STAT202 Assignment 4: Variable selection

David Ewing

Due on 1 pm 21st August

# Introduction

This document contains the analysis for Assignment 3. It explores the aquatic\_toxicity dataset using multiple linear regression to understand the relationships between various predictors and the response variable LC50.

# Step 0: setup

loading libraries:

# repository and seed  
options(repos = c(CRAN = "https://cloud.r-project.org"))  
set.seed(82171165)  
  
# libraries  
  
 library(conflicted)  
 library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.1 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.4 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2

suppressPackageStartupMessages({  
 library(performance)  
 library(GGally)  
 library(flextable)  
 library(broom)  
 library(skimr)  
 library(data.table)  
 library(lmtest)  
 library(leaps)  
 library(caret)  
 library(caTools)  
 library(Metrics)  
 library(car)  
 library(scales)  
library(patchwork)  
})  
  
# conflicts  
conflict\_prefer("filter", "dplyr")

## [conflicted] Will prefer dplyr::filter over any other package.

conflict\_prefer("select", "dplyr")

## [conflicted] Will prefer dplyr::select over any other package.

# Step 1: Load dataset

local <- "../data/kungsan\_full\_local.csv"  
url <- "http://stats.apiolaza.net/data/kungsan\_full.csv"  
  
  
if (file.exists(local)) { #check   
 kungsan <- read.csv(local) #read  
} else {  
 kungsan <- read.csv(url) #fetch  
 write.csv(kungsan, file = local, row.names = FALSE) #write  
}  
  
skim\_kungsan <- skim(kungsan) |>  
 select(skim\_variable, n\_missing)  
  
skim\_kungsan

## # A tibble: 4 × 2  
## skim\_variable n\_missing  
## <chr> <int>  
## 1 sex 0  
## 2 height 0  
## 3 weight 0  
## 4 age 0

head(kungsan)

## height weight age sex  
## 1 151.765 47.82561 63 male  
## 2 139.700 36.48581 63 female  
## 3 136.525 31.86484 65 female  
## 4 156.845 53.04191 41 male  
## 5 145.415 41.27687 51 female  
## 6 163.830 62.99259 35 male

# Step 2: Examine the Dataset

set.seed(82171165) # reproducibility  
  
kungsan <- kungsan |>  
 mutate(weight2 = weight^2) |> # mutate weight2  
 mutate(sex = factor(sex)) |> # mutate sex factor  
 filter(age >= 12) # Filter 12 years or older  
  
 my\_kungsan <- kungsan |> sample\_n(100) #sample

Purpose of the transformations:

* Filtering (age >= 12): Excludes individuals younger than 12, focusing on an adult or older age group for more meaningful relationships between height, weight, and other variables.
* Transformation (weight2): The square of weight was created to allow for testing of potential quadratic relationships between weight and the response variable (height) in later modelling steps.
* Random Sampling (sample\_n(100)): A random subset of 100 observations (my\_kungsan) was selected to simplify visualisation and analysis while ensuring representativeness.

summary(kungsan)

## height weight age sex weight2   
## Min. :118.0 Min. :18.26 Min. :12.00 female:221 Min. : 333.3   
## 1st Qu.:146.1 1st Qu.:37.93 1st Qu.:23.00 male :192 1st Qu.:1438.8   
## Median :152.4 Median :43.40 Median :35.00 Median :1883.8   
## Mean :152.0 Mean :42.59 Mean :37.18 Mean :1889.1   
## 3rd Qu.:159.4 3rd Qu.:48.48 3rd Qu.:49.00 3rd Qu.:2350.1   
## Max. :179.1 Max. :62.99 Max. :88.00 Max. :3968.1

summary(my\_kungsan)

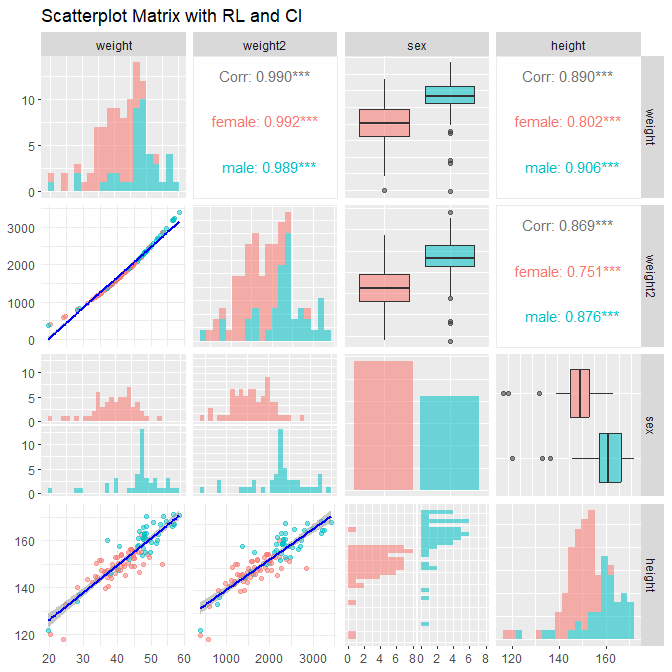
## height weight age sex weight2   
## Min. :118.0 Min. :19.62 Min. :12.00 female:58 Min. : 384.9   
## 1st Qu.:147.3 1st Qu.:37.92 1st Qu.:23.75 male :42 1st Qu.:1438.3   
## Median :153.0 Median :43.98 Median :33.50 Median :1934.6   
## Mean :152.7 Mean :42.81 Mean :37.25 Mean :1894.1   
## 3rd Qu.:160.0 3rd Qu.:48.12 3rd Qu.:49.12 3rd Qu.:2315.2   
## Max. :171.1 Max. :58.46 Max. :75.90 Max. :3417.2

# Step 3: Scatterplot Matrix

# function = add regression line and confidence interval  
add\_rlci <- function(data, mapping, ...) {  
 ggplot(data = data, mapping = mapping) +  
 geom\_point(alpha = 0.5) + # Scatterplot points  
 geom\_smooth(method = "lm", se = TRUE, colour = "blue", ...) + # rl ci  
 theme\_minimal()  
}  
  
add\_contour <- function(data, mapping, ...) {  
 ggplot(data = data, mapping = mapping) +  
 geom\_density2d(aes(colour = ..level..), ...) +  
 theme\_minimal()  
}  
upper = list(continuous = add\_contour)  
  
  
# Scatterplot   
kung\_splot1 <- ggpairs(  
 my\_kungsan,  
 columns = c("weight", "weight2", "sex", "height"),  
 lower = list(continuous = add\_rlci), #lwoer triangle  
 diag = list(continuous = wrap("barDiag", bins = 20)),   
# upper = list(continuous = add\_contour),  
 upper = list(continuous = "cor"),   
 aes(colour = sex, alpha = 0.7)  
) +   
 labs(title = "Scatterplot Matrix with RL and CI")

kung\_splot1

## `geom\_smooth()` using formula = 'y ~ x'  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `geom\_smooth()` using formula = 'y ~ x'  
## `geom\_smooth()` using formula = 'y ~ x'  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



: Purpose: The scatterplot matrix was created to explore the relationships between variables (height, weight, weight2, and sex) before modelling:

* Lower Triangle: Scatterplots with regression lines and confidence intervals (add\_rlci) allowed visualisation of potential linear or non-linear relationships.
* Diagonal: Histograms showed the distributions of variables, confirming normality or identifying skewness.
* Upper Triangle: Left blank (“blank”) to focus was on the lower triangle.

As this visual does not provide any satisfying conclusion to the model, additional investigation is required, which I have performed below.

# Load necessary libraries  
  
  
# args  
  
myk1 <- as.formula("height ~ weight")  
myk2 <- as.formula("height ~ weight + I(weight^2)")  
myk3 <- as.formula("height ~ weight + sex")  
myk4 <- as.formula("height ~ weight + I(weight^2) + sex")  
args <- list(myk1,myk2,myk3,myk4)  
n\_args <- length(args)  
  
models <- vector("list", n\_args) # Pre-define as a list  
for (i in seq\_along(args)){  
 models[[i]] <- lm(args[[i]], data = my\_kungsan)  
}  
  
# VIF for models with more than 1 predictor  
calc\_vif <- function(model) {  
 if (length(coef(model)) > 2) { # More than one predictor (Bo + B1 + 1)  
 return(round(max(vif(model)),6)) # max VIF  
 } else {  
 return("--NA--") # NA for 1 predictor  
 }  
}  
  
#RMSE simplification  
  
calc\_rmse <- function(args,numbr) {  
 train(args, data = my\_kungsan, method = "lm",  
 trControl = trainControl(method = "cv", number = numbr))$results$RMSE  
}  
# metrics  
  
adj\_r2 <- numeric(n\_args) # Adjusted R^2  
bp\_p <- numeric(n\_args) # Breusch-Pagan p-values  
shapiro\_p <- numeric(n\_args) # Shapiro-Wilk p-values  
cooks\_max <- numeric(n\_args) # Maximum Cook's Distance  
vif\_vals <- numeric(n\_args) #   
rmse <- numeric(n\_args) #  
  
set.seed(82171165)  
for (i in seq\_along(args)) {  
 adj\_r2[i] <- summary(models[[i]])$adj.r.squared #Adjusted R^2   
 bp\_p[i] <- bptest(models[[i]])$p.value #Homoscedasticity  
 shapiro\_p <- shapiro.test(models[[i]]$residuals)$p.value #Res Normality  
 cooks\_max[i] <- max(cooks.distance(models[[i]])) #Cook's Distance  
 vif\_vals[i] <- calc\_vif(models[[i]]) #VIF  
 rmse[i] <- calc\_rmse(args[[i]],10) #Posterior Predictive Performance  
}  
  
  
# Cross-Validation for RMSE (Posterior Predictive Performance)  
  
# Create a summary table  
comparison\_table <- data.frame(  
 Model = c("Model 1\n(Weight)", "Model 2\n(+Weight^2)",  
 "Model 3\n(Weight\n+Sex)", "Model 4\n(Weight\n+Weight^2\n+Sex)"),  
 Adj\_R2 = adj\_r2,  
 BP\_p = bp\_p,  
 Shapiro\_p = shapiro\_p,  
 Max\_Cooks\_D = cooks\_max,  
 Max\_VIF = vif\_vals,  
 RMSE = rmse  
)  
  
comparison\_table\_transposed <- as.data.frame(t(comparison\_table))  
colnames(comparison\_table\_transposed) <- comparison\_table$Model # Use Model names as column headers  
comparison\_table\_transposed <- comparison\_table\_transposed[-1, ] # Remove the row with model names  
comparison\_table\_transposed <- cbind(Metric = rownames(comparison\_table\_transposed), comparison\_table\_transposed) # Add row labels  
rownames(comparison\_table\_transposed) <- NULL # Clear rownames for better formatting  
  
# Generate a flextable for the transposed metrics  
comparison\_table\_flex <- flextable(comparison\_table\_transposed) %>%  
 colformat\_double(digits = 3) %>%  
 bg(part = "header", bg = "#D3D3D3") %>% # Grey background for the top header row  
 bg(i = NULL, j = 1, bg = "#D3D3D3", part = "body") %>% # Grey background for the left column  
 theme\_box() %>%  
 align(j = 1, align = "left", part = "all") %>% # Left-align the left column  
 autofit()  
  
# Display the table  
comparison\_table\_flex

| **Metric** | **Model 1 (Weight)** | **Model 2 (+Weight^2)** | **Model 3 (Weight +Sex)** | **Model 4 (Weight +Weight^2 +Sex)** |
| --- | --- | --- | --- | --- |
| Adj\_R2 | 0.7892028 | 0.7950624 | 0.8047912 | 0.8266101 |
| BP\_p | 0.6365383 | 0.8362766 | 0.1286418 | 0.7063236 |
| Shapiro\_p | 0.8381626 | 0.8381626 | 0.8381626 | 0.8381626 |
| Max\_Cooks\_D | 0.3045838 | 0.2969302 | 0.2741246 | 0.2189285 |
| Max\_VIF | --NA-- | 52.120366 | 1.268628 | 61.434462 |
| RMSE | 4.472237 | 4.673119 | 4.533823 | 4.378760 |

Model 4 is the best overall choice as it balances high Adjusted R-square, prediction accuracy (RMSE), and residual diagnostics, but it does have a high VIF which can suggest multicollinearity, and may require further investigation.

# Step4 : Check Collinearity

# Load necessary library for collinearity check  
  
# Fit the models  
m1 <- lm(height ~ weight, data = my\_kungsan)  
m2 <- lm(height ~ weight + I(weight^2), data = my\_kungsan)  
m3 <- lm(height ~ weight + I(weight^2) + sex, data = my\_kungsan)  
m5 <- lm(height ~ weight + weight2 + sex, data = my\_kungsan)  
  
# Check variance inflation for m2 and m3  
vif\_m2 <- check\_collinearity(m2)  
vif\_m3 <- check\_collinearity(m3)  
vif\_m5 <- check\_collinearity(m5)

print(vif\_m2)

## # Check for Multicollinearity  
##   
## High Correlation  
##   
## Term VIF VIF 95% CI Increased SE Tolerance Tolerance 95% CI  
## weight 52.12 [36.15, 75.34] 7.22 0.02 [0.01, 0.03]  
## I(weight^2) 52.12 [36.15, 75.34] 7.22 0.02 [0.01, 0.03]

print(vif\_m3)

## # Check for Multicollinearity  
##   
## Low Correlation  
##   
## Term VIF VIF 95% CI Increased SE Tolerance Tolerance 95% CI  
## sex 1.50 [ 1.24, 2.04] 1.22 0.67 [0.49, 0.81]  
##   
## High Correlation  
##   
## Term VIF VIF 95% CI Increased SE Tolerance Tolerance 95% CI  
## weight 58.17 [40.48, 83.78] 7.63 0.02 [0.01, 0.02]  
## I(weight^2) 61.43 [42.74, 88.49] 7.84 0.02 [0.01, 0.02]

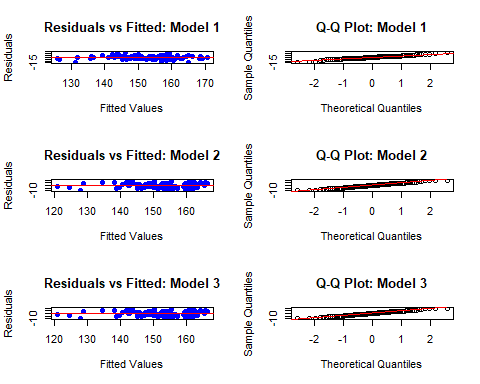
print(vif\_m5)

## # Check for Multicollinearity  
##   
## Low Correlation  
##   
## Term VIF VIF 95% CI Increased SE Tolerance Tolerance 95% CI  
## sex 1.50 [ 1.24, 2.04] 1.22 0.67 [0.49, 0.81]  
##   
## High Correlation  
##   
## Term VIF VIF 95% CI Increased SE Tolerance Tolerance 95% CI  
## weight 58.17 [40.48, 83.78] 7.63 0.02 [0.01, 0.02]  
## weight2 61.43 [42.74, 88.49] 7.84 0.02 [0.01, 0.02]

Multicollinearity exists between weight and I(weight^2), as indicated by the high VIF values. The variable sex does not contribute to collinearity. Adjustments to the model are required to address this issue.

# Step5: Examine Residual Plots and Normality of Residuals

# Residual Plots: Residuals vs Fitted Values and Q-Q Plots  
  
mmodels <- list(m1,m3,m5)  
par(mfrow = c(3, 2)) # Set up a 2x2 grid for plots  
  
for (i in seq\_along(mmodels)) {  
 model <- mmodels[[i]] # Extract the i-th model  
   
 # Residuals vs Fitted Plot  
 plot(model$fitted.values, model$residuals,  
 main = paste("Residuals vs Fitted: Model", i),  
 xlab = "Fitted Values", ylab = "Residuals",  
 pch = 19, col = "blue")  
 abline(h = 0, col = "red") # Add a horizontal line at 0  
   
 qqnorm(model$residuals, main = paste("Q-Q Plot: Model", i))  
 qqline(model$residuals, col = "red") # Add reference line  
}



par(mfrow = c(1, 1)) # Reset plot layout to single plot

Purpose:

* Residuals vs Fitted Plot: Checks for homoscedasticity (equal variance of residuals) and non-linear patterns.
* Q-Q Plot: Assesses whether residuals follow a normal distribution, critical for reliable statistical inference.

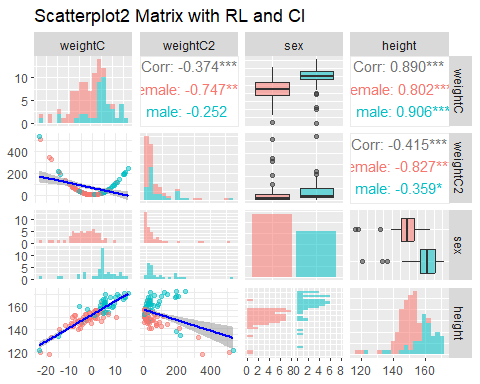
Insights:

* Residuals centred around zero, but some heteroscedasticity was observed in the residuals vs fitted plots, suggesting potential issues with variance stability.
* Q-Q plots indicated that residuals were approximately normally distributed, meeting this key assumption.

# Step 6: Centring weight and creating weightC2

# Centre the weight variable  
my\_kungsan <- my\_kungsan %>%  
 mutate(weightC = weight - mean(weight)) %>% # Centre weight  
 mutate(weightC2 = weightC^2) # centred weight squared  
  
  
# Scatterplot   
kung\_splot2 <- ggpairs(  
 my\_kungsan,  
 columns = c("weightC", "weightC2", "sex", "height"),  
 lower = list(continuous = add\_rlci), #lwoer triangle  
 diag = list(continuous = wrap("barDiag", bins = 20)),   
# upper = list(continuous = add\_contour),  
 upper = list(continuous = "cor"),   
 aes(colour = sex, alpha = 0.7)  
) +   
 labs(title = "Scatterplot2 Matrix with RL and CI")  
  
  
kung\_splot2

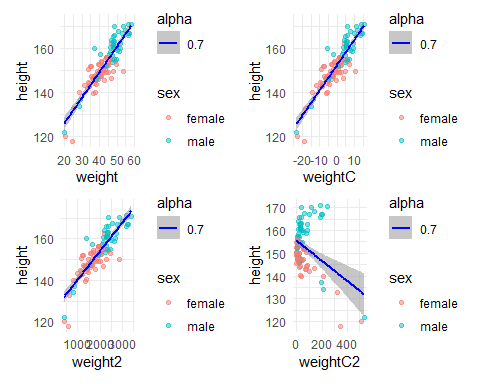
## `geom\_smooth()` using formula = 'y ~ x'  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `geom\_smooth()` using formula = 'y ~ x'  
## `geom\_smooth()` using formula = 'y ~ x'  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# Access specific panels in a `ggpairs` plot using indexing  
plot\_weight\_height\_3 <- kung\_splot1[4, 1] # Step 3: weight vs height  
plot\_weight2\_height\_3 <- kung\_splot1[4, 2] # Step 3: weight2 vs height  
# plot\_sex\_height\_3 <- kung\_splot1[4, 3] # Step 3: sex vs height  
  
plot\_weight\_height\_6 <- kung\_splot2[4, 1] # Step 6: weightC vs height  
plot\_weight2\_height\_6 <- kung\_splot2[4, 2] # Step 6: weightC2 vs height  
# plot\_sex\_height\_6 <- kung\_splot2[4, 3] # Step 6: sex vs height  
  
  
comparison\_plot <- (kung\_splot1[4, 1] + kung\_splot2[4, 1]) / # Row 1  
 (kung\_splot1[4, 2] + kung\_splot2[4, 2]) # / # Row 2  
 # (kung\_splot[4, 3] + kung\_splot2[4, 3]) # Row 3

comparison\_plot

## `geom\_smooth()` using formula = 'y ~ x'  
## `geom\_smooth()` using formula = 'y ~ x'  
## `geom\_smooth()` using formula = 'y ~ x'  
## `geom\_smooth()` using formula = 'y ~ x'



Centring variables (weightC and weightC2) improve symmetry, reduce extreme ranges, enhance visuals, and address multicollinearity. The opposite slopes for height vs weight2 and height vs weightC2 are because centring changes the reference point of the quadratic term to the mean. This reduces the dependency between the linear (weightC) and quadratic (weightC2) terms, providing better scaling and stability in the regression models. Centring appears to be necessary when exploring the introduction of a quadratic term.

# Step7: Fit m4 with weightC, weightC2, and sex as predictors

# m4 model  
m4 <- lm(height ~ weightC + weightC2 + sex, data = my\_kungsan)  
vif\_m4 <- check\_collinearity(m4) # VIF for the centred model  
  
# Merge collinearity results for m5 and m4  
vif\_comparison <- merge(vif\_m5, vif\_m4, by = "Term", all = TRUE, suffixes = c("\_m5", "\_m4"))  
  
# Combine data for common metrics across groups  
vif\_combined <- vif\_comparison %>%  
 mutate(  
 VIF = coalesce(VIF\_m5, VIF\_m4), # Combine VIF columns  
 VIF\_CI\_low = coalesce(VIF\_CI\_low\_m5, VIF\_CI\_low\_m4),  
 VIF\_CI\_high = coalesce(VIF\_CI\_high\_m5, VIF\_CI\_high\_m4),  
 SE\_factor = coalesce(SE\_factor\_m5, SE\_factor\_m4),  
 Tolerance = coalesce(Tolerance\_m5, Tolerance\_m4),  
 Tolerance\_CI\_low = coalesce(Tolerance\_CI\_low\_m5, Tolerance\_CI\_low\_m4),  
 Tolerance\_CI\_high = coalesce(Tolerance\_CI\_high\_m5, Tolerance\_CI\_high\_m4)  
 ) %>%  
 select(  
 Metric = Term,  
 VIF,  
 VIF\_CI\_low,  
 VIF\_CI\_high,  
 SE\_factor,  
 Tolerance,  
 Tolerance\_CI\_low,  
 Tolerance\_CI\_high  
 )  
  
# Ensure proper rounding  
vif\_combined <- vif\_combined %>%  
 mutate(across(where(is.numeric), ~ round(.x, 3)))  
  
# Transpose the table for final display  
vif\_combined\_transposed <- as.data.frame(t(vif\_combined))  
vif\_combined\_transposed <- tibble::rownames\_to\_column(vif\_combined\_transposed, var = "Metric") # Move row names to a column  
colnames(vif\_combined\_transposed)[-1] <- vif\_combined$Metric # Use terms as column names  
  
# Remove the duplicate "Metric" row if present  
vif\_combined\_transposed <- vif\_combined\_transposed[-1, ]  
  
# Reorder columns  
column\_order <- c("Metric", "sex", "weight", "weight2", "weightC", "weightC2")  
vif\_combined\_transposed <- vif\_combined\_transposed[, column\_order]  
  
  
vif\_combined\_transposed <- vif\_combined\_transposed %>%  
 mutate(  
 Guideline = c(  
 "Lower is better, indicates\nmulticollinearity\n(Variance Inflation Factor)",  
 "Lower is better, lower boundary\nof confidence interval for VIF",  
 "Lower is better, upper boundary\nof confidence interval for VIF",  
 "Lower is better, indicates the\nstandard error inflation",  
 "Higher is better, indicates\ntolerance of multicollinearity",  
 "Higher is better, lower boundary\nof confidence interval\nfor Tolerance",  
 "Higher is better, upper boundary\nof confidence interval\nfor Tolerance"  
 )  
 )  
  
# Create a flextable for the final table  
vif\_combined\_flex <- flextable(vif\_combined\_transposed) %>%  
 colformat\_double(digits = 3) %>% # Format numeric values to 3 decimal places  
 bg(part = "header", bg = "#D3D3D3") %>% # Grey background for the header row  
 bg(i = NULL, j = 1, bg = "#D3D3D3", part = "body") %>% # Grey background for the first column  
 theme\_box() %>%  
 align(j = 1, align = "left", part = "all") %>% # Left-align the first column  
 align(j = 2:(ncol(vif\_combined\_transposed) - 1), align = "center", part = "all") %>% # Center-align other columns  
 align(j = ncol(vif\_combined\_transposed), align = "left", part = "all") %>% # Left-align the last column  
 fontsize(size = 9, part = "all") %>%   
 width(j = 1, width = 1.5) %>% # Adjust width for the "Metric" column  
 width(j = ncol(vif\_combined\_transposed), width = 1) %>% # Adjust width for the "Guideline" column  
 autofit() # Adjust column widths automatically for remaining columns

# Display the transposed flextable  
vif\_combined\_flex

| **Metric** | **sex** | **weight** | **weight2** | **weightC** | **weightC2** | **Guideline** |
| --- | --- | --- | --- | --- | --- | --- |
| VIF | 1.495 | 58.171 | 61.434 | 1.701 | 1.371 | Lower is better, indicates multicollinearity (Variance Inflation Factor) |
| VIF\_CI\_low | 1.237 | 40.484 | 42.745 | 1.374 | 1.157 | Lower is better, lower boundary of confidence interval for VIF |
| VIF\_CI\_high | 2.035 | 83.781 | 88.492 | 2.314 | 1.878 | Lower is better, upper boundary of confidence interval for VIF |
| SE\_factor | 1.223 | 7.627 | 7.838 | 1.304 | 1.171 | Lower is better, indicates the standard error inflation |
| Tolerance | 0.669 | 0.017 | 0.016 | 0.588 | 0.730 | Higher is better, indicates tolerance of multicollinearity |
| Tolerance\_CI\_low | 0.491 | 0.012 | 0.011 | 0.432 | 0.533 | Higher is better, lower boundary of confidence interval for Tolerance |
| Tolerance\_CI\_high | 0.808 | 0.025 | 0.023 | 0.728 | 0.865 | Higher is better, upper boundary of confidence interval for Tolerance |

# Step 8: Predict Height with Intervals

# Step 7: Update `sex` to be a factor with levels 0 (Male) and 1 (Female)  
my\_kungsan <- my\_kungsan %>%  
 mutate(sex = ifelse(sex == "male", 0, 1)) %>% # Recode as 0 = Male, 1 = Female  
 mutate(sex = factor(sex, levels = c(0, 1))) # Ensure factor levels are correctly set  
  
# Step 7: Re-fit m4 with updated `sex`  
m4 <- lm(height ~ weightC + weightC2 + sex, data = my\_kungsan)  
  
# Step 8: Predict Height with Intervals  
  
# Calculate centred weight (50 kg - 36 kg = 14)  
weightC <- 50 - 36 # Consistent variable name for centred weight  
  
# Define input data for predictions  
# Use the updated factor levels for `sex`  
new\_data <- data.frame(  
 weightC = c(weightC, weightC), # Centred weight  
 weightC2 = c(weightC^2, weightC^2), # Centred weight squared  
 sex = factor(c(0, 1), levels = c(0, 1)) # 0 = Male, 1 = Female  
)  
  
# Generate predictions with prediction intervals  
predictions <- predict(  
 m4, # Model m4  
 newdata = new\_data, # Input data  
 interval = "prediction", # Get prediction intervals  
 level = 0.95 # 95% confidence level  
)  
  
# Generate predictions with confidence intervals  
confidence\_intervals <- predict(  
 m4, # Model m4  
 newdata = new\_data, # Input data  
 interval = "confidence", # Get confidence intervals  
 level = 0.95 # 95% confidence level  
)  
  
results <- data.frame(  
 Individual = c("50 kg Male", "50 kg Female"), # Character column  
 fit = confidence\_intervals[, "fit"], # Numeric column  
 lwr = confidence\_intervals[, "lwr"], # Numeric column  
 upr = confidence\_intervals[, "upr"], # Numeric column  
 PI\_Lower = predictions[, "lwr"], # Numeric column  
 PI\_Upper = predictions[, "upr"] # Numeric column  
)  
  
# Verify data types  
str(results)

## 'data.frame': 2 obs. of 6 variables:  
## $ Individual: chr "50 kg Male" "50 kg Female"  
## $ fit : num 166 161  
## $ lwr : num 163 158  
## $ upr : num 168 165  
## $ PI\_Lower : num 157 152  
## $ PI\_Upper : num 175 170

# Create a flextable using the formatted data  
results\_flextable <- flextable(results) %>%  
 colformat\_double(digits = 3) %>%  
 bg(part = "header", bg = "#D3D3D3") %>% # Add grey background for the header  
 theme\_box() %>%  
 autofit() # Adjust column widths  
  
# Display the flextable  
results\_flextable

| **Individual** | **fit** | **lwr** | **upr** | **PI\_Lower** | **PI\_Upper** |
| --- | --- | --- | --- | --- | --- |
| 50 kg Male | 165.836 | 163.253 | 168.419 | 157.019 | 174.653 |
| 50 kg Female | 161.292 | 157.711 | 164.872 | 152.133 | 170.451 |