Final Project STP 530

Adam Kurth

2023-11-08

Context:

Crystallography, an interdisciplinary field primarily centered on the study of solid-state chemistry and physics, delves into various aspects of matter. A significant branch of this field, protein crystallography, and its use of X-rays, focus on unraveling the three-dimensional structure and interactions of proteins. This interaction broadly speaking leads to strides in research pertaining to drug and vaccine development. To illustrate its breadth and importance, consider the following examples:

- Investigating the interaction mechanisms between the Sars-Cov2 virus and human cells, a topic that remains somewhat enigmatic.
- Exploring how various proteins within the human body respond to pharmaceuticals, allowing scientists to see how drugs fit into protein structures, a crucial aspect of increasing drug efficiency.
- Deepening our understanding of photosynthesis by revealing the structure of photosynthetic proteins and complexes, and how these function in the energy transfer.

Potential Predictors:

- `Mean.Intensity`: refers to the *average* intensity of the diffracted X-rays. This data is in units of the counted number of photons from the diffracted light. This intensity of the diffracted X-rays is proportional to the square of the structure factor amplitude. This predictor can give insight into the overall electron density and crucial for crystal structure.
- `Mean.Phase`: Despite the nuances about the "Phase Problem" in crystallography (below if curious), each diffracted wave has a phase angle. The phase problem, central to crystallography, as phases cannot be measured. However, they're still crucial for determining the electron density map. The mean phase represents the *average* measure of these phase angles across the data set.
- `a, b, c, alpha, beta, gamma`: These are unit cell parameters which are simply the lengths (a,b,c) and the angles of the unit cell measured in degrees (alpha, beta, gamma). This is fundamental for determining the system and size of the crystal lattice.
- `Unitcell.Volume`: The volume is calculated directly from the unit cell parameters. This is an indicator of the density of the crystal structure.
- `Crystal.Volume`: Volume of the actual crystal used in the diffraction experiment. This is important for understanding how the size of the crystal affects the diffraction pattern.
- `Vol.Unit.Ratio`: The ratio of the crystal volume to the unit cell volume, also called packing ratio. This is used to understand the packing efficiency of the crystal, in other words how tightly packed each unit cell is.

- `Space Group`: The crystal's group symmetry, this is based on symmetric properties which are crucial for understanding the crystal structure. Note why these are separated is that different group symmetries imply different atomic arrangements, therefore different properties of the crystal.
- `Calc.Structure.Weight`: Computed value to represent the total mass of atoms in a molecule. Particularly in the context of protein crystals, this weight is an important characteristic because it reflects sum of the atomic weights of all the atoms present in the molecule, or repeated unit within the crystal lattice.

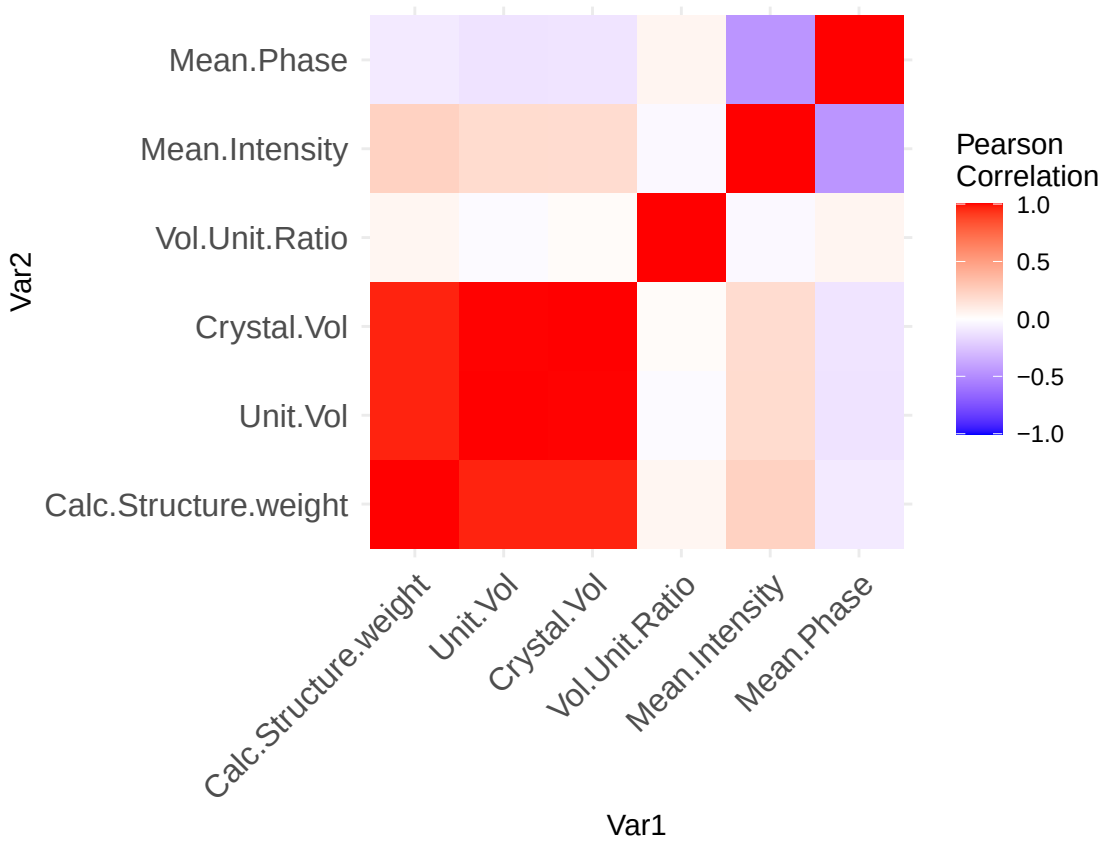
"The phase problem is the problem of information concerning the phase that can occur when making physical measurement [and] has to be solved for the determination of a structure from diffraction data." ("Phase Problem" n.d.).

Research Question:

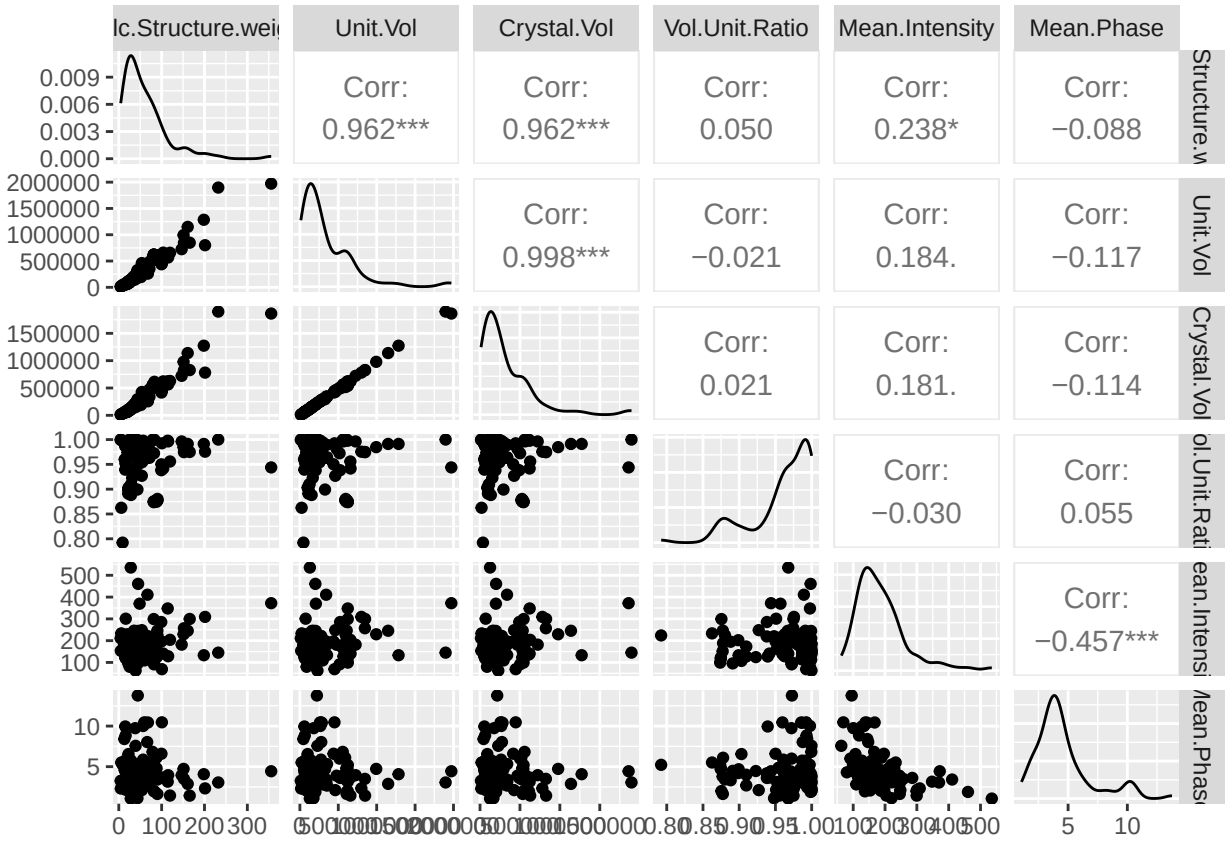
Let us see whether there is a linear dependence of X-ray diffraction intensities on the number of unit cells exposed to the X-rays. This further refines to the more specific, within the same space group, does the packing ratio have any affect on the diffracted intensity?

Data Exploration:

```
# To focus on only on numerical values interested in testing.
columns_to_remove = c("Spacegroup.ID", "PDB.ID", "a", "b", "c", "Alpha", "Beta", "Gamma", "Max.Intensity
                      "Min.Intensity", "MaxMin.Intensity.Dif", "Max.Phase",
                      "Min.Phase", "MaxMin.Phase.Difference", "PDB.ID")
numeric_df1_P1211 = df1_P1211[, !(names(df1_P1211) %in% columns_to_remove)]
# Visualize correlation matrix
cor_matrix_melt <- melt(cor_matrix <- cor(numeric_df1_P1211))
ggplot(data = cor_matrix_melt, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 12, hjust = 1),
        axis.text.y = element_text(size = 12)) +
  coord_fixed()
```




```
# Using gg
ggpairs(numeric_df1_P1211, cardinality_threshold = 100)
```



From the correlation matrices, its easy to see that as volume predictors increase, so does the structure weight. This makes sense, and is not interesting. The question rather, is whether the mean intensity predictor correlated with anything, specifically the packing ratio. As you can see that the largest Pearson's correlation in the matrix is that the predictor `Mean.Intensity` has is with `Structure.Weight` with 0.238. What we are interested in has a Pearson's correlation value of -0.03.

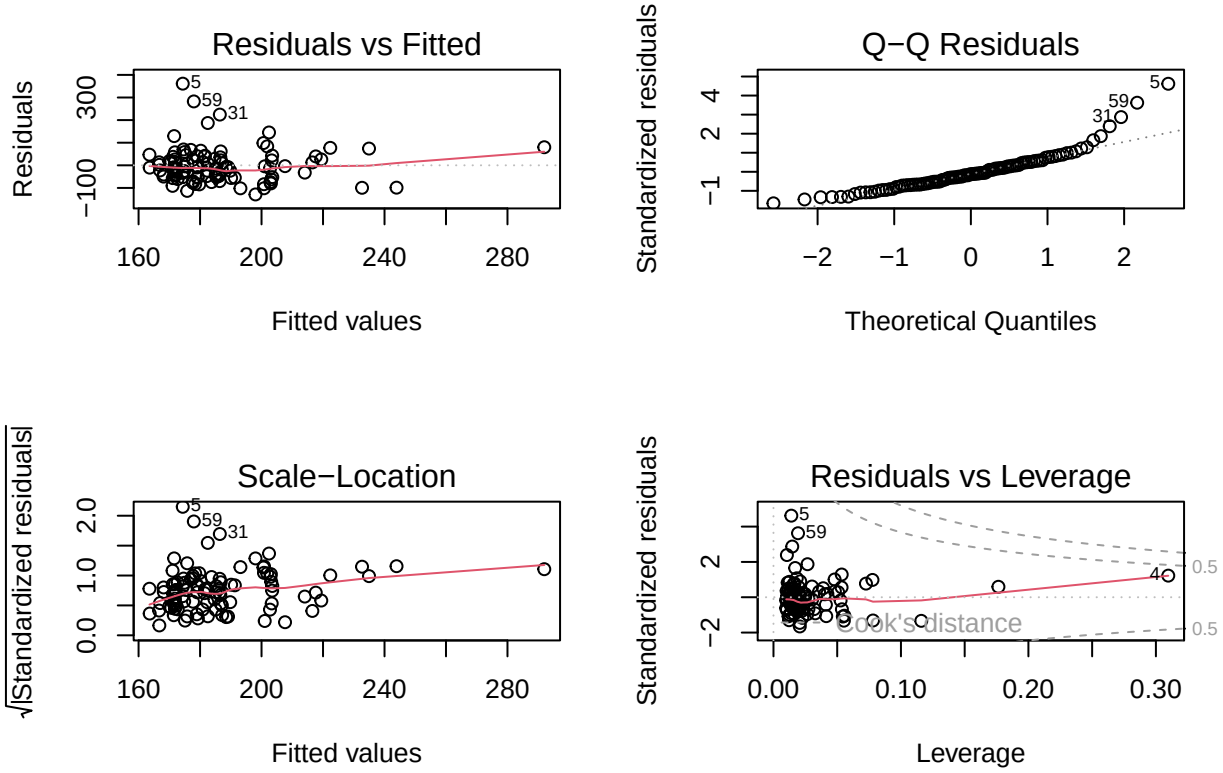
Also notice that the mean intensity and the mean phase have a seeming negative Pearson's correlation. The observed diffraction pattern is a result of the interference of X-rays scattered by electrons in the crystal. Of the diffracted waves, the interference pattern is governed by the intensity and the phase. Hence, there is interplay between when one varies significantly, then this can lead to a destructive interference pattern.

Another plausible explanation is that certain crystal structures may lead to destructive interference pattern due to the phase angle distributions, resulting in lower intensity.

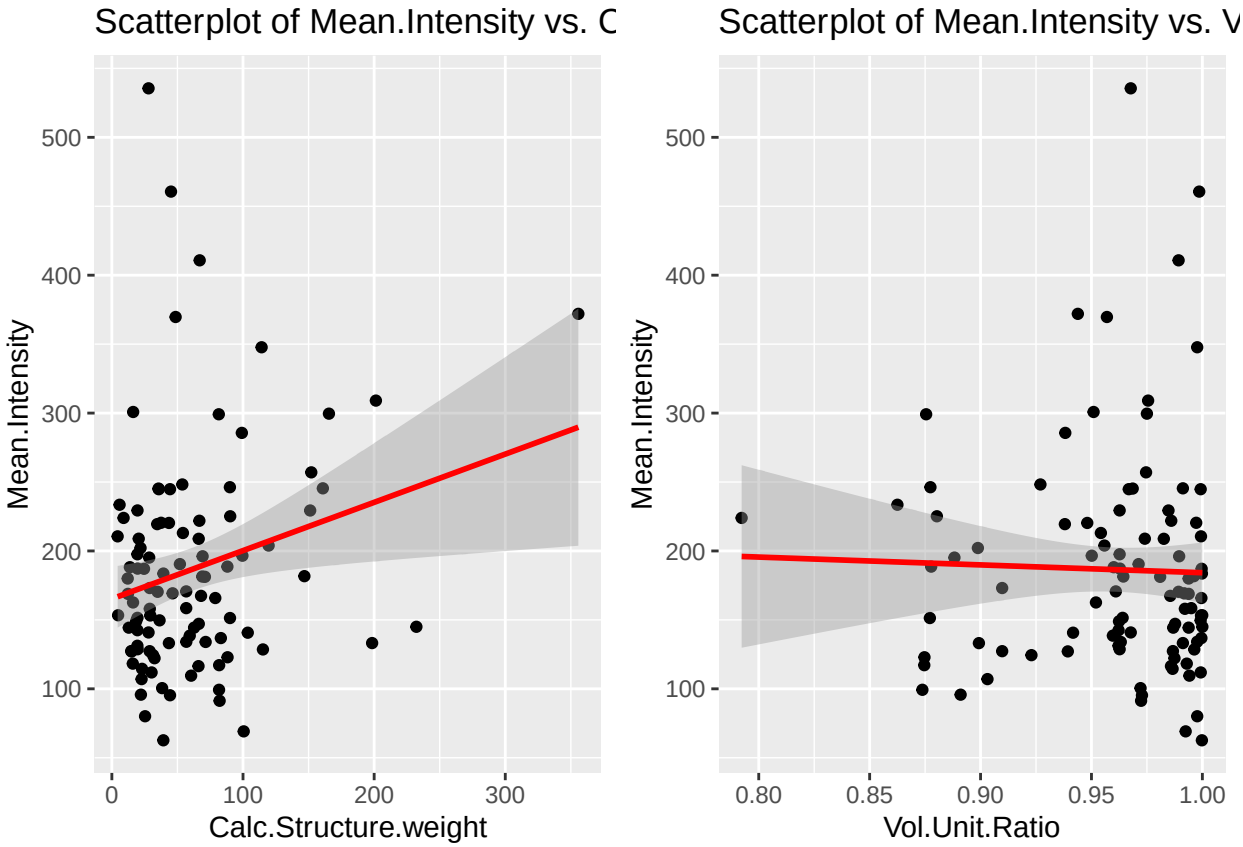
Model:

$$E\{\text{Mean.Intensity}\}_{P1211} = \beta_0 + \beta_1(\text{Calc.Structure.weight}) + \beta_2(\text{Vol.Unit.Ratio})$$

```
########## MODEL for P1211
# ------------------------------------------------------------------------
m.P1211 = lm(Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio, data = df1_P1211)
par(mfrow=c(2,2))
plot(m.P1211)
```




```
par(mfrow=c(1,1))
########## Diagnostics
# ------------------------------------------------------------------------
# Scatterplot of Calc.Structure.weight vs. Mean.Intensity with regression line
a = ggplot(data = df1_P1211, aes(x = Calc.Structure.weight, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Calc.Structure.weight", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Calc.Structure.weight")

# Scatterplot of Vol.Unit.Ratio vs. Mean.Intensity with regression line
b = ggplot(data = df1_P1211, aes(x = Vol.Unit.Ratio, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Vol.Unit.Ratio", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Vol.Unit.Ratio")
ggarrange(a,b)
```




```
# This does not show linear relationship off the bat.
# individually
par(mfrow=c(2,2))
assumption.1.check.linearity(m.P1211)
```

```
##
## Assumption 1: Checking for linearity...
```

```
assumption.2.check.outliers(m.P1211) # most definitely influence of outliers.
```

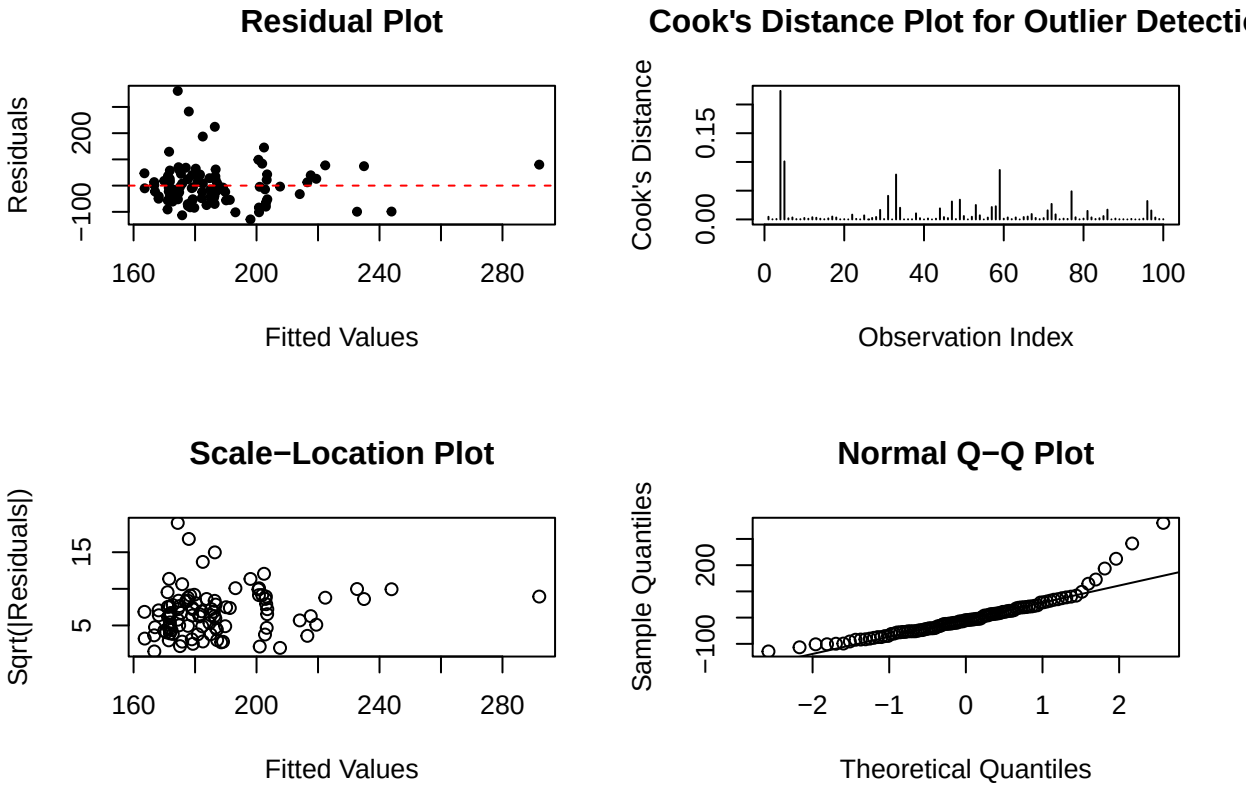
```
##
## Assumption 2: Checking for no excessive outliers...

## Observations with a Cook's distance greater than 0.04 may be considered influential:
##  4  5 31 33 59 77
##  4  5 31 33 59 77
```

```
assumption.3.check.homoskedasticity(m.P1211)
```

```
##
## Assumption 3: Checking for constant variance (homoskedasticity)...
```

```
assumption.4.check.normality(m.P1211)
```

```
##
## Assumption 4: Checking for normally distributed errors...
```

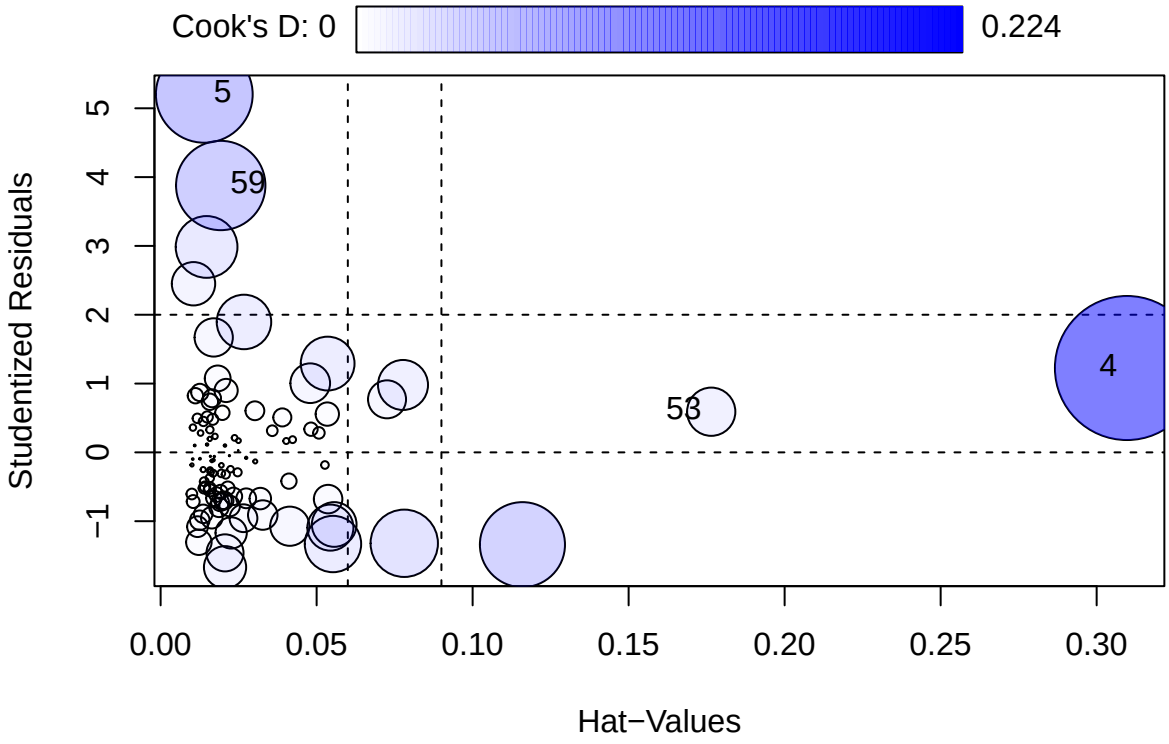



```
par(mfrow=c(1,1))
```

```
assumption.5.check.independence(m.P1211)
```

```
##
## Assumption 5: Checking for independence of errors...

##  lag Autocorrelation D-W Statistic p-value
##    1      0.05725871      1.878728   0.488
##  Alternative hypothesis: rho != 0
```

```
# Visualize outliers, leverage points, and influential points
car::influencePlot(m.P1211,id=list(labels=row.names(df1_P1211)))
```




```
##      StudRes        Hat      CookD
## 4  1.2259066 0.30971242 0.22360242
## 5  5.2025371 0.01399660 0.10094518
## 53 0.5907193 0.17647374 0.02509392
## 59 3.8779788 0.01926889 0.08603865
```

```
# Outliers detected rows: 4, 53, 59, 5
```

After checking the assumption, this model passes except for potentially having some outliers. As indicated on the scale-location plot, and qqplot.

Assumption 5: Cook's Distance:

Cook's Distance is a measure to estimate the influence of all the data points when using ordinary least squares in regression. This helps point out outliers with a strong influence on the data. Since the assumption 5 check, used the `cooks.distance()` function, and gave an autocorrelation of 0.05725871, this means that there's a low correlation between residuals. Thus, the residuals are independent.

D-W Test Statistic: Autocorrelation is the measure usually used in time series data which detects whether there is any correlation between the series of values.

This test statistic tests that the null hypothesis has no autocorrelation among the residuals. With a range of 0 to 4 for this particular test statistic shows a value very close to 2. When 4 means that there is positive autocorrelation, and 0 indicates that there's negative autocorrelation. Thus, there's little to no autocorrelation among the residuals in this model, since they're independent.

P-Value: Under the null hypothesis, the p-value states that there is no autocorrelation. With a p-value typically above 0.05, we fail to reject the null hypothesis. This does not give enough evidence to reject the null hypothesis. With a p-value of 0.496, this model indicates that there is strong statistical evidence of the contrary.

Influence Plot: As the influence plot shows, there is some high leverage points in this data.

```
assumption.6.check.multicollinearity(m.P1211)
```

```
##
## Assumption 6: Checking for multicollinearity...
## Calc.Structure.weight        Vol.Unit.Ratio
##              1.002484              1.002484
```

Assumption 6: Multicolinearity

With a VIF score of around 1, this suggests that there is very low multicolinearity in the model. We can proceed with this model that there are low correlation between the multiple independent variables in the model. Since these show VIF values of roughly around 1, these are perfectly colinear.

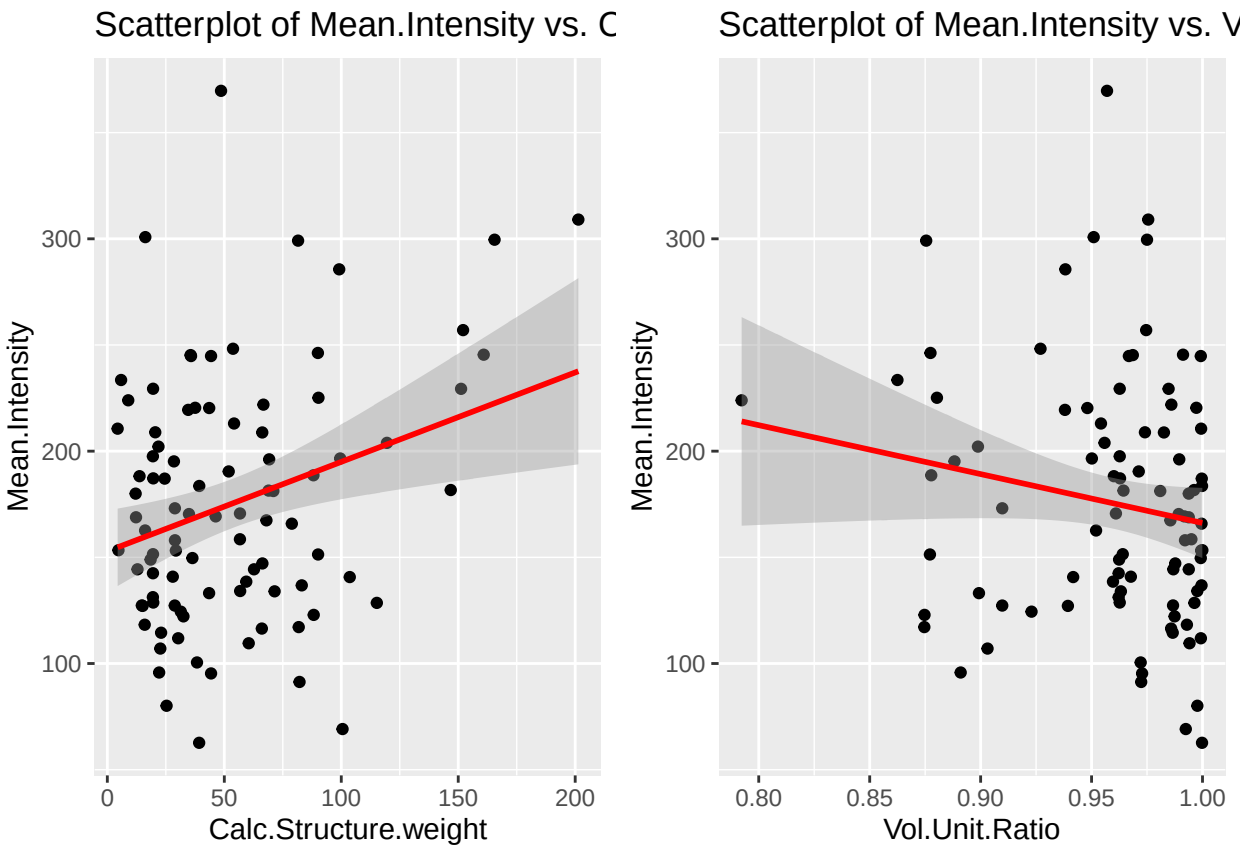
Proceeding to see whether removing high influence points, improves the model fit.

```
########## Treating outliers.
# ------------------------------------------------------------------------
cooks.d = cooks.distance(m.P1211)
influence = cooks.d[(cooks.d > (3*mean(cooks.d, na.rm = T)))]
influence # see that most of the rows stated above are in this list
```

```
##          4          5         31         33         49         59         77
## 0.22360242 0.10094518 0.04107084 0.07802573 0.03429384 0.08603865 0.04890137
##         96
## 0.03200839
```

```
row.influence = names(influence)
outliers = df1_P1211[row.influence,]
df1_P1211.reduced = df1_P1211 %>% anti_join(outliers)
########## Any improvement to the model?
# ------------------------------------------------------------------------
m.P1211.R = lm(Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio, data = df1_P1211.reduced)
# ------------------------------------------------------------------------
# Scatterplot of Calc.Structure.weight vs. Mean.Intensity with regression line
a = ggplot(data = df1_P1211.reduced, aes(x = Calc.Structure.weight, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Calc.Structure.weight", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Calc.Structure.weight")

# Scatterplot of Vol.Unit.Ratio vs. Mean.Intensity with regression line
b = ggplot(data = df1_P1211.reduced, aes(x = Vol.Unit.Ratio, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Vol.Unit.Ratio", y = "Mean.Intensity",
```

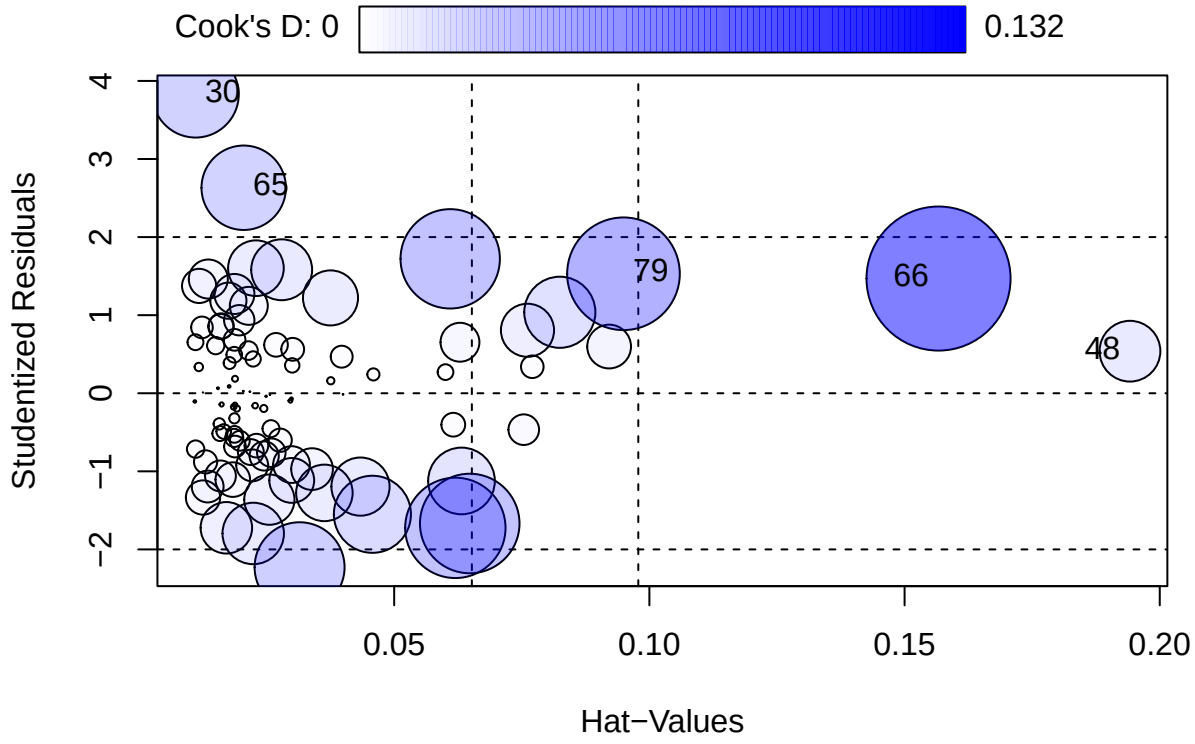
```
      title = "Scatterplot of Mean.Intensity vs. Vol.Unit.Ratio")
ggarrange(a,b)
```




```
# ------------------------------------------------------------------------
vif(m.P1211.R) # even lower but barely
```

```
## Calc.Structure.weight        Vol.Unit.Ratio
##              1.001138              1.001138
```

```
car::influencePlot(m.P1211.R,id=list(labels=row.names(df1_P1211.reduced))) # did not help much
```




```
##      StudRes         Hat      CookD
## 30 3.8280711 0.01107958 0.04744760
## 48 0.5385635 0.19415923 0.02348224
## 65 2.6318934 0.02051033 0.04533033
## 66 1.4690311 0.15665513 0.13190634
## 79 1.5303520 0.09492538 0.08066030
```

```
# ------------------------------------------------------------------------
summary(m.P1211.R)
```

```
##
## Call:
## lm(formula = Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio,
##     data = df1_P1211.reduced)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -118.467  -37.922   -1.539   33.620  195.710
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             386.8442   131.7820   2.935  0.00424 **
## Calc.Structure.weight     0.4295     0.1424   3.015  0.00334 **
## Vol.Unit.Ratio         -244.2656   137.2120  -1.780  0.07845 .
## ---
```

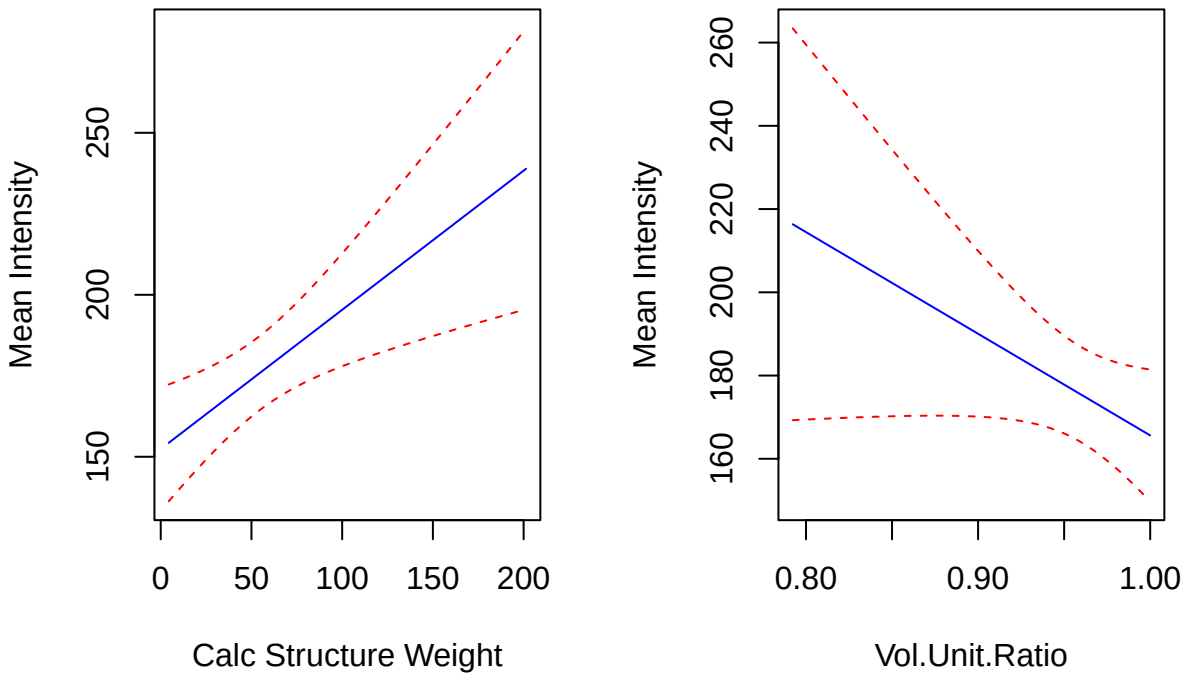
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55.21 on 89 degrees of freedom
## Multiple R-squared:  0.1181, Adjusted R-squared:  0.09824
## F-statistic: 5.957 on 2 and 89 DF,  p-value: 0.003733
```

```
r.a.2 = summary(m.P1211)$adj.r.squared
r.a.2.reduced = summary(m.P1211.R)$adj.r.squared
r.a.2.reduced - r.a.2
```

```
## [1] 0.05933491
```

```
# ------------------------------------------------------------------------
```

```
par(mfrow=c(1,2))
# ------------------------------------------------------------------------
# For Calc.Structure.weight
calc_structure_weight_mean <- mean(df1_P1211.reduced$Calc.Structure.weight, na.rm = TRUE)
calc_structure_weight_range <- seq(from = min(df1_P1211.reduced$Calc.Structure.weight),
                                   to = max(df1_P1211.reduced$Calc.Structure.weight), length.out = 100)
vol_unit_ratio_mean <- mean(df1_P1211.reduced$Vol.Unit.Ratio, na.rm = TRUE)

x_new <- data.frame(Calc.Structure.weight = calc_structure_weight_range,
                    Vol.Unit.Ratio = rep(vol_unit_ratio_mean, 100))

conf.int.1 <- predict(m.P1211.R, newdata = x_new, interval = "confidence")

plot(x_new$Calc.Structure.weight, conf.int.1[, "fit"], type = "l", col = "blue",
     ylim = range(conf.int.1[, c("lwr", "upr")]),
     ylab = "Mean Intensity", xlab = "Calc Structure Weight")
lines(x_new$Calc.Structure.weight, conf.int.1[, "lwr"], col = "red", lty = 2)
lines(x_new$Calc.Structure.weight, conf.int.1[, "upr"], col = "red", lty = 2)
# ------------------------------------------------------------------------
# For Vol.Unit.Ratio
vol_unit_ratio_range <- seq(from = min(df1_P1211.reduced$Vol.Unit.Ratio),
                            to = max(df1_P1211.reduced$Vol.Unit.Ratio), length.out = 100)

x_new <- data.frame(Vol.Unit.Ratio = vol_unit_ratio_range,
                    Calc.Structure.weight = rep(calc_structure_weight_mean, 100))

conf.int.2 <- predict(m.P1211.R, newdata = x_new, interval = "confidence")

plot(x_new$Vol.Unit.Ratio, conf.int.2[, "fit"], type = "l", col = "blue",
     ylim = range(conf.int.2[, c("lwr", "upr")]),
     ylab = "Mean Intensity", xlab = "Vol.Unit.Ratio")
lines(x_new$Vol.Unit.Ratio, conf.int.2[, "lwr"], col = "red", lty = 2)
lines(x_new$Vol.Unit.Ratio, conf.int.2[, "upr"], col = "red", lty = 2)
```




```
# ------------------------------------------------------------------------
par(mfrow=c(1,1))
confint(m.P1211.R)
```

```
##                                2.5 %      97.5 %
## (Intercept)             124.9962123 648.6922245
## Calc.Structure.weight     0.1465087   0.7125871
## Vol.Unit.Ratio         -516.9029261  28.3716603
```

By reducing the data set for this space group, there's an observed negative slope. When the packing ratio increases by 1 unit, there's a corresponding decrease in the average number of counted photons by -244.2656, holding all other predictors constant.

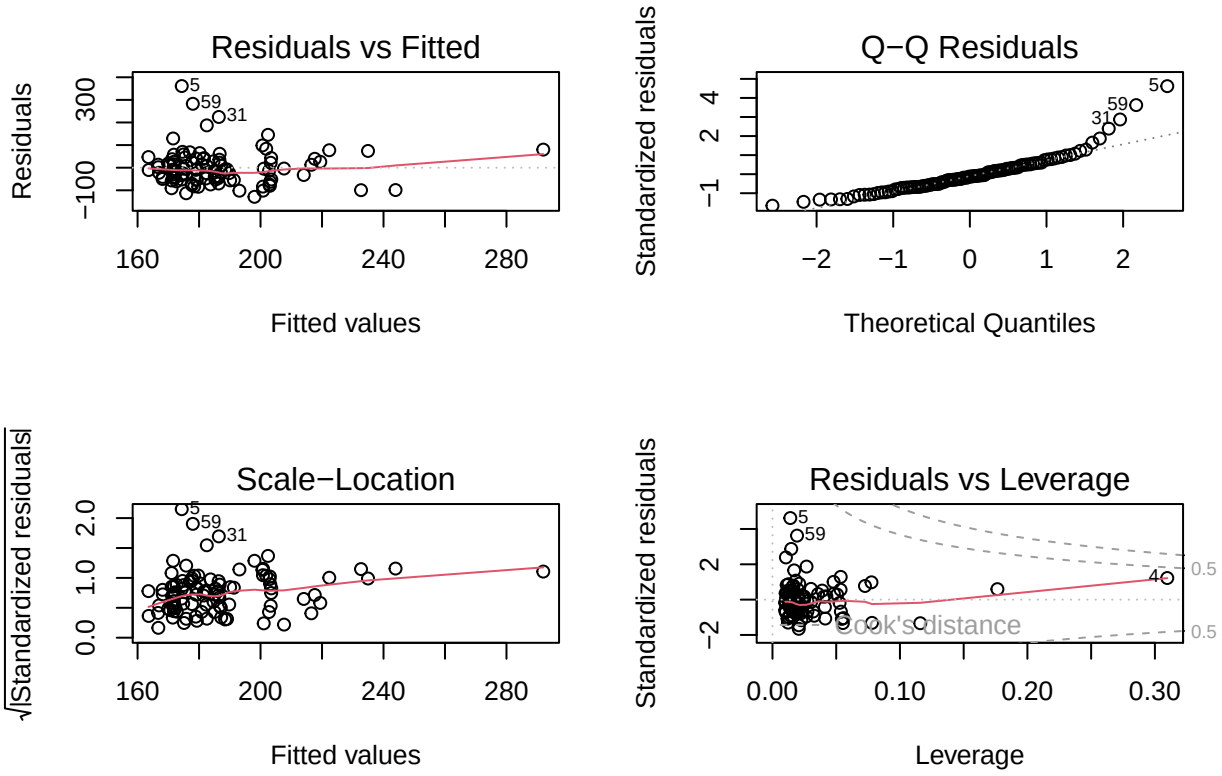
The confidence interval plots, and summary show that the linear relation between mean intensity and the packing ratio varies widely. Note that this interval's lower bound is negative, and the upper bound is positive. Thus, the slope could be 0.

Before testing whether the packing ratio can be dropped from the model, other space groups should be investigated.

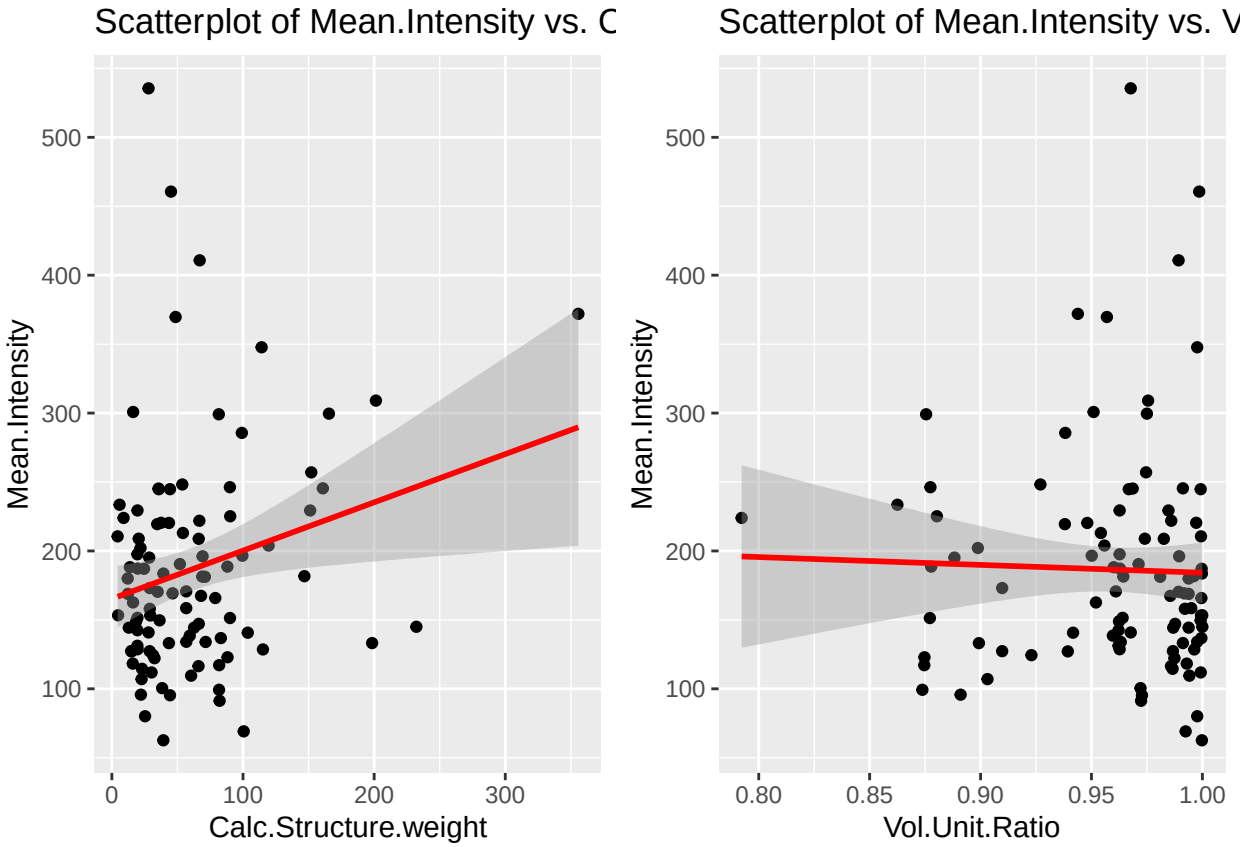
For P121 Space Group

```
########## MODEL for P121
# ------------------------------------------------------------------------
```

```
m.P121 = lm(Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio, data = df1_P121)
par(mfrow=c(2,2))
plot(m.P1211)
```




```
par(mfrow=c(1,1))
########## Diagnostics
# ------------------------------------------------------------------------
# Scatterplot of Calc.Structure.weight vs. Mean.Intensity with regression line
a = ggplot(data = df1_P121, aes(x = Calc.Structure.weight, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Calc.Structure.weight", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Calc.Structure.weight")

# Scatterplot of Vol.Unit.Ratio vs. Mean.Intensity with regression line
b = ggplot(data = df1_P121, aes(x = Vol.Unit.Ratio, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Vol.Unit.Ratio", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Vol.Unit.Ratio")
ggarrange(a,b)
```




```
# This does not show linear relationship off the bat.
# individually
par(mfrow=c(2,2))
assumption.1.check.linearity(m.P121)
```

```
##
## Assumption 1: Checking for linearity...
```

```
assumption.2.check.outliers(m.P121) # most definitely influence of outliers.
```

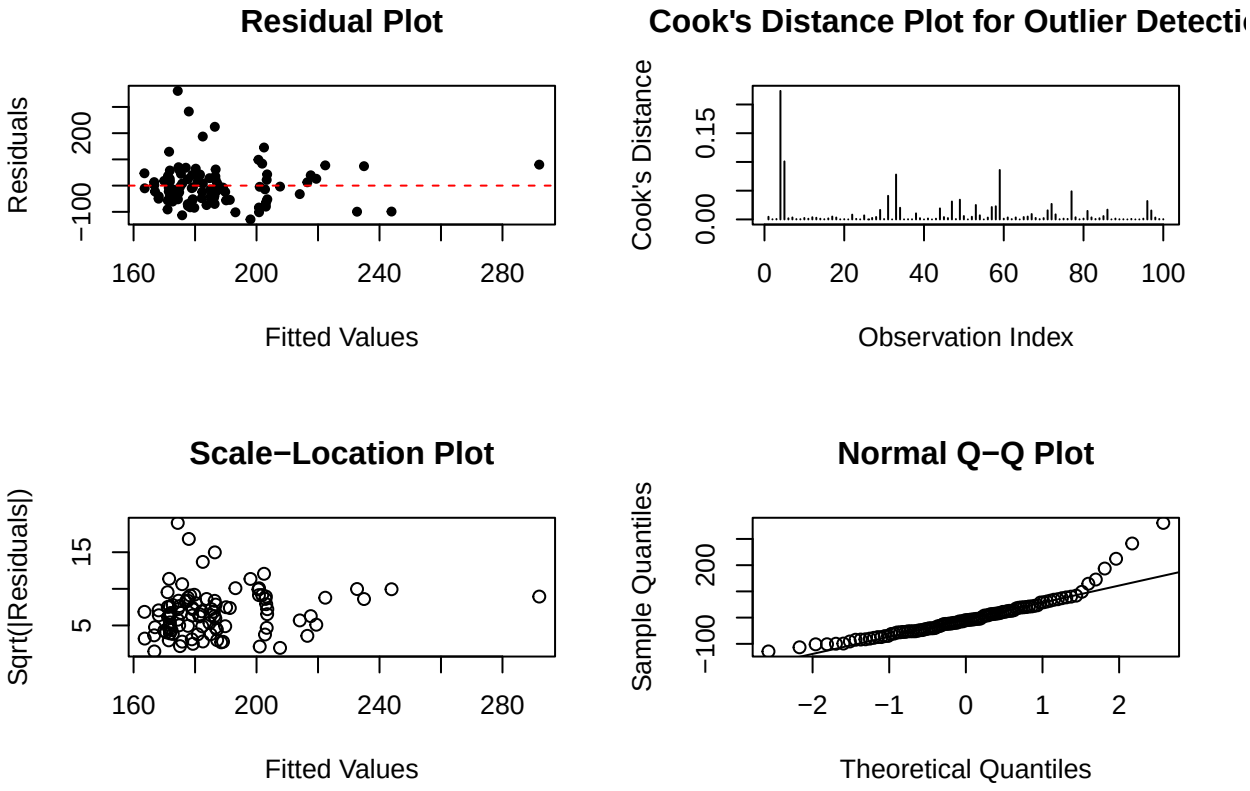
```
##
## Assumption 2: Checking for no excessive outliers...

## Observations with a Cook's distance greater than 0.04 may be considered influential:
##  4  5 31 33 59 77
##  4  5 31 33 59 77
```

```
assumption.3.check.homoskedasticity(m.P121)
```

```
##
## Assumption 3: Checking for constant variance (homoskedasticity)...
```

```
assumption.4.check.normality(m.P121)
```

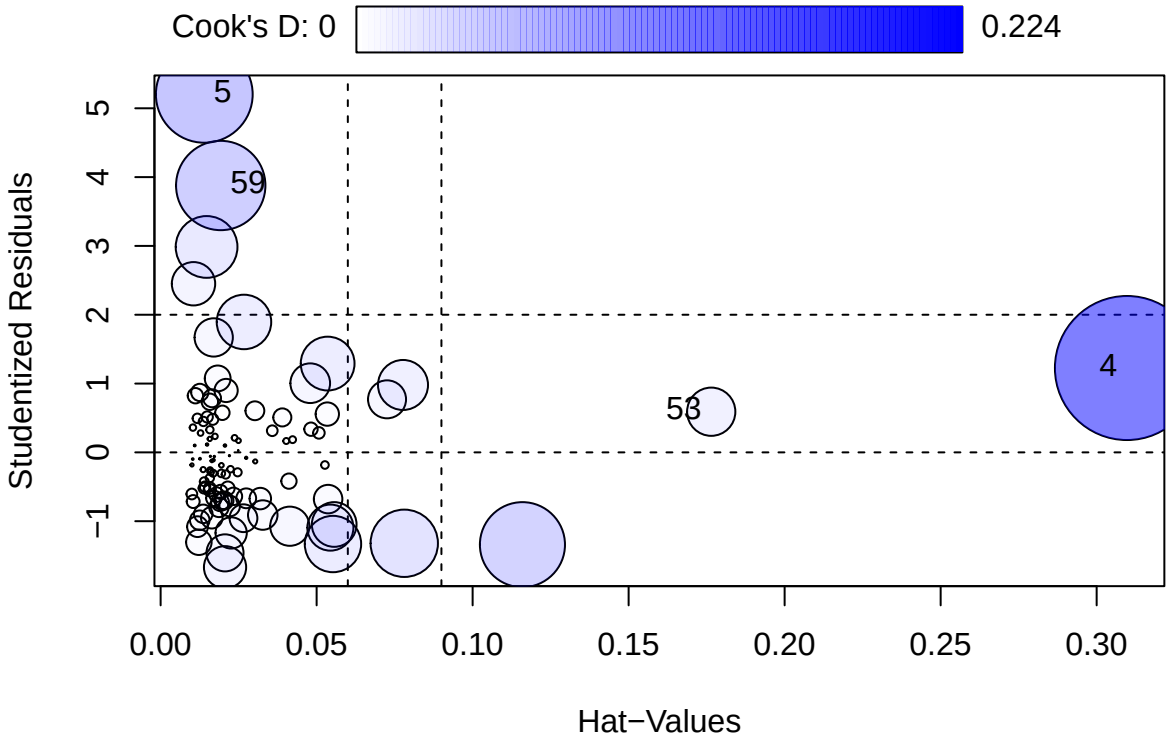
```
##
## Assumption 4: Checking for normally distributed errors...
```




```
par(mfrow=c(1,1))
assumption.5.check.independence(m.P121)
```

```
##
## Assumption 5: Checking for independence of errors...

##  lag Autocorrelation D-W Statistic p-value
##    1      0.05725871      1.878728   0.496
##  Alternative hypothesis: rho != 0
```

```
# Visualize outliers, leverage points, and influential points
car::influencePlot(m.P121,id=list(labels=row.names(df1_P121)))
```



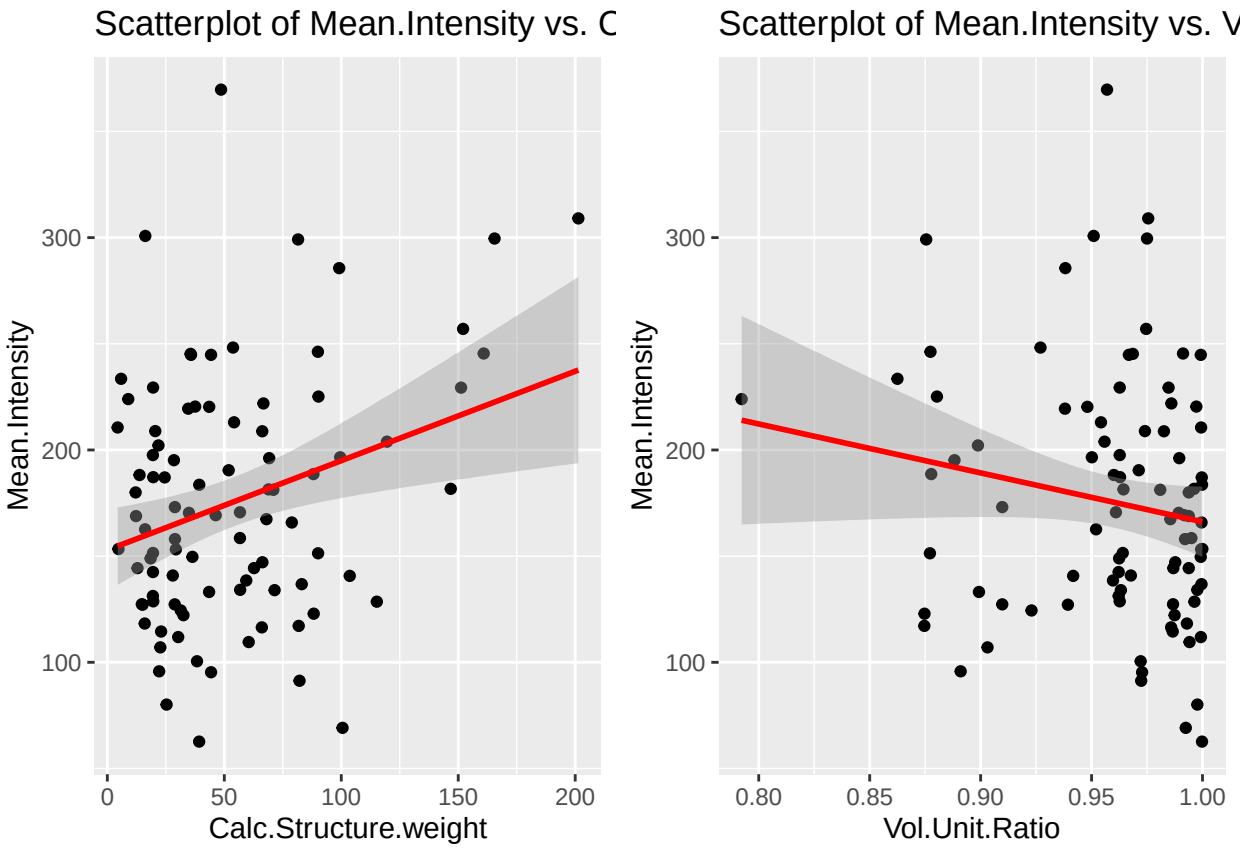

```
##      StudRes        Hat      CookD
## 4  1.2259066 0.30971242 0.22360242
## 5  5.2025371 0.01399660 0.10094518
## 53 0.5907193 0.17647374 0.02509392
## 59 3.8779788 0.01926889 0.08603865
```

```
# Outliers detected rows: 4, 53, 59, 5
########## Treating outliers.
# ------------------------------------------------------------------------
cooks.d = cooks.distance(m.P1211)
influence = cooks.d[(cooks.d > (3*mean(cooks.d, na.rm = T)))]
influence # see that most of the rows stated above are in this list
```

```
##          4          5         31         33         49         59         77
## 0.22360242 0.10094518 0.04107084 0.07802573 0.03429384 0.08603865 0.04890137
##         96
## 0.03200839
```

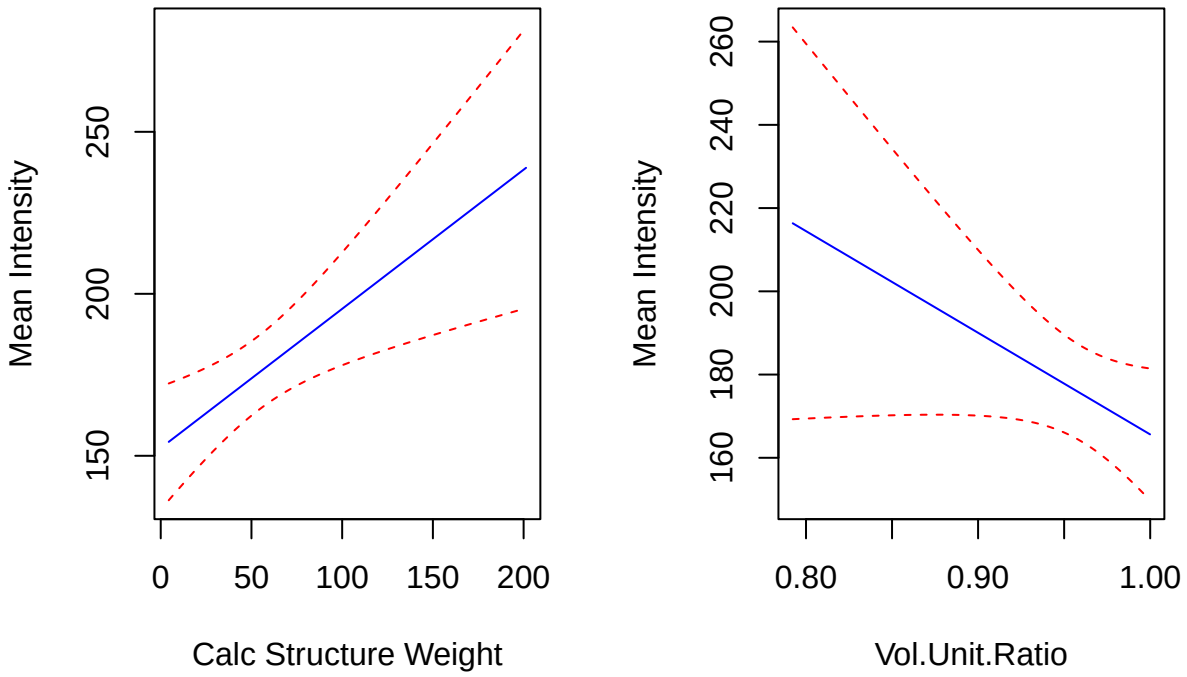
```
row.influence = names(influence)
outliers = df1_P121[row.influence,]
df1_P121.reduced = df1_P121 %>% anti_join(outliers)
########## Any improvement to the model?
# ------------------------------------------------------------------------
m.P121.R = lm(Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio, data = df1_P121.reduced)
# ------------------------------------------------------------------------
```

```r
# Scatterplot of Calc.Structure.weight vs. Mean.Intensity with regression line
a = ggplot(data = df1_P121.reduced, aes(x = Calc.Structure.weight, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Calc.Structure.weight", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Calc.Structure.weight")

# Scatterplot of Vol.Unit.Ratio vs. Mean.Intensity with regression line
b =ggplot(data = df1_P121.reduced, aes(x = Vol.Unit.Ratio, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Vol.Unit.Ratio", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Vol.Unit.Ratio")
ggarrange(a,b)
```




```r
# ------------------------------------------------------------------------
par(mfrow=c(1,2))
# ------------------------------------------------------------------------
# For Calc.Structure.weight
calc_structure_weight_mean <- mean(df1_P121.reduced$Calc.Structure.weight, na.rm = TRUE)
calc_structure_weight_range <- seq(from = min(df1_P121.reduced$Calc.Structure.weight),
                                   to = max(df1_P121.reduced$Calc.Structure.weight), length.out = 100)
vol_unit_ratio_mean <- mean(df1_P121.reduced$Vol.Unit.Ratio, na.rm = TRUE)

x_new <- data.frame(Calc.Structure.weight = calc_structure_weight_range,
```

```
                    Vol.Unit.Ratio = rep(vol_unit_ratio_mean, 100))

conf.int.1 <- predict(m.P121.R, newdata = x_new, interval = "confidence")

plot(x_new$Calc.Structure.weight, conf.int.1[, "fit"], type = "l", col = "blue",
     ylim = range(conf.int.1[, c("lwr", "upr")]),
     ylab = "Mean Intensity", xlab = "Calc Structure Weight")
lines(x_new$Calc.Structure.weight, conf.int.1[, "lwr"], col = "red", lty = 2)
lines(x_new$Calc.Structure.weight, conf.int.1[, "upr"], col = "red", lty = 2)
# ------------------------------------------------------------------------
# For Vol.Unit.Ratio
vol_unit_ratio_range <- seq(from = min(df1_P121.reduced$Vol.Unit.Ratio),
                            to = max(df1_P121.reduced$Vol.Unit.Ratio), length.out = 100)

x_new <- data.frame(Vol.Unit.Ratio = vol_unit_ratio_range,
                    Calc.Structure.weight = rep(calc_structure_weight_mean, 100))

conf.int.2 <- predict(m.P121.R, newdata = x_new, interval = "confidence")

plot(x_new$Vol.Unit.Ratio, conf.int.2[, "fit"], type = "l", col = "blue",
     ylim = range(conf.int.2[, c("lwr", "upr")]),
     ylab = "Mean Intensity", xlab = "Vol.Unit.Ratio")
lines(x_new$Vol.Unit.Ratio, conf.int.2[, "lwr"], col = "red", lty = 2)
lines(x_new$Vol.Unit.Ratio, conf.int.2[, "upr"], col = "red", lty = 2)
```

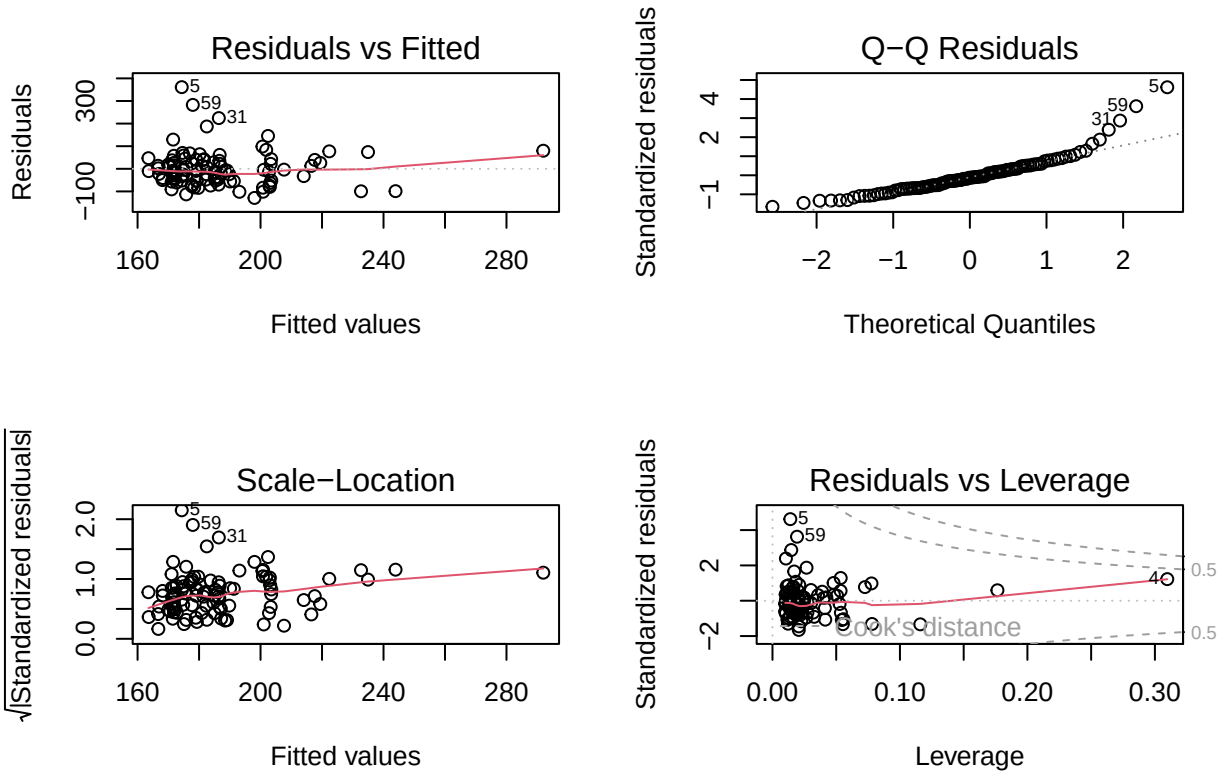


```
# ------------------------------------------------------------------------
par(mfrow=c(1,1))
confint(m.P121.R)
```

```
##                                2.5 %      97.5 %
## (Intercept)             124.9962123 648.6922245
## Calc.Structure.weight     0.1465087   0.7125871
## Vol.Unit.Ratio         -516.9029261  28.3716603
```

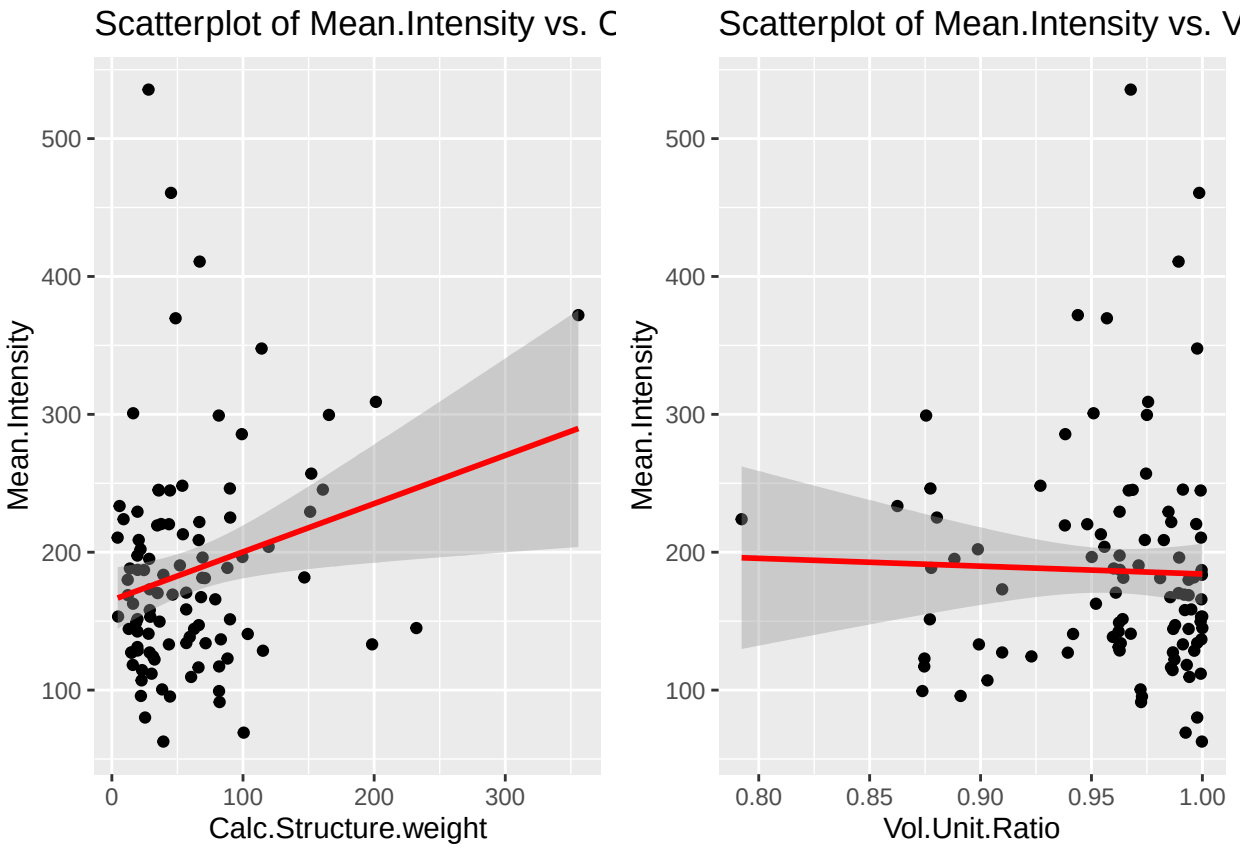
For C121 Space Group

```
########## MODEL for C121
# ------------------------------------------------------------------------
m.C121 = lm(Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio, data = df1_C121)
par(mfrow=c(2,2))
plot(m.C121)
```




```
par(mfrow=c(1,1))
########## Diagnostics
# ------------------------------------------------------------------------
# Scatterplot of Calc.Structure.weight vs. Mean.Intensity with regression line
a = ggplot(data = df1_C121, aes(x = Calc.Structure.weight, y = Mean.Intensity)) +
  geom_point() +
```

```
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Calc.Structure.weight", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Calc.Structure.weight")

# Scatterplot of Vol.Unit.Ratio vs. Mean.Intensity with regression line
b = ggplot(data = df1_C121, aes(x = Vol.Unit.Ratio, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Vol.Unit.Ratio", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Vol.Unit.Ratio")
ggarrange(a,b)
```




```
# This does not show linear relationship off the bat.
# individually
par(mfrow=c(2,2))
assumption.1.check.linearity(m.C121)
```

```
##
## Assumption 1: Checking for linearity...
```

```
assumption.2.check.outliers(m.C121) # most definitely influence of outliers.
```

```
##
## Assumption 2: Checking for no excessive outliers...
```

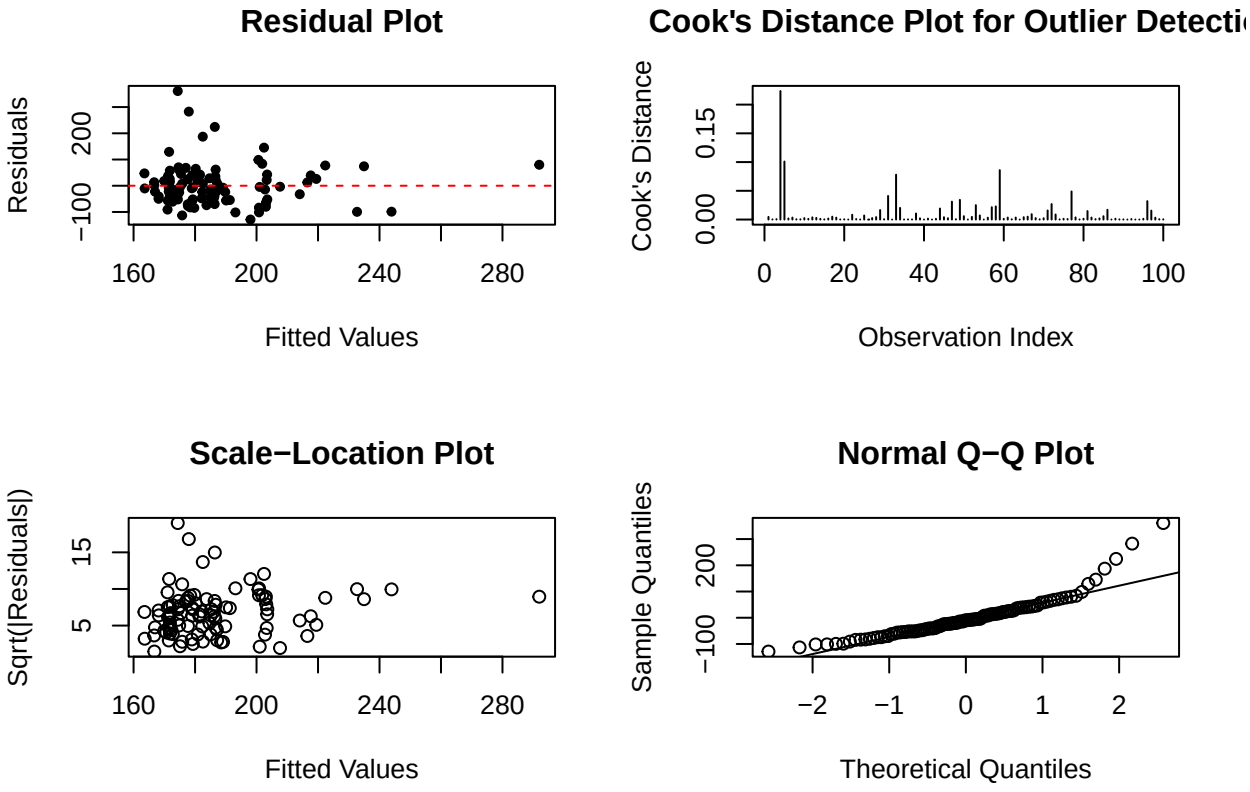
```
## Observations with a Cook's distance greater than 0.04 may be considered influential:
##  4  5 31 33 59 77
##  4  5 31 33 59 77
```

```
assumption.3.check.homoskedasticity(m.C121)
```

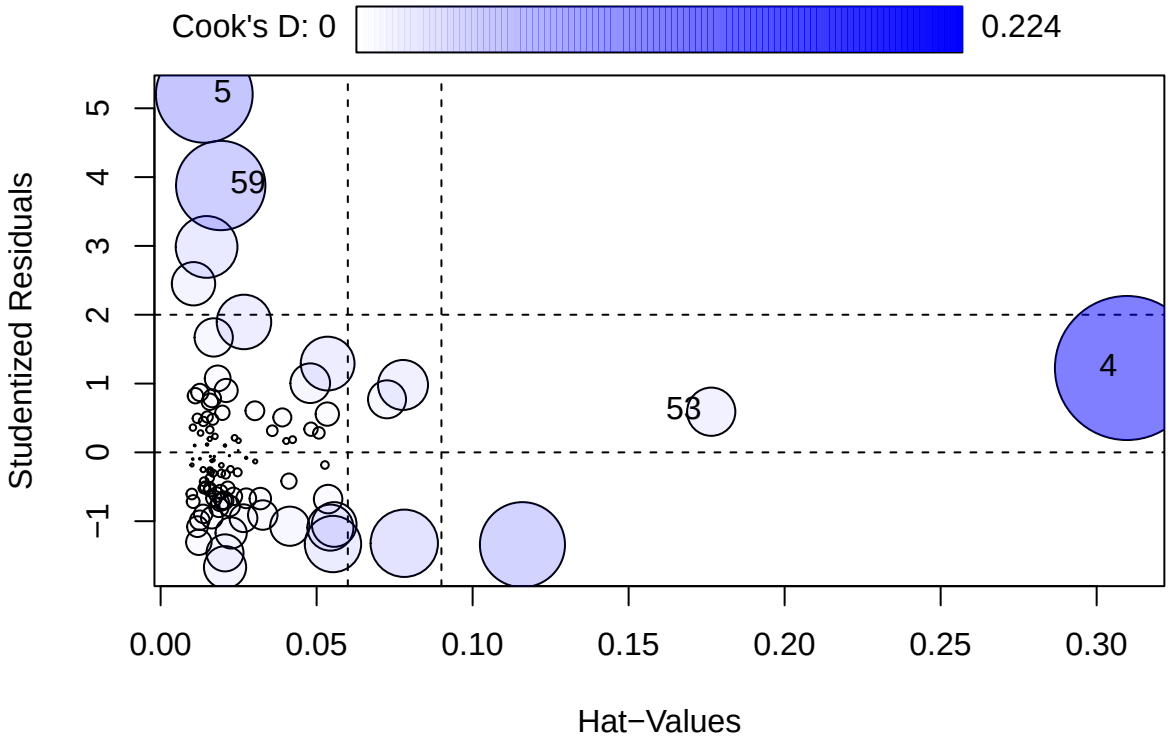
```
##
## Assumption 3: Checking for constant variance (homoskedasticity)...
```

```
assumption.4.check.normality(m.C121)
```

```
##
## Assumption 4: Checking for normally distributed errors...
```




```
par(mfrow=c(1,1))
# Visualize outliers, leverage points, and influential points
car::influencePlot(m.C121,id=list(labels=row.names(df1_C121)))
```



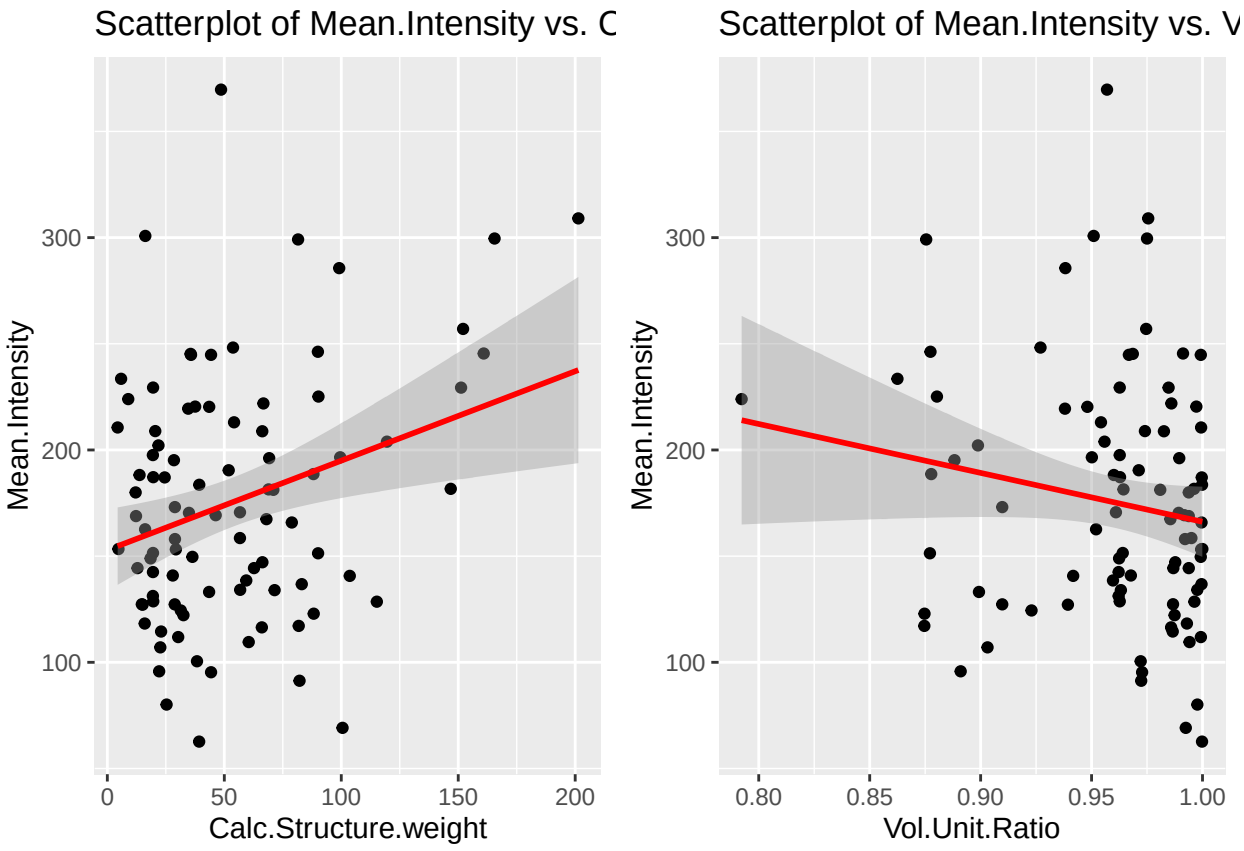

```
##      StudRes        Hat      CookD
## 4  1.2259066 0.30971242 0.22360242
## 5  5.2025371 0.01399660 0.10094518
## 53 0.5907193 0.17647374 0.02509392
## 59 3.8779788 0.01926889 0.08603865
```

```
# Outliers detected rows: 4, 53, 59, 5
########## Treating outliers.
# ------------------------------------------------------------------------
cooks.d = cooks.distance(m.C121)
influence = cooks.d[(cooks.d > (3*mean(cooks.d, na.rm = T)))]
influence # see that most of the rows stated above are in this list
```

```
##          4          5         31         33         49         59         77
## 0.22360242 0.10094518 0.04107084 0.07802573 0.03429384 0.08603865 0.04890137
##         96
## 0.03200839
```

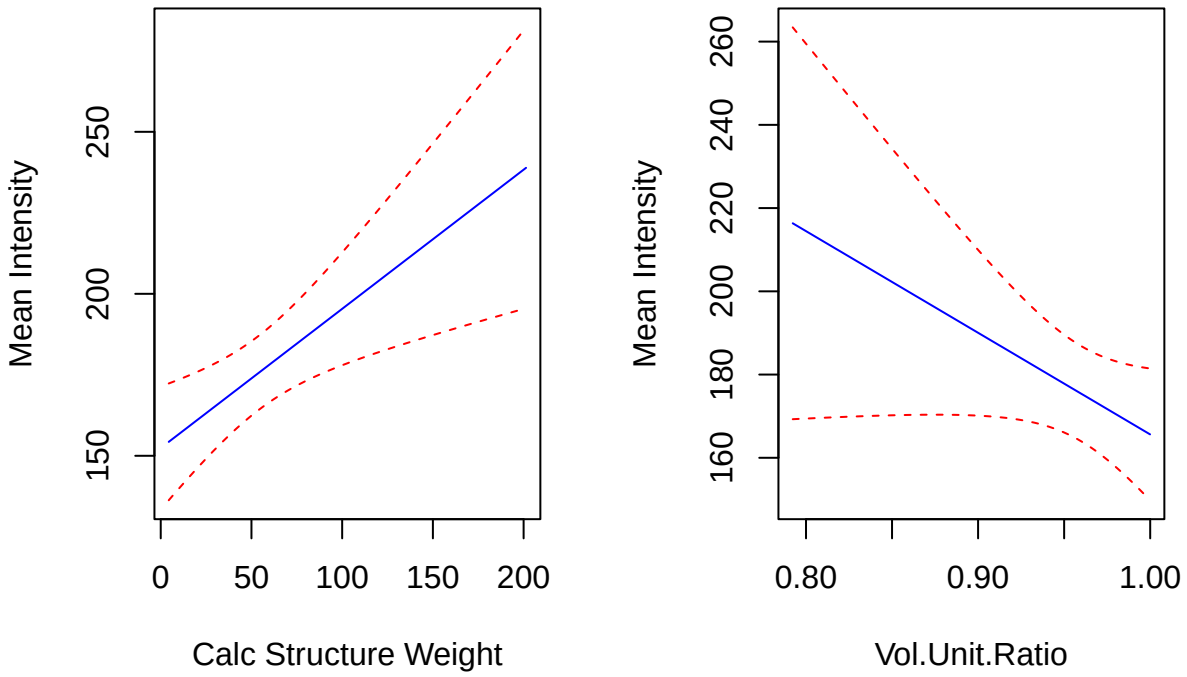
```
row.influence = names(influence)
outliers = df1_C121[row.influence,]
df1_C121.reduced = df1_C121 %>% anti_join(outliers)
########## Any improvement to the model?
# ------------------------------------------------------------------------
m.C121.R = lm(Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio, data = df1_C121.reduced)
# ------------------------------------------------------------------------
```

```r
# Scatterplot of Calc.Structure.weight vs. Mean.Intensity with regression line
a = ggplot(data = df1_C121.reduced, aes(x = Calc.Structure.weight, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Calc.Structure.weight", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Calc.Structure.weight")

# Scatterplot of Vol.Unit.Ratio vs. Mean.Intensity with regression line
b = ggplot(data = df1_C121.reduced, aes(x = Vol.Unit.Ratio, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Vol.Unit.Ratio", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Vol.Unit.Ratio")
ggarrange(a,b)
```




```r
# ------------------------------------------------------------------------
par(mfrow=c(1,2))
# ------------------------------------------------------------------------
# For Calc.Structure.weight
calc_structure_weight_mean <- mean(df1_C121.reduced$Calc.Structure.weight, na.rm = TRUE)
calc_structure_weight_range <- seq(from = min(df1_C121.reduced$Calc.Structure.weight),
                                   to = max(df1_C121.reduced$Calc.Structure.weight), length.out = 100)
vol_unit_ratio_mean <- mean(df1_C121.reduced$Vol.Unit.Ratio, na.rm = TRUE)

x_new <- data.frame(Calc.Structure.weight = calc_structure_weight_range,
```

```
                    Vol.Unit.Ratio = rep(vol_unit_ratio_mean, 100))

conf.int.1 <- predict(m.C121.R, newdata = x_new, interval = "confidence")

plot(x_new$Calc.Structure.weight, conf.int.1[, "fit"], type = "l", col = "blue",
     ylim = range(conf.int.1[, c("lwr", "upr")]),
     ylab = "Mean Intensity", xlab = "Calc Structure Weight")
lines(x_new$Calc.Structure.weight, conf.int.1[, "lwr"], col = "red", lty = 2)
lines(x_new$Calc.Structure.weight, conf.int.1[, "upr"], col = "red", lty = 2)
# ------------------------------------------------------------------------
# For Vol.Unit.Ratio
vol_unit_ratio_range <- seq(from = min(df1_C121.reduced$Vol.Unit.Ratio),
                            to = max(df1_C121.reduced$Vol.Unit.Ratio), length.out = 100)

x_new <- data.frame(Vol.Unit.Ratio = vol_unit_ratio_range,
                    Calc.Structure.weight = rep(calc_structure_weight_mean, 100))

conf.int.2 <- predict(m.C121.R, newdata = x_new, interval = "confidence")

plot(x_new$Vol.Unit.Ratio, conf.int.2[, "fit"], type = "l", col = "blue",
     ylim = range(conf.int.2[, c("lwr", "upr")]),
     ylab = "Mean Intensity", xlab = "Vol.Unit.Ratio")
lines(x_new$Vol.Unit.Ratio, conf.int.2[, "lwr"], col = "red", lty = 2)
lines(x_new$Vol.Unit.Ratio, conf.int.2[, "upr"], col = "red", lty = 2)
```

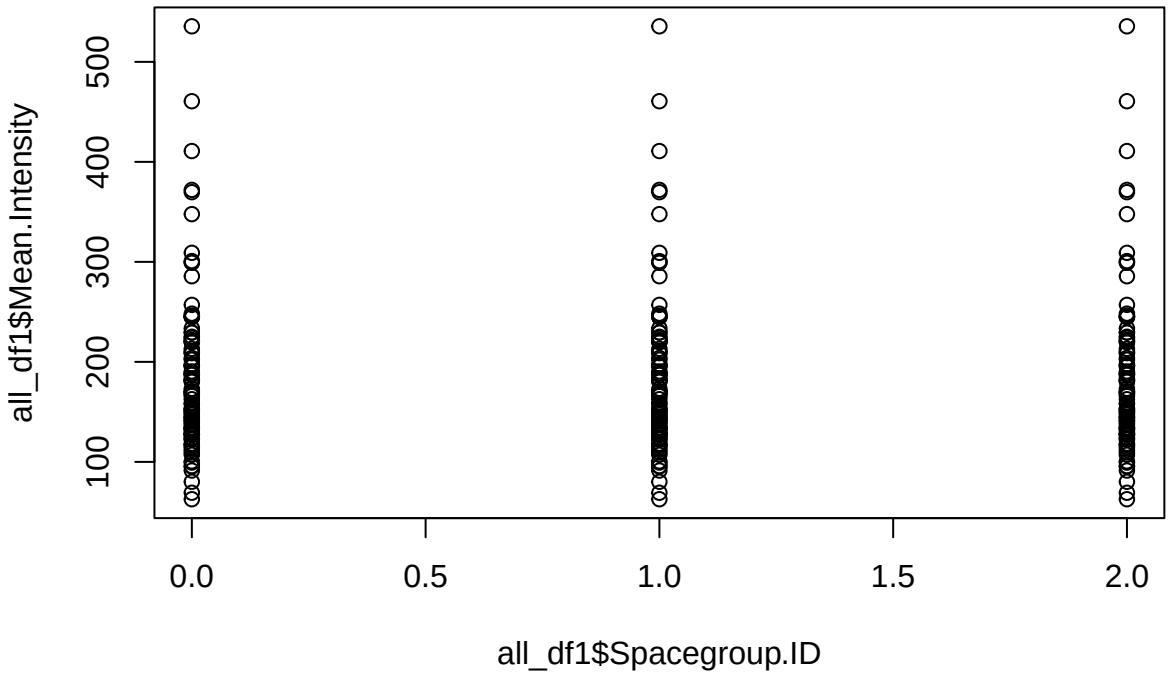


```
# ------------------------------------------------------------------------
par(mfrow=c(1,1))
confint(m.C121.R)
```

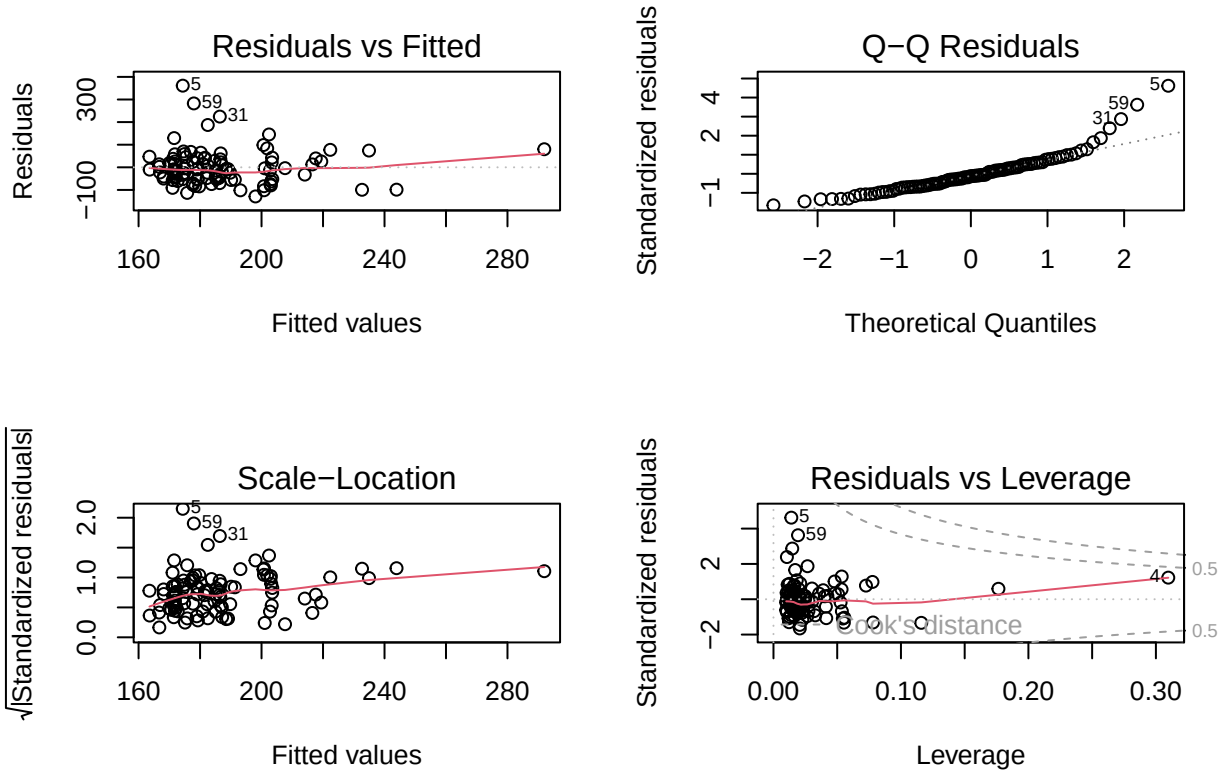
```
##                                2.5 %      97.5 %
## (Intercept)            124.9962123 648.6922245
## Calc.Structure.weight    0.1465087   0.7125871
## Vol.Unit.Ratio        -516.9029261  28.3716603
```

All of the steps for space group P1211 seem to hold for P121, and C121 space groups. As seen from the plot below, there's not a lot of variation in the points of the three space groups. With P1211 having a `Spacegroup.ID` of 0, P121 that of 1, and C121 has 2.

```
plot(all_df1$Spacegroup.ID, all_df1$Mean.Intensity, data=all_df1) # does not look like significant diff
```



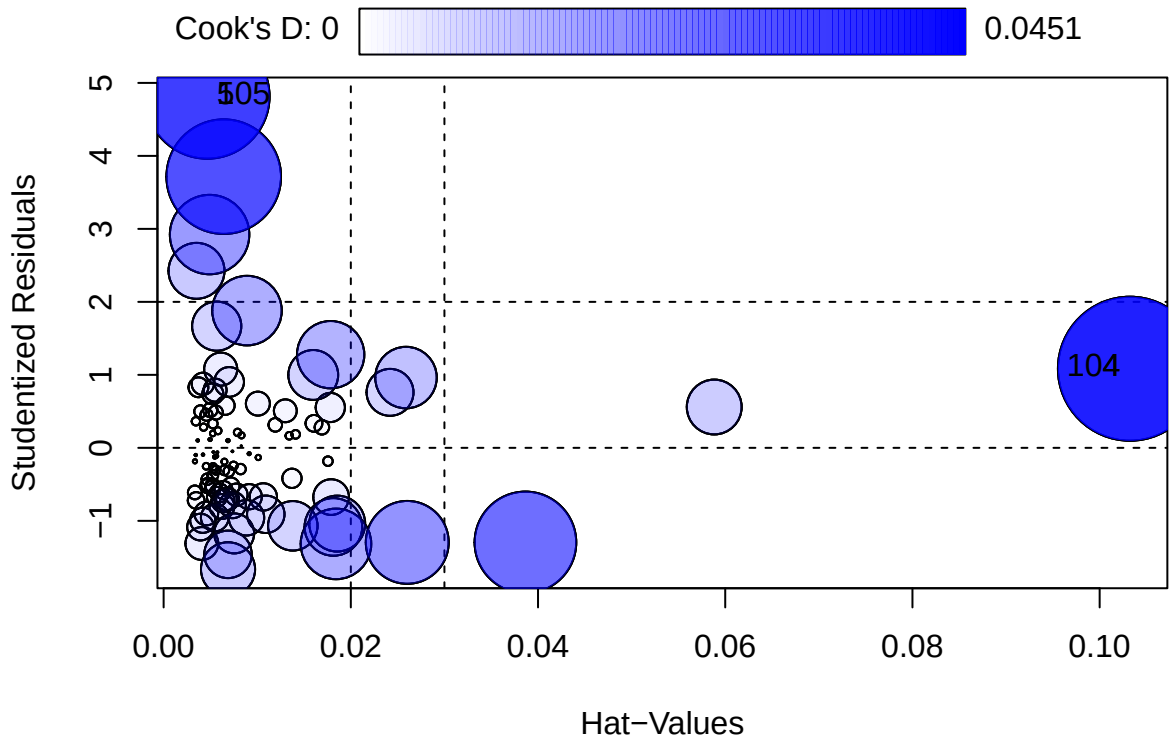

```
############### FULL MODEL
# Since the mean intensity does not seem to change for the given spacegroups...
model.all = lm(Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio, data = all_df1)
par(mfrow=c(2,2))
plot(m.C121)
```




```
par(mfrow=c(1,1))
summary(model.all)
```

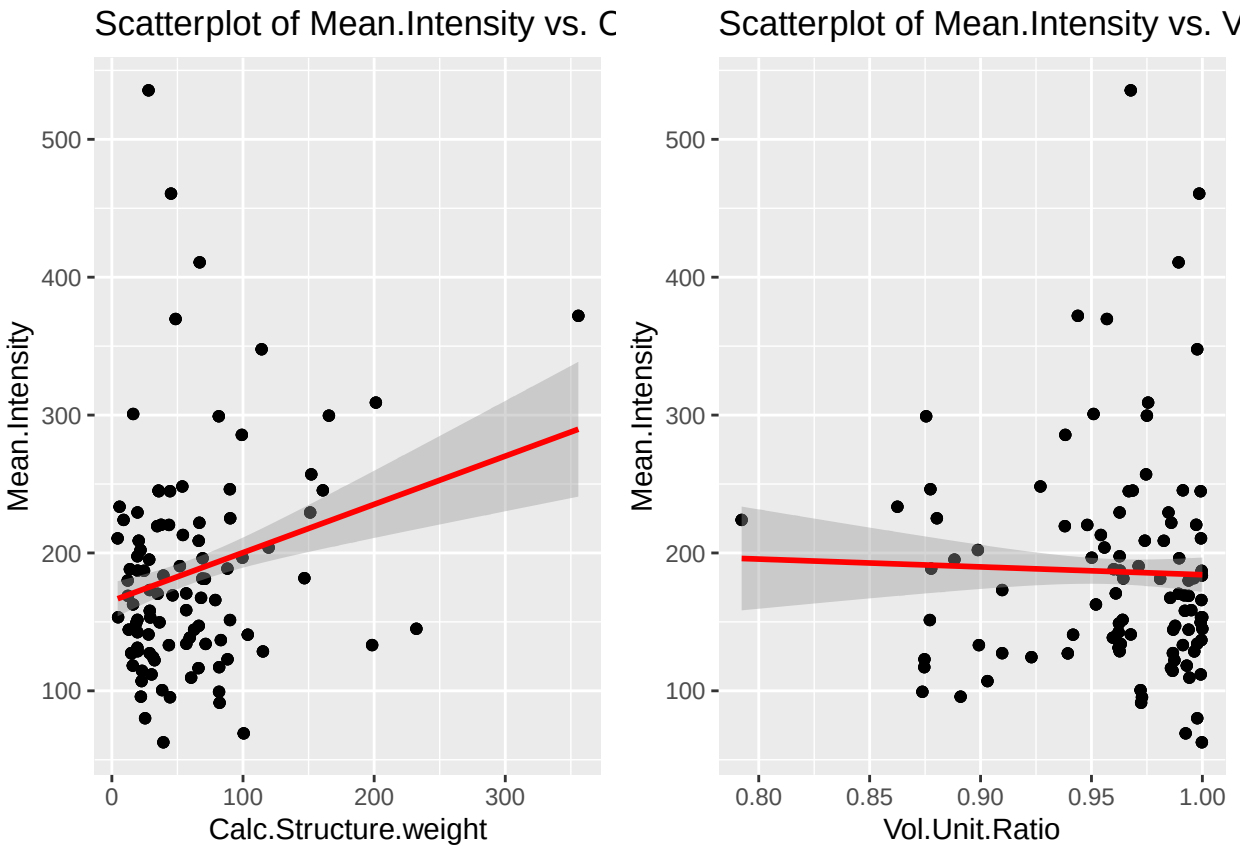
```
##
## Call:
## lm(formula = Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio,
##     data = all_df1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -128.89  -52.03   -9.96   35.89  361.16
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           241.40877  103.09396   2.342   0.0199 *
## Calc.Structure.weight   0.35302    0.08294   4.256 2.79e-05 ***
## Vol.Unit.Ratio        -79.47183  107.35195  -0.740   0.4597
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 77.95 on 297 degrees of freedom
## Multiple R-squared:  0.05832,    Adjusted R-squared:  0.05198
## F-statistic: 9.197 on 2 and 297 DF,  p-value: 0.0001332
```

```
car::influencePlot(model.all)
```




```
##       StudRes         Hat      CookD
## 4   1.084103 0.103237474 0.04507377
## 5   4.814569 0.004665534 0.03370129
## 104 1.084103 0.103237474 0.04507377
## 105 4.814569 0.004665534 0.03370129
```

```
########## Diagnostics
# ------------------------------------------------------------------------
# Scatterplot of Calc.Structure.weight vs. Mean.Intensity with regression line
a = ggplot(data = all_df1, aes(x = Calc.Structure.weight, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Calc.Structure.weight", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Calc.Structure.weight")

# Scatterplot of Vol.Unit.Ratio vs. Mean.Intensity with regression line
b = ggplot(data = all_df1, aes(x = Vol.Unit.Ratio, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Vol.Unit.Ratio", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Vol.Unit.Ratio")
ggarrange(a,b)
```



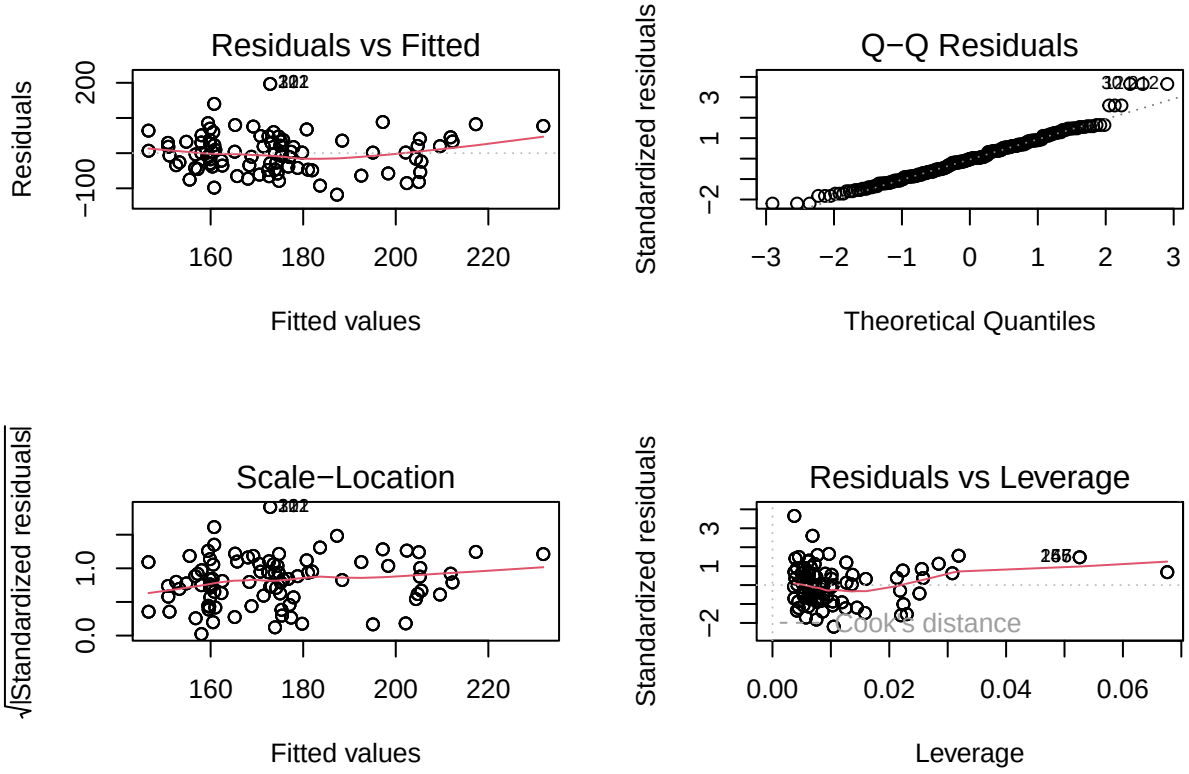

```
# ------------------------------------------------------------------------
######### Remove outliers for model.all
cooks.d = cooks.distance(model.all)
influence = cooks.d[(cooks.d > (3*mean(cooks.d, na.rm = T)))]
influence # see that most of the rows stated above are in this list
```

```
##           4           5          31          33          47          49
## 0.045073774 0.033701290 0.013698072 0.022447183 0.009850441 0.010807950
##          59          77          96         104         105         131
## 0.028518890 0.014905677 0.010501543 0.045073774 0.033701290 0.013698072
##         133         147         149         159         177         196
## 0.022447183 0.009850441 0.010807950 0.028518890 0.014905677 0.010501543
##         204         205         231         233         247         249
## 0.045073774 0.033701290 0.013698072 0.022447183 0.009850441 0.010807950
##         259         277         296
## 0.028518890 0.014905677 0.010501543
```

```
row.influence = names(influence)
outliers = all_df1[row.influence,]
all_df1.reduced = all_df1 %>% anti_join(outliers)
```

```
## Joining with `by = join_by(Spacegroup.ID, PDB.ID, Calc.Structure.weight, a, b,
## c, Alpha, Beta, Gamma, Unit.Vol, Crystal.Vol, Vol.Unit.Ratio, Mean.Intensity,
## Max.Intensity, Min.Intensity, MaxMin.Intensity.Dif, Mean.Phase, Max.Phase,
## Min.Phase, MaxMin.Phase.Difference)`
```

```r
########## Any improvement to the model?
model.all.reduced = lm(Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio, data = all_df1.reduced)
par(mfrow=c(2,2))
plot(model.all.reduced)
```

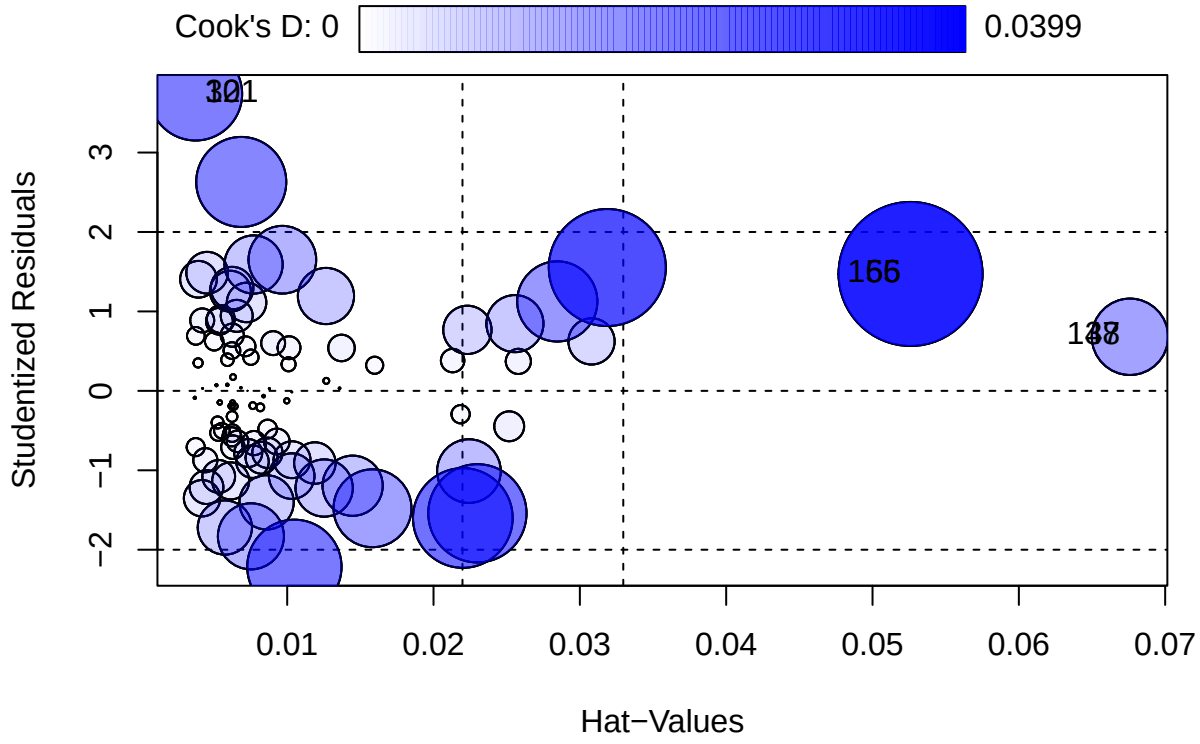



```r
par(mfrow=c(1,1))
summary(model.all.reduced)
```

```
##
## Call:
## lm(formula = Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio,
##     data = all_df1.reduced)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -118.154  -38.217   -0.805   33.668  196.809
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            337.80247   76.17394   4.435 1.34e-05 ***
## Calc.Structure.weight    0.40975    0.08071   5.077 7.15e-07 ***
## Vol.Unit.Ratio        -193.16104   79.31563  -2.435   0.0155 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 54 on 270 degrees of freedom
## Multiple R-squared:  0.1016, Adjusted R-squared:  0.09497
## F-statistic: 15.27 on 2 and 270 DF,  p-value: 5.212e-07
```

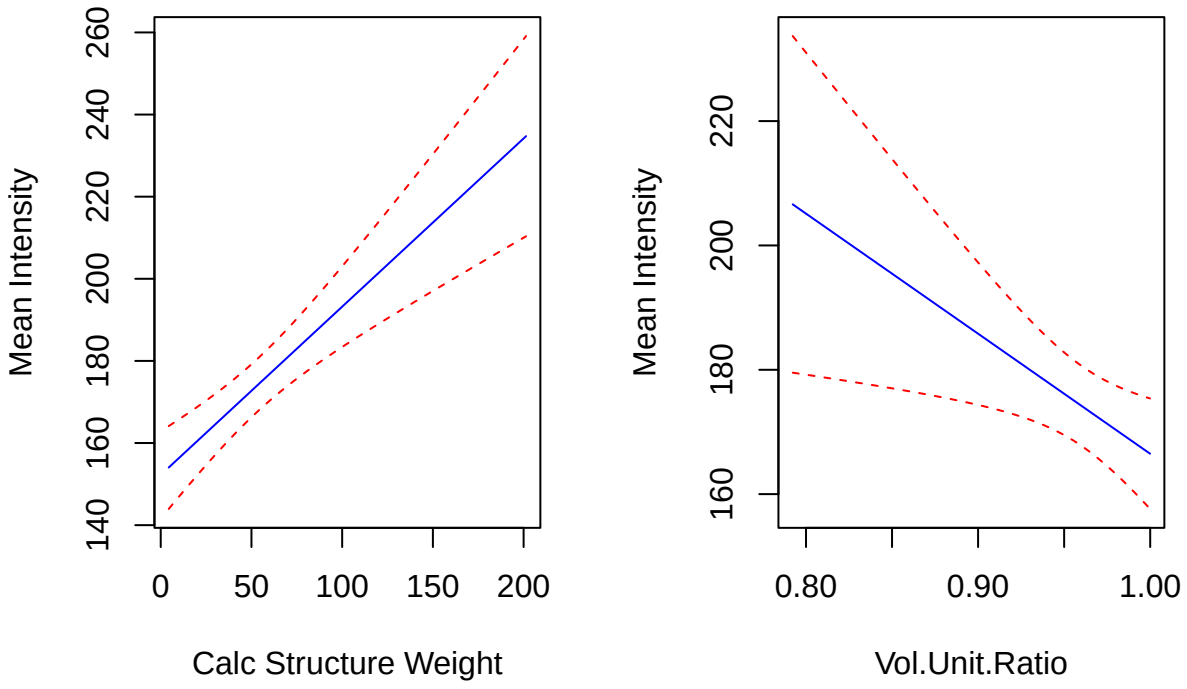
```
car::influencePlot(model.all.reduced)
```




```
##       StudRes         Hat      CookD
## 30  3.7378428 0.003738714 0.01667596
## 47  0.6806594 0.067592722 0.01121752
## 65  1.4714698 0.052598236 0.03989773
## 121 3.7378428 0.003738714 0.01667596
## 138 0.6806594 0.067592722 0.01121752
## 156 1.4714698 0.052598236 0.03989773
```

```
# ------------------------------------------------------------------------
par(mfrow=c(1,2))
# ------------------------------------------------------------------------
# For Calc.Structure.weight
calc_structure_weight_mean <- mean(all_df1.reduced$Calc.Structure.weight, na.rm = TRUE)
calc_structure_weight_range <- seq(from = min(all_df1.reduced$Calc.Structure.weight),
                                   to = max(all_df1.reduced$Calc.Structure.weight), length.out = 100)
vol_unit_ratio_mean <- mean(all_df1.reduced$Vol.Unit.Ratio, na.rm = TRUE)

x_new <- data.frame(Calc.Structure.weight = calc_structure_weight_range,
                    Vol.Unit.Ratio = rep(vol_unit_ratio_mean, 100))
```

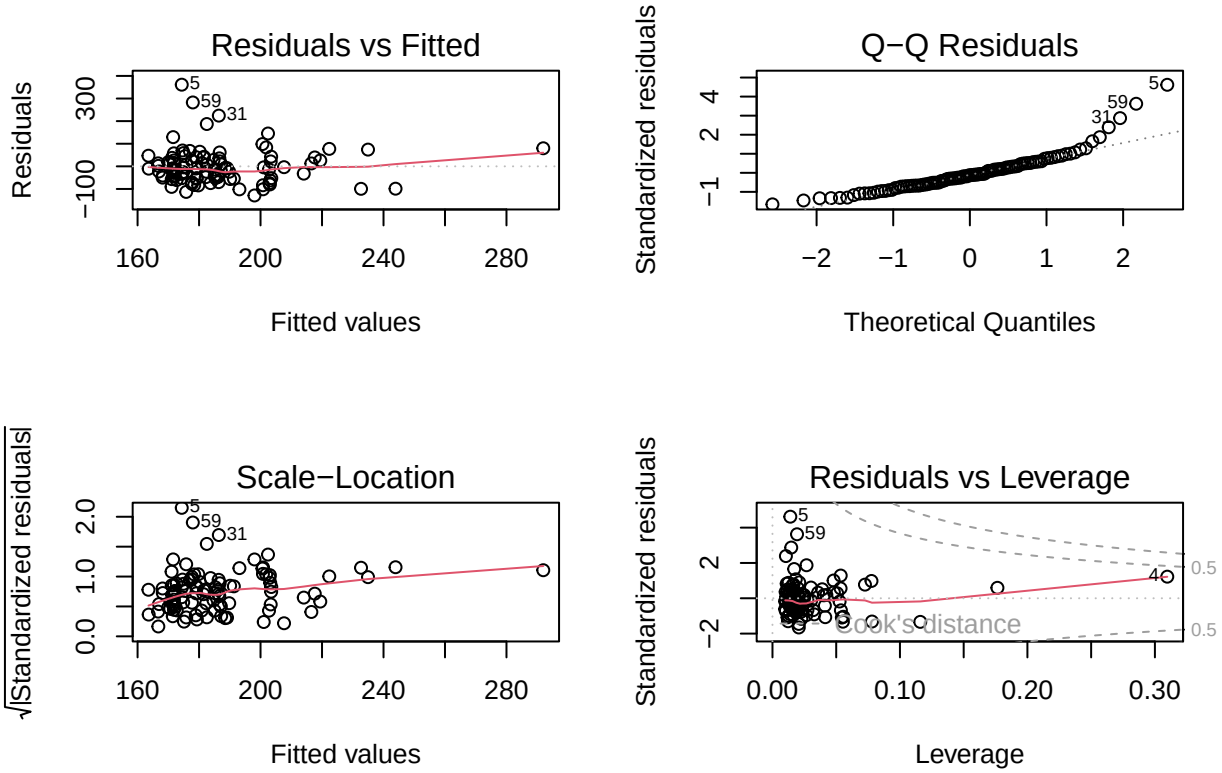
```r
conf.int.1 <- predict(model.all.reduced, newdata = x_new, interval = "confidence")

plot(x_new$Calc.Structure.weight, conf.int.1[, "fit"], type = "l", col = "blue",
     ylim = range(conf.int.1[, c("lwr", "upr")]),
     ylab = "Mean Intensity", xlab = "Calc Structure Weight")
lines(x_new$Calc.Structure.weight, conf.int.1[, "lwr"], col = "red", lty = 2)
lines(x_new$Calc.Structure.weight, conf.int.1[, "upr"], col = "red", lty = 2)
# ------------------------------------------------------------------------------
# For Vol.Unit.Ratio
vol_unit_ratio_range <- seq(from = min(all_df1.reduced$Vol.Unit.Ratio),
                            to = max(all_df1.reduced$Vol.Unit.Ratio), length.out = 100)

x_new <- data.frame(Vol.Unit.Ratio = vol_unit_ratio_range,
                    Calc.Structure.weight = rep(calc_structure_weight_mean, 100))

conf.int.2 <- predict(model.all.reduced, newdata = x_new, interval = "confidence")

plot(x_new$Vol.Unit.Ratio, conf.int.2[, "fit"], type = "l", col = "blue",
     ylim = range(conf.int.2[, c("lwr", "upr")]),
     ylab = "Mean Intensity", xlab = "Vol.Unit.Ratio")
lines(x_new$Vol.Unit.Ratio, conf.int.2[, "lwr"], col = "red", lty = 2)
lines(x_new$Vol.Unit.Ratio, conf.int.2[, "upr"], col = "red", lty = 2)
```



```
# ------------------------------------------------------------------------
par(mfrow=c(1,1))
confint(m.C121.R)
```

```
##                                2.5 %      97.5 %
## (Intercept)            124.9962123 648.6922245
## Calc.Structure.weight    0.1465087   0.7125871
## Vol.Unit.Ratio        -516.9029261  28.3716603
```

```
# Since the mean intensity does not seem to change for the given spacegroups...
model.all = lm(Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio, data = all_df1.reduced)
par(mfrow=c(2,2))
plot(m.C121)
```

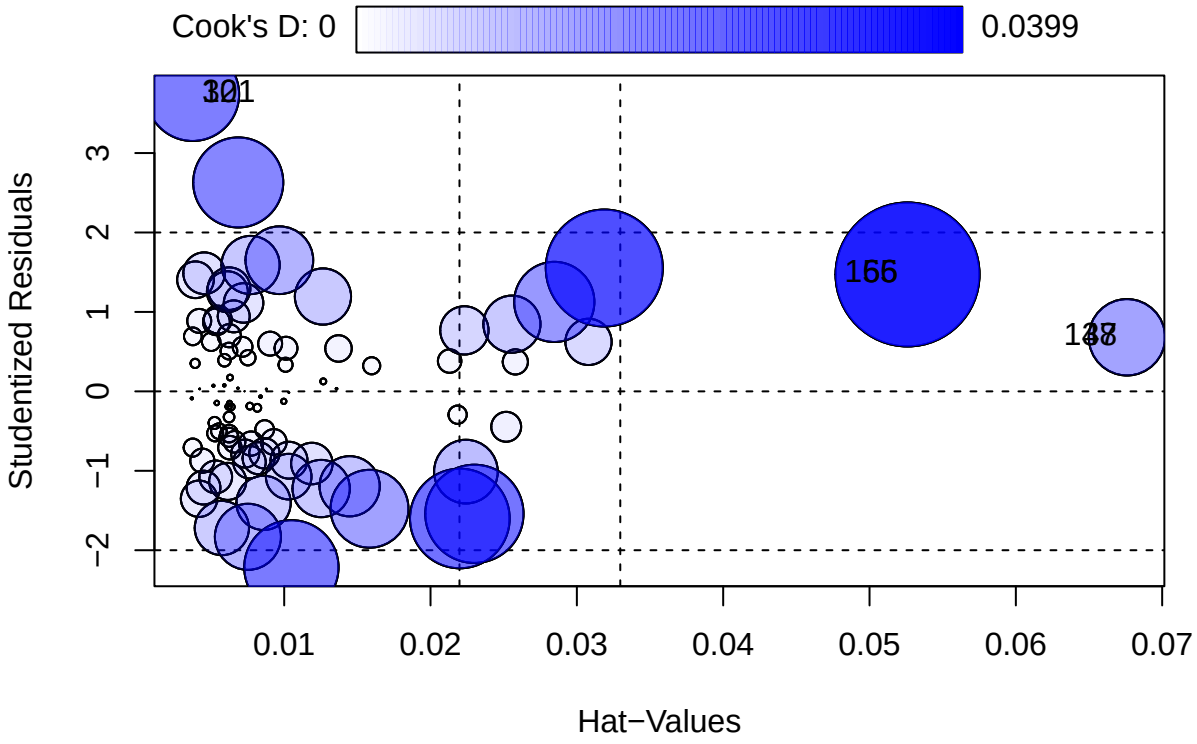



```
par(mfrow=c(1,1))
summary(model.all)
```

```
##
## Call:
## lm(formula = Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio,
##     data = all_df1.reduced)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

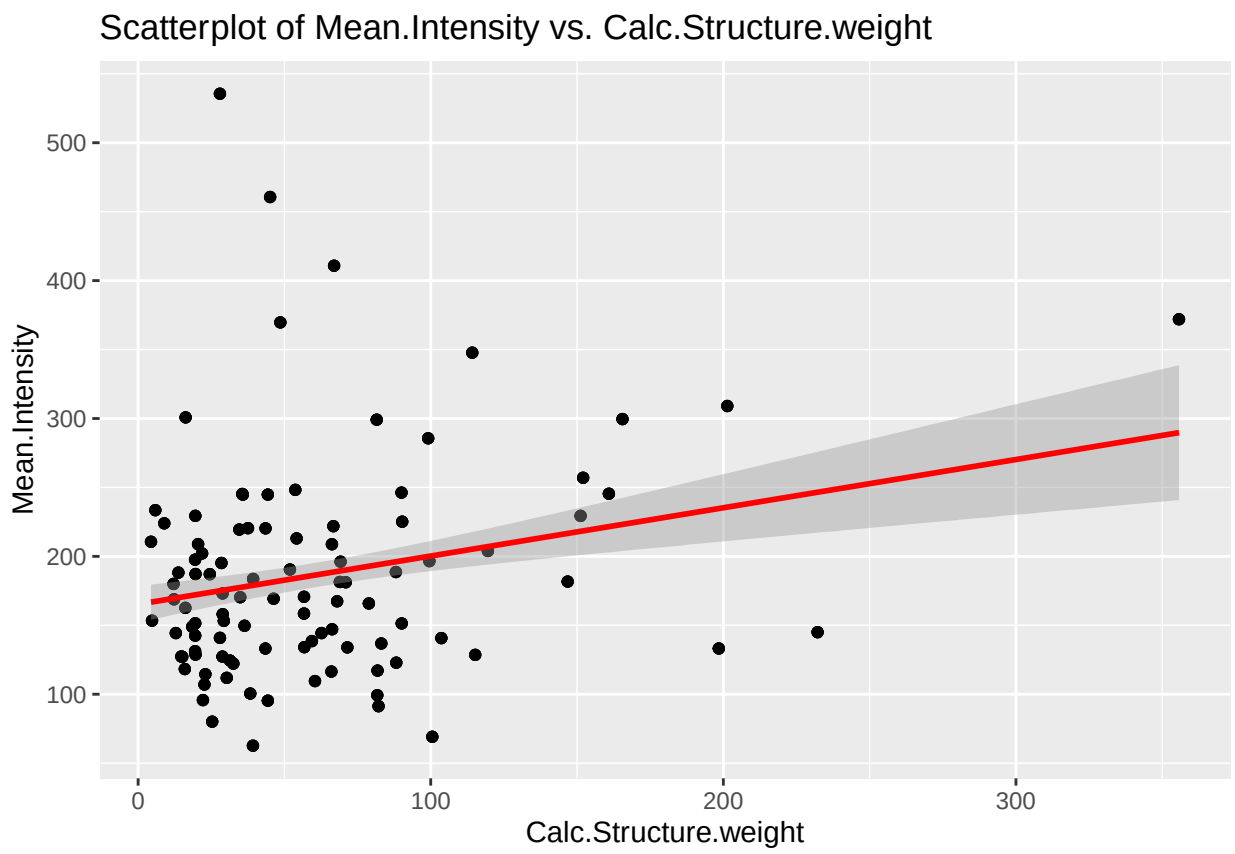
```
## -118.154  -38.217   -0.805   33.668  196.809
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            337.80247   76.17394   4.435 1.34e-05 ***
## Calc.Structure.weight    0.40975    0.08071   5.077 7.15e-07 ***
## Vol.Unit.Ratio        -193.16104   79.31563  -2.435   0.0155 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 54 on 270 degrees of freedom
## Multiple R-squared:  0.1016, Adjusted R-squared:  0.09497
## F-statistic: 15.27 on 2 and 270 DF,  p-value: 5.212e-07
```

```
car::influencePlot(model.all)
```

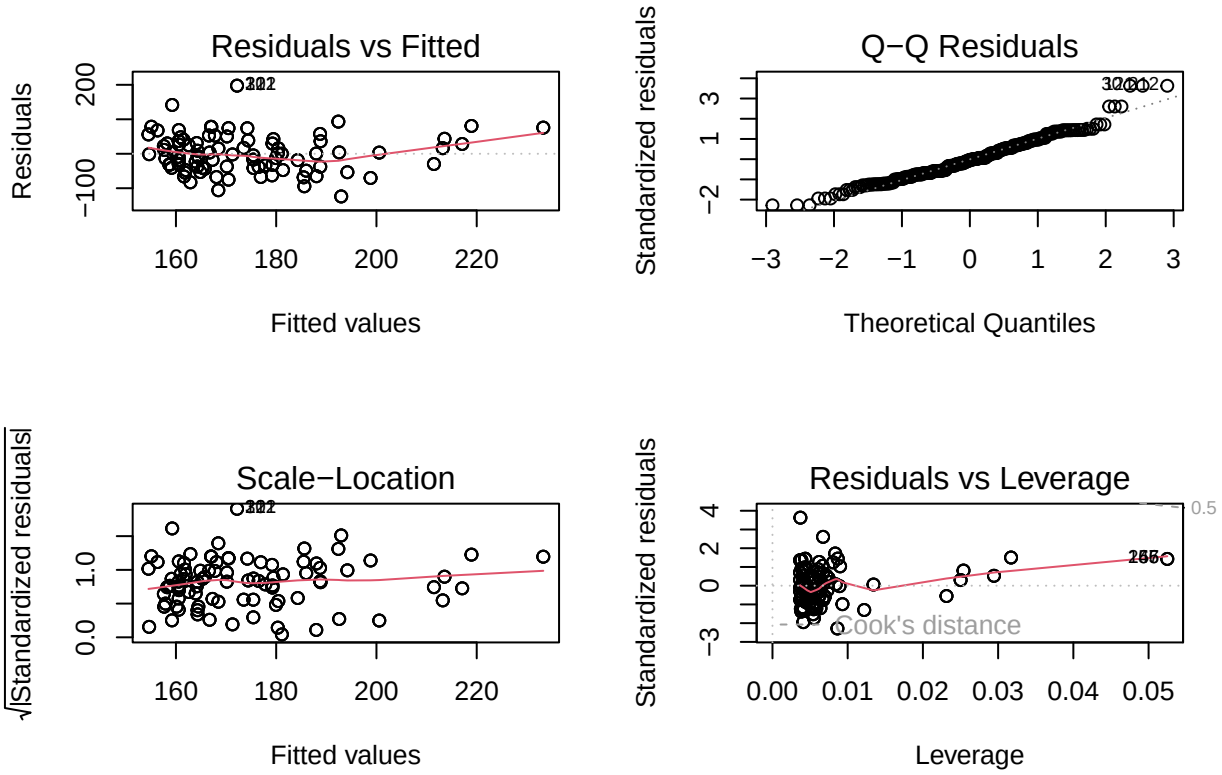



```
##       StudRes         Hat      CookD
## 30  3.7378428 0.003738714 0.01667596
## 47  0.6806594 0.067592722 0.01121752
## 65  1.4714698 0.052598236 0.03989773
## 121 3.7378428 0.003738714 0.01667596
## 138 0.6806594 0.067592722 0.01121752
## 156 1.4714698 0.052598236 0.03989773
```

```
########## Diagnostics
# ------------------------------------------------------------------------
# Scatterplot of Calc.Structure.weight vs. Mean.Intensity with regression line
ggplot(data = all_df1, aes(x = Calc.Structure.weight, y = Mean.Intensity)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(x = "Calc.Structure.weight", y = "Mean.Intensity",
       title = "Scatterplot of Mean.Intensity vs. Calc.Structure.weight")
```



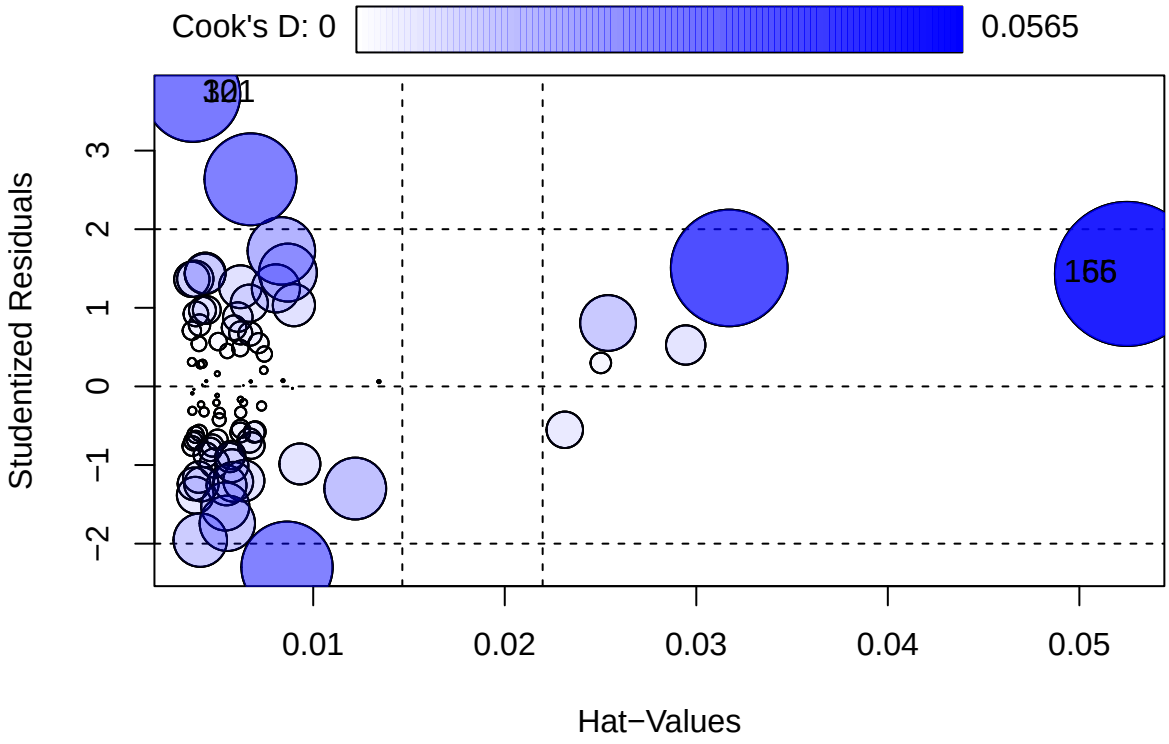

```
# ------------------------------------------------------------------------
########## Any improvement to the model?
model.all.R = lm(Mean.Intensity ~ Calc.Structure.weight, data = all_df1.reduced)
par(mfrow=c(2,2))
plot(model.all.R)
```




```
par(mfrow=c(1,1))
summary(model.all.R)
```

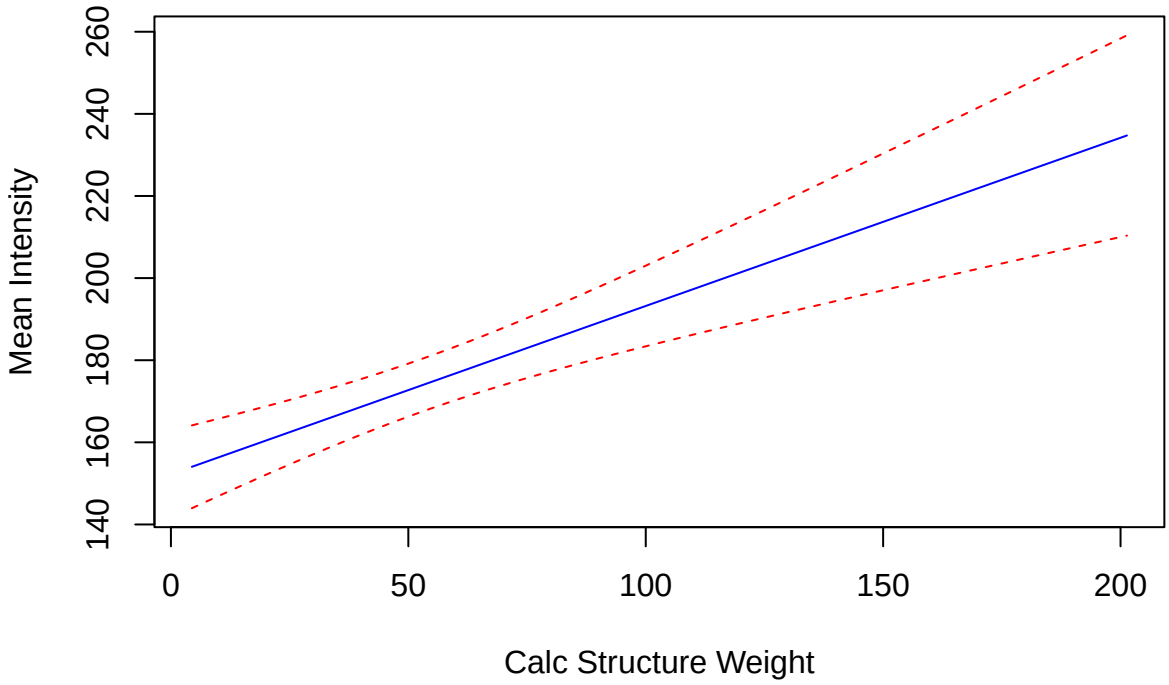
```
##
## Call:
## lm(formula = Mean.Intensity ~ Calc.Structure.weight, data = all_df1.reduced)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -123.823  -37.956   -2.001   37.045  197.483
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           152.75911    5.44814  28.039  < 2e-16 ***
## Calc.Structure.weight   0.39988    0.08134   4.916 1.53e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 54.49 on 271 degrees of freedom
## Multiple R-squared:  0.08189,    Adjusted R-squared:  0.0785
## F-statistic: 24.17 on 1 and 271 DF,  p-value: 1.529e-06
```

```
car::influencePlot(model.all.R)
```




```
##       StudRes         Hat      CookD
## 30   3.715539 0.003712407 0.02456029
## 65   1.431319 0.052484166 0.05652068
## 121 3.715539 0.003712407 0.02456029
## 156 1.431319 0.052484166 0.05652068
```

```r
# ------------------------------------------------------------------------
# For Calc.Structure.weight
calc_structure_weight_mean <- mean(all_df1.reduced$Calc.Structure.weight, na.rm = TRUE)
calc_structure_weight_range <- seq(from = min(all_df1.reduced$Calc.Structure.weight),
                                   to = max(all_df1.reduced$Calc.Structure.weight), length.out = 100)
vol_unit_ratio_mean <- mean(all_df1.reduced$Vol.Unit.Ratio, na.rm = TRUE)

x_new <- data.frame(Calc.Structure.weight = calc_structure_weight_range,
                    Vol.Unit.Ratio = rep(vol_unit_ratio_mean, 100))

conf.int.1 <- predict(model.all.reduced, newdata = x_new, interval = "confidence")

plot(x_new$Calc.Structure.weight, conf.int.1[, "fit"], type = "l", col = "blue",
     ylim = range(conf.int.1[, c("lwr", "upr")]),
     ylab = "Mean Intensity", xlab = "Calc Structure Weight")
lines(x_new$Calc.Structure.weight, conf.int.1[, "lwr"], col = "red", lty = 2)
lines(x_new$Calc.Structure.weight, conf.int.1[, "upr"], col = "red", lty = 2)
```




```
# ------------------------------------------------------------------------
confint(model.all.R)
```

```
##                                2.5 %      97.5 %
## (Intercept)              142.0330556 163.4851578
## Calc.Structure.weight      0.2397481   0.5600094
```

Hypothesis Test:

Testing whether the the reduced linear model provides a better fit. Going to use $\alpha = 0.05$ as the significance value.

1. Assumptions: $\varepsilon \overset{iid}{\sim} N(0, 1)$

Alternatives: Define Models: Y as `Mean.Intensity`. X_1 as `Calc.Structure.weight`. X_2 as `Vol.Unit.Ratio`.

$E\{Y\}_{\text{reduced.model}} = \beta_0 + \beta_1 X_1$ $E\{Y\}_{\text{full.model}} = \beta_0 + \beta_1 X_1 + \beta_2 X_2$

2. Hypotheses:

$$H_0 : \beta_2 = 0$$

$$H_1 : \beta_2 \neq 0$$

Decision Rule:

$$p_{val} \leq \alpha = 0.05 \quad \Rightarrow \quad \text{reject null hypothesis}$$

$$p_{val} > \alpha = 0.05 \quad \Rightarrow \quad \text{do not reject null hypothesis}$$

3. Test Statistic:

```
anova.table = anova(model.all, model.all.R)
anova.table
```

```
## Analysis of Variance Table
##
## Model 1: Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio
## Model 2: Mean.Intensity ~ Calc.Structure.weight
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1    270 787439
## 2    271 804736 -1    -17297 5.9309 0.01553 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
F.crit = anova.table[2,5]
```

4. P-Value:

```
alpha = 0.05
pval = anova.table[2,6]
pval < alpha
```

```
## [1] TRUE
```

5. Conclusion: Since the p_{value} is less than the specified significance value ($\alpha = 0.05$) by the specified decision rule, we have sufficient evidence to reject the null hypothesis. There is a significant effect on the calculated structure weight (in kilo Daltons) and the mean intensity (number of photons). Therefore, the calculated structure weight contributes meaningfully to the variations of the mean intensity.

Warranting further investigation, this result suggests that the *true* structure weight also has a significant effect on the observed, *experimental* intensity values.

```
# ------------------------------------------------------------------------
# TRIAL
# percentile.90 = apply(all_df2[, -c(1,2)], 1, function(x) quantile(x, 0.9, na.rm=T)
# top.10.percent.df = all_df2[ mapply(function(row, threshold) any(row > threshold), all_df2[, -c(1,2)]
# top.10.percent.df
#
#
#
# top.10.percent.logical = all_df2[,c(1,2)] > percentile.90
# row.in.top.10.percent = rowSums(top.10.percent.logical) > 0
# top.10.df = all_df2[row.in.top.10.percent, ]
#
#
#
```

```
# # ------------------------------------------------------------------------
# plot(Mean.Intensity ~ Spacegroup.ID, all_df1)
#
# plot(m.P1211)
#
# car::avPlots(m.P1211)
#
# ncol(all_df2)
#
# mean.top10 = apply(all_df2[, 3:ncol(all_df2)], MARGIN = 1, FUN = function(x) mean(sort(x, decreasing=
# mean.top10
#
# head(mean.top10)
#
# m = lm(mean.top10 ~ df1_P1211$Vol.Unit.Ratio)
#
#
# length(df1_P1211$Vol.Unit.Ratio)
# length(mean.top10)
```

```
# every crystal in df2_all corresponds to the unique intensity values. Which we want to try and model.
# # We retrieve the top 90th percentile of all of the rows (of the same crystal), and computes the mean
# mean.top10 = apply(all_df2[, 3:ncol(all_df2)], MARGIN = 1, FUN = function(x) {
#   non_missing_values = x[!is.na(x)] # Remove NAs
#   if (length(non_missing_values) > 0) {
#     mean(non_missing_values[non_missing_values >= quantile(non_missing_values, probs = 0.9, na.rm = T
#   } else {
#     NA # Handle rows with no non-missing values
#   }
# })
#
# #align the mean.top10 and the vol.unit.ratio by selecting all of the 299 rows of df1_P1211$Vol.Unit.R
# aligned_df = data.frame(mean.top10, Vol.Unit.Ratio = df1_P1211$Vol.Unit.Ratio[1:299])
# aligned_df = aligned_df[!is.na(df1_P1211$Vol.Unit.Ratio), ]
# aligned_df
#
# m.percentile.P1211 = lm(mean.top10 ~ Vol.Unit.Ratio, data = aligned_df)
# par(mfrow = c(2, 2))
# plot(m.percentile.P1211)
# par(mfrow = c(1, 1))
#
# # ------------------------------------------------------------------------
# # Scatterplot of Calc.Structure.weight vs. Mean.Intensity with regression line
# ggplot(data = aligned_df, aes(x = mean.top10, y = Vol.Unit.Ratio)) +
#   geom_point() +
#   geom_smooth(method = "lm", formula = y ~ x, col = "red") +
#   labs(x = "mean.top10", y = "vol.unit.ratio",
#        title = "Scatterplot of mean.top10 vs. vol.unit.ratio")
#
# fitted = fitted(m)
# residuals = residuals(m)
# residuals.df = data.frame(fitted, residuals)
#
```

```
# ggplot(data = residuals.df, aes(x = fitted, y = residuals)) +
#   geom_point() +
#  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
#   labs(x = "fitted", y = "residuals",
#        title = "fitted vs. residuals")
# # ------------------------------------------------------------------------
```

```
# ########## Treating outliers.
# # ------------------------------------------------------------------------
# cooks.d = cooks.distance(m.P1211)
# max(cooks.d)
# min(cooks.d)
# 3*mean(cooks.d)
# influence = cooks.d[(cooks.d > (3*mean(cooks.d, na.rm = T)))]
# influence # see that most of the rows stated above are in this list
# row.influence = names(influence)
# outliers = df1_P1211[row.influence,]
# df1_P1211.reduced = df1_P1211 %>% anti_join(outliers)
#
# m.P1211.R = lm(Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio, data = df1_P1211.reduced)
# vif(m.P1211.R) # even lower but barely
# # ------------------------------------------------------------------------
# # Scatterplot of Calc.Structure.weight vs. Mean.Intensity with regression line
# a = ggplot(data = df1_P1211.reduced, aes(x = Calc.Structure.weight, y = Mean.Intensity)) +
#   geom_point() +
#   geom_smooth(method = "lm", formula = y ~ x, col = "red") +
#   labs(x = "Calc.Structure.weight", y = "Mean.Intensity",
#        title = "Scatterplot of Mean.Intensity vs. Calc.Structure.weight")
#
# # Scatterplot of Vol.Unit.Ratio vs. Mean.Intensity with regression line
# b = ggplot(data = df1_P1211.reduced, aes(x = Vol.Unit.Ratio, y = Mean.Intensity)) +
#   geom_point() +
#   geom_smooth(method = "lm", formula = y ~ x, col = "red") +
#   labs(x = "Vol.Unit.Ratio", y = "Mean.Intensity",
#        title = "Scatterplot of Mean.Intensity vs. Vol.Unit.Ratio")
# ggarrange(a,b, nrow =2)
# # ------------------------------------------------------------------------
# car::influencePlot(m.P1211.R,id=list(labels=row.names(df1_P1211.reduced))) # did not help much
# plot(all_df1$Spacegroup.ID, all_df1$Mean.Intensity, data=all_df1) # does not look like significant di
#
```

```
# cooks.d = cooks.distance(m.P1211.R)
# max(cooks.d)
# min(cooks.d)
# 3*mean(cooks.d)
# influence = cooks.d[(cooks.d > (3*mean(cooks.d, na.rm = T)))]
# influence # see that most of the rows stated above are in this list
# row.influence = names(influence)
```

```
# outliers = df1_P1211.reduced[row.influence,]
# df1_P1211.reduced = df1_P1211.reduced %>% anti_join(outliers)
#
# m.P1211.R = lm(Mean.Intensity ~ Calc.Structure.weight + Vol.Unit.Ratio, data = df1_P1211.reduced)
#
# # ------------------------------------------------------------------------
#
# cat("Adjusted R-squared of reduced model: ", summary(m.P1211.R)$adj.r.squared, "\n")
#
# # ------------------------------------------------------------------------
# ggplot(data = df1_P1211.reduced, aes(x = Calc.Structure.weight, y = Mean.Intensity)) +
#   geom_point() +
#   geom_smooth(method = "lm", formula = y ~ x, col = "red") +
#   labs(x = "Calc.Structure.weight", y = "Mean.Intensity",
#        title = "Scatterplot of Mean.Intensity vs. Calc.Structure.weight")
#
# # Scatterplot of Vol.Unit.Ratio vs. Mean.Intensity with regression line
# ggplot(data = df1_P1211.reduced, aes(x = Vol.Unit.Ratio, y = Mean.Intensity)) +
#   geom_point() +
#   geom_smooth(method = "lm", formula = y ~ x, col = "red") +
#   labs(x = "Vol.Unit.Ratio", y = "Mean.Intensity",
#        title = "Scatterplot of Mean.Intensity vs. Vol.Unit.Ratio")
#
# car::influencePlot(m.P1211.R,id=list(labels=row.names(df1_P1211.reduced)))


# ------------------------------------------------------------------------
```

Advanced Diagnostics

Assumption 5: Cook's Distance:

Cook's Distance is a measure to estimate the influence of all the data points when using ordinary least squares in regression. This helps point out outliers with a strong influence on the data. Since the assumption 5 check, used the `cooks.distance()` function, and gave an autocorrelation of 0.05725871, this means that there's a low correlation between residuals. Thus, the residuals are independent.

D-W Test Statistic: This test statistic tests that the null hypothesis has no autocorrelation among the residuals. With a range of 0 to 4 for this particular test statistic shows a value very close to 2. When 4 means that there is positive autocorrelation, and 0 indicates that there's negative autocorrelation. Thus, there's little to no autocorrelation among the residuals in this model.

P-Value: Under the null hypothesis, the p-value states that there is no autocorrelation. With a p-value typically above 0.05, we fail to reject the null hypothesis. This does not give enough evidence to reject the null hypothesis. With a p-value of 0.542, this model indicates that there is not enough statistical evidence to conclude that the residuals are correlated.

Influence Plot:

Assumption 6: Multicolinearity

With a VIF score of around 1, this suggests that there is very low multicolinearity in the model. We can proceed with this model that there are low correlation between the multiple independent variables in the model. Since these show VIF values of roughly around 1, these are perfectly colinear.

```
#
# # trying
#
# ########## CURVILINEAR MODELS
# # ------------------------------------------------------------------------
# ggpairs(numeric_df1_P1211, cardinality_threshold = 100)
#
# # centering
# numeric_df1_P1211$Vol.Unit.Ratio.c = numeric_df1_P1211$Vol.Unit.Ratio - mean(numeric_df1_P1211$Vol.Un
# numeric_df1_P1211$Vol.Unit.Ratio.c.2 = numeric_df1_P1211$Vol.Unit.Ratio.c^2
#
# m.poly.P1211 = lm(Mean.Intensity ~ Vol.Unit.Ratio.c + Vol.Unit.Ratio.c.2, data = numeric_df1_P1211)
# par(mfrow=c(2,2))
# plot(m.poly.P1211)
# par(mfrow=c(1,1))
#
# car::vif(m.poly.P1211)
#
# cat("Adjusted R-squared:", summary(m.poly.P1211)$adj.r.squared, "\n") #!?
#
# ggplot(data = numeric_df1_P1211, aes(x = Vol.Unit.Ratio.c, y = Mean.Intensity)) +
#   geom_point() +  # Add scatterplot points
#   geom_smooth(method = "lm", formula = y ~ poly(x, 2), col = "red") +  # Add polynomial regression fi
#   labs(x = "Centered Vol.Unit.Ratio", y = "Mean.Intensity",
#        title = "Polynomial Regression Fit of Mean.Intensity vs. Centered Vol.Unit.Ratio")
#
# fitted_values = fitted(m.poly.P1211)
# residuals = resid(m.poly.P1211)
# residuals_df = data.frame(Fitted = fitted_values, Residuals = residuals)
#
# ggplot(data = residuals_df, aes(x = Fitted, y = Residuals)) +
#   geom_point() +  # Add scatterplot points
#   geom_hline(yintercept = 0, col = "red") +  # Add horizontal reference line at y = 0
#   labs(x = "Fitted Values", y = "Residuals",
#        title = "Residuals vs. Fitted Values for Polynomial Model")
#
# # ------------------------------------------------------------------------
# ######## TAKE CARE OF LEVERAGE POINTS
# car::influencePlot(m.poly.P1211)
# # high leverage point at row, 53
# # large influence at rows, 5, 59
#
# cooks.d = cooks.distance(m.poly.P1211)
# # Plot Cook's Distance
# plot(cooks.d, pch = 20, main = "Cook's Distance Plot", xlab = "Observation Index", ylab = "Cook's Dis
# abline(h = 4 / length(cooks.d), col = "red")  #threshold line
#
# # Identify influential points
```

```
# influence = which(cooks.d > 4 / length(cooks.d))
# cat("Influential point(s) based on Cook's distance:", influence, "\n")
# row.influence = names(influence)
# outliers = numeric_df1_P1211[row.influence,]
# numeric_df1_P1211_R = numeric_df1_P1211 %>% anti_join(outliers)
#
# # ------------------------------------------------------------------------
# ########## Try again without outliers
# numeric_df1_P1211_R$Vol.Unit.Ratio.c = numeric_df1_P1211_R$Vol.Unit.Ratio - mean(numeric_df1_P1211_R$
# numeric_df1_P1211_R$Vol.Unit.Ratio.c.2 = numeric_df1_P1211_R$Vol.Unit.Ratio.c^2
#
# m.poly.P1211 = lm(Mean.Intensity ~ Vol.Unit.Ratio.c + Vol.Unit.Ratio.c.2, data = numeric_df1_P1211_R)
# par(mfrow=c(2,2))
# plot(m.poly.P1211)
# par(mfrow=c(1,1))
#
# car::vif(m.poly.P1211)
#
# cat("Adjusted R-squared:", summary(m.poly.P1211)$adj.r.squared, "\n") #!?
#
# ggplot(data = numeric_df1_P1211_R, aes(x = Vol.Unit.Ratio.c, y = Mean.Intensity)) +
#   geom_point() +  # Add scatterplot points
#   geom_smooth(method = "lm", formula = y ~ poly(x, 2), col = "red") +  # Add polynomial regression fi
#   labs(x = "Centered Vol.Unit.Ratio", y = "Mean.Intensity",
#        title = "Polynomial Regression Fit of Mean.Intensity vs. Centered Vol.Unit.Ratio")
#
# fitted_values = fitted(m.poly.P1211)
# residuals = resid(m.poly.P1211)
# residuals_df = data.frame(Fitted = fitted_values, Residuals = residuals)
#
# ggplot(data = residuals_df, aes(x = Fitted, y = Residuals)) +
#   geom_point() +  # Add scatterplot points
#   geom_hline(yintercept = 0, col = "red") +  # Add horizontal reference line at y = 0
#   labs(x = "Fitted Values", y = "Residuals",
#        title = "Residuals vs. Fitted Values for Polynomial Model")
```

"Phase Problem." n.d. Accessed November 10, 2023. https://en.wikipedia.org/wiki/Phase_problem.