Supplementary Material: Folic Acid Follow-up

Authors: Dr. Luisa Barbanti, Dr. Matteo Tanadini | Zurich Data Scientists

May 8, 2026

Contents

1 Freezing package versions

```
## (messages are omitted in this chunk)
##
library(groundhog)
packages.to.load <- c('dplyr',
                      'ggplot2',
                      'ggbeeswarm',
                      # 'knitr',
                      'kableExtra',
                      'lubridate',
                      'gtsummary',
                      'tidyverse',
                      'magrittr',
                      'mgcViz',
                      'readxl',
                      'ggpubr', # to extract common legends
                      'ranger',
                      'mgcv',
                      'stringr',
                      'readxl',
                      'cowplot',
                      'tibble',
                      'gridExtra',
                      'grid',
                      'forcats', # to relevel factors
                      'car', # for Levene-test
                      'hypoRF', # for distributional testing
                      'tidyr')
groundhog.library(packages.to.load, date = "2024-10-17")
```

```
Warning: package 'tidyverse' was built under R version 4.4.3
```

```
Warning: package 'hypoRF' was built under R version 4.4.3
```

2 Settings

Global settings:

```
theme_set(theme_bw())

## For the tables created with the package gtsummary, we use the compact theme,
## with font size 11, to make the tables fit to a page.
theme_gtsummary_compact(set_theme = TRUE, font_size = 11)
```

```
Setting theme "Compact"
```

```
## to reset gtsummary themes:
# reset_gtsummary_theme()

## to activate tables generation
maketables <- FALSE
```

3 Getting the data

We load both the wide and long format of the data.

```
d.folic.acid.wide <- readRDS(file = paste0("Prepared_data_and_models/",
                                           "d.folic.acid.wide_final.RDS"))

d.folic.acid.long <- readRDS(file = paste0("Prepared_data_and_models/",
                                            "d.folic.acid.long_final.RDS"))
```

4 Modelling

We want to show clinical equivalence of 5-MTHF-Ca and Folic acid at FUV for body weight, body length and head circumference.

We fitted the following model, for each response separately (here with the example of weight):

Weight(FUV) $= \beta_0 + \beta_1$* Intervention.group $+ \beta_2$* Intervention.duration $+ \beta_3$* Sex $+ \beta_4$* Age $+ \beta_5$Weight(Birth).

The intervention effect in this model is β_1, and this is also what will be displayed later in the equivalence-plots as point estimator, along with the corresponding 95%-CI.

In the following sections, we define some helper functions and fit the models for the three responses for the PP and mITT populations respectively.

4.1 Helper functions

We now write a function that fits the respective model, for a population of choice and a response of choice.

```
#' @param response.to.consider character string, response variable for which we
#' fit the model, either InfantmeanBodyWeight.kg, InfantmeanBodyLength, or
#' InfantmeanHeadCM
#' @param pop.to.consider character string, either "PP" or "mITT" the population
#' in which the model should be fitted
#' @param log.response boolean, should the response be log-transformed
#' should be encoded with the Folic Acid level (=control) first.
fit.lm.func <- function(pop.to.consider,
                        response.to.consider,
                        log.response){

  d.tmp1 <- d.folic.acid.long %>%
    mutate(
      any.resp.missing = rowSums(is.na(select(.,
                                              InfantmeanBodyWeight,
                                              InfantmeanBodyLength,
                                              InfantmeanHeadCM))) > 0) %>%
    filter(any.resp.missing == FALSE) %>%
    ## compute equivalence at FUV
    filter(Visit.fac == "FUV") %>%
    ## we filter to only consider intervention and control
    filter(Treatmentgroup.fac.renamed.reord %in% c("5-MTHF-Ca", "Folic acid")) %>%
    ## drop unused levels
    droplevels()
```

```r
## We filter according to the PP or mITT population
if(pop.to.consider == "PP"){
  d.tmp1 <- d.tmp1 %>%
    filter(PPPopulation.fac == "yes")
}else if(pop.to.consider == "mITT"){
  d.tmp1 <- d.tmp1 %>%
    filter(ITTPopulation.fac == "yes")
}else{
  return(paste0("Please select either PP or mITT as value to the
                argument pop.to.consider."))
}

## check missing values in the covariates
any(is.na(d.tmp1 %>% select(Treatmentgroup.fac.renamed.reord,
                            Intervention.duration,
                            ChildSex.fac,
                            Birthweight_gram)))

if(!(response.to.consider %in% c("InfantmeanBodyWeight", ## agreed to show grams
                                 "InfantmeanBodyLength",
                                 "InfantmeanHeadCM"))){
  return(paste0("Please select either InfantmeanBodyWeight,
                InfantmeanBodyLength or InfantmeanHeadCM as value to the
                argument response.to.consider."))

}


## We relevel the intervention variable, such that the coefficient means Folic acid -
## 5-MTHF-Ca
d.tmp1 <-  d.tmp1 %>%
  mutate(Treatmentgroup.fac.renamed.reord =
           fct_relevel(d.tmp1$Treatmentgroup.fac.renamed.reord,
                       "Folic acid"))

if(log.response){
  fitted.lm <- lm(log(get(response.to.consider)) ~
                    Treatmentgroup.fac.renamed.reord +
                    Intervention.duration +
                    ChildSex.fac +
                    Age +
                    Birthweight_gram,
                  data = d.tmp1)
}else{
  fitted.lm <- lm(get(response.to.consider) ~
                    Treatmentgroup.fac.renamed.reord +
                    Intervention.duration +
                    ChildSex.fac +
                    Age +
                    Birthweight_gram,
                  data = d.tmp1)
}
```

```
  return(list(fitted.lm = fitted.lm,
              data.used = d.tmp1,
              pop.to.consider = pop.to.consider,
              response.to.consider = response.to.consider))
}
```

We create a function to plot the equivalence plot.

```
#' @param fitted.model A fitted model output from the function fit.lm.func
plot.equivalence.func <- function(fitted.model){

  fitted.model.mod <- fitted.model$fitted.lm
  fitted.model.data <- fitted.model$data.used
  response.to.consider <- fitted.model$response.to.consider
  pop.to.consider <- fitted.model$pop.to.consider

  SD.equiv.bounds <- fitted.model.data %>%
    filter(Treatmentgroup.fac.renamed.reord == "Folic acid") %>%
    pull(response.to.consider) %>%
    sd()

  point.estimate <- coef(fitted.model.mod)["Treatmentgroup.fac.renamed.reord5-MTHF-Ca"]
  confint.lower <- confint(fitted.model.mod)["Treatmentgroup.fac.renamed.reord5-MTHF-Ca", 1]
  confint.upper <- confint(fitted.model.mod)["Treatmentgroup.fac.renamed.reord5-MTHF-Ca", 2]

  # Define the equivalence range
  shade.min <- -0.5 * SD.equiv.bounds
  shade.max <- 0.5 * SD.equiv.bounds

  ## Create the plot
  gg.plot.euquiv <-
    ggplot() +
    ## Shaded area for equivalence range
    geom_rect(aes(xmin = shade.min, xmax = shade.max,
                  ymin = -Inf, ymax = Inf),
              fill = "grey", alpha = 0.3) +
    ## Confidence interval line
    geom_segment(aes(x = confint.lower, xend = confint.upper,
                     y = 0, yend = 0),
                 size = 1) +
    ## Point estimate
    geom_point(aes(x = point.estimate, y = 0), size = 3) +
    # Dashed line at x = 0
    geom_vline(xintercept = 0, linetype = "dashed") +
    ## Labels and theme adjustments
    xlab("Intervention effect") +
    ylab("") +
    theme_minimal() +
    theme(axis.text.y = element_blank(),
          axis.ticks.y = element_blank(),
          panel.grid.major.y = element_blank(),
          panel.grid.minor.y = element_blank())

  ## We add the title to the plot and fix the x-axis range
```

```
  # if(pop.to.consider == "PP"){
  #   title.tmp <- paste0("PP: ")
  # }else if(pop.to.consider == "mITT"){
  #   title.tmp <- paste0("mITT: ")
  # }

  if(response.to.consider == "InfantmeanBodyWeight"){
    gg.plot.euquiv <- gg.plot.euquiv +
      ggtitle(paste0("Mean body weight, equivalence range")) +
      coord_cartesian(xlim = c(-1000, 1000)) +
      theme( plot.title = element_text(size = 10))
  }else if(response.to.consider == "InfantmeanBodyLength"){
    gg.plot.euquiv <- gg.plot.euquiv +
      ggtitle(paste0("Mean body length, equivalence range")) +
      coord_cartesian(xlim = c(-2, 2)) +
      theme( plot.title = element_text(size = 10))
  }else if(response.to.consider == "InfantmeanHeadCM"){
    gg.plot.euquiv <- gg.plot.euquiv +
      ggtitle(paste0("Mean head circumference, equivalence range")) +
      coord_cartesian(xlim = c(-1, 1)) +
      theme(plot.title = element_text(size = 10))
  }

  return(gg.plot.euquiv)
}
```

We create a function to extract the model parameters, their confidence intervals and the equivalence ranges.

```
get.equivalence.data <- function(fitted.model){

  fitted.model.mod <- fitted.model$fitted.lm
  fitted.model.data <- fitted.model$data.used
  response.to.consider <- fitted.model$response.to.consider
  pop.to.consider <- fitted.model$pop.to.consider

  SD.equiv.bounds <- fitted.model.data %>%
    filter(Treatmentgroup.fac.renamed.reord == "Folic acid") %>%
    pull(response.to.consider) %>%
    sd()

  point.estimate <- coef(fitted.model.mod)["Treatmentgroup.fac.renamed.reord5-MTHF-Ca"]
  confint.lower <- confint(fitted.model.mod)["Treatmentgroup.fac.renamed.reord5-MTHF-Ca", 1]
  confint.upper <- confint(fitted.model.mod)["Treatmentgroup.fac.renamed.reord5-MTHF-Ca", 2]

  # Define the equivalence range
  shade.min <- -0.5 * SD.equiv.bounds
  shade.max <- 0.5 * SD.equiv.bounds

  return(data.frame(point.estimate = point.estimate,
                    confint.lower = confint.lower,
                    confint.upper = confint.upper,
                    shade.min = shade.min,
                    shade.max = shade.max))
}
```

For the model checking part, we need the follwing proto-plot functions.

```
## create plot fitted values against duration
#' @param fitted.model A fitted model output from the function fit.lm.func
#' @param xaxis.var character string, either Intervention.duration or Birthweigth_gram
#' the variable against which we plot the response.to.consider on the x-axis
#' @param log.response boolean, should the response be log-transformed
gg.proto.fitted <- function(fitted.model,
                            xaxis.var,
                            log.response){

  res.mod <- fitted.model$fitted.lm
  res.data <- fitted.model$data.used
  res.response <- fitted.model$response.to.consider
  pop.to.consider <- fitted.model$pop.to.consider

  gg.fitted <- res.data %>%
    ggplot() +
    aes(color = Treatmentgroup.fac.renamed.reord) +
    geom_point(alpha = 1) +
    facet_wrap(~ ChildSex.fac) +
    theme(plot.title = element_text(size = 9))


  if(log.response){
    gg.fitted <- gg.fitted +
      aes(x = get(xaxis.var), y = exp(fitted(res.mod))) +
      # theme(legend.position="none") +
      ggtitle(paste0(pop.to.consider, ": Exp-Fitted values ",
                     res.response, " ~ ", xaxis.var))
  }else{
    gg.fitted <- gg.fitted +
      aes(x = get(xaxis.var), y = fitted(res.mod)) +
      # theme(legend.position="none") +
      ggtitle(paste0(pop.to.consider, ": Fitted values ",
                     res.response, " ~ ", xaxis.var))
  }

  return(gg.fitted)
}


## Tukey-Anscombe plot with added smoother and line per patient and colors
## per Axis

gg.proto.TAorSL <- function(fitted.model, plot.TA, log.response){

  res.mod <- fitted.model$fitted.lm
  res.data <- fitted.model$data.used
  res.response <- fitted.model$response.to.consider
  pop.to.consider <- fitted.model$pop.to.consider

  gg.TAorSL <- res.data %>%
    ggplot() +
```

```
    aes(color = Treatmentgroup.fac.renamed.reord) +
    geom_point(alpha = 1) +
    geom_smooth(color = "blue", method = "loess")

  ## TA-plot
  if(plot.TA == TRUE){

    gg.TAorSL <- gg.TAorSL +
      theme(plot.title = element_text(size = 10))

    if(log.response){
      gg.TAorSL <- gg.TAorSL +
        aes(x = fitted(res.mod), y = exp(resid(res.mod))) +
        ggtitle(paste0(pop.to.consider, ": TA-plot ",
                       res.response)) +
        geom_hline(yintercept = 1, linetype = "dashed")

    }else{
      gg.TAorSL <- gg.TAorSL +
        aes(x = fitted(res.mod), y = resid(res.mod)) +
        ggtitle(paste0(pop.to.consider, ": TA-plot ",
                       res.response)) +
        geom_hline(yintercept = 0, linetype = "dashed")
    }

    ## Scale-Location plot
  }else{

    gg.TAorSL <- gg.TAorSL +
      theme(plot.title = element_text(size = 9))

    if(log.response){
      gg.TAorSL <- gg.TAorSL +
        aes(x = fitted(res.mod),  y = sqrt(abs(exp(resid(res.mod)))))+
        ggtitle(paste0(pop.to.consider, ": SL-plot ",
                       res.response))

    }else{
      gg.TAorSL <- gg.TAorSL +
        aes(x = fitted(res.mod),  y = sqrt(abs(resid(res.mod))))+
        ggtitle(paste0(pop.to.consider, ": SL-plot ",
                       res.response))

    }

  }

  return(gg.TAorSL)
}
```

This function allows us to look at Tukey-Anscombe, SL and QQ plots.

```
#' @param pop.to.consider character string, either "PP" or "mITT" the population
#' in which the model should be fitted
```

```r
#' @param plot.to.create character string, either "Fitted" for fitted values plot,
#' "TA_SL_QQ" for Tukey-Anscombe plot and Scale-Location plot.
#' @param log.response boolean, should the response be log-transformed
plot.fitted.or.TA_SL_QQ.func <- function(pop.to.consider,
                                         plot.to.create,
                                         log.response){

  ## Fit model body weight
  res.weightg <- fit.lm.func(pop.to.consider = pop.to.consider,
                             response.to.consider = "InfantmeanBodyWeight",
                             log.response = log.response)
  ## Fit model body length
  res.length <- fit.lm.func(pop.to.consider = pop.to.consider,
                            response.to.consider = "InfantmeanBodyLength",
                            log.response = log.response)
  ## Fit model head cm
  res.headcm <- fit.lm.func(pop.to.consider = pop.to.consider,
                            response.to.consider = "InfantmeanHeadCM",
                            log.response = log.response)

  if(plot.to.create == "Fitted"){

    gg.fitted.weight.duration <- gg.proto.fitted(fitted.model = res.weightg,
                                                 xaxis.var = "Intervention.duration",
                                                 log.response = log.response) +
      theme(legend.position = "none")

    gg.fitted.weight.birthweight <- gg.proto.fitted(fitted.model = res.weightg,
                                                    xaxis.var = "Birthweight_gram",
                                                    log.response = log.response) +
      theme(legend.position = "none")
    gg.fitted.length.duration <- gg.proto.fitted(fitted.model = res.length,
                                                 xaxis.var = "Intervention.duration",
                                                 log.response = log.response)+
      theme(legend.position = "none")
    gg.fitted.length.birthweight <- gg.proto.fitted(fitted.model = res.length,
                                                    xaxis.var = "Birthweight_gram",
                                                    log.response = log.response)+
      theme(legend.position = "none")
    gg.fitted.headcm.duration <- gg.proto.fitted(fitted.model = res.headcm,
                                                 xaxis.var = "Intervention.duration",
                                                 log.response = log.response)+
      theme(legend.position = "none")
    gg.fitted.headcm.birthweight <- gg.proto.fitted(fitted.model = res.headcm,
                                                    xaxis.var = "Birthweight_gram",
                                                    log.response = log.response)+
      theme(legend.position = "none")

    # Extract the legend from one of the plots
    test.plot <-  gg.proto.fitted(fitted.model = res.weightg,
                                  xaxis.var = "Intervention.duration",
                                  log.response = log.response) +
      theme(legend.position = "bottom")
```

```
  common.legend <- ggpubr::get_legend(test.plot)

  ## Arrange plots to one plot
  plot.grid <- plot_grid(
    gg.fitted.weight.duration,
    gg.fitted.weight.birthweight,
    gg.fitted.length.duration,
    gg.fitted.length.birthweight,
    gg.fitted.headcm.duration,
    gg.fitted.headcm.birthweight,
    ncol = 2,
    align = "v"
  )

  # Combine the grid and the common legend
  final.plot <- plot_grid(
    plot.grid,
    common.legend,
    ncol = 1,
    rel_heights = c(0.9, 0.1) # Allocate space for the legend
  )

  caption_text <- paste0("Fitted values",
                         " in population ", pop.to.consider,
                         " using log on the response = ", log.response, ".")

  # Add the caption below the final plot
  final.plot.with.caption <- ggdraw() +
    draw_plot(final.plot, x = 0, y = 0.05, width = 1, height = 0.95) +
    draw_label(caption_text, x = 0.5, y = 0.02, hjust = 0.5, size = 14)

  return(final.plot.with.caption)

}else if(plot.to.create == "TA_SL_QQ"){

  gg.TA.weight <- gg.proto.TAorSL(fitted.model = res.weightg,
                                  plot.TA = TRUE,
                                  log.response = log.response) +
    theme(legend.position = "none")
  gg.SL.weight <- gg.proto.TAorSL(fitted.model = res.weightg,
                                  plot.TA = FALSE,
                                  log.response = log.response) +
    theme(legend.position = "none")

  gg.qq.weight <- ggplot() +
    stat_qq(aes(sample = resid(res.weightg$fitted.lm))) +
    stat_qq_line(aes(sample = resid(res.weightg$fitted.lm))) +
    ggtitle("QQ-plot: Weight")+
    theme(plot.title = element_text(size = 9))

  gg.TA.length <- gg.proto.TAorSL(fitted.model = res.length,
                                  plot.TA = TRUE,
                                  log.response = log.response) +
```

```
  theme(legend.position = "none")
gg.SL.length <- gg.proto.TAorSL(fitted.model = res.length,
                                plot.TA = FALSE,
                                log.response = log.response) +
  theme(legend.position = "none")

gg.qq.length <- ggplot() +
  stat_qq(aes(sample = resid(res.length$fitted.lm))) +
  stat_qq_line(aes(sample = resid(res.length$fitted.lm))) +
  ggtitle("QQ-plot: Length")+
  theme(plot.title = element_text(size = 9))

gg.TA.headcm <- gg.proto.TAorSL(fitted.model = res.headcm,
                                plot.TA = TRUE,
                                log.response = log.response) +
  theme(legend.position = "none")
gg.SL.headcm <- gg.proto.TAorSL(fitted.model = res.headcm,
                                plot.TA = FALSE,
                                log.response = log.response) +
  theme(legend.position = "none")

gg.qq.headcm <- ggplot() +
  stat_qq(aes(sample = resid(res.headcm$fitted.lm))) +
  stat_qq_line(aes(sample = resid(res.headcm$fitted.lm))) +
  ggtitle("QQ-plot: Head CM") +
  theme(plot.title = element_text(size = 9))

# Extract the common legend from one of the plots
common.legend <- ggpubr::get_legend(
  gg.proto.TAorSL(fitted.model = res.weightg,
                  plot.TA = TRUE,
                  log.response = log.response) +
    theme(legend.position = "bottom")
)

# Arrange the subplots into a grid
plot.grid <- plot_grid(
  gg.TA.weight, gg.SL.weight, gg.qq.weight,
  gg.TA.length, gg.SL.length, gg.qq.length,
  gg.TA.headcm, gg.SL.headcm, gg.qq.headcm,
  ncol = 3,
  align = "v"
)

# Combine the grid and the common legend
final.plot <- plot_grid(
  plot.grid,
  common.legend,
  ncol = 1,
  rel_heights = c(0.9, 0.1) # Allocate space for the legend
)

caption_text <- paste0("Residual analysis",
```

```
                            " in population ", pop.to.consider,
                            " using log on the response = ", log.response, ".")

    # Add the caption below the final plot
    final.plot.with.caption <- ggdraw() +
      draw_plot(final.plot, x = 0, y = 0.05, width = 1, height = 0.95) +
      draw_label(caption_text, x = 0.5, y = 0.02, hjust = 0.5, size = 14)

    return(final.plot.with.caption)
  }
}
```

4.2 PP-Population

In this section we present the results for the PP population. We set the parameters.

```
## We do not log-transform the response
log.response <- FALSE
pop.to.consider <- "PP"
```

4.2.1 InfantmeanBodyWeight

For body weight, we first fit the model, then save the results and inspect the output of the model. Note that here we model body weight in grams.

```
options(width = 300)
res.PP.weightg <- fit.lm.func(pop.to.consider = pop.to.consider,
                              response.to.consider = "InfantmeanBodyWeight",
                              log.response = log.response)

res.PP.weightg.mod <- res.PP.weightg$fitted.lm
res.PP.weightg.data <- res.PP.weightg$data.used
summary(res.PP.weightg.mod)
```

```
Call:
lm(formula = get(response.to.consider) ~ Treatmentgroup.fac.renamed.reord +
    Intervention.duration + ChildSex.fac + Age + Birthweight_gram,
    data = d.tmp1)

Residuals:
   Min     1Q Median     3Q    Max
 -2197   -734    -71    610   3329

Coefficients:
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                               3726.989   5058.328    0.74  0.46248
Treatmentgroup.fac.renamed.reord5-MTHF-Ca    85.997    176.082    0.49  0.62604
Intervention.duration                         3.529     23.217    0.15  0.87940
ChildSex.facM                               453.523    173.542    2.61  0.00995 **
Age                                           8.059     13.026    0.62  0.53711
Birthweight_gram                              0.854      0.215    3.97  0.00012 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

```
Residual standard error: 1030 on 139 degrees of freedom
Multiple R-squared:  0.158, Adjusted R-squared:  0.127
F-statistic:  5.2 on 5 and 139 DF,  p-value: 0.000208
```

```
confint(res.PP.weightg.mod)
```

```
                                          2.5 %  97.5 %
(Intercept)                           -6274.22 13728.2
Treatmentgroup.fac.renamed.reord5-MTHF-Ca  -262.15   434.1
Intervention.duration                   -42.37    49.4
ChildSex.facM                           110.40   796.6
Age                                     -17.69    33.8
Birthweight_gram                          0.43     1.3
```

4.2.2 InfantmeanBodyLength

For body length, we first fit the model, then save the results and inspect the output of the model.

```
res.PP.length <- fit.lm.func(pop.to.consider = pop.to.consider,
                             response.to.consider = "InfantmeanBodyLength",
                             log.response = log.response)

res.PP.length.mod <- res.PP.length$fitted.lm
res.PP.length.data <- res.PP.length$data.used
summary(res.PP.length.mod)
```

```
Call:
lm(formula = get(response.to.consider) ~ Treatmentgroup.fac.renamed.reord +
    Intervention.duration + ChildSex.fac + Age + Birthweight_gram,
    data = d.tmp1)

Residuals:
   Min     1Q Median     3Q    Max
 -6.47  -1.42   0.07   1.53   6.59

Coefficients:
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                             74.053013  10.712457    6.91  1.6e-10 ***
Treatmentgroup.fac.renamed.reord5-MTHF-Ca -0.312210   0.372904   -0.84    0.404
Intervention.duration                   -0.034285   0.049168   -0.70    0.487
ChildSex.facM                            1.633436   0.367526    4.44  1.8e-05 ***
Age                                      0.003652   0.027586    0.13    0.895
Birthweight_gram                         0.001300   0.000456    2.85    0.005 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.2 on 139 degrees of freedom
Multiple R-squared:  0.186, Adjusted R-squared:  0.157
F-statistic: 6.37 on 5 and 139 DF,  p-value: 2.36e-05
```

```
confint(res.PP.length.mod)
```

```
                                          2.5 %  97.5 %
(Intercept)                             52.8726 95.2334
Treatmentgroup.fac.renamed.reord5-MTHF-Ca -1.0495  0.4251
```

```
Intervention.duration                      -0.1315  0.0629
ChildSex.facM                               0.9068  2.3601
Age                                        -0.0509  0.0582
Birthweight_gram                            0.0004  0.0022
```

4.2.3 InfantmeanHeadCM

For head circumference, we first fit the model, then save the results and inspect the output of the model.

```
res.PP.headcm <- fit.lm.func(pop.to.consider = pop.to.consider,
                             response.to.consider = "InfantmeanHeadCM",
                             log.response = log.response)

res.PP.headcm.mod <- res.PP.headcm$fitted.lm
res.PP.headcm.data <- res.PP.headcm$data.used
summary(res.PP.headcm.mod)
```

```
Call:
lm(formula = get(response.to.consider) ~ Treatmentgroup.fac.renamed.reord +
    Intervention.duration + ChildSex.fac + Age + Birthweight_gram,
    data = d.tmp1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.9469 -0.9247  0.0665  0.7898  2.3119

Coefficients:
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                               4.07e+01   5.57e+00    7.30  2.0e-11 ***
Treatmentgroup.fac.renamed.reord5-MTHF-Ca 2.14e-01   1.94e-01    1.10     0.27
Intervention.duration                     5.09e-03   2.56e-02    0.20     0.84
ChildSex.facM                             1.23e+00   1.91e-01    6.44  1.8e-09 ***
Age                                       2.84e-03   1.44e-02    0.20     0.84
Birthweight_gram                          9.68e-04   2.37e-04    4.08  7.6e-05 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 1.1 on 139 degrees of freedom
Multiple R-squared:  0.316, Adjusted R-squared:  0.291
F-statistic: 12.8 on 5 and 139 DF,  p-value: 2.89e-10
```

```
confint(res.PP.headcm.mod)
```

```
                                            2.5 %  97.5 %
(Intercept)                               29.6766 51.7178
Treatmentgroup.fac.renamed.reord5-MTHF-Ca -0.1698  0.5974
Intervention.duration                     -0.0455  0.0557
ChildSex.facM                              0.8534  1.6096
Age                                       -0.0255  0.0312
Birthweight_gram                           0.0005  0.0014
```

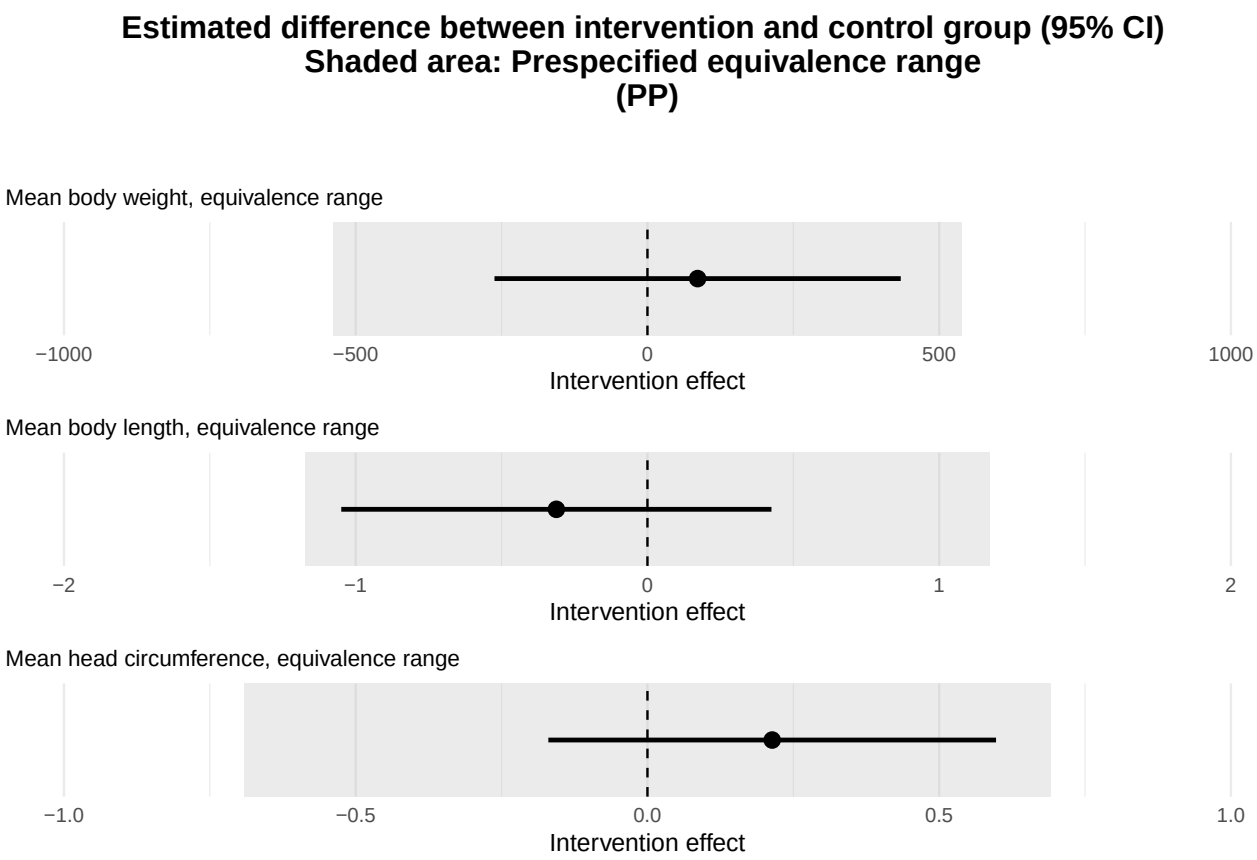
4.2.4 Equivalence Plots

We plot the results showing equivalence of intervention (5-MTHF-Ca) and control (Folic acid) for the three body measurements.

```
plot.PP.weightg <- plot.equivalence.func(res.PP.weightg)
```

```
Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use ‘linewidth‘ instead.
This warning is displayed once every 8 hours.
Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was generated.
```

```
plot.PP.length <- plot.equivalence.func(res.PP.length)
plot.PP.headcm <- plot.equivalence.func(res.PP.headcm)

title <- ggdraw() +
  draw_label(paste0("Estimated difference between intervention",
                    " and control group (95% CI) \n",
                    "Shaded area: Prespecified equivalence range \n(",
                    pop.to.consider, ")"),
             fontface = 'bold')

final.plot <- plot_grid(
  title,
  plot.PP.weightg,
  plot.PP.length,
  plot.PP.headcm,
  ncol = 1,
  align = "v")

# Add the caption below the final plot
ggdraw() +
  draw_plot(final.plot, x = 0, y = 0.05, width = 1, height = 0.95)
```



For safety reasons, we remove the two parameters.

```
rm(log.response)
rm(pop.to.consider)
```

4.2.5 Table with results

We collect the point estimates and the corresponding 95% confidence intervals (CI) + equivalence ranges (ER) in a table.

```
data.PP.weightg <- get.equivalence.data(res.PP.weightg)
data.PP.length <- get.equivalence.data(res.PP.length)
data.PP.headcm <- get.equivalence.data(res.PP.headcm)

rbind(data.PP.weightg,
      data.PP.length,
      data.PP.headcm) %>%
  as_tibble() %>%
  kable(caption = "Estimates, confidence intervals and equivalence ranges for PP population.",
        col.names = c("Point estimate", "CI lower bound", "CI upper bound",
                      "ER lower bound", "ER upper bound"))
```

Table 1: Estimates, confidence intervals and equivalence ranges for PP population.

Point estimate	CI lower bound	CI upper bound	ER lower bound	ER upper bound
86.00	-262.15	434.14	-538.39	538.39

Point estimate	CI lower bound	CI upper bound	ER lower bound	ER upper bound
-0.31	-1.05	0.43	-1.17	1.17
0.21	-0.17	0.60	-0.69	0.69

4.2.6 Model checking

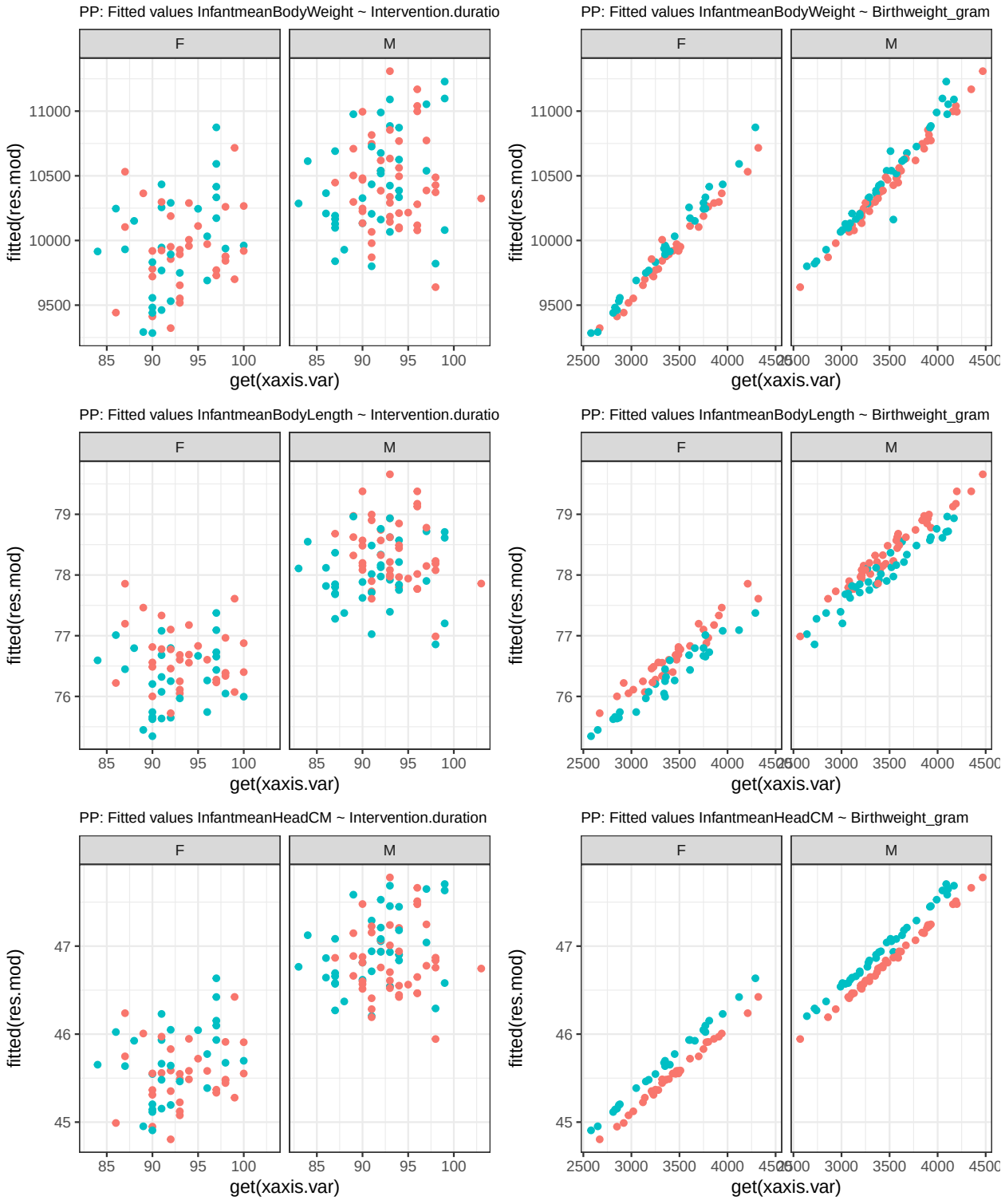
Here we perform the usual formal model diagnostics ("residual analysis"). All of the following plots show that the models fit the data well, and that there are no brutal outliers.

To see whether we fitted a meaningful model, it is important to look at the fitted values and perform residual analysis for model assessment.

Apart from the fitted values, displayed in two ways, we create for each model three residual plots:

- TA-plot: We first check the assumption that our model equation is correct. This is the case if the expected value of the residuals is zero. We look at the Tukey-Anscombe plot (TA-plot) that shows residuals vs fitted values and add a loess-smoother. If in the Tukey-Anscombe plot (TA plot) we see that the smoother is approximately a flat line on zero, we might deduce that there is no systematic structure left in the errros.
- SL-plot: To check the assumption of homoscedasticity of the residuals, we look at the Scale-Location plot that shows square-root of the absolute value of the residuals vs fitted values and we add a loess-smoother.
- QQ-plot: We also check whether the residuals are normally distributed with a QQ-plot.

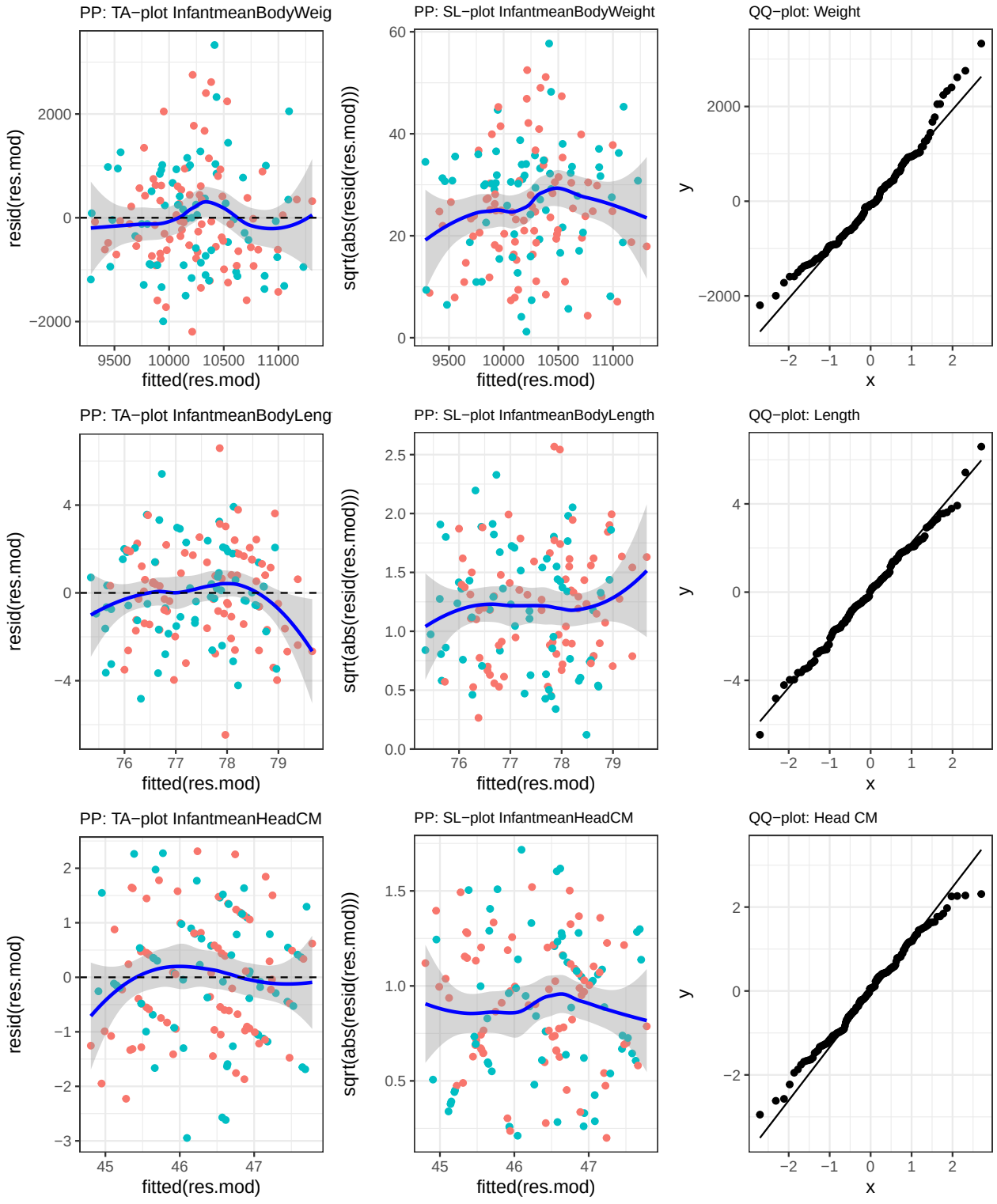
```
plot.fitted.or.TA_SL_QQ.func(pop.to.consider = "PP",
                             plot.to.create = "Fitted",
                             log.response = FALSE)
```



Fitted values in population PP using log on the response = FALSE.

```
plot.fitted.or.TA_SL_QQ.func(pop.to.consider = "PP",
                             plot.to.create = "TA_SL_QQ",
                             log.response = FALSE)
```

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



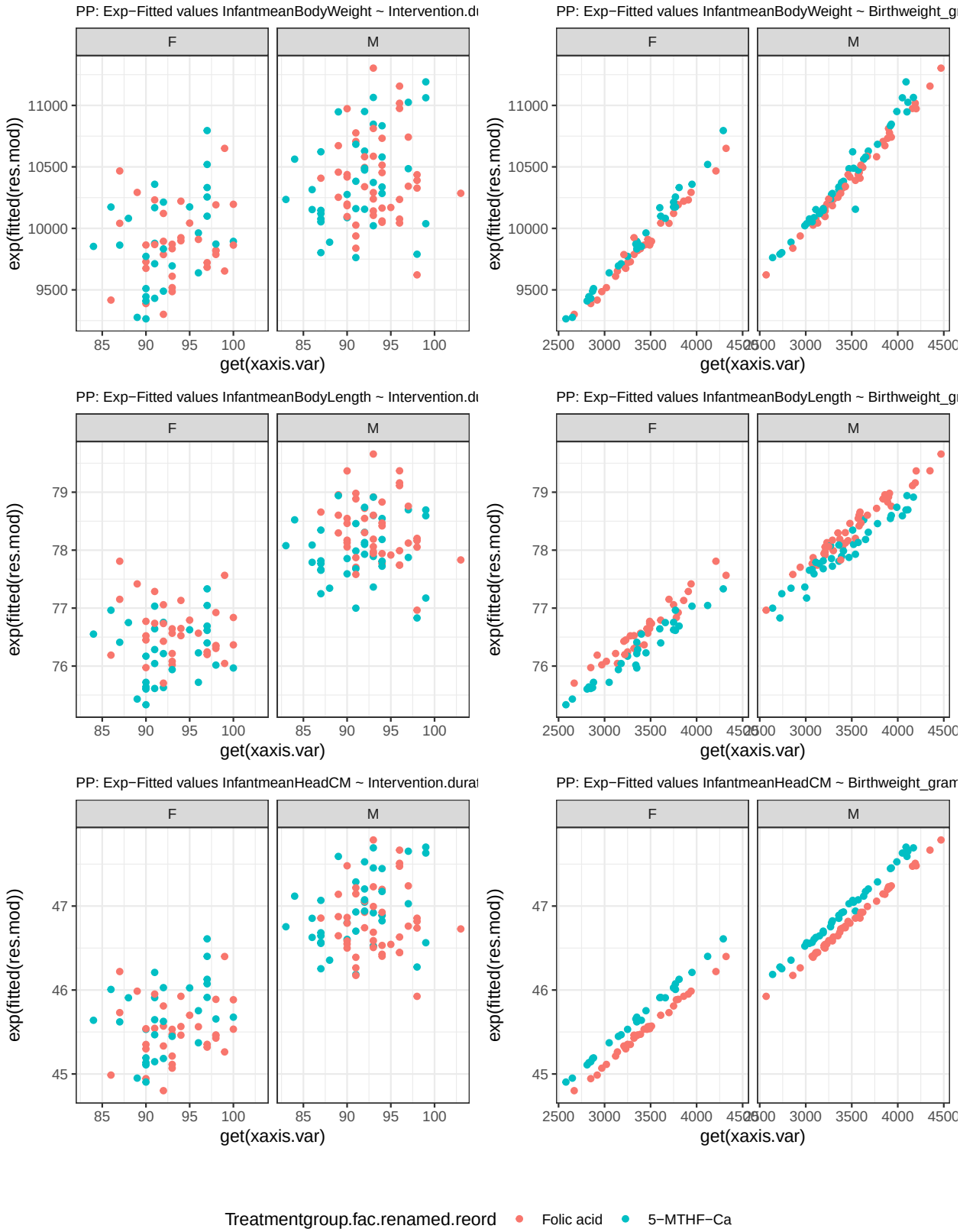
Residual analysis in population PP using log on the response = FALSE.

All the plots look fine for the three responses.

4.2.7 Showing that log-transformation is not necessary

In this section we show briefly that log-transforming the responses is not necessary. This does not lead to an improved model fit nor improves model diagnostics.

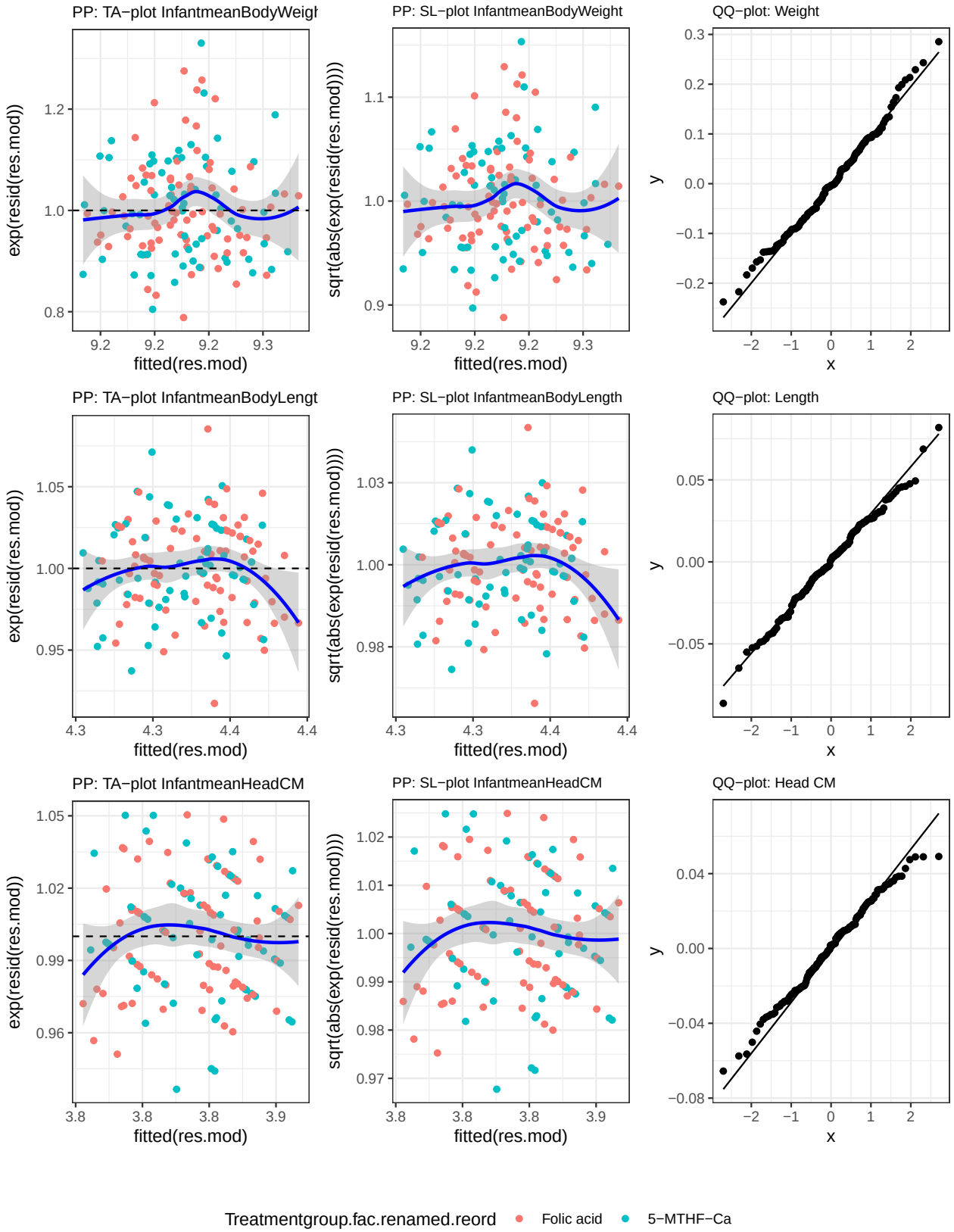
```
plot.fitted.or.TA_SL_QQ.func(pop.to.consider = "PP",
                             plot.to.create = "Fitted",
                             log.response = TRUE)
```



Fitted values in population PP using log on the response = TRUE.

```
plot.fitted.or.TA_SL_QQ.func(pop.to.consider = "PP",
                             plot.to.create = "TA_SL_QQ",
                             log.response = TRUE)
```

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



Residual analysis in population PP using log on the response = TRUE.

4.3 mITT-Population

We set the parameters.

```
## We do not log-transform the response
log.response <- FALSE
pop.to.consider <- "mITT"
```

4.3.1 InfantmeanBodyWeight

For body weight, we first fit the model, then save the results and inspect the output of the model. Note that here we model body weight in grams.

```
options(width = 300)
res.mITT.weightg <- fit.lm.func(pop.to.consider = pop.to.consider,
                                response.to.consider = "InfantmeanBodyWeight",
                                log.response = log.response)

res.mITT.weightg.mod <- res.mITT.weightg$fitted.lm
res.mITT.weightg.data <- res.mITT.weightg$data.used
summary(res.mITT.weightg.mod)
```

```
Call:
lm(formula = get(response.to.consider) ~ Treatmentgroup.fac.renamed.reord +
    Intervention.duration + ChildSex.fac + Age + Birthweight_gram,
    data = d.tmp1)

Residuals:
   Min     1Q Median     3Q    Max
 -2217   -702    -60    613   3401

Coefficients:
                                        Estimate Std. Error t value Pr(>|t|)
(Intercept)                             3456.792   4416.852    0.78   0.4349
Treatmentgroup.fac.renamed.reord5-MTHF-Ca   78.604    149.295    0.53   0.5992
Intervention.duration                     -17.850     15.277   -1.17   0.2443
ChildSex.facM                             462.977    151.068    3.06   0.0025 **
Age                                        14.933     11.620    1.29   0.2005
Birthweight_gram                            0.780      0.186    4.20  4.3e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 984 on 171 degrees of freedom
Multiple R-squared:  0.166, Adjusted R-squared:  0.142
F-statistic: 6.81 on 5 and 171 DF,  p-value: 8.14e-06
```

```
confint(res.mITT.weightg.mod)
```

```
                                          2.5 %  97.5 %
(Intercept)                             -5261.78 12175.4
Treatmentgroup.fac.renamed.reord5-MTHF-Ca  -216.09   373.3
Intervention.duration                       -48.00    12.3
ChildSex.facM                               164.78   761.2
Age                                          -8.00    37.9
Birthweight_gram                              0.41     1.1
```

4.3.2 InfantmeanBodyLength

For body length, we first fit the model, then save the results and inspect the output of the model.

```
res.mITT.length <- fit.lm.func(pop.to.consider = pop.to.consider,
                               response.to.consider = "InfantmeanBodyLength",
                               log.response = log.response)

res.mITT.length.mod <- res.mITT.length$fitted.lm
res.mITT.length.data <- res.mITT.length$data.used
summary(res.mITT.length.mod)
```

```
Call:
lm(formula = get(response.to.consider) ~ Treatmentgroup.fac.renamed.reord +
    Intervention.duration + ChildSex.fac + Age + Birthweight_gram,
    data = d.tmp1)

Residuals:
   Min     1Q Median     3Q    Max
-6.415 -1.438 -0.018  1.618  6.874

Coefficients:
                                        Estimate Std. Error t value Pr(>|t|)
(Intercept)                             69.94927    9.75380    7.17  2.1e-11 ***
Treatmentgroup.fac.renamed.reord5-MTHF-Ca -0.29806    0.32969   -0.90    0.367
Intervention.duration                   -0.02813    0.03374   -0.83    0.406
ChildSex.facM                            1.47648    0.33360    4.43  1.7e-05 ***
Age                                      0.01564    0.02566    0.61    0.543
Birthweight_gram                         0.00105    0.00041    2.57    0.011 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.2 on 171 degrees of freedom
Multiple R-squared:  0.155, Adjusted R-squared:  0.13
F-statistic: 6.28 on 5 and 171 DF,  p-value: 2.24e-05
```

```
confint(res.mITT.length.mod)
```

```
                                          2.5 %  97.5 %
(Intercept)                             50.69592 89.2026
Treatmentgroup.fac.renamed.reord5-MTHF-Ca -0.94884  0.3527
Intervention.duration                   -0.09472  0.0385
ChildSex.facM                            0.81796  2.1350
Age                                     -0.03501  0.0663
Birthweight_gram                         0.00025  0.0019
```

4.3.3 InfantmeanHeadCM

For head circumference, we first fit the model, then save the results and inspect the output of the model.

```
res.mITT.headcm <- fit.lm.func(pop.to.consider = pop.to.consider,
                               response.to.consider = "InfantmeanHeadCM",
                               log.response = log.response)

res.mITT.headcm.mod <- res.mITT.headcm$fitted.lm
res.mITT.headcm.data <- res.mITT.headcm$data.used
```

```
summary(res.mITT.headcm.mod)
```

```
Call:
lm(formula = get(response.to.consider) ~ Treatmentgroup.fac.renamed.reord +
    Intervention.duration + ChildSex.fac + Age + Birthweight_gram,
    data = d.tmp1)

Residuals:
    Min      1Q  Median      3Q     Max
-3.0099 -0.8838  0.0104  0.8152  2.5283

Coefficients:
                                        Estimate Std. Error t value Pr(>|t|)
(Intercept)                             3.99e+01   5.12e+00    7.78  6.5e-13 ***
Treatmentgroup.fac.renamed.reord5-MTHF-Ca 2.15e-01   1.73e-01    1.24     0.22
Intervention.duration                   5.17e-03   1.77e-02    0.29     0.77
ChildSex.facM                           1.15e+00   1.75e-01    6.54  6.9e-10 ***
Age                                     5.24e-03   1.35e-02    0.39     0.70
Birthweight_gram                        9.66e-04   2.16e-04    4.48  1.3e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.1 on 171 degrees of freedom
Multiple R-squared:  0.303, Adjusted R-squared:  0.282
F-statistic: 14.9 on 5 and 171 DF,  p-value: 4.22e-12
```

```
confint(res.mITT.headcm.mod)
```

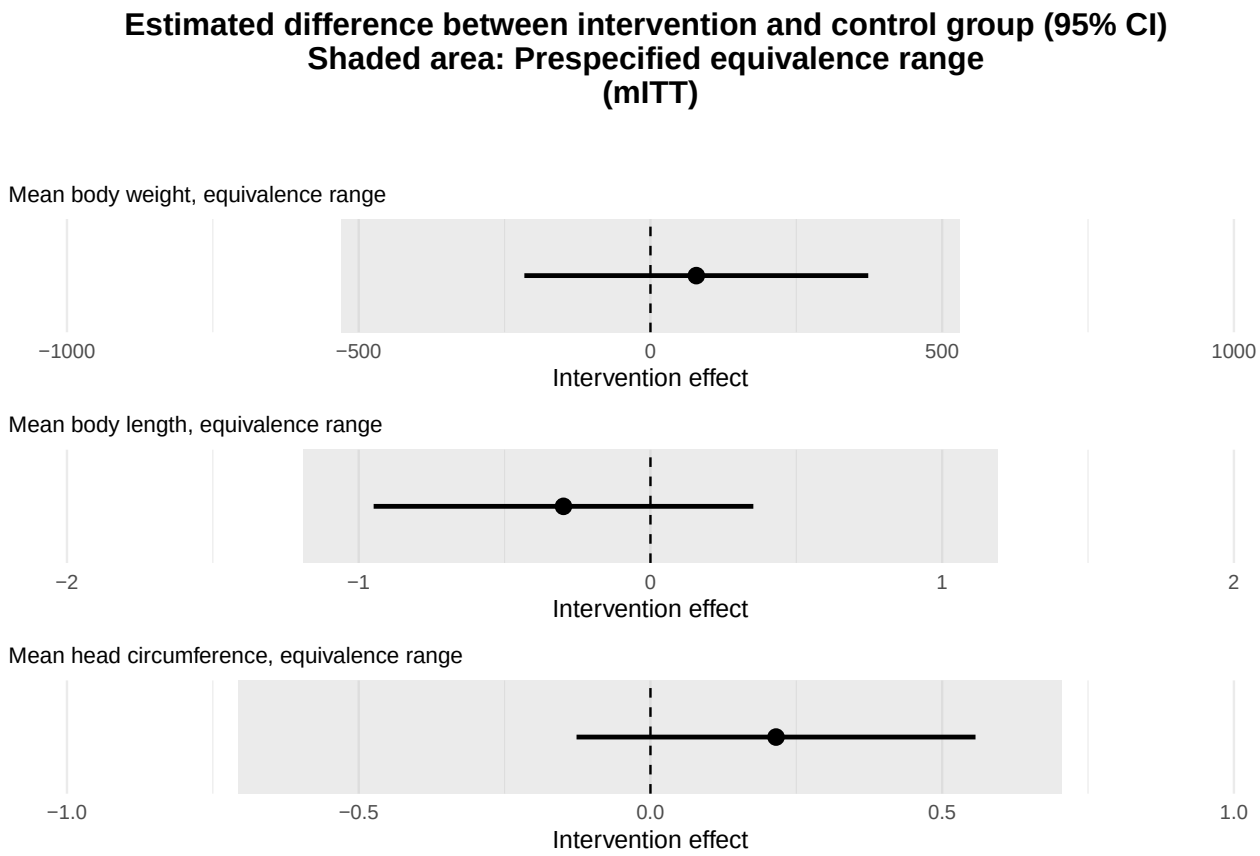
```
                                           2.5 %  97.5 %
(Intercept)                             29.76268 49.9927
Treatmentgroup.fac.renamed.reord5-MTHF-Ca -0.12667  0.5571
Intervention.duration                   -0.02981  0.0402
ChildSex.facM                            0.80018  1.4921
Age                                     -0.02137  0.0319
Birthweight_gram                         0.00054  0.0014
```

4.3.4 Equivalence Plots

We plot the results showing equivalence of intervention (5-MTHF-Ca) and control (Folic acid) for the three body measurements.

```
plot.mITT.weightg <- plot.equivalence.func(res.mITT.weightg)
plot.mITT.length <- plot.equivalence.func(res.mITT.length)
plot.mITT.headcm <- plot.equivalence.func(res.mITT.headcm)

title <- ggdraw() +
  draw_label(paste0("Estimated difference between intervention",
                    " and control group (95% CI) \n",
                    "Shaded area: Prespecified equivalence range \n(",
                    pop.to.consider, ")"),
             fontface = 'bold')

final.plot <- plot_grid(
  title,
```

```
  plot.mITT.weightg,
  plot.mITT.length,
  plot.mITT.headcm,
  ncol = 1,
  align = "v")

# Add the caption below the final plot
ggdraw() +
  draw_plot(final.plot, x = 0, y = 0.05, width = 1, height = 0.95)
```



For safety reasons, we remove the two parameters.

```
rm(log.response)
rm(pop.to.consider)
```

4.3.5 Table with results

We collect the point estimates and the corresponding 95% confidence intervals (CI) + equivalence ranges (ER) in a table.

```
data.mITT.weightg <- get.equivalence.data(res.mITT.weightg)
data.mITT.length <- get.equivalence.data(res.mITT.length)
data.mITT.headcm <- get.equivalence.data(res.mITT.headcm)

rbind(data.mITT.weightg,
      data.mITT.length,
      data.mITT.headcm) %>%
  as_tibble() %>%
```

```
kable(caption = "Estimates, confidence intervals and equivalence ranges for mITT population.",
      col.names = c("Point estimate", "CI lower bound", "CI upper bound",
                    "ER lower bound", "ER upper bound"))
```

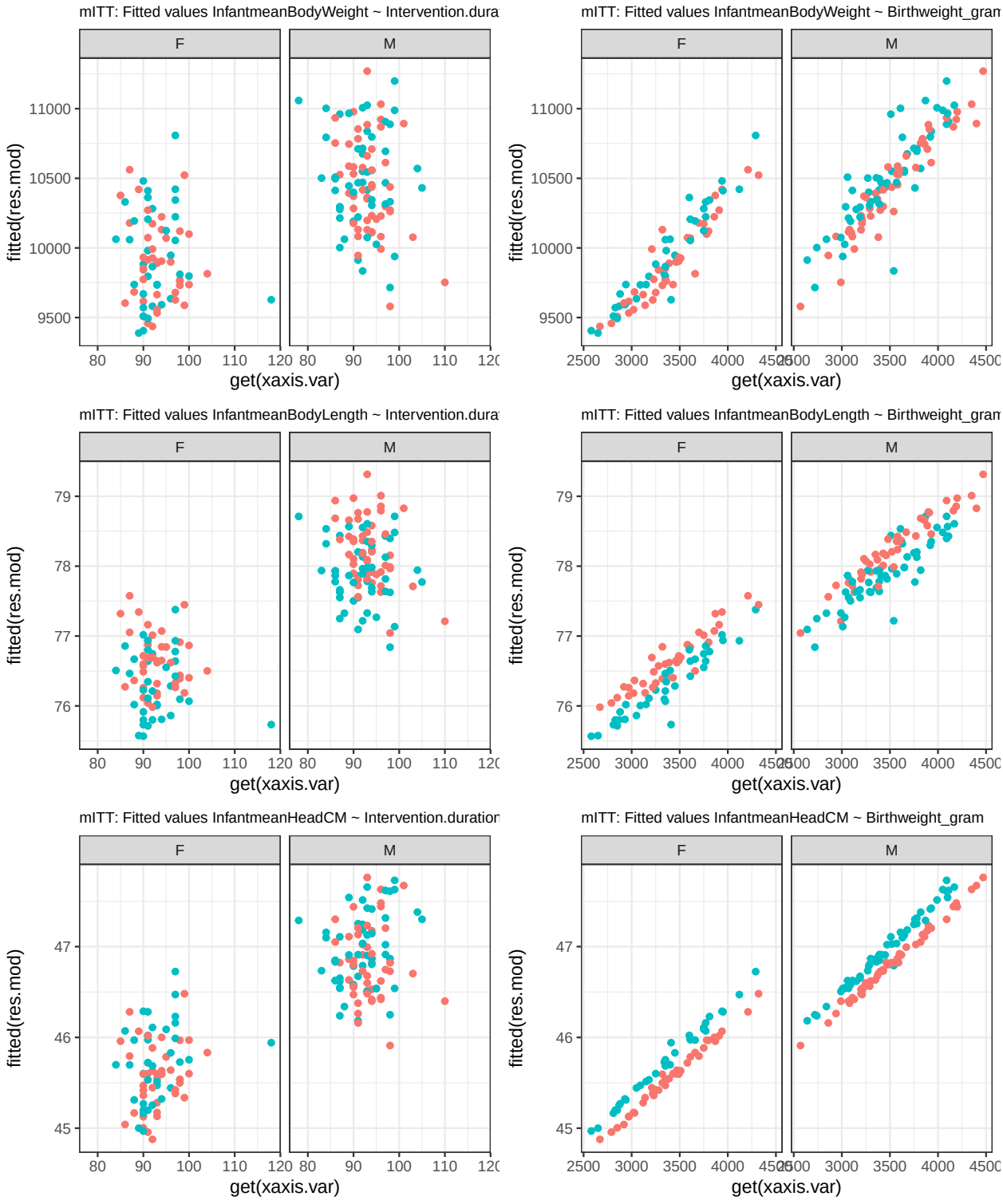
Table 2: Estimates, confidence intervals and equivalence ranges for mITT population.

Point estimate	CI lower bound	CI upper bound	ER lower bound	ER upper bound
78.60	-216.09	373.30	-529.69	529.69
-0.30	-0.95	0.35	-1.19	1.19
0.22	-0.13	0.56	-0.71	0.71

4.3.6 Model checking

For further details about the plots, see Section 4.2.6.

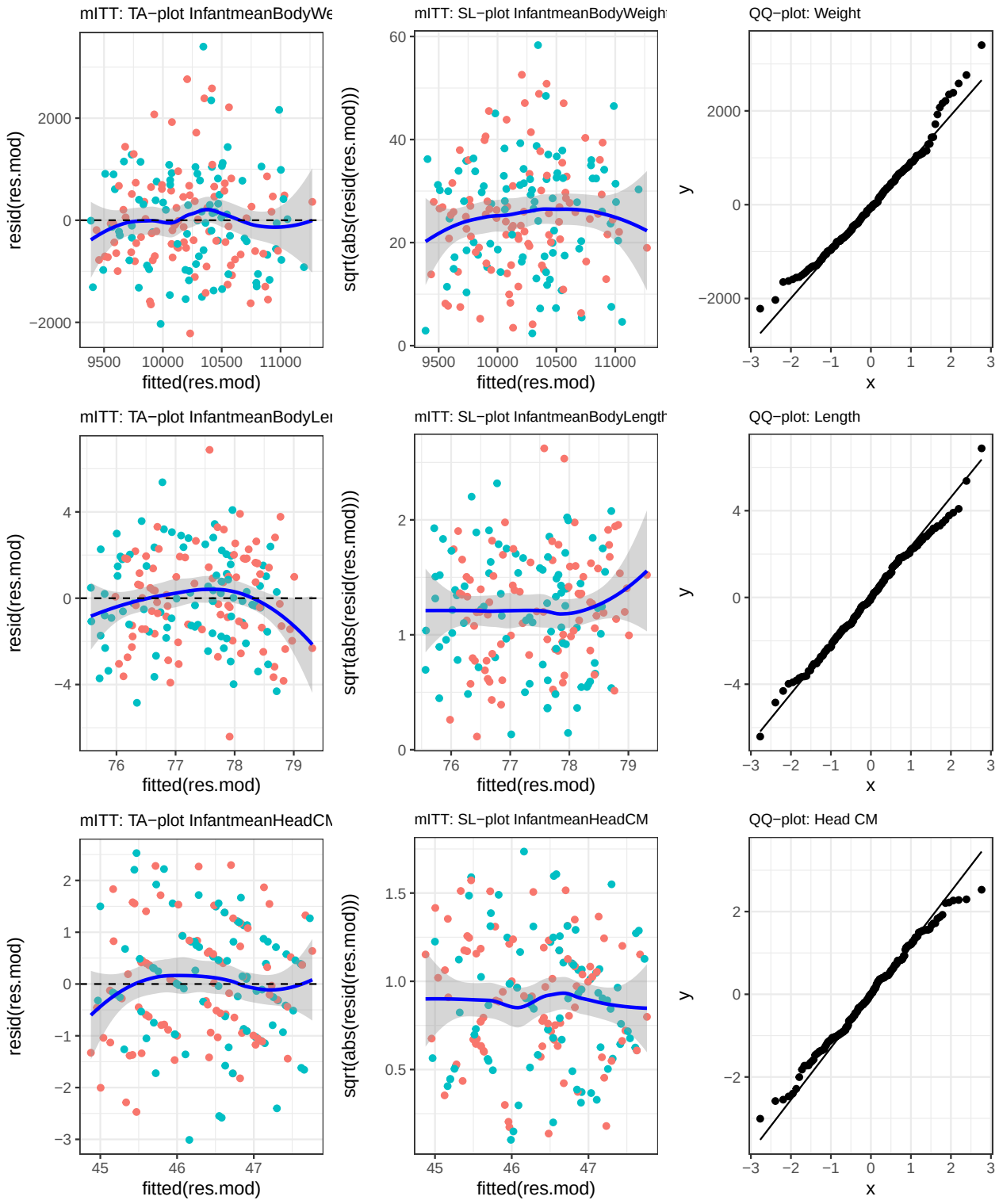
```
plot.fitted.or.TA_SL_QQ.func(pop.to.consider = "mITT",
                             plot.to.create = "Fitted",
                             log.response = FALSE)
```



Fitted values in population mITT using log on the response = FALSE.

```
plot.fitted.or.TA_SL_QQ.func(pop.to.consider = "mITT",
                             plot.to.create = "TA_SL_QQ",
                             log.response = FALSE)
```

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



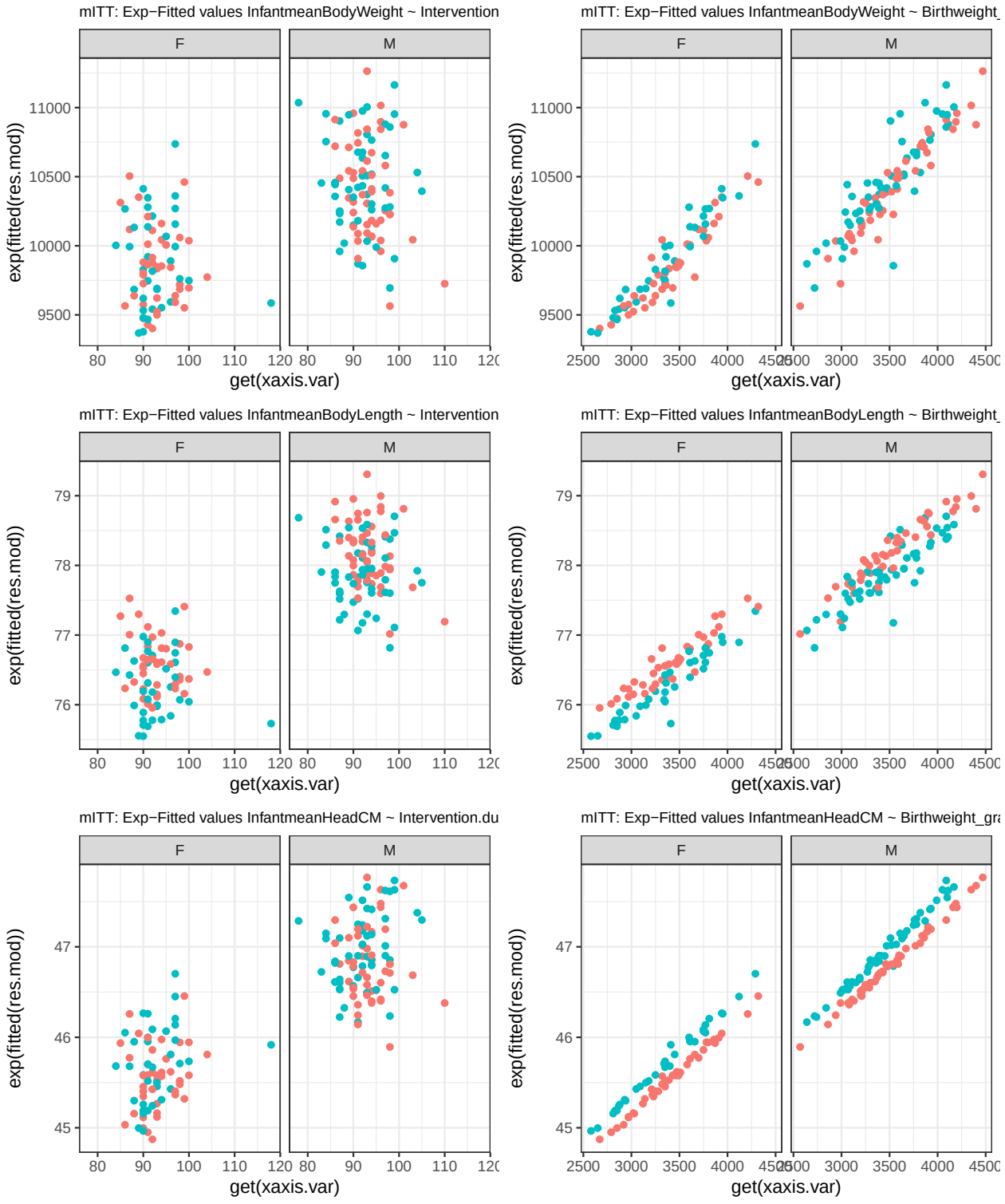
Residual analysis in population mITT using log on the response = FALSE.

All the plots look fine for the three responses.

4.3.7 Showing that log-transformation is not necessary

In this section we show briefly that log-transforming the responses is not necessary. This does not lead to an improved model fit nor improves model diagnostics.

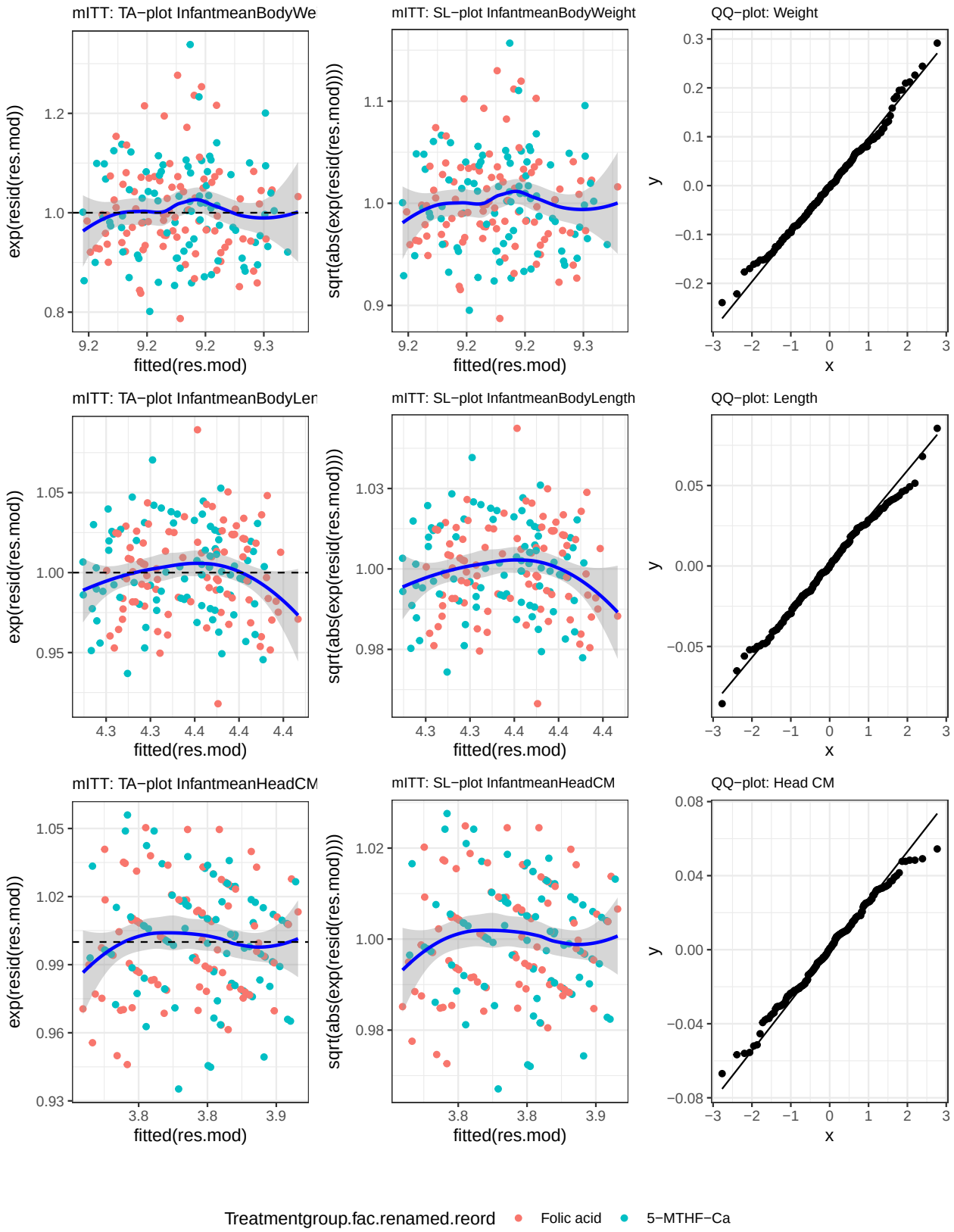
```
plot.fitted.or.TA_SL_QQ.func(pop.to.consider = "mITT",
                             plot.to.create = "Fitted",
                             log.response = TRUE)
```



Fitted values in population mITT using log on the response = TRUE.

```
plot.fitted.or.TA_SL_QQ.func(pop.to.consider = "mITT",
                             plot.to.create = "TA_SL_QQ",
                             log.response = TRUE)
```

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



Residual analysis in population mITT using log on the response = TRUE.

5 Comparison of patients showing up at FUV with those not showing up

The aim of this section is to ensure that there is no systematic bias in those patients that show up at FUV compared to those that did not show up.

This is done by comparing the joint distribution of several relevant variables as indicated by the team and checking if we have evidence that the joint distribution of these variable is different in the group of children that showed up at follow-up visit compared to those that did not show up.

We compare these infants once with respect to certain variables measured at baseline visit and then with respect to the same variables measured at visit 4. Note that these variables are not exclusively those appearing in the modelling, but some further variables that are relevant to compare the two groups are included as well.

To compare the infants at a specific visit with respect to the chosen variables, we do not inspect comparability individually for each variable but use an approach that tests the equality in the joint distribution between the two samples, in all its generality. That is, it does not only detect differences in means in one variable, but the differences can also be with respect to several variables jointly, and does not need to be a difference in mean (e.g. difference in variance etc). This approach is called *Hypothesis test random forest* and can be applied by using the package `hypoRF` from Hediger S, Michel L, Naef J (2024). hypoRF: Random Forest Two-Sample Tests. R package version 1.0.1.

5.1 Helper functions

We first write some helper functions that we can use for both populations.

```
## Create sample with those that show up to FUV and have no missing values there
create.two.samples.func <- function(compared.visit,
                                    pop.to.consider){

  d.idds.showed.up.at.FUV <- d.folic.acid.long %>%
    filter(Visit.fac == "FUV") %>%
    filter(Treatmentgroup.fac.renamed.reord %in% c("5-MTHF-Ca", "Folic acid")) %>%
    mutate(
      any.resp.missing = rowSums(is.na(select(.,
                                              InfantmeanBodyWeight,
                                              InfantmeanBodyLength,
                                              InfantmeanHeadCM))) > 0) %>%
    filter(any.resp.missing == FALSE)

  if(pop.to.consider == "PP"){
    d.idds.showed.up.at.FUV <- d.idds.showed.up.at.FUV %>%
      ## we filter the PP population
      filter(PPPopulation.fac == "yes") %>%
      pull(IDD.fac)
  }else{
    d.idds.showed.up.at.FUV <- d.idds.showed.up.at.FUV %>%
      ## we filter the PP population
      filter(ITTPopulation.fac == "yes") %>%
      pull(IDD.fac)
  }


  ## Create sample with those that did not show up to FUV and have no missing values there
  d.idds.NOT.showed.up.at.FUV <- d.folic.acid.long %>%
    filter(Visit.fac == "FUV") %>%
```

```
  filter(Treatmentgroup.fac.renamed.reord %in% c("5-MTHF-Ca", "Folic acid")) %>%
  mutate(
    any.resp.missing = rowSums(is.na(select(.,
                                            InfantmeanBodyWeight,
                                            InfantmeanBodyLength,
                                            InfantmeanHeadCM))) > 0) %>%
  filter(any.resp.missing == TRUE)


if(pop.to.consider == "PP"){
  d.idds.NOT.showed.up.at.FUV <- d.idds.NOT.showed.up.at.FUV %>%
    ## we filter the PP population
    filter(PPPopulation.fac == "yes") %>%
    pull(IDD.fac)
}else{
  d.idds.NOT.showed.up.at.FUV <- d.idds.NOT.showed.up.at.FUV %>%
    ## we filter the PP population
    filter(ITTPopulation.fac == "yes") %>%
    pull(IDD.fac)
}

## create the data set where we subset to these IDDs

if(compared.visit == "BV"){
  data1.showed.up <- d.folic.acid.long %>%
    filter(IDD.fac %in% d.idds.showed.up.at.FUV) %>%
    filter(Visit.fac == "BV") %>%
    select(- IDD.fac,
           - contains("healthy"),
           - ITTPopulation.fac,
           - InfantmeanBodyWeight.kg,
           ## The following line has been commented out because
           ## the client wanted the variable in grams and not kg
           # - InfantmeanBodyWeight,
           - Intervention.duration,
           - PPPopulation.fac,
           - starts_with("Visit"))

  data2.NOT.showed.up <- d.folic.acid.long %>%
    filter(IDD.fac %in% d.idds.NOT.showed.up.at.FUV) %>%
    filter(Visit.fac == "BV") %>%
    select(- IDD.fac,
           - contains("healthy"),
           - ITTPopulation.fac,
           - InfantmeanBodyWeight.kg,
           ## The following line has been commented out because
           ## the client wanted the variable in grams and not kg
           # - InfantmeanBodyWeight,
           - Intervention.duration,
           - PPPopulation.fac,
           - starts_with("Visit"))
}else if(compared.visit == "V4"){
  data1.showed.up <- d.folic.acid.long %>%
```

```
      filter(IDD.fac %in% d.idds.showed.up.at.FUV) %>%
      filter(Visit.fac == "V4") %>%
      select(-IDD.fac,
             - contains("healthy"),
             - ITTPopulation.fac,
             - InfantmeanBodyWeight.kg,
             ## The following line has been commented out because
             ## the client wanted the variable in grams and not kg
             # - InfantmeanBodyWeight,
             - PPPopulation.fac,
             - starts_with("Visit"))

    data2.NOT.showed.up <- d.folic.acid.long %>%
      filter(IDD.fac %in% d.idds.NOT.showed.up.at.FUV) %>%
      filter(Visit.fac == "V4") %>%
      select(- IDD.fac,
             - contains("healthy"),
             - ITTPopulation.fac,
             - InfantmeanBodyWeight.kg,
             ## The following line has been commented out because
             ## the client wanted the variable in grams and not kg
             # - InfantmeanBodyWeight,
             - PPPopulation.fac,
             - starts_with("Visit"))

  }

  data1.showed.up <- data1.showed.up[complete.cases(data1.showed.up), ]

  data2.NOT.showed.up <-
    data2.NOT.showed.up[complete.cases(data2.NOT.showed.up), ]

  return(list(data1.showed.up = data1.showed.up,
              data2.NOT.showed.up = data2.NOT.showed.up))
}
```

The following function is used to create the variable importance plot.

```
plot.var.imp.hypoRF.func <- function(res.hypoRF,
                                     pop.to.consider,
                                     compared.visit){
  hypo.imp <- res.hypoRF$importance_ranking %>% as.data.frame()
  hypo.imp <- rownames_to_column(hypo.imp)
  colnames(hypo.imp) <- c("variable", "importance")

  ggplot(hypo.imp, aes(x = fct_reorder(variable, importance, .desc = TRUE),
                       y = importance)) +
    geom_bar(stat = "identity") +
    theme_minimal() +
    xlab("Variable") +
    ylab("Importance") +
    geom_hline(yintercept = res.hypoRF$cutoff,
               linetype = "dashed",
               color = "red") +
```

```
    ggtitle(paste0(pop.to.consider, ": Variable Importance of comparison at ",
                   compared.visit, " hypoRF (Sorted)")) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
}
```

The following function is used to create the density plots.

```
plot.density.func <- function(pop.to.consider,
                              compared.visit,
                              data1.showed.up,
                              data2.NOT.showed.up){

  data1.showed.up$Group <- "Show at FUV"
  data2.NOT.showed.up$Group <- "No show at FUV"

  combined_data <- bind_rows(data1.showed.up, data2.NOT.showed.up)

  numeric_vars <- combined_data %>%
    select(where(is.numeric)) %>%
    colnames()
  ##
  ## !Careful: check this when plotting different variables!
  char_vars <- c("Birth weight", "Mean body weight", "Mean body length",
                 "Mean head circumference", "Hematocrit", "Haemoglobin",
                 "MCV", "Age")
  unit_vars <- c("(g)", "(g)", "(cm)", "(cm)", "(%)", "(gl)", "(fl)", "(d)")

  if ("Intervention.duration" %in% numeric_vars) {

    char_vars <- c(char_vars, "Intervention duration")
    unit_vars <- c(unit_vars, "(d)")

  }

  names(numeric_vars) <- char_vars
  names(unit_vars) <- char_vars
  ## check
  # numeric_vars
  # unit_vars

  # Generate plots for each variable
  plots <- lapply(numeric_vars, function(var) {

    var.name <- names(numeric_vars[var == numeric_vars])
    var.unit <- unit_vars[var.name == names(unit_vars)]

    ggplot(combined_data, aes_string(x = var, fill = "Group")) +
      geom_density(alpha = 0.5) +
      theme_minimal() +
      labs(title = paste0("Density plot for \n", str_to_lower(var.name), " (", pop.to.consider, ")"),
           x = paste(var.name, var.unit),
           y = "Density") +
      scale_fill_manual(values = c("blue", "red")) +
      theme(legend.title = element_blank(),
```

```
          legend.position = "none")
})

test.plot <- ggplot(combined_data,
                    mapping = aes(x = Age, fill = Group)) +
  geom_density(alpha = 0.5) +
  theme_minimal() +
  scale_fill_manual(values = c("blue", "red")) +
  theme(legend.title = element_blank())

common.legend <- ggpubr::get_legend(test.plot)

## The following is not elegant
if(length(plots)== 8){
  plot.grid <- plot_grid(
    plots[[1]],
    plots[[2]],
    plots[[3]],
    plots[[4]],
    plots[[5]],
    plots[[6]],
    plots[[7]],
    plots[[8]],
    ncol = 2,
    align = "v"
  )
}else if (length(plots)==9){
  plot.grid <- plot_grid(
    plots[[1]],
    plots[[2]],
    plots[[3]],
    plots[[4]],
    plots[[5]],
    plots[[6]],
    plots[[7]],
    plots[[8]],
    plots[[9]],
    ncol = 2,
    align = "v"
  )
}


## Combine the grid and the common legend
final.plot <- plot_grid(
  plot.grid,
  common.legend,
  ncol = 1,
  rel_heights = c(0.9, 0.1) # Allocate space for the legend
)

caption_text <- paste0(pop.to.consider,
                       ": Density plots comparing groups *Showed Up* and
```

```
                          *Did Not Show Up* at ", compared.visit, ".")

  # Add the caption below the final plot
  final.plot.with.caption <- ggdraw() +
    draw_plot(final.plot, x = 0, y = 0.05, width = 1, height = 0.95) +
    draw_label(caption_text, x = 0.5, y = 0.02, hjust = 0.5, size = 14)

  final.plot.with.caption
}
```

The following function is used to create the stacked bar plots for the categorical variables.

```
plot.stacked.bar.func <- function(pop.to.consider,
                                  compared.visit,
                                  data1.showed.up,
                                  data2.NOT.showed.up){

  data1.showed.up$Group <- "Show at FUV"
  data2.NOT.showed.up$Group <- "No show at FUV"

  combined_data <- bind_rows(data1.showed.up, data2.NOT.showed.up)

  factor.vars <- combined_data %>%
    select(!where(is.numeric)) %>%
    select(-Group) %>%
    colnames()

  char_vars <- c("child sex", "intervention group")
  names(factor.vars) <- char_vars


  # Generate plots for each variable
  plots <- lapply(factor.vars, function(var) {

    var.name <- names(factor.vars[var == factor.vars])

    tot.number.in.two.groups <- combined_data %>% count(Group)

    dat.summarised <- combined_data %>% count(get(var), Group)

    dat.summarised <- left_join(dat.summarised,
                                tot.number.in.two.groups, by = "Group")

    dat.summarised <- dat.summarised %>%
      mutate(percentage = n.x/n.y)

    # Create the stacked bar plot
    ggplot(dat.summarised, aes(x = Group,
                               y = percentage,
                               fill = `get(var)`)) +
      geom_bar(stat = "identity", position = "stack") +
      theme_minimal() +
      labs(
        x = "",
```

```
        y = "Percentage",
        fill = "Factor Levels",
        title = paste0("Percentage plot for \n", var.name, " (", pop.to.consider, ")"),
      ) +
      theme(
        axis.title = element_text(size = 14, face = "bold"),
        axis.text = element_text(size = 12),
        legend.title = element_text(size = 12),
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
        legend.text = element_text(size = 10)
      )
  })

  plot.grid <- plot_grid(
    plots[[1]],
    plots[[2]],
    ncol = 2,
    align = "v")

  caption_text <- paste0(pop.to.consider,
                         ": Percentage plots comparing groups *Showed Up* and
                         *Did Not Show Up* at ", compared.visit, ".")

  # Add the caption below the final plot
  final.plot.with.caption <- ggdraw() +
    draw_plot(plot.grid, x = 0, y = 0.05, width = 1, height = 0.95) +
    draw_label(caption_text, x = 0.5, y = 0.05, hjust = 0.5, size = 12)

  final.plot.with.caption

}
```

5.2 PP-Population

We check if we have an observation per time and patient. This should be the case, otherwise we cannot define the upcoming two subsets of patients who came to FUV and those who did not.

```
table(table(d.folic.acid.long$IDD.fac))
```

```
  3
360
```

```
## makes sense, for each patient we have three observations (= 3 visits)
```

5.2.1 Comparison at BV

We first compare to BV. We set the parameters for this setting here.

```
pop.to.consider <- "PP"
compared.visit <- "BV"
```

We create the two samples, those who showed up and those who did not.

```
res.samples <- create.two.samples.func(compared.visit = compared.visit,
                                       pop.to.consider = pop.to.consider)
```

```
data1.showed.up <- res.samples$data1.showed.up
data2.NOT.showed.up <- res.samples$data2.NOT.showed.up
```

We check how many kids did show up at FUV and how many did not.

```
## Number of kids who showed up
nrow(data1.showed.up)
```

```
[1] 143
```

```
## Number of kids how did not show up
nrow(data2.NOT.showed.up)
```

```
[1] 8
```

We now fit the hypoRF for the two sample testing. This test assesses the equality in the joint distribution between the two samples, in all its generality. That is, it does not only detect differences in means in one variable, but the differences can also be with respect to several variables jointly, and does not need to be a difference in mean (e.g. difference in variance etc).

```
res.hypoRF <- hypoRF(data1 = data1.showed.up,
                     data2 = data2.NOT.showed.up,
                     K = 1000,
                     seed = 2024,
                     num.trees = 500,
                     importance = "impurity")

## results
res.hypoRF$pvalue
```

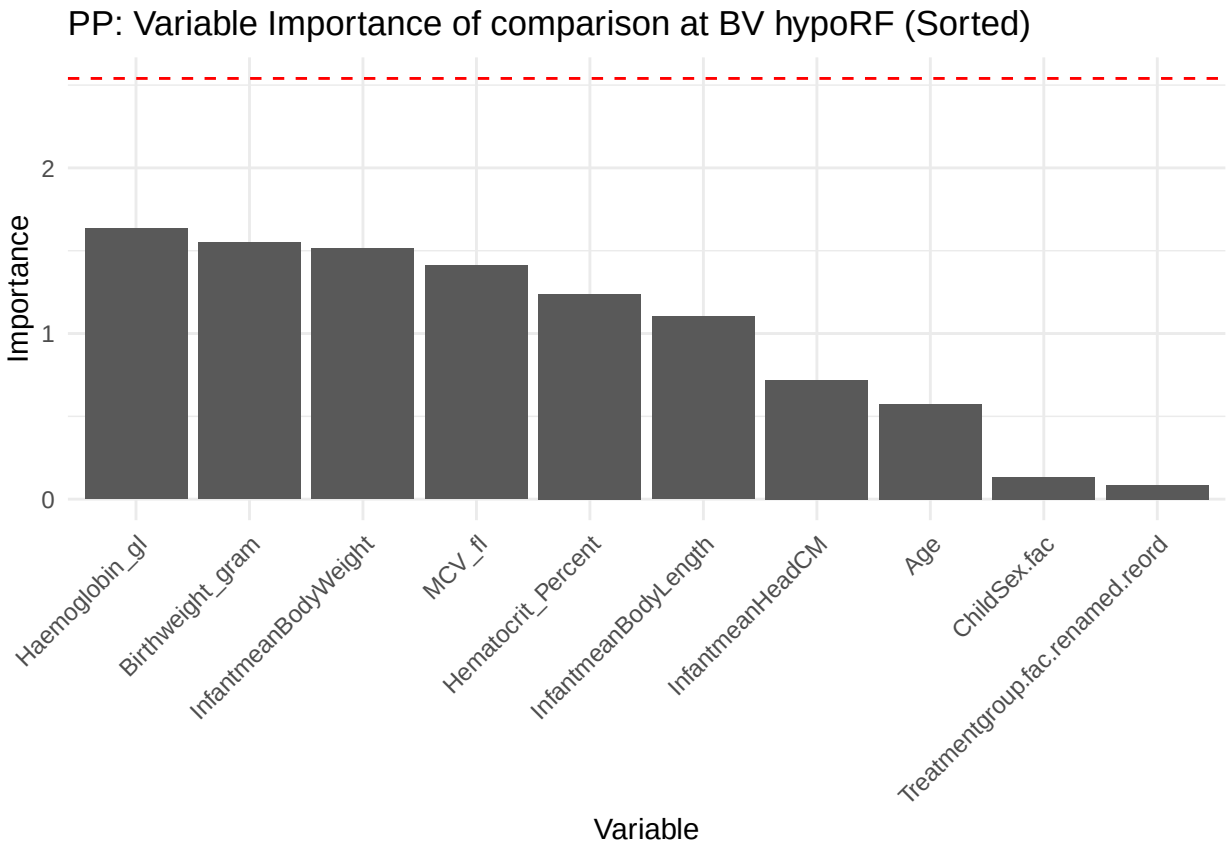
```
[1] 0.99
```

```
## the p-value is not significant
```

The p-value of the test of equality in distribution at BV for those who showed up at FUV and those who not not is not significant ($p = 0.99$), confirming that the two groups (children showing up and not af FUV) are comparable.

We also plot the variable importance. If the bar of a variable exceeds the red line, then there might be a violation of the equality in distribution that has to do with this variable. However, we see that this is not the case.

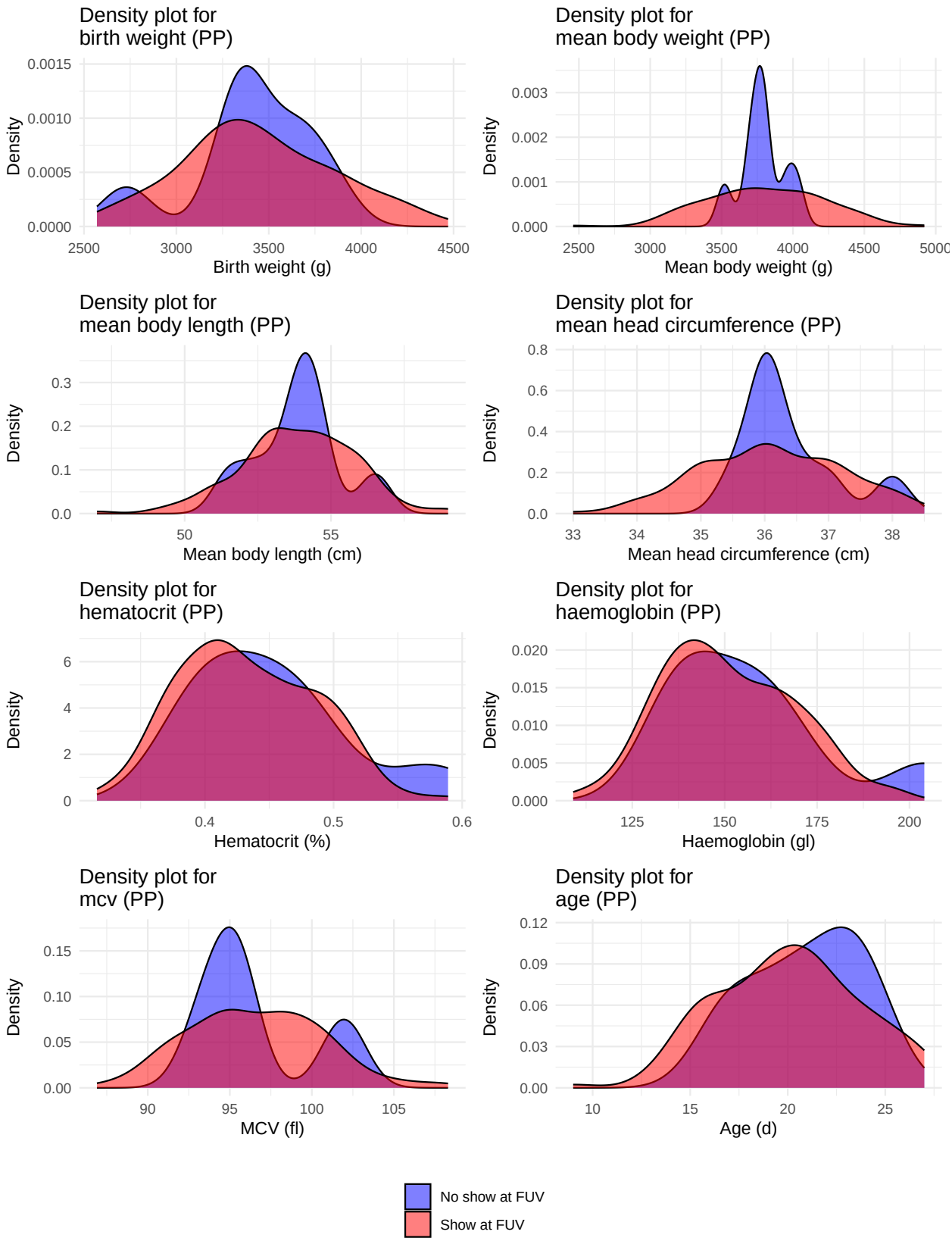
```
plot.var.imp.hypoRF.func(res.hypoRF = res.hypoRF,
                         pop.to.consider = pop.to.consider,
                         compared.visit = compared.visit)
```



We also show density plots for all of the numeric variables, comparing at BV visit the densities of those who did show up at FUV and those who did not.

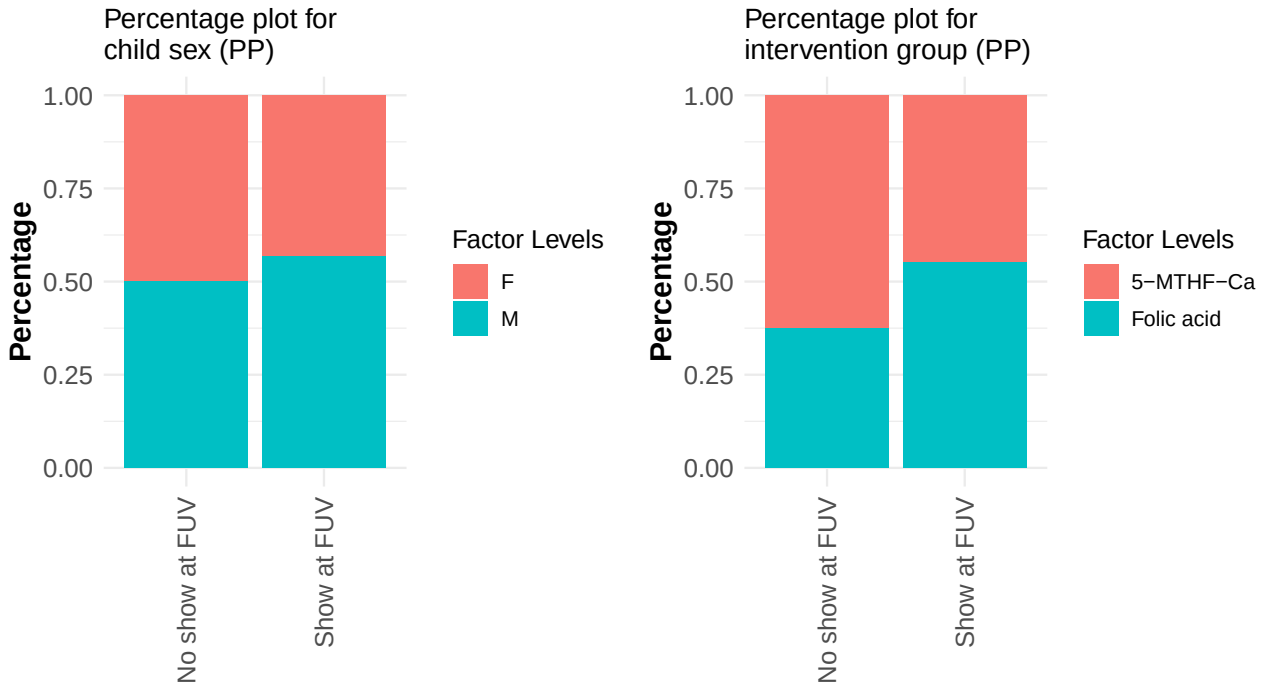
```
## Combine datasets and add a "Group" variable
plot.density.func(pop.to.consider = pop.to.consider,
                  compared.visit = compared.visit ,
                  data1.showed.up = data1.showed.up,
                  data2.NOT.showed.up = data2.NOT.showed.up)
```

```
Warning: ‘aes_string()‘ was deprecated in ggplot2 3.0.0.
i Please use tidy evaluation idioms with ‘aes()‘.
i See also ‘vignette("ggplot2-in-packages")‘ for more information.
This warning is displayed once every 8 hours.
Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was generated.
```



PP: Density plots comparing groups *Showed Up* and *Did Not Show Up* at BV.

```
plot.stacked.bar.func(pop.to.consider = pop.to.consider,
                      compared.visit = compared.visit,
                      data1.showed.up = data1.showed.up,
                      data2.NOT.showed.up = data2.NOT.showed.up)
```



PP: Percentage plots comparing groups *Showed Up* and *Did Not Show Up* at BV.

```
rm(pop.to.consider)
rm(compared.visit)
rm(data1.showed.up)
rm(data2.NOT.showed.up)
rm(res.samples)
rm(res.hypoRF)
```

5.2.2 Comparison at V4

We first compare to V4. We set the parameters for this setting here.

```
pop.to.consider <- "PP"
compared.visit <- "V4"
```

We create the two samples, those who showed up and those who did not.

```
res.samples <- create.two.samples.func(compared.visit = compared.visit,
                                       pop.to.consider = pop.to.consider)

data1.showed.up <- res.samples$data1.showed.up
data2.NOT.showed.up <- res.samples$data2.NOT.showed.up
```

We check how many kids did show up at FUV and how many did not.

```
## Number of kids who showed up
nrow(data1.showed.up)
```

```
[1] 142
```

```
## Number of kids how did not show up
nrow(data2.NOT.showed.up)
```

```
[1] 9
```

We now fit the hypoRF for the two sample testing.

```
res.hypoRF <- hypoRF(data1 = data1.showed.up,
                     data2 = data2.NOT.showed.up,
                     K = 1000,
                     seed = 2024,
                     num.trees = 500,
                     importance = "impurity")

## results
res.hypoRF$pvalue
```

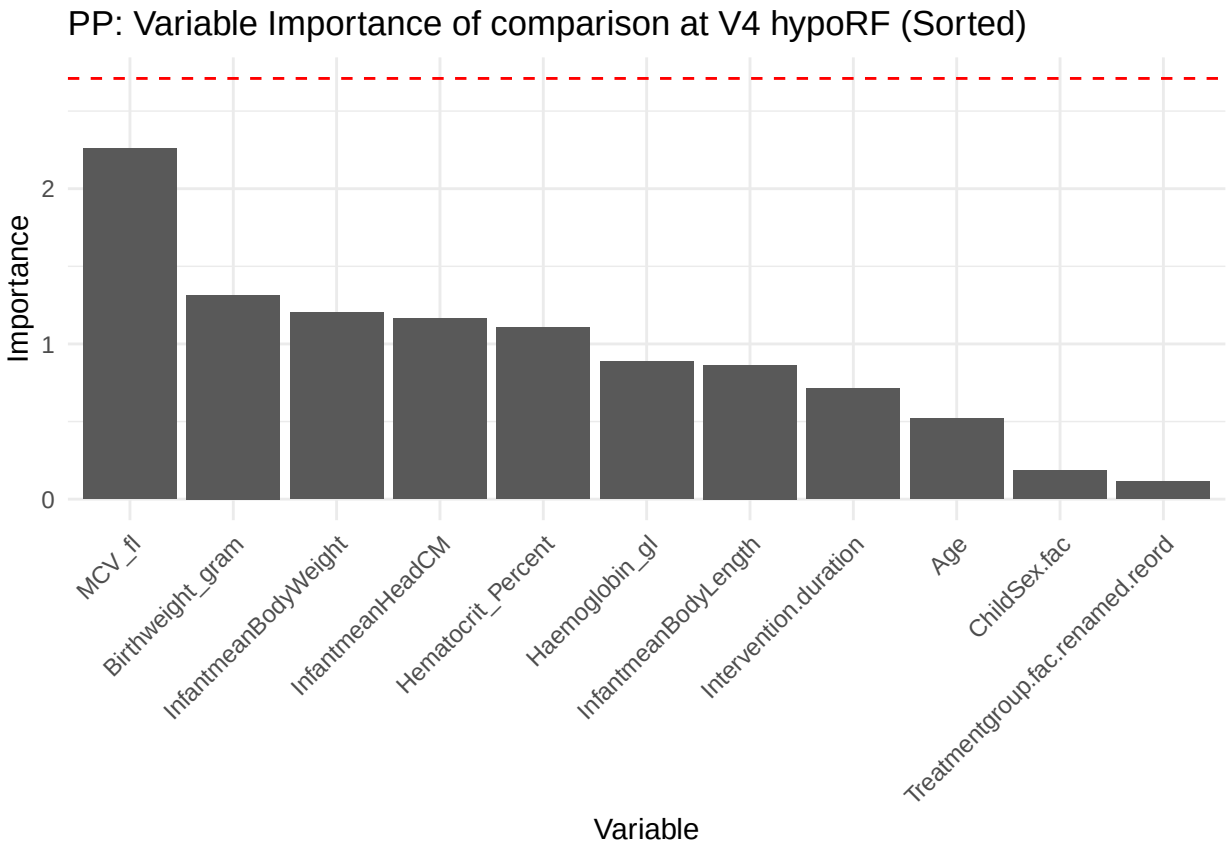
```
[1] 0.8
```

```
## the p-value is not significant
```

The p-value of the test of equality in distribution at V4 for those who showed up at FUV and those who not not is not significant (p = 0.8), confirming that the two groups (children showing up and not af FUV) are comparable.

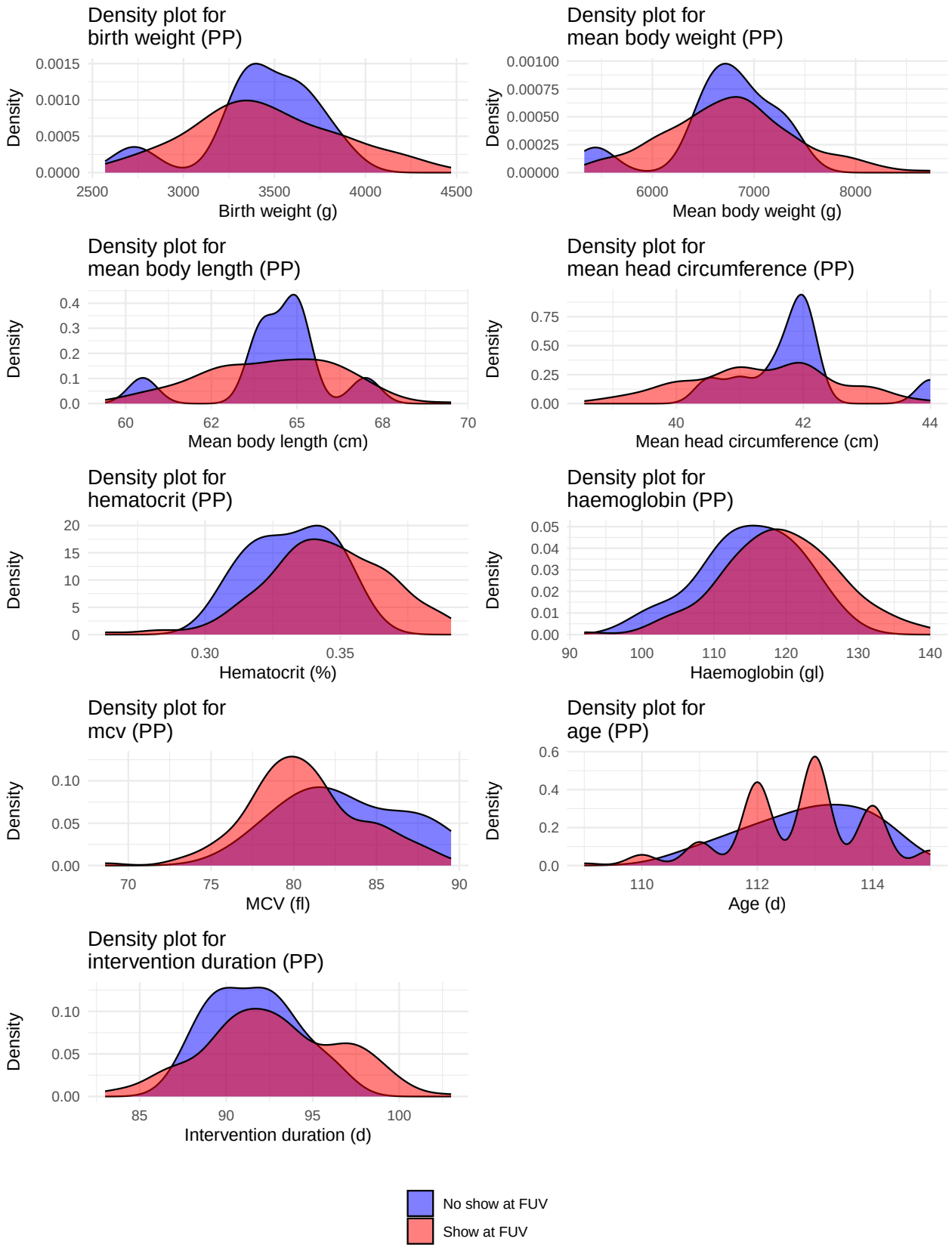
We also plot the variable importance. If the bar of a variable exceeds the red line, then there might be a violation of the equality in distribution that has to do with this variable. However, we see that this is not the case.

```
plot.var.imp.hypoRF.func(res.hypoRF = res.hypoRF,
                         pop.to.consider = pop.to.consider,
                         compared.visit = compared.visit)
```



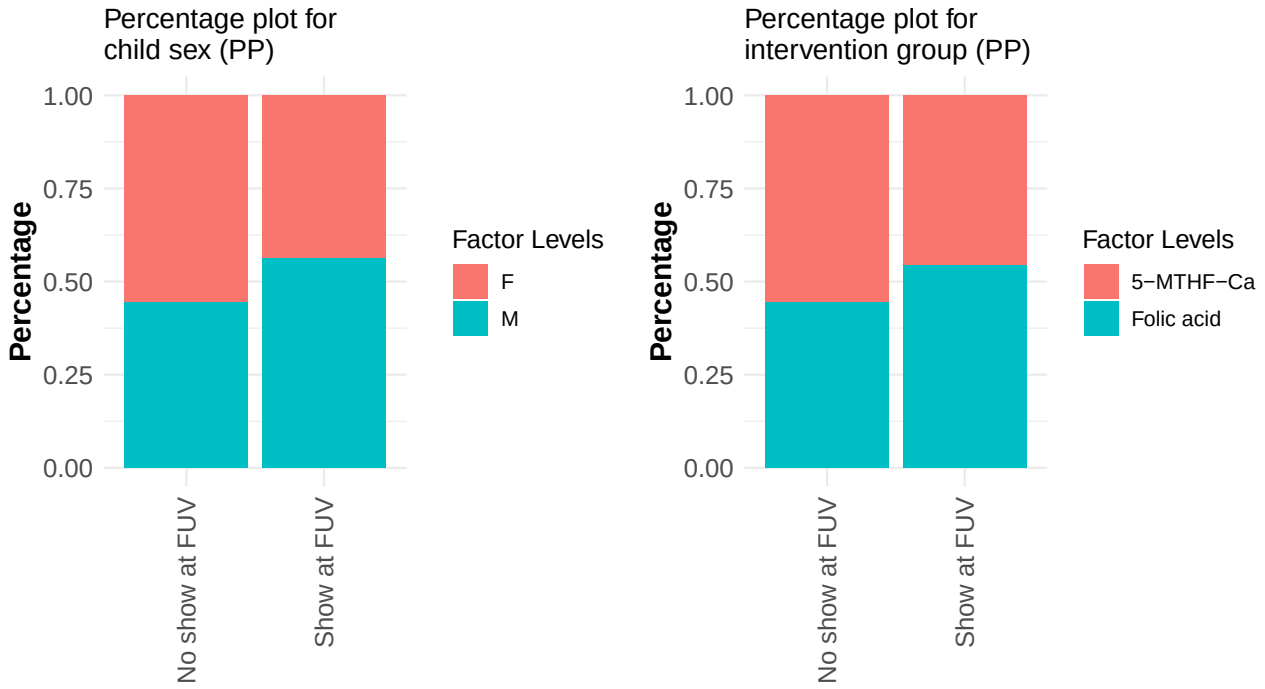
We also show density plots for all of the numeric variables, comparing at V4 visit the densities of those who did show up at FUV and those who did not.

```
## Combine datasets and add a "Group" variable
plot.density.func(pop.to.consider = pop.to.consider,
                  compared.visit = compared.visit ,
                  data1.showed.up = data1.showed.up,
                  data2.NOT.showed.up = data2.NOT.showed.up)
```



PP: Density plots comparing groups *Showed Up* and *Did Not Show Up* at V4.

```
plot.stacked.bar.func(pop.to.consider = pop.to.consider,
                      compared.visit = compared.visit ,
                      data1.showed.up = data1.showed.up,
                      data2.NOT.showed.up = data2.NOT.showed.up)
```



PP: Percentage plots comparing groups *Showed Up* and *Did Not Show Up* at V4.

```
rm(pop.to.consider)
rm(compared.visit)
rm(data1.showed.up)
rm(data2.NOT.showed.up)
rm(res.samples)
rm(res.hypoRF)
```

5.3 mITT-Population

5.3.1 Comparison at BV

We first compare to BV. We set the parameters for this setting here.

```
pop.to.consider <- "mITT"
compared.visit <- "BV"
```

```
## We create the two samples, those who showed up and those who did not.
res.samples <- create.two.samples.func(compared.visit = compared.visit,
                                       pop.to.consider = pop.to.consider)

data1.showed.up <- res.samples$data1.showed.up
data2.NOT.showed.up <- res.samples$data2.NOT.showed.up
```

We check how many kids did show up at FUV and how many did not.

```
## Number of kids who showed up
nrow(data1.showed.up)
```

```
[1] 174
```

```
## Number of kids who did not show up
nrow(data2.NOT.showed.up)
```

```
[1] 22
```

We now fit the hypoRF for the two sample testing.

```
res.hypoRF <- hypoRF(data1 = data1.showed.up,
                     data2 = data2.NOT.showed.up,
                     K = 1000,
                     seed = 2024,
                     num.trees = 500,
                     importance = "impurity")

## results
res.hypoRF$pvalue
```

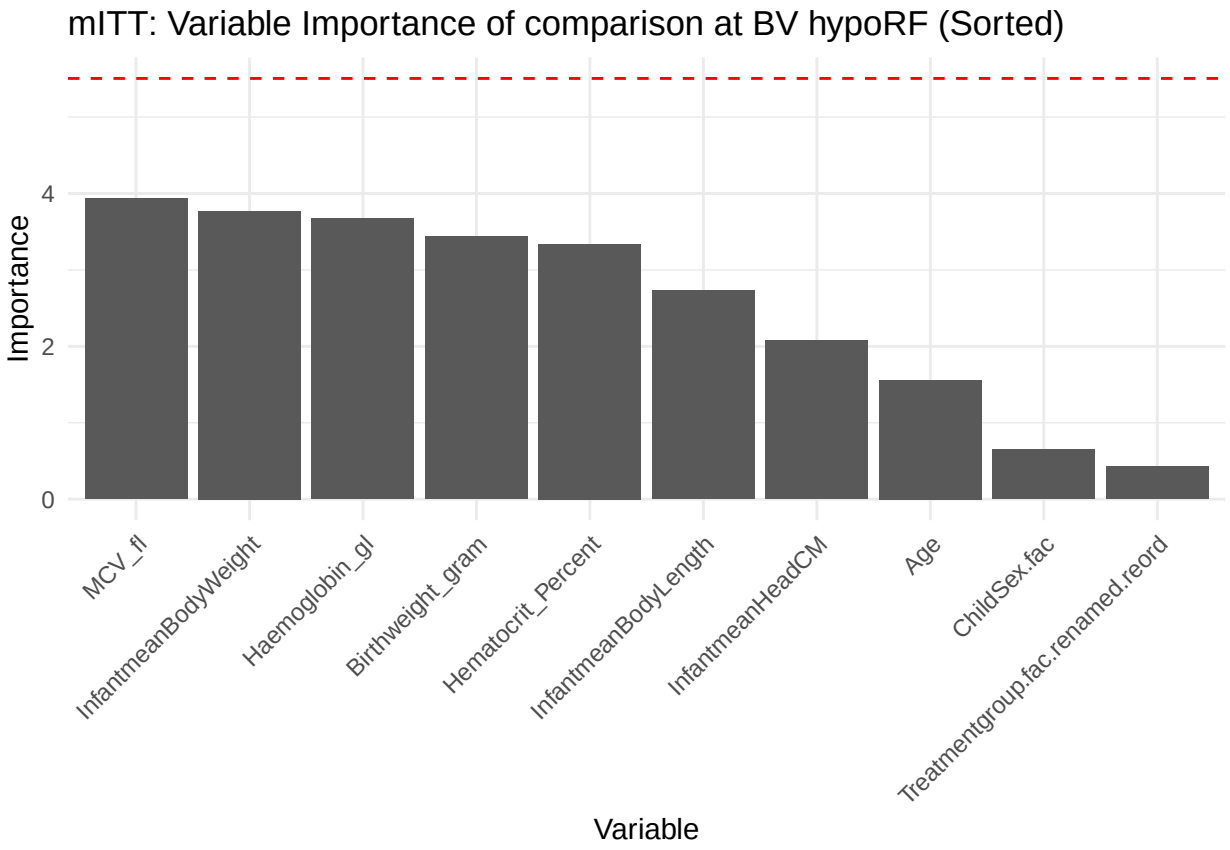
```
[1] 0.96
```

```
## the p-value is not significant
```

The p-value of the test of equality in distribution at BV for those who showed up at FUV and those who not not is not significant (p = 0.96), confirming that the two groups (children showing up and not at FUV) are comparable.

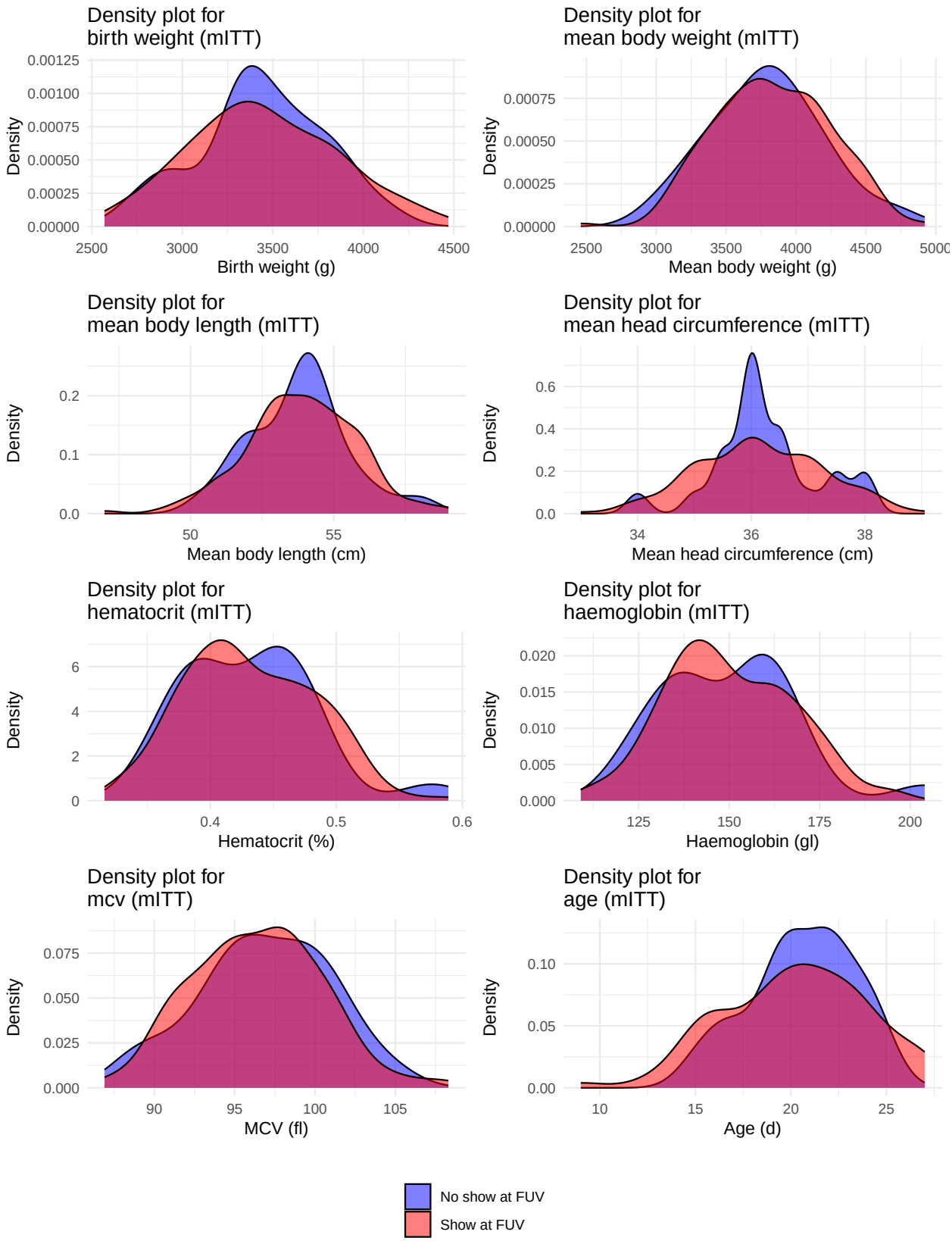
We also plot the variable importances. If the bar of a variable exceeds the red line, then there might be a violation of the equality in distribution that has to do with this variable. However, we see that this is not the case.

```
plot.var.imp.hypoRF.func(res.hypoRF = res.hypoRF,
                         pop.to.consider = pop.to.consider,
                         compared.visit = compared.visit)
```



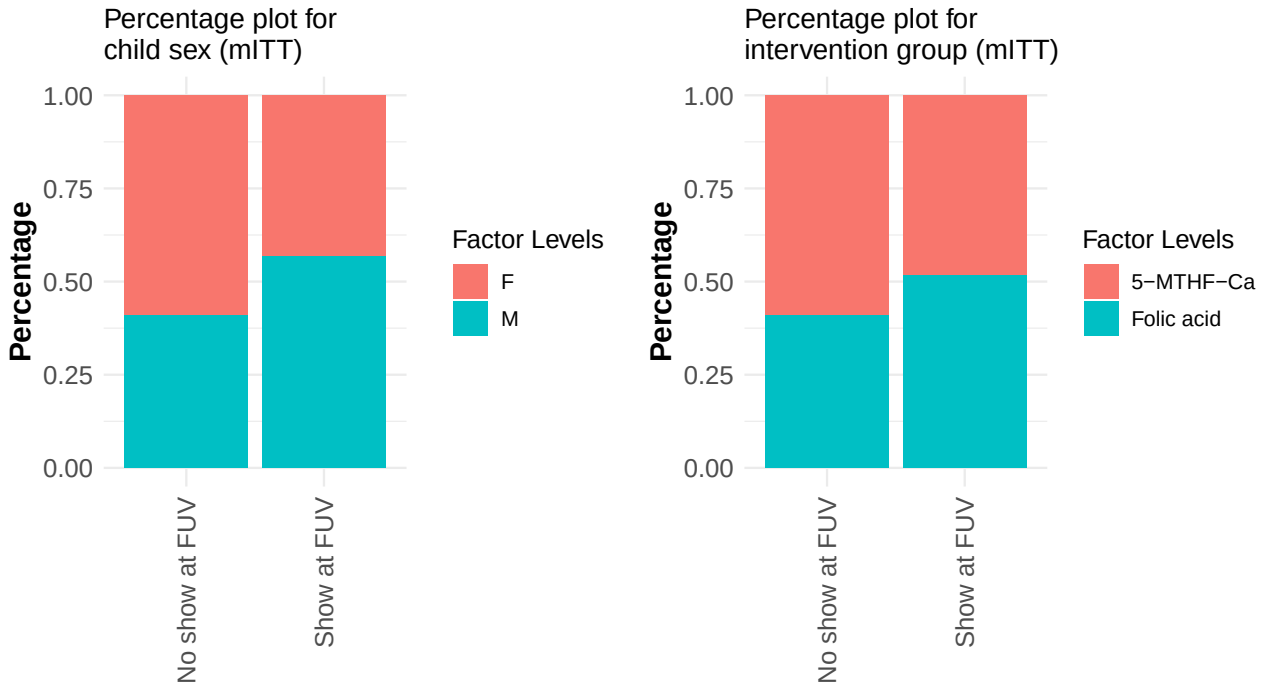
We also show density plots for all of the numeric variables, comparing at BV visit the densities of those who did show up at FUV and those who did not.

```
## Combine datasets and add a "Group" variable
plot.density.func(pop.to.consider = pop.to.consider,
                  compared.visit = compared.visit ,
                  data1.showed.up = data1.showed.up,
                  data2.NOT.showed.up = data2.NOT.showed.up)
```



mITT: Density plots comparing groups *Showed Up* and *Did Not Show Up* at BV.

```
plot.stacked.bar.func(pop.to.consider = pop.to.consider,
                      compared.visit = compared.visit ,
                      data1.showed.up = data1.showed.up,
                      data2.NOT.showed.up = data2.NOT.showed.up)
```



mITT: Percentage plots comparing groups *Showed Up* and *Did Not Show Up* at BV.

```
rm(pop.to.consider)
rm(compared.visit)
rm(data1.showed.up)
rm(data2.NOT.showed.up)
rm(res.samples)
rm(res.hypoRF)
```

5.3.2 Comparison at V4

We first compare to V4. We set the parameters for this setting here.

```
pop.to.consider <- "mITT"
compared.visit <- "V4"
```

We create the two samples, those who showed up and those who did not.

```
res.samples <- create.two.samples.func(compared.visit = compared.visit,
                                       pop.to.consider = pop.to.consider)

data1.showed.up <- res.samples$data1.showed.up
data2.NOT.showed.up <- res.samples$data2.NOT.showed.up
```

We check how many kids did show up at FUV and how many did not.

```
## Number of kids who showed up
nrow(data1.showed.up)
```

```
[1] 173
```

```
## Number of kids how did not show up
nrow(data2.NOT.showed.up)
```

```
[1] 9
```

We now fit the hypoRF for the two sample testing.

```
res.hypoRF <- hypoRF(data1 = data1.showed.up,
                     data2 = data2.NOT.showed.up,
                     K = 1000,
                     seed = 2024,
                     num.trees = 500,
                     importance = "impurity")

## results
res.hypoRF$pvalue
```

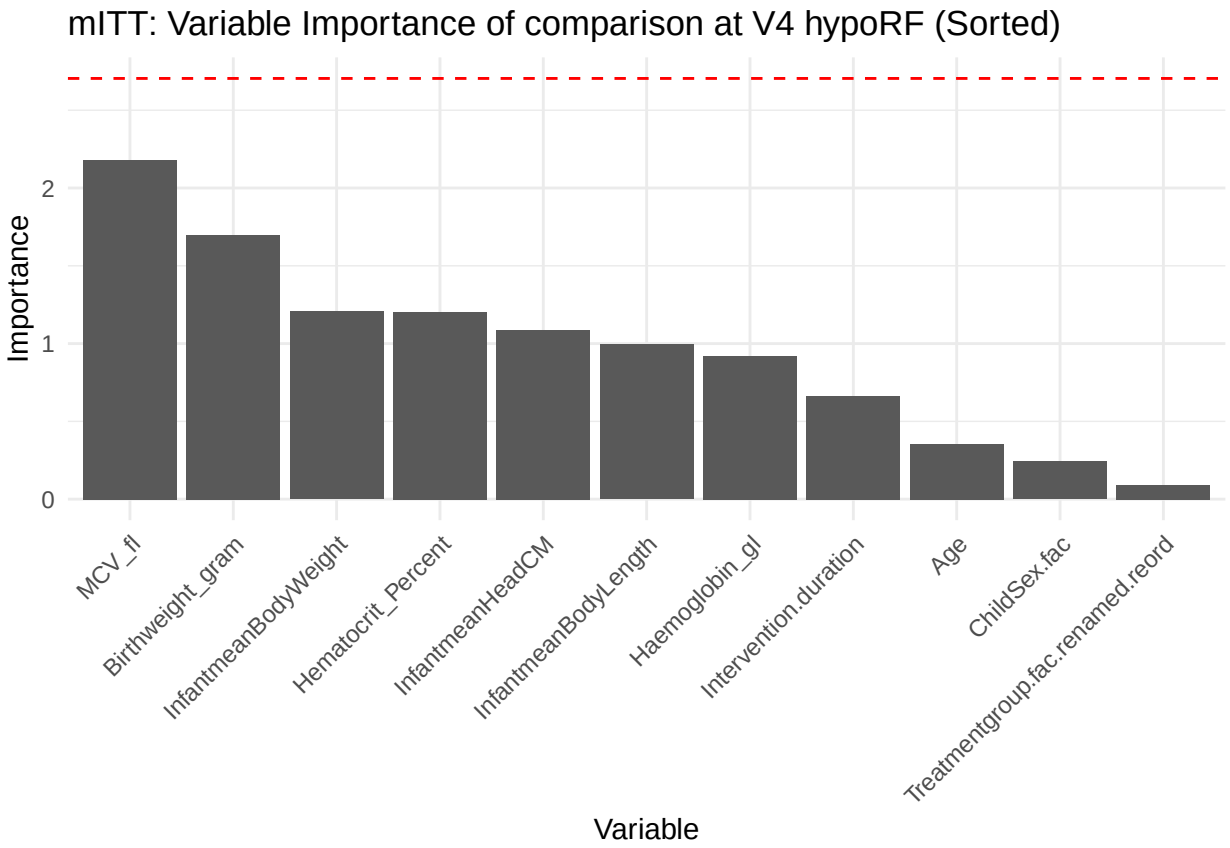
```
[1] 0.89
```

```
## the p-value is not significant
```

The p-value of the test of equality in distribution at V4 for those who showed up at FUV and those who not not is not significant (p = 0.89), confirming that the two groups (children showing up and not af FUV) are comparable.

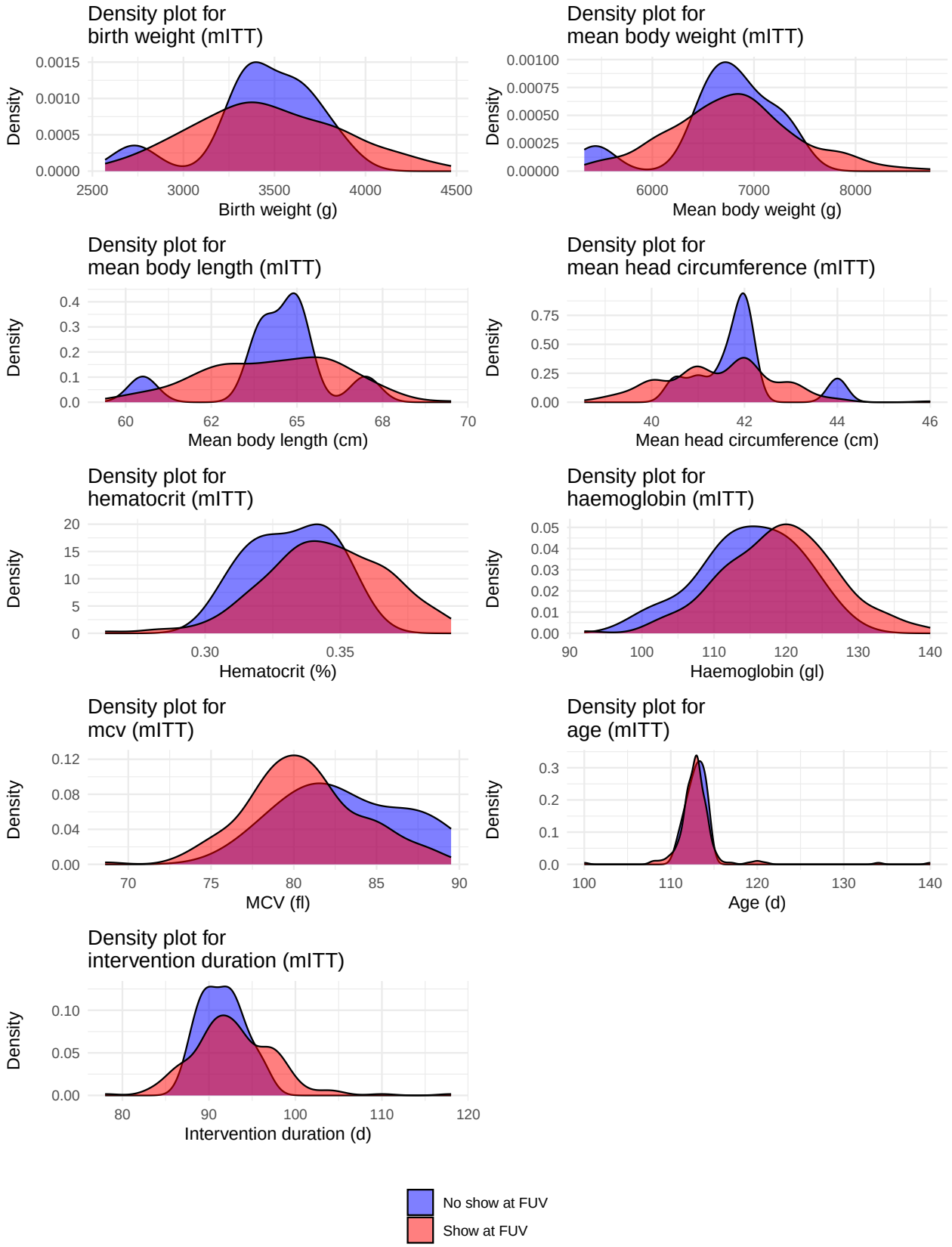
We also plot the variable importances. If the bar of a variable exceeds the red line, then there might be a violation of the equality in distribution that has to do with this variable. However, we see that this is not the case.

```
plot.var.imp.hypoRF.func(res.hypoRF = res.hypoRF,
                         pop.to.consider = pop.to.consider,
                         compared.visit = compared.visit)
```



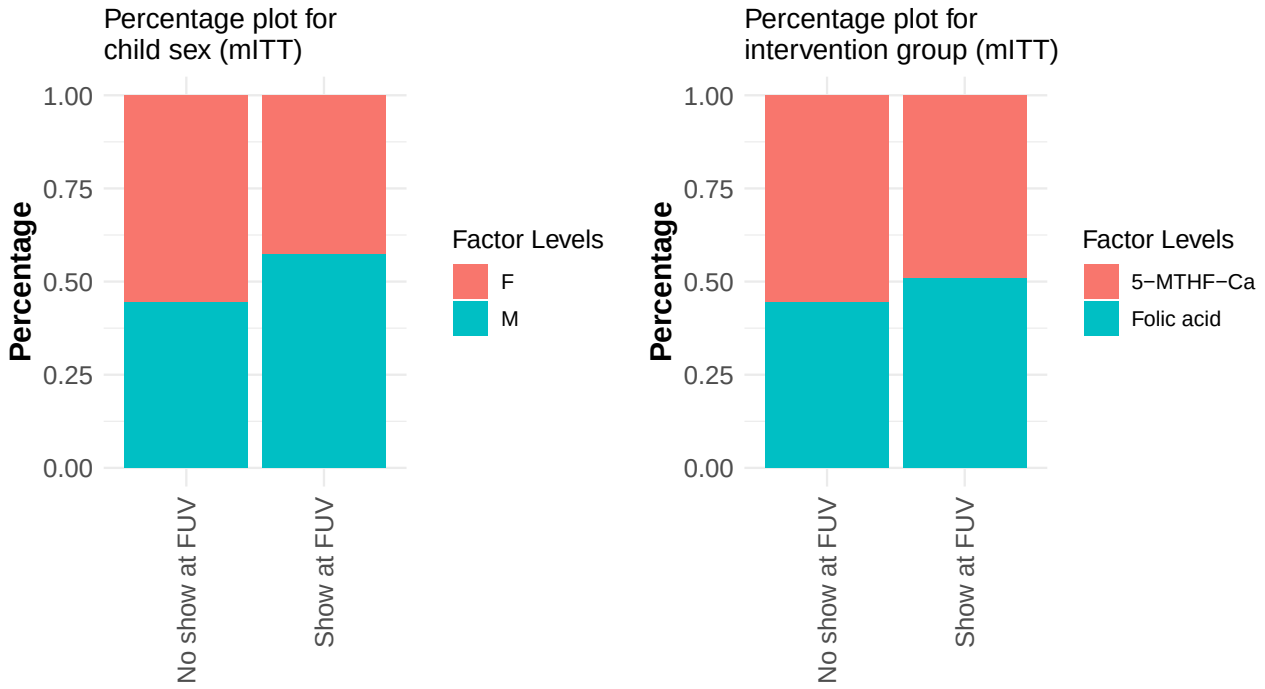
We also show density plots for all of the numeric variables, comparing at V4 visit the densities of those who did show up at FUV and those who did not.

```
## Combine datasets and add a "Group" variable
plot.density.func(pop.to.consider = pop.to.consider,
                  compared.visit = compared.visit ,
                  data1.showed.up = data1.showed.up,
                  data2.NOT.showed.up = data2.NOT.showed.up)
```



mITT: Density plots comparing groups *Showed Up* and *Did Not Show Up* at V4.

```
plot.stacked.bar.func(pop.to.consider = pop.to.consider,
                      compared.visit = compared.visit ,
                      data1.showed.up = data1.showed.up,
                      data2.NOT.showed.up = data2.NOT.showed.up)
```



mITT: Percentage plots comparing groups *Showed Up* and *Did Not Show Up* at V4.

```
rm(pop.to.consider)
rm(compared.visit)
rm(data1.showed.up)
rm(data2.NOT.showed.up)
rm(res.samples)
rm(res.hypoRF)
```

6 Code for figures

In this section, we report additional code used to generate figures for the manuscript.

6.1 Helper functions

We first create some proto-plot function for the descriptive plots.

```
#' @param response.to.consider character string, one of the three responses to
#' consider
#' @param xaxis.var character string, either Intervention.duration or Birthweigth_gram
#' the variable against which we plot the response.to.consider on the x-axis
#' @param pop.to.consider character string, either "PP" or "mITT" the population
#' in which the model should be fitted
gg.proto.descriptive <- function(response.to.consider,
                                 xaxis.var,
                                 pop.to.consider){

  d.tmp1 <- d.folic.acid.long %>%
    mutate(
      any.resp.missing = rowSums(is.na(select(.,
                                              InfantmeanBodyWeight,
                                              InfantmeanBodyLength,
                                              InfantmeanHeadCM))) > 0) %>%
    filter(any.resp.missing == FALSE) %>%
    filter(Visit.fac == "FUV") %>%
    filter(Treatmentgroup.fac.renamed.reord %in% c("5-MTHF-Ca", "Folic acid")) %>%
    droplevels()

  ## We relevel the intervention variable, such that the coefficient means Folic acid -
  ## 5-MTHF-Ca
  d.tmp1 <-  d.tmp1 %>%
    mutate(Treatmentgroup.fac.renamed.reord =
             fct_relevel(d.tmp1$Treatmentgroup.fac.renamed.reord, "Folic acid"))


  ## We filter according to the PP or mITT population
  if(pop.to.consider == "PP"){
    d.tmp1 <- d.tmp1 %>%
      filter(PPPopulation.fac == "yes")
  }else if(pop.to.consider == "mITT"){
    d.tmp1 <- d.tmp1 %>%
      filter(ITTPopulation.fac == "yes")
  }else{
    return(paste0("Please select either PP or mITT as value to the
                  argument pop.to.consider."))
  }

  gg.descriptive <- d.tmp1 %>%
    ggplot() +
    aes(color = Treatmentgroup.fac.renamed.reord,
        group = Treatmentgroup.fac.renamed.reord) +
    geom_point(alpha = 1) +
    facet_wrap(~ ChildSex.fac) +
    aes(x = get(xaxis.var), y = get(response.to.consider)) +
```

```
    # theme(legend.position="none") +
    ggtitle(paste0(pop.to.consider, ": ",
                   response.to.consider, " ~ ", xaxis.var)) +
    theme(plot.title = element_text(size = 9),
          axis.text.x = element_text(angle = 45, hjust = 1))+
    geom_smooth(method = "loess", alpha = 0.5)


  return(gg.descriptive)
}
```

This function allows to create descriptive plots for the two populations and observations at follow-up.

```
#' @param pop.to.consider character string, either "PP" or "mITT" the population
#' in which the model should be fitted
plot.descriptive <- function(pop.to.consider){


  ## Mean body weight ----------------------------------------------------------


  ## Mean body weight vs Intervention duration
  gg.descriptive.weight.duration <- gg.proto.descriptive(
    response.to.consider = "InfantmeanBodyWeight",
    xaxis.var = "Intervention.duration",
    pop.to.consider = pop.to.consider) +
    theme(legend.position = "none") +
    labs(y = "Mean body weight (g)",
         x = "Intervention duration (d)",
         title = paste0("Mean body weight at 1 year of age \nplotted for intervention duration (",
                        pop.to.consider, ")"))

  ## Mean body weight vs Birth weight
  gg.descriptive.weight.birthweight <- gg.proto.descriptive(
    response.to.consider ="InfantmeanBodyWeight",
    xaxis.var = "Birthweight_gram",
    pop.to.consider = pop.to.consider) +
    theme(legend.position = "none") +
    labs(y = "Mean body weight (g)",
         x = "Birth weight (g)",
         title = paste0("Mean body weight at 1 year of age \nplotted for birth weight (",
                        pop.to.consider, ")"))


  ## Mean body length ----------------------------------------------------------


  ## Mean body length vs Intervention duration
  gg.descriptive.length.duration <- gg.proto.descriptive(
    response.to.consider = "InfantmeanBodyLength",
    xaxis.var = "Intervention.duration",
    pop.to.consider = pop.to.consider) +
    theme(legend.position = "none") +
    labs(y = "Mean body length (cm)",
```

```
       x = "Intervention duration (d)",
       title = paste0("Mean body length at 1 year of age \nplotted for intervention duration (",
                      pop.to.consider, ")"))

gg.descriptive.length.birthweight <- gg.proto.descriptive(
  response.to.consider = "InfantmeanBodyLength",
  xaxis.var = "Birthweight_gram",
  pop.to.consider = pop.to.consider) +
  theme(legend.position = "none") +
  labs(y = "Mean body length (cm)",
       x = "Birth weight (g)",
       title = paste0("Mean body length at 1 year of age \nplotted for birth weight (",
                      pop.to.consider, ")"))



## Mean head circumference ----------------------------------------------------


gg.descriptive.headcm.duration <- gg.proto.descriptive(
  response.to.consider = "InfantmeanHeadCM",
  xaxis.var = "Intervention.duration",
  pop.to.consider = pop.to.consider) +
  theme(legend.position = "none") +
  labs(y = "Mean head circumference (cm)",
       x = "Intervention duration (d)",
       title = paste0("Mean head circumference at 1 year of age \nplotted for intervention duration (
                      pop.to.consider, ")"))

gg.descriptive.headcm.birthweight <- gg.proto.descriptive(
  response.to.consider = "InfantmeanHeadCM",
  xaxis.var = "Birthweight_gram",
  pop.to.consider = pop.to.consider) +
  theme(legend.position = "none") +
  labs(y = "Mean head circumference (cm)",
       x = "Birth weight (g)",
       title = paste0("Mean head circumference at 1 year of age \nplotted for birth weight (",
                      pop.to.consider, ")"))



## ---------------------------------------------------------------------------

# Extract the legend from one of the plots
test.plot <-  gg.proto.descriptive(
  response.to.consider ="InfantmeanBodyWeight",
  xaxis.var = "Intervention.duration",
  pop.to.consider = pop.to.consider) +
  theme(legend.position = "bottom") +
  labs(colour = "Intervention group")

common.legend <- ggpubr::get_legend(test.plot)
```

```
  ## Arrange plots to one plot
  plot.grid <- plot_grid(
    gg.descriptive.weight.duration,
    gg.descriptive.weight.birthweight,
    gg.descriptive.length.duration,
    gg.descriptive.length.birthweight,
    gg.descriptive.headcm.duration,
    gg.descriptive.headcm.birthweight,
    ncol = 2,
    align = "v"
  )


  # Combine the grid and the common legend
  final.plot <- plot_grid(
    plot.grid,
    common.legend,
    ncol = 1,
    rel_heights = c(0.9, 0.1) # Allocate space for the legend
  )

  caption_text <- paste0("Descriptive plot",
                         " in population ", pop.to.consider, ".")

  # Add the caption below the final plot
  final.plot.with.caption <- ggdraw() +
    draw_plot(final.plot, x = 0, y = 0.05, width = 1, height = 0.95) +
    draw_label(caption_text, x = 0.5, y = 0.02, hjust = 0.5, size = 14)


  return(final.plot.with.caption)

}
```

We first write a function that creates boxplots for the three anthropometric variables at the FUV for the population of choice, PP or mITT.

```
plot.boxplots <- function(pop.to.consider){

  d.tmp.boxplots <- d.folic.acid.long %>%
    mutate(any.resp.missing = rowSums(is.na(select(., InfantmeanBodyWeight,
                                                   InfantmeanBodyLength,
                                                   InfantmeanHeadCM))) > 0) %>%
    filter(any.resp.missing == FALSE) %>%
    filter(Visit.fac == "FUV")

  ## We filter according to the PP or mITT population
  if (pop.to.consider == "PP") {
    d.tmp.boxplots <- d.tmp.boxplots %>%
      filter(PPPopulation.fac == "yes")
  } else if (pop.to.consider == "mITT") {
    d.tmp.boxplots <- d.tmp.boxplots %>%
      filter(ITTPopulation.fac == "yes")
  }
```

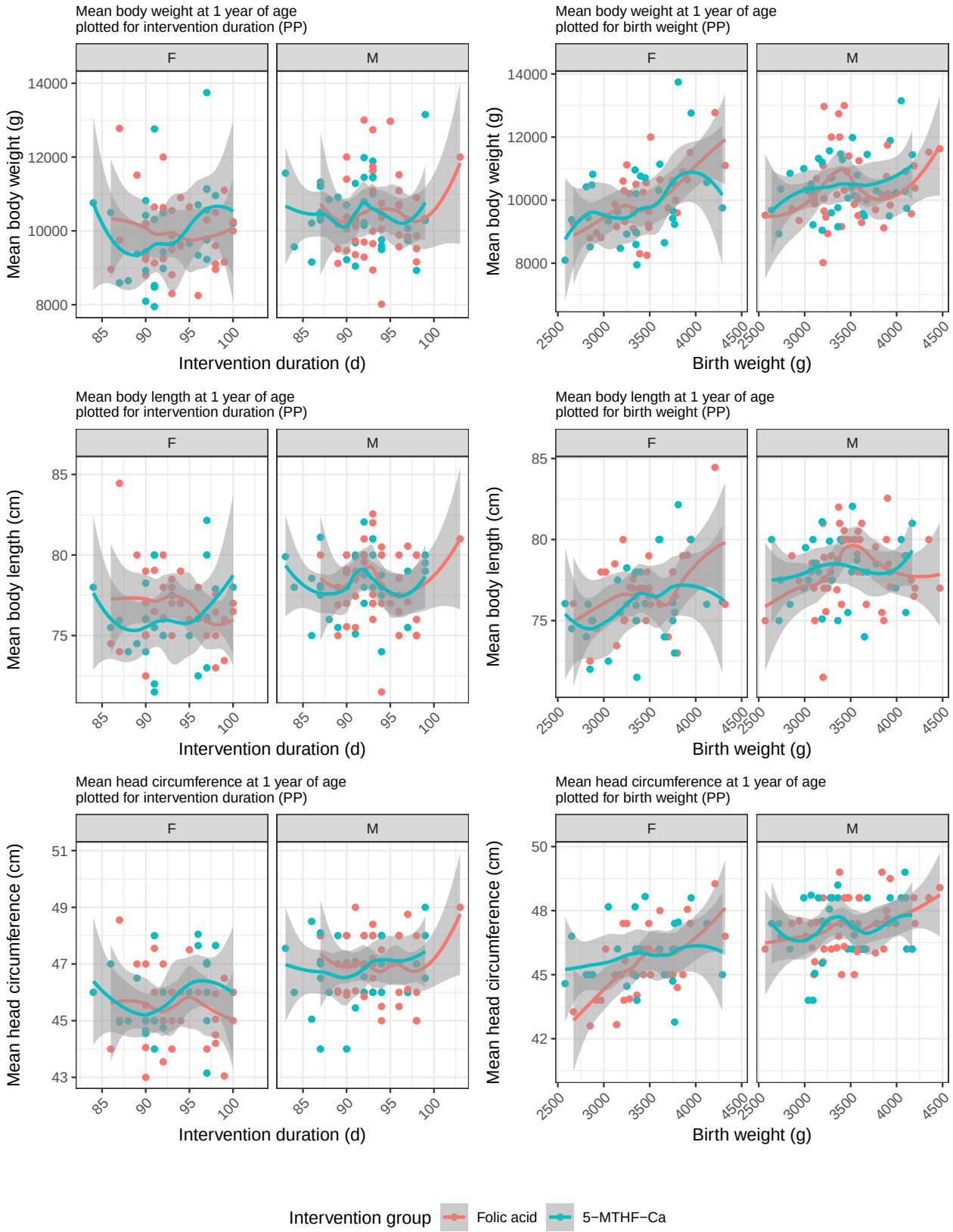
```
  box.weight <- ggplot(d.tmp.boxplots) +
    aes(y = `InfantmeanBodyWeight`,
        x = `Treatmentgroup.fac.renamed.reord`) +
    geom_boxplot() +
    labs(title = paste0("Body weight \nat FUV (", pop.to.consider, ")"),
         x = "",
         y = "Mean Body Weight (g)") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
          plot.title = element_text(size = 10))

  box.length <- ggplot(d.tmp.boxplots) +
    aes(y = `InfantmeanBodyLength`,
        x = `Treatmentgroup.fac.renamed.reord`) +
    geom_boxplot() +
    labs(title = paste0("Body length \nat FUV (",  pop.to.consider, ")"),
         x = "",
         y = "Mean Body Length (cm)") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
          plot.title = element_text(size = 10))

  box.head <- ggplot(d.tmp.boxplots) +
    aes(y = `InfantmeanHeadCM`,
        x = `Treatmentgroup.fac.renamed.reord`) +
    geom_boxplot() +
    labs(title = paste0("Head circumference \nat FUV (", pop.to.consider, ")"),
         x = "",
         y = "Mean Head Circumference (cm)") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
          plot.title = element_text(size = 10))

  grid.arrange(box.weight, box.length, box.head, ncol = 3, heights = 0.3)

}
```

6.2 PP-Population

We create the descriptive plots of the three responses for the PP-population.

```
plot.descriptive(pop.to.consider = "PP")
```

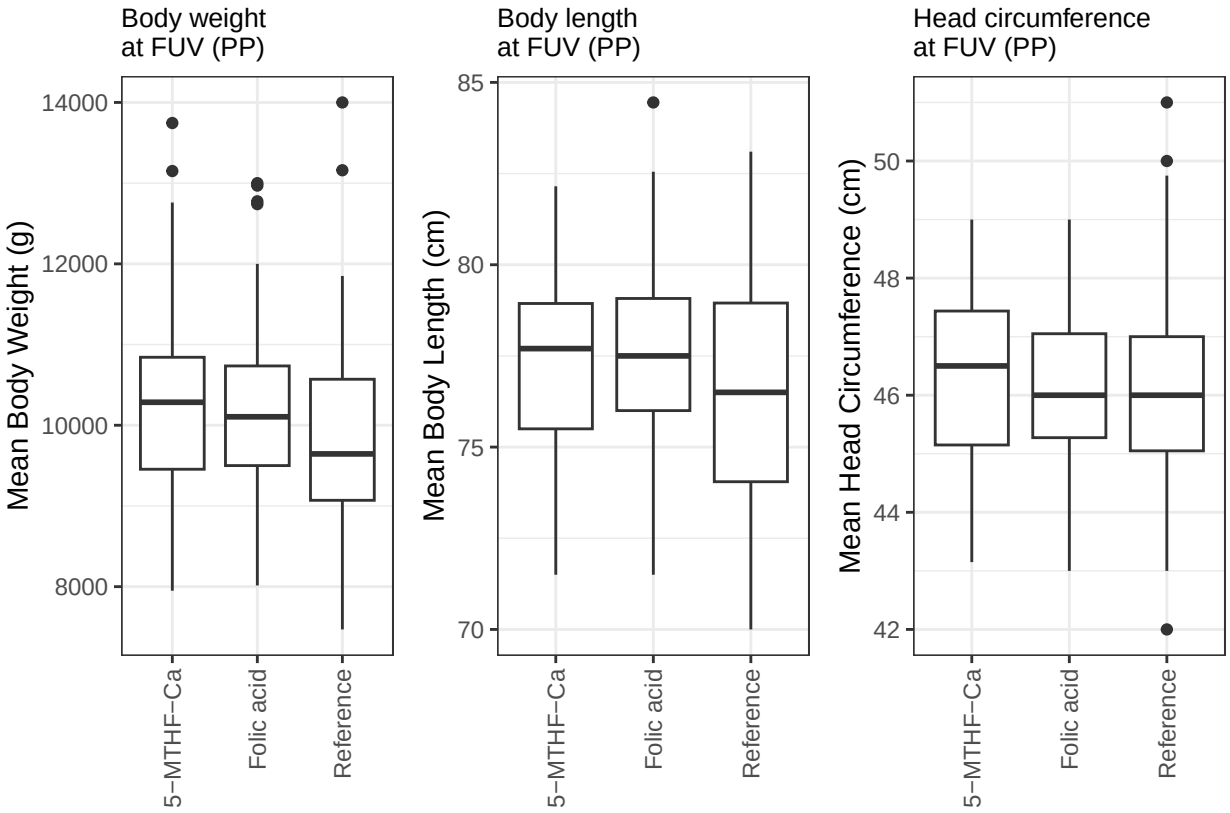
```
‘geom_smooth()‘ using formula = ’y ~ x’
‘geom_smooth()‘ using formula = ’y ~ x’
‘geom_smooth()‘ using formula = ’y ~ x’
‘geom_smooth()‘ using formula = ’y ~ x’
‘geom_smooth()‘ using formula = ’y ~ x’
‘geom_smooth()‘ using formula = ’y ~ x’
‘geom_smooth()‘ using formula = ’y ~ x’
```



Descriptive plot in population PP.

We create the boxplots of the three responses for the PP-population.

```
plot.boxplots(pop.to.consider = "PP")
```

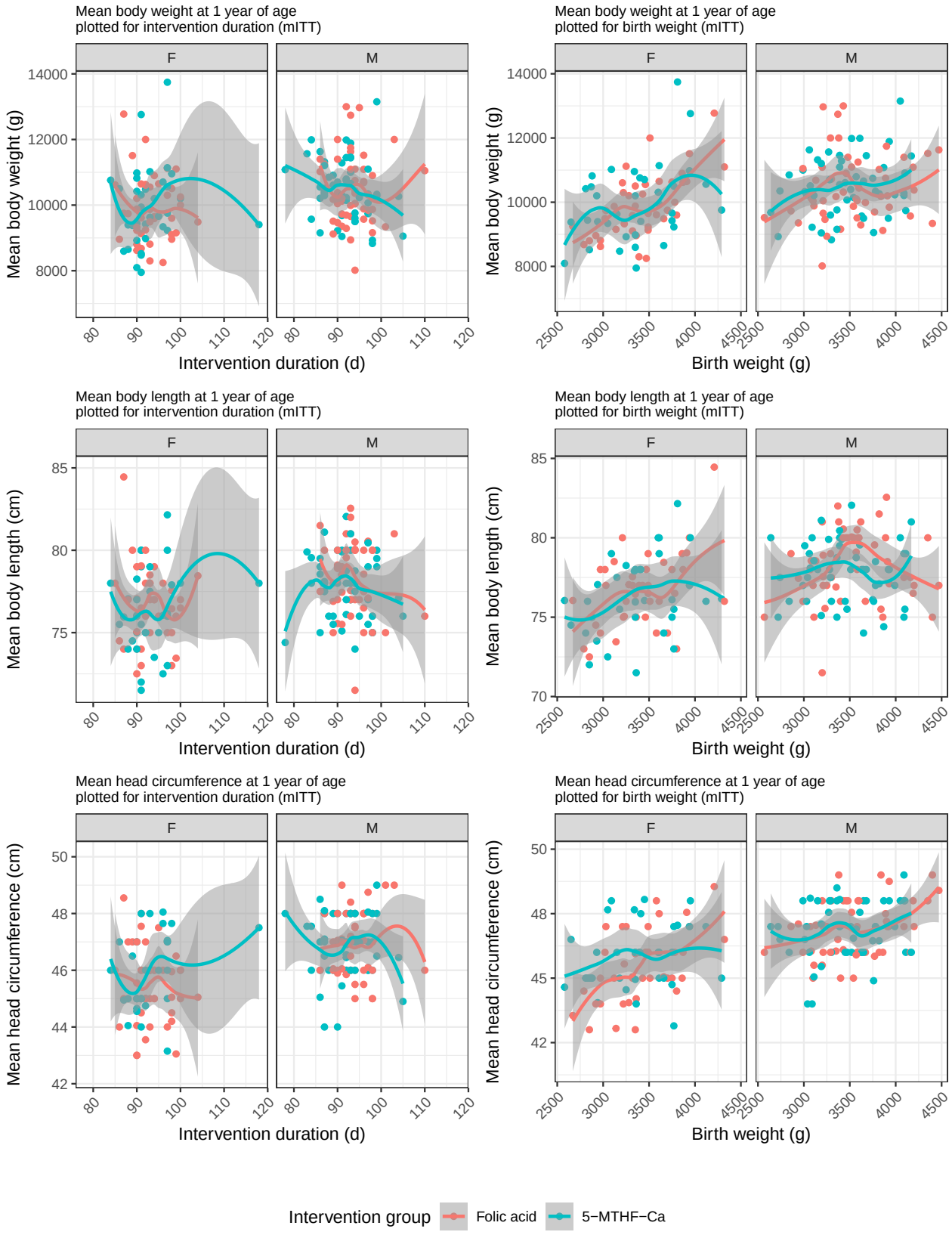


6.3 mITT-Population

We create the descriptive plots of the three responses for the mITT-population.

```
plot.descriptive(pop.to.consider = "mITT")
```

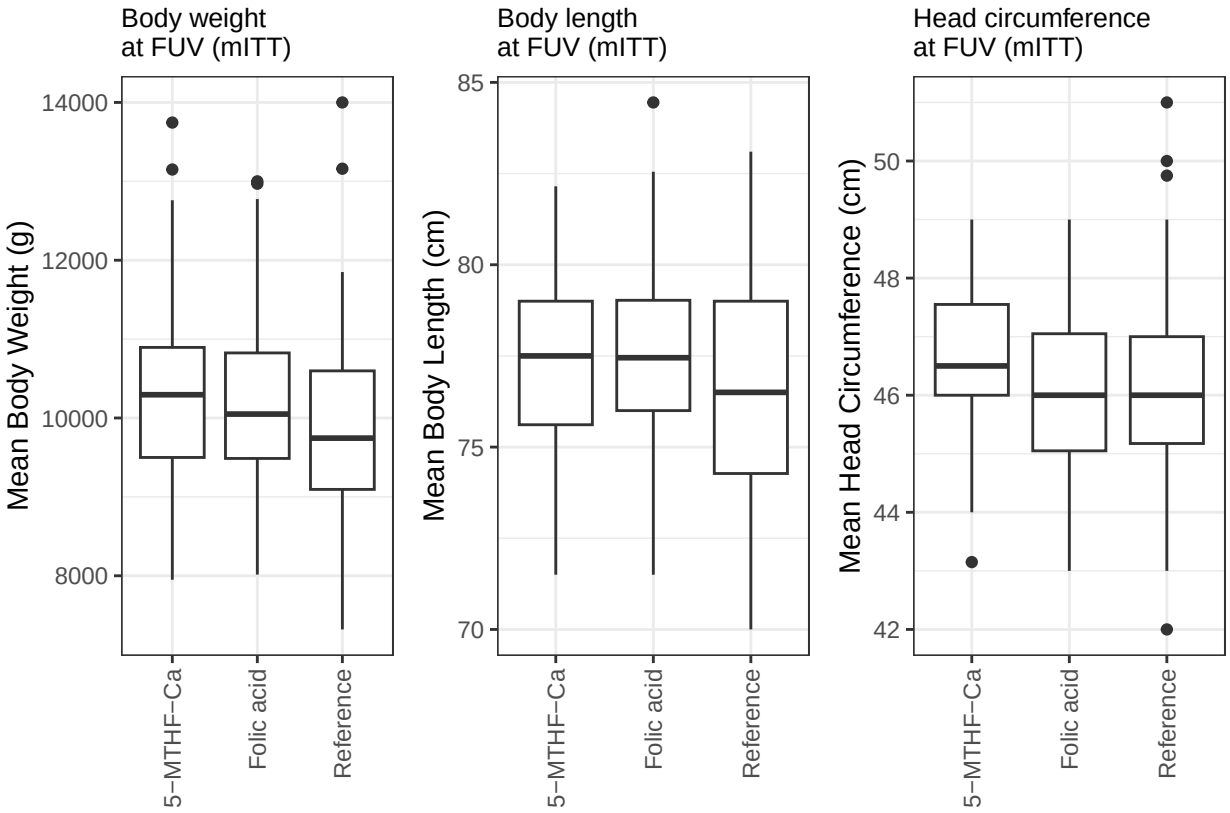
```
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
```



Descriptive plot in population mITT.

We create the boxplots of the three responses for the mITT-population.

```
plot.boxplots(pop.to.consider = "mITT")
```



7 Session Information

```
options(width = 80)
sessionInfo()
```

```
R version 4.4.2 (2024-10-31 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 11 x64 (build 26200)

Matrix products: default


locale:
[1] LC_COLLATE=English_Switzerland.utf8  LC_CTYPE=English_Switzerland.utf8
[3] LC_MONETARY=English_Switzerland.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Switzerland.utf8

time zone: Europe/Zurich
tzcode source: internal

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] hypoRF_1.0.1     car_3.1-3        carData_3.0-5    gridExtra_2.3
 [5] cowplot_1.1.3    ranger_0.16.0    ggpubr_0.6.0     readxl_1.4.3
 [9] mgcViz_0.1.11    qgam_1.3.4       mgcv_1.9-1       nlme_3.1-166
[13] magrittr_2.0.3   forcats_1.0.0    stringr_1.5.1    purrr_1.0.2
[17] readr_2.1.5      tidyr_1.3.1      tibble_3.2.1     tidyverse_2.0.0
[21] gtsummary_2.0.3  lubridate_1.9.3  kableExtra_1.4.0 ggbeeswarm_0.7.2
[25] ggplot2_3.5.1    dplyr_1.1.4      groundhog_3.2.0  knitr_1.48

loaded via a namespace (and not attached):
 [1] rlang_1.1.4         matrixStats_1.4.1  compiler_4.4.2     systemfonts_1.1.0
 [5] vctrs_0.6.5         pkgconfig_2.0.3    crayon_1.5.3       fastmap_1.2.0
 [9] backports_1.5.0     labeling_0.4.3     utf8_1.2.4         promises_1.3.0
[13] rmarkdown_2.28      tzdb_0.4.0         nloptr_2.1.1       tinytex_0.53
[17] xfun_0.48           later_1.3.2        broom_1.0.7        parallel_4.4.2
[21] R6_2.5.1            stringi_1.8.4      RColorBrewer_1.1-3 GGally_2.2.1
[25] boot_1.3-31         cellranger_1.1.0   Rcpp_1.0.13        iterators_1.0.14
[29] httpuv_1.6.15       Matrix_1.7-0       splines_4.4.2      timechange_0.3.0
[33] tidyselect_1.2.1    rstudioapi_0.17.0  abind_1.4-8        yaml_2.3.10
[37] viridis_0.6.5       doParallel_1.0.17  codetools_0.2-20   miniUI_0.1.1.1
[41] lattice_0.22-6      plyr_1.8.9         shiny_1.9.1        withr_3.0.1
[45] evaluate_1.0.1      ggstats_0.7.0      xml2_1.3.6         pillar_1.9.0
[49] KernSmooth_2.23-24  foreach_1.5.2      generics_0.1.3     hms_1.1.3
[53] munsell_0.5.1       scales_1.3.0       minqa_1.2.8        xtable_1.8-4
[57] gamm4_0.2-6         glue_1.8.0         tools_4.4.2        lme4_1.1-35.5
[61] ggsignif_0.6.4      colorspace_2.1-1   beeswarm_0.4.0     vipor_0.4.7
[65] Formula_1.2-5       cli_3.6.3          fansi_1.0.6        viridisLite_0.4.2
[69] svglite_2.1.3       gtable_0.3.5       rstatix_0.7.2      digest_0.6.37
[73] farver_2.1.2        htmltools_0.5.8.1  lifecycle_1.0.4    mime_0.12
[77] MASS_7.3-61
```