Project2

Group-11

```r
library(tidyverse)
library(moderndive)
library(gapminder)
library(sjPlot)
library(jtools)
library(GGally)
library(gt)
library(gridExtra)
library(knitr)
library(patchwork)
library(broom)
```

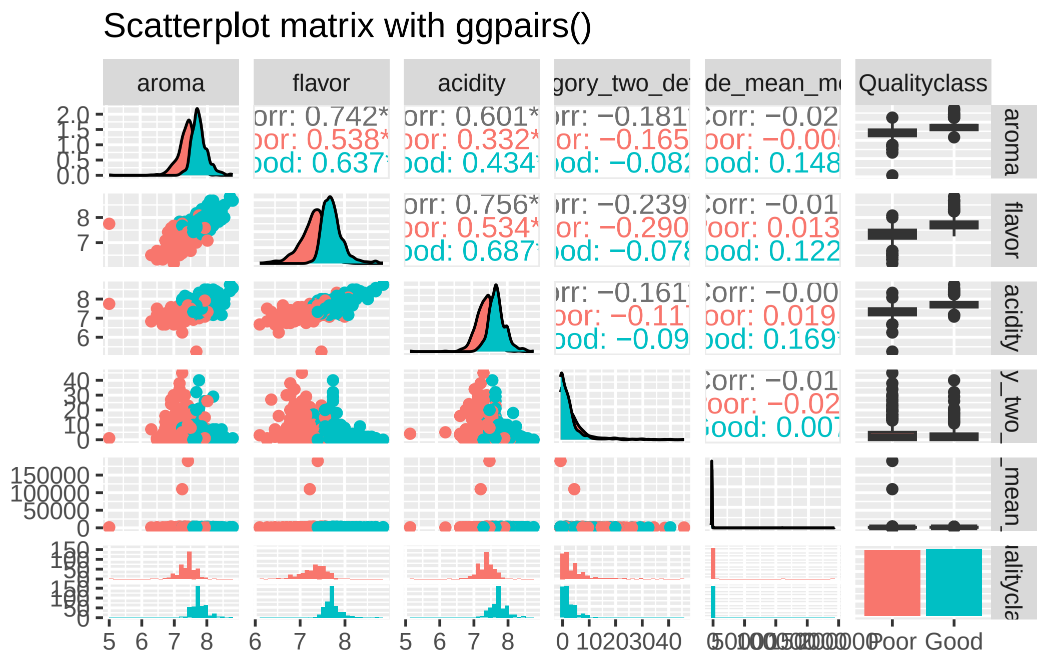
1 Data Visulization

Preprocessing the data and conducting summary statistics, while also generating visualizations to better understand the data.

```r
# Read the dataset
data <- read.csv("dataset11.csv")
data <- na.omit(data)
data$Qualityclass <- factor(data$Qualityclass, levels = c("Poor", "Good"))
str(data)
```

```
'data.frame':   894 obs. of  8 variables:
 $ country_of_origin   : chr  "Guatemala" "Taiwan" "Brazil" "Colombia" ...
 $ aroma               : num  7.58 7.92 7.83 7.5 7.33 7.83 7.83 7.5 7.08 7.08 ...
 $ flavor              : num  7.5 7.75 7.83 7.75 7.17 8.33 7.75 7.58 6.92 6.92 ...
 $ acidity             : num  7.58 7.67 7.67 7.75 7.08 8.25 7.58 7.75 7 7.25 ...
 $ category_two_defects: int  1 0 3 0 1 9 2 2 15 2 ...
$ altitude_mean_meters: num  1901 1000 1250 1450 1450 ...
$ harvested           : int  2017 2016 2012 2011 2013 2012 2015 2014 2013 2015 ...
$ Qualityclass        : Factor w/ 2 levels "Poor","Good": 1 2 2 2 1 2 2 2 1 1 ...
- attr(*, "na.action")= 'omit' Named int [1:200] 5 13 16 23 27 28 33 42 45 46 ...
 ..- attr(*, "names")= chr [1:200] "5" "13" "16" "23" ...
```

```
# Scatterplot matrix with ggpairs()
scatterplot = data %>%
  select(aroma, flavor, acidity, category_two_defects, altitude_mean_meters, Qualityclass)
ggpairs(scatterplot, aes(color = Qualityclass), title="Scatterplot matrix with ggpairs()")
```



```
# Summary Statistics for 'aroma', 'flavor', and 'acidity' across different quality classes
data |>
  summarize('ar.Mean' = mean(aroma),
            'ar.Sd' = sd(aroma),
            'ar.Min' = min(aroma),
            'ar.Max' = max(aroma),
            'fl.Mean' = mean(flavor),
            'fl.Sd' = sd(flavor),
            'fl.Min' = min(flavor),
            'fl.Max' = max(flavor),
            'ac.Mean' = mean(acidity),
          'ac.Sd' = sd(acidity),
          'ac.Min' = min(acidity),
          'ac.Max' = max(acidity),
             .by = Qualityclass) |>
gt() |>
  fmt_number(decimals = 2) |>
  tab_spanner(
    label = "aroma",
    columns = c(ar.Mean, ar.Sd, ar.Min, ar.Max)
  ) |>
  tab_spanner(
    label = "flavor",
    columns = c(fl.Mean, fl.Sd, fl.Min, fl.Max)
  ) |>
  tab_spanner(
    label = "acidity",
    columns = c(ac.Mean, ac.Sd, ac.Min, ac.Max)
  )
# Summary statistics for 'category_two_defects' and 'altitude_mean_meters' across differen
data |>
  summarize('C.Mean' = mean(category_two_defects),
          'C.Sd' = sd(category_two_defects),
          'C.Min' = min(category_two_defects),
          'C.Max' = max(category_two_defects),
          'A.Mean' = mean(log(altitude_mean_meters)),
          'A.Sd' = sd(log(altitude_mean_meters)),
          'A.Min' = min(log(altitude_mean_meters)),
          'A.Max' = max(log(altitude_mean_meters)),
             .by = Qualityclass) |>
gt() |>
  fmt_number(decimals = 2) |>
  tab_spanner(
    label = "Defects",
    columns = c(C.Mean, C.Sd, C.Min, C.Max)
  ) |>
  tab_spanner(
    label = "log_Altitude",
    columns = c(A.Mean, A.Sd, A.Min, A.Max)
  )
```

Table 1: Summary statistics

(a)

	aroma				flavor				acidity		
Qualityclass	ar.Mean	ar.Sd	ar.Min	ar.Max	fl.Mean	fl.Sd	fl.Min	fl.Max	ac.Mean	ac.Sd	ac.M
Poor	7.38	0.28	5.08	8.25	7.31	0.27	6.17	8.08	7.35	0.26	5.
Good	7.76	0.23	7.17	8.75	7.74	0.23	7.25	8.83	7.72	0.24	7.

(b)

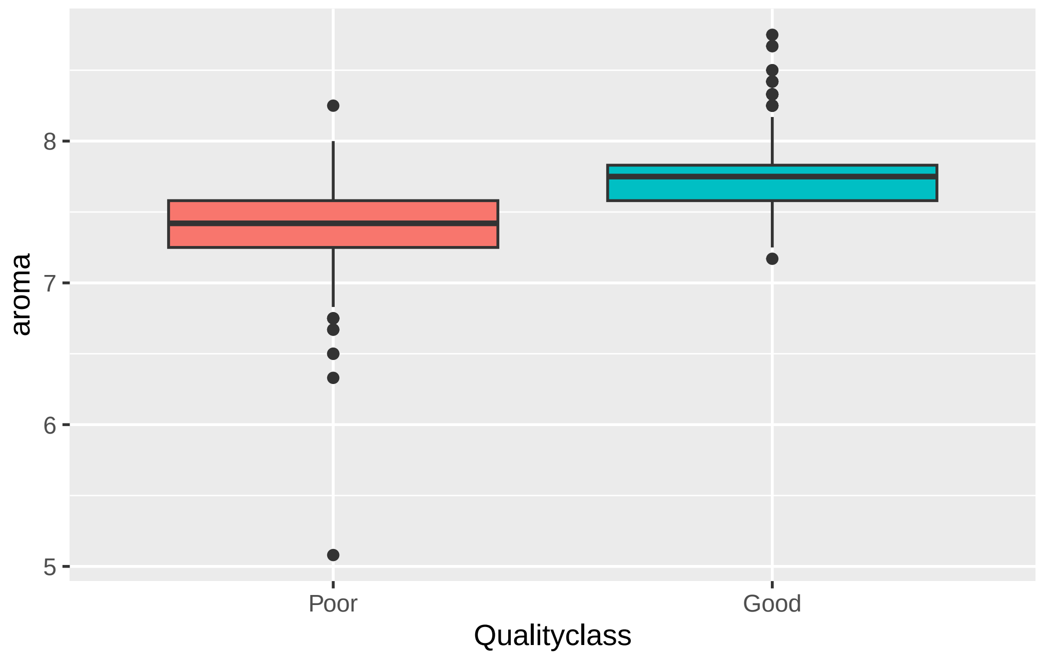
	Defects				log_Altitude			
Qualityclass	C.Mean	C.Sd	C.Min	C.Max	A.Mean	A.Sd	A.Min	A.Max
Poor	4.16	5.91	0.00	45.00	6.94	1.10	0.00	12.16
Good	2.85	4.32	0.00	40.00	7.13	0.77	0.00	8.10

2 Predictor Modeling and Evaluation

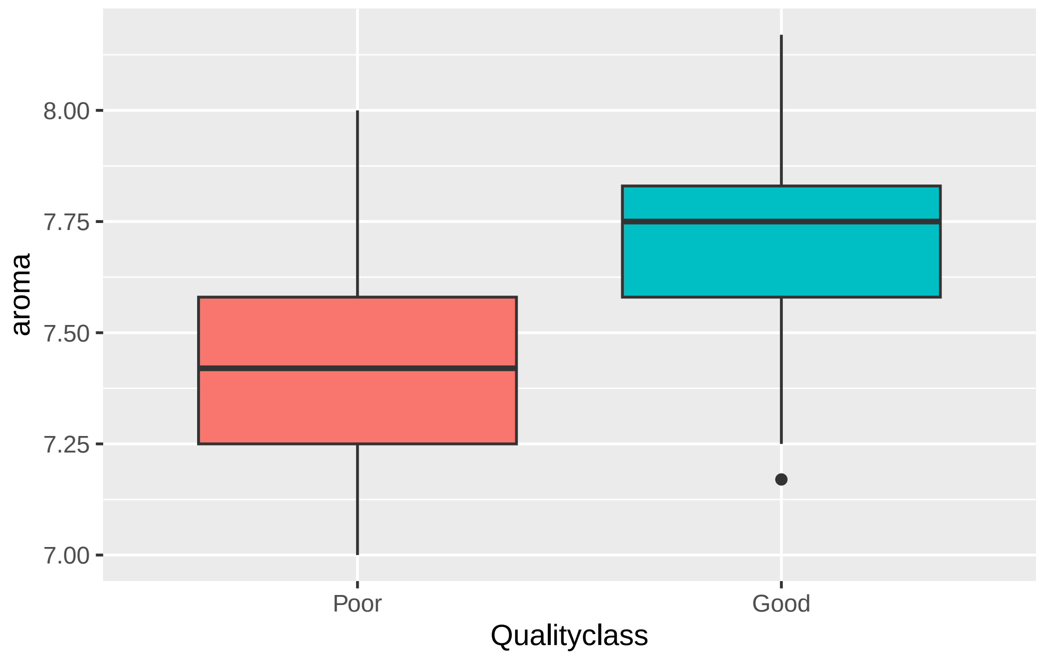
Modeling each predictor separately with the response variable to observe the individual impact of each feature on the quality of coffee.

2.1 Aroma and Qualityclass

```r
# Select 'aroma' and 'Qualityclass' columns
data_aroma <- data %>%
                  select(aroma, Qualityclass)

# Create a boxplot of 'aroma' across different 'Qualityclass' levels
p1 <- ggplot(data = data_aroma, aes(x = Qualityclass, y = aroma, fill = Qualityclass)) +
  geom_boxplot() +
  labs(x = "Qualityclass", y = "aroma")+
  theme(legend.position = "none")
p1
```



```
# Calculate quartiles and determine lower and upper bounds
q1_aroma <- quantile(data_aroma$aroma, 0.25)
q3_aroma <- quantile(data_aroma$aroma, 0.75)
iqr_aroma <- q3_aroma - q1_aroma
lower_bound_aroma <- q1_aroma - 1.5 * iqr_aroma
upper_bound_aroma <- q3_aroma + 1.5 * iqr_aroma

# Filter 'aroma' within the calculated bounds
data_aroma_filtered <- data_aroma %>%
  filter(aroma >= lower_bound_aroma & aroma <= upper_bound_aroma)

# Create a boxplot of 'aroma'
p2 <- ggplot(data = data_aroma_filtered, aes(x = Qualityclass, y = aroma, fill = Qualitycl
  geom_boxplot() +
  labs(x = "Qualityclass", y = "aroma")+
  theme(legend.position = "none")
p2
```



```r
# Fit logistic regression model with 'aroma' predictor and 'Qualityclass' response
model1 <- glm(Qualityclass ~ aroma, data = data_aroma_filtered,
              family = binomial(link = "logit"))
model1 %>%
  summary()
```

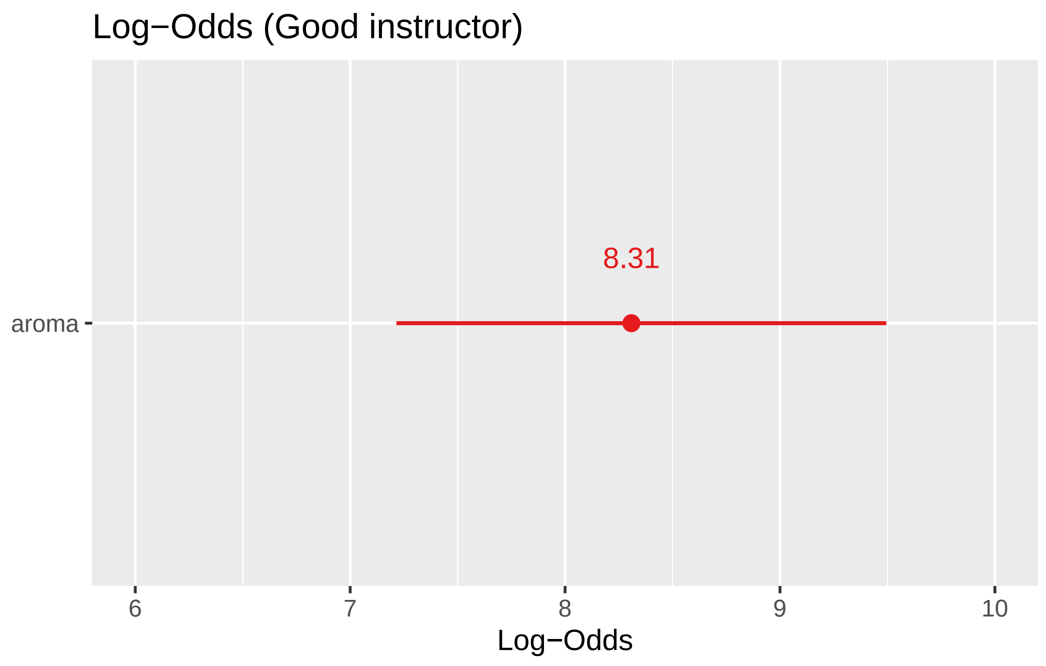
```

Call:
glm(formula = Qualityclass ~ aroma, family = binomial(link = "logit"),
    data = data_aroma_filtered)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -62.9417     4.4005  -14.30   <2e-16 ***
aroma         8.3088     0.5803   14.32   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1171  on 844  degrees of freedom
Residual deviance:  762  on 843  degrees of freedom
```

```
AIC: 766

Number of Fisher Scoring iterations: 5
```

```
# Calculate lower and upper bounds for 'aroma' log-odds
mod1.coef.logodds <- model1 %>%
                      summary() %>%
                      coef()
aroma.logodds.lower <- mod1.coef.logodds["aroma", "Estimate"] -
                      1.96 * mod1.coef.logodds["aroma", "Std. Error"]
aroma.logodds.upper <- mod1.coef.logodds["aroma", "Estimate"] +
                      1.96 * mod1.coef.logodds["aroma", "Std. Error"]

# Display the confidence interval
paste("(", aroma.logodds.lower, ",", aroma.logodds.upper, ")")
```

```
[1] "( 7.17134529409537 , 9.44618713401483 )"
```

```
# Plot log-odds of being a good instructor
plot_model(model1, show.values = TRUE, transform = NULL,
           title = "Log-Odds (Good instructor)", show.p = FALSE)
```



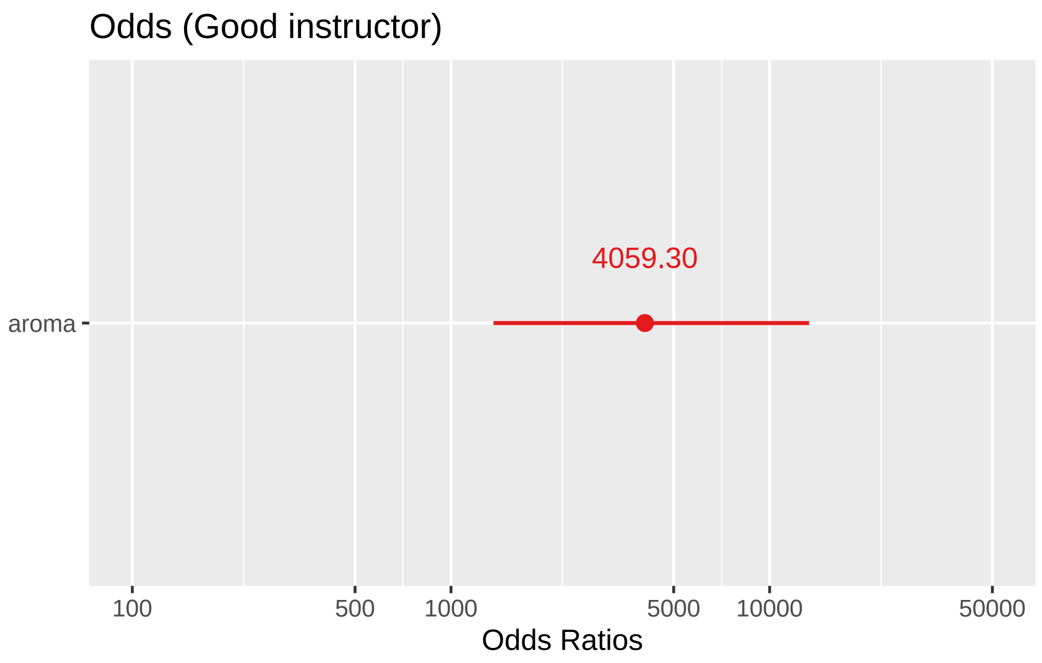
```r
# Calculate lower and upper bounds for 'aroma' odds
exp(mod1.coef.logodds)
```

```
                Estimate Std. Error      z value Pr(>|z|)
(Intercept) 4.621501e-28  81.494413 6.140521e-07        1
aroma       4.059302e+03   1.786604 1.652234e+06        1
```

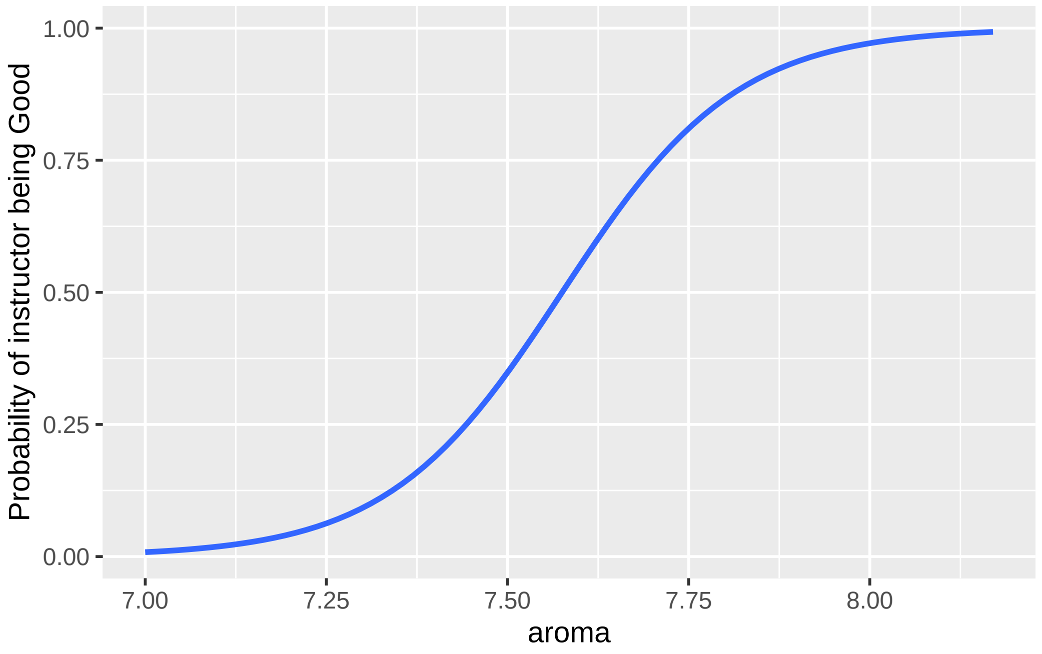
```r
aroma.odds.lower <- exp(aroma.logodds.lower)
aroma.odds.upper <- exp(aroma.logodds.upper)

# Display the confidence interval
paste("(", aroma.odds.lower, ",", aroma.odds.upper, ")")
```

```
[1] "( 1301.59445284063 , 12659.8029906219 )"
```

```r
# Plot odds of being a good instructor
plot_model(model1, show.values = TRUE,
           title = "Odds (Good instructor)", show.p = FALSE)
```



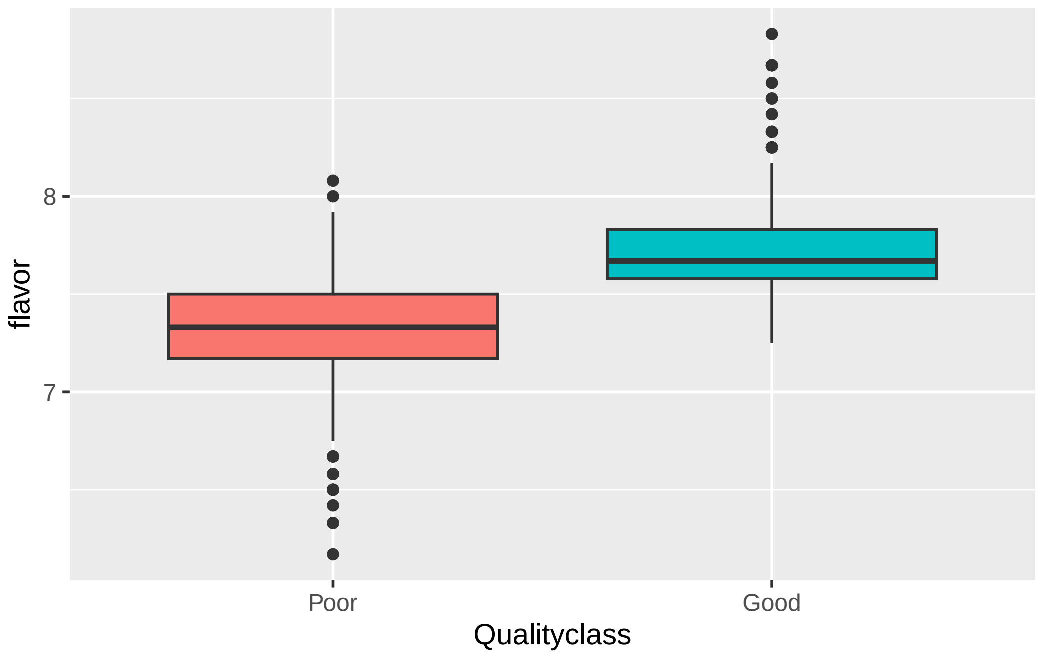
```
# Add predicted probabilities
data_aroma_filtered_after <- data_aroma_filtered %>%
                  mutate(logodds.Good = predict(model1, type = "response")) %>%
                  mutate(odds.Good = exp(logodds.Good)) %>%
                  mutate(probs.Good = fitted(model1))

# Plot the relationship between 'aroma' and probability of being a good instructor
ggplot(data = data_aroma_filtered_after, aes(x = aroma, y = probs.Good)) +
  geom_smooth(method = "glm",
              method.args = list(family = "binomial"),
              se = FALSE) +
  labs(x = "aroma", y = "Probability of instructor being Good")
```



2.2 Flavor and Qualityclass

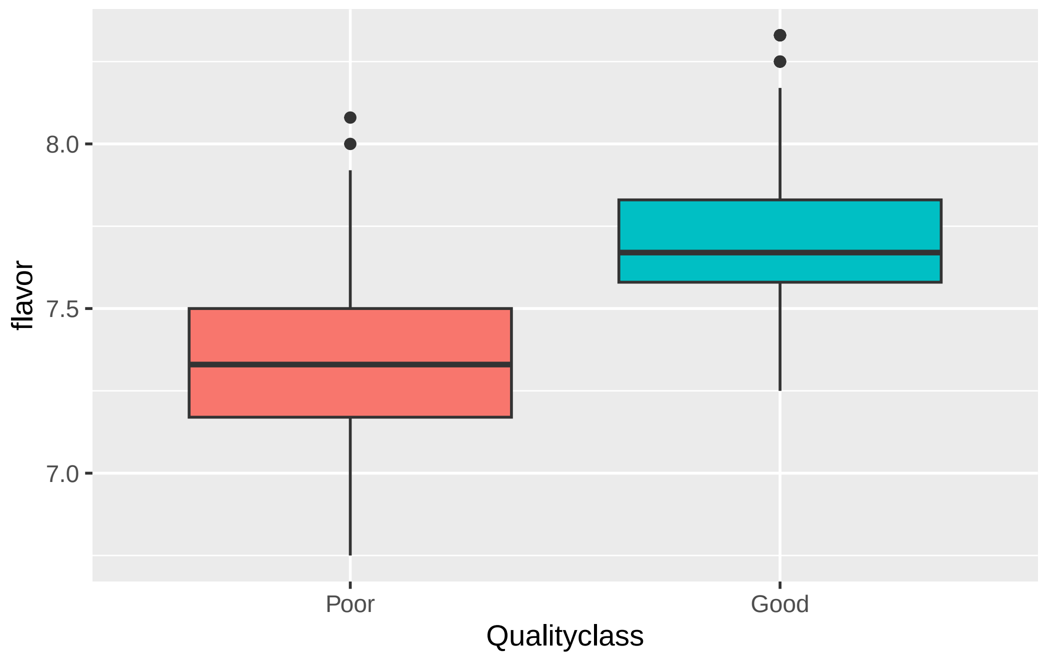
```
# Select 'flavor' and 'Qualityclass' columns
data_flavor <- data %>%
                  select(flavor, Qualityclass)

# Create a boxplot of 'flavor' across different 'Qualityclass' levels
p3 <- ggplot(data = data_flavor, aes(x = Qualityclass, y = flavor, fill = Qualityclass)) +
```

```r
  geom_boxplot() +
  labs(x = "Qualityclass", y = "flavor")+
  theme(legend.position = "none")
p3
```



```r
# Calculate quartiles and determine lower and upper bounds
q1_flavor <- quantile(data_flavor$flavor, 0.25)
q3_flavor <- quantile(data_flavor$flavor, 0.75)
iqr_flavor <- q3_flavor - q1_flavor
lower_bound_flavor <- q1_flavor - 1.5 * iqr_flavor
upper_bound_flavor <- q3_flavor + 1.5 * iqr_flavor

# Filter 'flavor' within the calculated bounds
data_flavor_filtered <- data_flavor %>%
  filter(flavor >= lower_bound_flavor & flavor <= upper_bound_flavor)

# Create a boxplot of 'flavor'
p4 <- ggplot(data = data_flavor_filtered, aes(x = Qualityclass, y = flavor, fill = Quality
  geom_boxplot() +
  labs(x = "Qualityclass", y = "flavor")+
  theme(legend.position = "none")
```

```r
p4
```



```r
# Fit logistic regression model with 'flavor' predictor and 'Qualityclass' response
model2 <- glm(Qualityclass ~ flavor, data = data_flavor_filtered,
              family = binomial(link = "logit"))
model2 %>%
  summary()
```

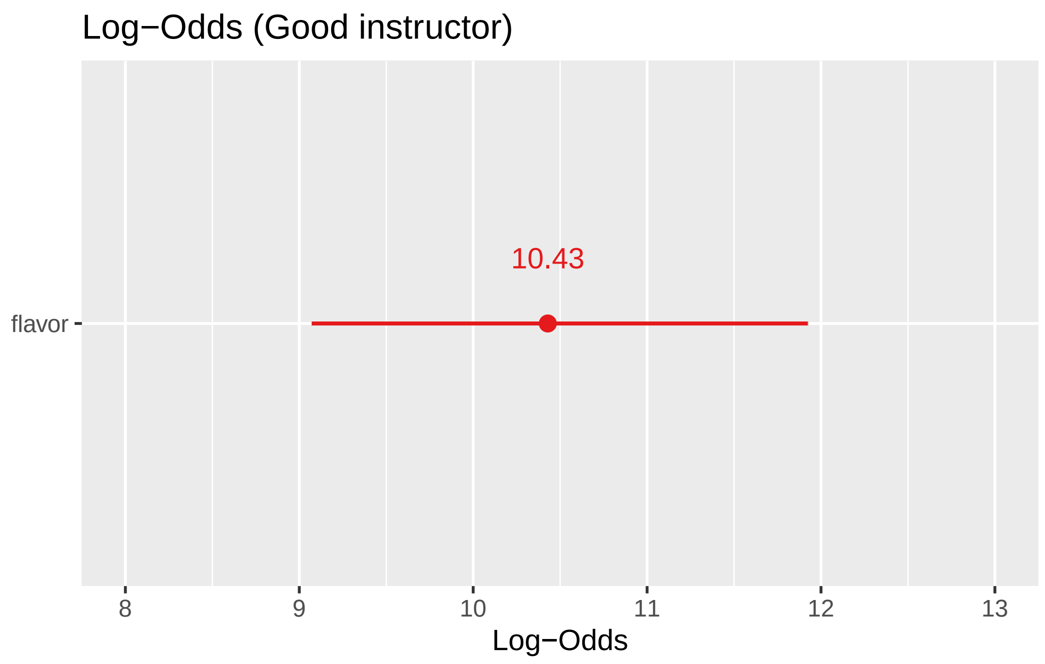
```

Call:
glm(formula = Qualityclass ~ flavor, family = binomial(link = "logit"),
    data = data_flavor_filtered)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -78.5900     5.4808  -14.34   <2e-16 ***
flavor       10.4290     0.7264   14.36   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 1207.32  on 870  degrees of freedom
Residual deviance:  648.91  on 869  degrees of freedom
AIC: 652.91

Number of Fisher Scoring iterations: 6
```

```
# Calculate lower and upper bounds for 'flavor' log-odds
mod2.coef.logodds <- model2 %>%
                      summary() %>%
                      coef()
flavor.logodds.lower <- mod2.coef.logodds["flavor", "Estimate"] -
                      1.96 * mod2.coef.logodds["flavor", "Std. Error"]
flavor.logodds.upper <- mod2.coef.logodds["flavor", "Estimate"] +
                      1.96 * mod2.coef.logodds["flavor", "Std. Error"]

# Display the confidence interval
paste("(", flavor.logodds.lower, ",", flavor.logodds.upper, ")")
```

```
[1] "( 9.00535216215199 , 11.8527408569355 )"
```

```
# Plot log-odds of being a good instructor
plot_model(model2, show.values = TRUE, transform = NULL,
           title = "Log-Odds (Good instructor)", show.p = FALSE)
```



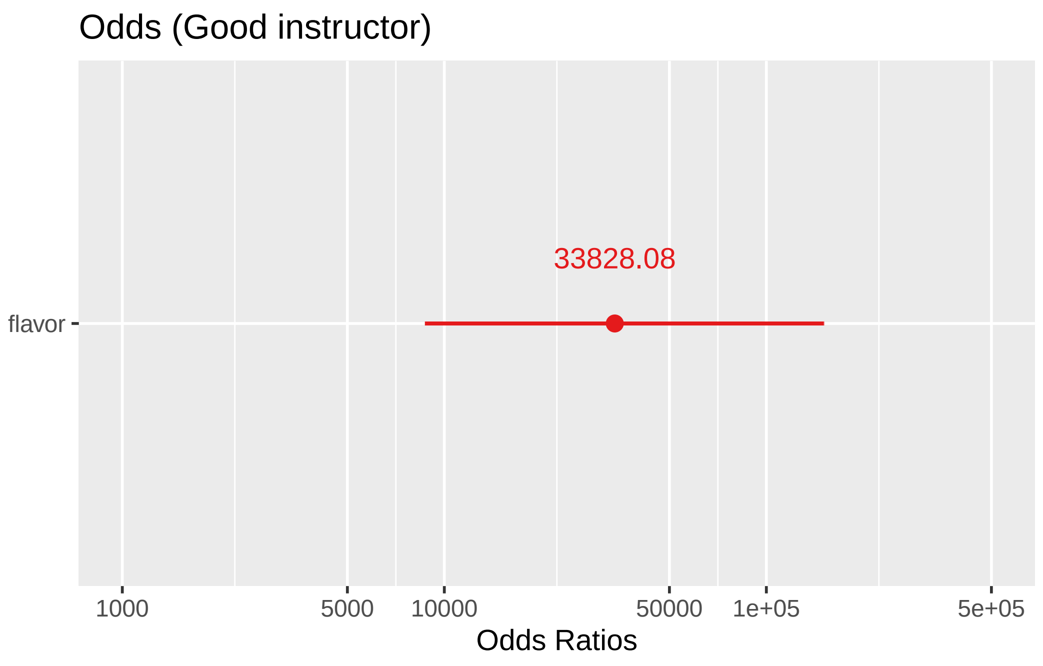
```r
# Calculate lower and upper bounds for 'flavor' odds
exp(mod2.coef.logodds)
```

```
                Estimate Std. Error      z value Pr(>|z|)
(Intercept) 7.392437e-35 240.035635 5.923404e-07        1
flavor      3.382808e+04   2.067571 1.719714e+06        1
```

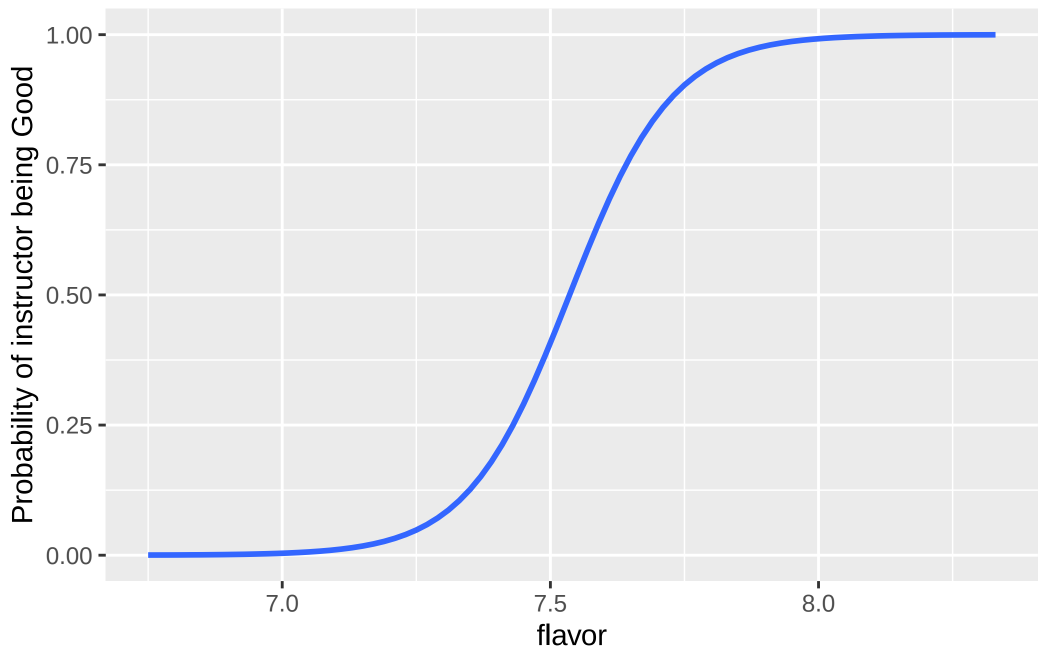
```r
flavor.odds.lower <- exp(flavor.logodds.lower)
flavor.odds.upper <- exp(flavor.logodds.upper)

# Display the confidence interval
paste("(", flavor.odds.lower, ",", flavor.odds.upper, ")")
```

```
[1] "( 8146.56921303119 , 140468.824988887 )"
```

```r
# Plot odds of being a good instructor
plot_model(model2, show.values = TRUE,
           title = "Odds (Good instructor)", show.p = FALSE)
```

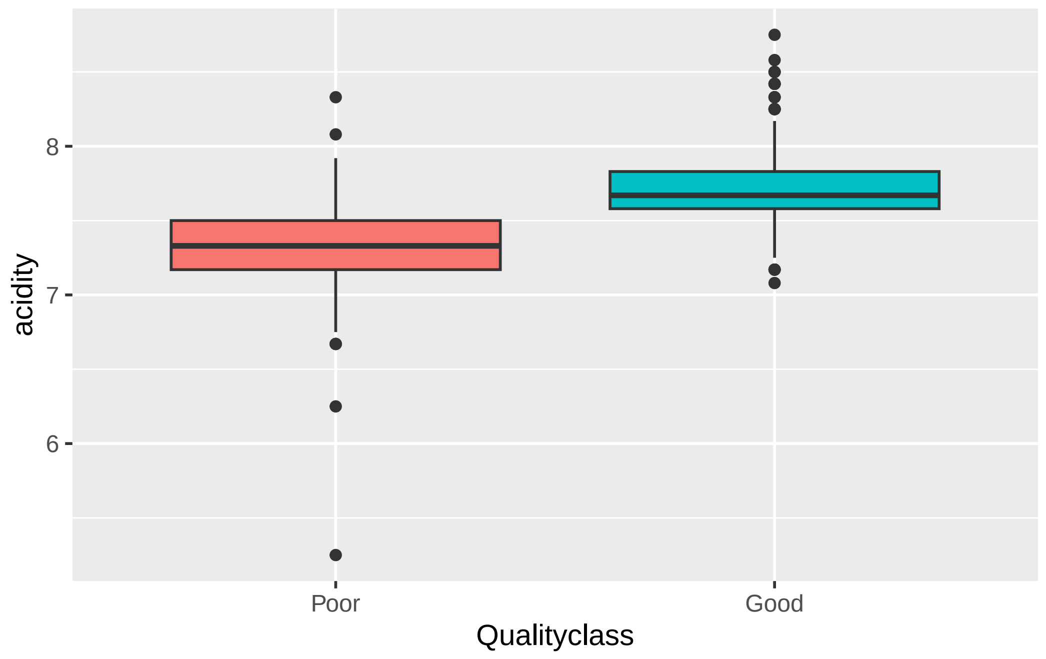


```r
# Add predicted probabilities
data_flavor_filtered_after <- data_flavor_filtered %>%
                  mutate(logodds.Good = predict(model2, type = "response")) %>%
                  mutate(odds.Good = exp(logodds.Good)) %>%
                  mutate(probs.Good = fitted(model2))

# Plot the relationship between 'flavor' and probability of being a good instructor
ggplot(data = data_flavor_filtered_after, aes(x = flavor, y = probs.Good)) +
  geom_smooth(method="glm",
              method.args = list(family="binomial"),
              se = FALSE) +
  labs(x = "flavor", y = "Probability of instructor being Good")
```

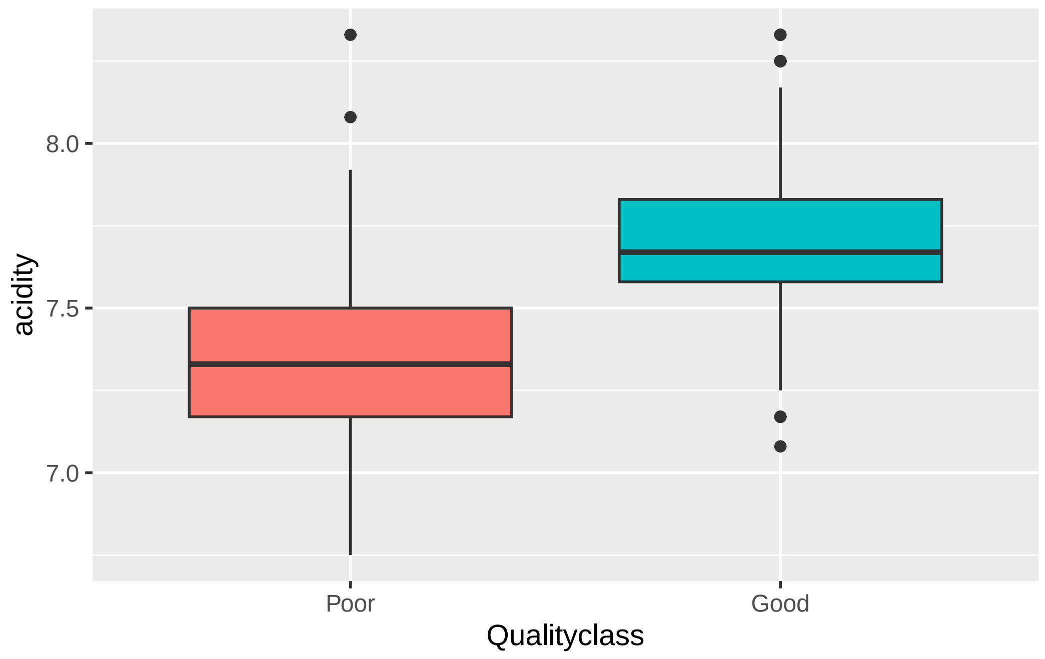


2.3 Acidity and Qualityclass

```r
# Select 'acidity' and 'Qualityclass' columns
data_acidity <- data %>%
                  select(acidity, Qualityclass)

# Create a boxplot of 'acidity' across different 'Qualityclass' levels
p5 <- ggplot(data = data_acidity, aes(x = Qualityclass, y = acidity, fill = Qualityclass))
  geom_boxplot() +
  labs(x = "Qualityclass", y = "acidity")+
  theme(legend.position = "none")
p5
```



```
# Calculate quartiles and determine lower and upper bounds
q1_acidity <- quantile(data_acidity$acidity, 0.25)
q3_acidity <- quantile(data_acidity$acidity, 0.75)
iqr_acidity <- q3_acidity - q1_acidity
lower_bound_acidity <- q1_acidity - 1.5 * iqr_acidity
upper_bound_acidity <- q3_acidity + 1.5 * iqr_acidity

# Filter 'acidity' within the calculated bounds
data_acidity_filtered <- data_acidity %>%
  filter(acidity >= lower_bound_acidity & acidity <= upper_bound_acidity)

# Create a boxplot of 'acidity'
p6 <- ggplot(data = data_acidity_filtered, aes(x = Qualityclass, y = acidity, fill = Quali
  geom_boxplot() +
  labs(x = "Qualityclass", y = "acidity")+
  theme(legend.position = "none")
p6
```



```r
# Fit logistic regression model with 'acidity' predictor and 'Qualityclass' response
model3 <- glm(Qualityclass ~ acidity, data = data_acidity_filtered,
             family = binomial(link = "logit"))
model3 %>%
  summary()
```

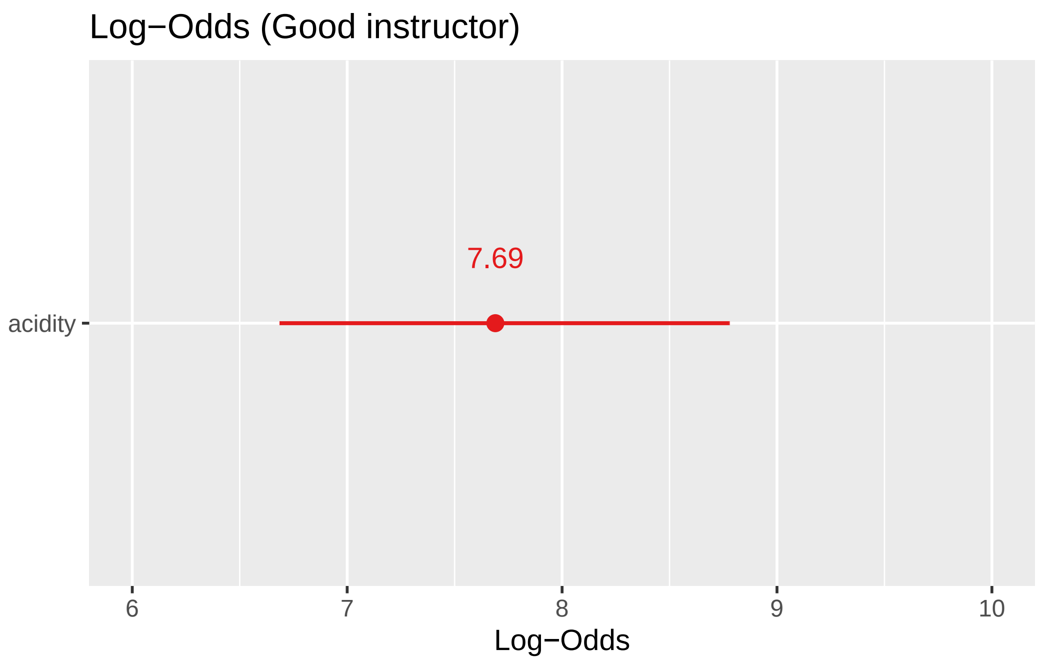
```
Call:
glm(formula = Qualityclass ~ acidity, family = binomial(link = "logit"),
    data = data_acidity_filtered)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -57.9204     4.0197  -14.41   <2e-16 ***
acidity       7.6895     0.5333   14.42   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1219.92  on 879  degrees of freedom
Residual deviance:  790.64  on 878  degrees of freedom
```

```
AIC: 794.64

Number of Fisher Scoring iterations: 5
```

```
# Calculate lower and upper bounds for 'acidity' log-odds
mod3.coef.logodds <- model3 %>%
                      summary() %>%
                      coef()
acidity.logodds.lower <- mod3.coef.logodds["acidity", "Estimate"] -
                      1.96 * mod3.coef.logodds["acidity", "Std. Error"]
acidity.logodds.upper <- mod3.coef.logodds["acidity", "Estimate"] +
                      1.96 * mod3.coef.logodds["acidity", "Std. Error"]

# Display the confidence interval
paste("(", acidity.logodds.lower, ",", acidity.logodds.upper, ")")
```

```
[1] "( 6.64420994797929 , 8.73471490913007 )"
```

```
# Plot log-odds of being a good instructor
plot_model(model3, show.values = TRUE, transform = NULL,
           title = "Log-Odds (Good instructor)", show.p = FALSE)
```



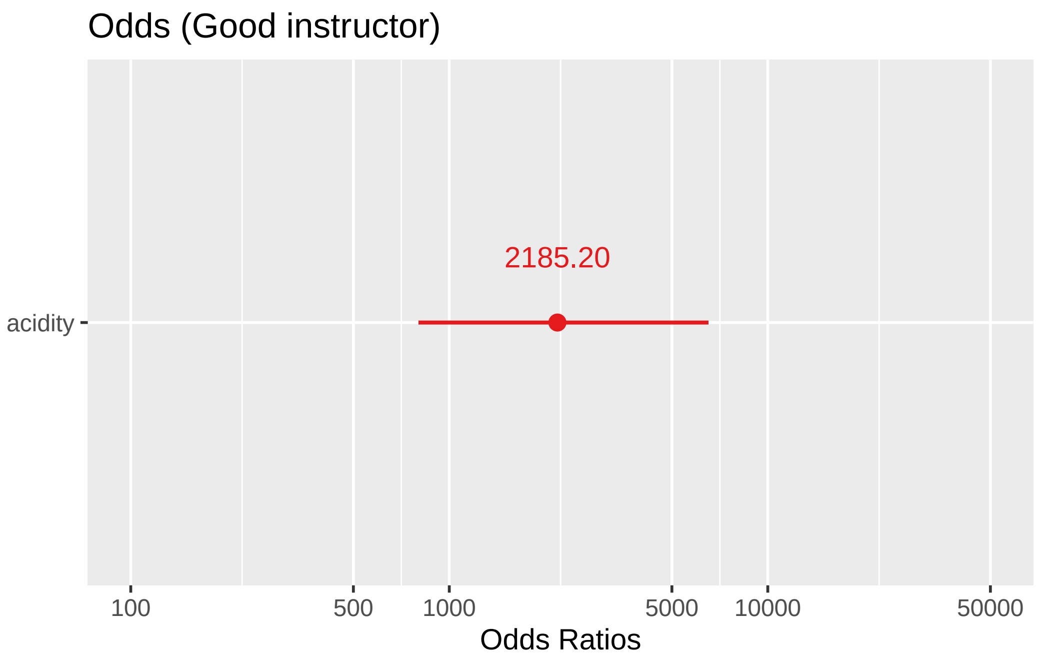
```r
# Calculate lower and upper bounds for 'acidity' odds
exp(mod3.coef.logodds)
```

```
                Estimate Std. Error      z value Pr(>|z|)
(Intercept) 7.00599e-26  55.681932 5.522277e-07        1
acidity     2.18520e+03   1.704535 1.828227e+06        1
```

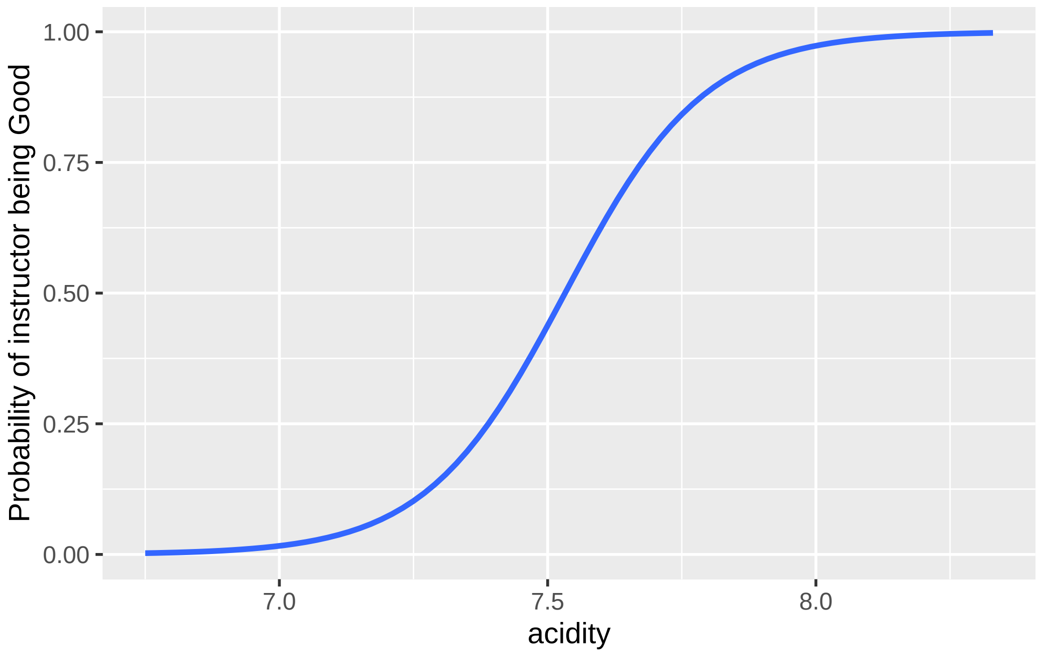
```r
acidity.odds.lower <- exp(acidity.logodds.lower)
acidity.odds.upper <- exp(acidity.logodds.upper)

# Display the confidence interval
paste("(", acidity.odds.lower, ",", acidity.odds.upper, ")")
```

```
[1] "( 768.322792807092 , 6214.96212084181 )"
```

```r
# Plot odds of being a good instructor
plot_model(model3, show.values = TRUE,
           title = "Odds (Good instructor)", show.p = FALSE)
```

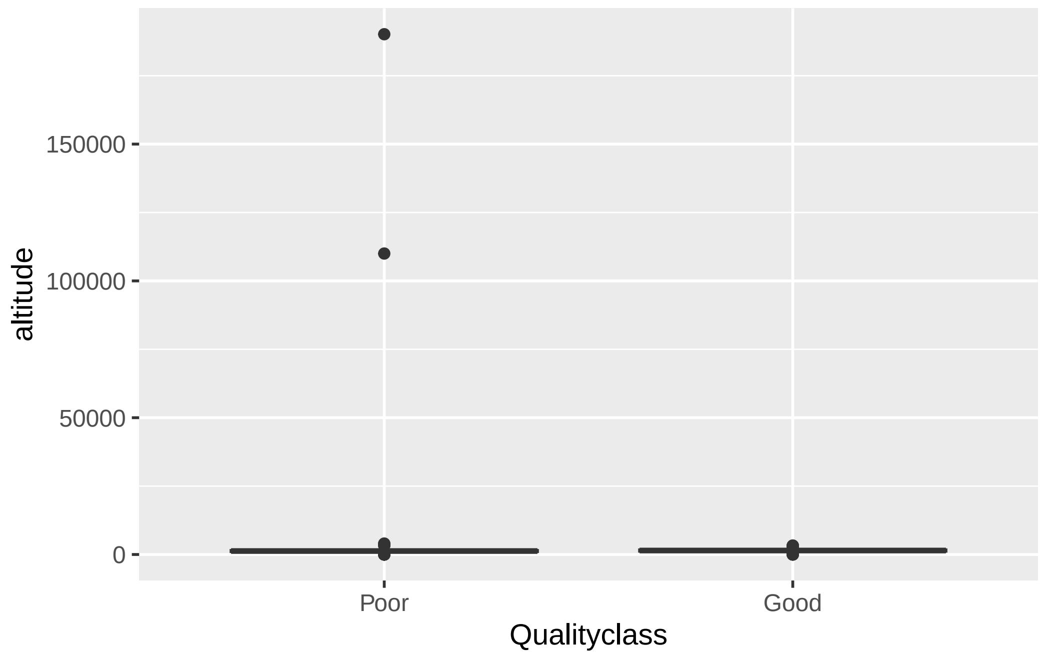


```r
# Add predicted probabilities
data_acidity_filtered_after <- data_acidity_filtered %>%
                  mutate(logodds.Good = predict(model3, type = "response")) %>%
                  mutate(odds.Good = exp(logodds.Good)) %>%
                  mutate(probs.Good = fitted(model3))

# Plot the relationship between 'acidity' and probability of being a good instructor
ggplot(data = data_acidity_filtered_after, aes(x = acidity, y = probs.Good)) +
  geom_smooth(method ="glm",
              method.args = list(family = "binomial"),
              se = FALSE) +
  labs(x = "acidity", y = "Probability of instructor being Good")
```



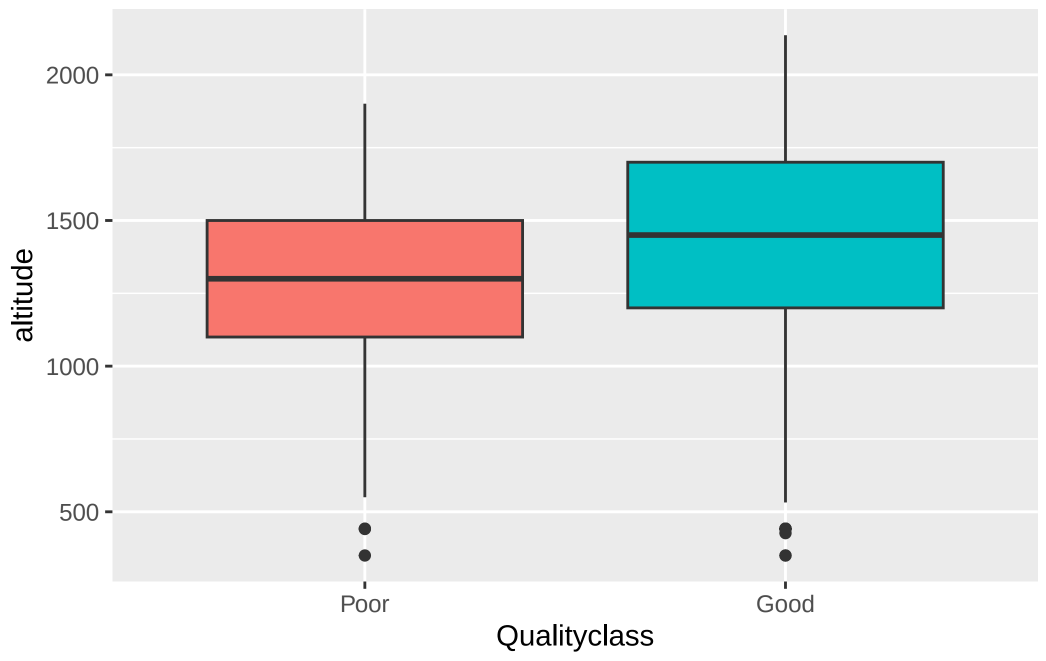
2.4 Altitude mean meters and Qualityclass

```r
# Select 'altitude_mean_meters' and 'Qualityclass' columns
data_altitude <- data %>%
                  select(altitude_mean_meters, Qualityclass)

# Create a boxplot of 'altitude_mean_meters' across different 'Qualityclass' levels
p7 <- ggplot(data = data_altitude, aes(x = Qualityclass, y = altitude_mean_meters, fill =
  geom_boxplot() +
  labs(x = "Qualityclass", y = "altitude")+
  theme(legend.position = "none")
p7
```



```
# Calculate quartiles and determine lower and upper bounds
q1_altitude <- quantile(data_altitude$altitude_mean_meters, 0.25)
q3_altitude <- quantile(data_altitude$altitude_mean_meters, 0.75)
iqr_altitude <- q3_altitude - q1_altitude
lower_bound_altitude <- q1_altitude - 1.5 * iqr_altitude
upper_bound_altitude <- q3_altitude + 1.5 * iqr_altitude

# Filter 'altitude_mean_meters' within the calculated bounds
data_altitude_filtered <- data_altitude %>%
  filter(altitude_mean_meters >= lower_bound_altitude & altitude_mean_meters <= upper_boun

# Create a boxplot of 'altitude_mean_meters'
p8 <- ggplot(data = data_altitude_filtered, aes(x = Qualityclass, y = altitude_mean_meters
  geom_boxplot() +
  labs(x = "Qualityclass", y = "altitude")+
  theme(legend.position = "none")
```

```
p8
```



```
# Fit logistic regression model with 'altitude_mean_meters' predictor and 'Qualityclass' r
model4 <- glm(Qualityclass ~ altitude_mean_meters, data = data_altitude_filtered,
             family = binomial(link = "logit"))
model4 %>%
  summary()
```

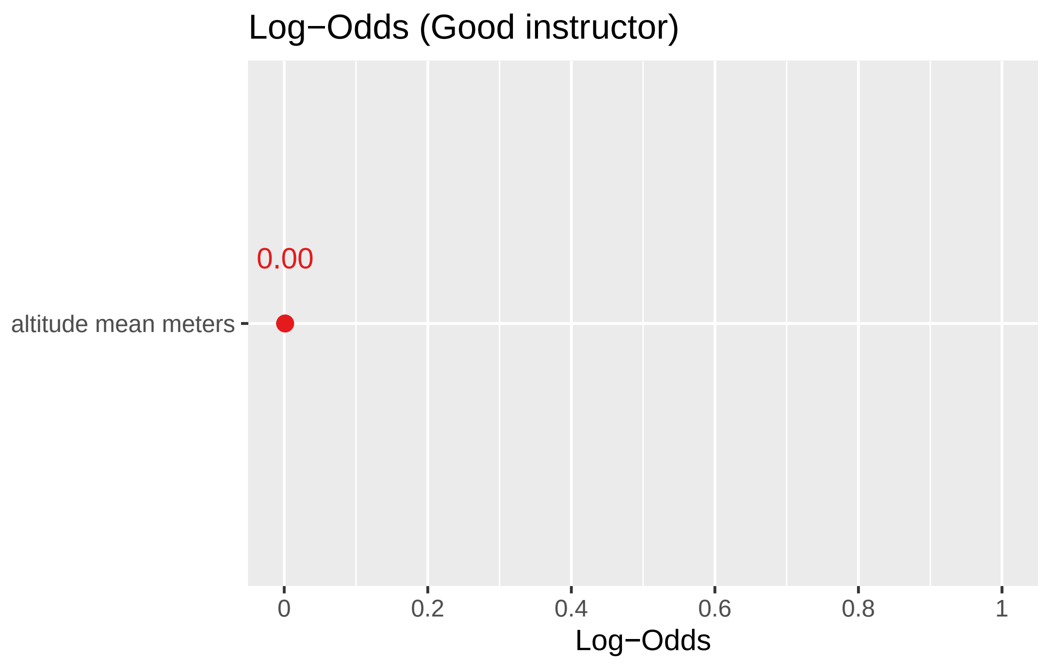
```
Call:
glm(formula = Qualityclass ~ altitude_mean_meters, family = binomial(link = "logit"),
    data = data_altitude_filtered)

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)    
(Intercept)          -1.6483638  0.3003784  -5.488 4.07e-08 ***
altitude_mean_meters  0.0012630  0.0002167   5.829 5.58e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 1169.5  on 843  degrees of freedom
Residual deviance: 1133.4  on 842  degrees of freedom
AIC: 1137.4

Number of Fisher Scoring iterations: 4
```

```
# Calculate lower and upper bounds for 'altitude_mean_meters' log-odds
mod4.coef.logodds <- model4 %>%
                      summary() %>%
                      coef()
altitude.logodds.lower <- mod4.coef.logodds["altitude_mean_meters", "Estimate"] -
                      1.96 * mod4.coef.logodds["altitude_mean_meters", "Std. Error"]
altitude.logodds.upper <- mod4.coef.logodds["altitude_mean_meters", "Estimate"] +
                      1.96 * mod4.coef.logodds["altitude_mean_meters", "Std. Error"]

# Display the confidence interval
paste("(", altitude.logodds.lower, ",", altitude.logodds.upper, ")")
```

```
[1] "( 0.000838328996571 46 , 0.00168775443306841 )"
```

```
# Plot log-odds of being a good instructor
plot_model(model4, show.values = TRUE, transform = NULL,
           title = "Log-Odds (Good instructor)", show.p = FALSE)
```



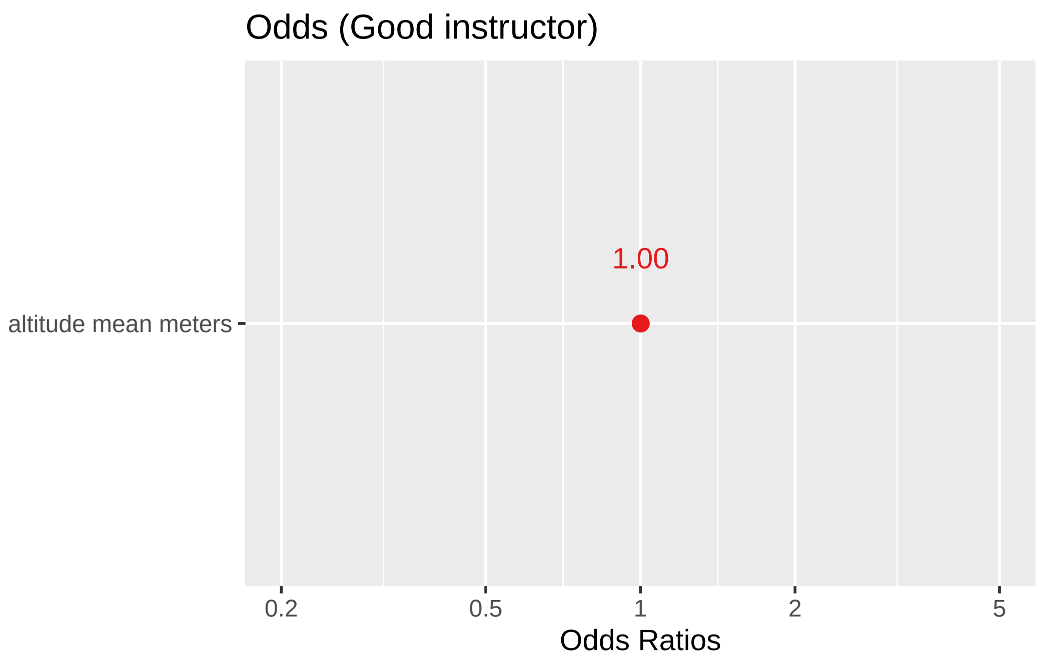
```r
# Calculate lower and upper bounds for 'altitude_mean_meters' odds
exp(mod4.coef.logodds)
```

```
                     Estimate Std. Error      z value Pr(>|z|)
(Intercept)         0.1923644   1.350370 4.137664e-03        1
altitude_mean_meters 1.0012638   1.000217 3.399475e+02        1
```

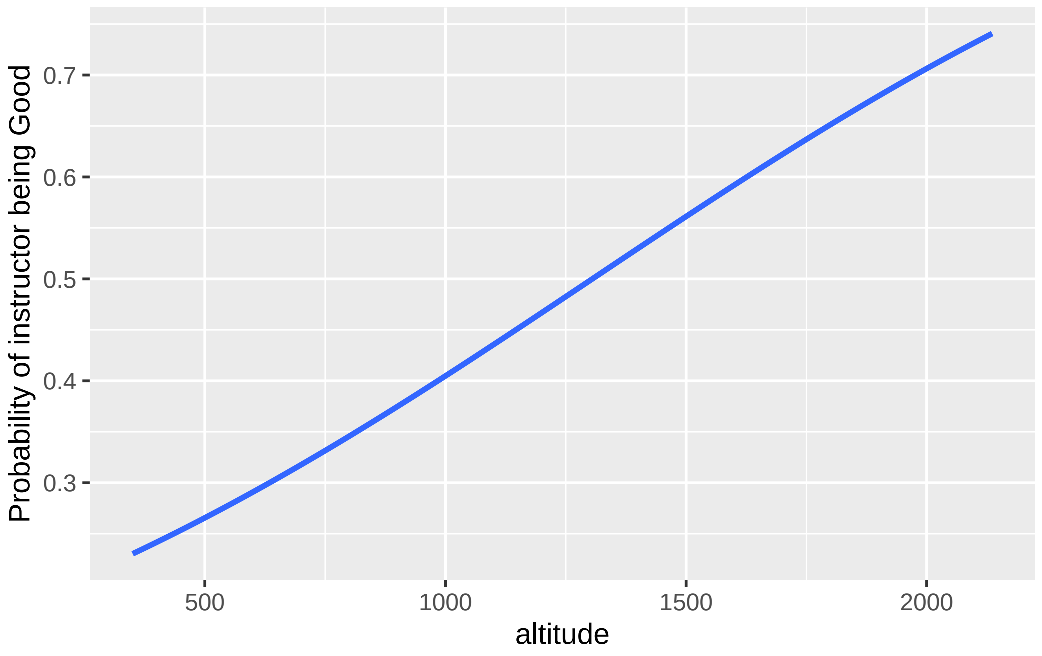
```r
altitude.odds.lower <- exp(altitude.logodds.lower)
altitude.odds.upper <- exp(altitude.logodds.upper)

# Display the confidence interval
paste("(", altitude.odds.lower, ",", altitude.odds.upper, ")")
```

```
[1] "( 1.00083868049254 , 1.00168917949219 )"
```

```r
# Plot odds of being a good instructor
plot_model(model4, show.values = TRUE,
           title = "Odds (Good instructor)", show.p = FALSE)
```

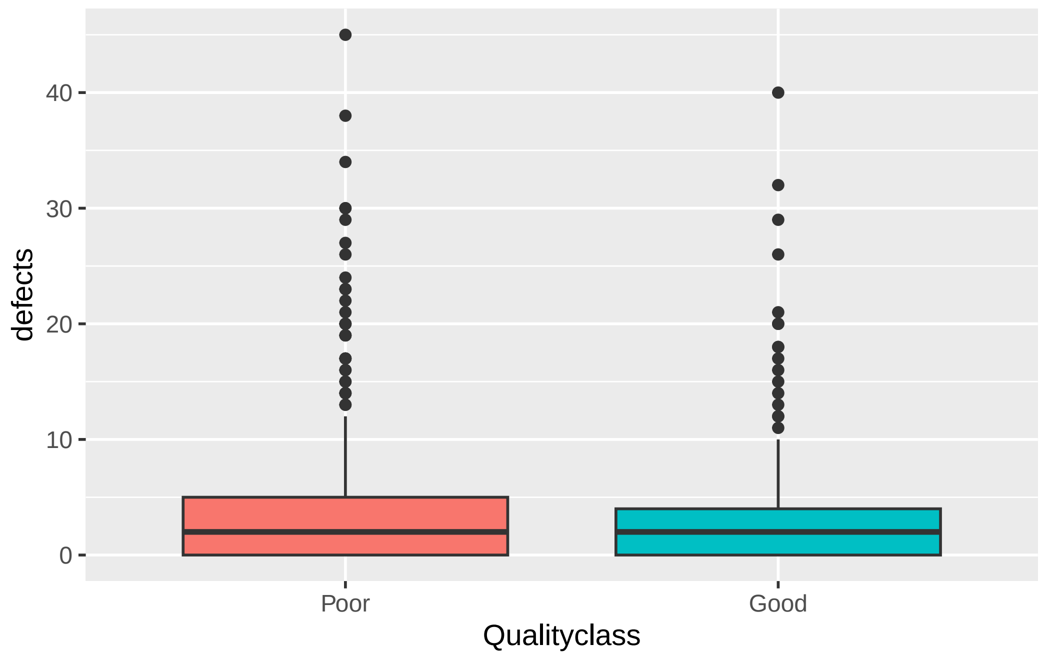


```
# Add predicted probabilities
data_altitude_filtered_after <- data_altitude_filtered %>%
                  mutate(logodds.Good = predict(model4), type = "response") %>%
                  mutate(odds.Good = exp(logodds.Good)) %>%
                  mutate(probs.Good = fitted(model4))

# Plot the relationship between 'altitude_mean_meters' and probability of being a good ins
ggplot(data = data_altitude_filtered_after, aes(x = altitude_mean_meters, y = probs.Good))
  geom_smooth(method="glm",
              method.args = list(family="binomial"),
              se = FALSE) +
  labs(x = "altitude", y = "Probability of instructor being Good")
```

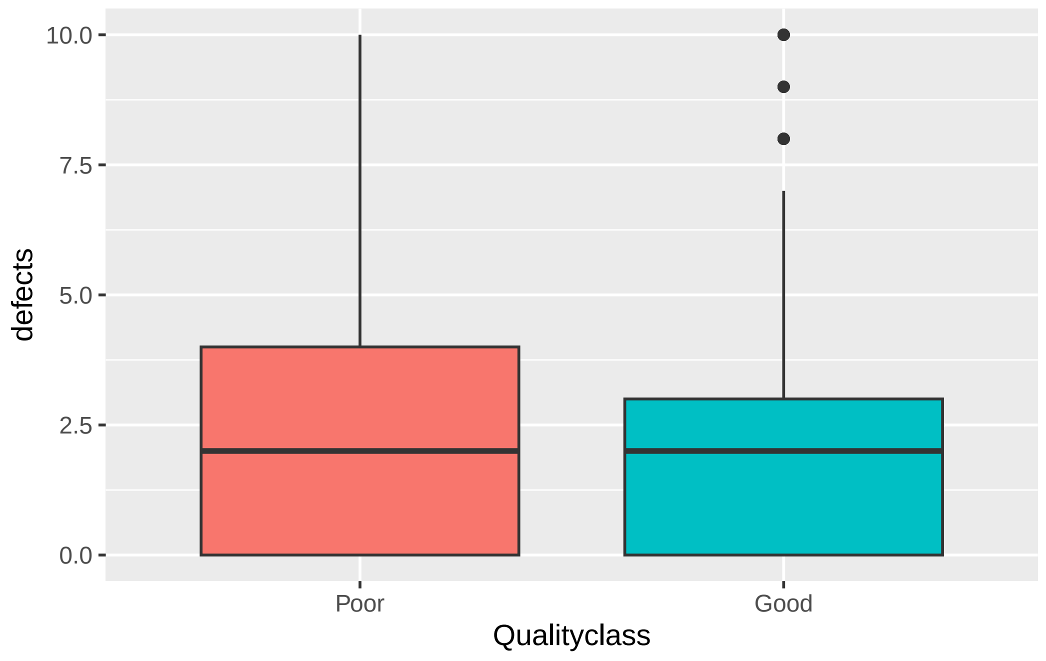


2.5 Category 2 type defects and Qualityclass

```
# Select 'category_two_defects' and 'Qualityclass' columns
data_defects <- data %>%
                  select(category_two_defects, Qualityclass)

# Create a boxplot of 'category_two_defects' across different 'Qualityclass' levels
p9 <- ggplot(data = data_defects, aes(x = Qualityclass, y = category_two_defects, fill = Q
  geom_boxplot() +
  labs(x = "Qualityclass", y = "defects")+
  theme(legend.position = "none")
p9
```



```
# Calculate quartiles and determine lower and upper bounds
q1_defect <- quantile(data_defects$category_two_defects, 0.25)
q3_defect <- quantile(data_defects$category_two_defects, 0.75)
iqr_defect <- q3_defect - q1_defect
lower_bound_defect <- q1_defect - 1.5 * iqr_defect
upper_bound_defect <- q3_defect + 1.5 * iqr_defect

# Filter 'category_two_defects' within the calculated bounds
data_defects_filtered <- data_defects %>%
  filter(category_two_defects >= lower_bound_defect & category_two_defects <= upper_bound_

# Create a boxplot of 'category_two_defects'
p10 <- ggplot(data = data_defects_filtered, aes(x = Qualityclass, y = category_two_defects
  geom_boxplot() +
  labs(x = "Qualityclass", y = "defects")+
  theme(legend.position = "none")
p10
```



```
# Fit logistic regression model with 'category_two_defects' predictor and 'Qualityclass' r
model5 <- glm(Qualityclass ~ category_two_defects, data = data_defects_filtered,
             family = binomial(link = "logit"))
model5 %>%
  summary()
```

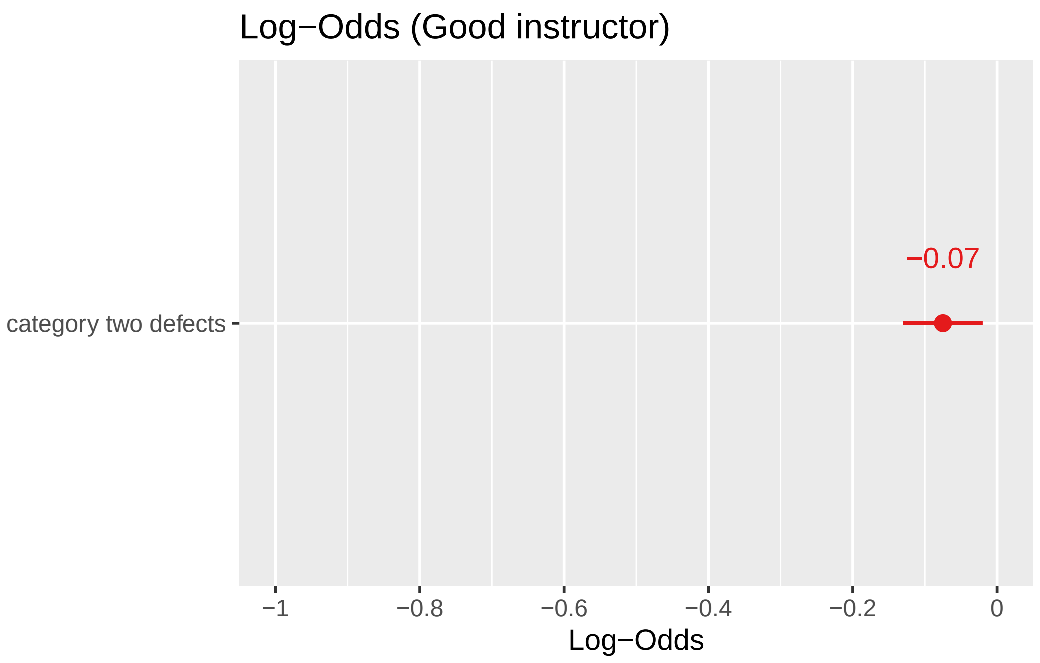
```
Call:
glm(formula = Qualityclass ~ category_two_defects, family = binomial(link = "logit"),
    data = data_defects_filtered)

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)           0.25514    0.09676   2.637  0.00837 **
category_two_defects -0.07499    0.02798  -2.680  0.00736 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1159.2  on 836  degrees of freedom
Residual deviance: 1151.9  on 835  degrees of freedom
```

```
AIC: 1155.9

Number of Fisher Scoring iterations: 3
```

```r
# Calculate lower and upper bounds for 'category_two_defects' log-odds
mod5.coef.logodds <- model5 %>%
                     summary() %>%
                     coef()
defects.logodds.lower <- mod5.coef.logodds["category_two_defects", "Estimate"] -
                     1.96 * mod5.coef.logodds["category_two_defects", "Std. Error"]
defects.logodds.upper <- mod5.coef.logodds["category_two_defects", "Estimate"] +
                     1.96 * mod5.coef.logodds["category_two_defects", "Std. Error"]

# Display the confidence interval
paste("(", defects.logodds.lower, ",", defects.logodds.upper, ")")
```

```
[1] "( -0.129830561393043 , -0.0201503717316868 )"
```

```r
# Plot log-odds of being a good instructor
plot_model(model5, show.values = TRUE, transform = NULL,
           title = "Log-Odds (Good instructor)", show.p = FALSE)
```



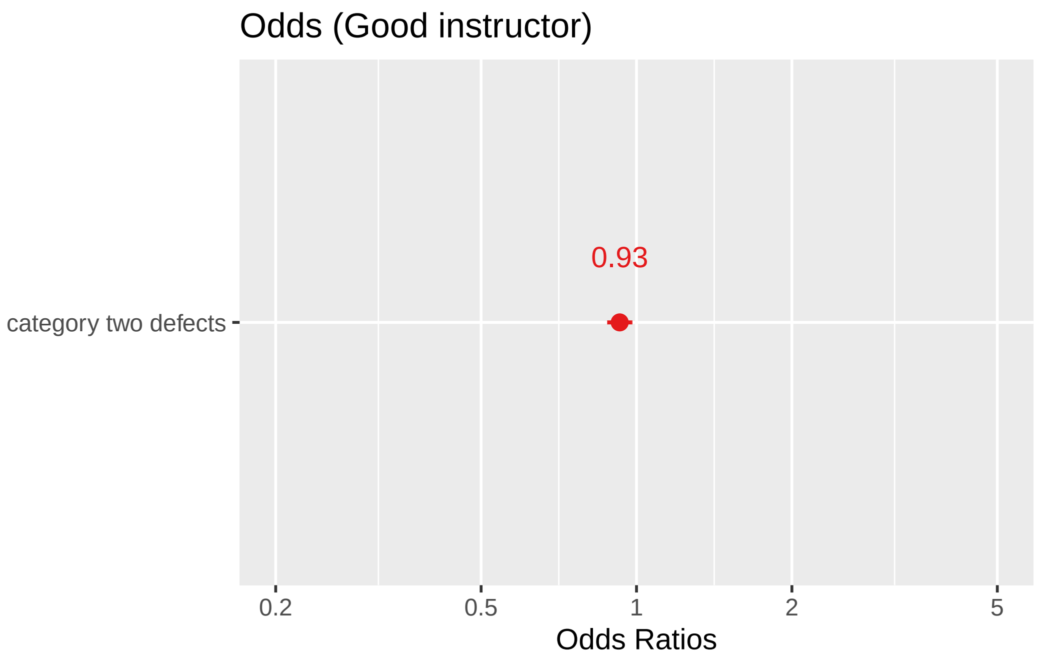
```r
# Calculate lower and upper bounds for 'category_two_defects' odds
exp(mod5.coef.logodds)
```

```
                      Estimate Std. Error     z value Pr(>|z|)
(Intercept)          1.2906462   1.101600 13.96805211 1.008405
category_two_defects 0.9277523   1.028375  0.06855083 1.007385
```

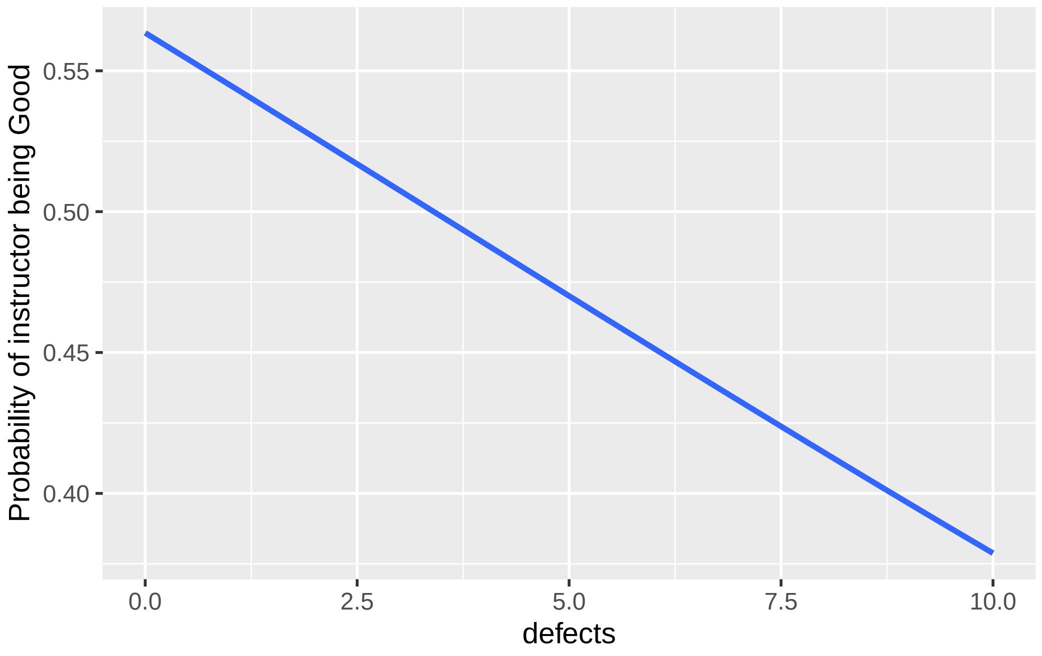
```r
defects.odds.lower <- exp(defects.logodds.lower)
defects.odds.upper <- exp(defects.logodds.upper)

# Display the confidence interval
paste("(", defects.odds.lower, ",", defects.odds.upper, ")")
```

```
[1] "( 0.878244226792678 , 0.980051290216256 )"
```

```r
# Plot odds of being a good instructor
plot_model(model5, show.values = TRUE,
           title = "Odds (Good instructor)", show.p = FALSE)
```

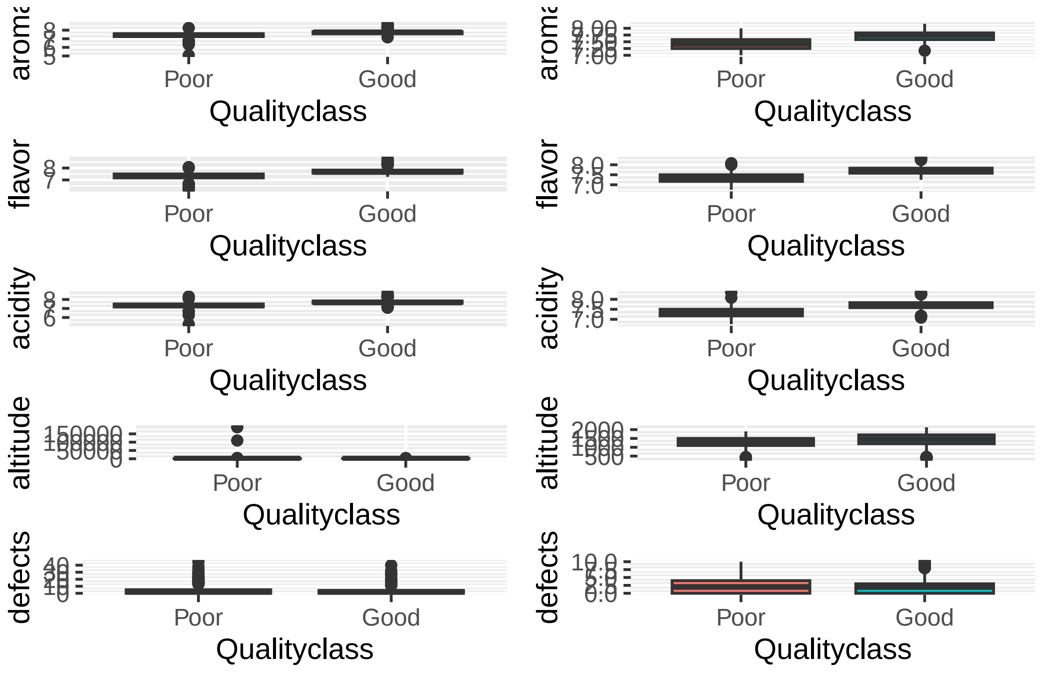


```
# Add predicted probabilities
data_defects_filtered_after <- data_defects_filtered %>%
                  mutate(logodds.Good = predict(model5, type = "response")) %>%
                  mutate(odds.Good = exp(logodds.Good)) %>%
                  mutate(probs.Good = fitted(model5))

# Plot the relationship between 'category_two_defects' and probability of being a good ins
ggplot(data = data_defects_filtered_after, aes(x = category_two_defects, y = probs.Good))
  geom_smooth(method="glm",
              method.args = list(family="binomial"),
              se = FALSE) +
  labs(x = "defects", y = "Probability of instructor being Good")
```



3 Stepwise Predictor Reduction in Model Fitting

```
# Arrange multiple plots
grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, ncol=2)
```



```
# Fit logistic regression model with all predictors
model_full1 <- glm(Qualityclass ~ aroma + flavor + acidity + category_two_defects + altitu
                family = binomial(link = "logit"))
model_full1 %>%
  summary()
```

```
Call:
glm(formula = Qualityclass ~ aroma + flavor + acidity + category_two_defects + 
    altitude_mean_meters, family = binomial(link = "logit"), 
    data = data)

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)    
(Intercept)          -1.193e+02  8.681e+00 -13.747  < 2e-16 ***
aroma                 4.817e+00  6.888e-01   6.994 2.68e-12 ***
flavor                6.913e+00  8.469e-01   8.163 3.27e-16 ***
acidity               4.071e+00  6.820e-01   5.969 2.39e-09 ***
category_two_defects  1.591e-02  2.777e-02   0.573    0.567    
altitude_mean_meters -7.169e-06  2.579e-05  -0.278    0.781    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1239.28  on 893  degrees of freedom
Residual deviance:  532.21  on 888  degrees of freedom
AIC: 544.21

Number of Fisher Scoring iterations: 7
```

```
# Summary statistics for the model
model_full1_summary <- glance(glm(Qualityclass ~ aroma + flavor + acidity + category_two_d
             family = binomial(link = "logit")))
kable(model_full1_summary,digits = 2)
```

null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual	nobs
1239.28	893	-266.11	544.21	572.99	532.21	888	894

Due to the non-significant p-values of ‘category two defects’ and ‘altitude mean meters’, these two variables are removed.

```
# Fit logistic regression model with 'aroma', 'flavor', and 'acidity' predictors
model_full2 <- glm(Qualityclass ~ aroma + flavor + acidity, data = data,
             family = binomial(link = "logit"))
model_full2 %>%
  summary()
```

```
Call:
glm(formula = Qualityclass ~ aroma + flavor + acidity, family = binomial(link = "logit"),
    data = data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept) -119.1572     8.6502 -13.775  < 2e-16 ***
aroma          4.8207     0.6899   6.987 2.80e-12 ***
flavor         6.9028     0.8487   8.133 4.19e-16 ***
acidity        4.0595     0.6826   5.947 2.74e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1239.28  on 893  degrees of freedom
Residual deviance:  532.67  on 890  degrees of freedom
AIC: 540.67

Number of Fisher Scoring iterations: 7
```

```r
# Summary statistics for the model
model_full2_summary <- glance(glm(Qualityclass ~ aroma + flavor + acidity, data = data,
             family = binomial(link = "logit")))
kable(model_full2_summary,digits = 2)
```

null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual	nobs
1239.28	893	-266.34	540.67	559.85	532.67	890	894

Try reducing one predictor to see if it improves the model.

```r
# Fit logistic regression model with 'flavor' and 'acidity' predictors
model_full3 <- glm(Qualityclass ~ flavor + acidity, data = data,
             family = binomial(link = "logit"))
model_full3 %>%
  summary()
```

```
Call:
glm(formula = Qualityclass ~ flavor + acidity, family = binomial(link = "logit"),
    data = data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -95.7956     6.8135 -14.060  < 2e-16 ***
flavor        8.4212     0.7767  10.842  < 2e-16 ***
acidity       4.2916     0.6352   6.756 1.42e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 1239.28  on 893  degrees of freedom
Residual deviance:  596.77  on 891  degrees of freedom
AIC: 602.77

Number of Fisher Scoring iterations: 6
```

```
# Summary statistics for the model
model_full3_summary <- glance(glm(Qualityclass ~ flavor + acidity, data = data,
             family = binomial(link = "logit")))
kable(model_full3_summary,digits = 2)
```

null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual	nobs
1239.28	893	-298.38	602.77	617.15	596.77	891	894

```
# Fit logistic regression model with 'aroma' and 'acidity' predictors
model_full4 <- glm(Qualityclass ~ aroma + acidity, data = data,
             family = binomial(link = "logit"))
model_full4 %>%
  summary()
```

```
Call:
glm(formula = Qualityclass ~ aroma + acidity, family = binomial(link = "logit"),
    data = data)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -96.6561     6.7407  -14.34   <2e-16 ***
aroma         6.5855     0.6308   10.44   <2e-16 ***
acidity       6.2039     0.6015   10.31   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1239.28  on 893  degrees of freedom
Residual deviance:  619.97  on 891  degrees of freedom
AIC: 625.97
```

```
Number of Fisher Scoring iterations: 6
```

```r
# Summary statistics for the model
model_full4_summary <- glance(glm(Qualityclass ~ aroma + acidity, data = data,
             family = binomial(link = "logit")))
kable(model_full4_summary,digits = 2)
```

null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual	nobs
1239.28	893	-309.99	625.97	640.36	619.97	891	894

```r
# Fit logistic regression model with 'aroma' and 'flavor' predictors
model_full5 <- glm(Qualityclass ~ aroma + flavor, data = data,
             family = binomial(link = "logit"))
model_full5 %>%
  summary()
```

```

Call:
glm(formula = Qualityclass ~ aroma + flavor, family = binomial(link = "logit"),
    data = data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -103.9829     7.4589 -13.941  < 2e-16 ***
aroma          5.0424     0.6582   7.661 1.85e-14 ***
flavor         8.7273     0.7896  11.053  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1239.28  on 893  degrees of freedom
Residual deviance:  573.05  on 891  degrees of freedom
AIC: 579.05

Number of Fisher Scoring iterations: 7
```

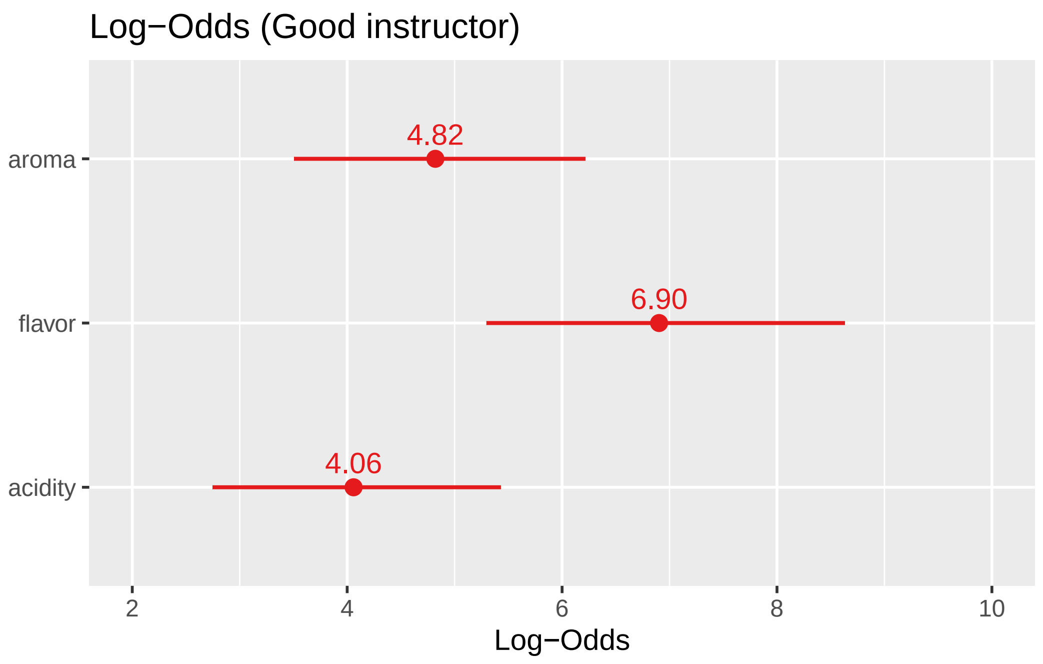
```r
# Summary statistics for the model
model_full5_summary <- glance(glm(Qualityclass ~ aroma + flavor, data = data,
             family = binomial(link = "logit")))
kable(model_full5_summary,digits = 2)
```

null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual	nobs
1239.28	893	-286.52	579.05	593.44	573.05	891	894

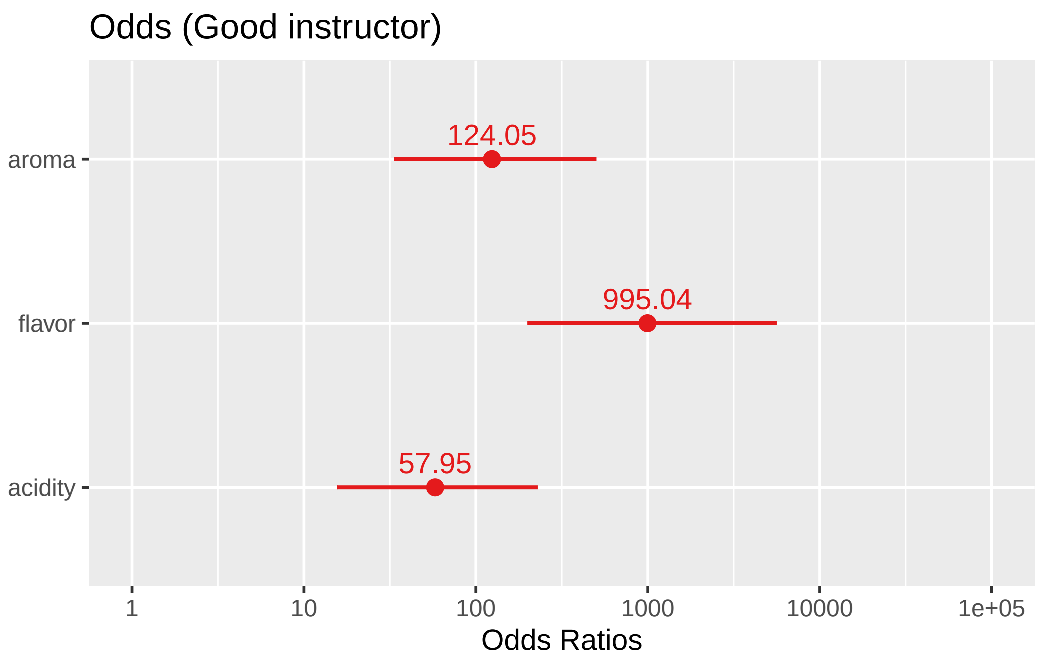
Based on the AIC and BIC values of the above models, it can be concluded that the model with 'aroma', 'flavor', and 'acidity' as predictors is optimal.

Plotting the log-odds and odds of the final model allows for a visual representation of the relationship between the predictors and the response variable.

```r
# Plot log-odds of being a good instructor for model_full2
plot_model(model_full2, show.values = TRUE, transform = NULL,
           title = "Log-Odds (Good instructor)", show.p = FALSE)
```



```r
# Plot odds of being a good instructor for model_full2
plot_model(model_full2, show.values = TRUE,
           title = "Odds (Good instructor)", show.p = FALSE)
```



```r
# Add predicted log-odds, odds, and probabilities of being a good instructor to the data
data_after <- data %>%
          mutate(logodds.Good = round(predict(model_full2), 4)) %>%
          mutate(odds.Good = round(exp(logodds.Good),4)) %>%
          mutate(probs.Good = round(fitted(model_full2),4))

# Select the final predictors and calculate the correlation matrix
data_after1 <- data_after %>%
   select(aroma, flavor, acidity)
data_after1 %>%
  cor()
```

```
            aroma    flavor   acidity
aroma   1.0000000 0.7423400 0.6005359
flavor  0.7423400 1.0000000 0.7555133
acidity 0.6005359 0.7555133 1.0000000
```

4 Principal Component Analysis

Based on the correlation matrix above, it is evident that the predictors exhibit high correlation. Therefore, we adopt principal component analysis (PCA) to help address multicollinearity, thereby enhancing the stability and interpretability of the model.

```r
# Perform principal component analysis (PCA)
data_pca <- data %>%
  select(aroma, flavor, acidity, Qualityclass)
data_scaled <- scale(data_pca[, -4])
pca_result <- prcomp(data_scaled)
pca_result
```

```
Standard deviations (1, .., p=3):
[1] 1.5495698 0.6322373 0.4462169

Rotation (n x k) = (3 x 3):
               PC1         PC2        PC3
aroma   -0.5620910  0.72400462  0.3998387
flavor  -0.6030176 -0.02787715 -0.7972406
acidity -0.5660595 -0.68923158  0.4522570
```

```r
summary(pca_result)
```

```
Importance of components:
                          PC1    PC2     PC3
Standard deviation     1.5496 0.6322 0.44622
Proportion of Variance 0.8004 0.1332 0.06637
Cumulative Proportion  0.8004 0.9336 1.00000
```

```r
# Predict PCA components
pca_result_selected <- predict(pca_result, newdata = data_scaled)
pca_result_selected_df <- as.data.frame(pca_result_selected)

# Combine PCA components with Qualityclass
data_pca_final <- pca_result_selected_df %>%
  mutate(Qualityclass = data_pca$Qualityclass)

# Fit logistic regression model with PCA components
pca_model <- glm(Qualityclass ~ ., data = data_pca_final, family = binomial(link = "logit"
pca_model_summary <- glance(pca_model)
kable(pca_model_summary,digits =2)
```

null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual	nobs
1239.28	893	-266.34	540.67	559.85	532.67	890	894

The cumulative proportion of the three predictor variables adds up to 1, indicating that these three principal components fully explain the variability in the original data without losing information. Therefore, adopting principal component analysis is justified.

Meanwhile, the AIC and BIC values have remained unchanged, and they are still significantly lower than those of other models. This suggests that the three predictors in the original model have already provided the best explanatory power for the response variable. Therefore, the model with three predictors: ‘aroma’, ‘flavor’, and ‘acidity’ is optimal.

$$\text{Qualityclass} = \beta_0 + \beta_1 \times \text{aroma} + \beta_2 \times \text{flavor} + \beta_3 \times \text{acidity} + \epsilon$$

- $Qualityclass$ is the response variable
- $aroma$, $flavor$, and $acidity$ are the predictor variables
- β_0 to β_3 are the coefficients of the model
- ϵ is the error term