TNDS\_2021\_Comp\_Across\_Langs

Kees Schipper

1/19/2021

## Set up libraries and working directory

knitr::opts\_chunk$set(  
 echo = TRUE,  
 message = FALSE,  
 warnings = FALSE  
)  
setwd("C:/Users/keess/Box/TNDS Workshop/R")  
  
library(tidyverse)  
library(readxl)  
library(haven)  
library(plotrix)  
library(expss)  
library(moments)

## Read in data

# load("Input\_Data/workshop\_data.RData")  
# write\_csv(x = workshop\_data, "Input\_Data/workshop\_data.csv")  
getwd() # working directory is different in Rmarkdown than in an R script

## [1] "C:/Users/keess/Box/TNDS Workshop/R/R scripts"

# load("../../data for workshop/workshop\_data.RData")  
df <- read\_csv("../../data for workshop/workshop\_data.csv")

## Describe, view, and edit data once imported

names(df) # variable names in df

## [1] "study\_id" "study\_arm" "agedays" "agemonths" "gender"   
## [6] "cgage" "edu" "av\_weight" "av\_len" "av\_muac"   
## [11] "\_zwei" "\_zwfl" "hfias\_score" "hfias\_cat" "ageyears"

nrow(df) # number of rows in df

## [1] 2653

ncol(df) # number of columns in df

## [1] 15

str(df) # structure

## tibble [2,653 x 15] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ study\_id : num [1:2653] 1 2 3 4 5 6 7 8 9 10 ...  
## $ study\_arm : num [1:2653] 1 1 1 1 1 1 1 1 1 1 ...  
## $ agedays : num [1:2653] 399 477 1506 429 527 ...  
## $ agemonths : num [1:2653] 13.1 15.7 49.5 14.1 17.3 ...  
## $ gender : num [1:2653] 1 1 1 1 0 0 0 1 0 0 ...  
## $ cgage : num [1:2653] 21 25 46 29 20 25 23 20 35 25 ...  
## $ edu : num [1:2653] 0 3 0 0 0 0 3 3 0 1 ...  
## $ av\_weight : num [1:2653] 7.14 7.2 12.79 6.76 6.93 ...  
## $ av\_len : num [1:2653] 70.6 70.6 103.3 68.2 72.3 ...  
## $ av\_muac : num [1:2653] 11.8 11.9 11.9 12 12 ...  
## $ \_zwei : num [1:2653] -2.95 -3.33 -2.06 -3.59 -3.19 -2.39 -2.91 -2.6 -2.89 -2.55 ...  
## $ \_zwfl : num [1:2653] -2.26 -2.16 -2.89 -2.14 -2.53 -1.55 -1.96 -1.45 -2.15 -1.63 ...  
## $ hfias\_score: num [1:2653] 16 21 15 0 10 12 15 16 11 26 ...  
## $ hfias\_cat : num [1:2653] 1 1 1 4 2 2 1 1 2 1 ...  
## $ ageyears : num [1:2653] 1.09 1.31 4.12 1.17 1.44 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. study\_id = col\_double(),  
## .. study\_arm = col\_double(),  
## .. agedays = col\_double(),  
## .. agemonths = col\_double(),  
## .. gender = col\_double(),  
## .. cgage = col\_double(),  
## .. edu = col\_double(),  
## .. av\_weight = col\_double(),  
## .. av\_len = col\_double(),  
## .. av\_muac = col\_double(),  
## .. `\_zwei` = col\_double(),  
## .. `\_zwfl` = col\_double(),  
## .. hfias\_score = col\_double(),  
## .. hfias\_cat = col\_double(),  
## .. ageyears = col\_double()  
## .. )

## find all unique values of gender  
unique(df$gender)

## [1] 1 0 -99

# find the number of unique values of age in days  
n\_distinct(df$agedays)

## [1] 690

# the number of rows with complete observations  
sum(complete.cases(df))

## [1] 2623

# the number of missing observations with agedays  
sum(is.na(df$agedays))

## [1] 13

## lapply can take a function (like unique), and apply it to all columns of a dataframe (df)  
unique\_vals <- lapply(df, unique)  
distinct\_vals <- lapply(df, n\_distinct)  
## missing vals for each column:  
missing <- lapply(df, function(x) sum(is.na(x)))

## Generate and label variables and observations

# creating variables in base R: do simple operations on already existing variables  
df$ageyears <- df$agedays/365.25  
  
# you can also create completely new variables, like this nonsensical one below  
# df$nonsense <- "this variable means nothing"  
  
# labelling categorical variables as factors  
df$gender <- factor(df$gender, levels = c(-99, 0, 1),   
 labels = c("not documented", "female", "male"))  
summary(df$gender)

## not documented female male   
## 3 1523 1127

# let's do the same with edu level  
df$edu <- factor(df$edu, levels = c(-99, 0, 1, 2, 3, 4, 5),  
 labels = c("not documented", "none", "some primary", "completed primary",  
 "some secondary", "completed secondary", "more than secondary"))  
summary(df$edu)

## not documented none some primary completed primary   
## 14 1419 574 59   
## some secondary completed secondary more than secondary   
## 563 21 3

## Get summary statistics

# simple base R summary function, giving min, mean, median, max, and Q1/Q3 for continuous variables, and frequency counts for categorical variables  
summary(df)

## study\_id study\_arm agedays agemonths   
## Min. : 1 Min. :1.000 Min. : 171.0 Min. : 5.622   
## 1st Qu.:2595 1st Qu.:2.000 1st Qu.: 231.0 1st Qu.: 7.595   
## Median :5115 Median :3.000 Median : 326.0 Median :10.718   
## Mean :4430 Mean :2.611 Mean : 403.1 Mean :13.254   
## 3rd Qu.:7624 3rd Qu.:4.000 3rd Qu.: 493.2 3rd Qu.:16.216   
## Max. :8500 Max. :4.000 Max. :1809.0 Max. :59.474   
## NA's :13 NA's :13   
## gender cgage edu   
## not documented: 3 Min. :-99.00 not documented : 14   
## female :1523 1st Qu.: 21.00 none :1419   
## male :1127 Median : 26.00 some primary : 574   
## Mean : 26.59 completed primary : 59   
## 3rd Qu.: 32.00 some secondary : 563   
## Max. : 80.00 completed secondary: 21   
## more than secondary: 3   
## av\_weight av\_len av\_muac \_zwei   
## Min. : 4.085 Min. : 53.15 Min. :11.50 Min. :-6.140   
## 1st Qu.: 5.955 1st Qu.: 63.55 1st Qu.:11.73 1st Qu.:-3.390   
## Median : 6.510 Median : 66.85 Median :11.97 Median :-2.870   
## Mean : 6.677 Mean : 68.03 Mean :11.97 Mean :-2.882   
## 3rd Qu.: 7.220 3rd Qu.: 71.45 3rd Qu.:12.20 3rd Qu.:-2.330   
## Max. :15.010 Max. :108.20 Max. :12.47 Max. : 1.260   
## NA's :1 NA's :3 NA's :17   
## \_zwfl hfias\_score hfias\_cat ageyears   
## Min. :-4.840 Min. : 0.000 Min. :1.0 Min. :0.4682   
## 1st Qu.:-2.270 1st Qu.: 1.000 1st Qu.:1.0 1st Qu.:0.6324   
## Median :-1.790 Median :10.000 Median :2.0 Median :0.8925   
## Mean :-1.799 Mean : 9.451 Mean :2.4 Mean :1.1037   
## 3rd Qu.:-1.320 3rd Qu.:14.000 3rd Qu.:3.0 3rd Qu.:1.3504   
## Max. : 1.370 Max. :27.000 Max. :4.0 Max. :4.9528   
## NA's :8 NA's :9 NA's :9 NA's :13

# individual summary functions. We need to use the na.rm option to remove NA values, because if we don't, the NA values will cause these aggregation functions to throw an error  
mean(df$agemonths, na.rm = T)

## [1] 13.254

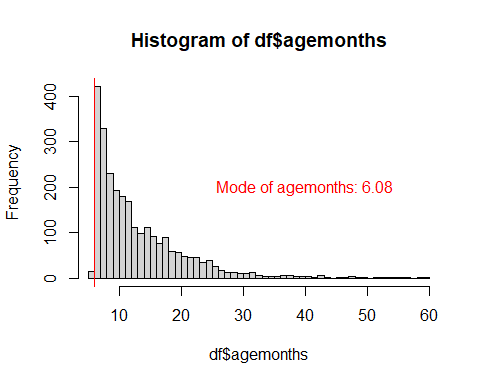
median(df$agemonths, na.rm = T)

## [1] 10.7178

sd(df$agemonths, na.rm = T)

## [1] 7.900913

# no explicit function for the mode of a vector, but R allows you to make functions to suit whatever operations your need!  
getmode <- function(x){  
 uniquev <- unique(x)  
 uniquev[which.max(tabulate(match(x, uniquev)))]  
}  
hist(df$agemonths, breaks = 50, xlim = c(min(df$agemonths, na.rm = T), max(df$agemonths, na.rm = T)))  
abline(v = getmode(df$agemonths), col = "red")  
text(x = 40, y = 200, paste0("Mode of agemonths: ", round(getmode(df$agemonths), 2)), col = "red")



## REMEMBER: getmode is not an actual R function, it is made up for the purposes of displaying R's function syntax  
  
# some other useful aggregation functions. You can think or range as returning the results of min and max to you but in one function call  
range(df$agemonths, na.rm = T)

## [1] 5.621912 59.473907

min(df$agemonths, na.rm = T)

## [1] 5.621912

max(df$agemonths, na.rm = T)

## [1] 59.47391

## skewness and kurtosis are from the "moments" package  
skewness(df$agemonths, na.rm = T)

## [1] 2.149153

kurtosis(df$agemonths, na.rm = T)

## [1] 9.280115

quantile(df$agemonths, probs = c(.10, .25, .50, .75, .90), na.rm = T)

## 10% 25% 50% 75% 90%   
## 6.542459 7.594512 10.717796 16.216421 22.921619

# ?quantile()

## Frequency table with row/column percentages

# to make a table, use the table function. You can then input two vectors that you want to cross-tabulate  
mytable <- table(df$gender, df$edu)  
mytable

##   
## not documented none some primary completed primary  
## not documented 0 2 0 0  
## female 8 820 336 31  
## male 6 597 238 28  
##   
## some secondary completed secondary more than secondary  
## not documented 1 0 0  
## female 315 11 2  
## male 247 10 1

# tidyverse function that does the same thing. Input your data frame (df), and the variable that you want to tabulate  
count(df, gender, sort = T)

## # A tibble: 3 x 2  
## gender n  
## <fct> <int>  
## 1 female 1523  
## 2 male 1127  
## 3 not documented 3

# can also tabulate by multiple variables  
count(df, gender, edu, sort = T)

## # A tibble: 16 x 3  
## gender edu n  
## <fct> <fct> <int>  
## 1 female none 820  
## 2 male none 597  
## 3 female some primary 336  
## 4 female some secondary 315  
## 5 male some secondary 247  
## 6 male some primary 238  
## 7 female completed primary 31  
## 8 male completed primary 28  
## 9 female completed secondary 11  
## 10 male completed secondary 10  
## 11 female not documented 8  
## 12 male not documented 6  
## 13 not documented none 2  
## 14 female more than secondary 2  
## 15 not documented some secondary 1  
## 16 male more than secondary 1

# only needed if we haven't already defined labels for our variables  
# rownames(mytable) = c("missing", "male", "female")  
# colnames(mytable) = c("missing", "none", "primary school", "secondary school",  
# "some high school", "some college", "graduate school")  
  
# back to our original, base R tables:   
# summarize table across rows with margin.table  
margin.table(mytable, 1)

##   
## not documented female male   
## 3 1523 1127

# summarize table across columns  
margin.table(mytable, 2)

##   
## not documented none some primary completed primary   
## 14 1419 574 59   
## some secondary completed secondary more than secondary   
## 563 21 3

# margin.table aggregates your data across the dimension that you provide (where 1 = sum across rows, and 2 = sum across columns)  
  
# calculate row percentages   
round(prop.table(mytable, 1), 2)

##   
## not documented none some primary completed primary  
## not documented 0.00 0.67 0.00 0.00  
## female 0.01 0.54 0.22 0.02  
## male 0.01 0.53 0.21 0.02  
##   
## some secondary completed secondary more than secondary  
## not documented 0.33 0.00 0.00  
## female 0.21 0.01 0.00  
## male 0.22 0.01 0.00

# calculate column percentages  
round(prop.table(mytable, 2), 2)

##   
## not documented none some primary completed primary  
## not documented 0.00 0.00 0.00 0.00  
## female 0.57 0.58 0.59 0.53  
## male 0.43 0.42 0.41 0.47  
##   
## some secondary completed secondary more than secondary  
## not documented 0.00 0.00 0.00  
## female 0.56 0.52 0.67  
## male 0.44 0.48 0.33

# prop.table calculates row and column percentages, based on the dimension you give it (1 = row percentages, 2 = column percentages).  
# the round function is just there to reduce the number of significant figures we keep

## Convert continuous to categorical variables with if/then statements

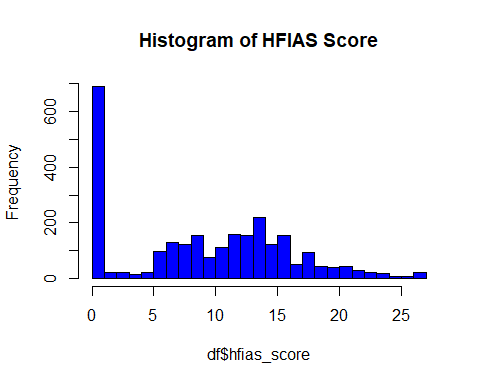
## tidyverse version. ifelse is a vectorized if/else statement, meaning it applies if/else logic across an entire column of a dataframe. R's native if/else logic can only be applied to one data point at a time, and therefore ifelse is often more efficient when working with data frames  
df\_cat <- df %>%  
 mutate(hfias\_cat = ifelse(hfias\_score < 1, 4,  
 ifelse(hfias\_score >= 1 & hfias\_score < 10, 3,  
 ifelse(hfias\_score >= 10 & hfias\_score < 14, 2,  
 ifelse(hfias\_score >= 14, 1, NA)))),  
 hfias\_cat = factor(hfias\_cat))  
  
  
## base r version. We are indexing all of the columns where our conditions are satisfied, and assigning them a category based on the condition. Im less of a fan of this version, because each of the assignments are separate, which leaves the chance that you may overwrite some values twice  
df$hfias\_cat[df$hfias\_score < 1] <- 4  
df$hfias\_cat[df$hfias\_score >= 1 & df$hfias\_score < 10] <- 3  
df$hfias\_cat[df$hfias\_score >= 10 & df$hfias\_score < 14] <- 2  
df$hfias\_cat[df$hfias\_score >= 14] <- 1  
  
## save your output as a permanent RData file with the save function  
save(df\_cat, file = "../output/output\_df.RData")  
  
## or save as a csv  
write.csv(df\_cat, file = "../output/output\_df.csv")  
  
## csv files are more memory intensive than RData files, so if you are working with large data sets, use RData if you are exclusively working in R

## Producing a histogram to inspect variables

n\_distinct(df$hfias\_score)

## [1] 29

## use breaks to specify the number of bins you want in your histogram.  
hist(df$hfias\_score, breaks = 29, main = "Histogram of HFIAS Score",  
 freq = TRUE, col = "blue")



# as each score has its own bin, this is technically a barplot, just made with the hist() function

## Comparison tests (t-test and Wilcoxon)

df\_ttest <- df[df$gender == "female" | df$gender == "male",]  
  
# simple unpaired t test  
t.test(df\_ttest$av\_len ~ df\_ttest$gender, conf.level = 0.95, paired = FALSE, var.equal = FALSE)

##   
## Welch Two Sample t-test  
##   
## data: df\_ttest$av\_len by df\_ttest$gender  
## t = -5.4411, df = 2337.4, p-value = 5.848e-08  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -1.7902305 -0.8416832  
## sample estimates:  
## mean in group female mean in group male   
## 67.46464 68.78060

# simple Wilcoxon rank sum  
wilcox.test(df\_ttest$av\_len ~ df\_ttest$gender, conf.int = T, conf.level = 0.95, exact = F,  
 correct = T)

##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: df\_ttest$av\_len by df\_ttest$gender  
## W = 736851, p-value = 8.454e-10  
## alternative hypothesis: true location shift is not equal to 0  
## 95 percent confidence interval:  
## -1.7500611 -0.9000049  
## sample estimates:  
## difference in location   
## -1.300048

## Measures of association (pearson vs. spearman)

mean(df$av\_len, na.rm = T)

## [1] 68.02908

# pearson correlation  
p\_corr <- cor(df$av\_weight, df$av\_len, use = "complete.obs", method = c("pearson"))  
# spearman correlation  
s\_corr <- cor(df$av\_weight, df$av\_len, use = "complete.obs", method = c("spearman"))  
p\_corr

## [1] 0.9449736

s\_corr

## [1] 0.9309364

## Simple Linear Regression

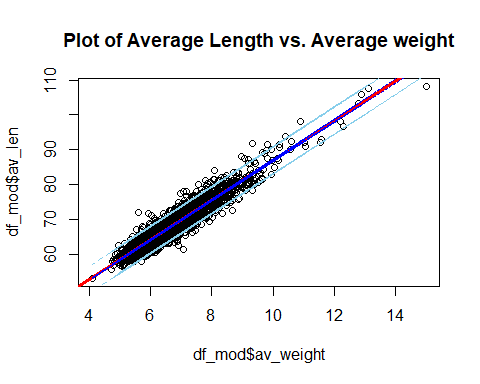
## using complete observations for the sake of simplicity  
df\_mod <- df[complete.cases(df),]  
  
# model of average length compared to weight  
length\_model <- lm(av\_len ~ av\_weight, data = df\_mod)  
summary(length\_model) # get description of model

##   
## Call:  
## lm(formula = av\_len ~ av\_weight, data = df\_mod)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -8.9180 -1.2868 -0.0845 1.2361 10.2290   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 30.05976 0.25404 118.3 <2e-16 \*\*\*  
## av\_weight 5.68820 0.03765 151.1 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.967 on 2621 degrees of freedom  
## Multiple R-squared: 0.897, Adjusted R-squared: 0.897   
## F-statistic: 2.283e+04 on 1 and 2621 DF, p-value: < 2.2e-16

# calculate prediction intervals for your model  
pred\_ints <- predict(length\_model, interval = "prediction", level = 0.95)

## Warning in predict.lm(length\_model, interval = "prediction", level = 0.95): predictions on current data refer to \_future\_ responses

# calculate confidence intervals as well  
conf\_ints <- predict(length\_model, interval = "confidence", level = 0.95)  
  
# plot model with prediction and confidence intervals, along with model fit  
plot(df\_mod$av\_weight, df\_mod$av\_len, main = "Plot of Average Length vs. Average weight")  
abline(length\_model, col = "red", lwd = 3)  
lines(df\_mod$av\_weight, pred\_ints[,2], col = "skyblue", lty = 2)  
lines(df\_mod$av\_weight, pred\_ints[,3], col = "skyblue", lty = 2)  
lines(df\_mod$av\_weight, conf\_ints[,2], col = "blue", lty = 3)  
lines(df\_mod$av\_weight, conf\_ints[,3], col = "blue", lty = 3)



## Multivariable Linear Regression

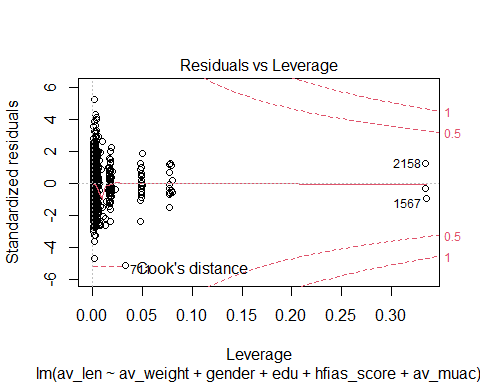
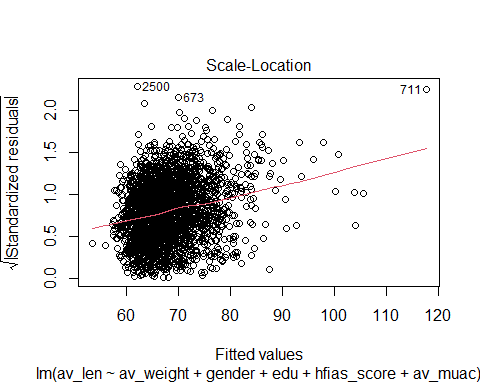
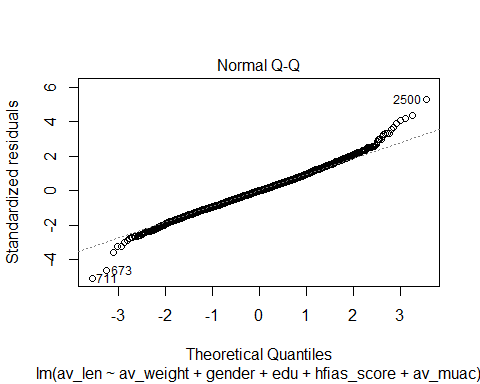
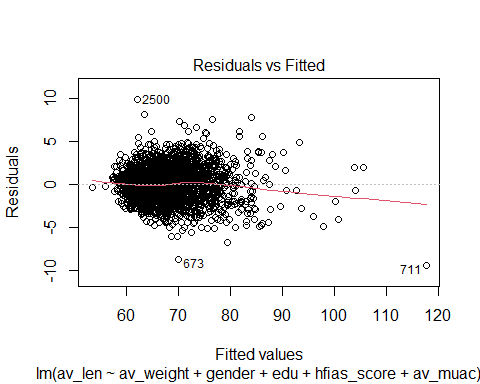
library(car)  
library(MASS)  
library(corrplot)  
  
# if you want to look at the correlations between variables  
# corrplot(cor(df\_mod[c("av\_len", "av\_weight", "hfias\_score", "av\_muac")], method = "pearson"))  
  
mv\_length\_model <- lm(av\_len ~ av\_weight + gender + edu + hfias\_score + av\_muac, data = df\_mod)  
summary(mv\_length\_model)

##   
## Call:  
## lm(formula = av\_len ~ av\_weight + gender + edu + hfias\_score +   
## av\_muac, data = df\_mod)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -9.4016 -1.1998 -0.0151 1.1311 9.8783   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 54.461058 1.741776 31.268 < 2e-16 \*\*\*  
## av\_weight 5.898436 0.038474 153.311 < 2e-16 \*\*\*  
## gendermale -0.321876 0.075036 -4.290 1.85e-05 \*\*\*  
## edunone 0.586457 0.524135 1.119 0.263   
## edusome primary 0.630966 0.528072 1.195 0.232   
## educompleted primary 0.715381 0.577402 1.239 0.215   
## edusome secondary 0.601719 0.528418 1.139 0.255   
## educompleted secondary 0.685406 0.664244 1.032 0.302   
## edumore than secondary 0.413619 1.204779 0.343 0.731   
## hfias\_score 0.007117 0.005307 1.341 0.180   
## av\_muac -2.200577 0.143265 -15.360 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.88 on 2612 degrees of freedom  
## Multiple R-squared: 0.9063, Adjusted R-squared: 0.9059   
## F-statistic: 2527 on 10 and 2612 DF, p-value: < 2.2e-16

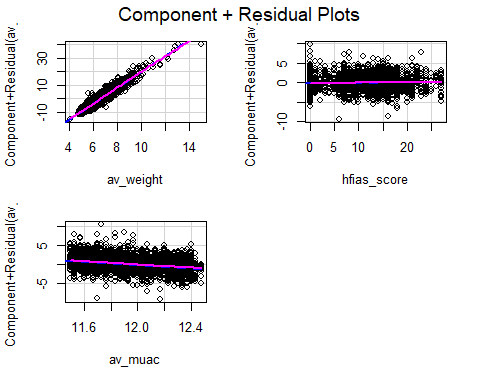
## to check linear model assumptions  
vif(mv\_length\_model) # variance inflation factor

## GVIF Df GVIF^(1/(2\*Df))  
## av\_weight 1.144008 1 1.069583  
## gender 1.021336 1 1.010612  
## edu 1.035453 6 1.002907  
## hfias\_score 1.032338 1 1.016041  
## av\_muac 1.119802 1 1.058207

plot(mv\_length\_model) # plots residuals vs fitted, normal q-q, cook's distance, and std residuals



crPlots(mv\_length\_model, terms = ~av\_weight + hfias\_score + av\_muac, smooth = T)



# shows the smoothed relationship of each variable with the outcome of interest