))
plot(mm_isar, parameter = "isar_level", lwd = 2,
    xlim = c(-2, 2), xlab = "Hedges' g",
    main = "Model-averaged marginal means by ISAR level")

# PNG export
png("RoBMA_isar.png", width = 4.5, height = 4.0, units = "in", res = 300)
par(mar = c(4, 4, 3, 4))                # must repeat inside device
plot(mm_isar, parameter = "isar_level", lwd = 2,
    xlim = c(-2, 2), xlab = "Hedges' g",
    main = "Model-averaged marginal means by ISAR level")
dev.off()

# ---- Combined ----

# RStudio preview
par(mfrow = c(1, 2), mar = c(4, 4, 3, 4))
plot(mm_outcome, parameter = "outcome", lwd = 2,
    xlim = c(-2, 2), xlab = "Hedges' g", main = "")
mtext("a) Learning outcome", side = 3, line = 1, adj = 0, font = 2)
plot(mm_isar, parameter = "isar_level", lwd = 2,
    xlim = c(-2, 2), xlab = "Hedges' g", main = "")
```

```
mtext("b) ISAR level", side = 3, line = 1, adj = 0, font = 2)
par(mfrow = c(1, 1), mar = c(5, 4, 4, 2) + 0.1)  # reset after combined

# PNG export
png("RoBMA_combined_ab.png", width = 6.85, height = 3.4, units = "in", res = 300)
par(mfrow = c(1, 2), mar = c(4, 4, 3, 4))        # must repeat inside device
plot(mm_outcome, parameter = "outcome", lwd = 2,
    xlim = c(-2, 2), xlab = "Hedges' g", main = "")
mtext("a) Learning outcome", side = 3, line = 1, adj = 0, font = 2)
plot(mm_isar, parameter = "isar_level", lwd = 2,
    xlim = c(-2, 2), xlab = "Hedges' g", main = "")
mtext("b) ISAR level", side = 3, line = 1, adj = 0, font = 2)
dev.off()
par(mfrow = c(1, 1), mar = c(5, 4, 4, 2) + 0.1)  # reset after combined
```