Project2

Group-11

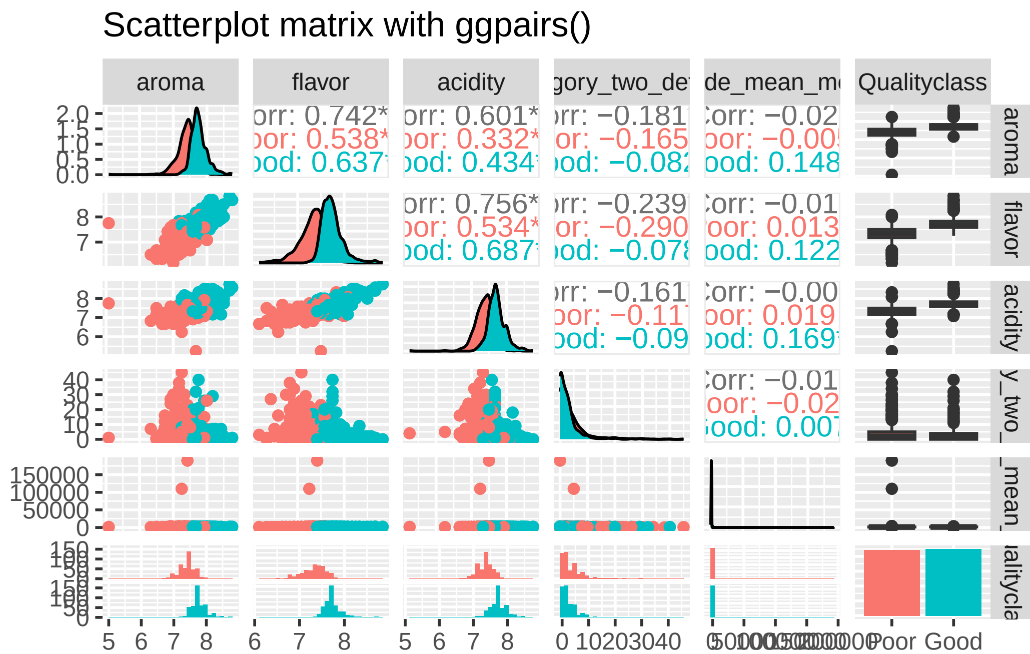
```
library(tidyverse)
library(moderndive)
library(gapminder)
library(sjPlot)
library(jtools)
library(GGally)
library(gt)
library(gridExtra)
library(knitr)
library(patchwork)
library(broom)
library(MASS)
library(janitor)
library(pscl)
library(ggfortify)
```

1 Data Wrangling

Preprocess the data and conduct summary statistics.

```
# Read the dataset
Data <- read.csv("dataset11.csv")
Data <- na.omit(Data)
Data$Qualityclass <- factor(Data$Qualityclass, levels = c("Poor", "Good"))
Data$harvested <- factor(Data$harvested, levels = 2010:2018)

# Scatterplot matrix with ggpairs()
scatterplot = Data %>%
  dplyr::select(aroma, flavor, acidity, category_two_defects, altitude_mean_meters, Qualit
```

```
ggpairs(scatterplot, aes(color = Qualityclass), title="Scatterplot matrix with ggpairs()")
```




```
# Remove outliers
q1_aroma <- quantile(Data$aroma, 0.25)
q3_aroma <- quantile(Data$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
Data1 <- Data %>%
  filter(aroma >= lower_bound_aroma & aroma <= upper_bound_aroma)

q1_flavor <- quantile(Data1$flavor, 0.25)
q3_flavor <- quantile(Data1$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
Data1 <- Data1 %>%
  filter(flavor >= lower_bound_flavor & flavor <= upper_bound_flavor)

q1_acidity <- quantile(Data1$acidity, 0.25)
q3_acidity <- quantile(Data1$acidity, 0.75)
iqr_acidity <- q3_acidity - q1_acidity
```

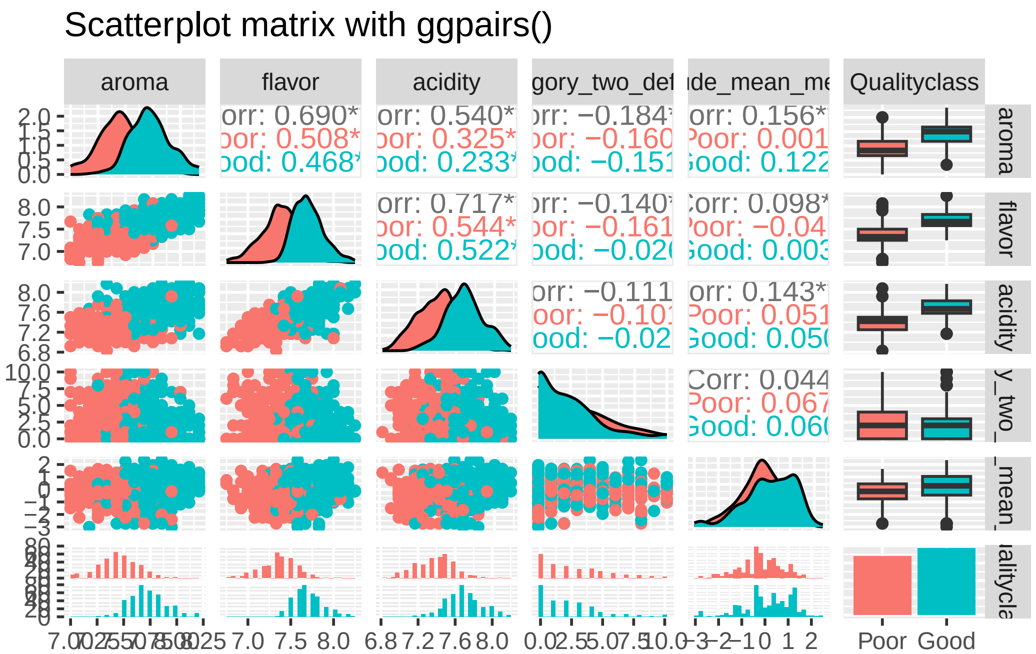
```
lower_bound_acidity <- q1_acidity - 1.5 * iqr_acidity
upper_bound_acidity <- q3_acidity+ 1.5 * iqr_acidity
Data1 <- Data1 %>%
  filter(acidity >= lower_bound_acidity & acidity <= upper_bound_acidity)

q1_defects <- quantile(Data1$category_two_defects, 0.25)
q3_defects <- quantile(Data1$category_two_defects, 0.75)
iqr_defects <- q3_defects - q1_defects
lower_bound_defects <- q1_defects - 1.5 * iqr_defects
upper_bound_defects <- q3_defects + 1.5 * iqr_defects
Data1 <- Data1 %>%
  filter(category_two_defects >= lower_bound_defects & category_two_defects <= upper_bound

q1_altitude <- quantile(Data1$altitude_mean_meters, 0.25)
q3_altitude <- quantile(Data1$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
Data1 <- Data1 %>%
  filter(altitude_mean_meters >= lower_bound_altitude & altitude_mean_meters <= upper_boun
```

```
# Standardize the 'altitude_mean_meters' column
mean_altitude <- mean(Data1$altitude_mean_meters)
sd_altitude <- sd(Data1$altitude_mean_meters)
Data1$altitude_mean_meters <- (Data1$altitude_mean_meters - mean_altitude) / sd_altitude
```

2 Data Visulization

Generate visualizations to better understand the data.

```
# ggpairs of the wrangling data
scatterplot = Data1 %>%
  dplyr::select(aroma, flavor, acidity, category_two_defects, altitude_mean_meters, Qualit
ggpairs(scatterplot, aes(color = Qualityclass), title="Scatterplot matrix with ggpairs()")
```




```
# Summary Statistics for 'aroma' and 'flavor' across different quality classes
Data1 |>
  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),
             .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)
  )
# Summary statistics for 'acidity' and 'category_two_defects' across different quality cla
Data1 |>
```

```r
  summarize('ac.Mean' = mean(acidity),
            'ac.Sd' = sd(acidity),
            'ac.Min' = min(acidity),
            'ac.Max' = max(acidity),
            'C.Mean' = mean(category_two_defects),
            'C.Sd' = sd(category_two_defects),
            'C.Min' = min(category_two_defects),
            'C.Max' = max(category_two_defects),
             .by = Qualityclass) |>
gt() |>
  fmt_number(decimals = 2) |>
  tab_spanner(
    label = "acidity",
    columns = c(ac.Mean, ac.Sd, ac.Min, ac.Max)
  ) |>
  tab_spanner(
    label = "Defects",
    columns = c(C.Mean, C.Sd, C.Min, C.Max)
  )
# Summary statistics for 'altitude_mean_meters' across different quality classes
Data1 |>
  summarize('A.Mean' = mean(altitude_mean_meters),
            'A.Sd' = sd(altitude_mean_meters),
            'A.Min' = min(altitude_mean_meters),
            'A.Max' = max(altitude_mean_meters),
             .by = Qualityclass) |>
gt() |>
  fmt_number(decimals = 2) |>
  tab_spanner(
    label = "Altitude mean meters",
    columns = c(A.Mean, A.Sd, A.Min, A.Max)
  )
```

```r
# Calculate the count of coffee bean qualities for each country
quality_counts <- Data1 %>%
  group_by(country_of_origin, Qualityclass) %>%
  summarise(count = n()) %>%
  spread(Qualityclass, count, fill = 0) %>%
  mutate(proportion_good = Good / (Good + Poor))

# Create a bar plot showing the proportion of good quality coffee beans by country
```

Table 1: Summary statistics

(a)

	aroma				flavor			
Qualityclass	ar.Mean	ar.Sd	ar.Min	ar.Max	fl.Mean	fl.Sd	fl.Min	fl.Max
Poor	7.44	0.19	7.00	8.00	7.36	0.21	6.75	8.08
Good	7.73	0.18	7.17	8.17	7.71	0.17	7.25	8.25

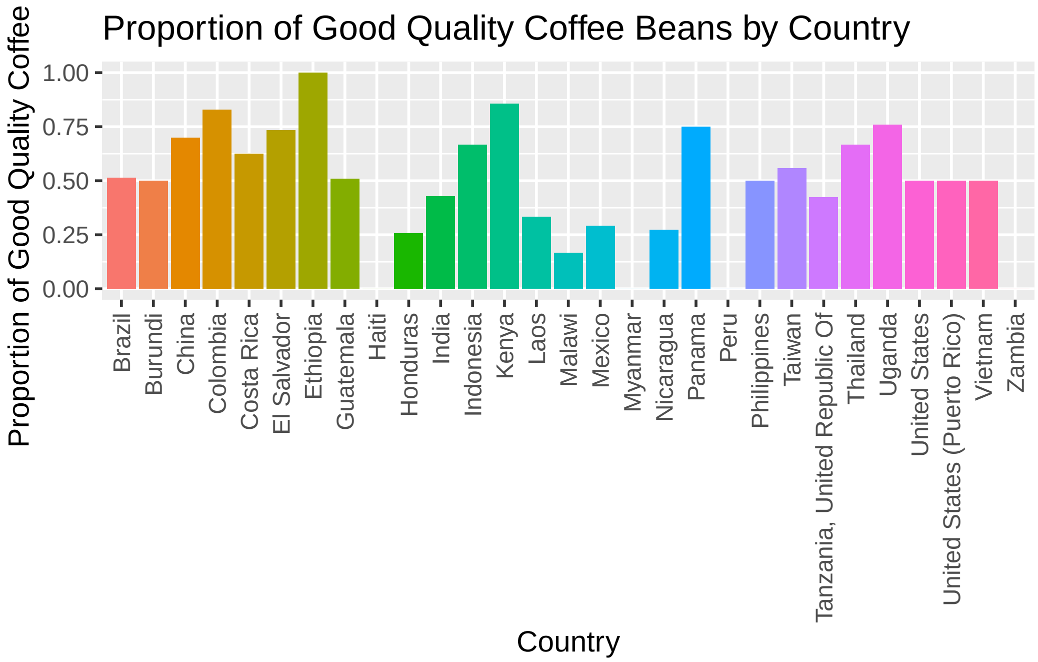
(b)

	acidity				Defects			
Qualityclass	ac.Mean	ac.Sd	ac.Min	ac.Max	C.Mean	C.Sd	C.Min	C.Max
Poor	7.38	0.20	6.83	8.08	2.75	2.64	0.00	10.00
Good	7.69	0.20	7.17	8.17	2.25	2.37	0.00	10.00

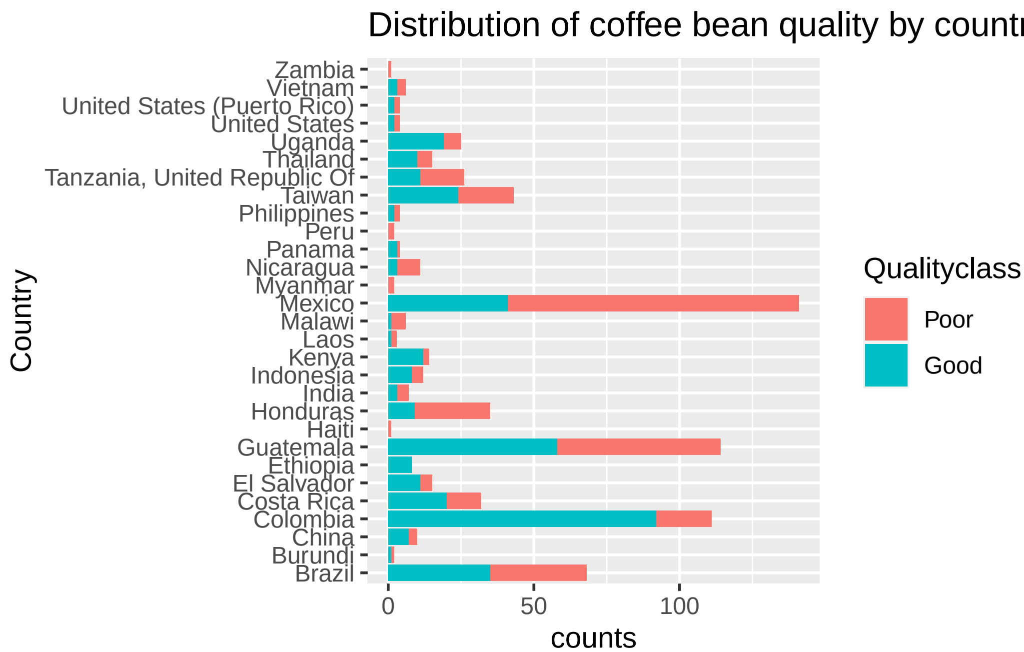
(c)

	Altitude mean meters			
Qualityclass	A.Mean	A.Sd	A.Min	A.Max
Poor	−0.18	0.91	−2.73	1.65
Good	0.16	1.05	−3.00	2.35

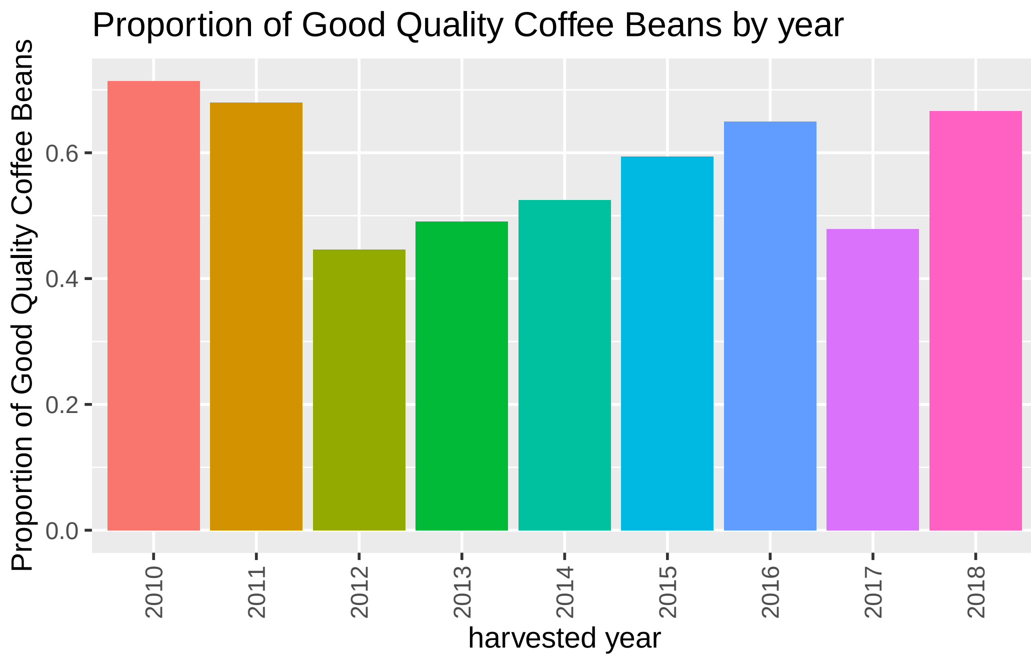
```r
ggplot(quality_counts, aes(x = country_of_origin, y = proportion_good, fill = country_of_o
  geom_bar(stat = "identity", show.legend = FALSE) +
  labs(x = "Country", y = "Proportion of Good Quality Coffee Beans",
       title = "Proportion of Good Quality Coffee Beans by Country") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



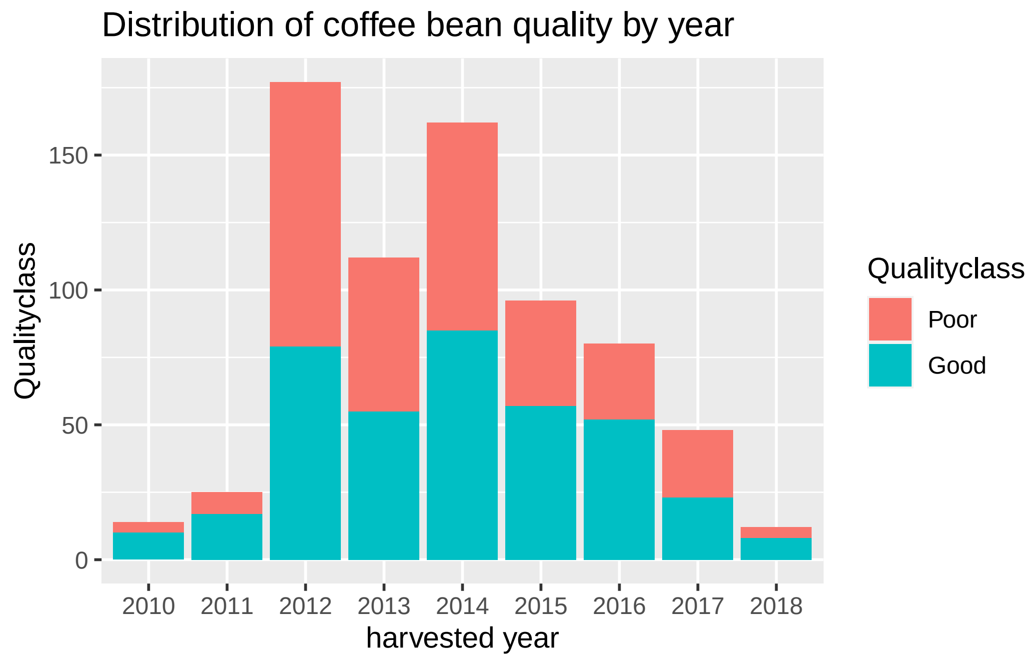

```r
# Create a bar plot to visualize the distribution of coffee bean quality by country
counts <- Data1 %>%
  group_by(country_of_origin, Qualityclass) %>%
  summarise(count = n())
ggplot(data = counts, mapping = aes(x = country_of_origin, y = count, fill = Qualityclass)
  geom_col() +
  labs(x = "Country", y = "counts",
        title = "Distribution of coffee bean quality by country")+
  coord_flip()
```




```r
# Create a bar plot showing the proportion of good quality coffee beans by year
quality_counts1 <- Data1 %>%
  group_by(harvested, Qualityclass) %>%
  summarise(count = n()) %>%
  spread(Qualityclass, count, fill = 0) %>%
  mutate(proportion_good = Good / (Good + Poor))
ggplot(quality_counts1, aes(x =harvested, y = proportion_good, fill = harvested)) +
  geom_bar(stat = "identity",show.legend = FALSE) +
  labs(x = "harvested year", y = "Proportion of Good Quality Coffee Beans",
       title = "Proportion of Good Quality Coffee Beans by year") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```




```r
# Create a bar plot to visualize the distribution of coffee bean quality by year
counts1 <- Data1 %>%
  group_by(harvested, Qualityclass) %>%
  summarise(count = n())
ggplot(data = counts1, mapping = aes(x = harvested, y = count, fill = Qualityclass)) +
  geom_col() +
  labs(x = "harvested year", y = "Qualityclass",
        title = "Distribution of coffee bean quality by year")
```



3 Exploratory Data Analysis

```
# Conduct an origin model
model_full <- glm(Qualityclass ~ country_of_origin + aroma + flavor + acidity + category_t
              family = binomial(link = "logit"))
model_full %>%
  summary()
```

```
Call:
glm(formula = Qualityclass ~ country_of_origin + aroma + flavor +
    acidity + category_two_defects + altitude_mean_meters + harvested,
    family = binomial(link = "logit"), data = Data1)

Coefficients:
                                                 Estimate Std. Error z value
(Intercept)                                    -155.93383   13.71901 -11.366
country_of_originBurundi                          1.92335    5.32186   0.361
country_of_originChina                            0.51607    1.23498   0.418
country_of_originColombia                         1.79012    0.63394   2.824
country_of_originCosta Rica                       0.38187    0.87151   0.438
```

```
country_of_originEl Salvador                        0.17324    0.97069   0.178
country_of_originEthiopia                          12.19028 1178.76284   0.010
country_of_originGuatemala                         -0.82776    0.61013  -1.357
country_of_originHaiti                            -13.36267 3956.18039  -0.003
country_of_originHonduras                          -1.10992    0.80864  -1.373
country_of_originIndia                             -3.07110    1.13658  -2.702
country_of_originIndonesia                         -0.68124    1.19806  -0.569
country_of_originKenya                              0.02514    1.77726   0.014
country_of_originLaos                               1.13360    1.96356   0.577
country_of_originMalawi                            -0.65762    1.41622  -0.464
country_of_originMexico                            -0.89996    0.57619  -1.562
country_of_originMyanmar                          -14.36618 2797.39843  -0.005
country_of_originNicaragua                          0.05486    1.82141   0.030
country_of_originPanama                             3.38904    1.78721   1.896
country_of_originPeru                             -18.66679 2192.69041  -0.009
country_of_originPhilippines                        2.80520    3.13355   0.895
country_of_originTaiwan                             0.62404    0.78671   0.793
country_of_originTanzania, United Republic Of       0.93548    0.91419   1.023
country_of_originThailand                           2.18907    0.95049   2.303
country_of_originUganda                            -1.48521    0.86689  -1.713
country_of_originUnited States                      1.84776    2.01063   0.919
country_of_originUnited States (Puerto Rico)       -1.41603    1.49613  -0.946
country_of_originVietnam                            1.67138    1.29784   1.288
country_of_originZambia                           -13.01366 3956.18042  -0.003
aroma                                               6.03872    0.99701   6.057
flavor                                              8.29375    1.15796   7.162
acidity                                             6.21276    0.98315   6.319
category_two_defects                                0.11822    0.05970   1.980
altitude_mean_meters                                0.24303    0.18054   1.346
harvested2011                                      -0.39748    1.21849  -0.326
harvested2012                                       0.01028    1.06170   0.010
harvested2013                                       0.41447    1.06015   0.391
harvested2014                                       0.52414    1.08744   0.482
harvested2015                                       0.43049    1.07645   0.400
harvested2016                                       1.34805    1.14006   1.182
harvested2017                                       1.29547    1.13936   1.137
harvested2018                                       2.35384    1.52305   1.545
                                                  Pr(>|z|)
(Intercept)                                        < 2e-16 ***
country_of_originBurundi                           0.71780
country_of_originChina                             0.67603
country_of_originColombia                          0.00475 **
country_of_originCosta Rica                        0.66126
```

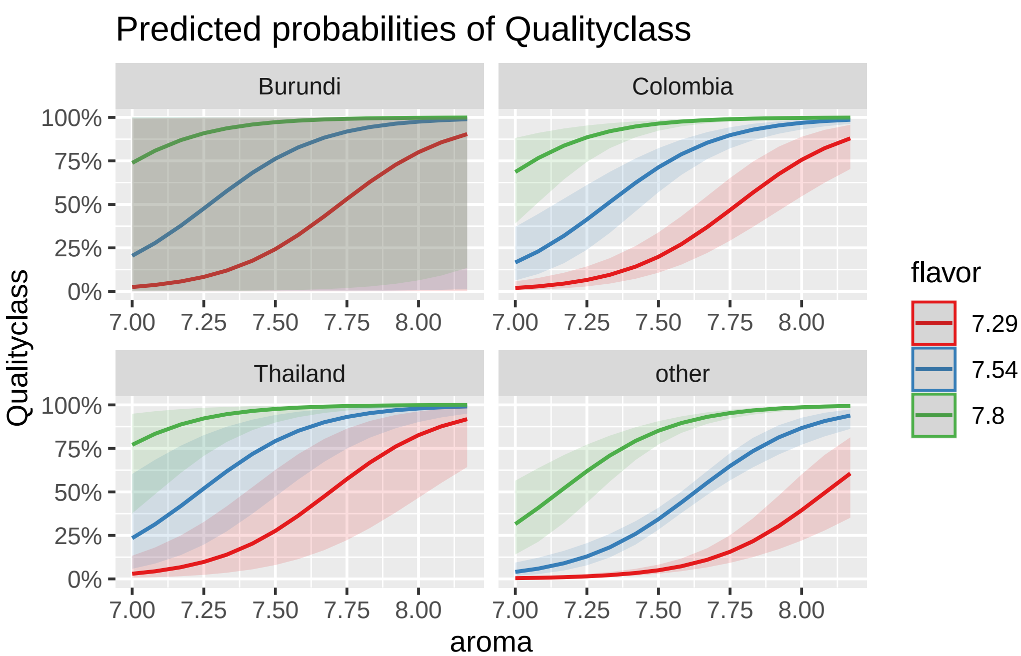
```
country_of_originEl Salvador                    0.85835
country_of_originEthiopia                       0.99175
country_of_originGuatemala                      0.17487
country_of_originHaiti                          0.99731
country_of_originHonduras                       0.16988
country_of_originIndia                          0.00689 **
country_of_originIndonesia                      0.56961
country_of_originKenya                          0.98871
country_of_originLaos                           0.56372
country_of_originMalawi                         0.64240
country_of_originMexico                         0.11831
country_of_originMyanmar                        0.99590
country_of_originNicaragua                      0.97597
country_of_originPanama                         0.05792 .
country_of_originPeru                           0.99321
country_of_originPhilippines                    0.37067
country_of_originTaiwan                         0.42765
country_of_originTanzania, United Republic Of   0.30617
country_of_originThailand                       0.02127 *
country_of_originUganda                         0.08666 .
country_of_originUnited States                  0.35810
country_of_originUnited States (Puerto Rico)    0.34391
country_of_originVietnam                        0.19781
country_of_originZambia                         0.99738
aroma                                          1.39e-09 ***
flavor                                         7.93e-13 ***
acidity                                        2.63e-10 ***
category_two_defects                            0.04767 *
altitude_mean_meters                            0.17827
harvested2011                                   0.74427
harvested2012                                   0.99228
harvested2013                                   0.69583
harvested2014                                   0.62981
harvested2015                                   0.68922
harvested2016                                   0.23703
harvested2017                                   0.25553
harvested2018                                   0.12223
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1003.53  on 725  degrees of freedom
```

```
Residual deviance:  361.12  on 684  degrees of freedom
AIC: 445.12

Number of Fisher Scoring iterations: 16
```

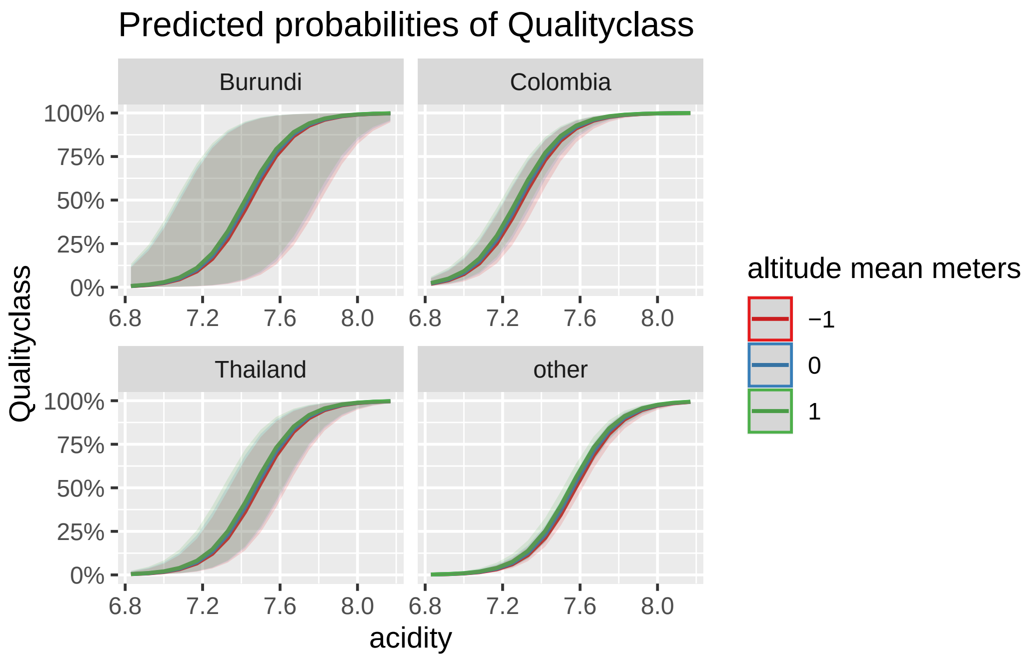
```
# Create a new variable 'country_category' based on 'country_of_origin' and convert 'count
data <- Data1 %>%
  mutate(country_category = ifelse(country_of_origin %in% c("Burundi",  "Colombia", "Thail
  dplyr::select(country_of_origin, country_category, everything())
data$country_category <- factor(data$country_category, levels = c("Burundi", "Colombia", "
```

3.1 Interaction effects plots

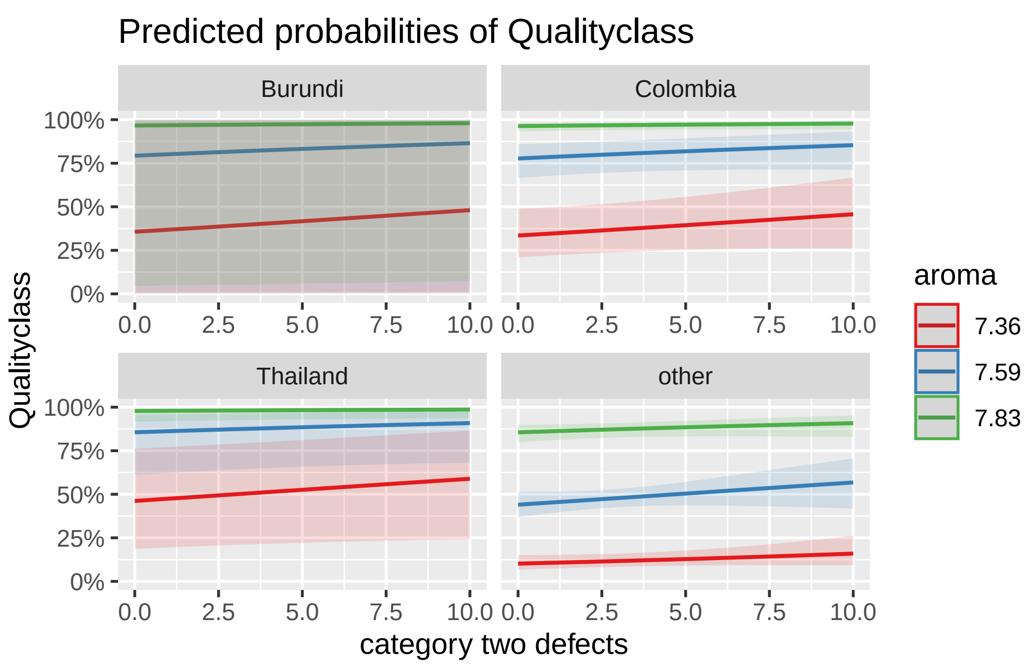
```
model_country_aroma_flavor <- glm(Qualityclass ~ country_category + aroma + flavor, data =
plot_model(model_country_aroma_flavor, type = "pred",
           terms = c("aroma", "flavor", "country_category"))
```




```
model_country_acidity_altitude <- glm(Qualityclass ~ country_category + acidity + altitude
plot_model(model_country_acidity_altitude, type = "pred",
           terms = c("acidity", "altitude_mean_meters", "country_category"))
```




```
model_aroma_flavor_acidity <- glm(Qualityclass ~ country_category + aroma + category_two_d
plot_model(model_aroma_flavor_acidity, type = "pred",
           terms = c("category_two_defects", "aroma", "country_category"))
```



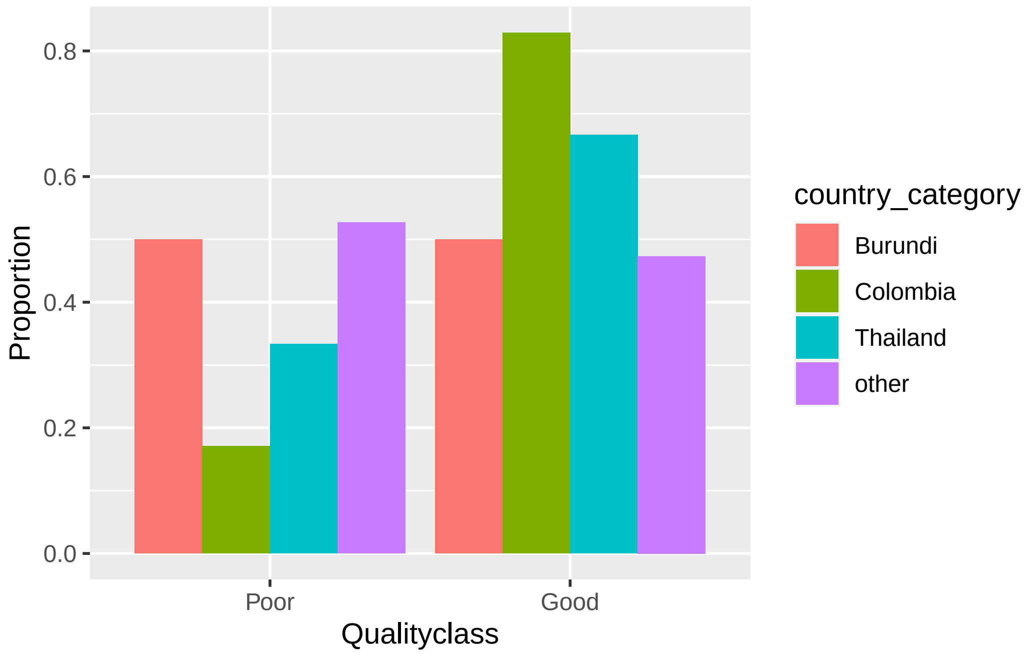
Modeling each predictor separately with the response variable to observe the individual impact of each feature on the quality of coffee.

3.2 Country and Qualityclass

```
# Select 'country_category' and 'Qualityclass' columns and generate a contingency table.
data_country_category <- data %>%
          dplyr::select(country_category, Qualityclass)
data_country_category %>%
  tabyl(country_category, Qualityclass) %>%
  adorn_percentages() %>%
  adorn_pct_formatting() %>%
  adorn_ns()
```

```
country_category        Poor        Good
         Burundi 50.0%   (1) 50.0%   (1)
        Colombia 17.1%  (19) 82.9%  (92)
        Thailand 33.3%   (5) 66.7%  (10)
           other 52.7% (315) 47.3                                   (283)
```

```
# Create a barplot of 'country_category' across different 'Qualityclass' levels
p0 <- ggplot(data_country_category, aes(x = Qualityclass, y = after_stat(prop), group = co
    geom_bar(position = "dodge", stat = "count") +
    labs(y = "Proportion")
p0
```




```
# Fit logistic regression model with 'country_category' predictor and 'Qualityclass' respo
model_country <- glm(Qualityclass ~ country_category, data = data_country_category, family
model_country %>%
  summary()
```

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

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)              -8.713e-14  1.414e+00   0.000    1.000
country_categoryColombia  1.577e+00  1.436e+00   1.098    0.272
country_categoryThailand  6.931e-01  1.517e+00   0.457    0.648
country_categoryother    -1.071e-01  1.417e+00  -0.076    0.940

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1003.53  on 725  degrees of freedom
Residual deviance:  950.78  on 722  degrees of freedom
AIC: 958.78
```

```
Number of Fisher Scoring iterations: 4
```

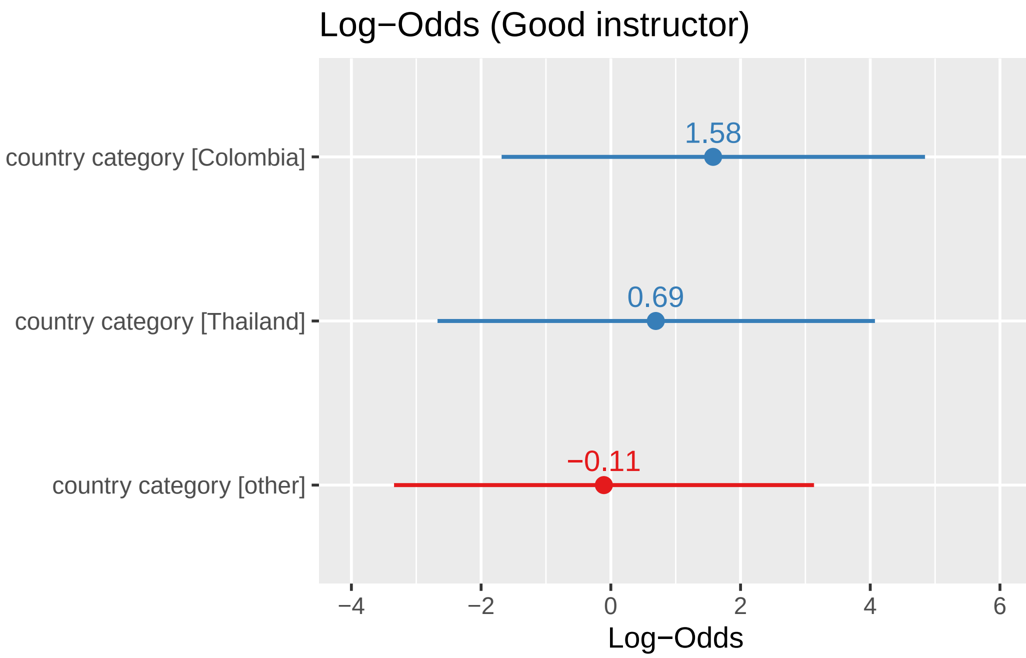
```r
# Extract coefficients from the model and calculate their confidence intervals.
model_country_coef_logodds <- model_country %>%
  summary() %>%
  coef()
model_country_coef_logodds
```

```
                             Estimate Std. Error       z value  Pr(>|z|)
(Intercept)              -8.712937e-14   1.414214 -6.160977e-14 1.0000000
country_categoryColombia  1.577350e+00   1.436489  1.098059e+00 0.2721788
country_categoryThailand  6.931472e-01   1.516575  4.570477e-01 0.6476367
country_categoryother    -1.071257e-01   1.416583 -7.562262e-02 0.9397193
```

```r
confint_logodds <- confint(model_country)
confint_logodds
```

```
                             2.5 %   97.5 %
(Intercept)              -3.230337 3.230337
country_categoryColombia -1.681791 4.837248
country_categoryThailand -2.670586 4.065181
country_categoryother    -3.340551 3.126299
```

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



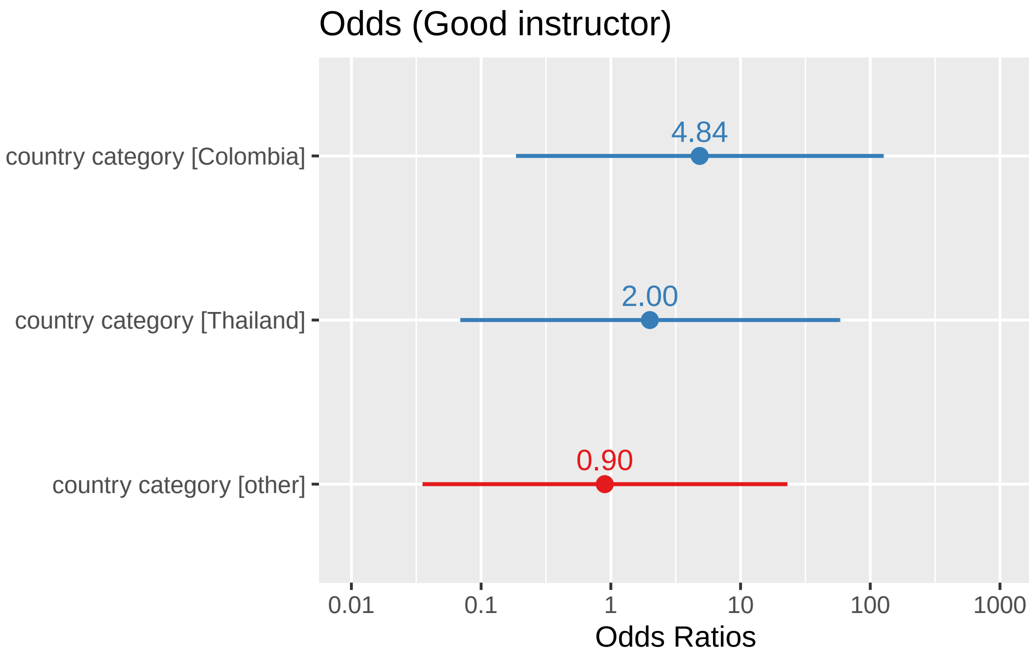

```
# Transform the coefficients into odds ratios and obtain their confidence intervals
model_country_coef_odds <- model_country %>%
  summary() %>%
  coef() %>%
  exp()
model_country_coef_odds
```

```
                          Estimate Std. Error  z value Pr(>|z|)
(Intercept)              1.0000000   4.113250 1.000000 2.718282
country_categoryColombia 4.8421053   4.205904 2.998340 1.312822
country_categoryThailand 2.0000000   4.556592 1.579404 1.911019
country_categoryother    0.8984127   4.123009 0.927166 2.559263
```

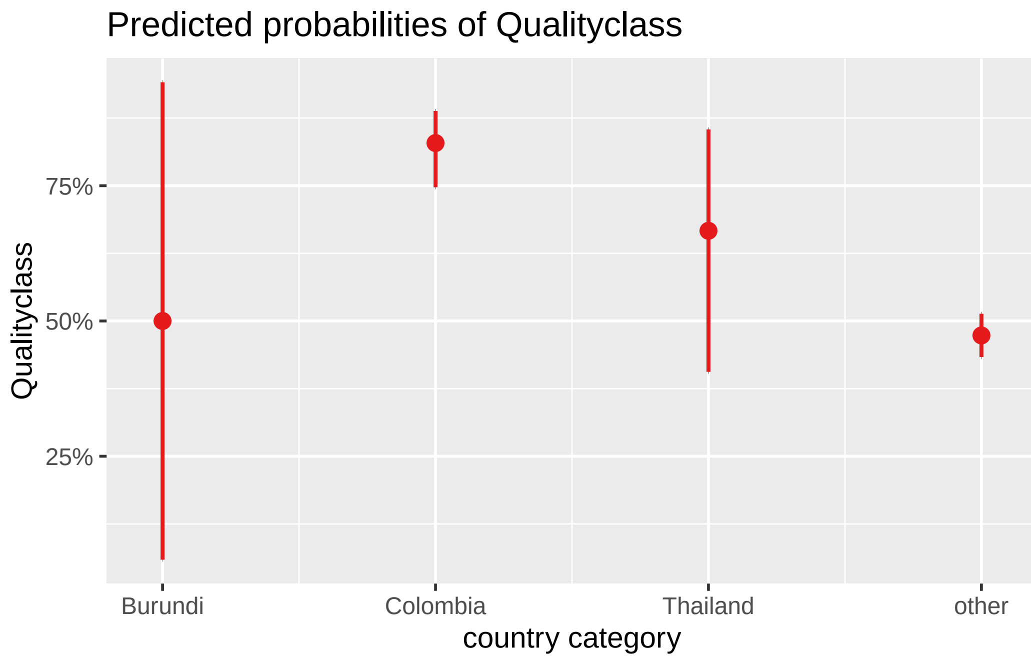
```
exp(confint_logodds)
```

```
                             2.5 %    97.5 %
(Intercept)              0.03954417  25.28818
country_categoryColombia 0.18604041 126.12174
country_categoryThailand 0.06921165  58.27545
country_categoryother    0.03541742  22.78949
```

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

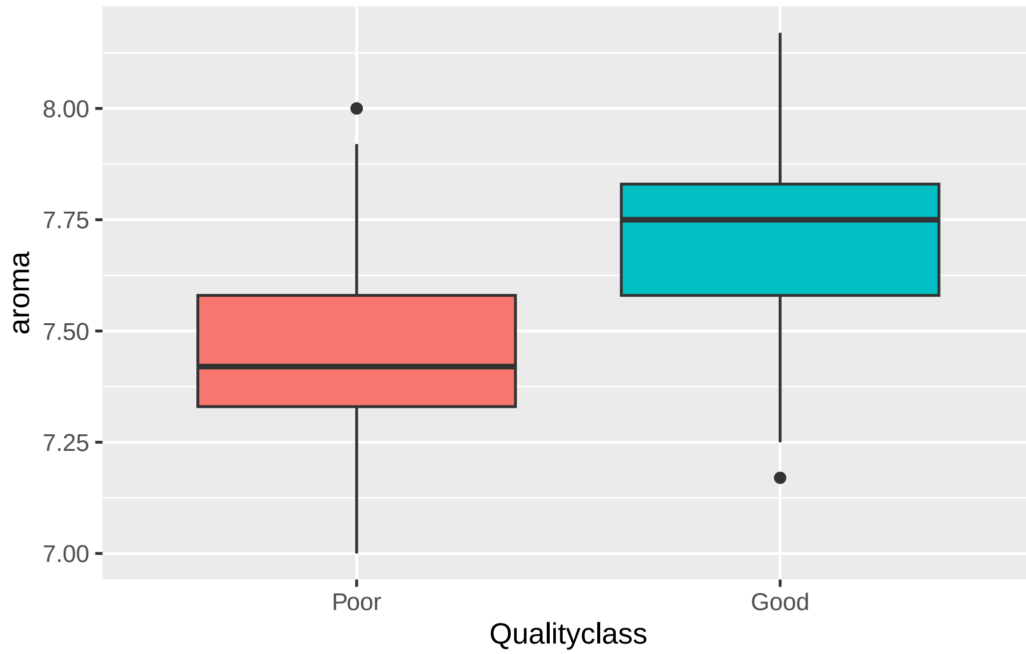



```r
# Calculate log odds, odds, and probabilities and store them
data_country_category_after <- data_country_category %>%
            mutate(logodds.Good = predict(model_country, type = "response")) %>%
            mutate(odds.Good = exp(logodds.Good)) %>%
            mutate(probs.Good = fitted(model_country))

# Generate a predictive plot
plot_model(model_country, type = "pred",
           terms = c("country_category"))
```



3.3 Aroma and Qualityclass

```
# Select 'aroma' and 'Qualityclass' columns
data_aroma <- data %>%
            dplyr::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
```




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

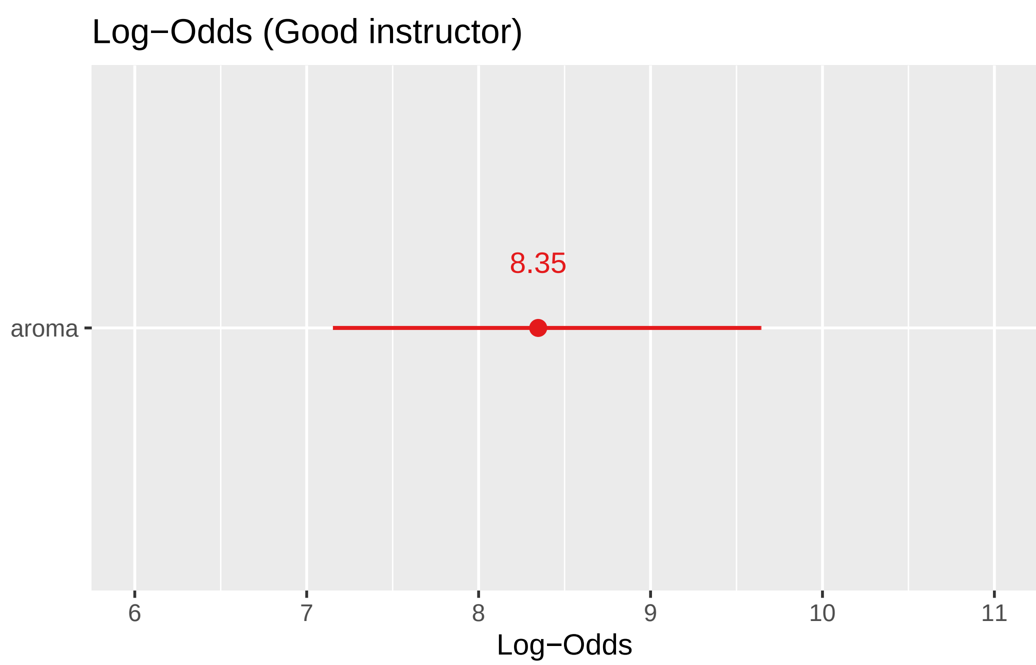
```

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

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -63.1944     4.8098  -13.14   <2e-16 ***
aroma         8.3465     0.6342   13.16   <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: 1003.53  on 725  degrees of freedom
Residual deviance:  676.23  on 724  degrees of freedom
```

```
AIC: 680.23

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] "( 7.1035545966061 , 9.58952426960609 )"
```

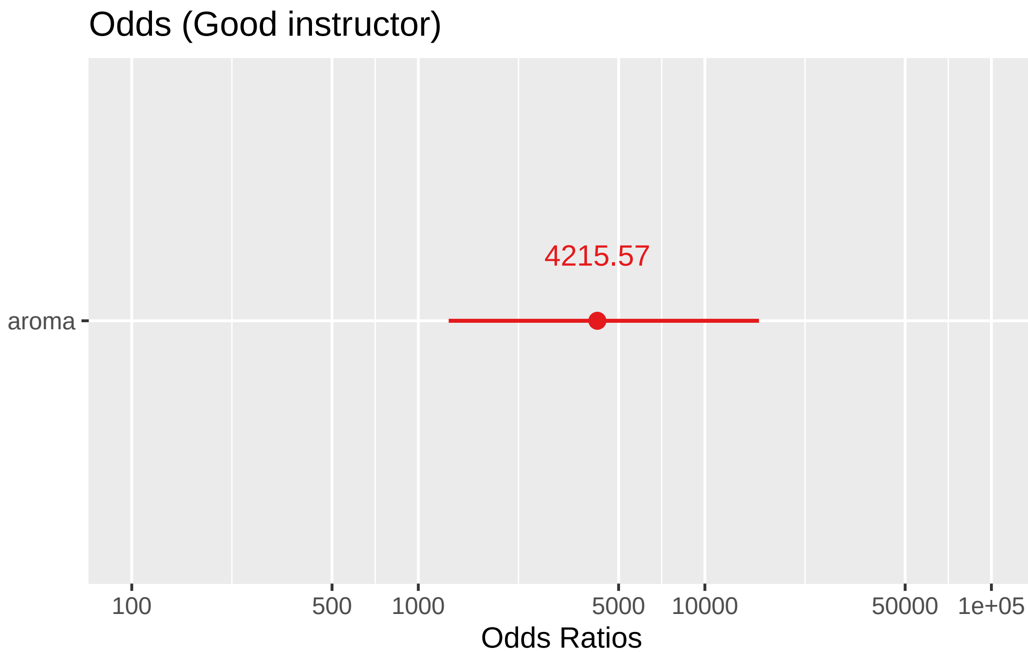
```
# Plot log-odds of being a good instructor
plot_model(model1, show.values = TRUE, transform = NULL,
           title = "Log-Odds (Good instructor)", show.p = FALSE)
```



```r
# Calculate lower and upper bounds for 'aroma' odds
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, ")")
```

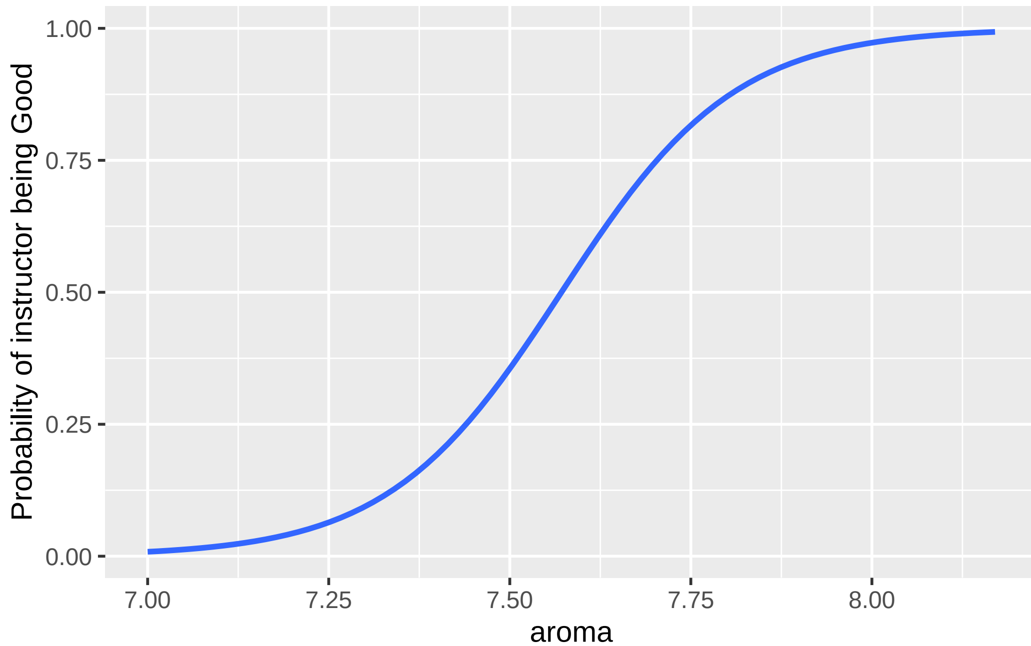
```
[1] "( 1216.28279431965 , 14610.9170234023 )"
```

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



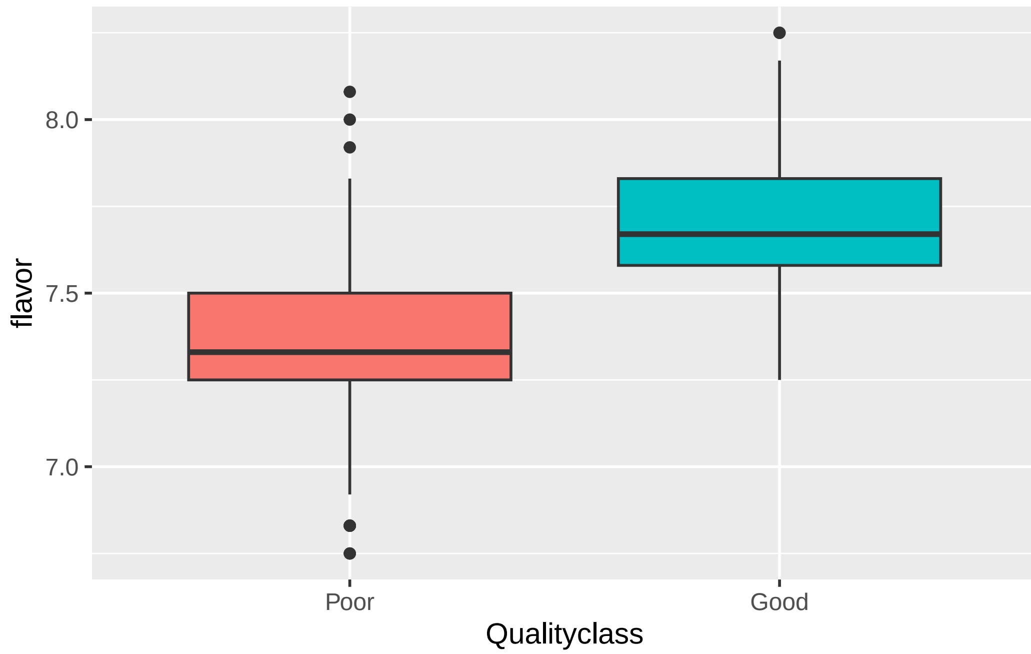

```r
# Add predicted probabilities
data_aroma_after <- data_aroma %>%
                    mutate(logodds.Good = predict(model1, type = "response")) %>%
                    mutate(odds.Good = exp(logodds.Good)) %>%
                    mutate(probs.Good = fitted(model1))

# Plot the relationship between 'aroma' and probability of being a good instructor
ggplot(data = data_aroma_after, aes(x = aroma, y = probs.Good)) +
```

```r
  geom_smooth(method = "glm",
              method.args = list(family = "binomial"),
              se = FALSE) +
  labs(x = "aroma", y = "Probability of instructor being Good")
```



3.4 Flavor and Qualityclass

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

# Create a boxplot of 'flavor' across different 'Qualityclass' levels
p2 <- ggplot(data = data_flavor, aes(x = Qualityclass, y = flavor, fill = Qualityclass)) +
  geom_boxplot() +
  labs(x = "Qualityclass", y = "flavor")+
  theme(legend.position = "none")
p2
```




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

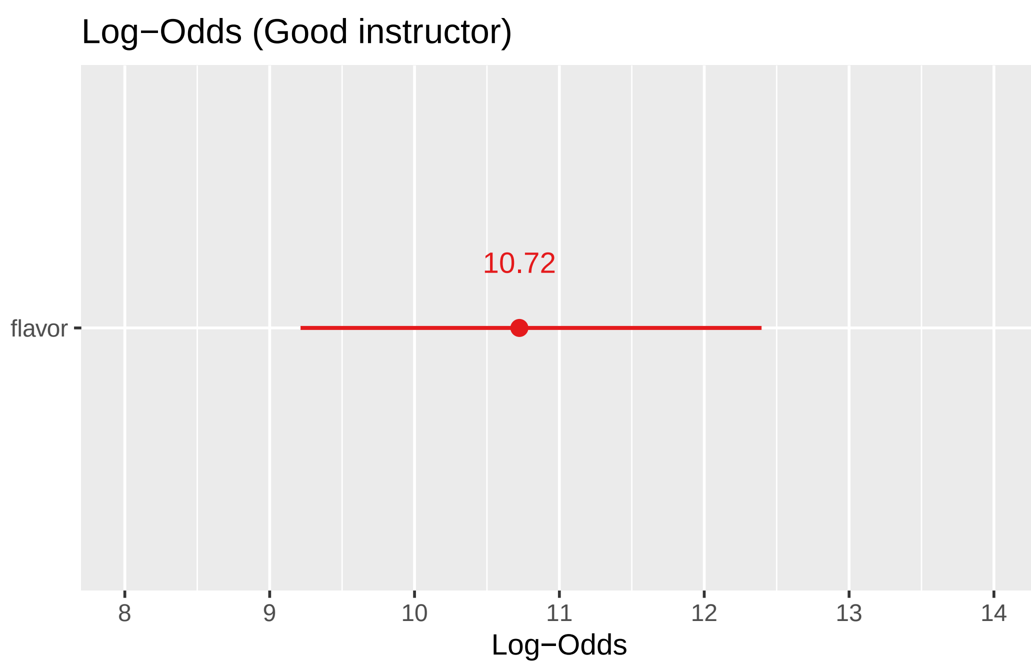
```
Call:
glm(formula = Qualityclass ~ flavor, family = binomial(link = "logit"),
    data = data_flavor)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -80.7455     6.1070  -13.22   <2e-16 ***
flavor       10.7238     0.8097   13.24   <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: 1003.53  on 725  degrees of freedom
Residual deviance:  563.98  on 724  degrees of freedom
```

```
AIC: 567.98

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] "( 9.1369267163387 , 12.3107616668265 )"
```

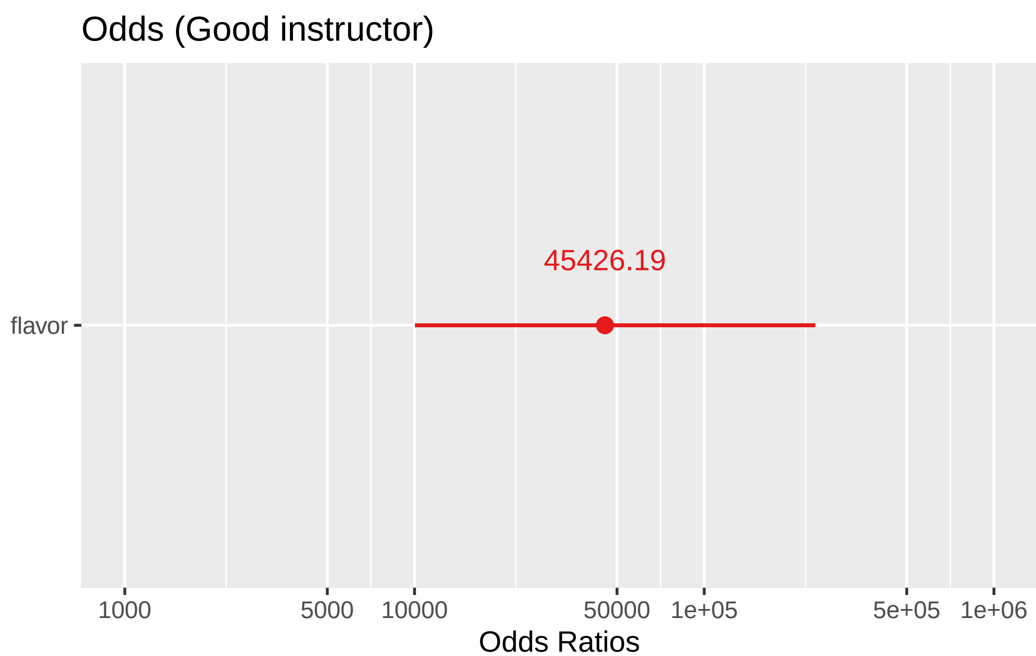
```
# Plot log-odds of being a good instructor
plot_model(model2, show.values = TRUE, transform = NULL,
           title = "Log-Odds (Good instructor)", show.p = FALSE)
```



```
# Calculate lower and upper bounds for 'flavor' odds
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, ")")
```

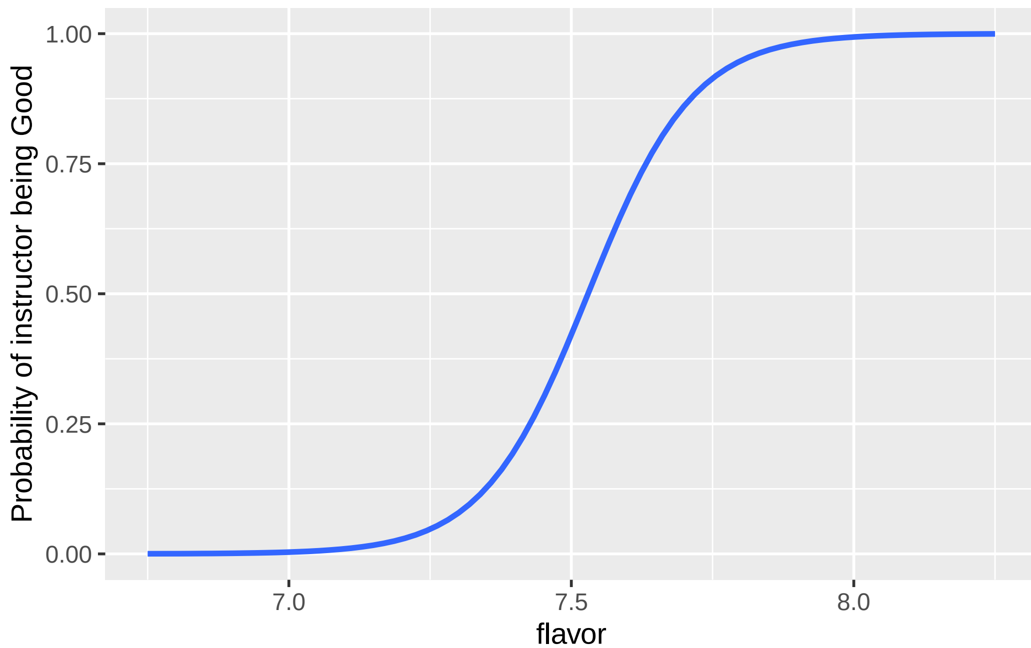
```
[1] "( 9292.16374923337 , 222073.051342309 )"
```

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



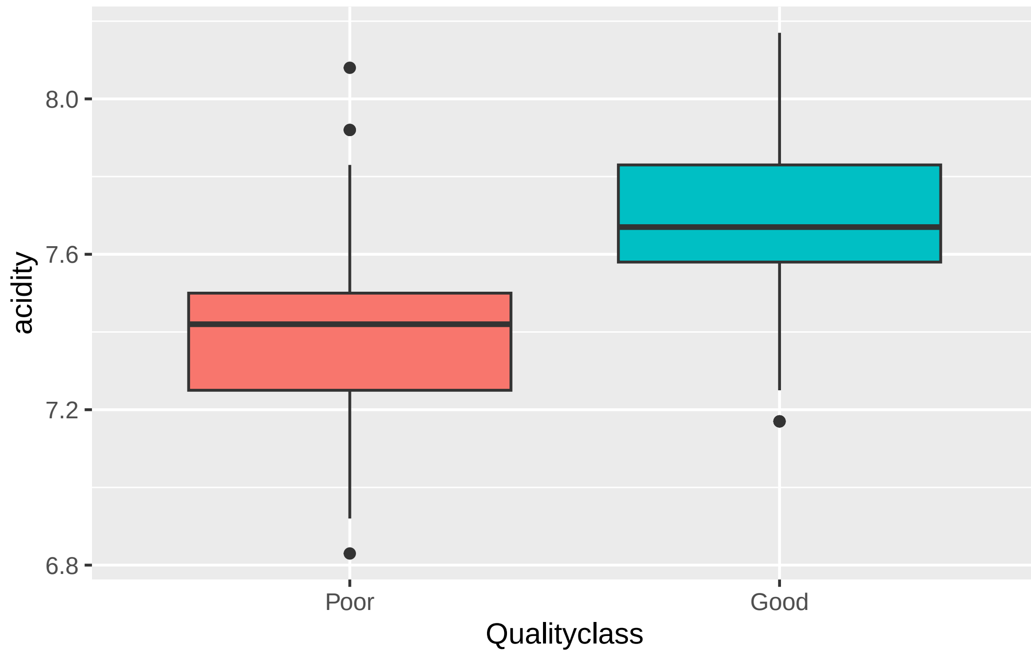

```
# Add predicted probabilities
data_flavor_after <- data_flavor %>%
                    mutate(logodds.Good = predict(model2, type = "response")) %>%
                    mutate(odds.Good = exp(logodds.Good)) %>%
                    mutate(probs.Good = fitted(model2))

# Plot the relationship between 'flavor' and probability of being a good instructor
ggplot(data = data_flavor_after, aes(x = flavor, y = probs.Good)) +
```

```r
  geom_smooth(method="glm",
              method.args = list(family="binomial"),
              se = FALSE) +
  labs(x = "flavor", y = "Probability of instructor being Good")
```



3.5 Acidity and Qualityclass

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

# Create a boxplot of 'acidity' across different 'Qualityclass' levels
p3 <- ggplot(data = data_acidity, aes(x = Qualityclass, y = acidity, fill = Qualityclass))
  geom_boxplot() +
  labs(x = "Qualityclass", y = "acidity")+
  theme(legend.position = "none")
p3
```




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

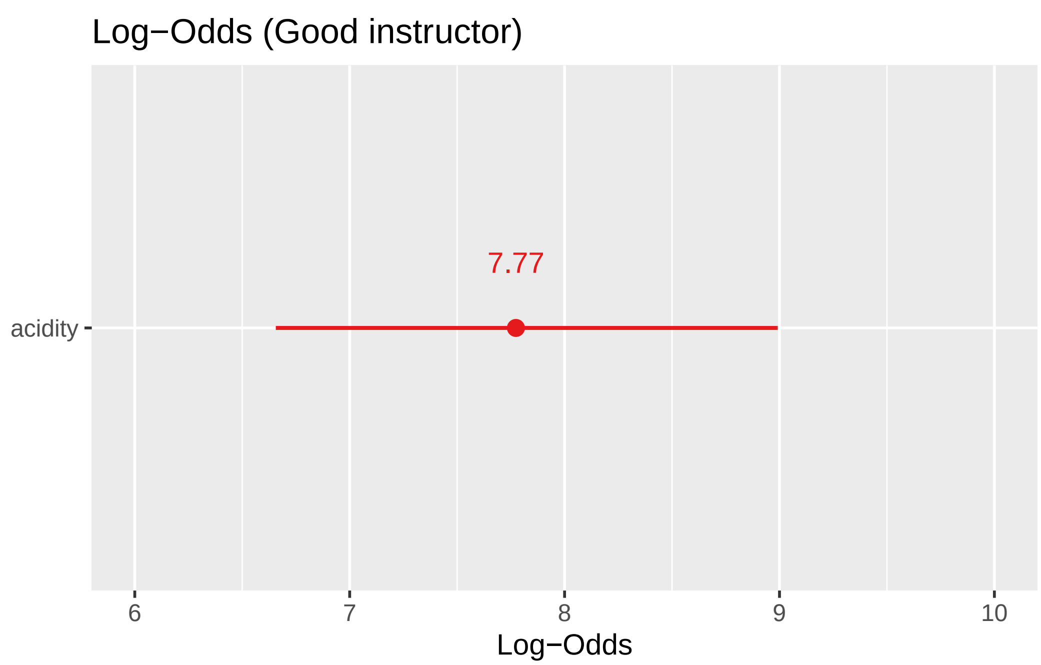
```
Call:
glm(formula = Qualityclass ~ acidity, family = binomial(link = "logit"),
    data = data_acidity)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -58.4727     4.4807  -13.05   <2e-16 ***
acidity       7.7736     0.5945   13.08   <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: 1003.5  on 725  degrees of freedom
Residual deviance:  675.4  on 724  degrees of freedom
```

```
AIC: 679.4

Number of Fisher Scoring iterations: 5
```

```r
# 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] "( 6.60847256990056 , 8.93873878119764 )"
```

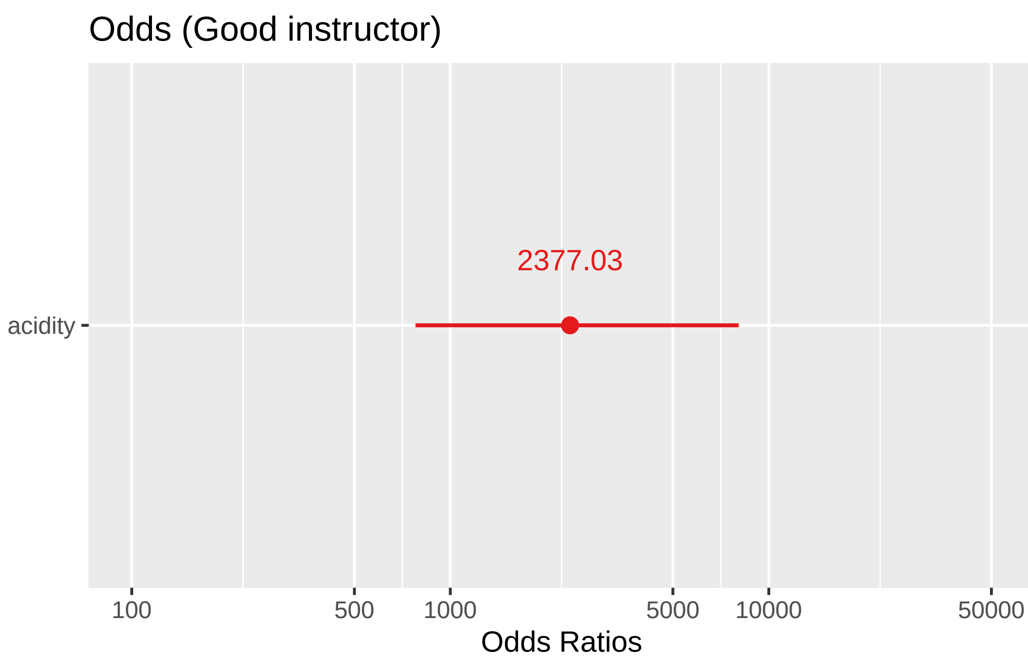
```r
# Plot log-odds of being a good instructor
plot_model(model3, show.values = TRUE, transform = NULL,
           title = "Log-Odds (Good instructor)", show.p = FALSE)
```



```
# Calculate lower and upper bounds for 'acidity' odds
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, ")")
```

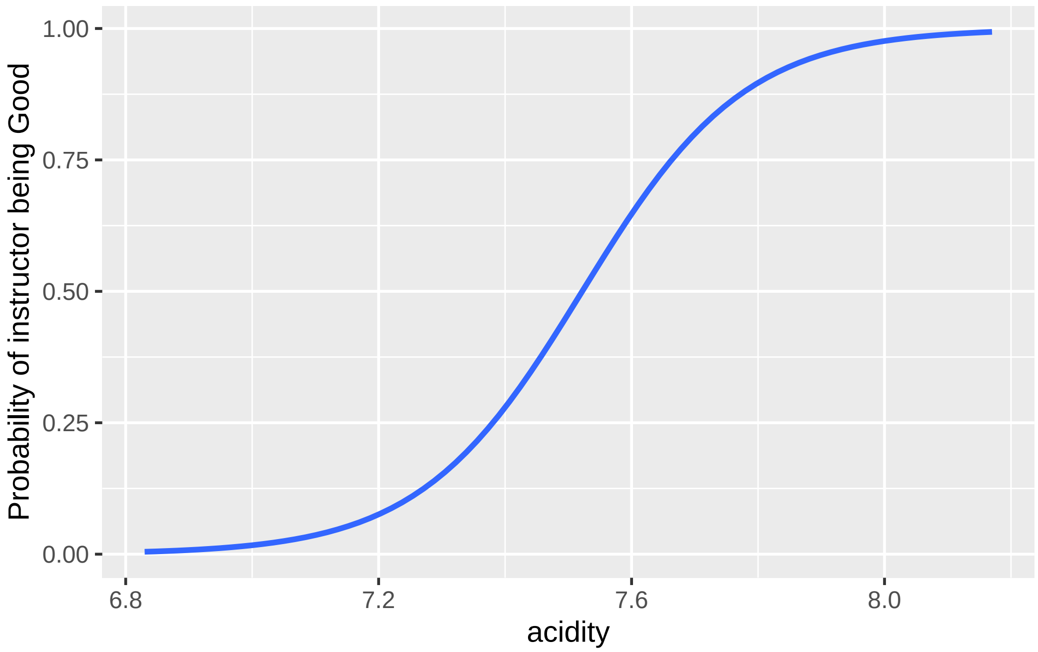
```
[1] "( 741.349793486999 , 7621.57851325419 )"
```

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



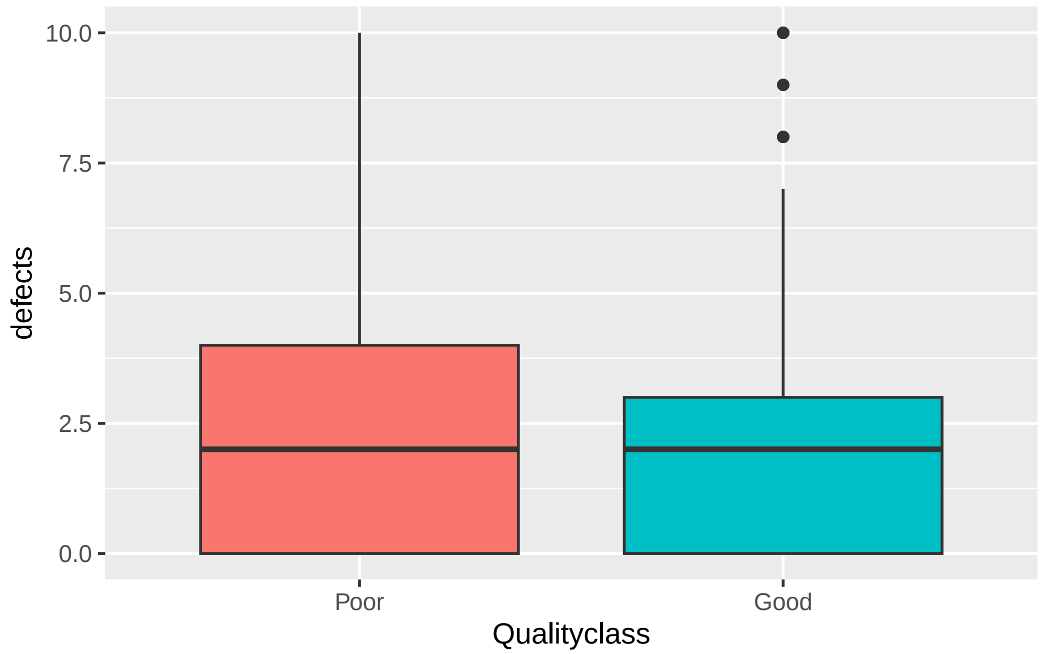

```
# Add predicted probabilities
data_acidity_after <- data_acidity %>%
                      mutate(logodds.Good = predict(model3, type = "response")) %>%
                      mutate(odds.Good = exp(logodds.Good)) %>%
                      mutate(probs.Good = fitted(model3))

# Plot the relationship between 'acidity' and probability of being a good instructor
ggplot(data = data_acidity_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")
```



3.6 Category 2 type defects and Qualityclass

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

# Create a boxplot of 'category_two_defects' across different 'Qualityclass' levels
p4 <- ggplot(data = data_defects, aes(x = Qualityclass, y = category_two_defects, fill = Q
  geom_boxplot() +
  labs(x = "Qualityclass", y = "defects")+
  theme(legend.position = "none")
p4
```




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

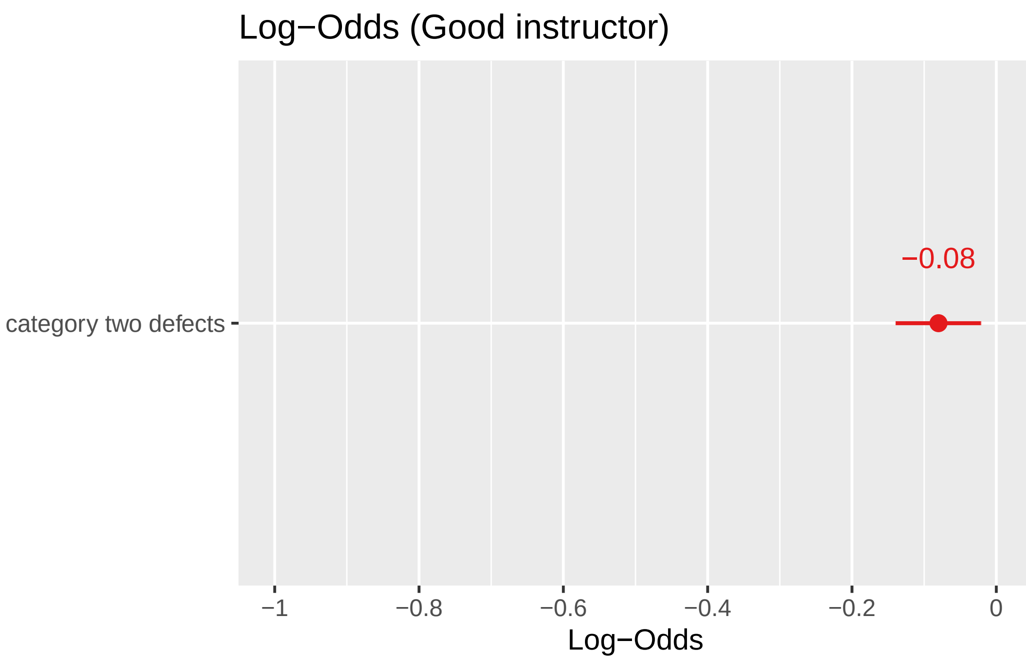
```
Call:
glm(formula = Qualityclass ~ category_two_defects, family = binomial(link = "logit"),
    data = data_defects)

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)           0.32616    0.10558   3.089  0.00201 **
category_two_defects -0.08010    0.02999  -2.671  0.00757 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1003.53  on 725  degrees of freedom
Residual deviance:  996.31  on 724  degrees of freedom
```

```
AIC: 1000.3

Number of Fisher Scoring iterations: 4
```

```
# 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] "( -0.138879080560977 , -0.0213191899815584 )"
```

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



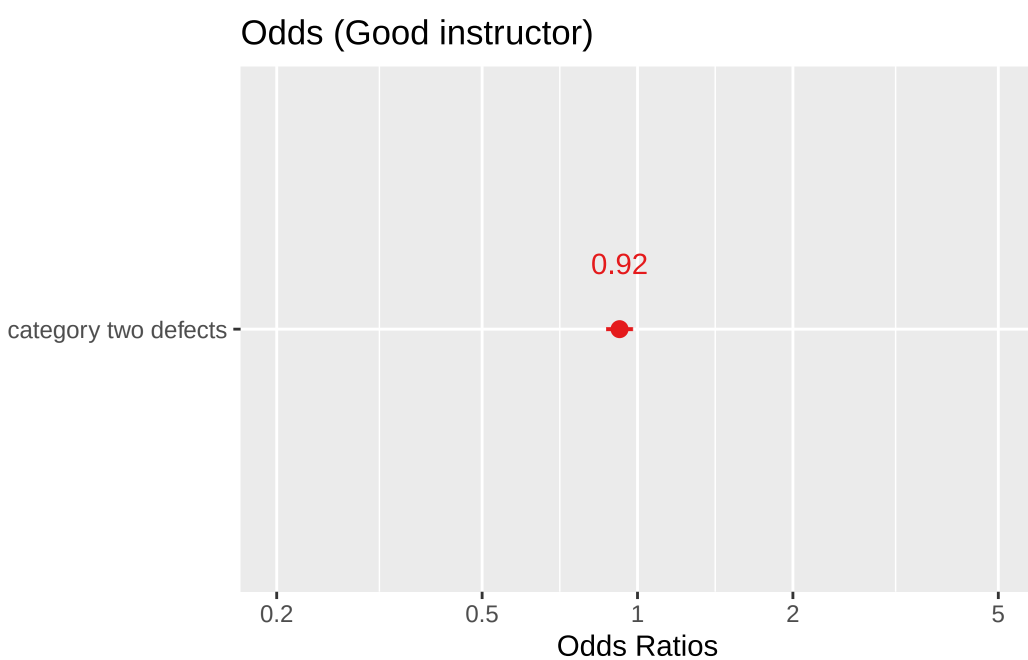
```r
# Calculate lower and upper bounds for 'category_two_defects' odds
exp(mod5.coef.logodds)
```

```
                     Estimate Std. Error     z value Pr(>|z|)
(Intercept)         1.3856408   1.111352 21.96245288 1.002008
category_two_defects 0.9230248   1.030444  0.06919116 1.007594
```

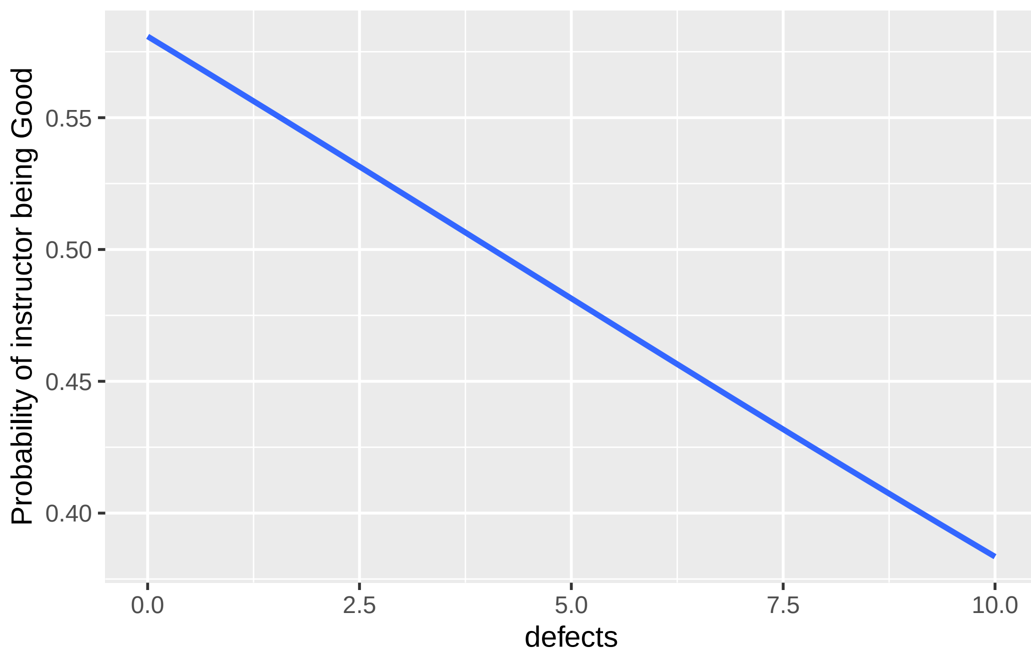
```r
defects.odds.lower <- exp(defects.logodds.lower)
defects.odds.upper <- exp(defects.logodds.upper)

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

```
[1] "( 0.870333262305556 , 0.978906457563423 )"
```

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



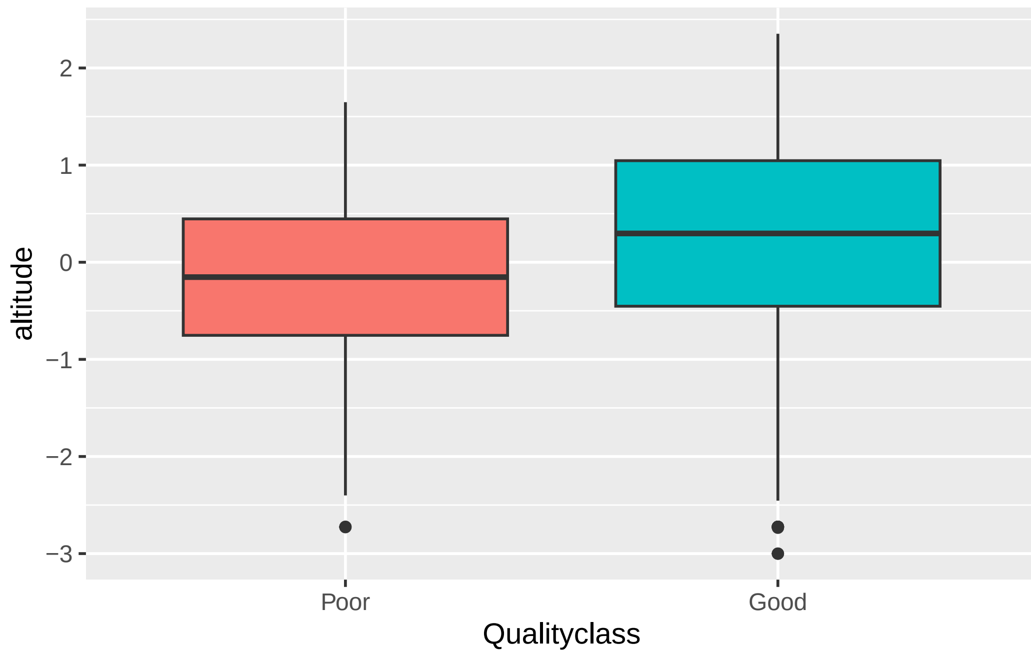
```r
# Add predicted probabilities
data_defects_after <- data_defects %>%
                  mutate(logodds.Good = predict(model5, type = "response")) %>%
                  mutate(odds.Good = exp(logodds.Good)) %>%
                  mutate(probs.Good = fitted(model5))

# Plot the relationship between 'category_two_defects' and probability of being a good ins
ggplot(data = data_defects_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")
```



3.7 Altitude mean meters and Qualityclass

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

# Create a boxplot of 'altitude_mean_meters' across different 'Qualityclass' levels
p5 <- ggplot(data = data_altitude, aes(x = Qualityclass, y = altitude_mean_meters, fill =
```

```r
  geom_boxplot() +
  labs(x = "Qualityclass", y = "altitude")+
  theme(legend.position = "none")
p5
```




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

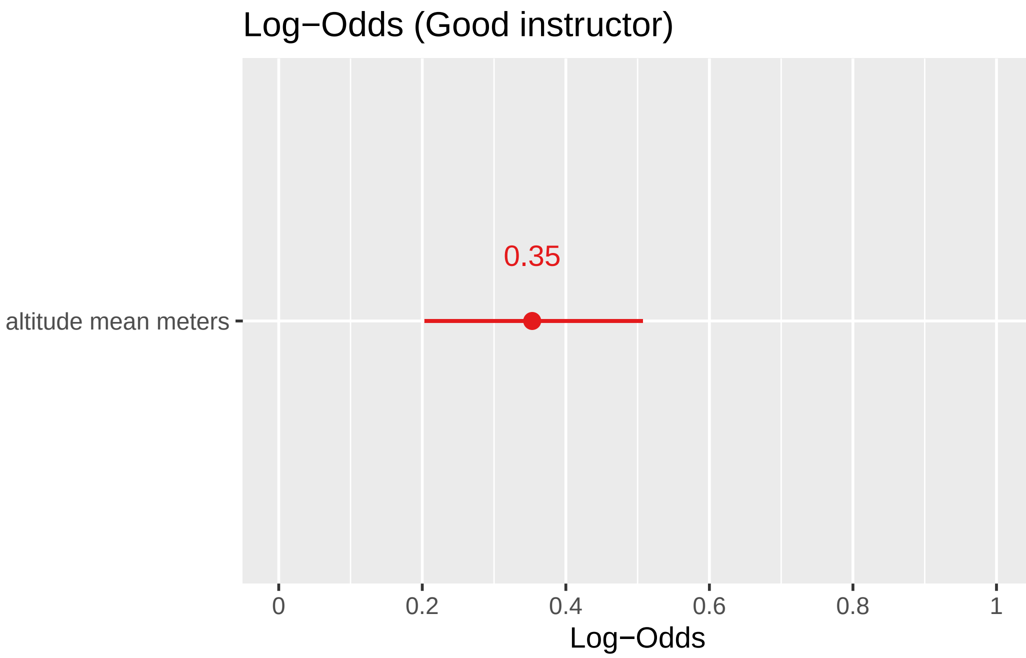
```
Call:
glm(formula = Qualityclass ~ altitude_mean_meters, family = binomial(link = "logit"),
    data = data_altitude)

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)            0.1292     0.0755   1.711    0.087 .
altitude_mean_meters   0.3531     0.0774   4.562 5.06e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

    Null deviance: 1003.53  on 725  degrees of freedom
Residual deviance:  981.86  on 724  degrees of freedom
AIC: 985.86

Number of Fisher Scoring iterations: 4
```

```r
# 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] "( 0.201429364953632 , 0.504856686546667 )"
```

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



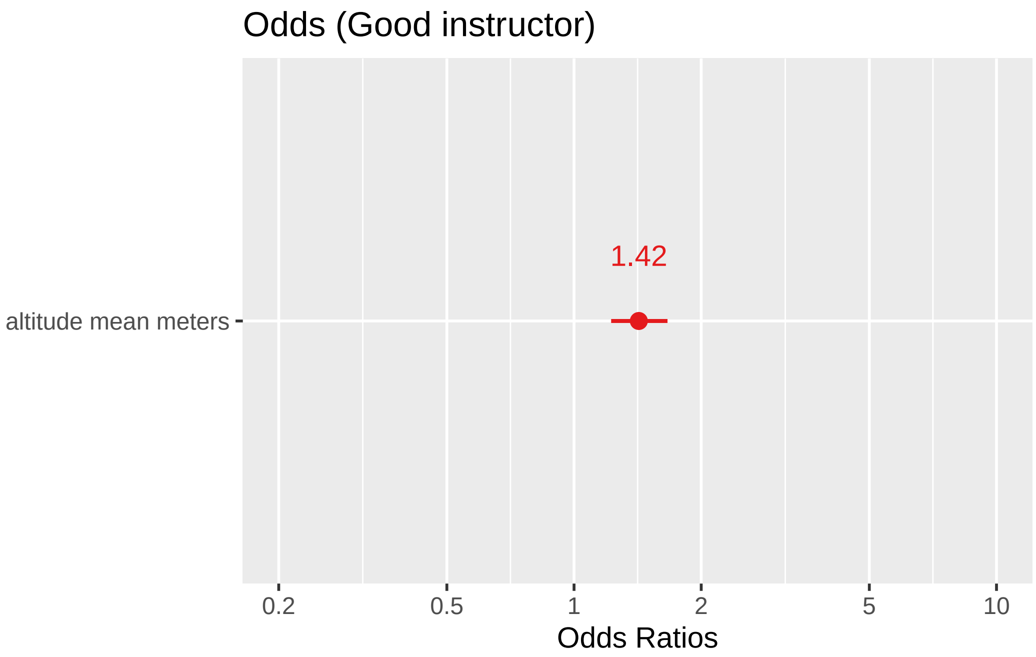

```
# Calculate lower and upper bounds for 'altitude_mean_meters' odds
exp(mod4.coef.logodds)
```

```
                     Estimate Std. Error   z value Pr(>|z|)
(Intercept)          1.137935   1.078426  5.536717 1.090904
altitude_mean_meters 1.423535   1.080480 95.801741 1.000005
```

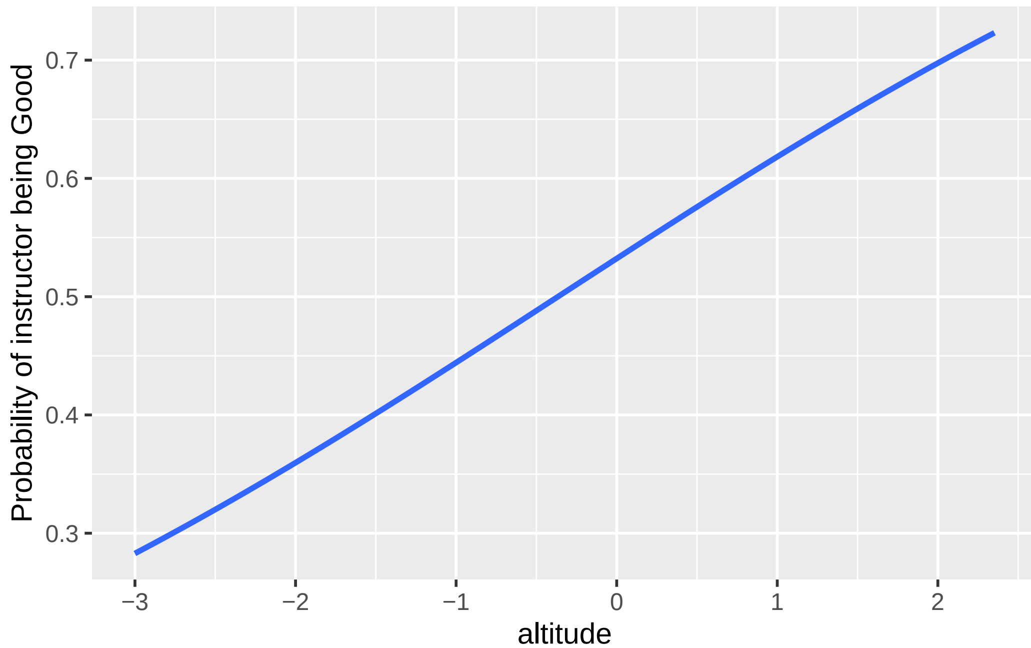
```
altitude.odds.lower <- exp(altitude.logodds.lower)
altitude.odds.upper <- exp(altitude.logodds.upper)

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

```
[1] "( 1.22314983676597 , 1.65674806915918 )"
```

```
# 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_after <- data_altitude %>%
                  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_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")
```

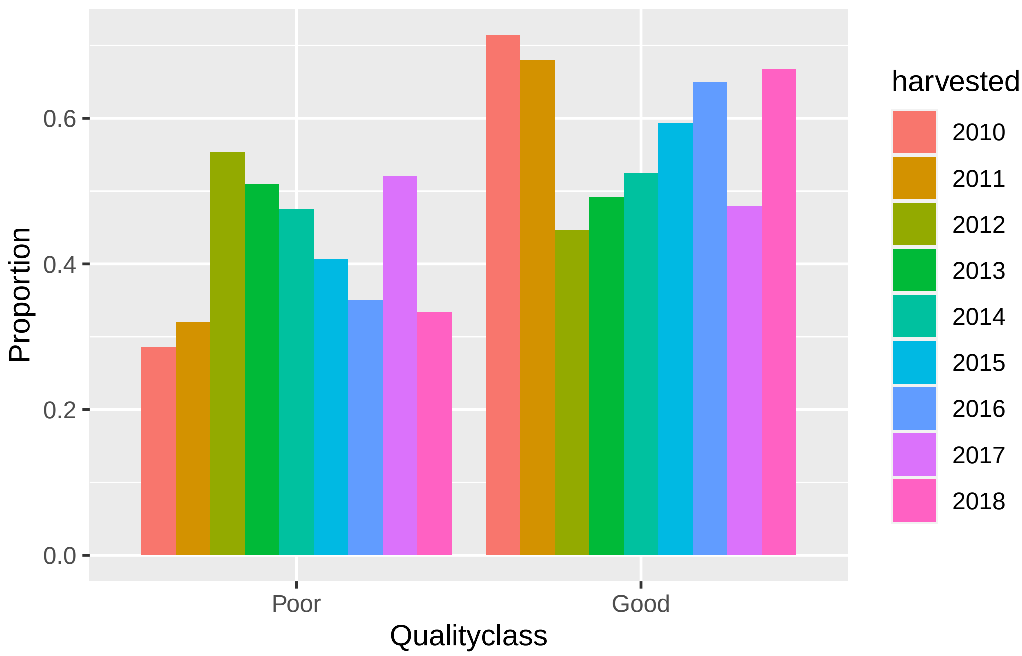


3.8 Harvested and Qualityclass

```
# Select 'harvested' and 'Qualityclass' columns and generate a contingency table.
data_harvested <- data %>%
          dplyr::select(harvested, Qualityclass)
data_harvested %>%
  tabyl(harvested, Qualityclass) %>%
  adorn_percentages() %>%
  adorn_pct_formatting() %>%
  adorn_ns()
```

```
harvested       Poor       Good
     2010 28.6%  (4) 71.4                                            (10)
     2011 32.0%  (8) 68.0                                            (17)
     2012 55.4% (98) 44.6                                            (79)
     2013 50.9% (57) 49.1                                            (55)
     2014 47.5% (77) 52.5                                            (85)
     2015 40.6% (39) 59.4                                            (57)
     2016 35.0% (28) 65.0                                            (52)
     2017 52.1% (25) 47.9                                            (23)
     2018 33.3%  (4) 66.7%  (8)
```

```r
# Create a barplot of 'harvested' across different 'Qualityclass' levels
p6 <- ggplot(data_harvested, aes(x = Qualityclass, y = after_stat(prop), group = harvested
    geom_bar(position = "dodge", stat = "count") +
    labs(y = "Proportion")
p6
```




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

```

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

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)     0.9163     0.5916   1.549   0.1214
harvested2011  -0.1625     0.7306  -0.222   0.8240
harvested2012  -1.1318     0.6106  -1.854   0.0638 .
```

```
harvested2013  -0.9520     0.6211  -1.533   0.1253
harvested2014  -0.8174     0.6122  -1.335   0.1818
harvested2015  -0.5368     0.6270  -0.856   0.3920
harvested2016  -0.2973     0.6364  -0.467   0.6404
harvested2017  -0.9997     0.6584  -1.518   0.1289
harvested2018  -0.2231     0.8515  -0.262   0.7933
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1003.53  on 725  degrees of freedom
Residual deviance:  985.86  on 717  degrees of freedom
AIC: 1003.9

Number of Fisher Scoring iterations: 4
```

```
# Extract coefficients from the model and calculate their confidence intervals.
model_harvested_coef_logodds <- model_harvested %>%
                                summary() %>%
                                coef()
model_harvested_coef_logodds
```

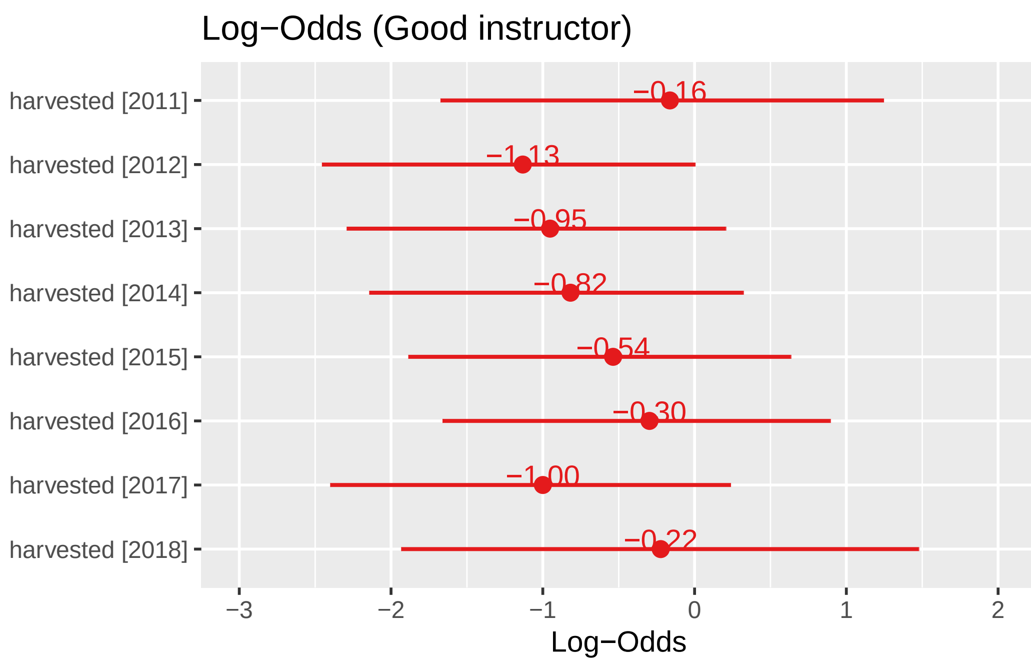
```
                Estimate Std. Error    z value   Pr(>|z|)
(Intercept)    0.9162907  0.5916076  1.5488150 0.12142620
harvested2011 -0.1625189  0.7306319 -0.2224361 0.82397439
harvested2012 -1.1318104  0.6106242 -1.8535303 0.06380639
harvested2013 -0.9520088  0.6210678 -1.5328581 0.12531083
harvested2014 -0.8174449  0.6121693 -1.3353248 0.18177005
harvested2015 -0.5368011  0.6270442 -0.8560818 0.39195256
harvested2016 -0.2972515  0.6363526 -0.4671177 0.64041570
harvested2017 -0.9996723  0.6583903 -1.5183582 0.12892412
harvested2018 -0.2231436  0.8514690 -0.2620689 0.79326831
```

```
confint_logodds <- confint(model_harvested)
confint_logodds
```

```
                   2.5 %      97.5 %
(Intercept)   -0.1788402 2.209833043
harvested2011 -1.6742313 1.245165574
```

```
harvested2012 -2.4551656 0.003857604
harvested2013 -2.2918766 0.205965175
harvested2014 -2.1431438 0.321691403
harvested2015 -1.8858074 0.634595032
harvested2016 -1.6606350 0.894930001
harvested2017 -2.4001925 0.236998318
harvested2018 -1.9324563 1.477106332
```

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




```r
# Transform the coefficients into odds ratios and obtain their confidence intervals
model_harvested_coef_odds <- model_harvested %>%
  summary() %>%
  coef() %>%
  exp()
model_harvested_coef_odds
```

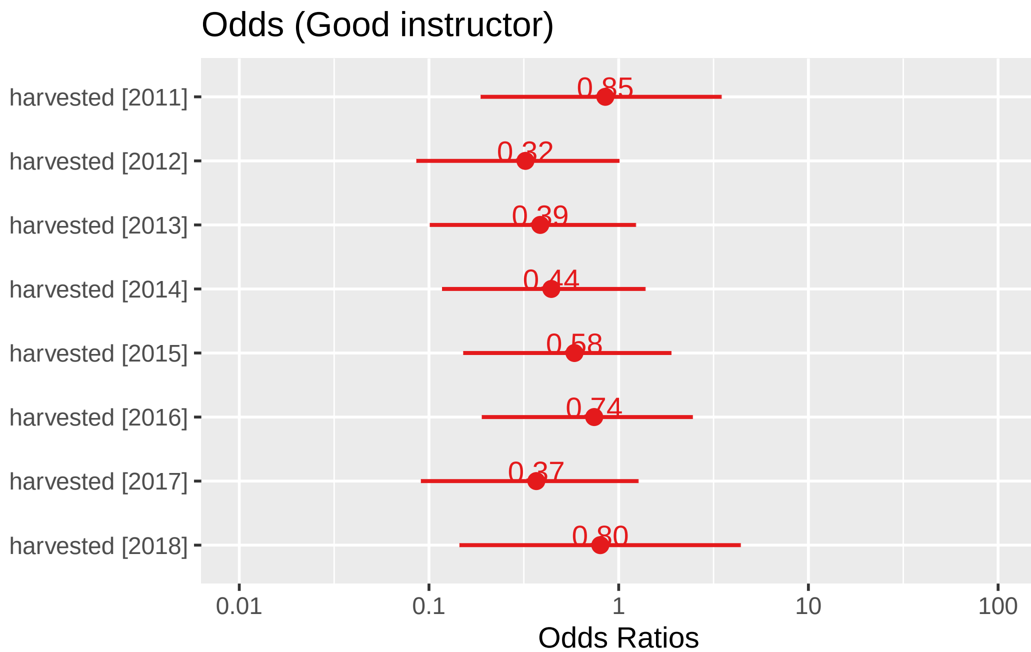
```
              Estimate Std. Error   z value Pr(>|z|)
(Intercept)  2.5000000   1.806891 4.7058905 1.129106
```

```
harvested2011 0.8500000   2.076392 0.8005661 2.279542
harvested2012 0.3224490   1.841580 0.1566831 1.065886
harvested2013 0.3859649   1.860914 0.2159177 1.133501
harvested2014 0.4415584   1.844428 0.2630727 1.199338
harvested2015 0.5846154   1.872069 0.4248234 1.479867
harvested2016 0.7428571   1.889576 0.6268063 1.897269
harvested2017 0.3680000   1.931680 0.2190713 1.137604
harvested2018 0.8000000   2.343086 0.7694580 2.210610
```

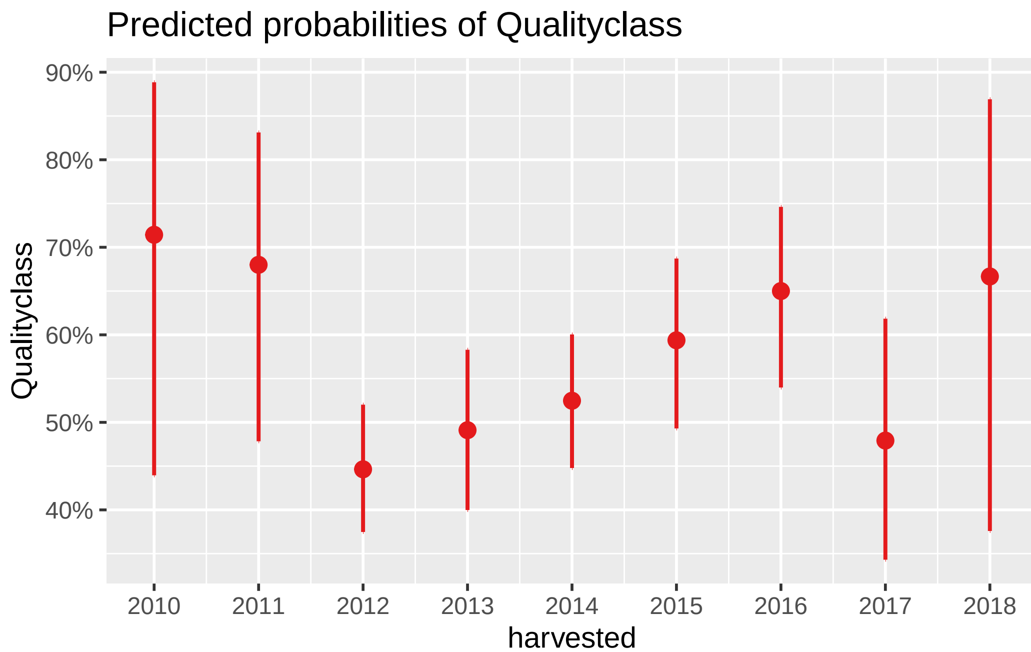
```
exp(confint_logodds)
```

```
                   2.5 %   97.5 %
(Intercept)   0.83623950 9.114195
harvested2011 0.18745223 3.473510
harvested2012 0.08584898 1.003865
harvested2013 0.10107661 1.228710
harvested2014 0.11728554 1.379459
harvested2015 0.15170652 1.886258
harvested2016 0.19001827 2.447164
harvested2017 0.09070049 1.267439
harvested2018 0.14479211 4.380252
```

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

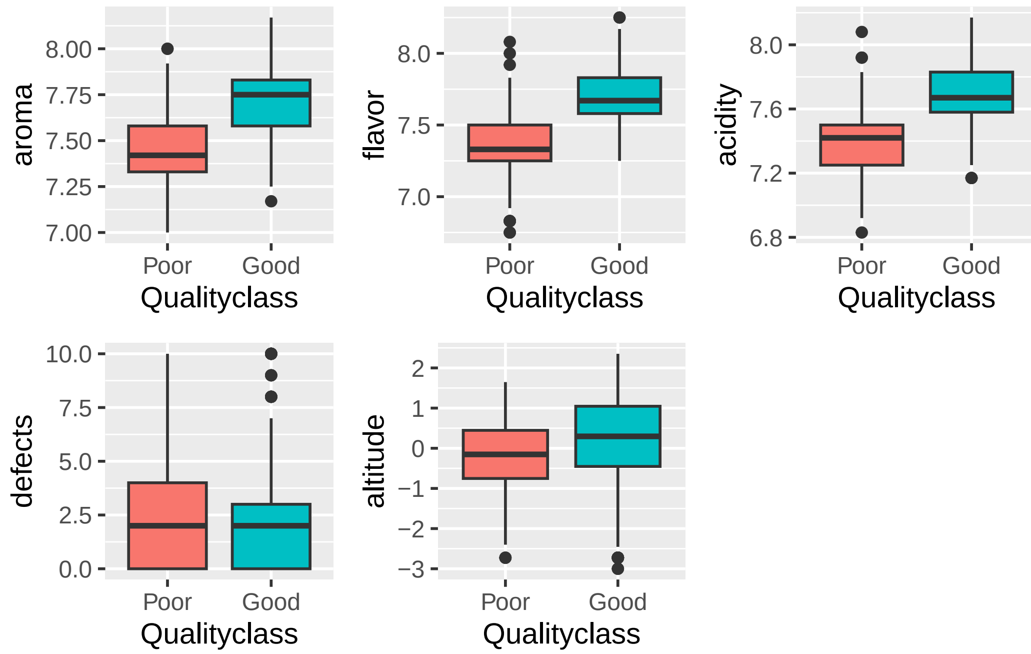



```
# Calculate log odds, odds, and probabilities and store them
data_harvested_after <- data_harvested %>%
                  mutate(logodds.Good = predict(model_harvested, type = "response")) %>%
                  mutate(odds.Good = exp(logodds.Good)) %>%
                  mutate(probs.Good = fitted(model_harvested))

# Generate a predictive plot
plot_model(model_harvested, type = "pred",
           terms = c("harvested"))
```

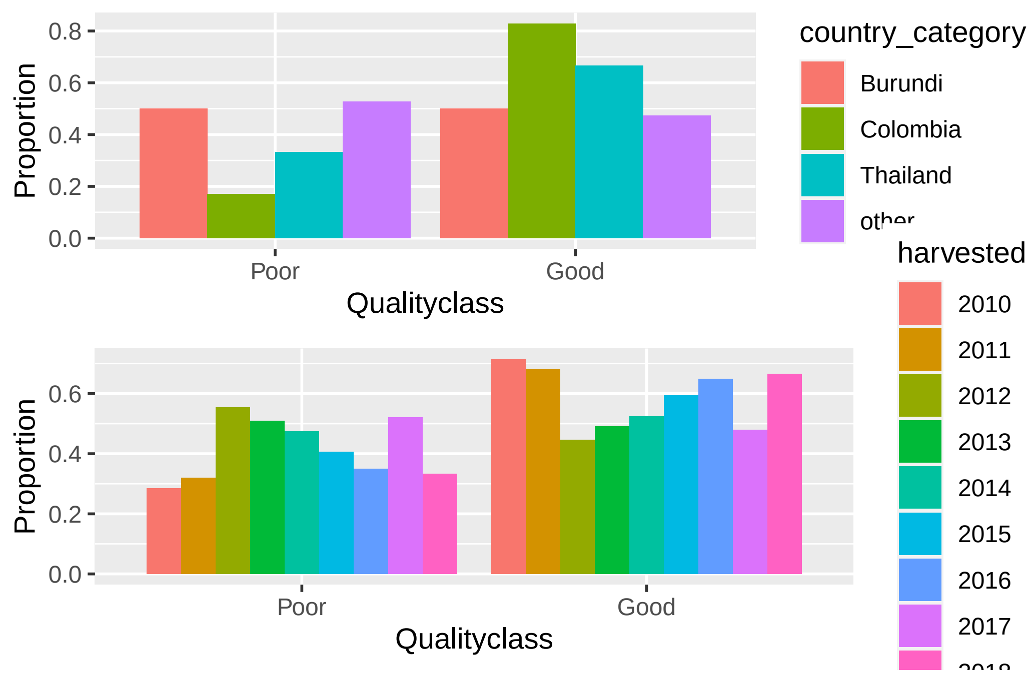


3.9 Plot Arrange

```
# Arrange multiple plots
grid.arrange(p1, p2, p3, p4, p5, ncol = 3)
```




```
grid.arrange(p0, p6)
```



4 Formal Analysis

4.1 Principal Component Analysis

Based on the correlation matrix, it is evident that some 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
# Principal principal component analysis (PCA) for 'aroma', 'flavor' and  'acidity'
data_pca <- data %>%
  dplyr::select(aroma, flavor, acidity, Qualityclass)
data_scaled <- scale(data_pca[, -4])
pca_result <- prcomp(data_scaled)
summary(pca_result)
```

```
Importance of components:
                          PC1    PC2     PC3
Standard deviation     1.5170 0.6790 0.48747
Proportion of Variance 0.7671 0.1537 0.07921
Cumulative Proportion  0.7671 0.9208 1.00000
```

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.

```r
# Predict PCA components and choose the first two components
pca_result_selected <- predict(pca_result, newdata = data_scaled)[, 1:2]

# Combine PCA components with other variables
data_pca_final <- data.frame(pca_result_selected, country_category = data$country_category

# Retrieve column names of the new data frame
names(data_pca_final)
```

```
[1] "PC1"                 "PC2"                 "country_category"
[4] "category_two_defects" "altitude_mean_meters" "harvested"
[7] "Qualityclass"
```

4.2 Model Selection

```
# Conduct an origin model
model_full_after <- glm(Qualityclass ~ country_category + aroma + flavor + acidity + categ

# Summarize the model
model_full_after %>%
  summary()
```

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

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)              -136.36544   11.87262 -11.486  < 2e-16 ***
country_categoryColombia    0.19714    3.83740   0.051   0.9590
country_categoryThailand    0.38580    3.91529   0.099   0.9215
country_categoryother      -1.98656    3.82843  -0.519   0.6038
aroma                       5.60251    0.89564   6.255 3.97e-10 ***
flavor                      7.41691    1.02128   7.262 3.80e-13 ***
acidity                     5.13315    0.84515   6.074 1.25e-09 ***
category_two_defects        0.08151    0.05280   1.544   0.1227
altitude_mean_meters        0.23102    0.14289   1.617   0.1059
harvested2011              -0.09750    1.08274  -0.090   0.9282
harvested2012               0.07982    0.91032   0.088   0.9301
harvested2013               0.22982    0.90818   0.253   0.8002
harvested2014               0.94042    0.92551   1.016   0.3096
harvested2015               0.68207    0.93880   0.727   0.4675
harvested2016               1.76568    0.98341   1.795   0.0726 .
harvested2017               1.23785    0.98308   1.259   0.2080
harvested2018               2.65646    1.30031   2.043   0.0411 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1003.53  on 725  degrees of freedom
Residual deviance:  396.36  on 709  degrees of freedom
AIC: 430.36
```

```
Number of Fisher Scoring iterations: 7
```

```r
# Perform stepwise variable selection using AIC
stepAIC(model_full_after)
```

```
Start:  AIC=430.36
Qualityclass ~ country_category + aroma + flavor + acidity + 
    category_two_defects + altitude_mean_meters + harvested

                       Df Deviance    AIC
<none>                      396.36 430.36
- category_two_defects  1   398.77 430.77
- altitude_mean_meters  1   398.97 430.97
- harvested             8   419.15 437.15
- country_category      3   435.09 463.09
- acidity               1   438.67 470.67
- aroma                 1   443.99 475.99
- flavor                1   462.60 494.60


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

Coefficients:
             (Intercept)  country_categoryColombia  country_categoryThailand  
              -136.36544                   0.19714                   0.38580  
   country_categoryother                     aroma                    flavor  
                -1.98656                   5.60251                   7.41691  
                 acidity      category_two_defects      altitude_mean_meters  
                 5.13315                   0.08151                   0.23102  
           harvested2011             harvested2012             harvested2013  
                -0.09750                   0.07982                   0.22982  
           harvested2014             harvested2015             harvested2016  
                 0.94042                   0.68207                   1.76568  
           harvested2017             harvested2018  
                 1.23785                   2.65646  

Degrees of Freedom: 725 Total (i.e. Null);  709 Residual
Null Deviance:      1004 
Residual Deviance: 396.4    AIC: 430.4
```

```r
# Fit logistic regression model with PCA components
pca_model <- glm(Qualityclass ~ ., data = data_pca_final, family = binomial(link = "logit"

# Summarize the model
pca_model %>%
  summary()
```

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

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                1.03127    3.89993   0.264   0.7914
PC1                        0.64526    0.05280  12.222   <2e-16 ***
PC2                       -0.01683    0.04837  -0.348   0.7278
country_categoryColombia   0.13759    3.81016   0.036   0.9712
country_categoryThailand   0.27024    3.88866   0.069   0.9446
country_categoryother     -2.13272    3.80058  -0.561   0.5747
category_two_defects       0.08881    0.05245   1.693   0.0904 .
altitude_mean_meters       0.18852    0.14058   1.341   0.1799
harvested2011             -0.16663    1.08668  -0.153   0.8781
harvested2012              0.07030    0.91843   0.077   0.9390
harvested2013              0.12648    0.91488   0.138   0.8900
harvested2014              0.91531    0.93415   0.980   0.3272
harvested2015              0.69761    0.94612   0.737   0.4609
harvested2016              1.75244    0.99220   1.766   0.0774 .
harvested2017              1.28878    0.98794   1.305   0.1921
harvested2018              2.48395    1.29643   1.916   0.0554 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1003.53  on 725  degrees of freedom
Residual deviance:  399.57  on 710  degrees of freedom
AIC: 431.57

Number of Fisher Scoring iterations: 7
```

```r
pca_model_summary <- glance(pca_model)
kable(pca_model_summary, digits = 2)
```

null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual	nobs
1003.53	725	-199.79	431.57	504.97	399.57	710	726

```r
# Perform stepwise variable selection using AIC
stepAIC(pca_model)
```

```
Start:  AIC=431.57
Qualityclass ~ PC1 + PC2 + country_category + category_two_defects +
    altitude_mean_meters + harvested

                       Df Deviance    AIC
- PC2                   1   399.69 429.69
- altitude_mean_meters  1   401.37 431.37
<none>                      399.57 431.57
- category_two_defects  1   402.48 432.48
- harvested             8   422.90 438.90
- country_category      3   441.10 467.10
- PC1                   1   915.48 945.48

Step:  AIC=429.69
Qualityclass ~ PC1 + country_category + category_two_defects +
    altitude_mean_meters + harvested

                       Df Deviance    AIC
- altitude_mean_meters  1   401.55 429.55
<none>                      399.69 429.69
- category_two_defects  1   402.65 430.65
- harvested             8   423.99 437.99
- country_category      3   441.13 465.13
- PC1                   1   916.55 944.55

Step:  AIC=429.55
Qualityclass ~ PC1 + country_category + category_two_defects +
    harvested

                       Df Deviance    AIC
<none>                      401.55 429.55
```

```
- category_two_defects  1   404.86 430.86
- harvested             8   424.94 436.94
- country_category      3   449.38 471.38
- PC1                   1   931.59 957.59
```

```
Call:  glm(formula = Qualityclass ~ PC1 + country_category + category_two_defects +
    harvested, family = binomial(link = "logit"), data = data_pca_final)

Coefficients:
             (Intercept)                       PC1  country_categoryColombia
                 1.39357                   0.64662                  -0.09348
country_categoryThailand     country_categoryother      category_two_defects
                -0.14086                  -2.48718                   0.09380
           harvested2011             harvested2012             harvested2013
                -0.19451                  -0.01981                   0.12008
           harvested2014             harvested2015             harvested2016
                 0.88638                   0.65190                   1.63107
           harvested2017             harvested2018
                 1.35077                   2.30535

Degrees of Freedom: 725 Total (i.e. Null);  712 Residual
Null Deviance:      1004
Residual Deviance: 401.6    AIC: 429.6
```

After reducing dimensionality using PCA, we selected the model with the lowest AIC, which is considered the optimal model.

```
# Final Logistic Regression Model for Qualityclass Prediction
optimal_model <- glm(Qualityclass ~ PC1 + country_category + category_two_defects + harves
optimal_model %>%
  summary()
```

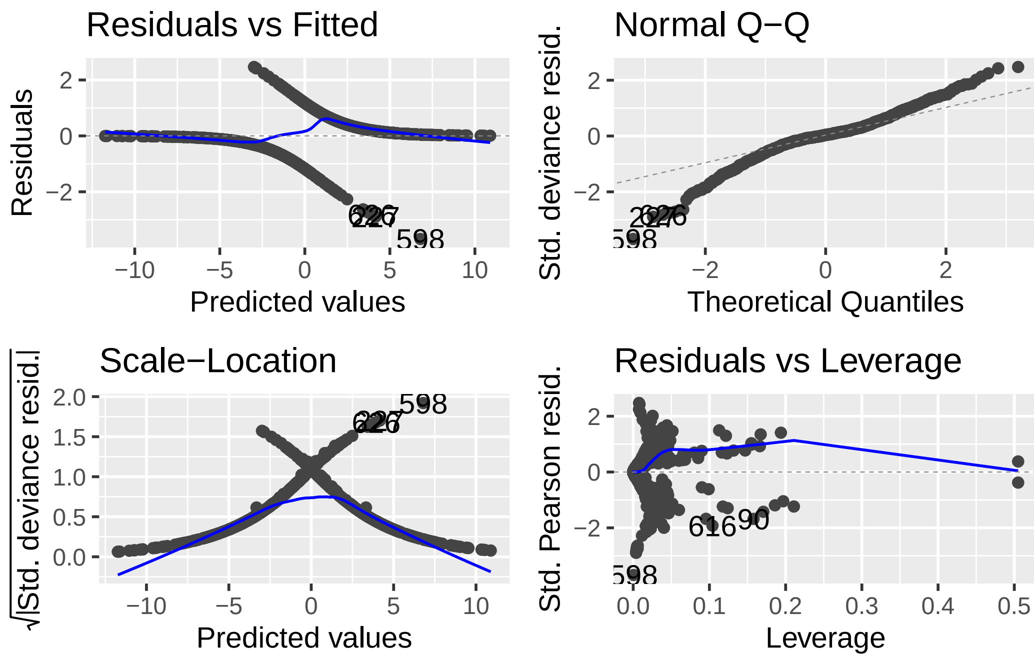
```
Call:
glm(formula = Qualityclass ~ PC1 + country_category + category_two_defects +
    harvested, family = binomial(link = "logit"), data = data_pca_final)

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)               1.39357    3.96280   0.352   0.7251
```

```
PC1                         0.64662    0.05276  12.256   <2e-16 ***
country_categoryColombia -0.09348    3.88231  -0.024   0.9808
country_categoryThailand -0.14086    3.94836  -0.036   0.9715
country_categoryother    -2.48718    3.86848  -0.643   0.5203
category_two_defects      0.09380    0.05198   1.804   0.0712 .
harvested2011            -0.19451    1.07478  -0.181   0.8564
harvested2012            -0.01981    0.91171  -0.022   0.9827
harvested2013             0.12008    0.91107   0.132   0.8951
harvested2014             0.88638    0.93212   0.951   0.3416
harvested2015             0.65190    0.94433   0.690   0.4900
harvested2016             1.63107    0.97215   1.678   0.0934 .
harvested2017             1.35077    0.98144   1.376   0.1687
harvested2018             2.30535    1.27791   1.804   0.0712 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1003.53  on 725  degrees of freedom
Residual deviance:  401.55  on 712  degrees of freedom
AIC: 429.55

Number of Fisher Scoring iterations: 7
```

```r
optimal_model_summary <- glance(optimal_model)
kable(optimal_model_summary,digits =2)
```

null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual	nobs
1003.53	725	-200.78	429.55	493.78	401.55	712	726

```r
# Check the assumptions
autoplot(optimal_model)
```



$$\text{Qualityclass} = \beta_0 + \beta_1 \times \text{PC1} + \beta_2 \times \text{country\_category} + \beta_3 \times \text{category\_two\_defects} + \beta_4 \times \text{harvested} + \epsilon$$

- $Qualityclass$ is the response variable
- $PC1$ is a variable derived from reducing the dimensions of $aroma$, $flavor$, and $acidity$
- $ountry\_category$, $category\_two\_defects$, and $harvested$ are the predictor variables
- β_0 to β_4 are the coefficients of the model
- ϵ is the error term