Predicting Customer Detractors (Part 1). Analyzing Contextual Factors Via Logistic Regression

1. Introduction

This case study explores opportunities to improve customer service, focusing on increasing the likelihood that customers recommend the company.

The project was conducted in two phases:

1. **Exploratory analysis** to identify areas needing improvement, based on internal data analysis and logistic regression modeling. This is the phase represented in this document.
2. **Identifying potential improvements and modeling their impact**, using text analysis and simulation to evaluate how addressing pain points could enhance customer perceptions. Due to length constraints, this second part is presented in a separate document: "Predicting Customer Detractors (Part 2): Assessing Improvement Opportunities via Text Analysis."

Although based on a real-world project, all data, variables, and insights presented here have been simulated to maintain confidentiality.

The analysis includes:

- Data simulation and cleaning
- Visualization techniques (e.g., heatmaps for high-dimensional data)
- Descriptive statistics
- Regression modeling evaluations (linear, ordinal logistic, and binomial logistic)
- Simulation-based recommendations
- Creation of reusable functions to automate procedures

2. Setup

We start by loading the required packages for data manipulation, visualization, modeling, and exporting results.

```
# Data handling
library(readxl)       # Read Excel files
library(openxlsx)     # Write Excel files
library(dplyr)        # Data manipulation
```

```
##
## Adjuntando el paquete: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)       # Data reshaping (long/wide formats)

# Visualization
library(ggplot2)     # General plotting
library(coefplot)    # Visualize model coefficients
```

```
## Warning: package 'coefplot' was built under R version 4.5.1
```

```
library(vcd)         # Visualizing categorical data
```

```
## Warning: package 'vcd' was built under R version 4.5.1
```

```
## Cargando paquete requerido: grid
```

```
# Statistical analysis
library(psych)       # Descriptive statistics
```

```
##
## Adjuntando el paquete: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(ordinal)     # Ordinal logistic regression
```

```
##
## Adjuntando el paquete: 'ordinal'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(VGAM)        # Alternative ordinal modeling
```

```
## Warning: package 'VGAM' was built under R version 4.5.1
```

```
## Cargando paquete requerido: stats4
```

```
## Cargando paquete requerido: splines
```

```
##
## Adjuntando el paquete: 'VGAM'
```

```
## The following objects are masked from 'package:ordinal':
##
##     dgumbel, dlgamma, pgumbel, plgamma, qgumbel, rgumbel, wine

## The following objects are masked from 'package:psych':
##
##     fisherz, logistic, logit
```

```r
library(car)          # VIF and regression diagnostics
```

```
## Cargando paquete requerido: carData

##
## Adjuntando el paquete: 'car'

## The following object is masked from 'package:VGAM':
##
##     logit

## The following object is masked from 'package:psych':
##
##     logit

## The following object is masked from 'package:dplyr':
##
##     recode
```

3. Data Simulation

3.1. Simulating Demographic and Contextual Variables

To create a realistic dataset, we simulate basic **demographic** information (e.g., age, gender) and **contextual variables** related to the customer service experience: country, contact method, and reason for contacting.

```r
# Set seed for reproducibility
set.seed(1000)

# Define dataset size
data.n <- 40000  # Large enough to allow subgroup analysis despite unbalanced category probabilities

# Initialize dataframe with IDs
data <- data.frame(id = factor(1:data.n))

# Simulate age with a log-normal distribution, capped between 15 and 90
data$age <- round(rlnorm(n = data.n, meanlog = log(40), sdlog = log(1.4)))
data$age[data$age > 90] <- 90
data$age[data$age < 15] <- 15

# Simulate gender
data$gender <- factor(sample(c("man", "woman"),
```

```
                              prob = c(0.5, 0.5),
                              replace = TRUE, size = data.n))

# Simulate reason for contacting
data$reason <- factor(sample(c("incorrect_item", "delay", "status", "cancellation", "other"),
                              prob = c(0.20, 0.30, 0.20, 0.10, 0.20),
                              replace = TRUE, size = data.n))

# Simulate contact method
data$method <- factor(sample(c("chat", "web_form", "email", "phone"),
                              prob = c(0.40, 0.10, 0.15, 0.35),
                              replace = TRUE, size = data.n))

# Simulate country (two fictional countries for anonymity)
data$country <- factor(sample(c("Drinkland", "Eatland"),
                               prob = c(0.5, 0.5),
                               replace = TRUE, size = data.n))
```

3.2. Simulation of Customer Complaints

We simulate five types of customer complaints: connection issues, slow response, slow resolution, repetition of information, and other unspecified complaints. These are conditioned on demographic and contextual variables as follows:

- **Connection issues**: more likely in Drinkland
- **Slow response**: more frequent in email, and moderately in chat/web_form
- **Slow resolution**: more frequent when "incorrect_item" is handled via chat
- **Repetition**: more likely when the contact reason is "cancellation"
- **Other complaints**: uniformly distributed

```
## Complaints about connection issues
# Base probability of a connection complaint: 10%
prob_connection <- rep(0.10, data.n)
# In Drinkland, probability is doubled
prob_connection[data$country == "Drinkland"] <- prob_connection[data$country == "Drinkland"] * 2
# Ensure probability values remain within [0, 1]
prob_connection [prob_connection > 1] <- 1
prob_connection [prob_connection < 0] <- 0
# Generate binary variable based on probabilities
data$complain.connect <- rbinom(n = data.n, size = 1, prob = prob_connection)
# Check distribution across countries
with(data, prop.table(table(complain.connect, country), margin = 2))
```

```
##                 country
## complain.connect Drinkland   Eatland
##                0 0.7981113 0.8995473
##                1 0.2018887 0.1004527
```

```
## Complaints about slow response
# Base probability of a slow response complaint: 5%
prob_slow_response <- rep(0.05, data.n)
```

```r
# Email contacts: 6x higher likelihood
prob_slow_response[data$method == "email"] <- prob_slow_response[data$method == "email"] * 6
# Chat and web_form: 3x higher likelihood
prob_slow_response[data$method %in% c("chat", "web_form")] <- prob_slow_response[data$method %in% c("cha
# Ensure probability values remain within [0, 1]
prob_slow_response [prob_slow_response > 1] <- 1
prob_slow_response [prob_slow_response < 0] <- 0
# Generate binary variable
data$complain.sp.respond <- rbinom(n = data.n, size = 1, prob = prob_slow_response)
# Check distribution across contact methods
with(data, prop.table(table(complain.sp.respond, method), margin = 2))
```

```
##                    method
## complain.sp.respond      chat     email     phone  web_form
##                   0 0.8513143 0.6948590 0.9489453 0.8577970
##                   1 0.1486857 0.3051410 0.0510547 0.1422030
```

```r
## Complaints about slow resolution
# Base probability of a slow resolution complaint: 10%
prob_slow_resolution <- rep(0.10, data.n)
# If contact is via chat and reason is incorrect_item → 5x higher likelihood
is_chat_incorrect <- data$method == "chat" & data$reason == "incorrect_item"
prob_slow_resolution[is_chat_incorrect] <- prob_slow_resolution[is_chat_incorrect] * 5
# Ensure probability values remain within [0, 1]
prob_slow_resolution [prob_slow_resolution > 1] <- 1
prob_slow_resolution [prob_slow_resolution < 0] <- 0
# Generate binary variable
data$complain.sp.solve <- rbinom(n = data.n, size = 1, prob = prob_slow_resolution)
# Check distribution by reason and method
with(data, prop.table(table(complain.sp.solve, reason, method), margin = c(2, 3)))
```

```
## , , method = chat
##
##                    reason
## complain.sp.solve cancellation      delay incorrect_item      other     status
##                 0   0.90458716 0.89556300     0.50478838 0.87899687 0.90385812
##                 1   0.09541284 0.10443700     0.49521162 0.12100313 0.09614188
##
## , , method = email
##
##                    reason
## complain.sp.solve cancellation      delay incorrect_item      other     status
##                 0   0.91840278 0.90481400     0.90909091 0.90033784 0.88842975
##                 1   0.08159722 0.09518600     0.09090909 0.09966216 0.11157025
##
## , , method = phone
##
##                    reason
## complain.sp.solve cancellation      delay incorrect_item      other     status
##                 0   0.90989660 0.90199856     0.90151249 0.90321441 0.89479315
##                 1   0.09010340 0.09800144     0.09848751 0.09678559 0.10520685
##
## , , method = web_form
```

```
##
##                   reason
## complain.sp.solve cancellation      delay incorrect_item      other     status
##                 0   0.88755981 0.89090909     0.88353414 0.90306748 0.90326633
##                 1   0.11244019 0.10909091     0.11646586 0.09693252 0.09673367
```

```r
## Complaints about having to repeat information
# Base probability of a repetition complaint: 2%
prob_repeat_info <- rep(0.02, data.n)
# If contact reason is cancellation → 10x higher likelihood
prob_repeat_info[data$reason == "cancellation"] <- prob_repeat_info[data$reason == "cancellation"] * 10
# Ensure probability values remain within [0, 1]
prob_repeat_info [prob_repeat_info > 1] <- 1
prob_repeat_info [prob_repeat_info < 0] <- 0
# Generate binary variable
data$complain.repeat <- rbinom(n = data.n, size = 1, prob = prob_repeat_info)
# Check distribution across contact reasons
with(data, prop.table(table(complain.repeat, reason), margin = 2))
```

```
##                 reason
## complain.repeat cancellation      delay incorrect_item      other     status
##               0   0.79010796 0.97868054     0.97952600 0.98004988 0.98118146
##               1   0.20989204 0.02131946     0.02047400 0.01995012 0.01881854
```

```r
## Other types of complaints (uniform distribution)
# Constant probability of 20% for other unspecified complaints
data$complain.other <- rbinom(n = data.n, size = 1, prob = 0.20)
# Check distribution
with(data, prop.table(table(complain.other)))
```

```
## complain.other
##       0       1
## 0.79655 0.20345
```

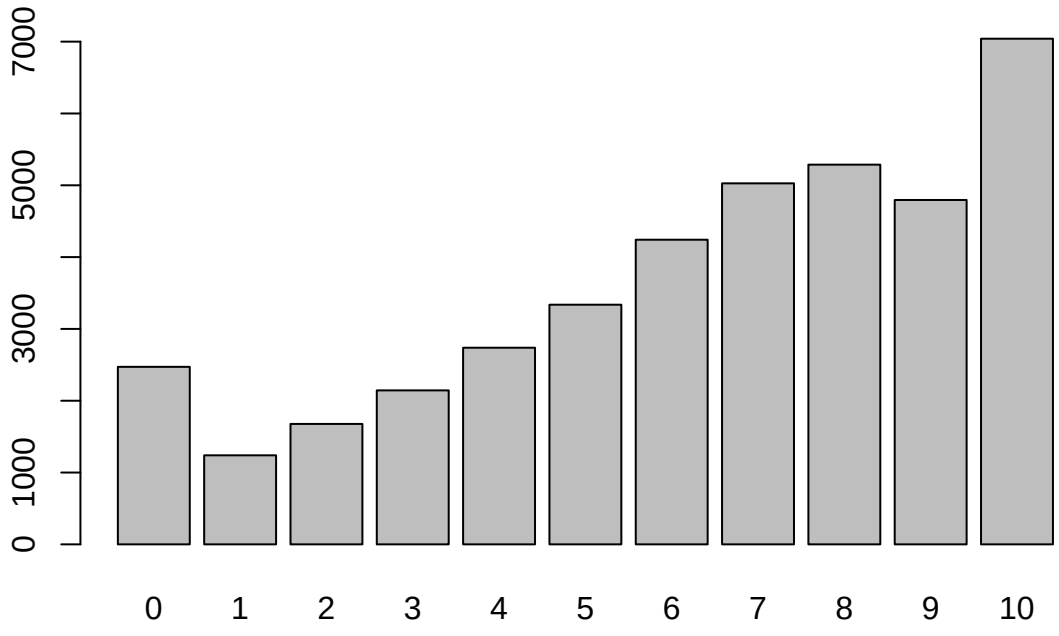
3.3. Simulation of NPS scores

We simulate NPS scores by first assuming a normally distributed base score in the ideal case where customers have no complaints.

Then, we apply fixed score penalties based on the presence of different types of complaints.
Finally, we constrain scores to the valid NPS range (0–10), convert them to integers, and add placeholders for open-text responses.

```r
## Simulation of NPS scores

# Step 1: Simulate base NPS scores assuming no complaints (mean = 9, sd = 2)
data$nps.score <- rnorm(data.n, mean = 9, sd = 2)

# Step 2: Apply fixed penalties based on complaints
# -3 for connection issues
# -4 for slow response
# -6 for slow resolution
```

```r
# -2 for repeated information
# -1 for other complaints
data$nps.score <- data$nps.score -
  3 * data$complain.connect -
  4 * data$complain.sp.respond -
  6 * data$complain.sp.solve -
  2 * data$complain.repeat -
  1 * data$complain.other

# Step 3: Ensure scores stay within the 0-10 range
data$nps.score[data$nps.score < 0] <- 0
data$nps.score[data$nps.score > 10] <- 10

# Step 4: Convert to integer values (using floor)
data$nps.score <- floor(data$nps.score)

# Step 5: Add placeholder for open-text comments
data$open.comment <- rep("bla bla", data.n)

# Step 6: Visualize score distribution
barplot(table(data$nps.score))
```




```r
with(data, prop.table(table(data$nps.score)))
```

```
##
```

```
##        0        1        2        3        4        5        6        7
## 0.061800 0.031000 0.041925 0.053600 0.068450 0.083425 0.106050 0.125675
##        8        9       10
## 0.132175 0.119875 0.176025
```

3.3.1. Specifying the NPS segments based on NPS scores

Based on standard NPS segmentation, we classify customers into three categories: - **Detractors**: scores from 0 to 6 - **Passives**: scores of 7 or 8 - **Promoters**: scores of 9 or 10

We create a new categorical variable to reflect these groupings.

```
## Create NPS segments based on score values

# Step 1: Initialize empty character variable
data$nps.segment <- rep(NA_character_, data.n)

# Step 2: Assign segment based on score range
data$nps.segment[data$nps.score <= 6] <- "detractor"
data$nps.segment[data$nps.score > 6 & data$nps.score < 9] <- "passive"
data$nps.segment[data$nps.score >= 9] <- "promoter"

# Step 3: Convert to factor for categorical analysis
data$nps.segment <- factor(data$nps.segment, levels = c("detractor", "passive", "promoter"))

# Step 4: Check distribution across NPS segments
with(data, prop.table(table(nps.segment)))
```

```
## nps.segment
## detractor   passive  promoter
##   0.44625   0.25785   0.29590
```

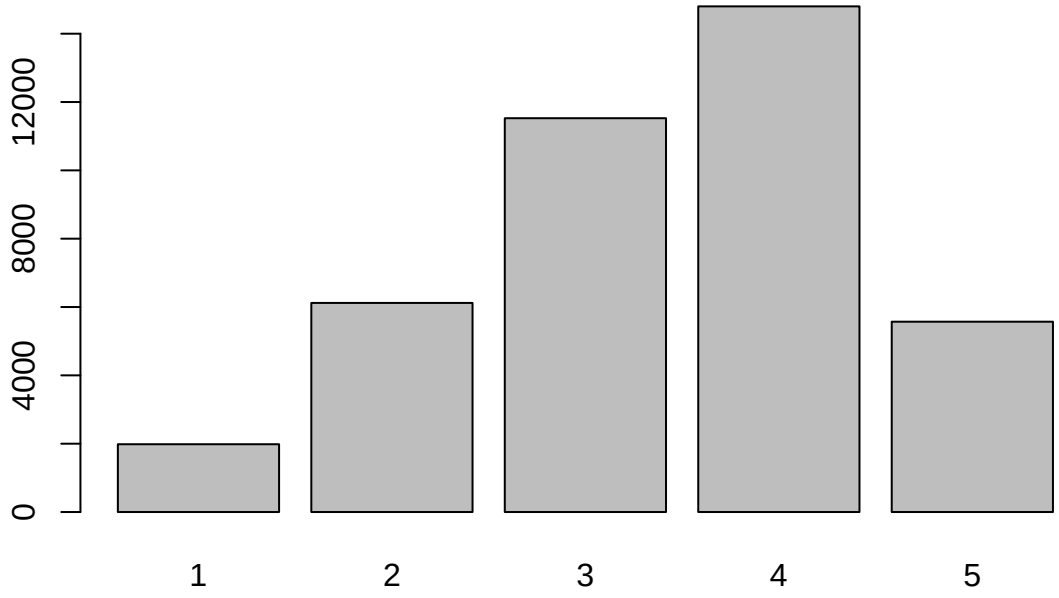
3.4. Simulation of satisfaction scores

In addition to our main dependent variable (NPS), we simulate satisfaction scores, which were available for some customer service categories in the original project.

These scores will be useful to explore the **convergent validity** of NPS, as they are expected to be positively correlated.

```
## Simulation of satisfaction scores (1 to 5 scale)

# Step 1: Generate satisfaction scores based on NPS
# We use a normal distribution centered at 3, with some variability (sd = 0.5)
# and add a scaled component based on NPS to simulate a positive relationship.
data$satisfaction <- floor(
  rnorm(data.n, mean = 3, sd = 0.5) +
  0.9 * (1 + as.numeric(scale(data$nps.score)))
)

# Step 2: Ensure scores stay within the 1-5 range
data$satisfaction[data$satisfaction < 1] <- 1
data$satisfaction[data$satisfaction > 5] <- 5
```

```r
# Step 3: Visualize the distribution
barplot(table(data$satisfaction))
```




```r
with(data, prop.table(table(satisfaction)))
```

```
## satisfaction
##       1       2       3       4       5
## 0.04960 0.15300 0.28815 0.37000 0.13925
```

```r
# Step 4: Check correlation with NPS scores (Spearman method)
with(data, cor(nps.score, satisfaction, method = "spearman"))
```

```
## [1] 0.8293966
```

3.4.1. Simulating missing data for satisfaction

To reflect the restrictions observed in the original project, we simulate that satisfaction data is **not available** in the following conditions:

- For customers from **Drinkland**
- For contacts via **email** or **web form**

In addition, we assume that even when satisfaction was supposed to be collected, there is a **10% probability of missingness**, simulating cases where customers simply skipped the question.

This missing data was a key consideration in the original project and contributed to the decision to focus on likelihood to recommend (Net Promoter Score) rather than satisfaction, as the latter was not consistently available across segments.

```
## Simulate missing data for satisfaction

# Step 1: Set satisfaction to missing for Drinkland
data$satisfaction[data$country == "Drinkland"] <- NA_real_

# Step 2: Set satisfaction to missing for email and web_form contacts
data$satisfaction[data$method %in% c("email", "web_form")] <- NA_real_

# Step 3: Introduce 10% additional random missingness among remaining valid values
# Create a random vector (0 = missing, 1 = keep) for non-NA entries
na_randomizer <- sample(
  c(0, 1),
  size = sum(!is.na(data$satisfaction)),
  replace = TRUE,
  prob = c(0.1, 0.9)
)
# Apply missing values where randomizer is 0
data$satisfaction[which(!is.na(data$satisfaction))[na_randomizer == 0]] <- NA_real_

# Step 4: Check final distribution of satisfaction (including NAs)
with(data, prop.table(table(is.na(satisfaction), method, country), margin = 2:3))
```

```
## , , country = Drinkland
##
##        method
##               chat      email      phone   web_form
##   FALSE 0.00000000 0.00000000 0.00000000 0.00000000
##   TRUE  1.00000000 1.00000000 1.00000000 1.00000000
##
## , , country = Eatland
##
##        method
##               chat      email      phone   web_form
##   FALSE 0.89319530 0.00000000 0.90619164 0.00000000
##   TRUE  0.10680470 1.00000000 0.09380836 1.00000000
```

4. Exploratory Analysis to Identify Key Areas for Improvement

4.1. Measurement Considerations: Convergent Validity of NPS Scores

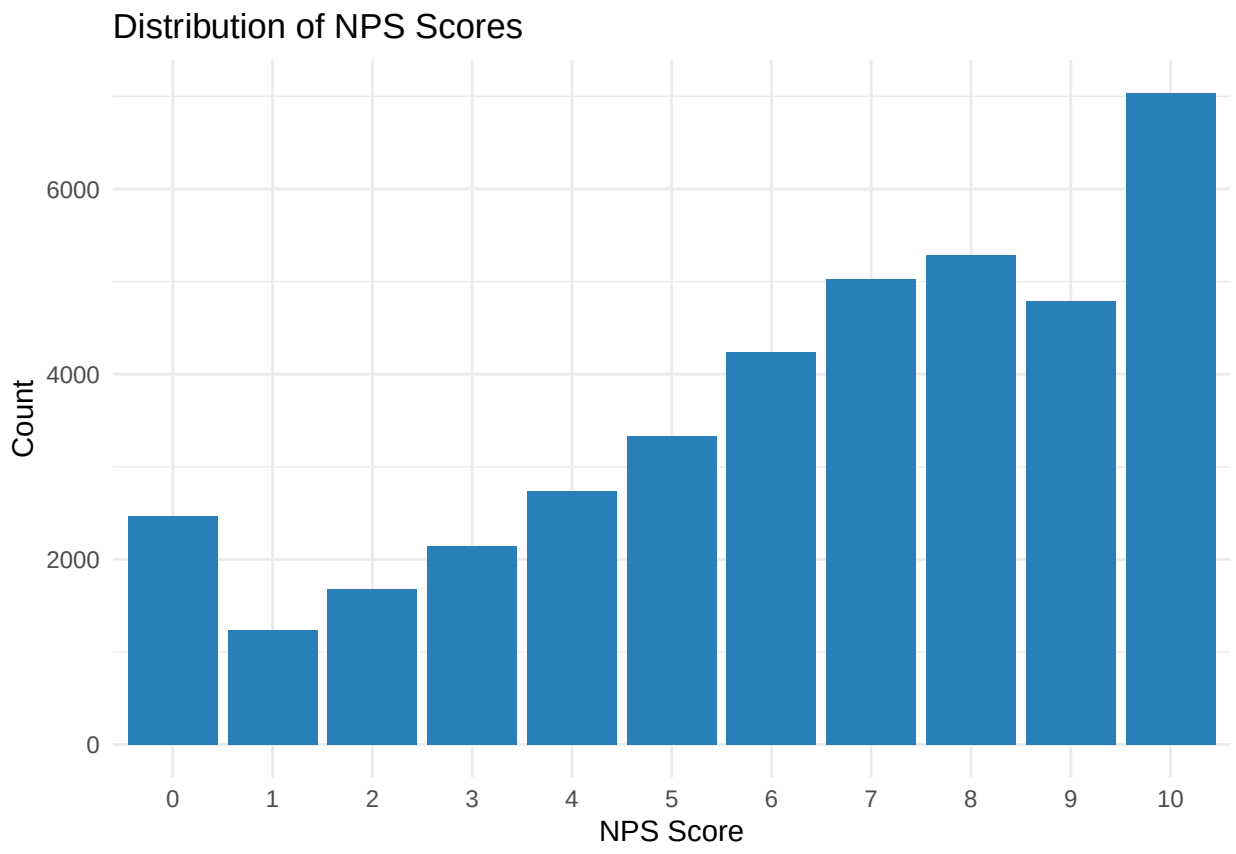
The validity of our main dependent variable, the Net Promoter Score (NPS), has been subject to criticism. Some of these critiques highlight that NPS relies on a single item, which may not fully capture the complexity of the likelihood to recommend. Additionally, the standard NPS formulation specifically asks about recommending the brand to friends or colleagues, which may not generalize to all recommendation scenarios.

To build confidence in the use of NPS within our context, we examine its convergent validity with satisfaction scores, which were available for a subset of markets and contact methods. A strong positive association between NPS and satisfaction would support the concurrent validity of NPS as a proxy for overall customer evaluation.

4.1.1. Preliminary considerations for correlations

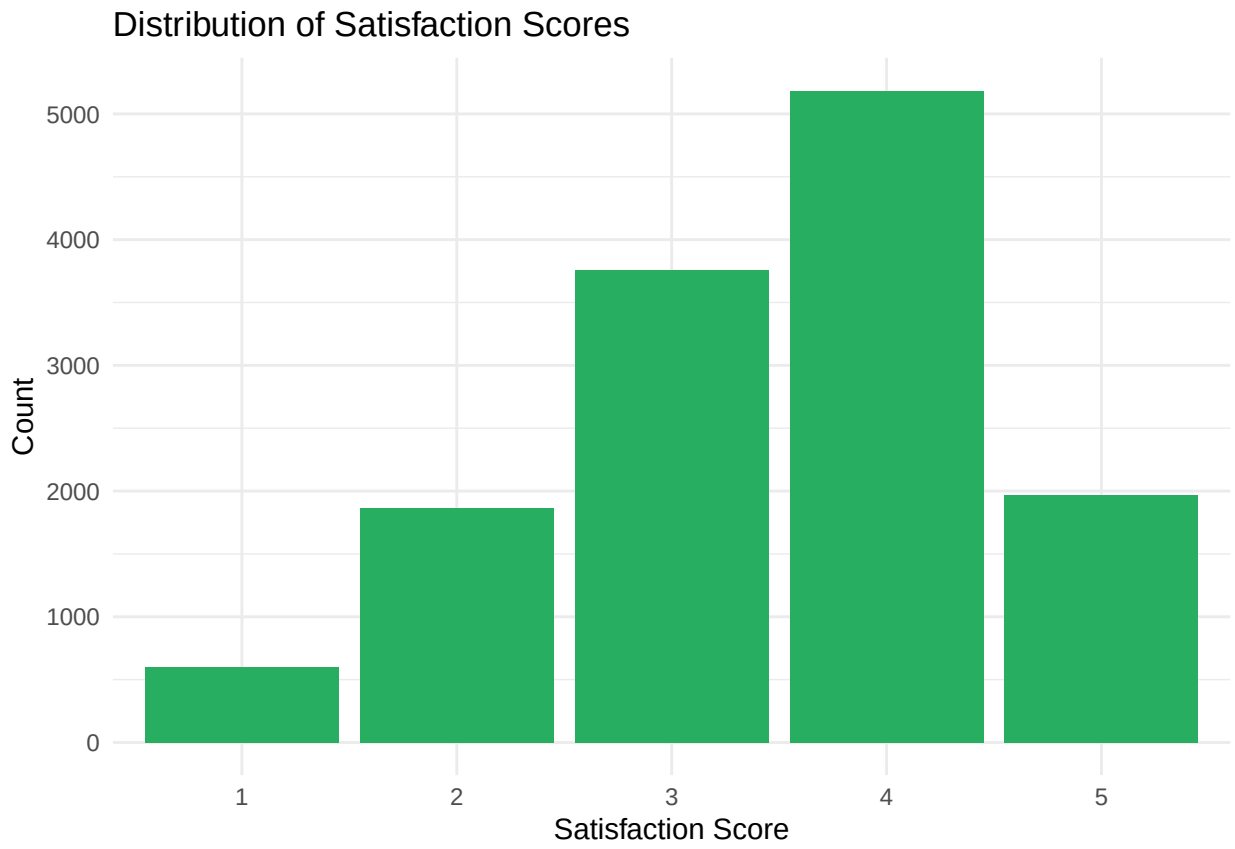
To determine the most appropriate method for computing correlations, we began by exploring the distributions of both satisfaction and NPS scores. For this, we used bar charts generated with the ggplot2 package:

```
# Distribution of NPS scores
ggplot(data[!is.na(data$nps.score), ], aes(x = factor(nps.score))) +
  geom_bar(fill = "#2980b9") +
  labs(
    title = "Distribution of NPS Scores",
    x = "NPS Score",
    y = "Count"
  ) +
  theme_minimal()
```




```
# Distribution of satisfaction scores
ggplot(data[!is.na(data$satisfaction), ], aes(x = factor(satisfaction))) +
  geom_bar(fill = "#27ae60") +
  labs(
    title = "Distribution of Satisfaction Scores",
```

```
    x = "Satisfaction Score",
    y = "Count"
  ) +
  theme_minimal()
```



Neither nps.score nor satisfaction follows a normal distribution. Both variables exhibit left-skewed patterns, with nps.score also showing signs of a ceiling effect, as a substantial proportion of scores accumulate near the top end of the scale.

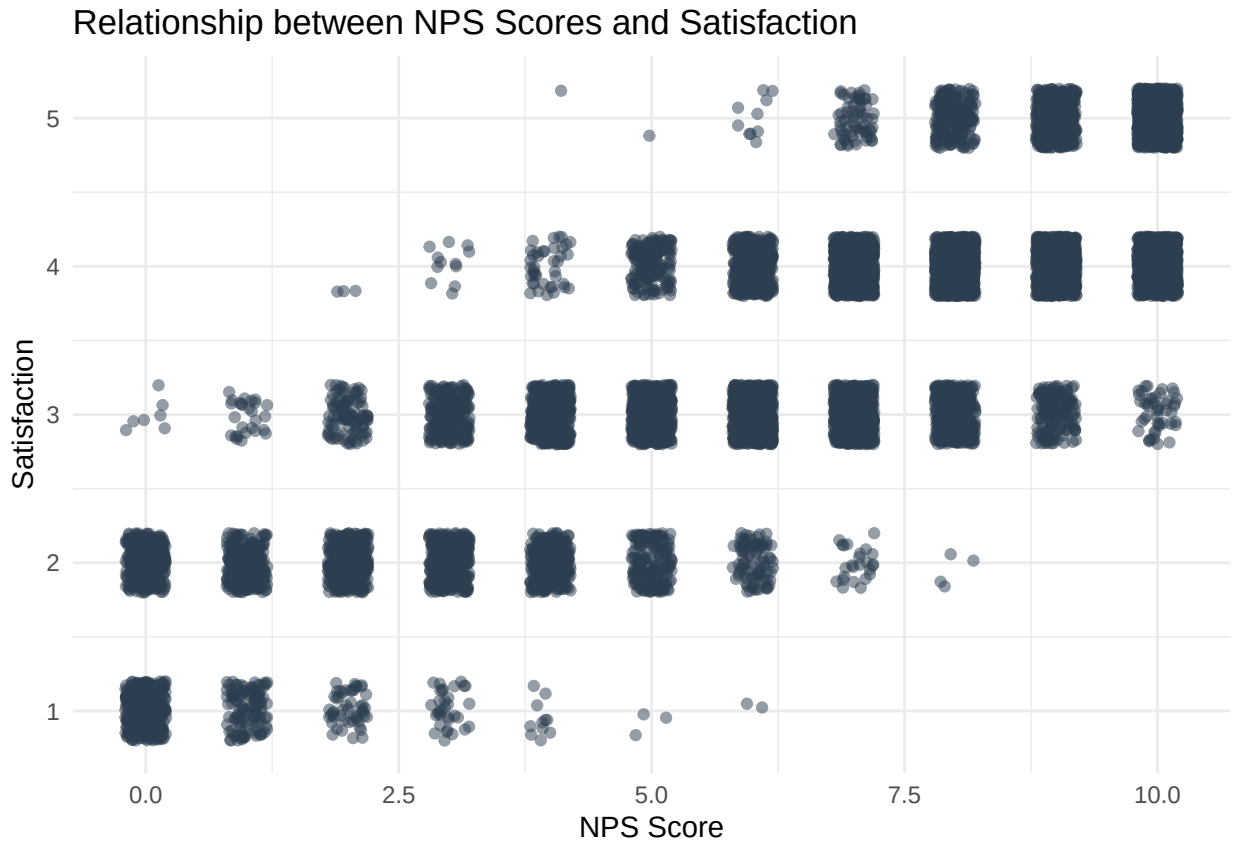
Given these distributional characteristics, Pearson's correlation is not suitable—it assumes linearity, homoscedasticity, and approximately normal distributions, and is highly sensitive to skewed or bounded data.

To inform the selection of a more appropriate correlation method—such as Spearman's rank correlation or polychoric correlation—we first inspect the relationship between nps.score and satisfaction visually. A scatterplot allows us to evaluate whether the association appears monotonic and to better understand its form and strength.

We use jittering to reduce overlap in the plot, given that both variables are discrete and bounded, which can otherwise obscure the pattern of relationships.

```
# Create a scatterplot to visualize the relationship between NPS and satisfaction

ggplot(na.omit(data[, c("nps.score", "satisfaction")]),
       aes(x = jitter(nps.score), y = jitter(satisfaction))) +
  geom_point(alpha = 0.5, color = "#2c3e50") +  # Add semi-transparent points for visual clarity
  labs(
    title = "Relationship between NPS Scores and Satisfaction",
```

```
    x = "NPS Score",
    y = "Satisfaction"
  ) +
  theme_minimal()
```



The scatterplot reveals a strong, monotonic—and approximately linear—relationship between NPS score and satisfaction.

4.1.2. Spearman Correlation between NPS and Satisfaction Scores

Given that both variables are ordinal in nature and not normally distributed, we use Spearman's rank correlation. This method evaluates the strength and direction of a monotonic association, without assuming interval-level measurement or normality.

```
# Compute Spearman's rank correlation between NPS and satisfaction using the 'psych' package
with(data, corr.test(nps.score, satisfaction, method = "spearman", use = "complete.obs"))
```

```
## Call:corr.test(x = nps.score, y = satisfaction, use = "complete.obs",
##     method = "spearman")
## Correlation matrix
## [1] 0.82
## Sample Size
## [1] 13383
## These are the unadjusted probability values.
```

```
##   The probability values  adjusted for multiple tests are in the p.adj object.
## [1] 0
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

The results indicate a strong positive Spearman correlation between satisfaction and NPS scores. This supports the convergent validity of NPS in our dataset and suggests it is a meaningful proxy for overall customer satisfaction.

4.1.3. Further Measurement Considerations

Despite the evidence supporting the convergent validity of the **raw NPS scores**, there remain concerns about the use of the **categorical NPS metric**, which is calculated by subtracting the percentage of detractors (scores 0–6) from the percentage of promoters (scores 9–10).

This transformation introduces ambiguity for two reasons:
1. **Loss of information**: Grouping continuous scores into categories discards nuance.
2. **Distributional blindness**: The categorical NPS does not account for the full shape of the score distribution. For example, an NPS of 20 could result from 20% promoters and 0% detractors—or from 60% promoters and 40% detractors—two very different satisfaction profiles.

While we will use the categorical NPS in certain summaries and visualizations—given its widespread familiarity among stakeholders, we will systematically compare those results with analyses based on **raw NPS scores**, to ensure consistency and transparency.

4.2. Preliminary Inspection of Key Customer Service Factors with Heatmaps

Given the large number of levels across the variables contact method, contact reason, and country, it is important to **explore potential interactions** between these factors before proceeding to formal modeling.

It is crucial to consider these interactions in the modeling to avoid confusions, such as attributing an effect to one factor when it is actually driven by an interaction with another.

To visually inspect these relationships, we will use heatmaps to examine the frequency of different combinations of these categorical variables.

4.2.1. Reusable Function to Generate Heatmaps

We first define a function that generates summary heatmaps for any dependent variable and any aggregation function we wish to apply (e.g., mean, categorical NPS, etc.), across our selected factors.

This function systematically provides results for all main effects as well as all levels of interaction between the factors, facilitating a comprehensive visual exploration of their relationships.

```r
f.heatmap.3f <- function(my.data, my.dv, factor.columns, factor.rows, factor.blocks, my.function) {

  # Initialize list to store results and a counter
  results <- list()
  k <- 1

  # Extract unique levels of each factor
  factor.column.levels <- unique(my.data[[factor.columns]])
  factor.rows.levels <- unique(my.data[[factor.rows]])
  factor.blocks.levels <- unique(my.data[[factor.blocks]])
```

```r
# 1. Overall results (no breakdown by block factor)

# 1.1 First row: totals for whole dataset and totals by column factor
row.label.t <- "Total all sample"

# Calculate overall summary metric (excluding NAs)
vector.total <- my.data[[my.dv]]
vector.total.clean <- vector.total[!is.na(vector.total)]
if (length(vector.total.clean) > 0) {
  total <- my.function(vector.total.clean)
} else {
  stop("The dataset is empty")
}

# Calculate summary metric for each level of the column factor
c.total <- sapply(factor.column.levels, function(c) {
  vector.c.total <- my.data[my.data[[factor.columns]] == c, my.dv]
  vector.c.total.clean <- vector.c.total[!is.na(vector.c.total)]
  if (length(vector.c.total.clean) > 0) {
    my.function(vector.c.total.clean)
  } else {
    NA_real_
  }
})

# Save totals row in results list
results[[k]] <- c(group = row.label.t, overall = total, setNames(object = c.total, factor.column.level
k <- k + 1

# 1.2 Additional rows: results for each level of the row factor,
# with column factor breakdowns

for (r in factor.rows.levels) {
  row.label.r <- r

  r.total.vector <- my.data[my.data[[factor.rows]] == r, my.dv]
  r.total.vector.clean <- r.total.vector[!is.na(r.total.vector)]
  r.total <- if (length(r.total.vector.clean) > 0) my.function(r.total.vector.clean) else NA_real_

  rc.crossed <- sapply(factor.column.levels, function(c) {
    rc.crossed.vector <- my.data[
      my.data[[factor.rows]] == r & my.data[[factor.columns]] == c,
      my.dv
    ]
    rc.crossed.vector.clean <- rc.crossed.vector[!is.na(rc.crossed.vector)]
    if (length(rc.crossed.vector.clean) > 0) {
      my.function(rc.crossed.vector.clean)
    } else {
      NA_real_
    }
  })

  results[[k]] <- c(group = row.label.r, overall = r.total, setNames(object = rc.crossed, factor.colu
```

```
  k <- k + 1
}

# 2. Results broken down by block factor

for (b in factor.blocks.levels) {
  row.label.b <- paste0("Total: ", b)

  b.vector <- my.data[my.data[[factor.blocks]] == b, my.dv]
  b.vector.clean <- b.vector[!is.na(b.vector)]
  b.total <- if (length(b.vector.clean) > 0) my.function(b.vector.clean) else NA_real_

  bc.crossed <- sapply(factor.column.levels, function(c) {
    bc.crossed.vector <- my.data[
      my.data[[factor.blocks]] == b & my.data[[factor.columns]] == c,
      my.dv
    ]
    bc.crossed.vector.clean <- bc.crossed.vector[!is.na(bc.crossed.vector)]
    if (length(bc.crossed.vector.clean) > 0) my.function(bc.crossed.vector.clean) else NA_real_
  })

  results[[k]] <- c(group = row.label.b, overall = b.total, setNames(object = bc.crossed, factor.colu
  k <- k + 1

  # 2.2 Rows within blocks: results for each row factor level within block,
  # with column factor breakdowns

  for (r in factor.rows.levels) {
    row.label.br <- paste0(b, ": ", r)

    br.crossed.vector <- my.data[
      my.data[[factor.blocks]] == b & my.data[[factor.rows]] == r,
      my.dv
    ]
    br.crossed.vector.clean <- br.crossed.vector[!is.na(br.crossed.vector)]
    br.crossed <- if (length(br.crossed.vector.clean) > 0) my.function(br.crossed.vector.clean) else

    all.crossed <- sapply(factor.column.levels, function(c) {
      all.crossed.vector <- my.data[
        my.data[[factor.blocks]] == b &
        my.data[[factor.rows]] == r &
        my.data[[factor.columns]] == c,
        my.dv
      ]
      all.crossed.vector.clean <- all.crossed.vector[!is.na(all.crossed.vector)]
      if (length(all.crossed.vector.clean) > 0) my.function(all.crossed.vector.clean) else NA_real_
    })

    results[[k]] <- c(group = row.label.br, overall = br.crossed, setNames(object = all.crossed, facto
    k <- k + 1
  }
}
```

```
  # 3. Combine results into a dataframe and convert numeric columns
  results.df <- as.data.frame(do.call(rbind, results), stringsAsFactors = FALSE)
  results.df[, -1] <- lapply(results.df[, -1], as.numeric)

  return(results.df)
}
```

4.2.2. Heatmap for NPS in Categorical Scores

We first created a heatmap using NPS in its categorical form, as this is the format most familiar to stakeholders. This choice facilitates collaboration and interpretation, enabling domain experts to visually identify potential patterns and effects across key service factors.

To do so, we implemented a simple, reusable function that computes the categorical NPS score from any vector of raw NPS values. This function includes checks for common data issues (e.g., missing or out-of-range values) and provides warnings when the sample size may be insufficient for stable estimates.

```
f.nps.cat <- function (my.nps.vector, na.rm=TRUE){
  # my.nps.vector is a numeric vector of raw NPS scores. It can contain NAs

  vector.c <- my.nps.vector[!is.na(my.nps.vector)] # vector cleaned of NAs

  # Error messages if sample size = 0 or if there are out-of-range NPS values:
  if(length(vector.c) == 0) {stop("error: no available NPS raw scores")}
  if(length(vector.c[vector.c > 10]) > 0) {stop("error: scores higher than 10 in the NPS raw scores")}
  if(length(vector.c[vector.c < 0]) > 0) {stop("error: scores lower than 0 in the NPS raw scores")}

  # Warning if sample size is below 70 (low reliability)
  if(length(vector.c) < 70) {warning("few scores available for calculation (n<70)")}

  n <- length(vector.c) # sample size (excluding NAs)
  prop.promoters <- (length(vector.c[vector.c >= 9])) / n # proportion of promoters
  prop.detractors <- (length(vector.c[vector.c <= 6])) / n # proportion of detractors
  nps.cat <- (prop.promoters - prop.detractors) * 100 # NPS score in categorical terms

  return(nps.cat) # returns NPS as a categorical score
}

# Check if it is working as expected
f.nps.cat(data$nps.score)
```

```
## [1] -15.035
```

```
with(data, prop.table(table(nps.segment)))
```

```
## nps.segment
## detractor   passive  promoter
##   0.44625   0.25785   0.29590
```

The function is working correctly (we can confirm that the result matches the subtraction of the percentage of promoters and detractors).

We are now ready to generate the heatmap for the categorical NPS using the reusable functions defined earlier.

```
# Use the function to calculate heatmaps
nps.cat.heatmap <- f.heatmap.3f(
  my.data = data,
  my.dv = "nps.score",
  factor.columns = "method",
  factor.rows = "reason",
  factor.blocks = "country",
  my.function = f.nps.cat  # Function to compute categorical NPS scores
)

# Export the results to Excel to apply conditional formatting (e.g., color scales)
write.xlsx(nps.cat.heatmap, "nps.cat.heatmap.xlsx", colnames = TRUE)

#Check if you have acceptable sample sizes for all cells, using also the heatmap function
nps.cat.heatmap.n <- f.heatmap.3f (
  my.data = data,
  my.dv = "nps.score",
  factor.columns = "method",
  factor.rows = "reason",
  factor.blocks = "country",
  my.function = length
)
```

All sample sizes across the heatmap cells are sufficiently large (n > 150), reducing the likelihood of sampling error and increasing confidence in the observed patterns.

After applying conditional formatting (color scales) in Excel, we obtain the heatmap, which helps us identify areas where the probability of recommending our customer service is particularly low. This visualization allows us to assess whether these effects are driven by main factors or by interactions between them.

```
# Import the image of the heatmap generated in Excel with conditional formatting
knitr::include_graphics("Heatmap.nps.cat.png")
```

NPS CAT					
group	overall	phone	chat	email	web_form
Total all sample	**-15.04**	**-3.65**	**-21.80**	**-25.11**	**-12.47**
status	-9.85	-1.39	-13.44	-21.32	-7.66
delay	-11.21	-2.62	-12.72	-26.91	-11.00
other	-11.07	-1.70	-13.57	-23.99	-15.09
incorrect_item	-26.62	-4.75	-50.05	-24.11	-12.45
cancellation	-21.47	-13.29	-24.83	-31.77	-20.57
Total: Eatland	**-11.65**	**1.26**	**-19.54**	**-23.13**	**-8.09**
Eatland: status	-6.44	2.71	-10.45	-17.73	-5.29
Eatland: delay	-8.07	2.83	-10.80	-27.83	-5.49
Eatland: other	-7.21	1.42	-9.42	-20.46	-10.53
Eatland: incorrect_item	-23.40	1.91	-50.50	-21.81	-5.50
Eatland: cancellation	-17.71	-8.67	-20.60	-28.07	-20.18
Total: Drinkland	**-18.37**	**-8.53**	**-23.99**	**-27.08**	**-16.94**
Drinkland: status	-13.28	-5.49	-16.40	-24.96	-10.26
Drinkland: delay	-14.32	-8.13	-14.58	-26.03	-16.61
Drinkland: other	-14.76	-4.79	-17.55	-27.17	-19.08
Drinkland: incorrect_item	-29.89	-11.36	-49.60	-26.68	-19.73
Drinkland: cancellation	-25.12	-17.51	-28.95	-35.40	-21.05

The heatmap reveals several **main effects**:

- Customers in Drinkland show a lower probability to promote compared to those in Eatland (see the yellow-colored area in the Drinkland block).
- Customers contacting for cancellations are less likely to promote than those contacting for other reasons (see the redder rows corresponding to cancellation reasons).
- Phone is associated with the highest probability to promote, while email shows the lowest (see the greener column for phone and the redder column for email).

More importantly, the heatmap helps us detect an **interaction** between contact reason and contact method: Customers who contact through chat for incorrect item deliveries show a particularly low probability to promote (see the deep red cells at the intersection of the chat column and incorrect item row).

Before proceeding to statistically model these effects, we will check whether similar patterns emerge when using alternative outcome measures, such as raw NPS scores and satisfaction ratings.

4.2.3. Heatmap for NPS Raw Scores

To ensure that the limited categorization in the NPS categorical scores does not lead to misleading patterns or ambiguities, we examine whether similar patterns emerge when using the NPS raw scores. For this, we apply our reusable heatmap function to calculate the mean of the NPS raw scores across the factors.

```
# Apply the heatmap function to NPS raw scores
nps.raw.heatmap <- f.heatmap.3f(
  my.data = data,
  my.dv = "nps.score",
  factor.columns = "method",
  factor.rows = "reason",
  factor.blocks = "country",
  my.function = mean  # Calculation of the mean of the NPS raw scores
)

# Export the heatmap data to Excel for conditional formatting (color scales)
write.xlsx(nps.raw.heatmap, "nps.raw.heatmap.xlsx", colnames = TRUE)

# Visualize the heatmap after adding colors in Excel
knitr::include_graphics("Heatmap.nps.2.png")
```

NPS RAW					
group	overall	phone	chat	email	web_form
Total all sample	**6.38**	**6.87**	**6.05**	**6.01**	**6.53**
status	**6.63**	6.93	6.56	6.10	6.67
delay	**6.57**	6.90	6.52	5.98	6.55
other	**6.56**	6.94	6.42	6.07	6.49
incorrect_item	**5.75**	6.87	4.47	6.05	6.53
cancellation	**6.21**	6.54	6.11	5.69	6.25
Total: Eatland	**6.52**	**7.04**	**6.17**	**6.11**	**6.66**
Eatland: status	**6.76**	7.09	6.67	6.22	6.73
Eatland: delay	**6.68**	7.04	6.61	6.00	6.71
Eatland: other	**6.72**	7.05	6.62	6.22	6.64
Eatland: incorrect_item	**5.91**	7.13	4.53	6.17	6.70
Eatland: cancellation	**6.38**	6.72	6.29	5.90	6.32
Total: Drinkland	**6.24**	**6.70**	**5.94**	**5.91**	**6.39**
Drinkland: status	**6.50**	6.76	6.45	5.99	6.60
Drinkland: delay	**6.47**	6.75	6.43	5.97	6.39
Drinkland: other	**6.41**	6.84	6.23	5.92	6.36
Drinkland: incorrect_item	**5.59**	6.61	4.40	5.92	6.35
Drinkland: cancellation	**6.04**	6.37	5.93	5.49	6.15

NPS CAT					
group	overall	phone	chat	email	web_form
Total all sample	**-15.04**	**-3.65**	**-21.80**	**-25.11**	**-12.47**
status	-9.85	-1.39	-13.44	-21.32	-7.66
delay	-11.21	-2.62	-12.72	-26.91	-11.00
other	-11.07	-1.70	-13.57	-23.99	-15.09
incorrect_item	-26.62	-4.75	-50.05	-24.11	-12.45
cancellation	-21.47	-13.29	-24.83	-31.77	-20.57
Total: Eatland	**-11.65**	**1.26**	**-19.54**	**-23.13**	**-8.09**
Eatland: status	-6.44	2.71	-10.45	-17.73	-5.29
Eatland: delay	-8.07	2.83	-10.80	-27.83	-5.49
Eatland: other	-7.21	1.42	-9.42	-20.46	-10.53
Eatland: incorrect_item	-23.40	1.91	-50.50	-21.81	-5.50
Eatland: cancellation	-17.71	-8.67	-20.60	-28.07	-20.18
Total: Drinkland	**-18.37**	**-8.53**	**-23.99**	**-27.08**	**-16.94**
Drinkland: status	-13.28	-5.49	-16.40	-24.96	-10.26
Drinkland: delay	-14.32	-8.13	-14.58	-26.03	-16.61
Drinkland: other	-14.76	-4.79	-17.55	-27.17	-19.08
Drinkland: incorrect_item	-29.89	-11.36	-49.60	-26.68	-19.73
Drinkland: cancellation	-25.12	-17.51	-28.95	-35.40	-21.05

We observe a similar pattern in both methods of calculating NPS results. However, it is also valuable to examine whether satisfaction scores reveal comparable patterns.

4.2.4. Heatmap for Satisfaction Scores

As further evidence of convergent validity between NPS and satisfaction scores, we assess whether these two measures show similar patterns across our factors.

To do this, we apply our reusable heatmap function to compute the mean satisfaction scores.

```
# Apply the heatmap function to satisfaction scores
satisfaction.heatmap <- f.heatmap.3f(
  my.data = data,
  my.dv = "satisfaction",
  factor.columns = "method",
  factor.rows = "reason",
  factor.blocks = "country",
  my.function = mean  # Calculation of the mean satisfaction scores
)

# Export the heatmap data to Excel for conditional formatting (color scales)
write.xlsx(satisfaction.heatmap, "satisfaction.heatmap.xlsx", colnames = TRUE)

# Visualize all heatmaps after applying colors in Excel
knitr::include_graphics("Heatmap.png")
```

SATISFACTION

group	overall	phone	chat	email	web_form
Total all sample	**3.45**	**3.59**	**3.33**	**0.00**	**0.00**
status	**3.56**	**3.63**	3.49	0.00	0.00
delay	**3.53**	**3.59**	3.47	0.00	0.00
other	**3.52**	**3.56**	3.48	0.00	0.00
incorrect_item	**3.21**	**3.63**	**2.83**	0.00	0.00
cancellation	**3.39**	3.47	3.33	0.00	0.00
Total: Eatland	**3.45**	**3.59**	**3.33**	**0.00**	**0.00**
Eatland: status	**3.56**	**3.63**	3.49	0.00	0.00
Eatland: delay	**3.53**	**3.59**	3.47	0.00	0.00
Eatland: other	**3.52**	**3.56**	3.48	0.00	0.00
Eatland: incorrect_item	**3.21**	**3.63**	**2.83**	0.00	0.00
Eatland: cancellation	**3.39**	3.47	3.33	0.00	0.00
Total: Drinkland	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Drinkland: status	**0.00**	0.00	0.00	0.00	0.00
Drinkland: delay	**0.00**	0.00	0.00	0.00	0.00
Drinkland: other	**0.00**	0.00	0.00	0.00	0.00
Drinkland: incorrect_item	**0.00**	0.00	0.00	0.00	0.00
Drinkland: cancellation	**0.00**	0.00	0.00	0.00	0.00

NPS RAW

group	overall	phone	chat	email	web_form
Total all sample	**6.38**	**6.87**	**6.05**	**6.01**	**6.53**
status	**6.63**	6.93	6.56	6.10	6.67
delay	**6.57**	6.90	6.52	5.98	6.55
other	**6.56**	6.94	6.42	6.07	6.49
incorrect_item	**5.75**	6.87	**4.47**	6.05	6.53
cancellation	**6.21**	6.54	6.11	5.69	6.25
Total: Eatland	**6.52**	**7.04**	**6.17**	**6.11**	**6.66**
Eatland: status	**6.76**	7.09	6.67	6.22	6.73
Eatland: delay	**6.68**	7.04	6.61	6.00	6.71
Eatland: other	**6.72**	7.05	6.62	6.22	6.64
Eatland: incorrect_item	**5.91**	7.13	**4.53**	6.17	6.70
Eatland: cancellation	**6.38**	6.72	6.29	5.90	6.32
Total: Drinkland	**6.24**	**6.70**	**5.94**	**5.91**	**6.39**
Drinkland: status	**6.50**	6.76	6.45	5.99	6.60
Drinkland: delay	**6.47**	6.75	6.43	5.97	6.39
Drinkland: other	**6.41**	6.84	6.23	5.92	6.36
Drinkland: incorrect_item	**5.59**	6.61	**4.40**	5.92	6.35
Drinkland: cancellation	**6.04**	6.37	5.93	5.49	6.15

NPS CAT

group	overall	phone	chat	email	web_form
Total all sample	**-15.04**	**-3.65**	**-21.80**	**-25.11**	**-12.47**
status	-9.85	-1.39	-13.44	-21.32	-7.66
delay	-11.21	-2.62	-12.72	-26.91	-11.00
other	-11.07	-1.70	-13.57	-23.99	-15.09
incorrect_item	-26.62	-4.75	**-50.05**	-24.11	-12.45
cancellation	-21.47	-13.29	-24.83	-31.77	-20.57
Total: Eatland	**-11.65**	**1.26**	**-19.54**	**-23.13**	**-8.09**
Eatland: status	-6.44	2.71	-10.45	-17.73	-5.29
Eatland: delay	-8.07	2.83	-10.80	-27.83	-5.49
Eatland: other	-7.21	1.42	-9.42	-20.46	-10.53
Eatland: incorrect_item	-23.40	1.91	**-50.50**	-21.81	-5.50
Eatland: cancellation	-17.71	-8.67	-20.60	-28.07	-20.18
Total: Drinkland	**-18.37**	**-8.53**	**-23.99**	**-27.08**	**-16.94**
Drinkland: status	-13.28	-5.49	-16.40	-24.96	-10.26
Drinkland: delay	-14.32	-8.13	-14.58	-26.03	-16.61
Drinkland: other	-14.76	-4.79	-17.55	-27.17	-19.08
Drinkland: incorrect_item	-29.89	-11.36	**-49.60**	-26.68	-19.73
Drinkland: cancellation	-25.12	-17.51	-28.95	**-35.40**	-21.05

We observe a similar pattern for satisfaction and both methods of calculating NPS results, which increases our confidence in the validity of NPS scores within this context.

4.2.5. Correlations between Results of Different Dependent Variables

To further assess convergence between the dependent variables, we calculate Pearson correlations between their heatmap results.

First, we reshape each heatmap dataframe into a long format with a single column for the values. Then, we merge these datasets and calculate the correlations.

```
# Reshape heatmap results into long format
sat.long <- satisfaction.heatmap %>% pivot_longer(cols = -1,
                                  names_to = "evaluation",
                                  values_to = "satisfaction")

nps.raw.long <- nps.raw.heatmap %>% pivot_longer(cols = -1,
                                  names_to = "evaluation",
                                  values_to = "nps.raw")

nps.cat.long <- nps.cat.heatmap %>% pivot_longer(cols = -1,
                                  names_to = "evaluation",
                                  values_to = "nps.cat")

# Merge all heatmap results into a single dataset
merged.heatmaps <- merge(nps.cat.long, merge(sat.long, nps.raw.long, by = c("group", "evaluation")), by

# Calculate Pearson correlations across the three measures
cor(merged.heatmaps[, 3:5], use = "complete.obs")
```

```
##                nps.cat satisfaction   nps.raw
## nps.cat      1.0000000    0.9693055 0.9867490
## satisfaction 0.9693055    1.0000000 0.9831536
## nps.raw      0.9867490    0.9831536 1.0000000
```

All three measures show strong correlations ($r > 0.90$), providing additional confidence that the NPS is a valid measure and that important patterns are not missed by relying on any single metric.

4.3. Modeling Key Factors Associated with the Probability to Promote the Brand

While the heatmaps provided valuable visual insights into areas where customer service performance appears to be lower, statistical modeling is necessary to formally quantify these relationships.

4.3.1. Setting Reference Levels for Factor Categories

To make the interpretation of model coefficients more intuitive, we set the reference levels of our categorical predictors based on the heatmap results. Specifically, we use the levels associated with higher NPS scores as the reference. This allows us to interpret model outputs as comparisons against the best-performing category for each factor.

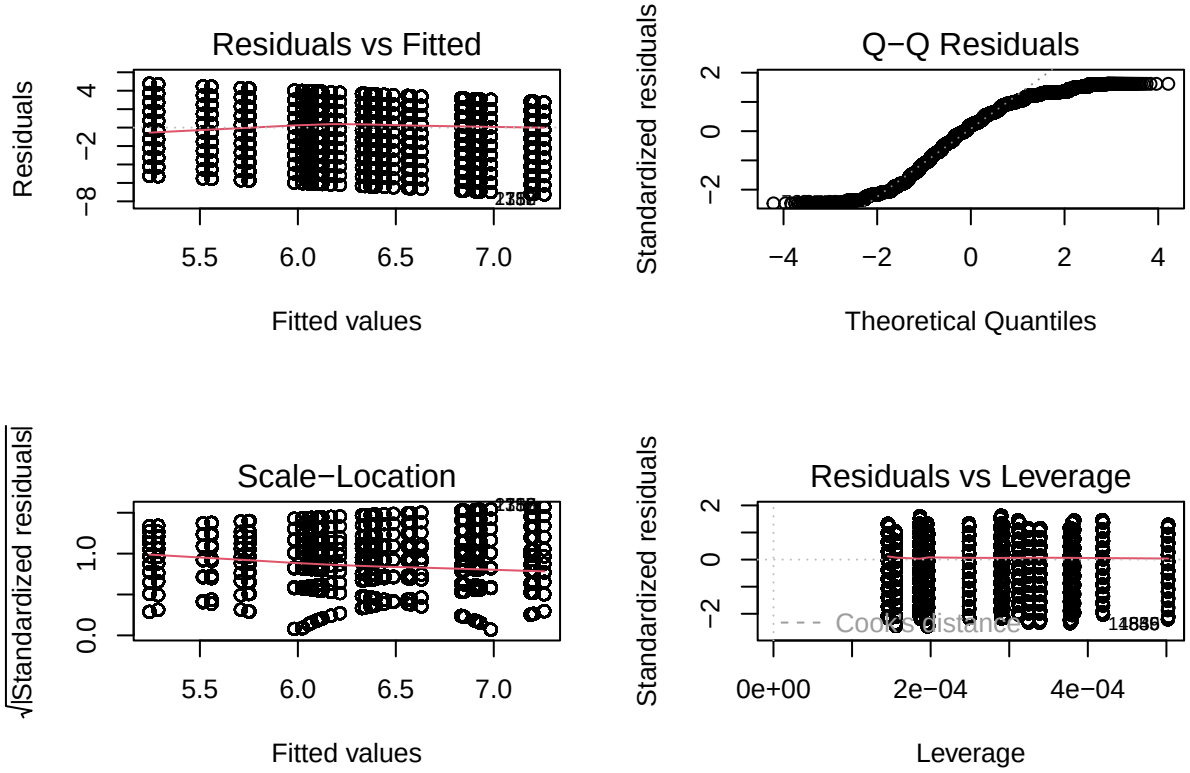
```
# Set the order of the factor levels
data$method <- factor(data$method, levels = c("phone", "web_form", "chat", "email"))
data$reason <- factor(data$reason, levels = c("other","status", "delay", "cancellation", "incorrect_iten
data$country <- factor(data$country, levels = c("Eatland", "Drinkland"))
```

4.3.2. Discarding a Linear Regression Model.

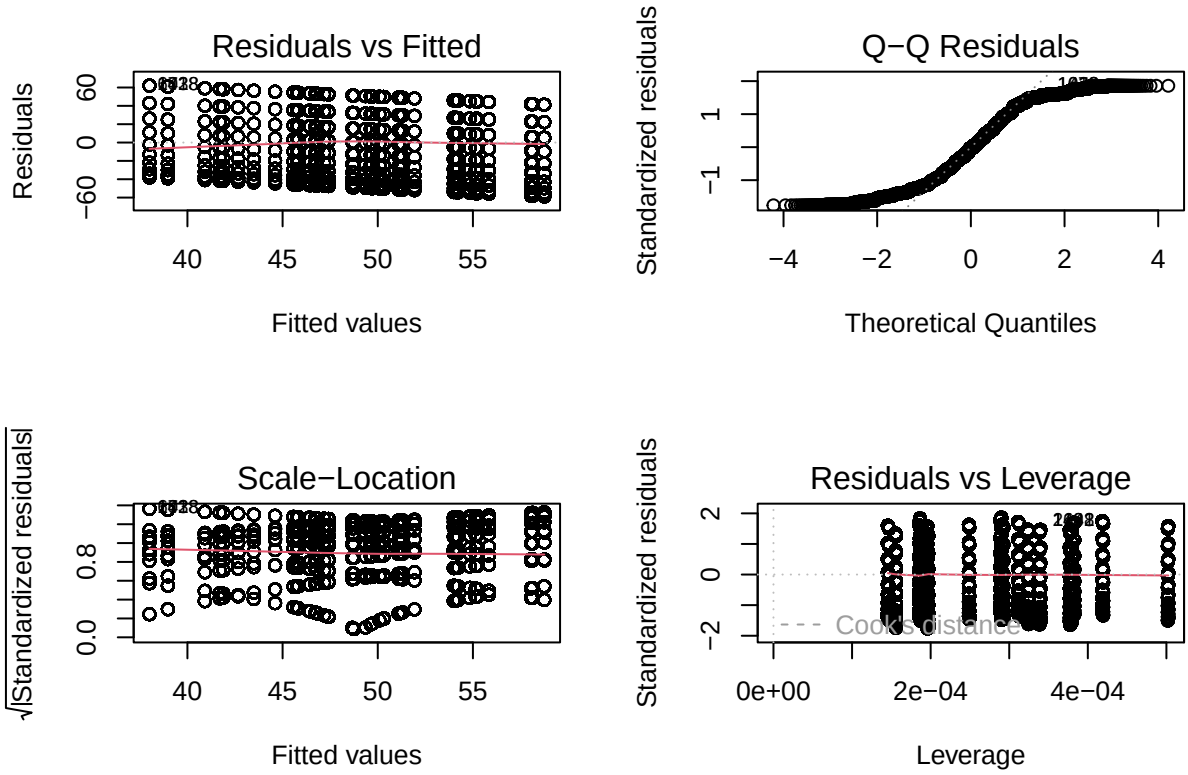
We discard the linear model because our outcome variable, NPS score, is ordinal in nature, and it is difficult to assume equal distances between adjacent scores given the ceiling and floor effects observed in its distribution (see Section 4.1.1).

In addition, a linear model fails to meet key assumptions like the normality of residuals, as shown in the Q-Q plots below, even when we apply a square transformation to the dependent variable.

```
#Set the configuration to see 4 plots in the same window:
par(mfrow = c(2, 2))

# Check normality and homoscedasticity of residuals for the linear model
linear.model <- lm(nps.score ~ method + reason + country, data = data)
plot(linear.model)
```




```
# Check the same for a squared transformation of the dependent variable
linear.model.sq <- lm((nps.score^2) ~ method + reason + country, data = data)
plot(linear.model.sq)
```



As shown in the Q-Q plots, the distribution of residuals deviates substantially from the theoretical normal distribution.

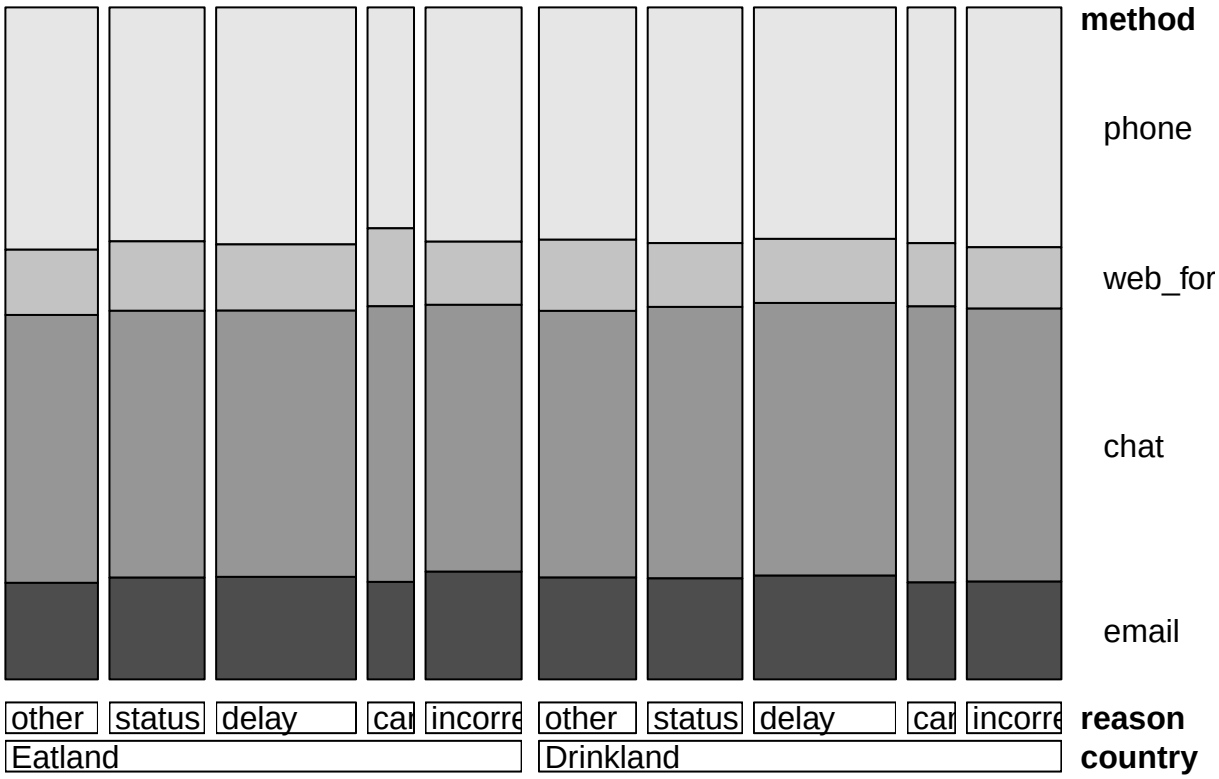
4.3.3. Evaluating the Appropriateness of an Ordinal Model

We prioritize testing an ordinal logistic model over a binomial logistic model, as it allows us to estimate the effects of our predictors across the full NPS scale, rather than focusing on a single outcome category (e.g., detractors).

However, before proceeding, we need to check whether key assumptions of this model are met—particularly the absence of multicollinearity and the proportional odds assumption across the levels of our dependent variable.

4.3.3.1. Checking the Collinearity Assumption We assess potential collinearity among our categorical predictors using both visualizations and statistical diagnostics.

```
# Mosaic plot to visualize the relationships between categorical factors (vcd package)
doubledecker(with(data, table(country, reason, method)))
```




```
# VIF scores from the linear model previously computed (car package)
vif(linear.model)
```

```
##             GVIF Df GVIF^(1/(2*Df))
## method  1.000321  3        1.000054
## reason  1.000397  4        1.000050
## country 1.000188  1        1.000094
```

The VIF analysis shows no indication of multicollinearity among the predictors. Specifically, the adjustment of the Generalized Variance Inflation Factors (GVIF^(1/(2*Df))), which allows for the comparison of categorical variables with different degrees of freedom, yields values that are all very close to 1.0.

This statistical result is further supported by the mosaic plot, which shows no strong associations between the levels of the factors. The distribution of frequencies appears relatively uniform across these levels.

4.3.3.2. Checking the Proportional Odds Assumption Before testing the proportional odds assumption, we create dummy variables for the factors with multiple categories. This allows us to assess the assumption separately for each category.

We also ensure that our dependent variable, the NPS score, is properly defined as an ordinal factor.

```
# Dummy variables for method (reference: phone)
data$chat <- ifelse(data$method == "chat", 1, 0)
data$email <- ifelse(data$method == "email", 1, 0)
data$web_form <- ifelse(data$method == "web_form", 1, 0)
```

```r
# Dummy variables for reason (reference: other)
data$delay <- ifelse(data$reason == "delay", 1, 0)
data$cancellation <- ifelse(data$reason == "cancellation", 1, 0)
data$incorrect_item <- ifelse(data$reason == "incorrect_item", 1, 0)
data$status <- ifelse(data$reason == "status", 1, 0)

# Register the NPS score as an ordinal factor
data$nps.score <- factor(data$nps.score, levels = 0:10, ordered = TRUE)
```

We now test the proportional odds assumption using the ordinal package.

```r
# Specify the ordinal logistic regression model with main effects
ordinal.model <- clm(nps.score ~ web_form + chat + email + delay + status + cancellation + incorrect_it

# Summary of the model
summary(ordinal.model)
```

```
## formula:
## nps.score ~ web_form + chat + email + delay + status + cancellation + incorrect_item + country
## data:    data
##
##  link  threshold nobs  logLik    AIC       niter max.grad cond.H
##  logit flexible  40000 -90899.01 181834.03 7(0)  4.32e-12 5.6e+02
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## web_form         -0.201977   0.031405  -6.431 1.26e-10 ***
## chat             -0.460248   0.020263 -22.713  < 2e-16 ***
## email            -0.493613   0.027018 -18.270  < 2e-16 ***
## delay             0.005406   0.025112   0.215    0.830
## status            0.037223   0.027441   1.356    0.175
## cancellation     -0.210165   0.033712  -6.234 4.54e-10 ***
## incorrect_item   -0.449426   0.027844 -16.141  < 2e-16 ***
## countryDrinkland -0.155422   0.017474  -8.895  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##      Estimate Std. Error  z value
## 0|1  -3.21679    0.03170 -101.467
## 1|2  -2.77176    0.02942  -94.204
## 2|3  -2.34536    0.02787  -84.155
## 3|4  -1.93914    0.02680  -72.358
## 4|5  -1.53253    0.02601  -58.920
## 5|6  -1.12321    0.02544  -44.159
## 6|7  -0.66723    0.02502  -26.667
## 7|8  -0.15237    0.02483   -6.138
## 8|9   0.43336    0.02498   17.346
## 9|10  1.11668    0.02585   43.192
```

```r
# Test of the proportional odds assumption
nominal_test(ordinal.model)
```

```
## Tests of nominal effects
##
## formula: nps.score ~ web_form + chat + email + delay + status + cancellation + incorrect_item + coun
##                Df logLik    AIC     LRT  Pr(>Chi)
## <none>            -90899 181834
## web_form        9 -90894 181843   9.170  0.421712
## chat            9 -90859 181772  80.235 1.451e-13 ***
## email           9 -90886 181826  26.397  0.001758 **
## delay           9 -90888 181830  22.051  0.008720 **
## status          9 -90889 181832  19.642  0.020257 *
## cancellation    9 -90892 181838  14.331  0.111028
## incorrect_item  9 -90819 181693 159.520 < 2.2e-16 ***
## country         9 -90888 181831  21.084  0.012283 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results suggest that the proportional odds assumption is violated for most of the predictors (all except contacting via web form and contacting for cancellation issues). This implies that the effect of these predictors varies depending on which threshold along the NPS scale is considered.

We will inspect several ways to solve this issue.

4.3.3.2.1 Addressing Non-Proportional Odds by Incorporating an Interaction Term

To address the observed violation of the proportional odds assumption, we explore the inclusion of a theoretically meaningful interaction in the model.

In the heatmaps, we identified a strong interaction: customers who contacted customer service via chat regarding incorrect item deliveries showed particularly low NPS scores. Including this interaction in the model may not only improve interpretability but also help address the previously observed violations of the proportional odds assumption for the method and reason factors.

We begin by evaluating whether this model that includes the interaction term offers a more parsimonious and appropriate fit:

```r
# Step 1: Create a binary variable for the interaction between 'chat' and 'incorrect_item'
data$chat_incorrect_item <- NA_real_
data$chat_incorrect_item[data$method == "chat" & data$reason == "incorrect_item"] <- 1
data$chat_incorrect_item[!(data$method == "chat" & data$reason == "incorrect_item")] <- 0

# Check that the variable has been created correctly
with(data, table(chat_incorrect_item, method, reason))
```

```
## , , reason = other
##
##                    method
## chat_incorrect_item phone web_form chat email
##                   0  2831      815 3190  1184
##                   1     0        0    0     0
##
## , , reason = status
##
##                    method
## chat_incorrect_item phone web_form chat email
##                   0  2804      796 3214  1210
##                   1     0        0    0     0
```

```
##
## , , reason = delay
##
##                     method
## chat_incorrect_item phone web_form chat email
##                   0  4153     1155 4778  1828
##                   1     0        0    0     0
##
## , , reason = cancellation
##
##                     method
## chat_incorrect_item phone web_form chat email
##                   0  1354      418 1635   576
##                   1     0        0    0     0
##
## , , reason = incorrect_item
##
##                     method
## chat_incorrect_item phone web_form chat email
##                   0  2843      747    0  1232
##                   1     0        0 3237     0
```

```
# Step 2: Build the ordinal logistic model including the interaction term
ordinal.model.int <- clm(nps.score ~ web_form + chat + email + delay + status + cancellation + incorrec

# Review model summary
summary(ordinal.model.int)
```

```
## formula:
## nps.score ~ web_form + chat + email + delay + status + cancellation + incorrect_item + country + cha
## data:    data
##
##  link  threshold nobs  logLik    AIC       niter max.grad cond.H
##  logit flexible  40000 -90577.31 181192.62 7(0)  7.22e-12 5.6e+02
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## web_form             -0.197043   0.031418  -6.272 3.57e-10 ***
## chat                 -0.242901   0.021997 -11.043  < 2e-16 ***
## email                -0.497631   0.027037 -18.405  < 2e-16 ***
## delay                 0.004497   0.025141   0.179    0.858
## status                0.035698   0.027478   1.299    0.194
## cancellation         -0.215385   0.033768  -6.378 1.79e-10 ***
## incorrect_item       -0.018957   0.032589  -0.582    0.561
## countryDrinkland     -0.156083   0.017471  -8.934  < 2e-16 ***
## chat_incorrect_item -1.157258   0.045684 -25.332  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Threshold coefficients:
##      Estimate Std. Error z value
## 0|1  -3.17140    0.03191 -99.385
## 1|2  -2.72022    0.02962 -91.832
## 2|3  -2.28646    0.02806 -81.491
```

```
## 3|4  -1.87243    0.02698 -69.394
## 4|5  -1.45807    0.02620 -55.651
## 5|6  -1.04174    0.02565 -40.621
## 6|7  -0.57980    0.02526 -22.953
## 7|8  -0.06078    0.02510  -2.422
## 8|9   0.52741    0.02528  20.859
## 9|10  1.21201    0.02617  46.313
```

```
# Step 3: Compare the models with and without the interaction
anova(ordinal.model, ordinal.model.int)
```

```
## Likelihood ratio tests of cumulative link models:
##
##                   formula:
## ordinal.model     nps.score ~ web_form + chat + email + delay + status + cancellation + incorrect_ite
## ordinal.model.int nps.score ~ web_form + chat + email + delay + status + cancellation + incorrect_ite
##                   link: threshold:
## ordinal.model     logit flexible
## ordinal.model.int logit flexible
##
##                   no.par    AIC logLik LR.stat df Pr(>Chisq)
## ordinal.model         18 181834 -90899
## ordinal.model.int     19 181193 -90577   643.4  1  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The model shows that the interaction is significant, and the likelihood ratio test indicates a statistically significant improvement in model fit after incorporating the interaction (chi-squared test $p < .001$ and a lower AIC of 181193 compared to 181834 for the simpler model).

Next, we proceed to check the proportional odds assumption for the updated model:

```
# Check the proportional odds assumption using the ordinal package
nominal_test(ordinal.model.int)
```

```
## Tests of nominal effects
##
## formula: nps.score ~ web_form + chat + email + delay + status + cancellation + incorrect_item + coun
##                     Df logLik    AIC     LRT  Pr(>Chi)
## <none>                 -90577 181193
## web_form             9 -90574 181204   6.719 0.6663400
## chat                 9 -90559 181175  35.754 4.383e-05 ***
## email                9 -90563 181182  28.447 0.0008028 ***
## delay                9 -90572 181201   9.899 0.3587145
## status               9 -90572 181200  10.965 0.2781007
## cancellation         9 -90572 181200  10.972 0.2776197
## incorrect_item       9 -90545 181145  65.506 1.150e-10 ***
## country              9 -90566 181188  22.319 0.0079198 **
## chat_incorrect_item  9 -90468 180992 218.703 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

After including the interaction, the proportional odds assumption is now satisfied for the contact reason delay. However, it remains violated for most other factors.

To address this, we will explore a model that allows coefficients to vary across thresholds for predictors that violate the assumption.

4.3.3.2.2 Addressing Non-Proportional Odds by Modeling Varying Coefficients

We fit a model that relaxes the proportional odds assumption for selected predictors, allowing their effects to differ at each NPS score interval, using the VGAM package.

Since delay met the proportional odds assumption and showed no significant effect, we exclude it from this model.

```
# Model allowing some predictors to have non-proportional effects
ordinal.model.int.non_proportional <- vglm(
  nps.score ~ web_form + chat + email + status + cancellation + incorrect_item + country + chat_incorre
  family = cumulative(link = logitlink, parallel = ~ country + web_form),
  data = data
)
```

```
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in eval(slot(family, "deriv")): some probabilities are very close to 0

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1
```

```
## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.
```

```
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1
```

```
## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.
```

```
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1
```

```
## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.
```

```
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1
```

```
## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.
```

```
## Warning in vglm.fitter(x = x, y = y, w = w, offset = offset, Xm2 = Xm2, :
## iterations terminated because half-step sizes are very small
```

```
## Warning in vglm.fitter(x = x, y = y, w = w, offset = offset, Xm2 = Xm2, : some
## quantities such as z, residuals, SEs may be inaccurate due to convergence at a
## half-step
```

```
summary(ordinal.model.int.non_proportional)
```

```
## Warning in eval(expr): some probabilities are very close to 0
```

```
##
## Call:
## vglm(formula = nps.score ~ web_form + chat + email + status +
##     cancellation + incorrect_item + country + chat_incorrect_item,
##     family = cumulative(link = logitlink, parallel = ~country +
##         web_form), data = data)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -0.42190    0.02160 -19.536  < 2e-16 ***
## web_form               0.19960    0.03454   5.779 7.53e-09 ***
## chat:1                -1.93165    0.03943 -48.986  < 2e-16 ***
## chat:2                -1.56548    0.03472 -45.083  < 2e-16 ***
## chat:3                -1.21348    0.03125 -38.835  < 2e-16 ***
## chat:4                -0.89505    0.02882 -31.057  < 2e-16 ***
## chat:5                -0.57340    0.02701 -21.228  < 2e-16 ***
## chat:6                -0.25066    0.02579  -9.719  < 2e-16 ***
## chat:7                 0.11700    0.02508   4.666 3.07e-06 ***
## chat:8                 0.53194    0.02507  21.216  < 2e-16 ***
```

```
## chat:9                    0.98845    0.02606  37.931  < 2e-16 ***
## chat:10                   1.52202    0.02861  53.206  < 2e-16 ***
## email:1                  -1.28375    0.04593 -27.953  < 2e-16 ***
## email:2                  -0.95743    0.04078 -23.478  < 2e-16 ***
## email:3                  -0.66537    0.03707 -17.947  < 2e-16 ***
## email:4                  -0.39665    0.03432 -11.556  < 2e-16 ***
## email:5                  -0.08349    0.03202  -2.608 0.009115 **
## email:6                   0.16013    0.03085   5.192 2.09e-07 ***
## email:7                   0.42071    0.03038  13.848  < 2e-16 ***
## email:8                   0.69152    0.03089  22.384  < 2e-16 ***
## email:9                   1.03677    0.03296  31.458  < 2e-16 ***
## email:10                  1.45832    0.03727  39.127  < 2e-16 ***
## status:1                 -1.66305    0.04467 -37.231  < 2e-16 ***
## status:2                 -1.41002    0.03968 -35.531  < 2e-16 ***
## status:3                 -1.17635    0.03576 -32.898  < 2e-16 ***
## status:4                 -0.92717    0.03234 -28.671  < 2e-16 ***
## status:5                 -0.66468    0.02952 -22.515  < 2e-16 ***
## status:6                 -0.40838    0.02760 -14.799  < 2e-16 ***
## status:7                 -0.13678    0.02644  -5.173 2.30e-07 ***
## status:8                  0.18467    0.02628   7.027 2.12e-12 ***
## status:9                  0.55337    0.02754  20.091  < 2e-16 ***
## status:10                 0.94569    0.03035  31.156  < 2e-16 ***
## cancellation:1           -1.40564    0.05638 -24.931  < 2e-16 ***
## cancellation:2           -1.15036    0.05008 -22.969  < 2e-16 ***
## cancellation:3           -0.94113    0.04561 -20.633  < 2e-16 ***
## cancellation:4           -0.67012    0.04102 -16.335  < 2e-16 ***
## cancellation:5           -0.40205    0.03757 -10.700  < 2e-16 ***
## cancellation:6           -0.13056    0.03534  -3.694 0.000221 ***
## cancellation:7            0.15890    0.03438   4.622 3.80e-06 ***
## cancellation:8            0.43331    0.03497  12.392  < 2e-16 ***
## cancellation:9            0.75349    0.03744  20.124  < 2e-16 ***
## cancellation:10           1.15615    0.04266  27.103  < 2e-16 ***
## incorrect_item:1         -2.34455    0.06492 -36.116  < 2e-16 ***
## incorrect_item:2         -1.94980    0.05477 -35.598  < 2e-16 ***
## incorrect_item:3         -1.58866    0.04750 -33.448  < 2e-16 ***
## incorrect_item:4         -1.22032    0.04181 -29.188  < 2e-16 ***
## incorrect_item:5         -0.89431    0.03803 -23.515  < 2e-16 ***
## incorrect_item:6         -0.55887    0.03541 -15.783  < 2e-16 ***
## incorrect_item:7         -0.13328    0.03365  -3.961 7.46e-05 ***
## incorrect_item:8          0.34171    0.03346  10.212  < 2e-16 ***
## incorrect_item:9          0.86785    0.03528  24.601  < 2e-16 ***
## incorrect_item:10         1.42920    0.03959  36.096  < 2e-16 ***
## countryDrinkland          0.15288    0.01811   8.441  < 2e-16 ***
## chat_incorrect_item:1     4.04837    0.08202  49.361  < 2e-16 ***
## chat_incorrect_item:2     3.63775    0.07123  51.068  < 2e-16 ***
## chat_incorrect_item:3     3.18739    0.06391  49.869  < 2e-16 ***
## chat_incorrect_item:4     2.68196    0.05870  45.692  < 2e-16 ***
## chat_incorrect_item:5     2.17923    0.05539  39.340  < 2e-16 ***
## chat_incorrect_item:6     1.66879    0.05341  31.244  < 2e-16 ***
## chat_incorrect_item:7     1.05517    0.05260  20.061  < 2e-16 ***
## chat_incorrect_item:8     0.47888    0.05423   8.830  < 2e-16 ***
## chat_incorrect_item:9    -0.04240    0.05976  -0.709 0.478027
## chat_incorrect_item:10   -0.56450    0.07099  -7.952 1.83e-15 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of linear predictors:  10
##
## Names of linear predictors: logitlink(P[Y<=1]), logitlink(P[Y<=2]),
## logitlink(P[Y<=3]), logitlink(P[Y<=4]), logitlink(P[Y<=5]), logitlink(P[Y<=6]),
## logitlink(P[Y<=7]), logitlink(P[Y<=8]), logitlink(P[Y<=9]), logitlink(P[Y<=10])
##
## Residual deviance: 4942097 on 399937 degrees of freedom
##
## Log-likelihood: NA on 399937 degrees of freedom
##
## Number of Fisher scoring iterations: 2
##
## Warning: Hauck-Donner effect detected in the following estimate(s):
## 'incorrect_item:1'
```

The previous model did not converge properly and generated warnings about over-predicting certain outcomes, resulting in unstable parameter estimates.

To address this, we fit a simpler model that only relaxes the proportional odds assumption for the predictors with the strongest violations: the contact reason incorrect item, and the interaction term of incorrect items attended by chat.

```
# Model with fewer predictors allowed to have non-proportional effects
ordinal.model.int.non_proportional2 <- vglm(
  nps.score ~ web_form + chat + email + status + cancellation + incorrect_item + country + chat_incorrec
  family = cumulative(link = logitlink, parallel = ~ country + web_form + chat + email + status + cancel
  data = data
)
```

```
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in eval(slot(family, "deriv")): some probabilities are very close to 0

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.
```

```
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1
```

```
## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.
```

```
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in vglm.fitter(x = x, y = y, w = w, offset = offset, Xm2 = Xm2, :
## iterations terminated because half-step sizes are very small

## Warning in vglm.fitter(x = x, y = y, w = w, offset = offset, Xm2 = Xm2, : some
## quantities such as z, residuals, SEs may be inaccurate due to convergence at a
## half-step
```

```r
summary(ordinal.model.int.non_proportional2)
```

```
## Warning in eval(expr): some probabilities are very close to 0
```

```
##
## Call:
## vglm(formula = nps.score ~ web_form + chat + email + status +
##     cancellation + incorrect_item + country + chat_incorrect_item,
##     family = cumulative(link = logitlink, parallel = ~country +
##         web_form + chat + email + status + cancellation), data = data)
##
## Coefficients:
```

```
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -0.42190    0.02144 -19.673  < 2e-16 ***
## web_form                 0.19959    0.03506   5.693 1.25e-08 ***
## chat                     0.24346    0.02536   9.600  < 2e-16 ***
## email                    0.49162    0.02984  16.473  < 2e-16 ***
## status                  -0.03397    0.02672  -1.271    0.204
## cancellation             0.21551    0.03491   6.174 6.67e-10 ***
## incorrect_item:1        -2.79815    0.06881 -40.666  < 2e-16 ***
## incorrect_item:2        -2.32003    0.05670 -40.920  < 2e-16 ***
## incorrect_item:3        -1.88427    0.04841 -38.921  < 2e-16 ***
## incorrect_item:4        -1.44728    0.04227 -34.238  < 2e-16 ***
## incorrect_item:5        -1.04125    0.03819 -27.264  < 2e-16 ***
## incorrect_item:6        -0.64357    0.03548 -18.137  < 2e-16 ***
## incorrect_item:7        -0.15140    0.03366  -4.498 6.85e-06 ***
## incorrect_item:8         0.39278    0.03342  11.754  < 2e-16 ***
## incorrect_item:9         1.00712    0.03537  28.476  < 2e-16 ***
## incorrect_item:10        1.67619    0.04055  41.336  < 2e-16 ***
## countryDrinkland         0.15288    0.01952   7.833 4.77e-15 ***
## chat_incorrect_item:1    2.32686    0.07962  29.226  < 2e-16 ***
## chat_incorrect_item:2    2.19904    0.06878  31.973  < 2e-16 ***
## chat_incorrect_item:3    2.02606    0.06201  32.675  < 2e-16 ***
## chat_incorrect_item:4    1.77040    0.05747  30.807  < 2e-16 ***
## chat_incorrect_item:5    1.50932    0.05478  27.552  < 2e-16 ***
## chat_incorrect_item:6    1.25937    0.05332  23.620  < 2e-16 ***
## chat_incorrect_item:7    0.94683    0.05280  17.932  < 2e-16 ***
## chat_incorrect_item:8    0.71629    0.05438  13.172  < 2e-16 ***
## chat_incorrect_item:9    0.56332    0.05954   9.462  < 2e-16 ***
## chat_incorrect_item:10   0.46708    0.07030   6.644 3.05e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of linear predictors:  10
##
## Names of linear predictors: logitlink(P[Y<=1]), logitlink(P[Y<=2]),
## logitlink(P[Y<=3]), logitlink(P[Y<=4]), logitlink(P[Y<=5]), logitlink(P[Y<=6]),
## logitlink(P[Y<=7]), logitlink(P[Y<=8]), logitlink(P[Y<=9]), logitlink(P[Y<=10])
##
## Residual deviance: 17388117 on 399973 degrees of freedom
##
## Log-likelihood: NA on 399973 degrees of freedom
##
## Number of Fisher scoring iterations: 2
##
## Warning: Hauck-Donner effect detected in the following estimate(s):
## 'chat', 'incorrect_item:1'
```

This model still did not fit properly and produced warnings indicating unstable parameter estimates. Next, we simplify further by assuming non-proportional odds only for the interaction factor incorrect items attended by chat:

```
# Model where only the interaction is assumed to violate proportional odds
ordinal.model.int.non_proportional3 <- vglm(
  nps.score ~ web_form + chat + email + status + cancellation + incorrect_item + country + chat_incorre
  family = cumulative(link = logitlink, parallel = ~ country + web_form + chat + email + status + cancel
```

```r
  data = data
)
```

```
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in eval(slot(family, "deriv")): some probabilities are very close to 0

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.
```

```
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1
```

```
## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.

## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1

## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.
```

```
## Warning in Deviance.categorical.data.vgam(mu = mu, y = y, w = w, residuals =
## residuals, : fitted values close to 0 or 1
```

```
## Warning in slot(family, "validparams")(eta, y, extra = extra): It seems that
## the nonparallelism assumption has resulted in intersecting linear/additive
## predictors. Try propodds() or fitting a partial nonproportional odds model or
## choosing some other link function, etc.
```

```
## Warning in vglm.fitter(x = x, y = y, w = w, offset = offset, Xm2 = Xm2, :
## iterations terminated because half-step sizes are very small
```

```
## Warning in vglm.fitter(x = x, y = y, w = w, offset = offset, Xm2 = Xm2, : some
## quantities such as z, residuals, SEs may be inaccurate due to convergence at a
## half-step
```

```
summary(ordinal.model.int.non_proportional3)
```

```
## Warning in eval(expr): some probabilities are very close to 0
```

```
##
## Call:
## vglm(formula = nps.score ~ web_form + chat + email + status +
##     cancellation + incorrect_item + country + chat_incorrect_item,
##     family = cumulative(link = logitlink, parallel = ~country +
##         web_form + chat + email + status + cancellation + incorrect_item),
##     data = data)
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -0.42189    0.02908 -14.506  < 2e-16 ***
## web_form                0.19959    0.03854   5.178 2.24e-07 ***
## chat                    0.24345    0.02991   8.138 4.00e-16 ***
## email                   0.49162    0.03392  14.493  < 2e-16 ***
## status                 -0.03397    0.02866  -1.185  0.23584
## cancellation            0.21551    0.03657   5.892 3.80e-09 ***
## incorrect_item          0.01861    0.03617   0.514  0.60691
## countryDrinkland        0.15288    0.02130   7.176 7.18e-13 ***
## chat_incorrect_item:1  -0.48990    0.05311  -9.224  < 2e-16 ***
## chat_incorrect_item:2  -0.13960    0.05227  -2.671  0.00757 **
## chat_incorrect_item:3   0.12318    0.05213   2.363  0.01813 *
## chat_incorrect_item:4   0.30451    0.05227   5.826 5.69e-09 ***
## chat_incorrect_item:5   0.44946    0.05252   8.557  < 2e-16 ***
## chat_incorrect_item:6   0.59719    0.05292  11.285  < 2e-16 ***
## chat_incorrect_item:7   0.77682    0.05359  14.497  < 2e-16 ***
## chat_incorrect_item:8   1.09046    0.05529  19.722  < 2e-16 ***
## chat_incorrect_item:9   1.55183    0.05925  26.193  < 2e-16 ***
## chat_incorrect_item:10  2.12466    0.06719  31.624  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of linear predictors:  10
##
```

```
## Names of linear predictors: logitlink(P[Y<=1]), logitlink(P[Y<=2]),
## logitlink(P[Y<=3]), logitlink(P[Y<=4]), logitlink(P[Y<=5]), logitlink(P[Y<=6]),
## logitlink(P[Y<=7]), logitlink(P[Y<=8]), logitlink(P[Y<=9]), logitlink(P[Y<=10])
##
## Residual deviance: 19996065 on 399982 degrees of freedom
##
## Log-likelihood: NA on 399982 degrees of freedom
##
## Number of Fisher scoring iterations: 2
##
## Warning: Hauck-Donner effect detected in the following estimate(s):
## 'chat', 'email'
```

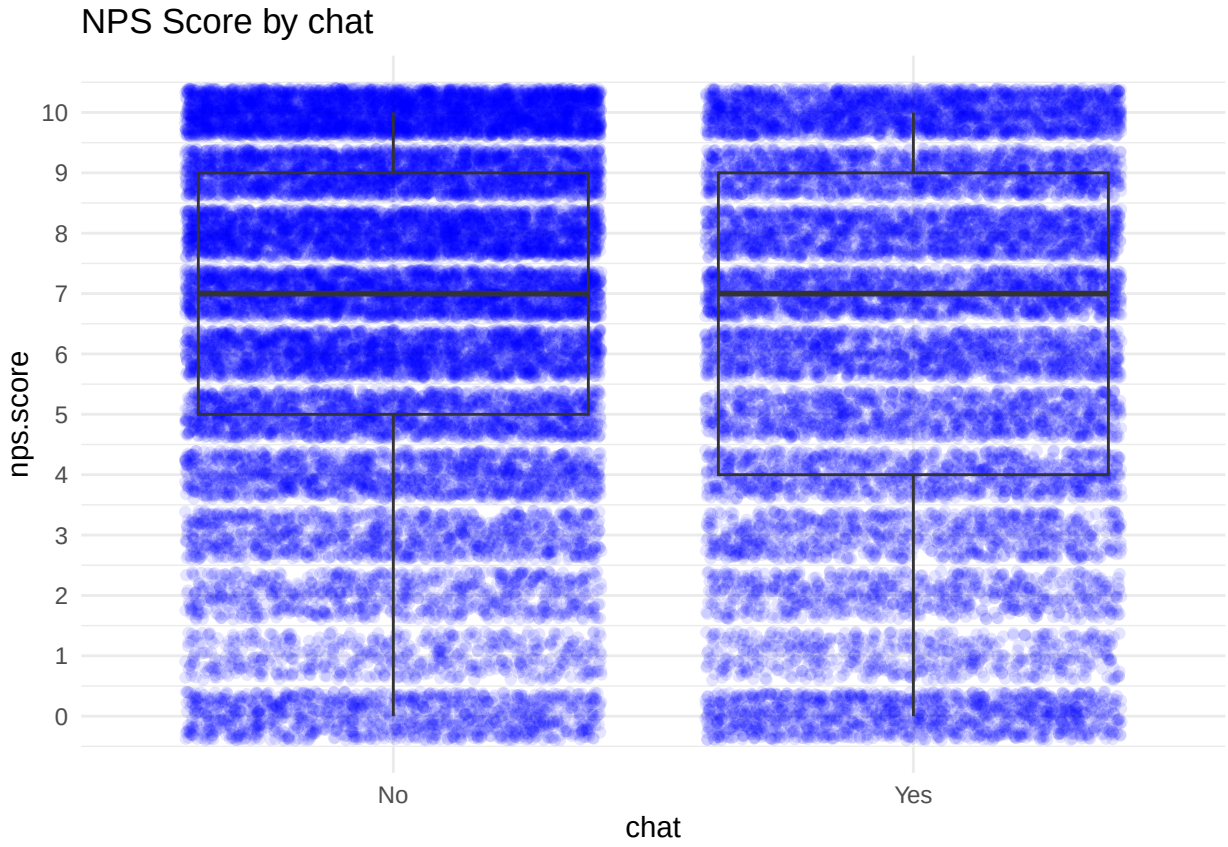
This model still did not fit adequately, suggesting that our data may not conform well to the non-proportional odds assumptions.

Given this, we explore whether grouping categories of the NPS scale and focusing on predicting one specific group might help resolve this issue.

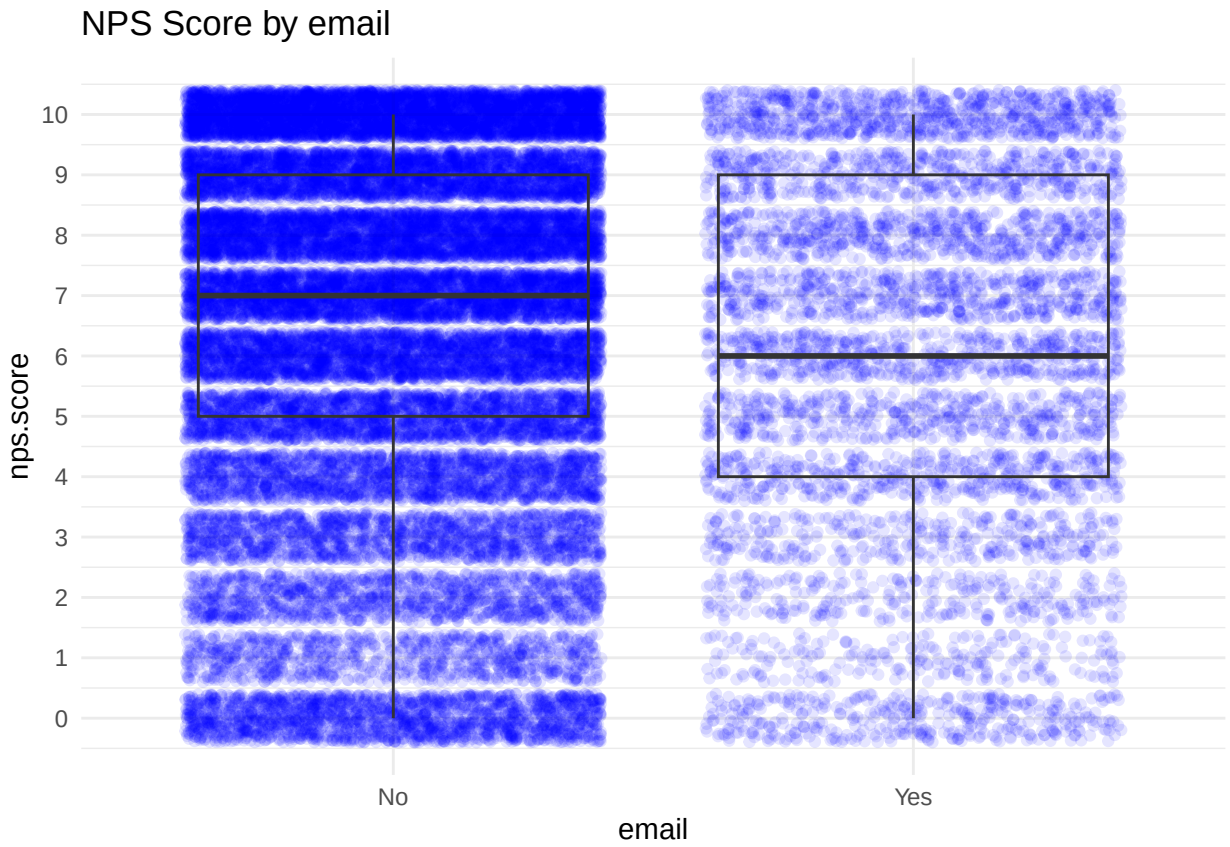
4.3.3.2.3. Identifying Sections in the NPS Scale with Common Non-Proportional Odds

To better understand how non-proportional odds manifest across the NPS scale, we visualize jitter and boxplots using the ggplot2 package:

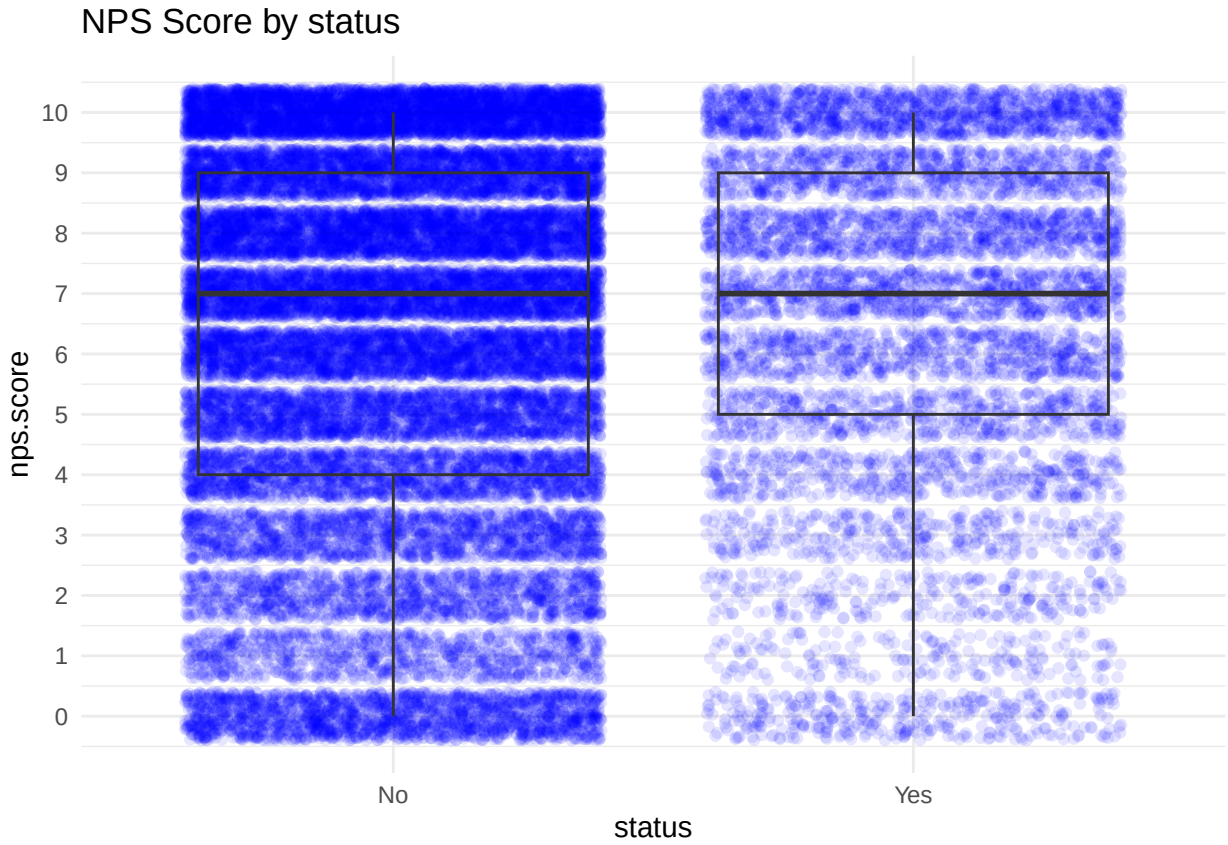
```r
# Reusable function for combining jitter plots and boxplots in dicotomous factors
f.nps.jitter_box <- function(my.data, my.factor, my.dv) {
  ggplot(my.data, aes(x = factor(.data[[my.factor]]), y = as.numeric(as.character(.data[[my.dv]])))) +
    geom_jitter(width = 0.4, alpha = 0.1, color = "blue") +  # Adjusted width and alpha for clarity
    geom_boxplot(alpha = 0, outlier.shape = NA) +  # Transparent boxplots to show quartiles
    labs(title = paste("NPS Score by", my.factor),
         y = my.dv,
         x = my.factor) +
    theme_minimal() +
    scale_y_continuous(breaks = 0:10) +  # Show all NPS scores on y-axis
    scale_x_discrete(labels = c("No", "Yes"))
}

# Apply the function to relevant factors
f.nps.jitter_box(data, "chat", "nps.score")
```



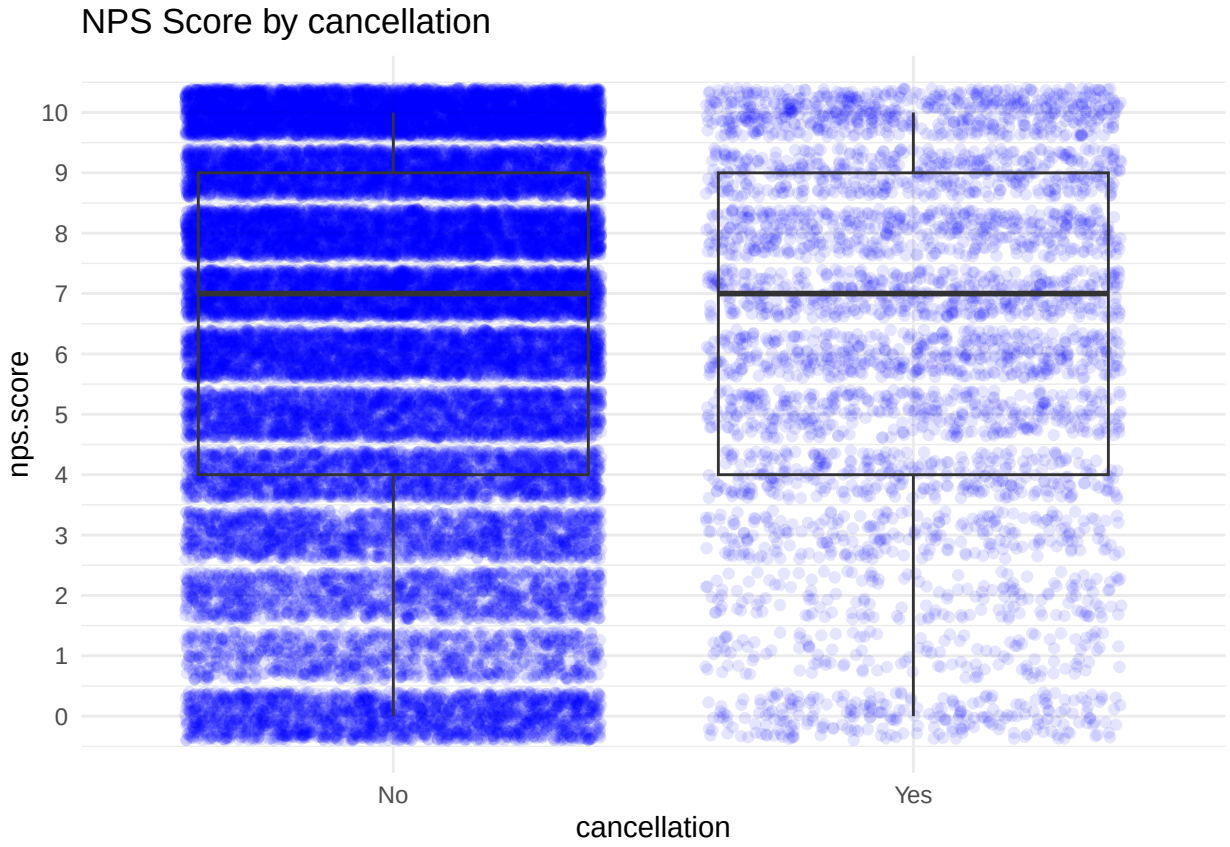

```
f.nps.jitter_box(data, "email", "nps.score")
```



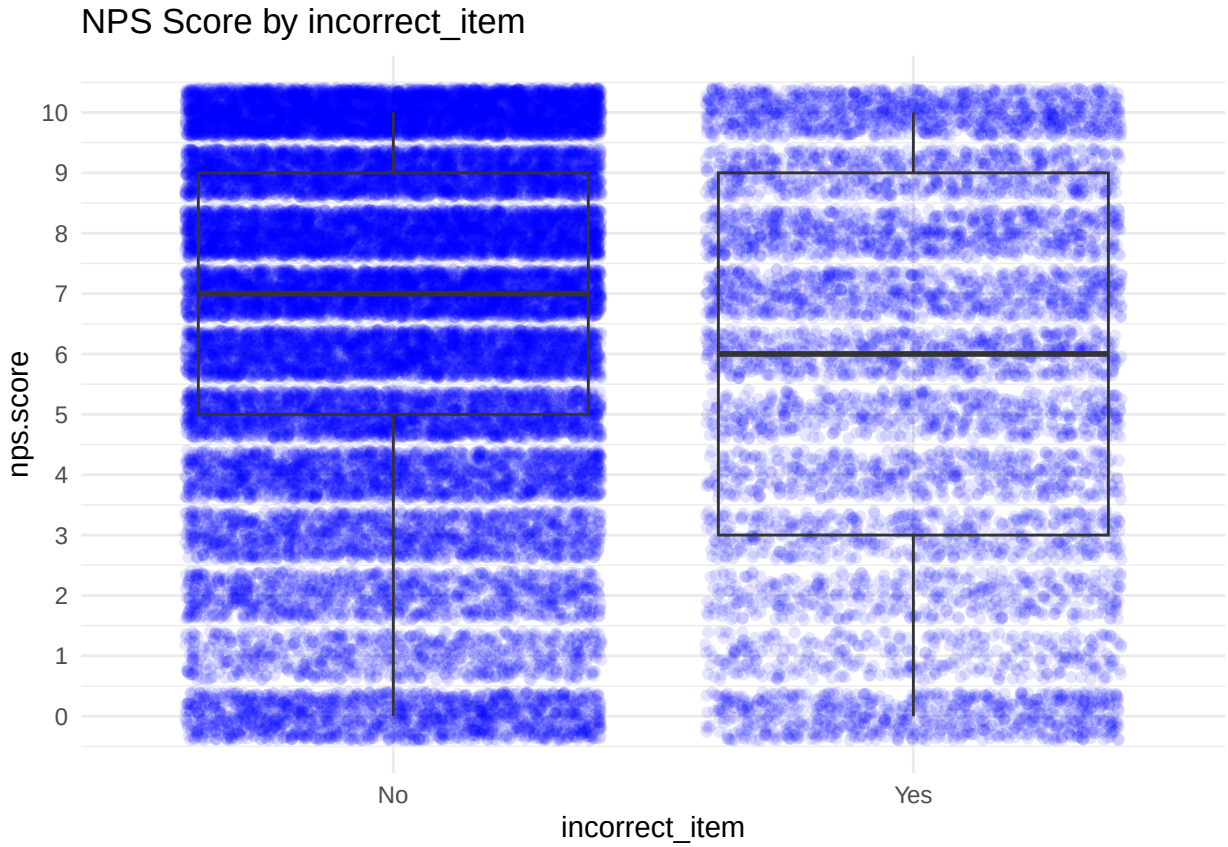

```
f.nps.jitter_box(data, "status", "nps.score")
```



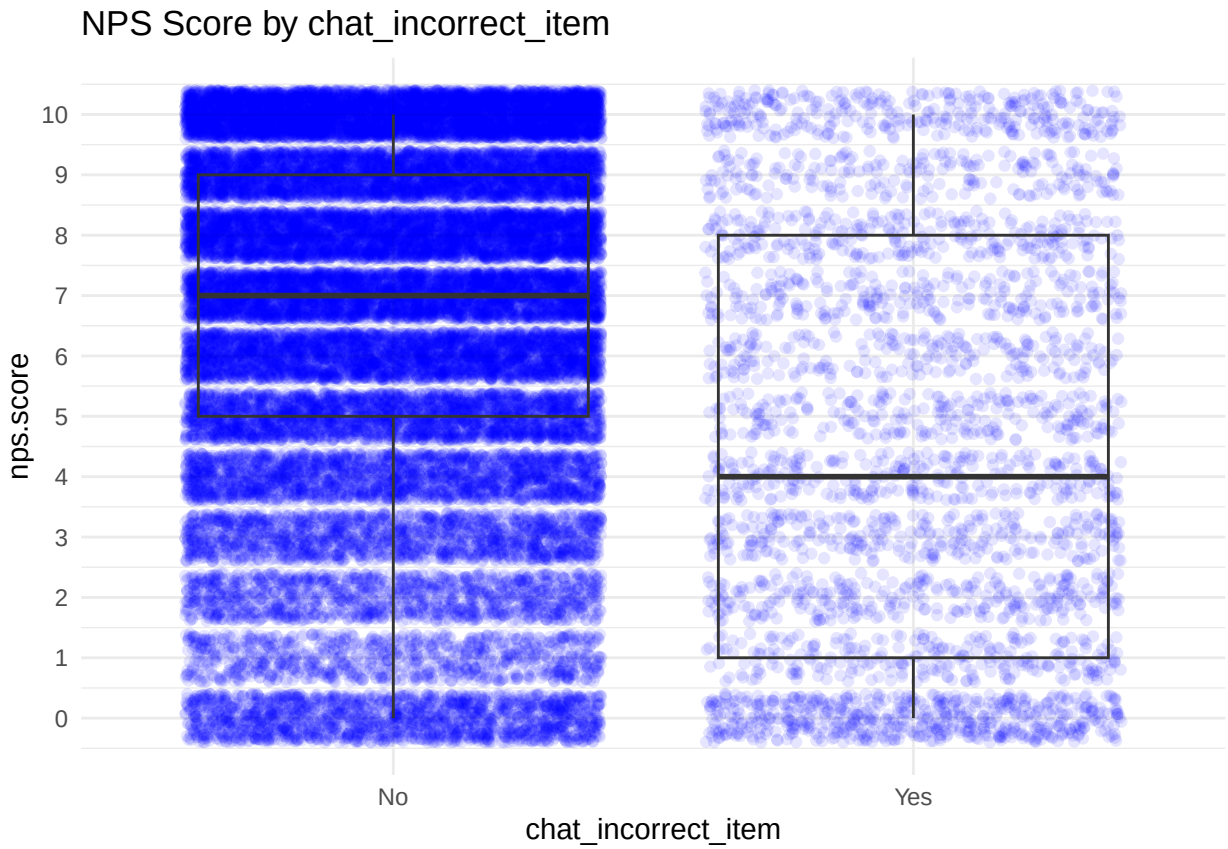

```
f.nps.jitter_box(data, "cancellation", "nps.score")
```




```
f.nps.jitter_box(data, "incorrect_item", "nps.score")
```




```
f.nps.jitter_box(data, "chat_incorrect_item", "nps.score")
```



The plots show that factors violating the proportional odds assumption have stronger effects in the lower range of the NPS scale—that is, differences are more pronounced among the lower quartile and below. This suggests these factors primarily influence the formation of detractors.

To address the non-proportional odds issue and focus on the NPS scale segment where these factors have the greatest impact, we will next explore a binomial logistic model predicting the presence of detractors (those whose NPS scores are 6 or below).

4.3.4. Binomial Logistic Regression Model to Predict Detractors

Given the limitations of the ordinal model due to violated proportional odds assumptions, we consider a binomial logistic regression model more appropriate for predicting detractors. This approach is also more parsimonious and efficient, focusing specifically on the group of NPS scores most affected by our factors.

Additionally, prior analyses confirmed that the non-collinearity assumptions are met.

We then proceed to specify and fit our binomial model:

```
# Step 1: Create a binary variable for detractors
data$detractor <- NA
data$detractor[data$nps.segment == "detractor"] <- 1
data$detractor[data$nps.segment != "detractor"] <- 0

# Step 2: Fit the binomial logistic regression model
binomial.model <- glm(detractor ~ web_form + chat + email + status + cancellation + incorrect_item + cou

summary(binomial.model)
```

```
##
## Call:
## glm(formula = detractor ~ web_form + chat + email + status +
##     cancellation + incorrect_item + country + chat_incorrect_item,
##     family = "binomial", data = data)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.60896    0.02323 -26.214  < 2e-16 ***
## web_form             0.22084    0.03670   6.017 1.77e-09 ***
## chat                 0.25318    0.02584   9.796  < 2e-16 ***
## email                0.53630    0.03117  17.207  < 2e-16 ***
## status              -0.04188    0.02696  -1.553    0.120
## cancellation         0.25598    0.03497   7.321 2.46e-13 ***
## incorrect_item       0.00231    0.03410   0.068    0.946
## countryDrinkland     0.18005    0.02042   8.816  < 2e-16 ***
## chat_incorrect_item  0.97725    0.05296  18.452  < 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: 54989  on 39999  degrees of freedom
## Residual deviance: 53813  on 39991  degrees of freedom
## AIC: 53831
##
## Number of Fisher Scoring iterations: 4
```

The results are consistent with those from the ordinal model, but now with stable coefficients specifically focused on predicting the likelihood of being a detractor.

However, the model's parsimony can still be improved. The contact reason status does not show any statistically significant effect, so we test whether the model can be simplified by removing this variable:

```r
# Fit the binomial model excluding the non-significant status factor
binomial.model.c <- glm(detractor ~ web_form + chat + email + cancellation + incorrect_item + country +
                        data = data, family = "binomial")

summary(binomial.model.c)
```

```
##
## Call:
## glm(formula = detractor ~ web_form + chat + email + cancellation +
##     incorrect_item + country + chat_incorrect_item, family = "binomial",
##     data = data)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.62103    0.02190 -28.351  < 2e-16 ***
## web_form             0.22079    0.03670   6.016 1.78e-09 ***
## chat                 0.25314    0.02584   9.795  < 2e-16 ***
## email                0.53627    0.03117  17.206  < 2e-16 ***
## cancellation         0.26797    0.03411   7.857 3.94e-15 ***
## incorrect_item       0.01429    0.03322   0.430    0.667
```

```
## countryDrinkland     0.18025    0.02042   8.826  < 2e-16 ***
## chat_incorrect_item  0.97729    0.05296  18.453  < 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: 54989  on 39999  degrees of freedom
## Residual deviance: 53815  on 39992  degrees of freedom
## AIC: 53831
##
## Number of Fisher Scoring iterations: 4
```

```
# Compare the models with and without the status factor
anova(binomial.model.c, binomial.model, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: detractor ~ web_form + chat + email + cancellation + incorrect_item +
##     country + chat_incorrect_item
## Model 2: detractor ~ web_form + chat + email + status + cancellation +
##     incorrect_item + country + chat_incorrect_item
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1     39992      53815
## 2     39991      53813  1   2.4151   0.1202
```

The likelihood ratio test confirms that removing the status variable does not significantly reduce model fit. Therefore, we simplify the model by excluding it. This allows us to focus on the most relevant and actionable predictors of detractor status.

We are now ready to derive meaningful metrics from the model to understand the extent to which the presence of detractors is associated with specific customer service factors.

4.3.4.1. Creating Metrics to Interpret the Model To interpret the results of our binomial logistic regression, we will compute several complementary metrics for each of the factors:

- **Predicted percentage of detractors** for the presence of each customer service factor (while holding others constant).
- **Differences in predicted percentages** compared to the baseline.
- **Relative probabilities** (risk ratios).
- **Odds ratios**.

To calculate the predicted proportions, we first need to build a design matrix that represents the isolated presence of each factor (and the intercept) in a way compatible with our model specification. This will allow us to generate predictions for each scenario separately.

```
#####
# Design Matrix for Predictions
#####
```

```r
# Extract the factor names from the model, with and without the intercept
f.names.interc <- names(coef(binomial.model.c))
f.names <- f.names.interc[-1]

# Create a matrix with 1s on the diagonal (each case represents one factor set to 1)
my.matrix.df <- as.data.frame(diag(length(f.names.interc)))
my.matrix.df[, 1] <- f.names.interc
names(my.matrix.df) <- c("case", f.names)

# Correct the interaction term so its components are also activated
my.matrix.df$chat[my.matrix.df$case == "chat_incorrect_item"] <- 1
my.matrix.df$incorrect_item[my.matrix.df$case == "chat_incorrect_item"] <- 1

# Ensure factor variables match the dataset format: we need to do it for country.
names(my.matrix.df)[names(my.matrix.df) == "countryDrinkland"] <- "country"
my.matrix.df$country[my.matrix.df$country == 1] <- "Drinkland"
my.matrix.df$country[my.matrix.df$country == 0] <- "Eatland"
my.matrix.df$country <- as.factor(my.matrix.df$country)

# Quick check of the structure
str(my.matrix.df)
```

```
## 'data.frame':    8 obs. of  8 variables:
##  $ case               : chr  "(Intercept)" "web_form" "chat" "email" ...
##  $ web_form           : num  0 1 0 0 0 0 0 0
##  $ chat               : num  0 0 1 0 0 0 0 1
##  $ email              : num  0 0 0 1 0 0 0 0
##  $ cancellation       : num  0 0 0 0 1 0 0 0
##  $ incorrect_item     : num  0 0 0 0 0 1 0 1
##  $ country            : Factor w/ 2 levels "Drinkland","Eatland": 2 2 2 2 2 2 1 2
##  $ chat_incorrect_item: num  0 0 0 0 0 0 0 1
```

We then create a function to compute the metrics and their 95% confidence intervals. Confidence intervals for differences in predicted proportions are estimated via parametric bootstrap simulations

```r
#####
#Function to obtain key interpretation metrics for a binomial logistic model
#####


f.metrics.binomial <- function(my.model, my.matrix){

  #--------1. Preliminary Checks:

  # Making sure the model is glm binomial
  if (!inherits(my.model, "glm") || my.model$family$family != "binomial") {
    stop("The function requires a binomial model.")
  }


  #--------2. Odds Ratios

  # Obtaining the coefficients and confidence intervals
```

```r
coefs <- coef(my.model)
ci <- confint(my.model)
intercept <- coefs[1]

# Exponentiation of the coefficients and their CIs to obtain odds ratios
or <- exp(coefs)
or_ci <- exp(ci)


#--------3. Relative Probabilities (Risk Ratios)

# Calculating the base probability from the intercept
p0 <- or[1] / (1 + or[1])

# Obtaining relative probabilities from risk ratios and base probability
rp <- or / ((1 - p0) + (p0 * or))
rp_ci_low <- or_ci[, 1] / ((1 - p0) + (p0 * or_ci[, 1]))
rp_ci_high <- or_ci[, 2] / ((1 - p0) + (p0 * or_ci[, 2]))


#--------4. Predicted Proportions

# Extract log-odds and their sd for the individual presence of the factors (using the specified matri
pred <- predict(my.model, newdata = my.matrix, type = "link", se.fit = TRUE)

#Calculate confidence intervals in log-odds
lower_logit <- pred$fit - 1.96 * pred$se.fit
upper_logit <- pred$fit + 1.96 * pred$se.fit

#Specify a function to transform log-odds to probabilities
f.inv_logit <- function(x) {1 / (1 + exp(-x))}

#Apply the function to the log-odds and the limits of their ci
predicted_prob <- f.inv_logit(pred$fit) # Predicted probabilities
lower_predicted_prob <- f.inv_logit(lower_logit) # Lower limit of CI for predicted probabilities
upper_predicted_prob <- f.inv_logit(upper_logit) # Upper limit of CI for predicted probabilities


#--------5. Differences in Predicted Probabilities via Bootstrap

# Simulate a distribution of log-odds, for example with 10000 cases
set.seed(123)
simulated_log_odds <- as.data.frame(matrix(
rnorm(10000 * length(pred$fit), mean = pred$fit, sd = pred$se.fit),
ncol = length(pred$fit), byrow = TRUE
))

# Transform logg-odds to expected probabilities (using the formula specified above)
simulated_probs <- f.inv_logit(simulated_log_odds)

# Calculate differences from the first column, which correspond to the intercept probabilities
diffs <- apply(simulated_probs, 2, function(col){
col - simulated_probs[[1]]
```

```
  })

  # Calculate the means of the differences and 95% CI with percentiles
  prob_diff <- colMeans(diffs)
  lower_prob_diff  <- apply(diffs, 2, quantile, probs = 0.025)
  upper_prob_diff <- apply(diffs, 2, quantile, probs = 0.975)


  #-------- 6. Combine Results

  results <- data.frame(
    Term = names(rp),
    Prediction = as.numeric(predicted_prob)*100,
    Lower_Prediction = as.numeric(lower_predicted_prob)*100,
    Upper_Prediction = as.numeric(upper_predicted_prob)*100,
    Predicted_Change = as.numeric(prob_diff)*100,
    Lower_Predicted_Change = as.numeric(lower_prob_diff)*100,
    Upper_Predicted_Change = as.numeric(upper_prob_diff)*100,
    Relative_Probability = as.numeric(rp),
    Lower_Relative_Probability = as.numeric(rp_ci_low),
    Upper_Relative_Probability = as.numeric(rp_ci_high),
    Odds_Ratio = as.numeric(or),
    Lower_Odds_Ratio = as.numeric(or_ci[,1]),
    Upper_Odds_Ratio = as.numeric(or_ci[,2])
)

  # Remove meaningless metrics for intercept
  results[1, c("Predicted_Change",
               "Lower_Predicted_Change",
               "Upper_Predicted_Change",
               "Relative_Probability",
               "Lower_Relative_Probability",
               "Upper_Relative_Probability",
               "Odds_Ratio",
               "Lower_Odds_Ratio",
               "Upper_Odds_Ratio")] <- NA_real_


  # Return the results
  return(results)
}
```

Now we can apply the function to our final binomial model in order to generate the set of metrics described above.

```
# Apply the function to the final binomial model
model.metrics.df <- f.metrics.binomial(binomial.model.c, my.matrix.df)
```

```
## Waiting for profiling to be done...
```

```
# Display the resulting metrics
model.metrics.df
```

```
##                  Term Prediction Lower_Prediction Upper_Prediction
## 1         (Intercept)   34.95480         33.98506         35.93715
## 2            web_form   40.12560         38.50178         41.77138
## 3                chat   40.90509         39.90661         41.91113
## 4               email   47.88224         46.47219         49.29567
## 5        cancellation   41.26410         39.53295         43.01712
## 6      incorrect_item   35.28039         33.81002         36.77916
## 7    countryDrinkland   39.15569         38.15204         40.16859
## 8 chat_incorrect_item   65.10563         63.36170         66.81003
##   Predicted_Change Lower_Predicted_Change Upper_Predicted_Change
## 1               NA                    NA                    NA
## 2        5.1654411              3.286841              7.068182
## 3        5.9443321              4.541586              7.331843
## 4       12.9220210             11.203495             14.649027
## 5        6.2982647              4.347519              8.265228
## 6        0.3301661             -1.428853              2.169299
## 7        4.1972629              2.803931              5.607530
## 8       30.1460325             28.144926             32.142751
##   Relative_Probability Lower_Relative_Probability Upper_Relative_Probability
## 1                   NA                         NA                         NA
## 2             1.147928                  1.0988167                   1.197658
## 3             1.170228                  1.1353773                   1.205422
## 4             1.369833                  1.3263056                   1.413502
## 5             1.180499                  1.1344224                   1.227097
## 6             1.009314                  0.9671599                   1.052196
## 7             1.120180                  1.0930251                   1.147582
## 8             1.682558                  1.6101524                   1.753902
##   Odds_Ratio Lower_Odds_Ratio Upper_Odds_Ratio
## 1         NA               NA               NA
## 2   1.247064        1.1604400         1.339992
## 3   1.288059        1.2244574         1.355003
## 4   1.709610        1.6083338         1.817337
## 5   1.307305        1.2227509         1.397669
## 6   1.014392        0.9503875         1.082562
## 7   1.197521        1.1505418         1.246436
## 8   2.657235        2.3956728         2.948434
```

To facilitate interpretation, we rearrange the results in descending order of the predicted percentage change, keeping the intercept at the top of the table:

```
# Arrange results by predicted percentage change, keeping the intercept at the top
int.df <- model.metrics.df[1, ]
factors.df <- arrange(model.metrics.df[-1, ], desc(Predicted_Change))
ord.model.metrics.df <- bind_rows(int.df, factors.df)

# Display the reordered results
ord.model.metrics.df
```

```
##                  Term Prediction Lower_Prediction Upper_Prediction
## 1         (Intercept)   34.95480         33.98506         35.93715
## 2 chat_incorrect_item   65.10563         63.36170         66.81003
## 3               email   47.88224         46.47219         49.29567
## 4        cancellation   41.26410         39.53295         43.01712
## 5                chat   40.90509         39.90661         41.91113
```

```
## 6            web_form   40.12560         38.50178         41.77138
## 7    countryDrinkland   39.15569         38.15204         40.16859
## 8      incorrect_item   35.28039         33.81002         36.77916
##   Predicted_Change Lower_Predicted_Change Upper_Predicted_Change
## 1               NA                    NA                    NA
## 2       30.1460325             28.144926             32.142751
## 3       12.9220210             11.203495             14.649027
## 4        6.2982647              4.347519              8.265228
## 5        5.9443321              4.541586              7.331843
## 6        5.1654411              3.286841              7.068182
## 7        4.1972629              2.803931              5.607530
## 8        0.3301661             -1.428853              2.169299
##   Relative_Probability Lower_Relative_Probability Upper_Relative_Probability
## 1                   NA                         NA                         NA
## 2             1.682558                  1.6101524                   1.753902
## 3             1.369833                  1.3263056                   1.413502
## 4             1.180499                  1.1344224                   1.227097
## 5             1.170228                  1.1353773                   1.205422
## 6             1.147928                  1.0988167                   1.197658
## 7             1.120180                  1.0930251                   1.147582
## 8             1.009314                  0.9671599                   1.052196
##   Odds_Ratio Lower_Odds_Ratio Upper_Odds_Ratio
## 1         NA               NA               NA
## 2   2.657235        2.3956728         2.948434
## 3   1.709610        1.6083338         1.817337
## 4   1.307305        1.2227509         1.397669
## 5   1.288059        1.2244574         1.355003
## 6   1.247064        1.1604400         1.339992
## 7   1.197521        1.1505418         1.246436
## 8   1.014392        0.9503875         1.082562
```

We can further enhance interpretability by adding clear, reader-oriented labels for each factor in the results table:

```
# Add descriptive labels for each factor
ord.model.metrics.df$Label <- NA_character_
ord.model.metrics.df$Label[ord.model.metrics.df$Term == "(Intercept)"] <-
  "Baseline: Typical issues handled via phone in Eatland"
ord.model.metrics.df$Label[ord.model.metrics.df$Term == "chat_incorrect_item"] <-
  "Chat support for incorrect item issues"
ord.model.metrics.df$Label[ord.model.metrics.df$Term == "email"] <-
  "Support via email"
ord.model.metrics.df$Label[ord.model.metrics.df$Term == "chat"] <-
  "Support via chat"
ord.model.metrics.df$Label[ord.model.metrics.df$Term == "web_form"] <-
  "Support via web form"
ord.model.metrics.df$Label[ord.model.metrics.df$Term == "countryDrinkland"] <-
  "Support in Drinkland"
ord.model.metrics.df$Label[ord.model.metrics.df$Term == "cancellation"] <-
  "Support for cancellation issues"
ord.model.metrics.df$Label[ord.model.metrics.df$Term == "incorrect_item"] <-
  "Support for incorrect item issues"

# Quick check to ensure labels are as expected
```

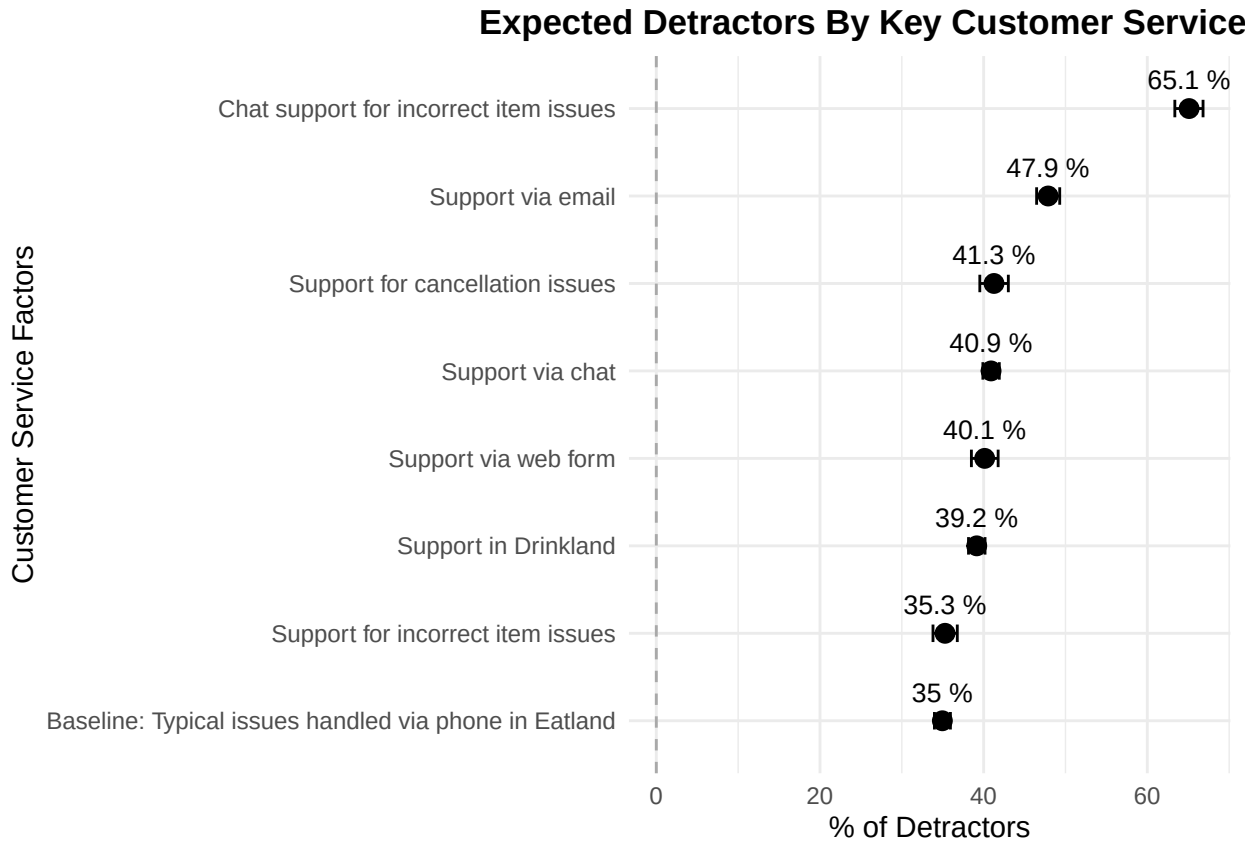
```
ord.model.metrics.df
```

```
##                  Term Prediction Lower_Prediction Upper_Prediction
## 1         (Intercept)   34.95480         33.98506         35.93715
## 2 chat_incorrect_item   65.10563         63.36170         66.81003
## 3               email   47.88224         46.47219         49.29567
## 4        cancellation   41.26410         39.53295         43.01712
## 5                chat   40.90509         39.90661         41.91113
## 6            web_form   40.12560         38.50178         41.77138
## 7    countryDrinkland   39.15569         38.15204         40.16859
## 8      incorrect_item   35.28039         33.81002         36.77916
##   Predicted_Change Lower_Predicted_Change Upper_Predicted_Change
## 1               NA                    NA                    NA
## 2       30.1460325              28.144926             32.142751
## 3       12.9220210              11.203495             14.649027
## 4        6.2982647               4.347519              8.265228
## 5        5.9443321               4.541586              7.331843
## 6        5.1654411               3.286841              7.068182
## 7        4.1972629               2.803931              5.607530
## 8        0.3301661              -1.428853              2.169299
##   Relative_Probability Lower_Relative_Probability Upper_Relative_Probability
## 1                   NA                         NA                         NA
## 2             1.682558                  1.6101524                   1.753902
## 3             1.369833                  1.3263056                   1.413502
## 4             1.180499                  1.1344224                   1.227097
## 5             1.170228                  1.1353773                   1.205422
## 6             1.147928                  1.0988167                   1.197658
## 7             1.120180                  1.0930251                   1.147582
## 8             1.009314                  0.9671599                   1.052196
##   Odds_Ratio Lower_Odds_Ratio Upper_Odds_Ratio
## 1         NA               NA               NA
## 2   2.657235        2.3956728         2.948434
## 3   1.709610        1.6083338         1.817337
## 4   1.307305        1.2227509         1.397669
## 5   1.288059        1.2244574         1.355003
## 6   1.247064        1.1604400         1.339992
## 7   1.197521        1.1505418         1.246436
## 8   1.014392        0.9503875         1.082562
##                                                 Label
## 1 Baseline: Typical issues handled via phone in Eatland
## 2                Chat support for incorrect item issues
## 3                                    Support via email
## 4                       Support for cancellation issues
## 5                                     Support via chat
## 6                                  Support via web form
## 7                                  Support in Drinkland
## 8                     Support for incorrect item issues
```

Lastly, we visually represent the predicted proportions of detractors to clearly identify which key customer service factors are associated with higher likelihoods of detractor status.

```
# Create the dot-and-whisker plot for the predictions
ggplot(ord.model.metrics.df, aes(x = Prediction,
```

```
                    y = reorder(Label, Prediction))) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "darkgrey") +
  geom_errorbarh(aes(xmin = Lower_Prediction, xmax = Upper_Prediction), height = 0.2) +
  geom_point(size = 3) +
    geom_text(aes(label = paste(round(Prediction,1), "%")),
              vjust = -1, size = 3.5) +
  labs(
    title = "Expected Detractors By Key Customer Service Factors",
    x = "% of Detractors",
    y = "Customer Service Factors"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```



The graph shows that contacting customer service through channels other than telephone is linked to a higher proportion of detractors—especially via email, or via chat when addressing incorrect item issues.

Additionally, customer service interactions in Drinkland correspond to more detractors compared to Eatland.

Finally, customers reaching out about cancellation issues also tend to have more detractors than those contacting for typical issues.

It is important to note, however, that the overall impact of these factors depends on the volume of customers contacting us for each reason. We will account for this in the next section.

4.4. Evaluating the Impact of Contextual Customer Service Factors

To assess the impact in terms of additional detractors associated with each customer service factor, we weight the predicted proportions of detractors by the proportion of cases associated with each factor.

We assume the following distribution of cases across the factors:

- Baseline: Typical issues handled via phone in Eatland — 10%
- Attention via chat for incorrect item issues — 10%
- Attention via email — 20%
- Attention via chat — 40%
- Attention via web form — 5%
- Attention in the country Drinkland — 50%
- Attention for cancellation issues — 5%
- Attention for incorrect item issues — 30%

Using these proportions, we create impact variables by weighting both the predicted percentages of detractors and their differences across factors:

```
# Create a variable specifying the proportion of cases
ord.model.metrics.df$prop.cases <- c(.10, .10, .20, .40, .05, .50, .05, .30)

# Define variables to weight
variables_to_mutate <- c(
  "Prediction",
  "Lower_Prediction",
  "Upper_Prediction",
  "Predicted_Change",
  "Lower_Predicted_Change",
  "Upper_Predicted_Change"
)

# Calculate weighted impact variables
for(variable in variables_to_mutate) {
  new_col <- paste0("Impact_", variable)
  ord.model.metrics.df[[new_col]] <- ord.model.metrics.df[["prop.cases"]] * ord.model.metrics.df[[varial
}

# Review the updated dataframe
ord.model.metrics.df
```

```
##                  Term Prediction Lower_Prediction Upper_Prediction
## 1         (Intercept)   34.95480         33.98506         35.93715
## 2 chat_incorrect_item   65.10563         63.36170         66.81003
## 3               email   47.88224         46.47219         49.29567
## 4        cancellation   41.26410         39.53295         43.01712
## 5                chat   40.90509         39.90661         41.91113
## 6            web_form   40.12560         38.50178         41.77138
## 7    countryDrinkland   39.15569         38.15204         40.16859
## 8      incorrect_item   35.28039         33.81002         36.77916
##   Predicted_Change Lower_Predicted_Change Upper_Predicted_Change
## 1               NA                    NA                    NA
## 2       30.1460325             28.144926             32.142751
## 3       12.9220210             11.203495             14.649027
```

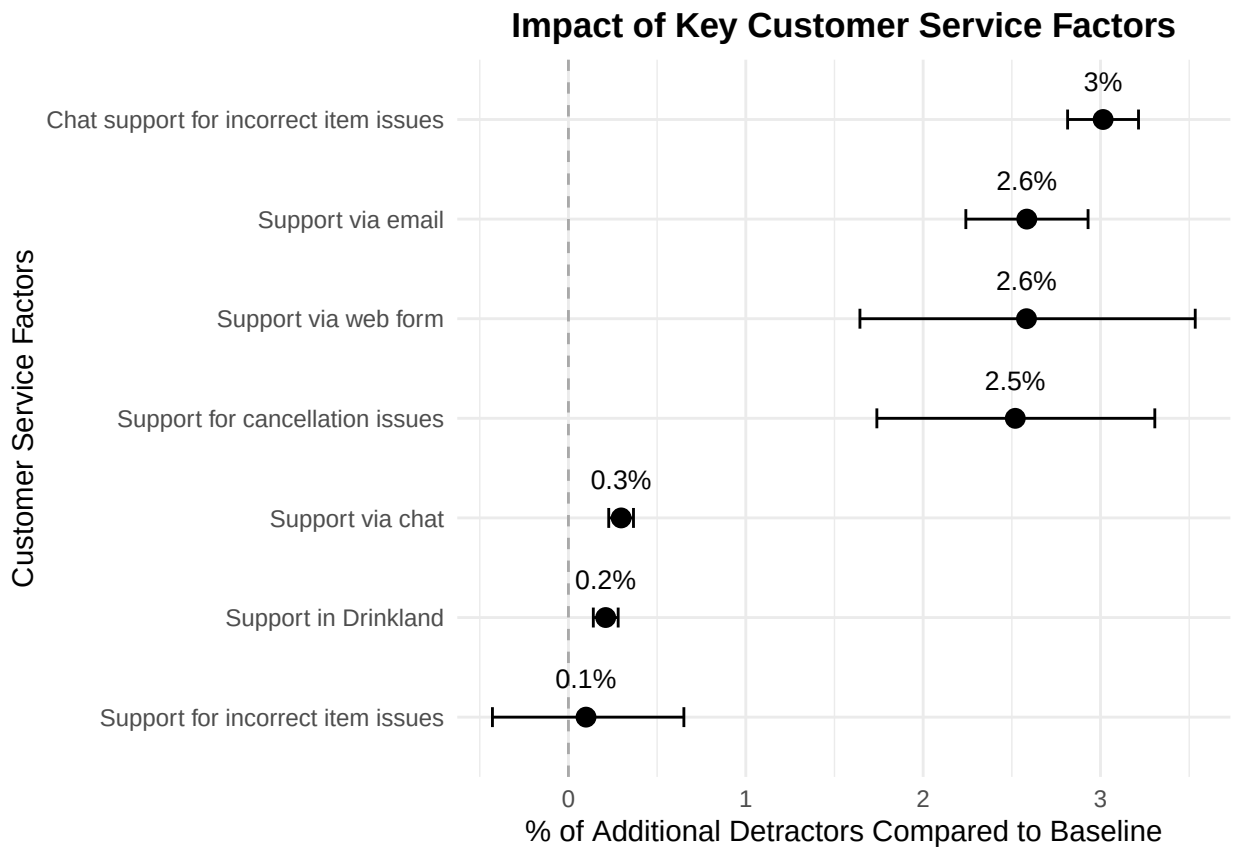
```
## 4        6.2982647              4.347519              8.265228
## 5        5.9443321              4.541586              7.331843
## 6        5.1654411              3.286841              7.068182
## 7        4.1972629              2.803931              5.607530
## 8        0.3301661             -1.428853              2.169299
##   Relative_Probability Lower_Relative_Probability Upper_Relative_Probability
## 1                   NA                         NA                         NA
## 2             1.682558                  1.6101524                   1.753902
## 3             1.369833                  1.3263056                   1.413502
## 4             1.180499                  1.1344224                   1.227097
## 5             1.170228                  1.1353773                   1.205422
## 6             1.147928                  1.0988167                   1.197658
## 7             1.120180                  1.0930251                   1.147582
## 8             1.009314                  0.9671599                   1.052196
##   Odds_Ratio Lower_Odds_Ratio Upper_Odds_Ratio
## 1         NA               NA               NA
## 2   2.657235        2.3956728         2.948434
## 3   1.709610        1.6083338         1.817337
## 4   1.307305        1.2227509         1.397669
## 5   1.288059        1.2244574         1.355003
## 6   1.247064        1.1604400         1.339992
## 7   1.197521        1.1505418         1.246436
## 8   1.014392        0.9503875         1.082562
##                                                Label prop.cases
## 1 Baseline: Typical issues handled via phone in Eatland       0.10
## 2               Chat support for incorrect item issues       0.10
## 3                                    Support via email       0.20
## 4                      Support for cancellation issues       0.40
## 5                                     Support via chat       0.05
## 6                                  Support via web form       0.50
## 7                                  Support in Drinkland       0.05
## 8                    Support for incorrect item issues       0.30
##   Impact_Prediction Impact_Lower_Prediction Impact_Upper_Prediction
## 1          3.495480                3.398506                3.593715
## 2          6.510563                6.336170                6.681003
## 3          9.576447                9.294437                9.859134
## 4         16.505638               15.813180               17.206847
## 5          2.045255                1.995331                2.095557
## 6         20.062798               19.250892               20.885688
## 7          1.957784                1.907602                2.008429
## 8         10.584116               10.143007               11.033749
##   Impact_Predicted_Change Impact_Lower_Predicted_Change
## 1                      NA                           NA
## 2              3.01460325                    2.8144926
## 3              2.58440420                    2.2406990
## 4              2.51930586                    1.7390077
## 5              0.29721660                    0.2270793
## 6              2.58272054                    1.6434204
## 7              0.20986315                    0.1401966
## 8              0.09904984                   -0.4286560
##   Impact_Upper_Predicted_Change
## 1                            NA
## 2                     3.2142751
## 3                     2.9298054
```

```
## 4                    3.3060911
## 5                    0.3665921
## 6                    3.5340910
## 7                    0.2803765
## 8                    0.6507896
```

To better understand the relative impact of each customer service factor on the increase in detractors, we visualize the additional predicted detractors associated with each factor.

The following dot-and-whisker plot displays the estimated percentage increase in detractors, adjusted by the proportion of cases for each factor. Error bars represent the 95% confidence intervals.

```
# Create the dot-and-whisker plot for the additional detractors associated to each case.
ggplot(ord.model.metrics.df[-1, ], aes(x = Impact_Predicted_Change,
                    y = reorder(Label, Impact_Predicted_Change))) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "darkgrey") +
  geom_errorbarh(aes(xmin = Impact_Lower_Predicted_Change, xmax = Impact_Upper_Predicted_Change), heigh
  geom_point(size = 3) +
  geom_text(aes(label = paste0(round(Impact_Predicted_Change, 1), "%")),
            vjust = -1.5, size = 3.5) +
  labs(
    title = "Impact of Key Customer Service Factors",
    x = "% of Additional Detractors Compared to Baseline",
    y = "Customer Service Factors"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```



This graph allows us to visualize the absolute percentage of detractors expected under each key contextual customer service factor, compared to the baseline scenario (support via phone in Eatland for typical issues).

For example, we can communicate to stakeholders that if we successfully address problems experienced by customers contacting via chat for incorrect item issues—bringing the detractor probability down to the baseline level—we could expect an approximate 3% reduction in overall detractors related to customer service.

However, we do not yet know the specific issues customers encounter within each contextual factor. This will be explored in the next study case: **"Predicting Customer Detractors (Part 2): Assessing Improvement Opportunities via Text Analysis"**