Appendix A: Data and Scripts

related to "Zero-shot performance of selected large language and multimodal models on the

2023 Brazilian Portuguese medical residency exam"

César Augusto Madid Truyts[1,2], Amanda Gomes Rabelo[1,2], Gabriel Mesquita de Souza[1], Daniel Scaldaferri Lages[1], Adriano José Pereira[1,2], Uri Adrian Prync Flato[2], Eduardo Pontes dos Reis[1,3], Joaquim Edson Vieira[4,6], Paulo Sergio Panse Silveira[5], and Edson Amaro Junior[1]

[1]Einstein Global Advanced Technologies for Equity, Hospital Israelita Albert Einstein
[2]Departamento de Pacientes Graves, Hospital Israelita Albert Einstein
[3]Stanford Center for Artificial Intelligence in Medicine and Imaging, Stanford University
[4]Anestesiologia, Departmento de Cirurgia, Faculdade de Medicina da Universidade de São Paulo, SP, Brasil
[5]Informática Médica, Departamento de Patologia, Faculdade de Medicina da Universidade de São Paulo
[6]Faculdade Israelita de Ciências da Saúde Albert Einstein, SP, Brasil

Contact (supplementary material): Paulo S. P. Silveira (silveira@usp.br)
Avaliable at Harvard Dataverse - https://doi.org/10.7910/DVN/OLKIL3

October 15, 2025

Contents

Scripts and data **1**

About this appendix . . . 1

How to Use This Document . . . 1

Data . . . 2

Raw data . . . 2

Processed data . . . 3

Prompt Design . . . 4

R scripts . . . 6

Verifying the Correct Answers (R) . . . 6

Mean scores . . . 8

Descriptive and repeated measures One-way ANOVA . . . 12

Distribution of Human Responses vs. Artificial Intelligences . . . 21

Radar graph . . . 25

Agreement between observers . . . 28

Python scripts . . . 33

Verifying the Correct Answers (Python) . . . 33

Descriptive One-way ANOVA . . . 34

Scripts and data

About this appendix

Here you will find the raw data and reorganized data used to produce the tables and graphs in the corresponding article. The raw data consist of the responses provided by each artificial intelligence (AI) model to each of the questions (see Appendix 'HCFMUSP Medical Residency Exam Questions') in each executed trial. Models of Large Language Models (LLM) and Multimodal Models (MMM) were tested.

The reorganized data include measures of central tendency or location that were presented in the tables or used in statistical analyses and graph construction. The scripts are presented in the order we suggest executing them to reproduce the results presented in the article.

To understand the context of the data and the R and Python routines presented below, it is necessary to refer to the article "Zero-shot performance of selected large language and multimodal models on the 2023 Brazilian Portuguese medical residency examPerformance of Generative AI in Brazilian Portuguese Medical Exam" by Truyts et al., 2025.

How to Use This Document

If you wish to use this PDF and access its local links, download all files into a single folder and open it with a reader capable of opening local files. In some readers, you might encounter a message like `\Unable to open external link:  Operation not supported,"` which is common in browser-based PDF viewers (such as Chrome, Firefox, or Evince on Linux), as they block local file links for security reasons. If this is the case, some dedicated PDF readers may be attempted:

- Adobe Acrobat Reader (Windows, macOS, Linux via Wine)
- Okular (Linux)
- Foxit Reader (Windows and macOS)
- PDF-XChange Editor (Windows)

Data

Raw data

The two most basic data sources are the responses provided by the AI models, available in `Exp1RawData.xlsx` and `Exp2RawData.xlsx`, and the official answer key, publicly provided by the residency exam organizers, available in `AnswerKey.xlsx`.

For comparison, `AnswerHuman.xlsx` contains the number of correct answers provided by each candidate. This is the only information that is publicly and mandatorily disclosed. Even in an anonymized form, data holders do not allow access to individual responses to each question. Therefore, the analysis of artificial intelligence models that do not handle questions containing images is limited.

`AccTimeToken.xlsx` data were directly extracted from the database and include the total number of correct answers (`Correct`), the already calculated accuracy (`Accuracy`), and the average processing time per question (`Time`) and number of tokens (`Tokens`) required by each model to process the exam. It contains two tabs: `LLM` corresponds to Experiment 1, showing the accuracy results for the 74 textual questions (without images) resolved in five trials by all ten models studied (Titan Text, LLhama-3-8b, Mixtral 8x7B, LLhama-3-70b, Command R +, GPT-4 Turbo, Claude-3-haiku, Claude-3-opus, Claude-3-sonnet, Claude-3.5-sonnet); `MMM` corresponds to Experiment 2, presenting the results for all 177 valid questions resolved in five trials by the four multimodal models (solving all questions) of the Claude family capable of handling image-based questions at the time of the study (Claude-3-haiku, Claude-3-opus, Claude-3-sonnet, Claude-3.5-sonnet).

`AnswerHuman.xlsx` has only two columns: `Student`, which contains a sequential number that does not reveal the candidate's identity, and `Correct`, the number of correct answers among the 117 valid questions.

`ObserverEval.xlsx` contains the evaluations made by three independent observers regarding the justifications provided by a single trial of Claude-3-Opus, with the following columns (some are redundant with other spreadsheets, and some were not used in the statistical analysis):

- **Number** ... question number.
- **Question** ... question label.
- **Area** ... major medical areas (Internal Medicine, Surgery, Public Health, Gynecology and Obstetrics, Pediatrics).
- **Answer.key** ... correct answer.
- **Model.answer** ... response from Claude-3-Opus.

- **Justificative.pt** ... justification provided by the model, in Portuguese (occasionally, the model gave the response in English).
- **Justificative.en** ... justification provided by the model, translated into English.
- **Cam01.CorrectAnswer** ... Layer 1, whether `Answer.key` matches `Model.answer` (Y/N).
- **Observer evaluations (labeled 01, 02, 03)**:
  - **InterpJust#** ... were the interpretation and justifications correct? (Y/N).
  - **Harm#** ... would there be potential harm to the patient? (Y/N).
- **Cam02.CorrectInterp** ... Layer 2, did the problem occur in the interpretation? (Y/N).
- **Cam03.Coherent** ... Layer 3, is the answer coherent with the interpretation (regardless of whether the answer was correct or not; only considering whether the justification supports the chosen alternative)? (Y/N).
- **Tokens.input** ... number of tokens for question input (not used).
- **Tokens.output** ... number of tokens for question output (not used).
- **Image.kind** ... type of image (textual, radiological, non-radiological).

Processed data

- `ScoresLLM.xlsx` and `ScoresMMM.xlsx`: correct answers per model and trial generated with R - see Verifying the Correct Answers.
- `ScoresMMMPy.xlsx`: worksheet with correct answers per model and trial generated with Python - see Verifying the Correct Answers.
- `MeansMMM.xlsx`: mean scores per model, trial, area, and question type (see Mean scores).
- `Descriptive.xlsx` contains mean scores per model, trial, area, and question type.
- `ANOVA.xlsx` contains the statistical results of repeated measure One-Way ANOVA and post-hoc with estimated marginal means (see Descriptive statistics and repeated measures One-way ANOVA). Another alternative using One-Way ANOVA and post-hoc pairwise comparison using Tukey's HSD is in Descriptive statistics and One-way ANOVA. The results obtained can be verified in `StatisticExperiment2.xlsx`.
- `Agreement.xlsx`: within and between agreement of observer evaluations (see Agreement between observers).

Prompt Design

These files constitute the reproducibility materials for the large language model (LLM) queries performed in the study. Here is an explanation of each type:

1. `GeneralPrompt.txt`

   This is the base prompt, that is, the fixed text used in all queries to ensure that all models received the same structured initial command. It contains the instructions defining the model's role (for example, "you are a physician answering residency exam questions") and the expected response format, with tags such as `<resp>`, `<just>`, `<diag>`, etc. It also defines the formatting of the prompt sent to all models.

2. `.ipynb` files

   These are Jupyter notebooks with the Python code used to:

   - load the questions (as text or images);
   - apply the prompt GeneralPrompt.txt;
   - executes the queries to the specific model (Claude 3, GPT, Llama 3, Mistral, etc.);
   - receive and save the responses.

   The filename indicates:

   - the model tested (`claude3`, `gpt`, `llama3`, `mixtral`, etc.);
   - the question type (`txt` = text, `img` = image OCR);
   - the dataset used (`usp-74-questoes` or `usp-117-questoes`).

   The set of *.ipynb files is available for download in `Prompts.zip`

GeneralPrompt.txt

```
Human:
Você é um médico que terá que responder a questão de múltipla escolha que está
dentro da tag <questao> de uma prova de concurso para residência médica e seu
público alvo serão outros médicos
<questao>
(Enunciado da questão)
</questao>
Aqui estão as instruções:
- Responda apenas a letra da resposta correta que considera a mais adequada
dentro da tag <resp> e depois escreva uma justificativa dentro da tag <just>.
- Descreva o diagnóstico dentro da tag <diag>.
- Explique a fisiopatologia dentro da tag <fisiop>.
- Descreva o tratamento dentro da tag <trat> para o diagnóstico
localizado na tag <diag>.
- Explique a farmacologia dentro da tag <farma>.
- Responda em portugûes.
Assistant:
<resp></resp>
<just></just>
<diag></diag>
<fisiop></fisiop>
<trat></trat>
<farma></farma>
```

⟶

R scripts

Verifying the Correct Answers

Respectively, ScoresLLM.R uses `Exp1RawData.xlsx` and ScoresMMM.R uses `Exp2RawData.xlsx` to verify the correct answers from Experiment 1 and Experiment 2 with `AnswerKey.xlsx`. They generate `ScoresLLM.xlsx` and `ScoresMMM.xlsx`, which contain the counts of correct answers from the five trials of all large language models (LLMs) capable of solving textual questions and the four multimodal models (MMMs), the only models capable of handling image-based questions tested in Experiment 2 to which we had access.

Rscript: ScoresLLM.R

```
# Verifying the Correct Answers
dt_rawdata <- data.frame(readxl::read_excel("Exp1RawData.xlsx"))
models <- c("Titan Text", "Command R +", "LLhama-3-8b", "LLhama-3-70b", "Mixtral 8x7B",
            "Opus", "Sonnet 3", "Haiku") #, "Sonnet 3.5", "GPT 4o")
dt_answerkey <-  data.frame(readxl::read_excel("AnswerKey.xlsx"))
dt_scores <-  dt_answerkey
trials <- sort(unique(dt_rawdata$Trial))
dt_scores$find.qst <- FALSE # to register existing questions
for(m.aux in models) # models
{
  for(i.aux in trials) # iteractions
  {
    o.names <- names(dt_scores)
    dt_scores$new <- 0 # column to compute correct answers
    find.qst <- FALSE
    for(g.aux in 1:nrow(dt_scores))
    {
      answer.model <- dt_rawdata$Answer.model[dt_rawdata$Model==m.aux &
                        dt_rawdata$Trial==i.aux &
                        dt_rawdata$Question==dt_scores$Question[g.aux]]
      if(length(answer.model)>0)
      {
        find.qst <- TRUE # add of column
        dt_scores$find.qst[g.aux] <- TRUE # the question exists
        find.qst <- TRUE
        if(!is.na(answer.model))
        {
          if(answer.model == dt_scores$Answer.key[g.aux])
          {
            dt_scores$new[dt_scores$Question==dt_scores$Question[g.aux]] <- 1
          }
        }
      }
    }
    if(find.qst)
      names(dt_scores) <- c(o.names,paste0(m.aux,"_i",i.aux))
  }
}
dt_scores <- dt_scores[dt_scores$find.qst,]
dt_scores$find.qst <- NULL
openxlsx2::write_xlsx(dt_scores,"ScoresLLM.xlsx")
```

ScoresMMM.R is very similar to ScoresLLM.R; the only changes are the

input worksheet, the model list, and the output worksheet:

Rscript: ScoresMMM.R

```
# Verifying the Correct Answers
dt_rawdata <- data.frame(readxl::read_excel("Exp2RawData.xlsx"))
models <- models <- c("Sonnet 3.5", "Opus", "Sonnet 3", "Haiku")
dt_answerkey <-  data.frame(readxl::read_excel("AnswerKey.xlsx"))
dt_scores <-  dt_answerkey
trials <- sort(unique(dt_rawdata$Trial))
dt_scores$find.qst <- FALSE # to register existing questions
for(m.aux in models) # models
{
  for(i.aux in trials) # iteractions
  {
    o.names <- names(dt_scores)
    dt_scores$new <- 0 # column to compute correct answers
    find.qst <- FALSE
    for(g.aux in 1:nrow(dt_scores))
    {
      answer.model <- dt_rawdata$Answer.model[dt_rawdata$Model==m.aux &
                        dt_rawdata$Trial==i.aux &
                        dt_rawdata$Question==dt_scores$Question[g.aux]]
      if(length(answer.model)>0)
      {
        find.qst <- TRUE # add of column
        dt_scores$find.qst[g.aux] <- TRUE # the question exists
        find.qst <- TRUE
        if(!is.na(answer.model))
        {
          if(answer.model == dt_scores$Answer.key[g.aux])
          {
            dt_scores$new[dt_scores$Question==dt_scores$Question[g.aux]] <- 1
          }
        }
      }
    }
    if(find.qst)
      names(dt_scores) <- c(o.names,paste0(m.aux,"_i",i.aux))
  }
}
dt_scores <- dt_scores[dt_scores$find.qst,]
dt_scores$find.qst <- NULL
openxlsx2::write_xlsx(dt_scores,"ScoresMMM.xlsx")
```

Mean scores

MeansLLM.R uses `ScoresLLM.xlsx` to generate `MeansLLM.xlsx`, which contains the mean score segmented by model, trial, and area.

The five major areas are Gynecology and Obstetrics, Internal Medicine, Pediatrics, Public Health, and Surgery. Of the 117 total questions, 74 are textual (containing only text) while the others include non-radiological or radiological images (posing an additional challenge for the interpretation by artificial intelligence models).

Rscript: MeansLLM.R

```
# Means (experiments 1)
# per area, model, and kind of question
dt_scores <- readxl::read_excel("ScoresLLM.xlsx")
models <- c("Titan Text", "Command R +", "LLhama-3-8b", "LLhama-3-70b", "Mixtral 8x7B",
            "Opus", "Sonnet 3", "Haiku") #, "Sonnet 3.5", "GPT 4o")
kinds <- unique(dt_scores$Image.kind) # it has only textual questions
trials <- 1:5

# data per kind of question
lstradardata <- list()
for(k.aux in 1:length(kinds))
{
  # counting correct answers
  Areas <- c("All areas",unique(dt_scores$Area))
  dt_mean <- data.frame(matrix(
    nrow=length(names(dt_scores)[5:ncol(dt_scores)]),
    ncol=length(Areas)))
  colnames(dt_mean) <- Areas
  rownames(dt_mean) <- names(dt_scores)[5:ncol(dt_scores)]
  numqst <- c()
  lstqstperarea <- list()
  for (a.aux in 1:length(Areas))
  {
    if(k.aux==1)
    {
      if(a.aux==1)
      {
        rg.aux <- 1:nrow(dt_scores)
      } else
      {
        rg.aux <- which(dt_scores$Area==Areas[a.aux])
      }
    } else
    {
      if(a.aux==1)
      {
        rg.aux <- which(dt_scores$Image.kind==kinds[k.aux])
      } else
      {
        rg.aux <- which(dt_scores$Image.kind==kinds[k.aux] &
                        dt_scores$Area==Areas[a.aux])
      }
    }
```

⟶

```
    r.aux <- c()
    for(q.aux in rg.aux)
    {
      r.aux <- c(r.aux,which(dt_scores$Question==dt_scores$Question[q.aux]))
    }
    numqst <- c(numqst,dt_scores$Question[r.aux])
    lstqstperarea[[Areas[a.aux]]] <- dt_scores$Question[r.aux]
    for (an.aux in models)
    {
      for (ai.aux in 1:5)
      {
        model <- paste0(an.aux,"_i",ai.aux)
        c.aux <- which(names(dt_scores)==model)
        dt_mean[rownames(dt_mean)==model,colnames(dt_mean)==Areas[a.aux]] <-
          sum(dt_scores[r.aux,c.aux],na.rm=TRUE)/length(r.aux)*100
      }
    }
  }

  order <- c(
    "Gynec. and Obst.",
    "Surgery",
    "Pub. Health",
    "Pediatrics",
    "Int. Med.",
    "All areas"
  )
  dt_mean <- dt_mean[,order]

  # find the number of questions
  o.numqst <- c()
  for (a.aux in Areas)
  {
    v.qst <- lstqstperarea[[a.aux]]
    o.numqst <- c(o.numqst,length(v.qst))
  }

  # save radar data
  dt_mean <- cbind(rownames(dt_mean),dt_mean)
  rownames(dt_mean) <- NULL
  names(dt_mean) <- c("Model",names(dt_mean)[2:ncol(dt_mean)])
  dt_mean <- rbind(rep(NA,ncol(dt_mean)),dt_mean)
  dt_mean[1,1] <- "Questions"
  for(a.aux in 1:length(Areas))
  {
    dt_mean[1,which(names(dt_mean)==Areas[a.aux])] <- o.numqst[a.aux]
  }
  lstradardata[[kinds[k.aux]]] <- dt_mean
}
openxlsx2::write_xlsx(lstradardata,"MeansLLM.xlsx")
```

Similarly, MeansMMM.R uses `ScoresMMM.xlsx` to generate `MeansMMM.xlsx`, which contains the mean score segmented by model, trial, area, and (additionally) the question types. The modifications are the input

worksheet, the model list, and the output worksheet:

Rscript: MeansMMM.R

```
# Means (experiments 2 and 3)
# per area, model, and kind of question
dt_scores <- readxl::read_excel("ScoresMMM.xlsx")
models <- c("Sonnet 3.5", "Opus", "Sonnet 3", "Haiku")
kinds <- c("All questions",unique(dt_scores$Image.kind))
trials <- 1:5

# data per kind of question
lstradardata <- list()
for(k.aux in 1:length(kinds))
{
  # counting correct answers
  Areas <- c("All areas",unique(dt_scores$Area))
  dt_mean <- data.frame(matrix(
    nrow=length(names(dt_scores)[5:ncol(dt_scores)]),
    ncol=length(Areas)))
  colnames(dt_mean) <- Areas
  rownames(dt_mean) <- names(dt_scores)[5:ncol(dt_scores)]
  numqst <- c()
  lstqstperarea <- list()
  for (a.aux in 1:length(Areas))
  {
    if(k.aux==1)
    {
      if(a.aux==1)
      {
        rg.aux <- 1:nrow(dt_scores)
      } else
      {
        rg.aux <- which(dt_scores$Area==Areas[a.aux])
      }
    } else
    {
      if(a.aux==1)
      {
        rg.aux <- which(dt_scores$Image.kind==kinds[k.aux])
      } else
      {
        rg.aux <- which(dt_scores$Image.kind==kinds[k.aux] &
                        dt_scores$Area==Areas[a.aux])
      }
    }
    r.aux <- c()
```

⟶

⟶

```
    for(q.aux in rg.aux)
    {
      r.aux <- c(r.aux,which(dt_scores$Question==dt_scores$Question[q.aux]))
    }
    numqst <- c(numqst,dt_scores$Question[r.aux])
    lstqstperarea[[Areas[a.aux]]] <- dt_scores$Question[r.aux]
    for (an.aux in models)
    {
      for (ai.aux in 1:5)
      {
        model <- paste0(an.aux,"_i",ai.aux)
        c.aux <- which(names(dt_scores)==model)
        dt_mean[rownames(dt_mean)==model,colnames(dt_mean)==Areas[a.aux]] <-
          sum(dt_scores[r.aux,c.aux],na.rm=TRUE)/length(r.aux)*100
      }
    }
  }

  order <- c(
    "Gynec. and Obst.",
    "Surgery",
    "Pub. Health",
    "Pediatrics",
    "Int. Med.",
    "All areas"
  )
  dt_mean <- dt_mean[,order]

  # find the number of questions
  o.numqst <- c()
  for (a.aux in Areas)
  {
    v.qst <- lstqstperarea[[a.aux]]
    o.numqst <- c(o.numqst,length(v.qst))
  }

  # save radar data
  dt_mean <- cbind(rownames(dt_mean),dt_mean)
  rownames(dt_mean) <- NULL
  names(dt_mean) <- c("Model",names(dt_mean)[2:ncol(dt_mean)])
  dt_mean <- rbind(rep(NA,ncol(dt_mean)),dt_mean)
  dt_mean[1,1] <- "Questions"
  for(a.aux in 1:length(Areas))
  {
    dt_mean[1,which(names(dt_mean)==Areas[a.aux])] <- o.numqst[a.aux]
  }
  lstradardata[[kinds[k.aux]]] <- dt_mean
}

openxlsx2::write_xlsx(lstradardata,"MeansMMM.xlsx")
```

Descriptive statistics and repeated measures One-way ANOVA

Anova.R uses `AccTimeToken.xlsx` to record its results in `Descriptive.xlsx` and `ANOVA.xlsx`. The computed values correspond to Tables 1 and 2 and Figure 3 of the manuscript to which this appendix refers (token analysis was not included in the manuscript). This script also generates additional graphs with estimated marginal means, depending on the `fig.in.file` setting: displayed on screen (`FALSE`) or saved to disk (`TRUE`). These graphs, not included in the manuscript, are described below.

Rscript: Anova.R

```
library(ggplot2)

source("summarySEwithin2.R")

fig.in.file <- FALSE # TRUE to generate PDFs with graphs

alpha <- 0.05

exp1 <- data.frame(readxl::read_excel("AccTimeToken.xlsx",sheet="LLM"))
exp1$Model <- gsub("Mixtral 8x7B Instruct","Mixtral 8x7B",exp1$Model)
exp1$Model <- gsub("Titan Text G1 - Express","Titan Text",exp1$Model)
exp1$Model <- factor(exp1$Model,
                     levels=c(
                       "Titan Text",
                       "LLhama-3-8b",
                       "Mixtral 8x7B",
                       "LLhama-3-70b",
                       "Command R +",
                       "Claude-3-haiku",
                       "GPT-4 Turbo",
                       "Claude-3.5-sonnet",
                       "Claude-3-opus",
                       "Claude-3-sonnet"
                     ))
exp1$Accuracy <- as.numeric(exp1$Accuracy)*100
exp1$Time <- as.numeric(exp1$Time)
exp1$Tokens <- gsub("[^0-9]", "", exp1$Tokens)
exp1$Tokens <- as.numeric(exp1$Tokens)

exp2 <- data.frame(readxl::read_excel("AccTimeToken.xlsx",sheet="MMM"))
exp2$Model <- factor(exp2$Model,
                     levels=c(
                       "Claude-3-haiku",
                       "Claude-3.5-sonnet",
                       "Claude-3-opus",
                       "Claude-3-sonnet"
                     ))
exp2$Accuracy <- as.numeric(exp2$Accuracy)*100
exp2$Time <- as.numeric(exp2$Time)
exp2$Tokens <- gsub("[^0-9]", "", exp2$Tokens)
exp2$Tokens <- as.numeric(exp2$Tokens)
```

⟶

⟶

```
o.par <- par(no.readonly=TRUE)

lst.descriptive <- list()
lst.anv <- list()
vds <- c("Accuracy","Time","Tokens")
for (v.aux in vds)
{
  desc.report <- NULL
  desc.report2 <- NULL

  c.aux <- which(names(exp1)==v.aux)
  cat("\n---------\n",v.aux,"\n")
  cat("\n\t- descriptive\n")
  dsc <- psych::describeBy(exp1[[c.aux]], group=exp1$Model, mat=TRUE)
  dsc <- dsc[,c(2,5,6,7,10,11)]
  print(dsc)
  dsc2 <- psych::describeBy(exp2[[c.aux]], group=exp2$Model, mat=TRUE)
  dsc2 <- dsc2[,c(2,5,6,7,10,11)]
  print(dsc2)
  # confidence interval, repeated measures
  ic <- summarySEwithin2(exp1,
                         measurevar=v.aux, withinvars="Model",
                         idvar="Trial", na.rm=TRUE,
                         conf.interval=1-alpha/length(unique(exp1$Model)))
  names(ic) <- gsub(v.aux,"Measure",names(ic))
  ic <- ic[,c(1,3,7)]
  ic$lwr <- ic$Measure-ic$ci
  ic$upr <- ic$Measure+ic$ci
  ic$ci <- NULL
  names(ic)[2:4] <- c (
    v.aux,
    paste0("lwr.CI",round((1-alpha)*100,1),"%"),
    paste0("upr.CI",round((1-alpha)*100,1),"%")
  )
  # ic and dsc in levels order
  ic$tmporder <- 0
  dsc$tmporder <- 0
  tmplevels <- levels(exp1$Model)
  for(r.aux in 1:length(tmplevels))
  {
    ic$tmporder[ic$Model==tmplevels[r.aux]] <- r.aux
    dsc$tmporder[dsc$group1==tmplevels[r.aux]] <- r.aux
  }
  ic <- ic[order(ic$tmporder),]
  dsc <- dsc[order(dsc$tmporder),]
  ic$tmporder <- NULL
  dsc$tmporder <- NULL
  # experiment 2
  ic2 <- summarySEwithin2(exp2,
                          measurevar=v.aux,
                          withinvars="Model",
                          idvar="Trial",
                          na.rm=TRUE,
                          conf.interval=1-alpha/length(unique(exp2$Model)))
  names(ic2) <- gsub(v.aux,"Measure",names(ic2))
  ic2 <- ic2[,c(1,3,7)]
  ic2$lwr <- ic2$Measure-ic2$ci
  ic2$upr <- ic2$Measure+ic2$ci
  ic2$ci <- NULL
  names(ic2)[2:4] <- c (
    v.aux,
    paste0("lwr.CI",round((1-alpha)*100,1),"%"),
    paste0("upr.CI",round((1-alpha)*100,1),"%")
  )
  # ic2 and dsc2 in levels order
  ic2$tmporder <- 0
  dsc2$tmporder <- 0
  tmplevels <- levels(exp2$Model)
  for(r.aux in 1:length(tmplevels))
  {
    ic2$tmporder[ic2$Model==tmplevels[r.aux]] <- r.aux
    dsc2$tmporder[dsc2$group1==tmplevels[r.aux]] <- r.aux
  }
  ic2 <- ic2[order(ic2$tmporder),]
  dsc2 <- dsc2[order(dsc2$tmporder),]
  ic2$tmporder <- NULL
  dsc2$tmporder <- NULL

  print(ic)
```

⟶

```
# Experiment 1
# colocar modelos na mesma ordem
desc <- as.data.frame(cbind(
  ic$Model,ic[,2],dsc$sd,dsc$median,dsc$min,dsc$max,ic$`lwr.CI95%`,ic$`upr.CI95%`))
names(desc) <- c("Model",v.aux,"St.Dev.","Median","Min.","Max.",names(ic)[3:4])
desc$Model <- ic$Model
print(desc,digits = 2)
names(desc)[3:ncol(desc)] <- paste0(v.aux," ",names(desc)[3:ncol(desc)])
if(is.null(desc.report))
  desc.report <- desc
else
  desc.report <- cbind(desc.report,desc)
lst.descriptive[[paste0(v.aux,"_exp1")]] <- desc.report

# Experiment 2
desc <- as.data.frame(cbind(
  ic2$Model,ic2[,2],dsc2$sd,dsc2$median,
  dsc2$min,dsc2$max,ic2$`lwr.CI95%`,ic2$`upr.CI95%`))
names(desc) <- c("Model",v.aux,"St.Dev.","Median","Min.","Max.",names(ic2)[3:4])
desc$Model <- ic2$Model
print(desc,digits = 2)
names(desc)[3:ncol(desc)] <- paste0(v.aux," ",names(desc)[3:ncol(desc)])
if(is.null(desc.report2))
  desc.report2 <- desc
else
  desc.report2 <- cbind(desc.report2,desc)
lst.descriptive[[paste0(v.aux,"_exp2")]] <- desc.report2

cat("\n\t- repeated Measures one-way ANOVA\n")
modelo <- lmerTest::lmer(exp1[[c.aux]] ~ Model + (1|Trial), data=exp1)

print(anv <- car::Anova(modelo,
                        test.statistic="F"))
lst.anv[[paste0(v.aux,"_ANV_Exp1")]] <- anv
# cat("\nRegression")
# print(summary(modelo, correl=FALSE))
# cat("\nEffect size analysis")
# eta2 <- effectsize::eta_squared(anv,
#                                 partial=FALSE,
#                                 generalized=FALSE,
#                                 ci=1-alpha,
#                                 alternative="two.sided",
#                                 verbose=TRUE)
# eta2$interpret <- effectsize::interpret_eta_squared(eta2$Eta2)
# print(eta2, digits=4)

cat("\n\t\t--- post hoc tests\n")
emm <- emmeans::emmeans(modelo,
                        pairwise~"Model",
                        adjust="holm",
                        level=1-alpha,
                        lmer.df="satterthwaite",
                        lmerTest.limit=nrow(exp1))
# print(emm$emmeans)
if(fig.in.file)
  pdf(paste0(v.aux,"_MargMeans_exp1.pdf"),width=6,height=3)
p <- plot(emm$emmeans,
          xlab=paste0("Experiment 1: ", v.aux,
                      " (Estimated Marginal Means)"),
          colors="black") +
  theme(
    panel.grid.major = element_line(color = "black", linetype = "dotted"),
    panel.grid.minor = element_line(color = "gray", linetype = "dotted")
  )
print(p)
if(fig.in.file)
  dev.off()
```

```
if(fig.in.file)
  pdf(paste0(v.aux,"_Contrasts_exp1.pdf"),width=8,height=7)
p <- plot(emm$contrasts,
          xlab=paste0("Experiment 1: ", v.aux,
                      " (Differences of Estimated Marginal Means)"),
          colors="black") +
  geom_vline(xintercept = 0, linetype = "solid", color = "black", size = 1) +
  theme(
    panel.grid.major = element_line(color = "black", linetype = "dotted"),
    panel.grid.minor = element_line(color = "gray", linetype = "dotted")
  )
print(p)
if(fig.in.file)
  dev.off()

print(mc <- multcomp::cld(object=emm$emmeans,
                          level=1-alpha,
                          adjust="holm",
                          Letters=letters,
                          alpha=alpha))
lst.anv[[paste0(v.aux,"_post_Exp1")]] <- mc

cat("\n\t- repeated Measures one-way ANOVA (MMM only)\n")
modelo <- lmerTest::lmer(exp2[[c.aux]] ~ Model + (1|Trial), data=exp2)
print(anv <- car::Anova(modelo,
                        test.statistic="F"))
lst.anv[[paste0(v.aux,"_ANV_Exp2")]] <- anv

# cat("\nRegression")
# print(summary(modelo, correl=FALSE))
# cat("\nEffect size analysis")
# eta2 <- effectsize::eta_squared(anv,
#                                 partial=FALSE,
#                                 generalized=FALSE,
#                                 ci=1-alpha,
#                                 alternative="two.sided",
#                                 verbose=TRUE)
# eta2$interpret <- effectsize::interpret_eta_squared(eta2$Eta2)
# print(eta2, digits=4)

cat("\n\t\t--- post hoc tests\n")
emm <- emmeans::emmeans(modelo,
                        pairwise~"Model",
                        adjust="holm",
                        level=1-alpha,
                        lmer.df="satterthwaite",
                        lmerTest.limit=nrow(exp1))
print(emm$emmeans)
if(fig.in.file)
  pdf(paste0(v.aux,"_MargMeans_exp2.pdf"),width=6,height=2)
p <- plot(emm$emmeans,
          xlab=paste0("Experiment 2: ", v.aux,
                      " (Estimated Marginal Means)"),
          colors="black") +
  theme(
    panel.grid.major = element_line(color = "black", linetype = "dotted"),
    panel.grid.minor = element_line(color = "gray", linetype = "dotted")
  )
print(p)
if(fig.in.file)
  dev.off()

if(fig.in.file)
  pdf(paste0(v.aux,"_Contrasts_exp2.pdf"),width=8,height=3)
p <- plot(emm$contrasts,
          xlab=paste0("Experiment 2: ", v.aux,
                      " (Differences of Estimated Marginal Means)"),
          colors="black") +
  geom_vline(xintercept = 0, linetype = "solid", color = "black", size = 1) +
  theme(
    panel.grid.major = element_line(color = "black", linetype = "dotted"),
    panel.grid.minor = element_line(color = "gray", linetype = "dotted")
  )
print(p)
if(fig.in.file)
  dev.off()
```

⟶

```
  print(mc <- multcomp::cld(object=emm$emmeans,
                            level=1-alpha,
                            adjust="holm",
                            Letters=letters,
                            alpha=alpha))
  lst.anv[[paste0(v.aux,"_post_Exp2")]] <- mc

  # graphical representation
  par(mar=c(9.5,5,0,0))
  if(v.aux=="Accuracy")
  {
    ymin <- 0
    ymax <- 100
  }
  if(v.aux=="Time")
  {
    ymin <- 0 # min(ic[[2]],na.rm=TRUE)*0.9
    ymax <- 30 # max(ic[[2]],na.rm=TRUE)*1.1
  }
  if(v.aux=="Tokens")
  {
    ymin <- 0 # min(ic[[2]],na.rm=TRUE)*0.9
    ymax <- 2000 # max(ic[[2]],na.rm=TRUE)*1.1
  }
  if(fig.in.file)
    pdf(paste0(v.aux,"_Descriptive_exp12.pdf"),width=6,height=6)
  par(mar=c(9.5,5,0,0))

  plot(0:(length(ic$Model)+1),
       rep(NA,length(ic$Model)+2),
       xlab="",
       ylab=latex2exp::TeX(sprintf(r'($%s \pm CI(95)$)',v.aux)),
       ylim=c(ymin,ymax),
       pch=21,cex=0.1,cex.lab=1.4,axes=FALSE)
  axis(1,
       cex.axis=1.2,
       at=1:length(ic$Model),labels=unique(ic$Model),las=2)
  axis(2,cex.axis=1.2,las=1)
  w <- length(ic$Model)/80
  for(r.aux in 1:nrow(ic))
  {
    x <- c(r.aux-w,r.aux+w,r.aux,r.aux,r.aux+w,r.aux-w)
    y <- c(rep(ic[r.aux,3],3),rep(ic[r.aux,4],3))
    lines(x,y)
  }
  points(1:length(ic$Model),ic[[2]],
         pch=21,col="black",bg="white",cex=1.2)
  # adiciona ic2
  ic$Model <- as.character(ic$Model)
  ic2$Model <- as.character(ic2$Model)
  for(r.aux in 1:nrow(ic))
  {
    dt_tmp <- ic2[ic2$Model==ic$Model[r.aux],]
    if(nrow(dt_tmp)>0)
    {
      x <- c(r.aux-w,r.aux+w,r.aux,r.aux,r.aux+w,r.aux-w)+w*2
      y <- c(rep(dt_tmp[1,3],3),rep(dt_tmp[1,4],3))
      lines(x,y,col="black")
      points(r.aux+w*2,dt_tmp[,2],
             pch=21,col="black",bg="black",cex=1.2)
      lines(c(r.aux,r.aux+w*2),c(ic[r.aux,2],dt_tmp[,2]),lty=3,lwd=0.7)
    }
  }
  points(1:length(ic$Model),ic[[2]],
         pch=21,col="black",bg="white",cex=1.2)
  if(fig.in.file)
    dev.off()

}

par(o.par)

openxlsx::write.xlsx(lst.descriptive,"Descriptive.xlsx")
openxlsx::write.xlsx(lst.anv,"ANOVA.xlsx")
```

Descriptive statistics applied `psych::describeBy` for central tendency (mean and median) and dispersion (standard deviation and range) measures. Standard errors assuming repeated measures (five trials of each model using the same question set) for descriptive statistics were based on the `seWithin` function from

> Lin H. hausekeep: A Collection of Utility Functions for Data Science and Statistics;. Available from: https://hauselin.github.io/hausekeep/.

The modified function applied here is implemented in `summarySEwithin2.R`. From these corrected standard errors, 95% confidence intervals were estimated.

Differences in accuracy and processing time were analyzed using a linear mixed-effects model with a fixed effect for the `Model` and a random intercept for `Trial` (to account for repeated measures), which is equivalent to a repeated measures one-way ANOVA in this context, using

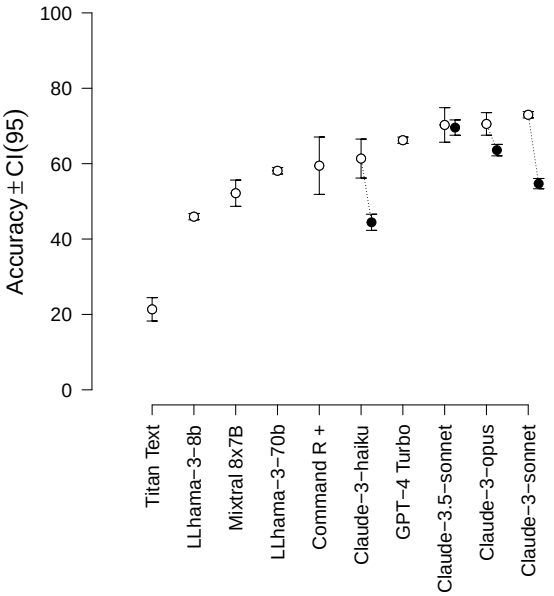
```
lmerTest::lmer(Var ~ Model + (1|Trial))
```

in which `Var` is `Accuracy`, `Time`, or `Tokens`.

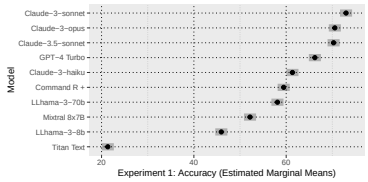
Post-hoc pairwise comparisons between models were conducted using estimated marginal means with Holm's adjustment for multiple comparisons, implemented via the `emmeans::emmeans` function from

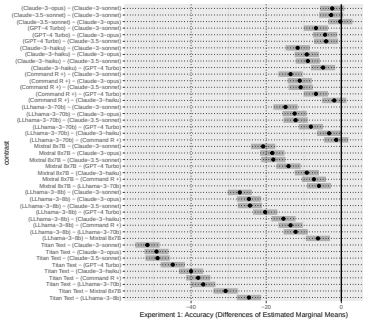
> Lenth RV. emmeans: Estimated marginal means, aka least-squares means. R package (version 1.7.1). R Foundation for Statistical Computing. 2021;34.

with confidence level set to 95% and Satterthwaite's method to approximate degrees of freedom. The corresponding graphical outputs are shown in Figs. A-1, A-2, and A-3.

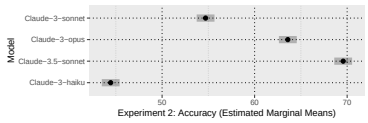


Experiment 1:





Experiment 2:



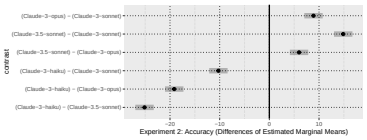
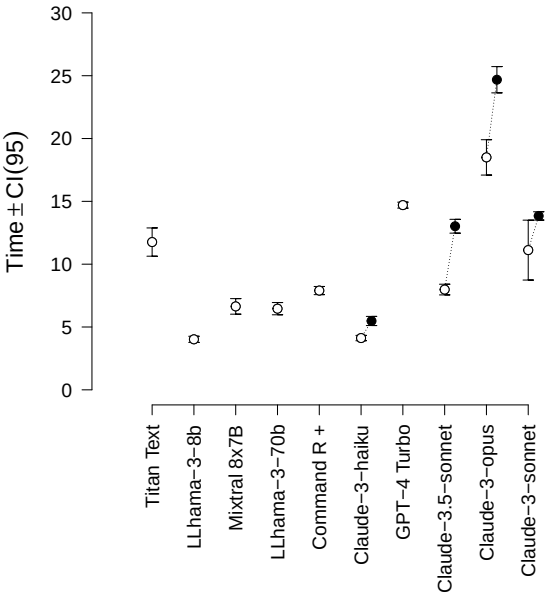
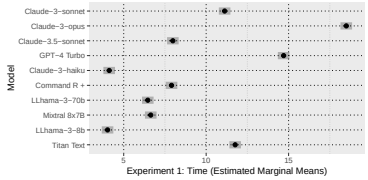
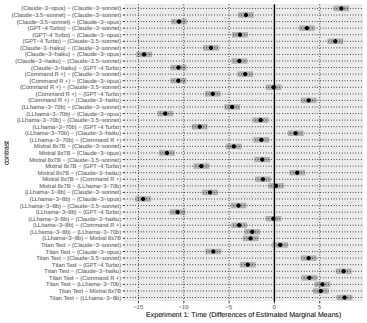


Figure A-1: Comparison of mean accuracy (`Accuracy_Descriptive_exp12.pdf`). Empty circles: experiment 1 with 74 textual questions; filled circles: experiment 2 with 117 questions (including questions containing images). Emmeans representations: `Accuracy_MargMeans_exp1.pdf`, `Accuracy_Contrasts_exp1.pdf`, `Accuracy_MargMeans_exp2.pdf`, `Accuracy_Contrasts_exp2.pdf`.

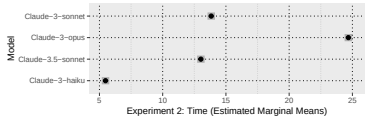


Experiment 1:





Experiment 2:



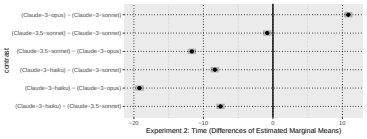
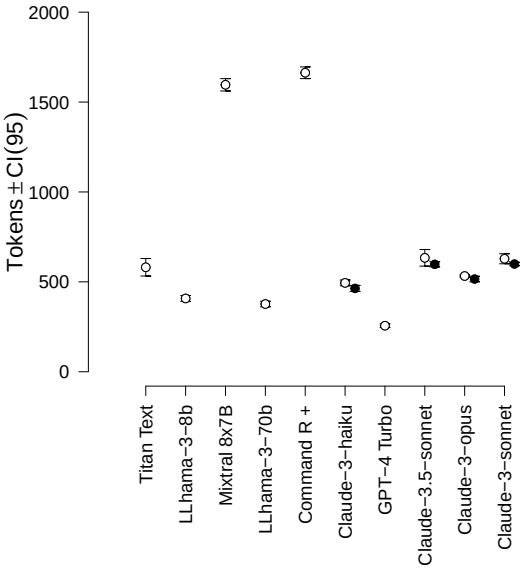
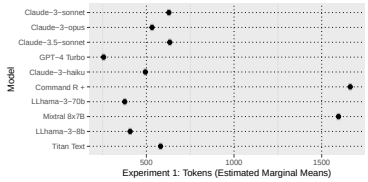
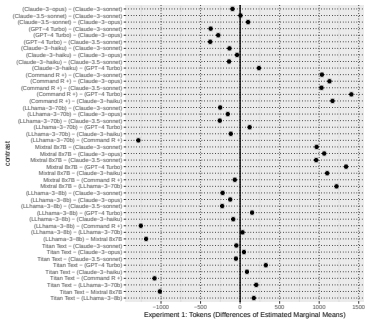


Figure A-2: Comparison of mean processing time (`Time_Descriptive_exp12.pdf`). Empty circles: experiment 1 with 74 textual questions; filled circles: experiment 2 with 117 questions (including questions containing images). Emmeans representations: `Time_MargMeans_exp1.pdf`, `Time_Contrasts_exp1.pdf`, `Time_MargMeans_exp2.pdf`, `Time_Contrasts_exp2.pdf`.

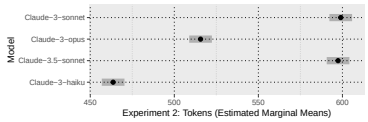


Experiment 1:





Experiment 2:



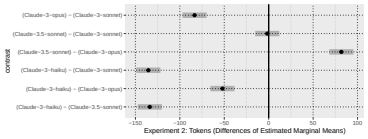


Figure A-3: Comparison of mean number of required tokens (`Tokens_Descriptive_exp12.pdf`). Empty circles: experiment 1 with 74 textual questions; filled circles: experiment 2 with 117 questions (including questions containing images). Emmeans representations: `Tokens_MargMeans_exp1.pdf`, `Tokens_Contrasts_exp1.pdf`, `Tokens_MargMeans_exp2.pdf`, `Tokens_Contrasts_exp2.pdf`.

Distribution of Human Responses vs. Artificial Intelligences

In experiment 2, the four MMMs were positioned within the distribution of responses given by candidates applying for medical residency.

DistExp2.R uses `AnswerHuman.xlsx` and `AccTimeToken.xlsx` to replicate the Figure 5 of the manuscript to which this appendix refers, showing the accuracy distribution achieved by candidates in the residency exam and the average performance location in five attempts by the MMMs (Fig. A-4). If `fig.in.file` is set to TRUE, the figure is saved to disk and available as `CandidateDistributionExp2.pdf` (replicated here as Fig. A-4).

Rscript: DistExp2.R

```
fig.in.file <- FALSE # TRUE to generate PDFs with graphs

experiment <- data.frame(readxl::read_excel("AccTimeToken.xlsx",sheet="MMM"))
experiment$Model <- factor(experiment$Model)
experiment$Accuracy <- as.numeric(experiment$Accuracy)*100
experiment$Time <- as.numeric(experiment$Time)
experiment$Tokens <- gsub("[^0-9]", "", experiment$Tokens)
experiment$Tokens <- as.numeric(experiment$Tokens)
AI_name <- unique(as.character(experiment$Model))

human <- readxl::read_excel("AnswerHuman.xlsx")
human$Accuracy <- human$Correct/117*100
density <- density(human$Accuracy,na.rm=TRUE)
median <- median(human$Accuracy,na.rm=TRUE)
accuracy <- human$Accuracy
n <- length(accuracy)
mean <- mean(accuracy, na.rm=TRUE)
stddev <- sd(accuracy)
stderr <- stddev / sqrt(n)
# Nível de confiança (95%)
z <- qnorm(0.975)  # critical value 95% (1 - alpha/2)
# Intervalo de Confiança
ci.accuracy <- c(mean - z * stderr,mean + z * stderr)

# location of AIs in distribution
mean.model <- c()
for (i in 1:length(AI_name))
{
  mmod <- mean(experiment$Accuracy[experiment$Model==AI_name[i]],na.rm=TRUE)
  mean.model <- c(mean.model,mmod)
}
# order of model accuracy
dt_tmp <- data.frame(AI_name,mean.model)
dt_tmp <- dt_tmp[order(dt_tmp$mean.model,decreasing=TRUE),]
AI_name <- dt_tmp$AI_name
mean.model <- dt_tmp$mean.model
diff <- list()
for (i in 1:length(AI_name))
{
  mmod <- mean(experiment$Accuracy[experiment$Model==AI_name[i]],na.rm=TRUE)
  diff[[AI_name[i]]] <- abs(density$x - mmod)
  min <- min(diff[[AI_name[i]]])
  pos <- which(diff[[AI_name[i]]]==min)
  diff[[AI_name[i]]] <- c(mmod,density$y[pos])
}
```

⟶

⟶

```
if(fig.in.file)
  pdf("CandidateDistributionExp2.pdf", width=9, height=8)

ymin <- 0
ymax <- max(density$y)*1.1;
wy <- ymax/80
plot(density,
     main="Performance of Residency Applicants",
     xlab="Accuracy (%)", ylab="Density probability",
     ylim=c(ymin,ymax),
     bty="n")
abline(v=mean,lty=2)
for (i in 1:length(AI_name))
{
  # ic <- c(ic2$`lwr.CI95%`[ic2$Model==AI_name[i]],
  #         ic2$`upr.CI95%`[ic2$Model==AI_name[i]])
  # x <- c(rep(ic[1],3),rep(ic[2],3))
  # y <- rep(diff[[AI_name[i]]][2],6) + c(wy,-wy,0,0,-wy,wy)
  # lines(x,y)
  points(diff[[AI_name[i]]][1],diff[[AI_name[i]]][2],
         cex=2.5, pch=21, col="black", bg="white")
  dt_tmp <- experiment[experiment$Model==AI_name[i],]
  dt_tmp <- dt_tmp[order(dt_tmp$Accuracy),]
  dt_tmp$addy <- 0.0014
  old.acc <- dt_tmp$Accuracy[1]
  for (i.aux in 2:nrow(dt_tmp))
  {
    if(old.acc == dt_tmp$Accuracy[i.aux])
    {
      dt_tmp$addy[i.aux] <- dt_tmp$addy[i.aux-1]+0.0005
    }
    old.acc <- dt_tmp$Accuracy[i.aux]
  }
  text(dt_tmp$Accuracy,
       diff[[AI_name[i]]][2]+dt_tmp$addy,
       paste0(i),cex=0.5)
  # points(dt_tmp$Accuracy,
  #      diff[[AI_name[i]]][2]+dt_tmp$addy,
  #      cex=0.5)
  text(diff[[AI_name[i]]][1],diff[[AI_name[i]]][2],paste0(i),cex=0.5)
}
legend ("topleft",
        c(paste0(1:length(AI_name),": ",AI_name," (",
              round(mean.model,1),"%)"),"- Candidates' distribution"),
        bty="n", box.lwd=0, bg="transparent")
```

We can roughly observe that the performance of the MMMs (see Fig. A-1) showed some deterioration when handling questions including images (experiment 2) compared to the condition where they only responded to textual questions (experiment 1). Although not entirely accurate, considering the hypothesis that this difference may disappear with the evolution of AIs, we compared the same distribution of human scores with the performances from experiment 1. Simply changing the spreadsheet tab in `DistExp2.R` from `sheet="MMM"` to `sheet="LLM"` generates Fig. A-5. This modification is implemented in `DistExp1.R`.

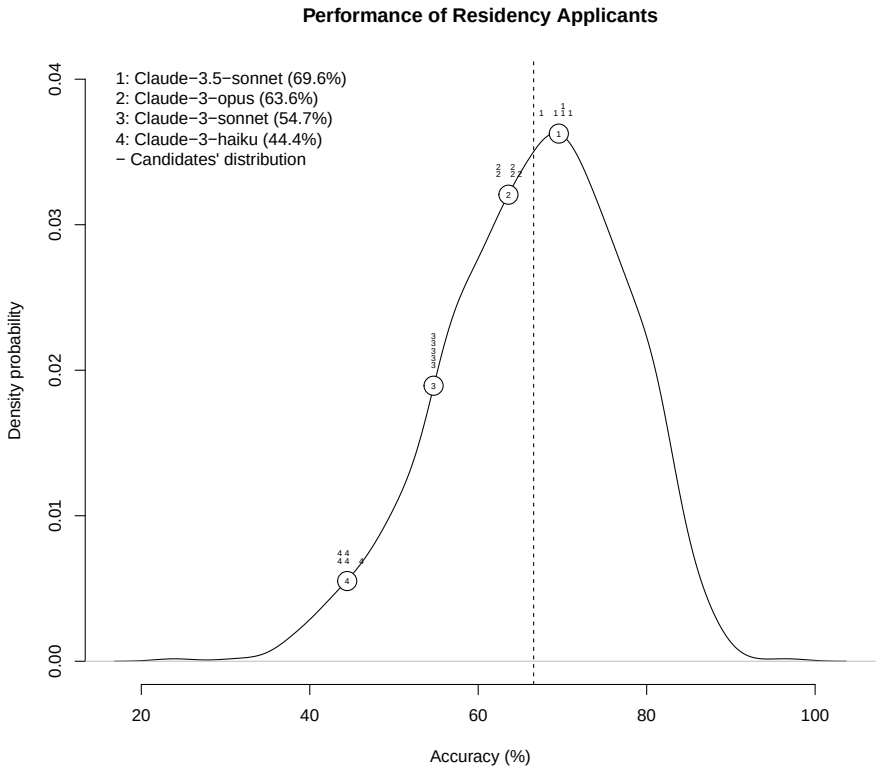


Figure A-4: Distribution of accuracy (solid line) and median (dashed line) of residency applicants, along with the accuracy achieved by different AI multimodal models (small numbers represent the accuracy obtained in five individual trials).

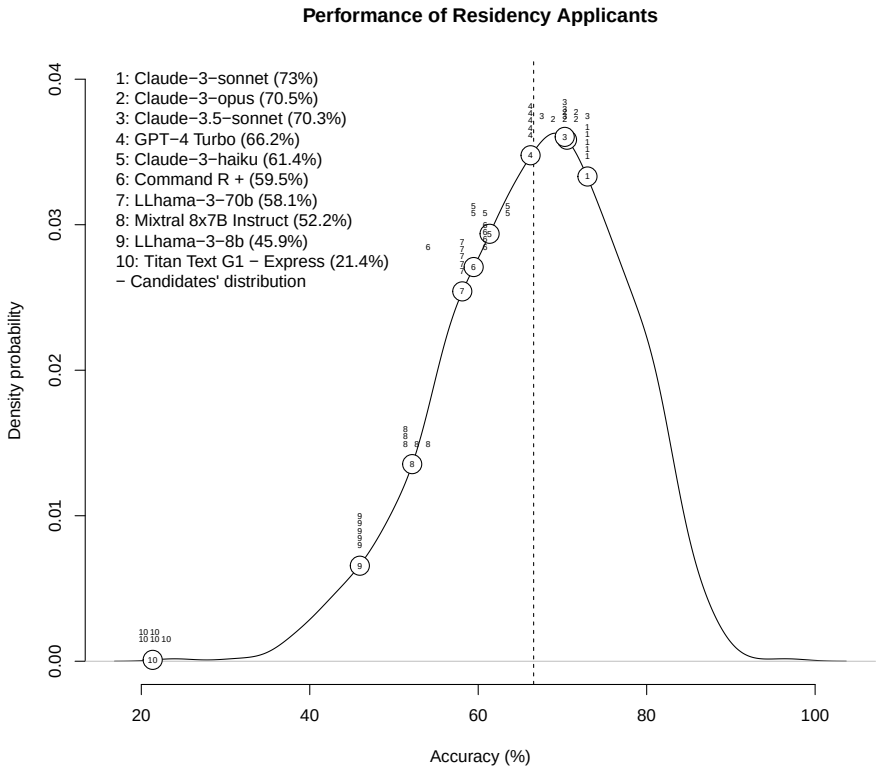


Figure A-5: Distribution of accuracy (solid line) and median (dashed line) of residency applicants, along with the accuracy achieved by different AI models (small numbers represent the accuracy obtained in five individual trials).

Radar graph

RadarPerArea.R uses `MeansMMM.xlsx` to replicate Figure 4 from the manuscript to which this appendix refers, showing the percentage of correct answers obtained by the four MMMs, segmented by area and question type (Fig. A-6). If `fig.in.file` is set to TRUE, the figure is saved to disk and available as `AnswerRadarPerImage.pdf`.

Rscript: RadarPerArea.R

```
# Radar (experiment 3)
# per area, model, and kind of question
fig.in.file <- FALSE # TRUE to generate PDF with radar graphs

kinds <- readxl::excel_sheets("MeansMMM.xlsx")
models <- c("Sonnet 3.5", "Opus", "Sonnet 3", "Haiku")
trials <- 1:5

# Create radar graphs
if(fig.in.file)
  pdf("AnswerRadarPerImage.pdf",width=12,height=10)

pos <- c(
  1, 1, 1, 1, 1, 1,
  2, 2, 2, 3, 3, 3,
  2, 2, 6, 6, 3, 3,
  4, 4, 6, 6, 5, 5,
  4, 4, 4, 5, 5, 5
)
m <- matrix(pos,nrow=5,ncol=6,byrow=TRUE)
o.par <- par(no.readonly = TRUE)
layout(m,heights=c(1,5,5,5,5),widths=c(1,1,0.3,0.3,1,1))

par(mai=c(0,0.9,0,0.42))
plot(NA,
     xlim=c(0,10),ylim=c(0,3),
     xlab="",ylab="",
     axes=FALSE)
text(5,1,"Correct Answers by AI and Type of Question",cex=2,pos=3)
par(mai=c(0,0,0.5,0))
for(k.aux in 1:length(kinds))
{
  # read pre-processed data (see MeanPerAreaMMM.R)
  dt_mean <- readxl::read_excel("MeansMMM.xlsx",sheet = kinds[k.aux])
  # obtain number o questions and remove first line
  numqst <- dt_mean[1,2:ncol(dt_mean)]
  dt_mean <- dt_mean[2:nrow(dt_mean),]
  # transfer Model column to rownames and remove first column
  dt_mean <- data.frame(dt_mean)
  rownames(dt_mean) <- dt_mean$Model
  dt_mean$Model <- NULL
  # max and min for the correct graph
  min <- 0
  max <- 100 # max(data)
  dt_mean <- rbind(
    rep(max, ncol(dt_mean)),
    rep(min, ncol(dt_mean)),
    dt_mean
  )
```

⟶

⟶

```
  # colnames including number of questions
  colnames(dt_mean) <- gsub("\\.\\."," ",colnames(dt_mean))
  colnames(dt_mean) <- gsub("and\\."," ",colnames(dt_mean))
  for (a.aux in 1:ncol(numqst))
  {
    colnames(dt_mean) <- gsub(names(numqst)[a.aux],
                              paste0(names(numqst)[a.aux]," (",numqst[a.aux],")"),
                              colnames(dt_mean))
  }
  # colors for AI
  colors <- c(
    rep("#E65518",5),
    rep("#26a169",5),
    rep("#1965B0",5),
    rep("#72190E",5)
  )
  colorstp <- paste0(colors,"50")
  pch <- c(
    rep(21,5),
    rep(22,5),
    rep(23,5),
    rep(24,5)
  )
  lwd <- c(
    rep(4.5,5),
    rep(4.0,5),
    rep(3.5,5),
    rep(3.0,5)
  )
  lty <- c(
    rep(1,5),
    rep(1,5),
    rep(1,5),
    rep(1,5)
  )

  # Criar o gráfico radar
  fmsb::radarchartcirc(
    dt_mean,
    cex.main=1.7,
    calcex=1,
    vlcex=1.5,
    axistype = 1,
    axislabcol="black",
    cglcol="gray",
    pcol = colorstp,
    plty=lty,
    pty=pch,
    # pfcol = adjustcolor(colorstp, alpha.f = 0.15),  # transparency fill
    plwd = lwd,
    title=paste0(kinds[k.aux]," (",numqst[length(numqst)],")")
  )
}

# Adicionar legenda
plot(NA,xlim=c(0,10),ylim=c(0,10),
     xlab="",ylab="",
     axes=FALSE)
legend(
  x = "topright",
  legend = models,
  col = paste0(unique(colors),"90"),
  lty = 1,
  lwd = seq(from=4.5,to=3.0,by=-0.5),
  pch=21:24,
  cex=1.4,
  bty = "n"
)

if(fig.in.file)
  dev.off()
```

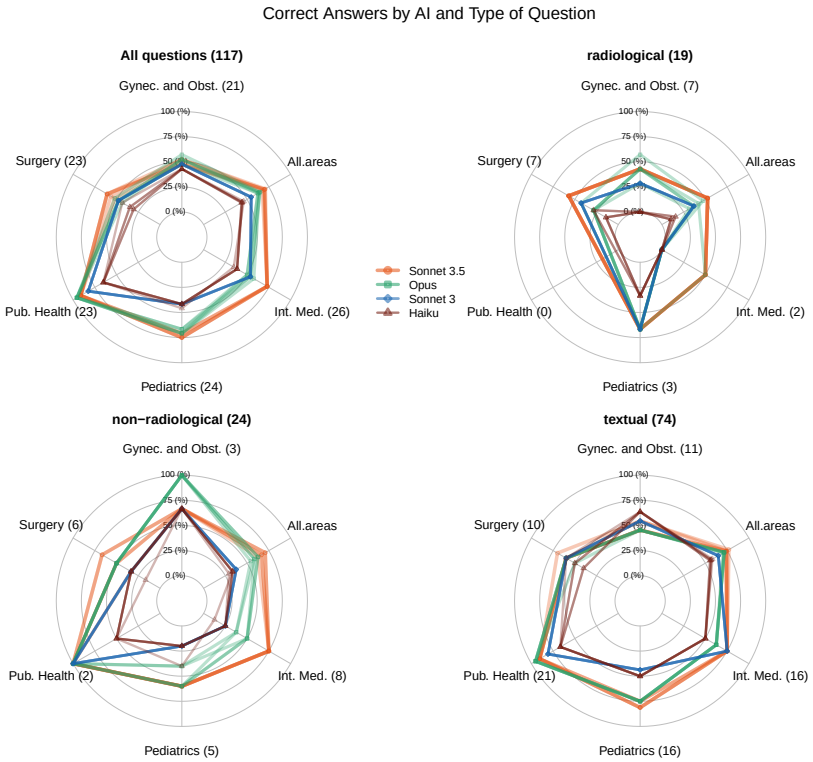


Figure A-6: Comparison of model accuracy (five trials for each model) based on the type of questions (questions containing radiological or non-radiological images, or text only) across five medical areas: Gynecology and Obstetrics, Internal Medicine, Pediatrics, Public Health, and Surgery. Values in parentheses indicate the number of questions.

Agreement between observers

AgreementExp3 uses `ObserverEval.xlsx` to generate `Agreement.xlsx`.

The observers' evaluation of the correctness of the justifications provided by a single trial of the Claude-3-Opus model, their coherence between such justifications and the chosen alternative, and whether the responses from this artificial intelligence model, if used as a decision for a patient, could potentially harm a patient's health, were analyzed in Tables 3 and 4 of the manuscript to which this appendix refers. Since the evaluations were conducted independently, it was also necessary to assess the agreement among observers using Gwet's AC1 statistical test. The analysis was performed for all questions and (for the manuscript) subdivided into those the model answered correctly or incorrectly.

Rscript: AgreementExp3.R

```
source("agr2x2_ac1Gwet.R")

exp <- data.frame(readxl::read_excel("ObserverEval.xlsx"))

cat("\nNumber of correct and incorrect answers of the model:\n")
AIxGaba <- table(exp$Cam01.CorrectAnswer)
print(AIxGaba)
# subset by correct and incorrect model answers
expY <- exp[exp$Cam01.CorrectAnswer=="Y",]
expN <- exp[exp$Cam01.CorrectAnswer=="N",]

# Agreement intraobserver (Justificative x Harm)
cat("\n\tAmong all questions:\n")
AIxGaba <- table(exp$Cam01.CorrectAnswer)
print(AIxGaba)

agreement <- data.frame(matrix(nrow=3*9,ncol=11))
names(agreement) <- c(
  "Case","Observer","Correct.Answer",
  "YY","YN","NY","NN","AC1","AC1.lwr","AC1.upr","p")
w.aux <- 0

cat("\nWithin agreement\n")

cat("\n\tAmong all questions:\n")

cat("\n- Obs1:\n")
t <- table(exp$InterpJust01,exp$Harm01)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Within JH","Obs. 1","All",as.vector(t(t)),as.vector(res))
```

⟶

```
cat("\n- Obs2:\n")
t <- table(exp$InterpJust02,exp$Harm02)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Within JH","Obs. 2","All",as.vector(t(t)),as.vector(res))

cat("\n- Obs3:\n")
t <- table(exp$InterpJust03,exp$Harm03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Within JH","Obs. 3","All",as.vector(t(t)),as.vector(res))


cat("\n\tAmong the questions the model got right:\n")

AIxGaba <- table(expY$Cam01.CorrectAnswer)
print(AIxGaba)

cat("\n- Obs1:\n")
t <- table(expY$InterpJust01,expY$Harm01)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Within JH","Obs. 1","Y",as.vector(t(t)),as.vector(res))

cat("\n- Obs2:\n")
t <- table(expY$InterpJust02,expY$Harm02)
t <- cbind(t,c(0,0))
colnames(t) <- c("N","Y")
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Within JH","Obs. 2","Y",as.vector(t(t)),as.vector(res))

cat("\n- Obs3:\n")
t <- table(expY$InterpJust03,expY$Harm03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Within JH","Obs. 3","Y",as.vector(t(t)),as.vector(res))

cat("\n\tAmong the questions the model got wrong:\n")

AIxGaba <- table(expN$Cam01.CorrectAnswer)
print(AIxGaba)

cat("\n- Obs1:\n")
t <- table(expN$InterpJust01,expN$Harm01)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Within JH","Obs. 1","N",as.vector(t(t)),as.vector(res))
```

```
cat("\n- Obs2:\n")
t <- table(expN$InterpJust02,expN$Harm02)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Within JH","Obs. 2","N",as.vector(t(t)),as.vector(res))

cat("\n- Obs3:\n")
t <- table(expN$InterpJust03,expN$Harm03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Within JH","Obs. 3","N",as.vector(t(t)),as.vector(res))

cat("\nBetween agreement: Justificatives\n")

cat("\n\tAmong all questions:\n")

cat("\n- 1x2:\n")
t <- table(exp$InterpJust01,exp$InterpJust02)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between JJ","Obs. 1x2","All",as.vector(t(t)),as.vector(res))

cat("\n- 1x3:\n")
t <- table(exp$InterpJust01,exp$InterpJust03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between JJ","Obs. 1x3","All",as.vector(t(t)),as.vector(res))

cat("\n- 2x3:\n")
t <- table(exp$InterpJust02,exp$InterpJust03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between JJ","Obs. 2x3","All",as.vector(t(t)),as.vector(res))

cat("\n\tAmong the questions the model got right:\n")

cat("\n- 1x2:\n")
t <- table(expY$InterpJust01,expY$InterpJust02)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between JJ","Obs. 1x2","Y",as.vector(t(t)),as.vector(res))

cat("\n- 1x3:\n")
t <- table(expY$InterpJust01,expY$InterpJust03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between JJ","Obs. 1x3","Y",as.vector(t(t)),as.vector(res))
```

```
cat("\n- 2x3:\n")
t <- table(expY$InterpJust02,expY$InterpJust03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between JJ","Obs. 2x3","Y",as.vector(t(t)),as.vector(res))

cat("\n\tAmong the questions the model got wrong:\n")

cat("\n- 1x2:\n")
t <- table(expN$InterpJust01,expN$InterpJust02)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between JJ","Obs. 1x2","N",as.vector(t(t)),as.vector(res))

cat("\n- 1x3:\n")
t <- table(expN$InterpJust01,expN$InterpJust03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between JJ","Obs. 1x3","N",as.vector(t(t)),as.vector(res))

cat("\n- 2x3:\n")
t <- table(expN$InterpJust02,expN$InterpJust03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between JJ","Obs. 2x3","N",as.vector(t(t)),as.vector(res))

cat("\nBetween agreement: Harm\n")

cat("\n\tAmong all questions:\n")

cat("\n- 1x2:\n")
t <- table(exp$Harm01,exp$Harm02)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between HH","Obs. 1x2","All",as.vector(t(t)),as.vector(res))

cat("\n- 1x3:\n")
t <- table(exp$Harm01,exp$Harm03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between HH","Obs. 1x3","All",as.vector(t(t)),as.vector(res))

cat("\n- 2x3:\n")
t <- table(exp$Harm02,exp$Harm03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between HH","Obs. 2x3","All",as.vector(t(t)),as.vector(res))
```

⟶

```
cat("\n\tAmong the questions the model got right:\n")

cat("\nBetween agreement: Harm\n")
cat("\n- 1x2:\n")
t <- table(expY$Harm01,expY$Harm02)
t <- cbind(t,c(0,0))
colnames(t) <- c("N","Y")
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between HH","Obs. 1x2","Y",as.vector(t(t)),as.vector(res))

cat("\n- 1x3:\n")
t <- table(expY$Harm01,expY$Harm03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between HH","Obs. 1x3","Y",as.vector(t(t)),as.vector(res))

cat("\n- 2x3:\n")
t <- table(expY$Harm02,expY$Harm03)
t <- rbind(t,c(0,0))
rownames(t) <- c("N","Y")
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between HH","Obs. 2x3","Y",as.vector(t(t)),as.vector(res))

cat("\n\tAmong the questions the model got wrong:\n")

cat("\n- 1x2:\n")
t <- table(expN$Harm01,expN$Harm02)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between HH","Obs. 1x2","N",as.vector(t(t)),as.vector(res))

cat("\n- 1x3:\n")
t <- table(expN$Harm01,expN$Harm03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between HH","Obs. 1x3","N",as.vector(t(t)),as.vector(res))

cat("\n- 2x3:\n")
t <- table(expN$Harm02,expN$Harm03)
t <- t[c(2,1),]
t <- t[,c(2,1)]
print(t)
res <- ac1Gwet(t,test=TRUE)
print(res)
w.aux <- w.aux+1
agreement[w.aux,] <- c("Between HH","Obs. 2x3","N",as.vector(t(t)),as.vector(res))

# save agreement measures
agreement <- na.omit(agreement)
agreement$AC1 <- format(round(as.numeric(agreement$AC1),4),
                        digits=4, nsmall=4)
agreement$p <- as.numeric(agreement$p)
agreement$p[agreement$p<0.0001] <- sprintf("%.2e",agreement$p[agreement$p<0.0001])
agreement$p <- as.numeric(agreement$p)
agreement$p[agreement$p>=0.0001] <- format(round(agreement$p[agreement$p>=0.0001],4),
                                          digits=4, nsmall=4)
agreement$p[agreement$p=="0"] <- "<< 0.0001"
openxlsx::write.xlsx(agreement,"Agreement.xlsx")
```

Python scripts

Verifying the Correct Answers

`ScoresMMM.py` uses `Exp2RawData.xlsx` and `AnswerKey.xlsx` to generate `ScoresMMMPy.xlsx`, which contains the count of correct answers for the five trials of the four multimodal models (MMM). These are the only models capable of handling image-based questions tested in Experiment 2.

Pythonscript: ScoresMMM.py

```
import pandas as pd

# Load the data
dt_rawdata = pd.read_excel("Exp2RawData.xlsx")
dt_answerkey = pd.read_excel("AnswerKey.xlsx")
dt_scores = dt_answerkey.copy()

# Define models and trials
models = ["Sonnet 3.5", "Opus", "Sonnet 3", "Haiku"]
trials = sorted(dt_rawdata['Trial'].unique())

# Compute scores
for model in models:
    for trial in trials:
        dt_scores[f"{model}_i{trial}"] = dt_scores['Question'].apply(
            lambda q: int(
                dt_rawdata.loc[
                    (dt_rawdata['Model'] == model) &
                    (dt_rawdata['Trial'] == trial) &
                    (dt_rawdata['Question'] == q),
                    'Answer.model'
                ].eq(dt_scores.loc[dt_scores['Question'] == q,\
                                   'Answer.key'].values[0]).any()
            )
        )

# Save the results
dt_scores.to_excel("ScoresMMM_Py.xlsx", index=False)
```

Descriptive statistics and One-way ANOVA

`Anova.py` uses `ScoresMMMPy.xlsx` to generate `StatisticExperiment2.xlsx`, which contains preprocessed accuracy corresponds to experiment 2 (four models, all questions). Descriptive statistics applied ANOVA one-way to find differences in accuracy.

Post-hoc pairwise comparisons between models were conducted using Tukey's HSD test with confidence level set to 95% .

Pythonscript: Anova.py

```
import pandas as pd
from scipy.stats import f_oneway
from statsmodels.stats.multicomp import pairwise_tukeyhsd

# Load the data
df = pd.read_excel("ScoresMMM_Py.xlsx")

# Automatically identify numeric columns (ignoring "Question",
#                                       "Area", "Image.kind", "Answer.key")
value_vars = [col for col in df.columns\
              if col not in ["Question", "Area", "Image.kind", "Answer.key"]]

# Melt the DataFrame to bring the Model columns into a single column
melted_df = df.melt(id_vars=["Question", "Area", "Image.kind", "Answer.key"],
                    value_vars=value_vars,
                    var_name="Model",
                    value_name="Value")

# Group by model, calculate sums and accuracy
df_result = (
    melted_df.groupby("Model")["Value"]
    .agg(Sum="sum", Accuracy=lambda x: x.sum()/117)
    .reset_index()
)

# Extract base model names (e.g., "Haiku", "Opus", "Sonnet 3.5", "Sonnet 3")
df_result['BaseModel'] = df_result['Model'].str.extract(r'([^_]+)')


def calculate_anova_and_tukey(df, model_column, value_column):
    # Create a list to store the accuracy values for each model
    groups = []
    models = df[model_column].unique()

    # Iterate over each unique model in the model column
    for model in models:
        # Filter the accuracy values of the current model
        # and add them to the list of groups

        values = df[df[model_column] == model][value_column].values
        groups.append(values)

    # Perform ANOVA
    stat, p_value = f_oneway(*groups)
```

⟶

$\longrightarrow$

Pythonscript: Anova.py

```
    # Interpret the results
    if p_value < 0.05:

        # Perform the Tukey test
        tukey = pairwise_tukeyhsd(endog=df[value_column], \
                                  groups=df[model_column], alpha=0.05)
        tukey_result = tukey.summary()
    else:
        tukey_result = None

    return stat, p_value, tukey_result


stat, p_value, tukey_result = calculate_anova_and_tukey(df_result,\
                                            'BaseModel', 'Accuracy')
print(f'F-Statistic: {stat}, p-Value: {p_value}')
if tukey_result:
       # Convert the Tukey result to a pandas DataFrame
    tukey_df = pd.DataFrame(data=tukey_result.data[1:],\
                            columns=tukey_result.data[0])

    # Save the DataFrame to an Excel file
    tukey_df.to_excel('StatisticExperiment2.xlsx', index=False)
```