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
suppressPackageStartupMessages({
  library(conflicted)
  library(tidyverse)
  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(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 x 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

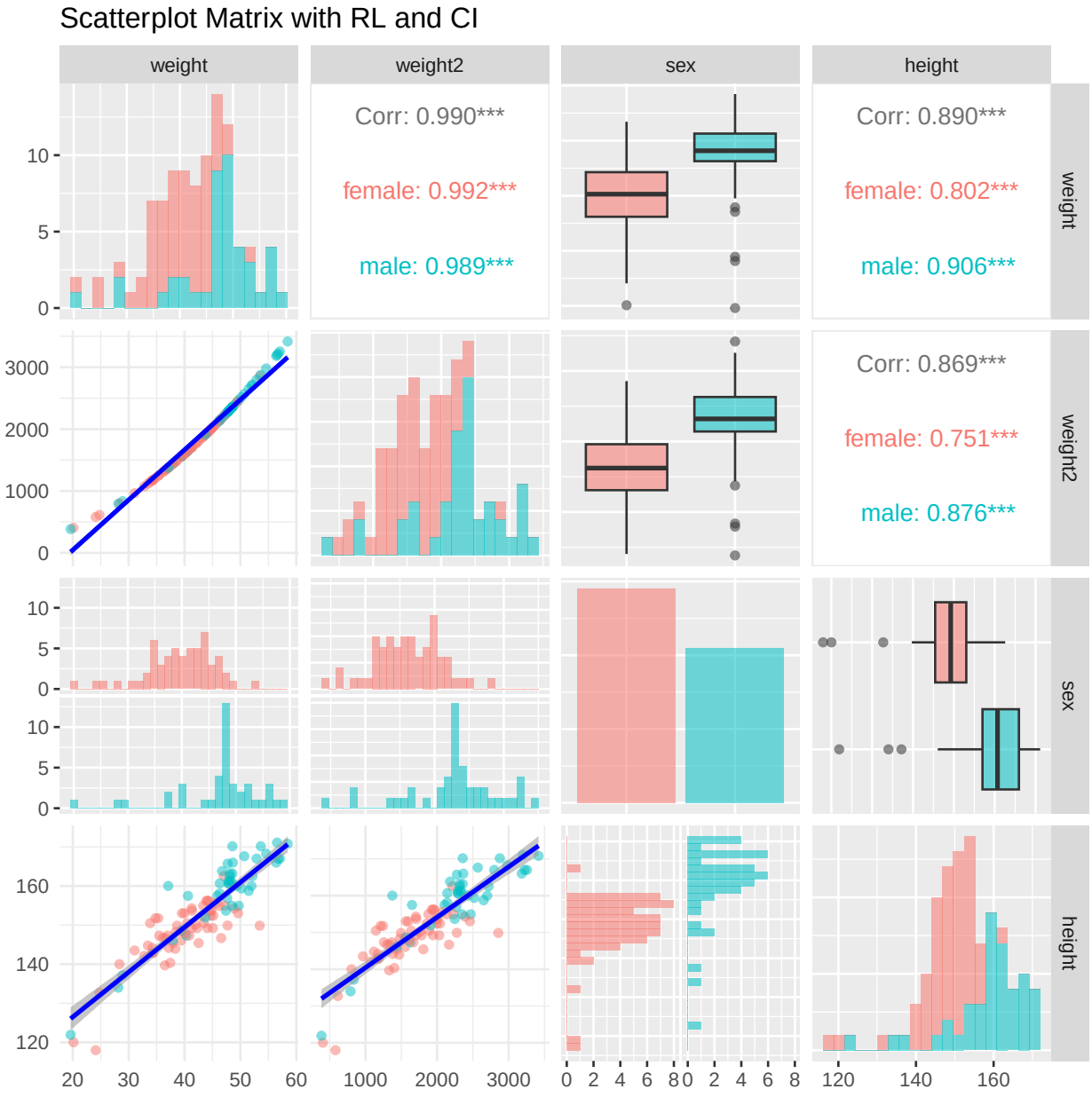
```
# function = add regression line and confidence interval
add_rlci <- function(data, mapping, ...) {
  ggplot(data = data, mapping = mapping) +
    geom_point(alpha = 0.5) +  # Scatterplot points
    geom_smooth(method = "lm", se = TRUE, colour = "blue", ...) +  # rl ci
    theme_minimal()
}

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.

```r
# 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)
```

```r
# 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_t
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

```r
# 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)
```

```r
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]
```

```r
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]
```

```r
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

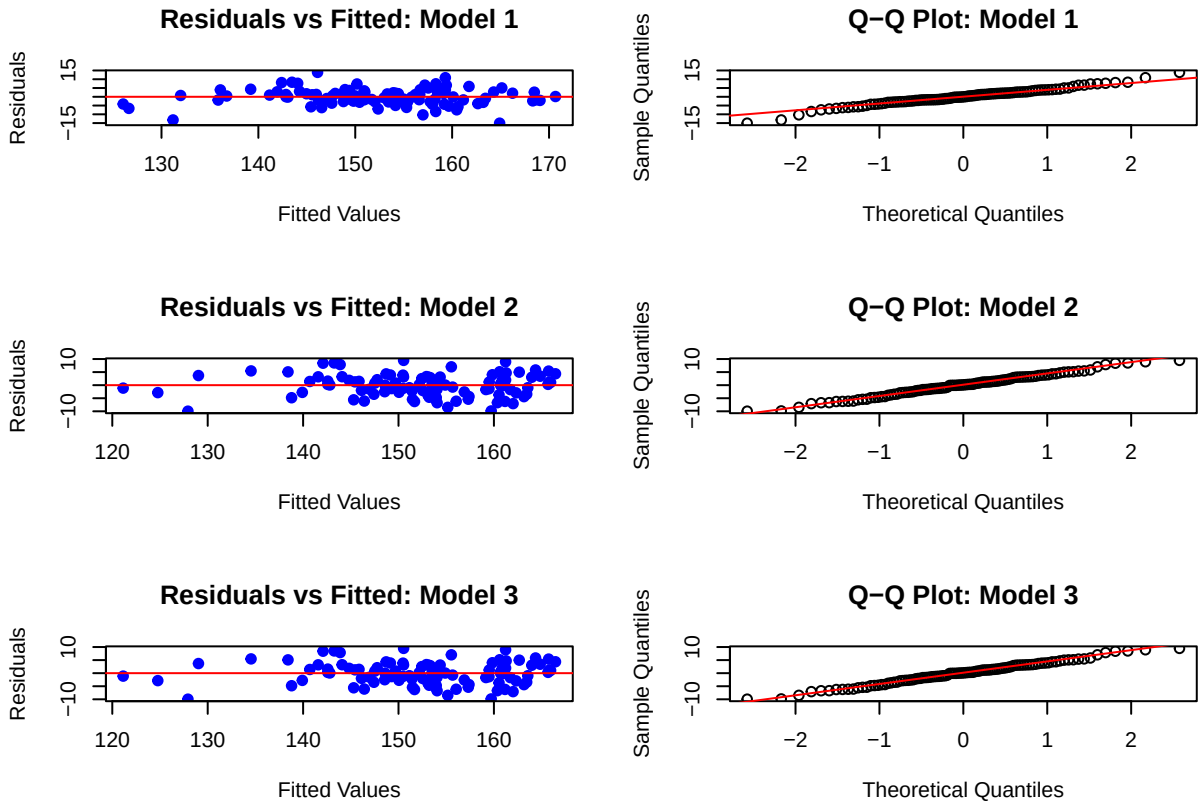
```r
# Residual Plots: Residuals vs Fitted Values and Q-Q Plots

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
}
```




```r
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.

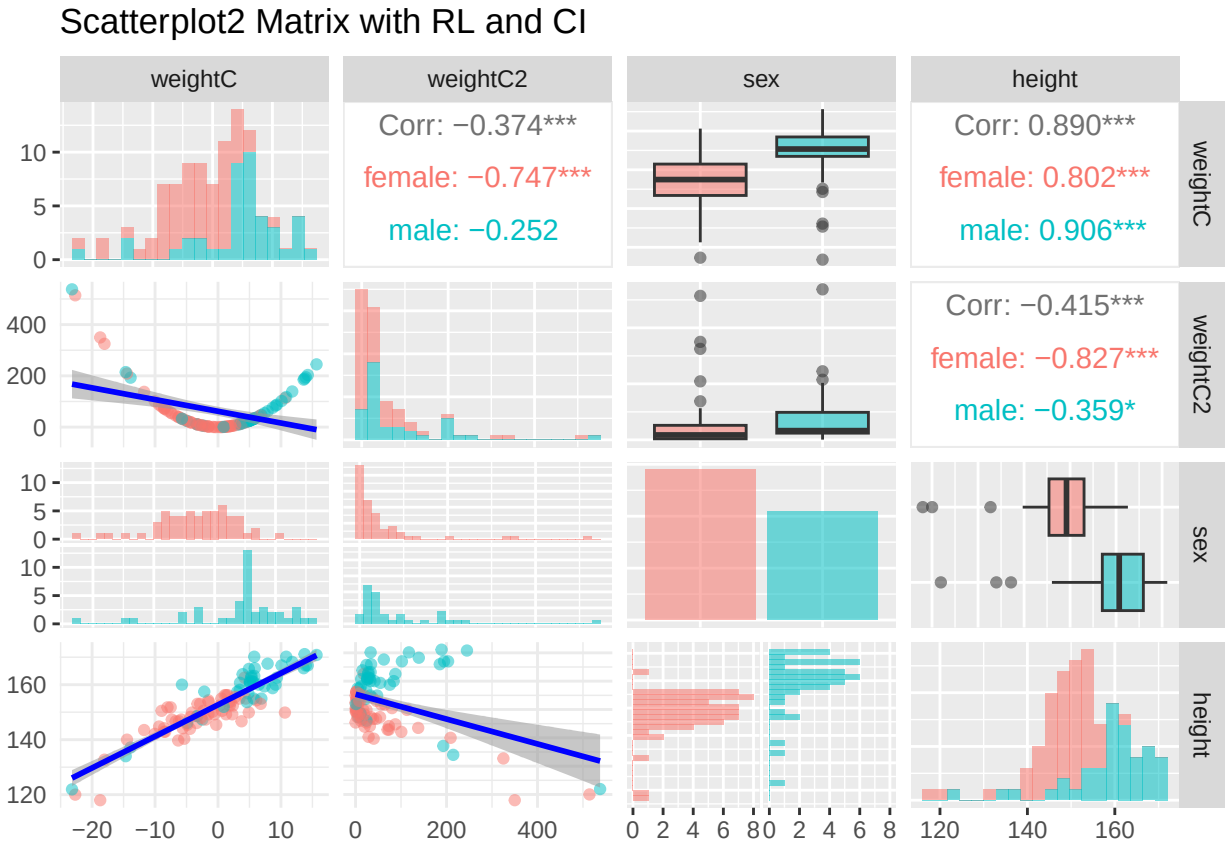
Step 6: Centring weight and creating weightC2

```
# 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`.
```


Scatterplot2 Matrix with RL and CI
weightC
weightC2
sex
height
Corr: −0.374***
female: −0.747***
male: −0.252
Corr: 0.890***
female: 0.802***
male: 0.906***
Corr: −0.415***
female: −0.827***
male: −0.359*
weightC
weightC2
sex
height

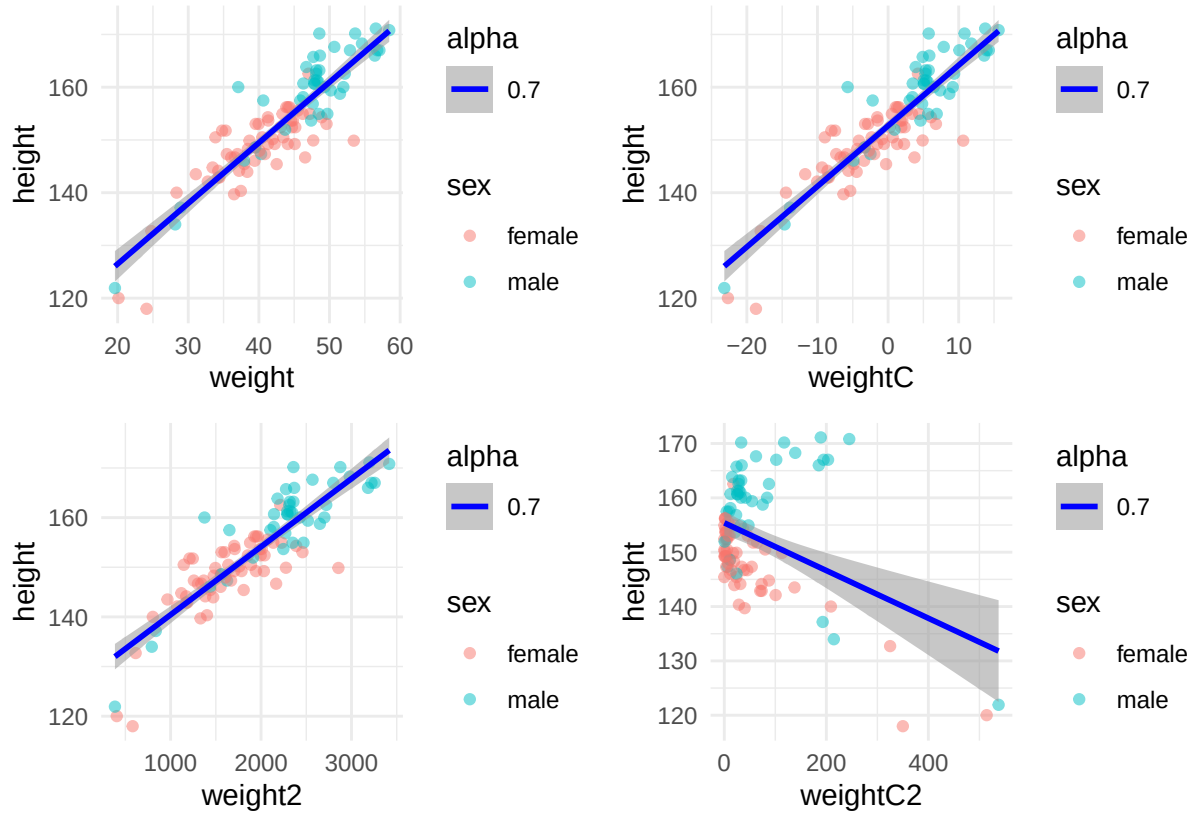
```r
# Access specific panels in a `ggpairs` plot using indexing
plot_weight_height_3 <- kung_splot1[4, 1]  # Step 3: weight vs height
plot_weight2_height_3 <- kung_splot1[4, 2] # Step 3: weight2 vs height
# plot_sex_height_3 <- kung_splot1[4, 3]     # Step 3: sex vs height

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
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
  align(j = ncol(vif_combined_transposed), align = "left", part = "all") %>%  # Left-align the last col
  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 final 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

```
# 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