MicrobialBiotechnology_Project_Markdown

2024-03-05

By George Strevens

#Install packages

```
options(repos = c(CRAN = "https://cran.rstudio.com/"))
install.packages("dplyr")
```

```
## Installing package into 'M:/R/Win-Library/4.2'
## (as 'lib' is unspecified)
```

```
## package 'dplyr' successfully unpacked and MD5 sums checked
```

```
## Warning: cannot remove prior installation of package 'dplyr'
```

```
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## M:\R\Win-Library\4.2\00LOCK\dplyr\libs\x64\dplyr.dll to
## M:\R\Win-Library\4.2\dplyr\libs\x64\dplyr.dll: Permission denied
```

```
## Warning: restored 'dplyr'
```

```
##
## The downloaded binary packages are in
##  C:\Users\s2146807\AppData\Local\Temp\RtmpGEDx57\downloaded_packages
```

```
install.packages("lubridate")
```

```
## Installing package into 'M:/R/Win-Library/4.2'
## (as 'lib' is unspecified)
```

```
## package 'lubridate' successfully unpacked and MD5 sums checked
```

```
## Warning: cannot remove prior installation of package 'lubridate'
```

```
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## M:\R\Win-Library\4.2\00LOCK\lubridate\libs\x64\lubridate.dll to
## M:\R\Win-Library\4.2\lubridate\libs\x64\lubridate.dll: Permission denied
```

```
## Warning: restored 'lubridate'
```

```
##
## The downloaded binary packages are in
##  C:\Users\s2146807\AppData\Local\Temp\RtmpGEDx57\downloaded_packages
```

```
install.packages("ggplot2")
```

```
## Installing package into 'M:/R/Win-Library/4.2'
## (as 'lib' is unspecified)
```

```
## package 'ggplot2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\s2146807\AppData\Local\Temp\RtmpGEDx57\downloaded_packages

install.packages("tidyr")

## Installing package into 'M:/R/Win-Library/4.2'
## (as 'lib' is unspecified)

## package 'tidyr' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'tidyr'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## M:\R\Win-Library\4.2\00LOCK\tidyr\libs\x64\tidyr.dll to
## M:\R\Win-Library\4.2\tidyr\libs\x64\tidyr.dll: Permission denied

## Warning: restored 'tidyr'

##
## The downloaded binary packages are in
##  C:\Users\s2146807\AppData\Local\Temp\RtmpGEDx57\downloaded_packages

install.packages("scales")

## Installing package into 'M:/R/Win-Library/4.2'
## (as 'lib' is unspecified)

## package 'scales' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\s2146807\AppData\Local\Temp\RtmpGEDx57\downloaded_packages

install.packages("MASS")

## Installing package into 'M:/R/Win-Library/4.2'
## (as 'lib' is unspecified)

## package 'MASS' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'MASS'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## M:\R\Win-Library\4.2\00LOCK\MASS\libs\x64\MASS.dll to
## M:\R\Win-Library\4.2\MASS\libs\x64\MASS.dll: Permission denied

## Warning: restored 'MASS'

##
## The downloaded binary packages are in
##  C:\Users\s2146807\AppData\Local\Temp\RtmpGEDx57\downloaded_packages
```

#library import

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
##
## Attaching package: 'dplyr'
```

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

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

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.2.3
```

```
##
## Attaching package: 'lubridate'
```

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

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.2.3
```

```
library(scales)
```

```
## Warning: package 'scales' was built under R version 4.2.3
```

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 4.2.3
```

```
##
## Attaching package: 'MASS'
```

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

```
data <- read.csv ("datarecord.csv")
head(data)
```

```
##   Paper.No.
## 1         1
```

```
## 2         2
## 3         3
## 4         4
## 5         5
## 6         6
##
Title
## 1                              A metabolomic landscape of maize plants trea
ted with a microbial biostimulant under well-watered and drought conditions
## 2                       Streamlined identifcation of strain engineering tar
gets for bioprocess improvement using metabolic pathway enrichment analysis
## 3                                                  On-line untargeted me
tabolomics monitoring of an Escherichia coli succinate fermentation process
## 4                      A novel targeted/untargeted GC-Orbitrap metabolomics
methodology applied to Candida albicans and Staphylococcus aureus biofilms
## 5                             Glycolysis and pyrimidine biosynthesis are re
quired for replication of adherent–invasive escherichia coli in macrophages
## 6 Substantial Extracellular Metabolic Differences Found Between Phylogenet
ically Closely Related Probiotic and Pathogenic Strains of Escherichia coli
##                            DOI    ResGrp.PI Type Institution
## 1   10.3389/fpls.2021.676632 Karl Burgess IQB3        SBS
## 2 10.1038/s41598-023-39661-x Karl Burgess IQB3        SBS
## 3          10.1002/bit.28173 Karl Burgess IQB3        SBS
## 4  10.1007/s11306-016-1134-2 Karl Burgess IQB3        SBS
## 5       10.1099/mic.0.000289 Karl Burgess IQB3        SBS
## 6   10.3389/fmicb.2019.00252 Karl Burgess IQB3        SBS
##                             Journal Year AnalysisPgrm CodeArchived DAS
## 1       Frontiers in Plant Science 2021            1           NA   1
## 2               Scientific reports 2023            1           NA   1
## 3 Biotechnology and Bioengineering 2022            1           NA   1
## 4                     Metabolomics 2016            1           NA  NA
## 5 Biotechnology and Bioengineering 2016            1           NA  NA
## 6        Frontiers in Microbiology 2019            1           NA   1
##   CorresAuthor Preprint Complete Reuse Access Licence Image Genomics
## 1            1        0        2     3      4       4    NA       NA
## 2            1        0        4     4      4       4    NA       NA
## 3            1        0        3     4      4       4    NA       NA
## 4            1        0        4     4      2       4    NA       NA
## 5            1        0        2     2      2       4     0       NA
## 6            1        0        4     4      4       4    NA       NA
##   Proteometabolomic                              Data.Storage Funding.Source
## 1                 0                   Supplementary material              1
## 2                 1    Supplementary material, MetaboLights              0
## 3                 0    Supplementary material, MetaboLights              0
## 4                 1                   Supplementary material              0
## 5                NA                   Supplementary material              1
## 6                 1 Supplementary material, GNPS repository              0
##   Number.of.times.cited.according.to.Google.Scholar CitationsPA
## 1                                                40    13.33333
## 2                                                 2     2.00000
```

```
## 3                                                    6     3.00000
## 4                                                   56     7.00000
## 5                                                    5     0.62500
## 6                                                   29     5.80000
##   Competing.interests EdinburghGroupLeader
## 1                   1                   0
## 2                   0                   1
## 3                  NA                   1
## 4                   1                   0
## 5                  NA                   0
## 6                   0                   0
```

#1 - Descriptive Statistics

#How many papers in each type

```
type_frequency <- table(data$Type)
print(type_frequency)
```

```
##
##  IQB3 Other
##    80    50
```

#frequency and percentage of paper in each year

```
total_counts_by_type <- data %>%
  group_by(Type) %>%
  summarise(Total = n(), .groups = "drop")

# Calculate frequency and percentage for each Type and Year
summarized_data <- data %>%
  group_by(Year, Type) %>%
  summarise(count = n(), .groups = "drop") %>%
  left_join(total_counts_by_type, by = "Type") %>%
  mutate(Percentage = (count / Total) * 100)  # Removed the trailing %>%

print(summarized_data)
```

```
## # A tibble: 20 × 5
##     Year Type  count Total Percentage
##    <int> <chr> <int> <int>      <dbl>
##  1  2014 IQB3      8    80      10
##  2  2014 Other     7    50      14
##  3  2015 IQB3      7    80       8.75
##  4  2015 Other     5    50      10
##  5  2016 IQB3      5    80       6.25
##  6  2016 Other     8    50      16
##  7  2017 IQB3      9    80      11.2
##  8  2017 Other     3    50       6
##  9  2018 IQB3      7    80       8.75
## 10  2018 Other     4    50       8
```

```
## 11  2019 IQB3      9    80      11.2
## 12  2019 Other     4    50       8
## 13  2020 IQB3      7    80       8.75
## 14  2020 Other     7    50      14
## 15  2021 IQB3      8    80      10
## 16  2021 Other     5    50      10
## 17  2022 IQB3     11    80      13.8
## 18  2022 Other     3    50       6
## 19  2023 IQB3      9    80      11.2
## 20  2023 Other     4    50       8
```

Calculate the total number of papers for each year

```
yearly_totals <- data %>%
  group_by(Year) %>%
  summarise(Total = n(), .groups = "drop")

# Calculate the grand total of all papers
grand_total <- sum(yearly_totals$Total)

# Add a column for the percentage of each year's total relative to the grand
total
yearly_totals <- yearly_totals %>%
  mutate(Percentage = (Total / grand_total) * 100)

# Print the yearly totals and percentages
print(yearly_totals)
```

```
## # A tibble: 10 × 3
##     Year Total Percentage
##    <int> <int>      <dbl>
##  1  2014    15      11.5
##  2  2015    12       9.23
##  3  2016    13      10
##  4  2017    12       9.23
##  5  2018    11       8.46
##  6  2019    13      10
##  7  2020    14      10.8
##  8  2021    13      10
##  9  2022    14      10.8
## 10  2023    13      10
```

#to categorize papers based on the year of publication compared to FAIR principles implementation 2016 (overall)

```
data <- data %>%
  mutate(Period = ifelse(Year <= 2016, "Before or in 2016", "After 2016"))

# Calculate the total number of papers for each period
period_totals <- data %>%
```

```
  group_by(Period) %>%
  summarise(Total = n(), .groups = "drop")

# Calculate the grand total of all papers
grand_total <- sum(period_totals$Total)

# Add a column for the percentage of each period's total relative to the gran
d total
period_totals <- period_totals %>%
  mutate(Percentage = (Total / grand_total) * 100)

# Print the period totals and percentages
print(period_totals)
```

```
## # A tibble: 2 × 3
##   Period            Total Percentage
##   <chr>             <int>      <dbl>
## 1 After 2016           90       69.2
## 2 Before or in 2016    40       30.8
```

#to categorize papers based on the year of publication compared to FAIR principles implementation 2016 (depend on type)

```
#Frequency and Percantage of paper before and After FAIR principles (2016)
total_counts_by_type <- data %>%
  group_by(Type) %>%
  summarise(Total = n(), .groups = "drop")

# Categorize, calculate frequency and percentage for each Type based on year
summarized_data <- data %>%
  mutate(Category = ifelse(Year <= 2016, "On/Before 2016", "After 2020")) %>%
  group_by(Category, Type) %>%
  summarise(count = n(), .groups = "drop") %>%
  left_join(total_counts_by_type, by = "Type") %>%
  mutate(Percentage = (count / Total) * 100)

print(summarized_data)
```

```
## # A tibble: 4 × 5
##   Category       Type  count Total Percentage
##   <chr>          <chr> <int> <int>      <dbl>
## 1 After 2020     IQB3     60    80         75
## 2 After 2020     Other    30    50         60
## 3 On/Before 2016 IQB3     20    80         25
## 4 On/Before 2016 Other    20    50         40
```

#to categorise papers based on the year of publication compared to COVID 19 - 2020 (overall)

```
data <- data %>%
  mutate(Period = ifelse(Year <= 2020, "Before or in 2020", "After 2020"))

# Calculate the total number of papers for each period
period_totals <- data %>%
  group_by(Period) %>%
  summarise(Total = n(), .groups = "drop")

# Calculate the grand total of all papers
grand_total <- sum(period_totals$Total)

# Add a column for the percentage of each period's total relative to the gran
d total
period_totals <- period_totals %>%
  mutate(Percentage = (Total / grand_total) * 100)

# Print the period totals and percentages
print(period_totals)
```

```
## # A tibble: 2 × 3
##   Period            Total Percentage
##   <chr>             <int>      <dbl>
## 1 After 2020           40       30.8
## 2 Before or in 2020    90       69.2
```

#to categorise papers based on the year of publication compared to COVID 19 - 2020 (IQB3 vs Other)

```
total_counts_by_type <- data %>%
  group_by(Type) %>%
  summarise(Total = n(), .groups = "drop")

# Categorize, calculate frequency and percentage for each Type based on year
summarized_data <- data %>%
  mutate(Category = ifelse(Year <= 2020, "On/Before 2020", "After 2020")) %>%
  group_by(Category, Type) %>%
  summarise(count = n(), .groups = "drop") %>%
  left_join(total_counts_by_type, by = "Type") %>%
  mutate(Percentage = (count / Total) * 100)

print(summarized_data)
```

```
## # A tibble: 4 × 5
##   Category       Type  count Total Percentage
##   <chr>          <chr> <int> <int>      <dbl>
## 1 After 2020     IQB3     28    80         35
## 2 After 2020     Other    12    50         24
## 3 On/Before 2020 IQB3     52    80         65
## 4 On/Before 2020 Other    38    50         76
```

Frequency and Percentage of the papers that share data (Data Sharing= Data Completeness score >1) Overall

```
data <- data %>%
  mutate(Period = ifelse(Complete > 1, "Completeness > 1", "Completeness = 1"
))

# Calculate the total number of papers for each period
period_totals <- data %>%
  group_by(Period) %>%
  summarise(Total = n(), .groups = "drop")

# Calculate the grand total of all papers
grand_total <- sum(period_totals$Total)

# Add a column for the percentage of each period's total relative to the gran
d total
period_totals <- period_totals %>%
  mutate(Percentage = (Total / grand_total) * 100)

# Print the period totals and percentages
print(period_totals)
```

```
## # A tibble: 2 × 3
##   Period           Total Percentage
##   <chr>            <int>      <dbl>
## 1 Completeness = 1     6       4.62
## 2 Completeness > 1   124      95.4
```

Frequency and Percentage of the papers that share data (Data Sharing= Data Completeness score >1) by Type

```
total_counts_by_type <- data %>%
  group_by(Type) %>%
  summarise(Total = n(), .groups = "drop")

# Categorize based on Completeness, calculate frequency and percentage for ea
ch Type
summarized_data <- data %>%
  filter(Complete == 1 | Complete > 1) %>%
  mutate(Category = ifelse(Complete == 1, "Completeness = 1", "Completeness >
1")) %>%
  group_by(Category, Type) %>%
  summarise(count = n(), .groups = "drop") %>%
  left_join(total_counts_by_type, by = "Type") %>%
  mutate(Percentage = (count / Total) * 100)

print(summarized_data)
```

```
## # A tibble: 4 × 5
##   Category         Type  count Total Percentage
##   <chr>            <chr> <int> <int>      <dbl>
## 1 Completeness = 1 IQB3      3    80       3.75
## 2 Completeness = 1 Other     3    50       6
## 3 Completeness > 1 IQB3     77    80      96.2
## 4 Completeness > 1 Other    47    50      94
```

#Chi square test to assess if there is any difference in data share between the two types (IQB3 vs Other)

```
# Create a contingency table
table_completeness <- table(data$Type, data$Complete > 1)

# Perform the Chi-square test
result <- chisq.test(table_completeness)

## Warning in chisq.test(table_completeness): Chi-squared approximation may b
e
## incorrect

#Print the chi-square results
print(result)

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table_completeness
## X-squared = 0.027302, df = 1, p-value = 0.8688

# Get the expected values
expected_values <- result$expected

# Print the expected values
print(expected_values) ##in order to check the assumption of chi square test:
Each cell in the contingency table should have an expected count of 5 or more
.

##
##            FALSE     TRUE
##   IQB3  3.692308 76.30769
##   Other 2.307692 47.69231

#The assumptions of that test were met:

#### Sample Size: Each cell in the contingency table has an expected count of
5 or more.
#### Independence: The observations are  independent of each other. This mean
s that the selection of one observation should not influence or affect the se
lection of another observation.
### Random Sampling: The data is a random sample.
```

```
### Categorical Data: Both variables should be categorical (either nominal or ordinal).
```

#Frequency and percentage of Preprint

```
# Calculate frequency and percentage for each Preprint category, across all types
Preprint_summary <- data %>%
  group_by(Preprint) %>%
  summarise(count = n(), .groups = "drop")

# Calculate the overall total
overall_total <- sum(Preprint_summary$count)

# Add a column for the percentage of each Preprint category relative to the overall total
Preprint_summary <- Preprint_summary %>%
  mutate(percentage = (count / overall_total) * 100)

# Print the summary
print(Preprint_summary)
```

```
## # A tibble: 2 × 3
##   Preprint count percentage
##      <int> <int>      <dbl>
## 1        0   110       84.6
## 2        1    20       15.4
```

```
# Frequency and Percentage of the papers with Preprint
data %>%
  group_by(Type, Preprint) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(Type) %>%
  mutate(total = sum(count),
         percentage = (count/total)*100) %>%
  ungroup() %>%
  arrange(Type, Preprint)
```

```
## # A tibble: 4 × 5
##   Type  Preprint count total percentage
##   <chr>    <int> <int> <int>      <dbl>
## 1 IQB3         0    69    80       86.2
## 2 IQB3         1    11    80       13.8
## 3 Other        0    41    50       82
## 4 Other        1     9    50       18
```

#Chi square test to assess if there is any difference in preprint between the two types (IQB3, Other)

```
# Create a contingency table
table_preprint <- table(data$Type, data$Preprint)
```

```r
# Perform the Chi-square test
result <- chisq.test(table_preprint)

#Print the chi-square results
print(result)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table_preprint
## X-squared = 0.16287, df = 1, p-value = 0.6865
```

```r
# Get the expected values
expected_values <- result$expected

# Print the expected values
print(expected_values) ##in order to check the assumption of chi square test: Each cell in the contingency table should have an expected count of 5 or more.
```

```
##
##                 0         1
##   IQB3  67.69231 12.307692
##   Other 42.30769  7.692308
```

```r
#The assumptions of that test were met:

#### Sample Size: Each cell in the contingency table has an expected count of 5 or more.
#### Independence: The observations are  independent of each other. This means that the selection of one observation should not influence or affect the selection of another observation.
### Random Sampling: The data is a random sample.
### Categorical Data: Both variables should be categorical (either nominal or ordinal).
```

Calculate frequency and percentage for DAS

```r
DAS_summary <- data %>%
  group_by(DAS) %>%
  summarise(count = n(), .groups = "drop")

# Calculate the overall total
overall_total <- sum(DAS_summary$count)

# Add a column for the percentage of each DAS category relative to the overall total
DAS_summary <- DAS_summary %>%
  mutate(percentage = (count / overall_total) * 100)
```

```
print(DAS_summary)
```

```
## # A tibble: 3 × 3
##     DAS count percentage
##   <int> <int>      <dbl>
## 1     0     8       6.15
## 2     1    48      36.9
## 3    NA    74      56.9
```

```
data$DAS <- as.factor(data$DAS)

# Frequency and Percentage of the papers with DAS in the publications
data %>%
  group_by(Type, DAS) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(Type) %>%
  mutate(total = sum(count),
         percentage = (count/total)*100) %>%
  ungroup() %>%
  arrange(Type, DAS)
```

```
## # A tibble: 6 × 5
##   Type  DAS   count total percentage
##   <chr> <fct> <int> <int>      <dbl>
## 1 IQB3  0         6    80        7.5
## 2 IQB3  1        28    80       35
## 3 IQB3  <NA>     46    80       57.5
## 4 Other 0         2    50        4
## 5 Other 1        20    50       40
## 6 Other <NA>     28    50       56
```

#Chi square test to assess if there is any difference in DAS between the two types (IQB3, Other)

```
# Create a contingency table
table_das <- table(data$Type, data$DAS)

# Perform the Chi-square test
result <- chisq.test(table_das)
```

```
## Warning in chisq.test(table_das): Chi-squared approximation may be incorre
ct
```

```
#Print the chi-square results
print(result)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
```

```
## data:  table_das
## X-squared = 0.25267, df = 1, p-value = 0.6152

# Get the expected values
expected_values <- result$expected

# Print the expected values
print(expected_values) ##in order to check the assumption of chi square test: Each cell in the contingency table should have an expected count of 5 or more .

##
##                0        1
##   IQB3  4.857143 29.14286
##   Other 3.142857 18.85714

#The assumptions of that test were met:

#### Sample Size: Each cell in the contingency table has an expected count of 5 or more.
#### Independence: The observations are  independent of each other. This means that the selection of one observation should not influence or affect the selection of another observation.
### Random Sampling: The data is a random sample.
### Categorical Data: Both variables should be categorical (either nominal or ordinal).
```

Calculate frequency and percentage for CorresAuthor

```
CorresAuthor_summary <- data %>%
  group_by(CorresAuthor) %>%
  summarise(count = n(), .groups = "drop")

# Calculate the overall total
overall_total <- sum(CorresAuthor_summary$count)

# Add a column for the percentage of each category relative to the overall total
CorresAuthor_summary <- CorresAuthor_summary %>%
  mutate(percentage = (count / overall_total) * 100)

print(CorresAuthor_summary)

## # A tibble: 2 × 3
##   CorresAuthor count percentage
##          <int> <int>      <dbl>
## 1            0    21       16.2
## 2            1   109       83.8
```

```r
# Frequency and Percentage of the papers with CorresAuthor being 0 or 1
data %>%
  group_by(Type, CorresAuthor) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(Type) %>%
  mutate(total = sum(count),
         percentage = (count/total)*100) %>%
  ungroup() %>%
  arrange(Type, CorresAuthor)
```

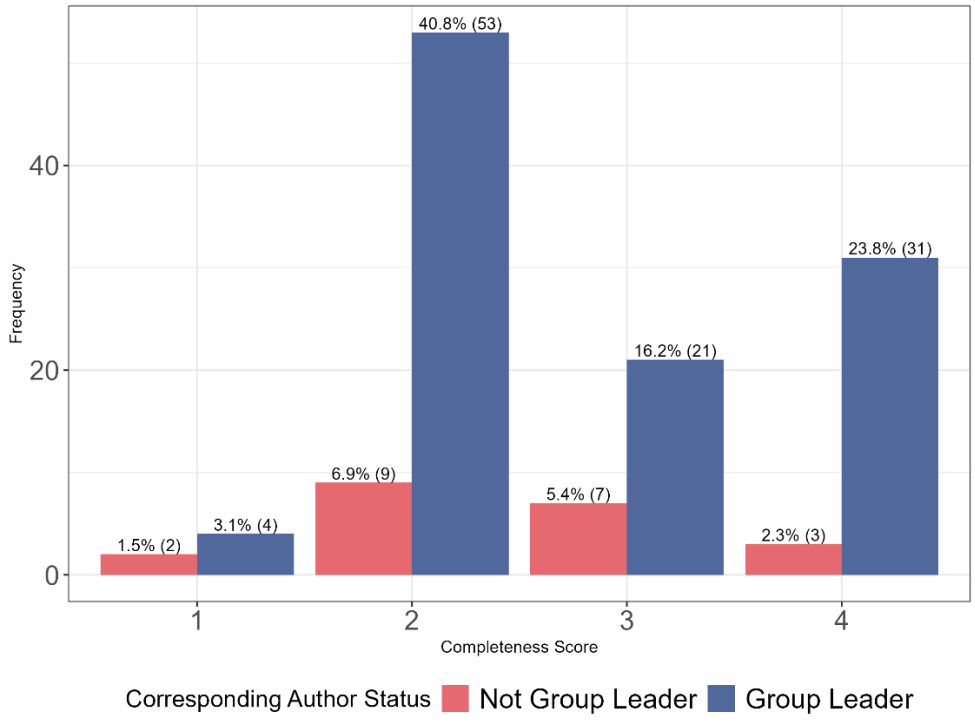
```
## # A tibble: 4 × 5
##   Type  CorresAuthor count total percentage
##   <chr>        <int> <int> <int>      <dbl>
## 1 IQB3             0    14    80       17.5
## 2 IQB3             1    66    80       82.5
## 3 Other            0     7    50       14
## 4 Other            1    43    50       86
```

```r
# Assuming 'data' is your DataFrame and has been loaded correctly
# Transform the CorresAuthor variable
data <- data %>%
  mutate(CorresAuthorStatus = case_when(
    CorresAuthor == 1 ~ "Group Leader",
    CorresAuthor == 0 ~ "Not Group Leader",
    TRUE ~ as.character(CorresAuthor)  # Fallback for unexpected values
  ))

# Check the class of the new variable
class(data$CorresAuthorStatus)
```

```
## [1] "character"
```

```r
# Convert the new Funding Source Status variable to a factor for plotting
data <- data %>%
  mutate(CorresAuthorStatus = factor(CorresAuthorStatus, levels = c("Not Group Leader", "Group Leader")))

# Calculate frequency and percentage for each 'Complete' score and 'Funding Source Status'
CorresAuthor_frequency <- data %>%
  group_by(Complete, CorresAuthorStatus) %>%
  summarise(Frequency = n(), .groups = 'drop') %>%
  mutate(Percentage = (Frequency / sum(Frequency)) * 100)

# Define specific colors for the Funding Source status
colors <- c("Not Group Leader" = "#E5696F", "Group Leader" = "#50689B")  # Correct the color names to match factor levels

# Create the plot
CorresAuthor_plot <- ggplot(CorresAuthor_frequency, aes(x
```

```
= as.factor(Complete), y = Frequency, fill = CorresAuthorStatus)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = paste0(round(Percentage, 1), "% (", Frequency, ")")),
            position = position_dodge(width = 0.9), vjust = -0.25, size
= 3.5) +
  scale_fill_manual(values = colors, name = "Corresponding Author
Status") +  # Ensure the name matches the legend
  theme_bw() +
  theme(
    axis.title = element_text(size = 10),
    axis.text = element_text(size = 16),
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 16),
    legend.position = "bottom"
  ) +
  labs(x = "Completeness Score", y = "Frequency")

# Print the plot
print(CorresAuthor_plot)
```




```
# Save the plot to a file
ggsave("CorresAuthor_status_distribution_plot.png", CorresAuthor_plot, width
= 8, height = 6, bg = "white")
```

```r
# Assuming 'data' is your DataFrame and has been loaded correctly
library(dplyr)

# Transform the Completeness score into a binary variable (high vs. low)
data <- data %>%
  mutate(CompletenessCategory = case_when(
    Complete %in% 3:4 ~ "High",
    Complete %in% 1:2 ~ "Low",
    TRUE ~ NA_character_  # Handle potential unexpected values
  ))

# Assuming CorresAuthor is already in a format that can be directly
categorized
# If not, you should transform CorresAuthor into categorical variable similar
to CompletenessCategory
# Example transformation is shown in the comment below. Adjust it based on
your data structure.
# data <- data %>%
#   mutate(CorresAuthorCategory = case_when(
#     CorresAuthor == some_condition ~ "Category1",
#     CorresAuthor == another_condition ~ "Category2",
#     TRUE ~ NA_character_  # Handle potential unexpected values
#   ))

# Drop rows with NA in either category if any exist due to unexpected values
data <- data %>%
  filter(!is.na(CompletenessCategory) & !is.na(CorresAuthor))

# Create a contingency table
contingency_table <- table(data$CompletenessCategory, data$CorresAuthor)

# Conduct a Chi-square test of independence
chi_square_result <- chisq.test(contingency_table)

# Print the result of the Chi-square test
print(chi_square_result)
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 4.6417e-31, df = 1, p-value = 1
```

Calculate frequency and percentage for Code Sharing

```r
CodeArchived_summary <- data %>%
  group_by(CodeArchived) %>%
  summarise(count = n(), .groups = "drop")
```

```r
# Calculate the overall total
overall_total <- sum(CodeArchived_summary$count)

# Add a column for the percentage of each Code Archived category relative to
the overall total
CodeArchived_summary <- CodeArchived_summary %>%
  mutate(percentage = (count / overall_total) * 100)

print(CodeArchived_summary)
```

```
## # A tibble: 3 × 3
##   CodeArchived count percentage
##          <int> <int>      <dbl>
## 1            0     3       2.31
## 2            1    17      13.1
## 3           NA   110      84.6
```

```r
# Frequency and Percentage of the papers that shared the code used
data %>%
  group_by(Type, CodeArchived) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(Type) %>%
  mutate(total = sum(count),
         percentage = (count/total)*100) %>%
  ungroup() %>%

  arrange(Type, CodeArchived)
```

```
## # A tibble: 6 × 5
##   Type  CodeArchived count total percentage
##   <chr>        <int> <int> <int>      <dbl>
## 1 IQB3             0     2    80        2.5
## 2 IQB3             1    12    80       15
## 3 IQB3            NA    66    80       82.5
## 4 Other            0     1    50        2
## 5 Other            1     5    50       10
## 6 Other           NA    44    50       88
```

#Chi square test to assess if there is any difference in Code Sharing between the two types (IQB3, Other)

```r
# Create a contingency table
table_code <- table(data$Type, data$CodeArchived)

# Perform the Chi-square test
result <- chisq.test(table_code)
```

```
## Warning in chisq.test(table_code): Chi-squared approximation may be incorr
ect
```

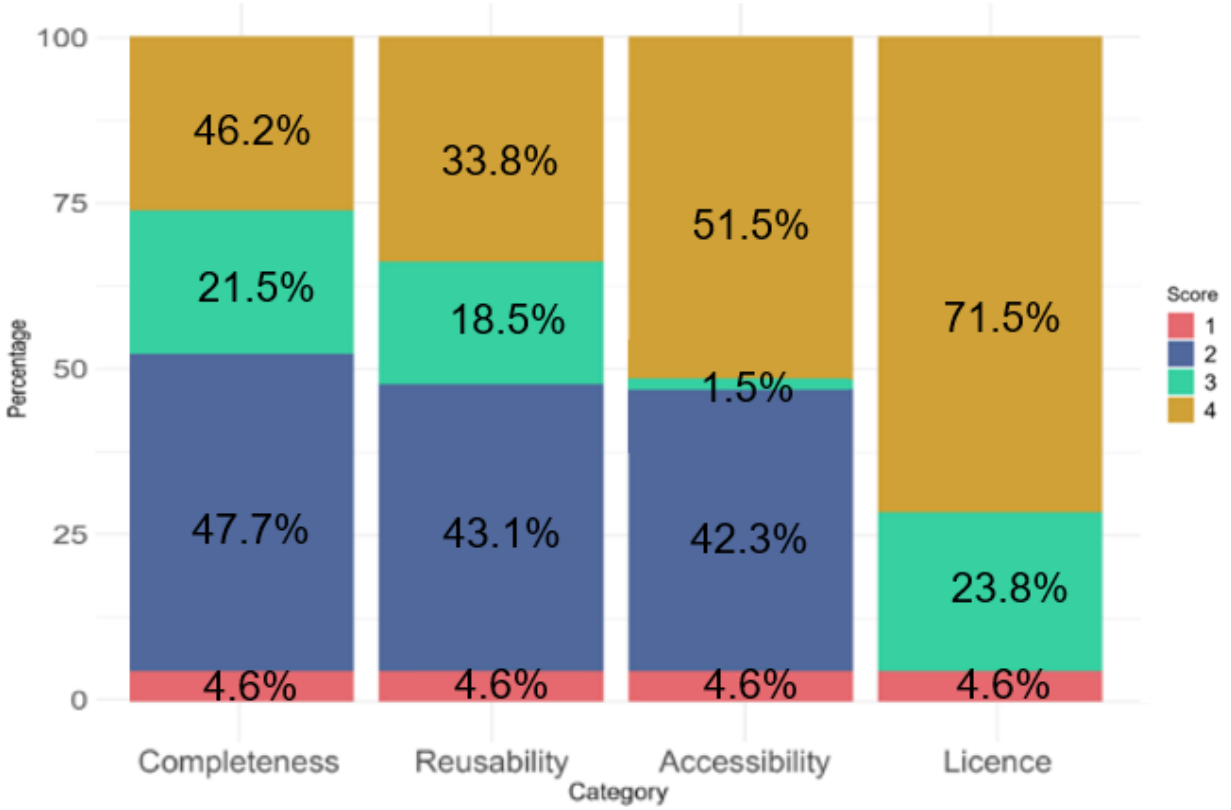
```
#Print the chi-square results
print(result)

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table_code
## X-squared = 2.1717e-31, df = 1, p-value = 1

# Get the expected values
expected_values <- result$expected

# Print the expected values
print(expected_values) ##in order to check the assumption of chi square test: Each cell in the contingency table should have an expected count of 5 or more .

##
##           0    1
##   IQB3  2.1 11.9
##   Other 0.9  5.1

#The assumptions of that test were met:

#### Sample Size: Each cell in the contingency table has an expected count of 5 or more.
#### Independence: The observations are  independent of each other. This means that the selection of one observation should not influence or affect the selection of another observation.
### Random Sampling: The data is a random sample.
### Categorical Data: Both variables should be categorical (either nominal or ordinal).
```

#Plot the 4 scoring criteria for all the papers

```
# Convert all relevant columns to factors first to avoid the error
data <- data %>%
  mutate(across(c(Complete, Reuse, Access, Licence), as.character), .groups = "drop")

# Reshape data to long format
long_data <- data %>%
  pivot_longer(cols = c(Complete, Reuse, Access, Licence), names_to = "Category", values_to = "Value") %>%
  mutate(Value = as.character(Value))

# Rename and reorder categories
long_data$Category <- recode(long_data$Category,
                             "Complete" = "Completeness",
                             "Reuse" = "Reusability",
                             "Access" = "Accessibility",
```

```
                                "Licence" = "Licence")
long_data$Category <- factor(long_data$Category, levels = c("Completeness", "
Reusability", "Accessibility", "Licence"))

# Calculate counts and percentages ensuring percentages do not exceed 100%
long_data <- long_data %>%
  group_by(Category, Value) %>%
  summarise(Count = n(), .groups = "drop") %>%
  ungroup() %>%
  group_by(Category) %>%
  mutate(Total = sum(Count),
         Percentage = pmin(Count / Total * 100, 100)) %>%
  ungroup() %>%
  arrange(Category, desc(Value)) %>%
  group_by(Category) %>%
  mutate(Label = ifelse(Percentage > 5, paste0(round(Percentage, 1), "%"), ""
),
         CumPercentage = pmin(cumsum(Percentage), 100)) %>%
  ungroup()

# Plot
plot_all <- ggplot(long_data, aes(x = Category, y = Percentage, fill = Value)
) +
  geom_bar(stat = "identity", position = "stack") +
  scale_fill_manual(values = c("#E5696F", "#50689B", "#36D0A1", "#D0A136")) +
  labs(y = "Percentage", fill = "Score", title = "Scoring Criteria for Papers
") +
  theme_minimal() +
  geom_text(aes(label = Label, y = pmin(CumPercentage - 0.5 * Percentage, 100
)),
            position = position_stack(vjust = 0.5), size = 6, check_overlap =
TRUE) +
  theme(
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 18),
    legend.title = element_text(size = 12),
    legend.text = element_text(size = 14)
  ) +
  ylim(0, 100)

# Ensure the values are factored in reverse order so that 4 is at the top of
the stack
long_data$Value <- factor(long_data$Value, levels = c("4", "3", "2", "1"))

# Show the plot
print(plot_all)
```

```
## Warning: Removed 5 rows containing missing values or values outside the sc
ale range
## (`geom_text()`).
```



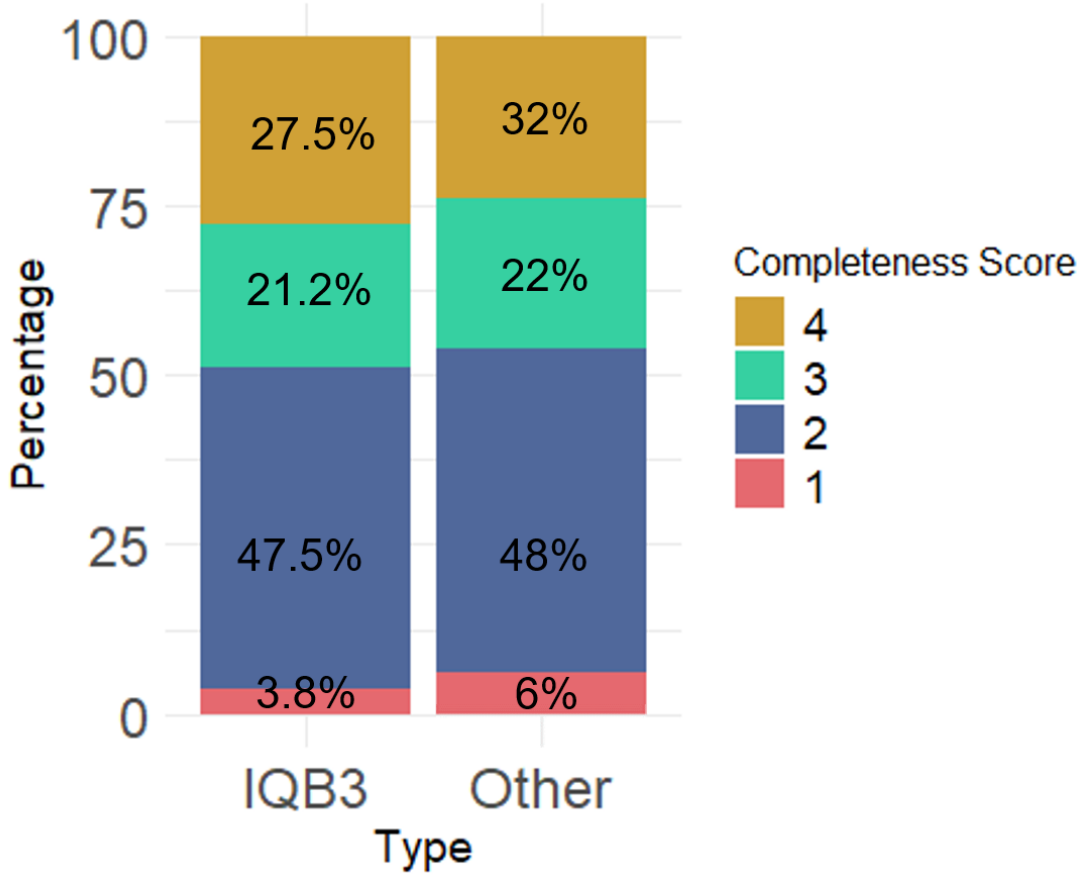

```
# Save the plot
ggsave("plotall.png", plot_all, width = 10, height = 8, dpi = 300, bg="white"
)

## Warning: Removed 5 rows containing missing values or values outside the sc
ale range
## (`geom_text()`).
```

#Cluster bar chart fro the completeness criteria by the Type of the study

```
library(ggplot2)
library(dplyr)

# Reorder the levels of variables
data$Complete <- factor(data$Complete, levels = c("4", "3", "2", "1"))

# Summarize data
summarized_data <- data %>%
  group_by(Type, Complete) %>%
  summarise(Count = n(), .groups = "drop") %>%
  left_join(data %>% group_by(Type) %>% summarise(Total = n(), .groups = "dro
```

```
p"), by = "Type") %>%
  mutate(Percentage = Count/Total * 100)
# For cumulative percentages
summarized_data <- summarized_data %>%
  arrange(Type, desc(Complete)) %>%
  group_by(Type) %>%
  mutate(CumPercentage = cumsum(Percentage))

# Create the stacked bar chart for Completeness with colors and annotations
gcomp <- ggplot(summarized_data, aes(x = Type, y = Percentage, fill = as.fact
or(Complete))) +
  geom_bar(stat = "identity", position = "stack") +
  scale_fill_manual(values = c("#D0A136", "#36D0A1", "#50689B", "#E5696F")) +
  labs(x = "Type", y = "Percentage", fill = "Completeness Score") +
  theme_minimal() +
  geom_text(aes(label = ifelse(Percentage > 5, paste0(round(Percentage, 1), "
%"), ""),
                y = pmin(pmax(ifelse(Complete == 1, Percentage / 2, CumPerce
ntage - (0.5 * Percentage)), 0), 100)),
            position = position_stack(vjust = 0.5), color = "Black", size = 5
) +
  theme(
    axis.title = element_text(size = 14), # Increase axis titles
    axis.text = element_text(size = 18),  # Increase axis text
    legend.title = element_text(size = 12), # Increase legend title
    legend.text = element_text(size = 14)  # Increase legend text
  ) +
  ylim(0, 100)

print(gcomp)
```

```
## Warning: Removed 2 rows containing missing values or values outside the sc
ale range
## (`geom_text()`).
```



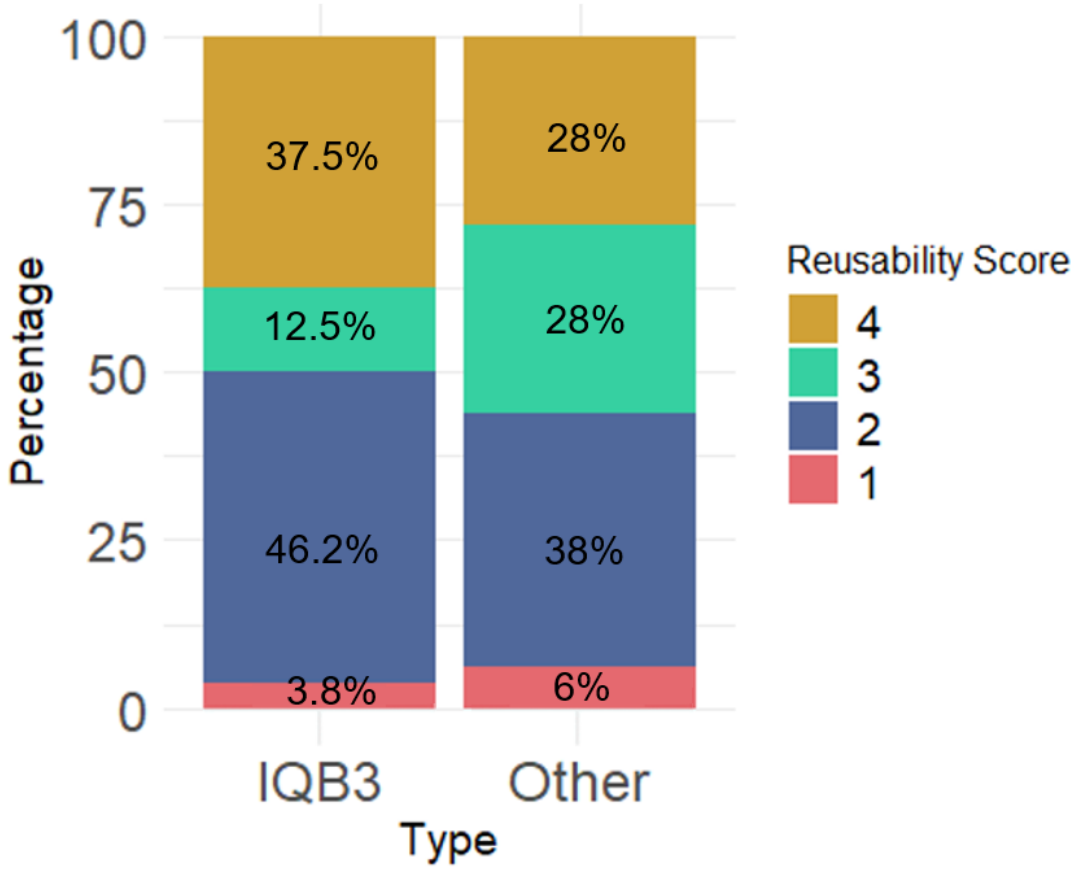

```
ggsave("gcomp.png", gcomp, width = 10, height = 8, dpi = 300)

## Warning: Removed 2 rows containing missing values or values outside the sc
ale range
## (`geom_text()`).
```

` ##Cluster bar chart for the Reuse criteria by the Type of the study

```
# Reorder the levels of variables
data$Reuse <- factor(data$Reuse, levels = c("4", "3", "2", "1"))

summarized_data_Reuse <- data %>%
  group_by(Type, Reuse) %>%
  summarise(Count = n(), .groups = "drop") %>%
  left_join(data %>% group_by(Type) %>% summarise(Total = n(), .groups = "dro
p"), by = "Type") %>%
  mutate(Percentage = Count/Total * 100)

summarized_data_Reuse <- summarized_data_Reuse %>%
  arrange(Type, desc(Reuse)) %>%
  group_by(Type) %>%
```

```
  mutate(CumPercentage = cumsum(Percentage))
print(summarized_data_Reuse)

## # A tibble: 8 × 6
## # Groups:   Type [2]
##   Type  Reuse Count Total Percentage CumPercentage
##   <chr> <fct> <int> <int>      <dbl>         <dbl>
## 1 IQB3  1         3    80       3.75          3.75
## 2 IQB3  2        37    80      46.2           50
## 3 IQB3  3        10    80      12.5           62.5
## 4 IQB3  4        30    80      37.5          100
## 5 Other 1         3    50       6             6
## 6 Other 2        19    50      38            44
## 7 Other 3        14    50      28            72
## 8 Other 4        14    50      28           100

colors <- c("#D0A136", "#36D0A1", "#50689B", "#E5696F")

# Visualization
gReuse <- ggplot(summarized_data_Reuse, aes(x = Type, y = Percentage, fill =
as.factor(Reuse))) +
  geom_bar(stat = "identity", position = "stack") +
  scale_fill_manual(values = colors) +
  labs(x = "Type", y = "Percentage", fill = "Reusability Score") +
  theme_minimal() +
  geom_text(aes(label = paste0(round(Percentage, 1), "%"),
                y = CumPercentage - (0.5 * Percentage)),
            position = position_stack(vjust = 0.5), color = "Black", size = 5
) +
  theme(
    axis.title = element_text(size = 14), # Increase axis titles
    axis.text = element_text(size = 18),  # Increase axis text
    legend.title = element_text(size = 12), # Increase legend title
    legend.text = element_text(size = 14)  # Increase legend text
  ) +
  ylim(0, 100)

print(gReuse)
```

```
## Warning: Removed 2 rows containing missing values or values outside the sc
ale range
## (`geom_text()`).
```




```
ggsave("gReuse.png", gReuse, width = 10, height = 8, dpi = 300)
```

```
## Warning: Removed 2 rows containing missing values or values outside the sc
ale range
## (`geom_text()`).
```

#bar chart for the Accessibility criteria by the Type of the study

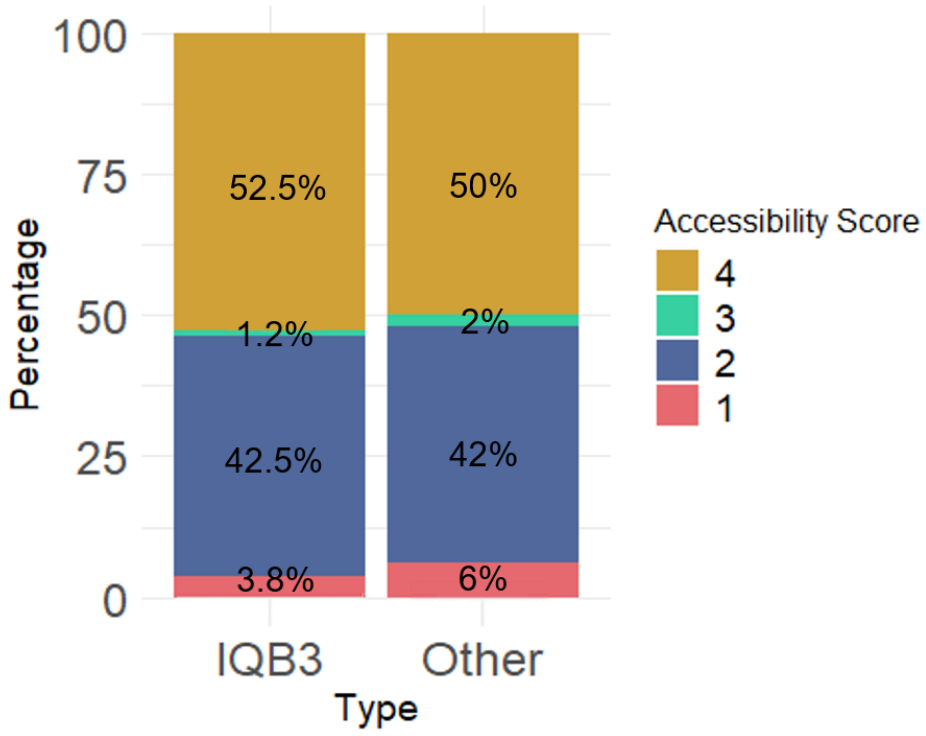
```
# Reorder the levels of variables
data$Access <- factor(data$Access, levels = c("4", "3", "2", "1"))

summarized_data_Access <- data %>%
  group_by(Type, Access) %>%
  summarise(Count = n(), .groups = "drop") %>%
  left_join(data %>% group_by(Type) %>% summarise(Total = n(), .groups = "dro
p"), by = "Type") %>%
  mutate(Percentage = Count/Total * 100)

summarized_data_Access <- summarized_data_Access %>%
  arrange(Type, desc(Access)) %>%
```

```
  group_by(Type) %>%
  mutate(CumPercentage = cumsum(Percentage))
print(summarized_data_Access)
```

```
## # A tibble: 8 × 6
## # Groups:   Type [2]
##   Type  Access Count Total Percentage CumPercentage
##   <chr> <fct>  <int> <int>      <dbl>         <dbl>
## 1 IQB3  1          3    80       3.75          3.75
## 2 IQB3  2         34    80      42.5           46.2
## 3 IQB3  3          1    80       1.25          47.5
## 4 IQB3  4         42    80      52.5          100
## 5 Other 1          3    50       6             6
## 6 Other 2         21    50      42            48
## 7 Other 3          1    50       2            50
## 8 Other 4         25    50      50           100
```

```
colors <- c("#D0A136", "#36D0A1", "#50689B", "#E5696F")

# Visualization
gaccess <- ggplot(summarized_data_Access, aes(x = Type, y = Percentage, fill
= as.factor(Access))) +
  geom_bar(stat = "identity", position = "stack") +
  scale_fill_manual(values = colors) +
  labs(x = "Type", y = "Percentage", fill = "Accessibility Score") +
  theme_minimal() +
  geom_text(aes(label = paste0(round(Percentage, 1), "%"),
                y = CumPercentage - (0.5 * Percentage)),
            position = position_stack(vjust = 0.5), color = "Black", size = 5
) +
  theme(
    axis.title = element_text(size = 14), # Increase axis titles
    axis.text = element_text(size = 18),  # Increase axis text
    legend.title = element_text(size = 12), # Increase legend title
    legend.text = element_text(size = 14)  # Increase legend text
  ) +
  ylim(0, 100)

print(gaccess)
```

```
## Warning: Removed 2 rows containing missing values or values outside the sc
ale range
## (`geom_text()`).
```



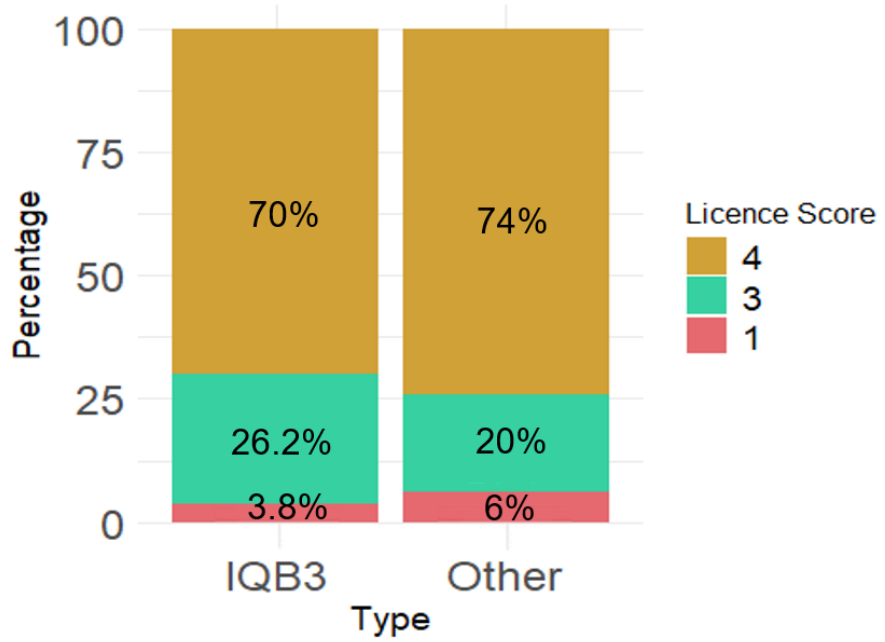

```
ggsave("gaccess.png", gaccess, width = 10, height = 8, dpi = 300)

## Warning: Removed 2 rows containing missing values or values outside the sc
ale range
## (`geom_text()`).
```

#bar chart for the Licence criteria by the Type of the study

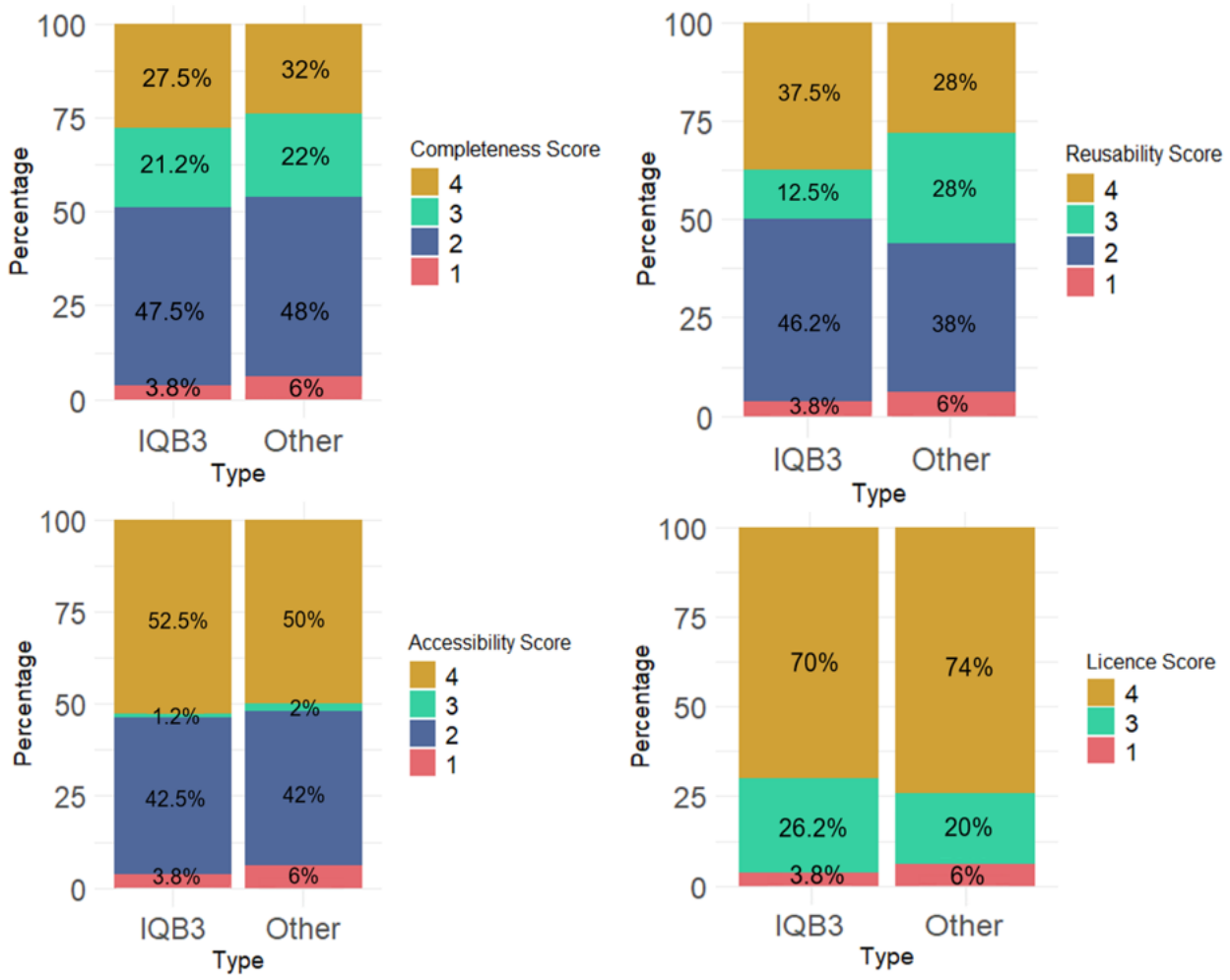
```
# Reorder the levels of variables
data$Licence <- factor(data$Licence, levels = c("4", "3", "2", "1"))

summarized_data_Licence <- data %>%
  group_by(Type, Licence) %>%
  summarise(Count = n(), .groups = "drop") %>%
  left_join(data %>% group_by(Type) %>% summarise(Total = n(), .groups = "dro
p"), by = "Type") %>%
  mutate(Percentage = Count/Total * 100)

summarized_data_Licence <- summarized_data_Licence %>%
  arrange(Type, desc(Licence)) %>%
  group_by(Type) %>%
  mutate(CumPercentage = cumsum(Percentage))
print(summarized_data_Licence)

## # A tibble: 6 × 6
## # Groups:   Type [2]
##   Type  Licence Count Total Percentage CumPercentage
```

```
##   <chr> <fct>   <int> <int>      <dbl>         <dbl>
## 1 IQB3  1           3    80       3.75          3.75
## 2 IQB3  3          21    80      26.2          30
## 3 IQB3  4          56    80      70           100
## 4 Other 1           3    50       6            6
## 5 Other 3          10    50      20           26
## 6 Other 4          37    50      74           100
```

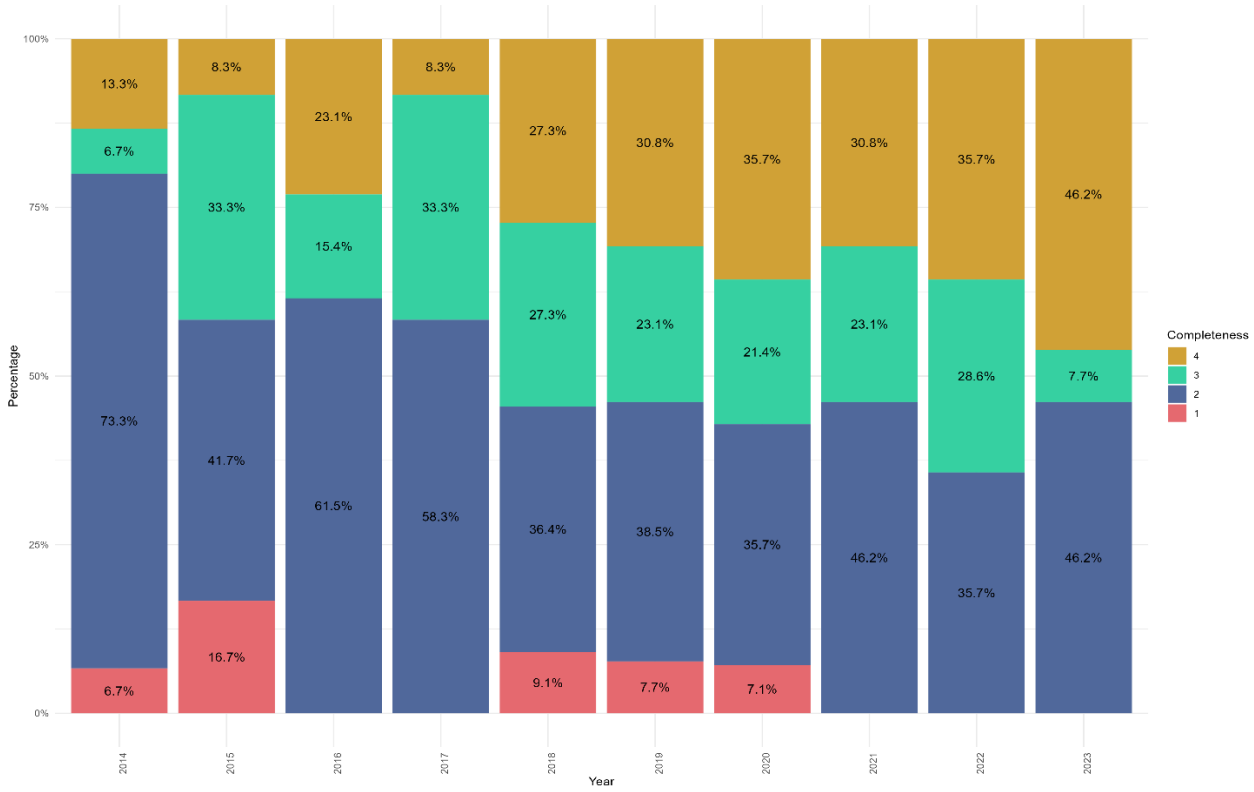
```r
colors <- c("#D0A136", "#36D0A1", "#E5696F")

# Visualization
glicence <- ggplot(summarized_data_Licence, aes(x = Type, y = Percentage, fil
l = as.factor(Licence))) +
  geom_bar(stat = "identity", position = "stack") +
  scale_fill_manual(values = colors) +
  labs(x = "Type", y = "Percentage", fill = "Licence Score") +
  theme_minimal() +
  geom_text(aes(label = paste0(round(Percentage, 1), "%"),
                y = CumPercentage - (0.5 * Percentage)),
            position = position_stack(vjust = 0.5), color = "Black", size = 5
) +
  theme(
    axis.title = element_text(size = 14), # Increase axis titles
    axis.text = element_text(size = 18),  # Increase axis text
    legend.title = element_text(size = 12), # Increase legend title
    legend.text = element_text(size = 14)  # Increase legend text
  ) +
  ylim(0, 100)

print(glicence)
```



```
ggsave("glicence.png", glicence, width = 10, height = 8, dpi = 300)

## Warning: Removed 2 rows containing missing values or values outside the
scale range
## (`geom_text()`).
library(patchwork)
## Warning: package 'patchwork' was built under R version 4.3.3
##
## Attaching package: 'patchwork'
## The following object is masked from 'package:MASS':
##
##     area
combinedplot <- gcomp + greuse + gaccess + glicence +
  plot_layout(
    ncol = 2, heights = c(10, 10), widths = c(10, 10)
  )
print(combinedplot)
```

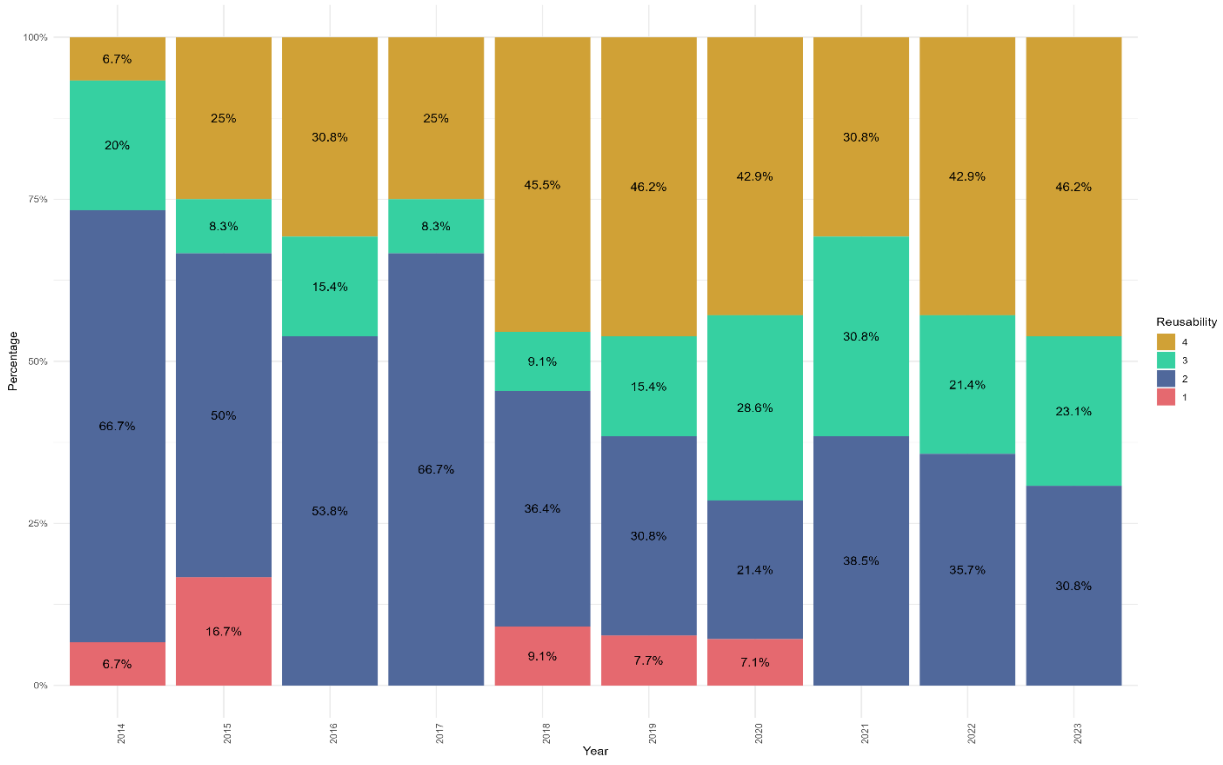


```r
##Calculate the frequency and percentage for each 'Complete' score within eac
h 'Year'
long_data <- data %>%
  count(Year, Complete) %>%
  group_by(Year) %>%
  mutate(Percentage = n / sum(n) * 100) %>%
  mutate(Label = ifelse(Percentage > 5, paste0(round(Percentage, 1), "%"), ""
)) %>%
  mutate(Cumulative_Percentage = cumsum(Percentage) - (0.5 * Percentage)) %>%
  ungroup()

# Custom colors
colors <- c("#D0A136", "#36D0A1", "#50689B", "#E5696F")

# Creating the stacked bar chart with percentage labels
Compyear <- ggplot(long_data, aes(x = as.factor(Year), y = n, fill = as.facto
r(Complete))) +
  geom_bar(stat = "identity", position = "fill") +  # Scale the bar heights t
o proportions
  scale_y_continuous(labels = percent_format()) +  # Convert the y-axis to pe
rcentage
  geom_text(
    aes(label = Label, y = Percentage),
    size = 4,
    color = "black",
    position = position_fill(vjust = 0.5)
  ) +
  scale_fill_manual(values = colors) +
  labs(x = "Year",
       y = "Percentage",
       fill = "Completeness") +
  theme(
    axis.title = element_text(size = 14), # Increase axis titles
    axis.text = element_text(size = 20),  # Increase axis text
    legend.title = element_text(size = 18), # Increase legend title
    legend.text = element_text(size = 20)  # Increase legend text
  ) +
  theme_minimal() +
  scale_x_discrete(name = "Year", labels = unique(long_data$Year)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))  # Adjust text ang
le for readability

# Print the plot
print(Compyear)
```



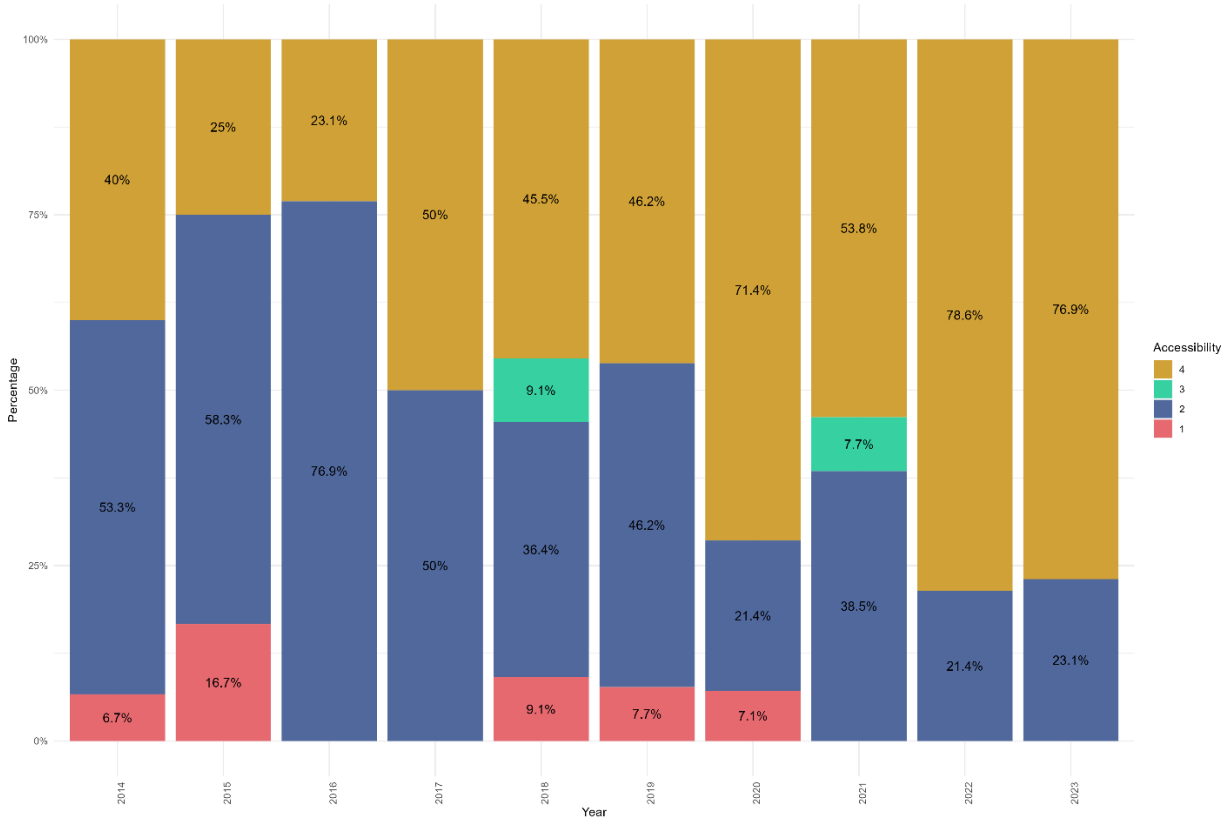

```
ggsave("Compyear.png", Compyear, width = 15, height = 10, units = "in", bg = "white")

# Calculate the frequency and percentage for each 'Reuse' score within each 'Year'
long_data <- data %>%
  count(Year, Reuse) %>%
  group_by(Year) %>%
  mutate(Percentage = n / sum(n) * 100) %>%
  mutate(Label = ifelse(Percentage > 5, paste0(round(Percentage, 1), "%"), "")) %>%
  mutate(Cumulative_Percentage = cumsum(Percentage) - (0.5 * Percentage)) %>%
  ungroup()

# Custom colors
colors <- c("#D0A136", "#36D0A1", "#50689B", "#E5696F")

# Creating the stacked bar chart with percentage labels
Reuseyear <- ggplot(long_data, aes(x = as.factor(Year), y = n, fill = as.factor(Reuse))) +
  geom_bar(stat = "identity", position = "fill") +  # Scale the bar heights to proportions
  scale_y_continuous(labels = percent_format()) +  # Convert the y-axis to percentage
  geom_text(
    aes(label = Label, y = Percentage),
```

```
    size = 4,
    color = "black",
    position = position_fill(vjust = 0.5)
  ) +
  scale_fill_manual(values = colors) +
  labs(x = "Year",
       y = "Percentage",
       fill = "Reusability") +
    theme(
    axis.title = element_text(size = 14), # Increase axis titles
    axis.text = element_text(size = 20),  # Increase axis text
    legend.title = element_text(size = 18), # Increase legend title
    legend.text = element_text(size = 20)  # Increase legend text
  ) +
  theme_minimal() +
  scale_x_discrete(name = "Year", labels = unique(long_data$Year)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))  # Adjust text ang
le for readability

# Print the plot
print(Reuseyear)
```



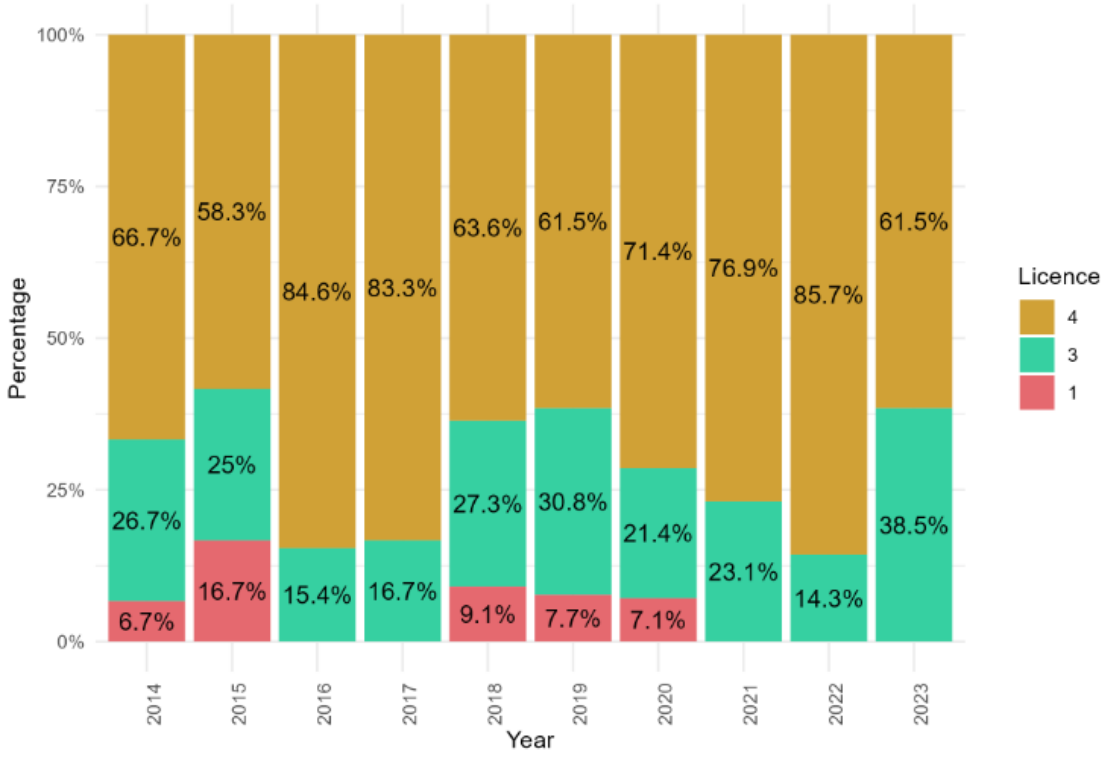

```
ggsave("Reuseyear.png", Reuseyear, width = 15, height = 10, units = "in", bg
= "white")

# Calculate the frequency and percentage for each 'Access' score within each
'Year'
```

```r
long_data <- data %>%
  count(Year, Access) %>%
  group_by(Year) %>%
  mutate(Percentage = n / sum(n) * 100) %>%
  mutate(Label = ifelse(Percentage > 5, paste0(round(Percentage, 1), "%"), ""
)) %>%
  mutate(Cumulative_Percentage = cumsum(Percentage) - (0.5 * Percentage)) %>%
  ungroup()

# Custom colors
colors <- c("#D0A136", "#36D0A1", "#50689B", "#E5696F")

# Creating the stacked bar chart with percentage labels
accessyear <- ggplot(long_data, aes(x = as.factor(Year), y = n, fill = as.fac
tor(Access))) +
  geom_bar(stat = "identity", position = "fill") +  # Scale the bar heights t
o proportions
  scale_y_continuous(labels = percent_format()) +  # Convert the y-axis to pe
rcentage
  geom_text(
    aes(label = Label, y = Percentage),
    size = 4,
    color = "black",
    position = position_fill(vjust = 0.5)
  ) +
  scale_fill_manual(values = colors) +
  labs(x = "Year",
       y = "Percentage",
       fill = "Accessibility") +
  theme(
    axis.title = element_text(size = 14), # Increase axis titles
    axis.text = element_text(size = 20),  # Increase axis text
    legend.title = element_text(size = 18), # Increase legend title
    legend.text = element_text(size = 20)  # Increase legend text
  ) +
    theme_minimal() +
  scale_x_discrete(name = "Year", labels = unique(long_data$Year)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))  # Adjust text ang
le for readability

# Print the plot
print(accessyear)
```



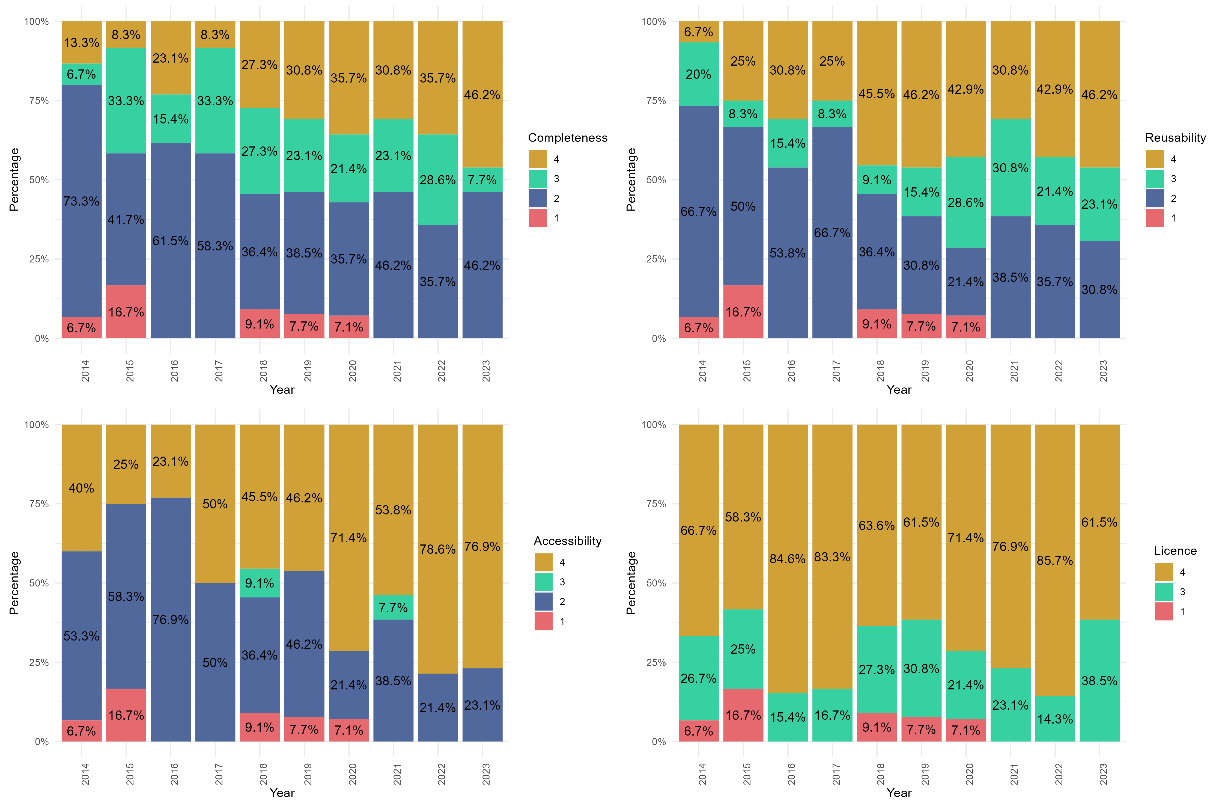

```
ggsave("accessyear.png", accessyear, width = 15, height = 10, units = "in", b
g = "white")

# Calculate the frequency and percentage for each 'Licence' score within each
'Year'
long_data <- data %>%
  count(Year, Licence) %>%
  group_by(Year) %>%
  mutate(Percentage = n / sum(n) * 100) %>%
  mutate(Label = ifelse(Percentage > 5, paste0(round(Percentage, 1), "%"), ""
)) %>%
  mutate(Cumulative_Percentage = cumsum(Percentage) - (0.5 * Percentage)) %>%
  ungroup()

# Custom colors
colors <- c("#D0A136", "#36D0A1", "#50689B", "#E5696F")

# Creating the stacked bar chart with percentage labels
licenceyear <- ggplot(long_data, aes(x = as.factor(Year), y = n, fill = as.fa
ctor(Licence))) +
  geom_bar(stat = "identity", position = "fill") +  # Scale the bar heights t
o proportions
  scale_y_continuous(labels = percent_format()) +  # Convert the y-axis to pe
rcentage
  geom_text(
```

```
    aes(label = Label, y = Percentage),
    size = 4,
    color = "black",
    position = position_fill(vjust = 0.5)
  ) +
  scale_fill_manual(values = colors) +
  labs(x = "Year",
       y = "Percentage",
       fill = "Licence") +
  theme(
    axis.title = element_text(size = 14), # Increase axis titles
    axis.text = element_text(size = 20),  # Increase axis text
    legend.title = element_text(size = 18), # Increase legend title
    legend.text = element_text(size = 20)  # Increase legend text
  ) +
  theme_minimal() +
  scale_x_discrete(name = "Year", labels = unique(long_data$Year)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))  # Adjust text ang
le for readability

# Print the plot
print(licenceyear)
```



```
ggsave("licenceyear.png", licenceyear, width = 15, height = 10, units = "in",
bg = "white")
```

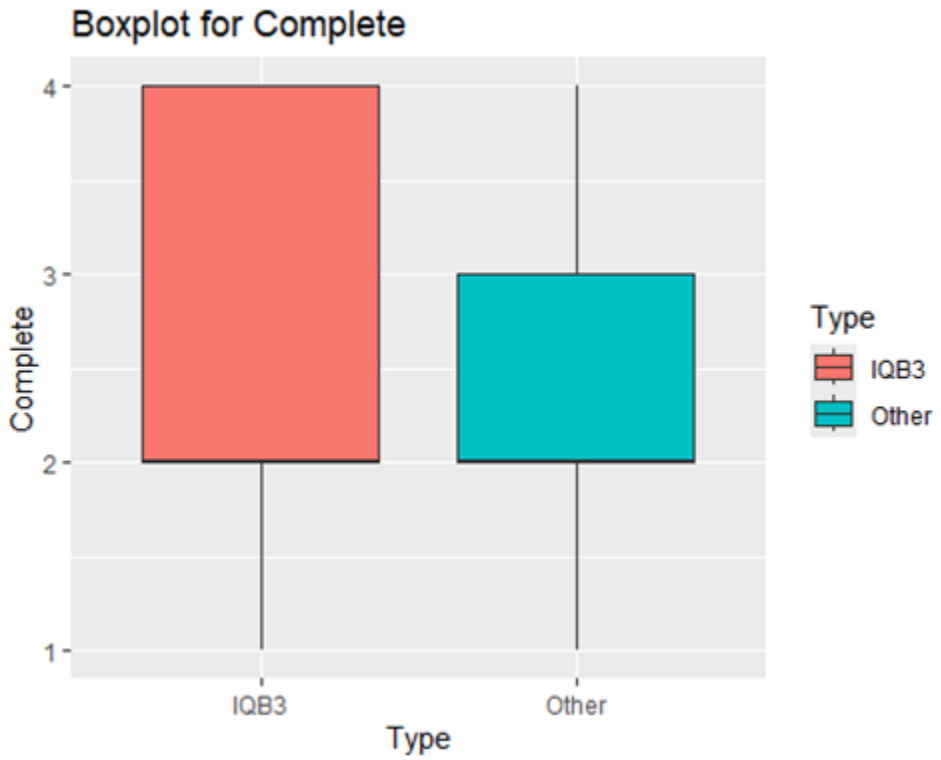
```
plotyear <- Compyear + reuseyear + accessyear + licenceyear +
  plot_layout(
    ncol = 2, heights = c(10, 10), widths = c(10, 10)
  )
print(plotyear)
```



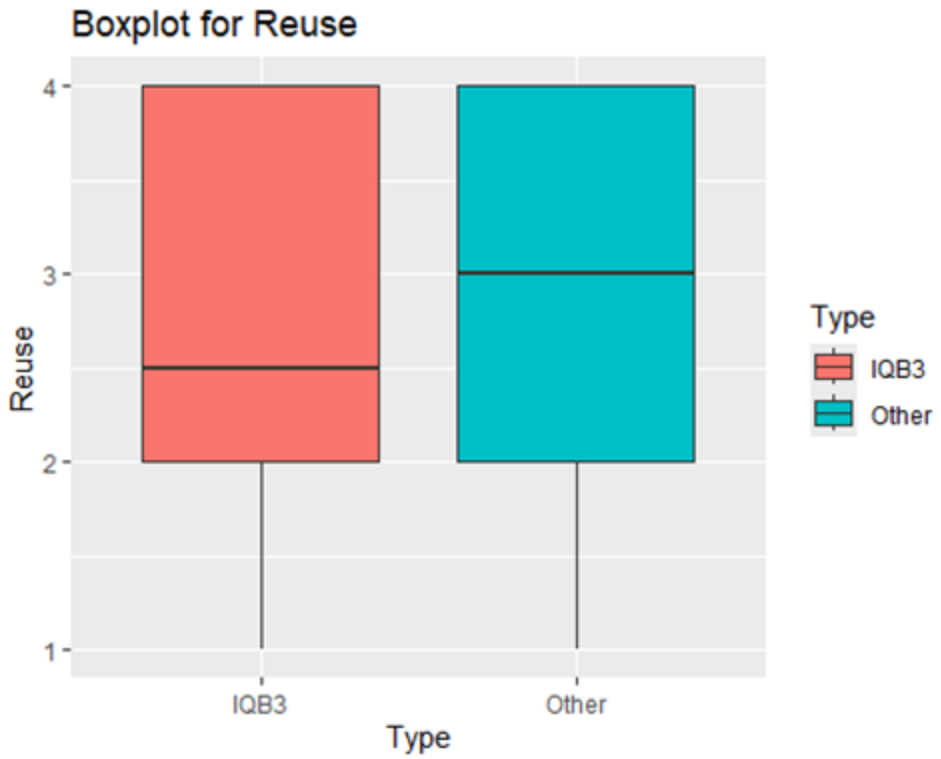
#study the significant difference between two type of the research for each scoring criteria

```
# Convert scoring criteria to numeric (if not already)
data <- data %>%
  mutate(across(c(Complete, Reuse, Access, Licence),
~as.numeric(as.character(.))))

# Assumption checks for each criterion
for (criterion in c("Complete", "Reuse", "Access", "Licence")) {
  # Create and display a box plot to check the distribution shape and spread
  print(ggplot(data, aes_string(x = "Type", y = criterion, fill = "Type")) +
    geom_boxplot() +
    ggtitle(paste("Boxplot for", criterion)))

  # Print the median for a basic comparison
```

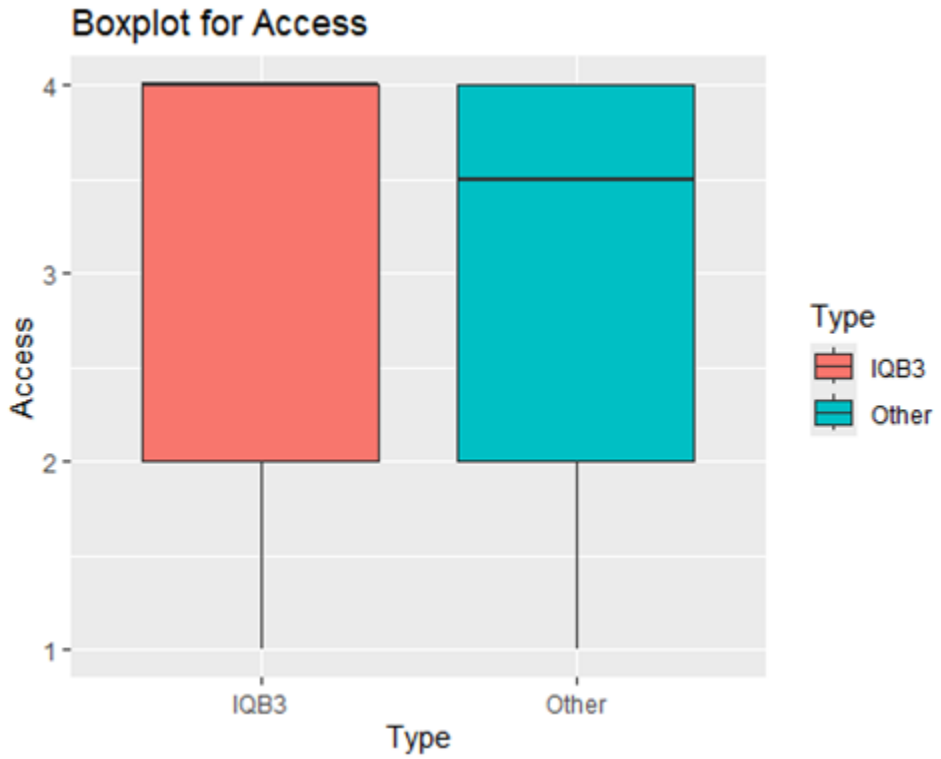
```
  cat(paste("Median of", criterion, "by Type:\n"))
  print(aggregate(. ~ Type, data[c("Type", criterion)], median))

  cat("\n")
}
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



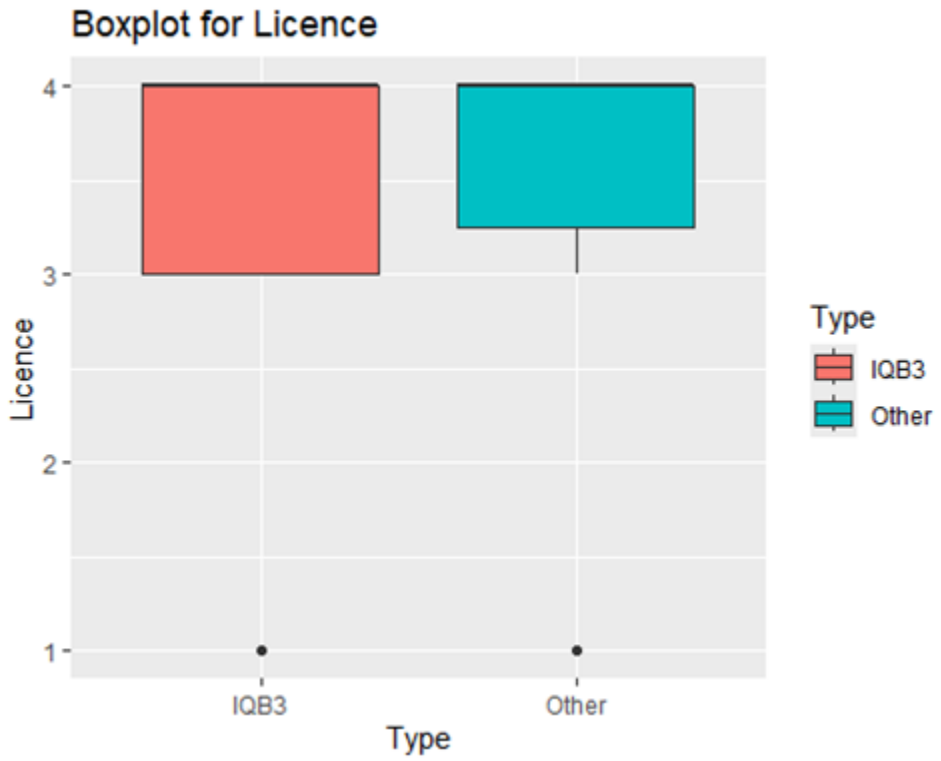

```
## Median of Complete by Type:
##    Type Complete
## 1  IQB3        2
## 2 Other        2
```




```
## Median of Reuse by Type:
##    Type Reuse
## 1  IQB3   2.5
## 2 Other   3.0
```



```
## Median of Access by Type:
##    Type Access
## 1  IQB3    4.0
## 2 Other    3.5
```




```
## Median of Licence by Type:
##    Type Licence
## 1  IQB3       4
## 2 Other       4
```

```r
# Function to perform Mann-Whitney U test
perform_mann_whitney <- function(data, criterion) {
  test_result <- wilcox.test(
    reformulate("Type", response = criterion),
    data = data,
    exact = FALSE
  )
  list(criterion = criterion, p.value = test_result$p.value)
}

# Perform the test for each criterion
mw_results <- lapply(c("Complete", "Reuse", "Access", "Licence"),
function(criterion) {
  perform_mann_whitney(data, criterion)
})

# Convert the list of results to a dataframe
mw_results_df <- do.call(rbind, mw_results)
```

```
print(mw_results_df)
##      criterion  p.value
## [1,] "Complete" 0.6265178
## [2,] "Reuse"    0.8119625
## [3,] "Access"   0.7360586
## [4,] "Licence"  0.7018642
#The assumptions of the test were met:
### Ordinal Data Check: The scoring criteria are ordinal variables
### Similar Distribution Shapes: For each criterion, it creates a box plot to
visually inspect the distribution shapes. This is crucial to check if the
distributions are similar across groups.
### Independence of Observations:the variables are independence
observations.
```

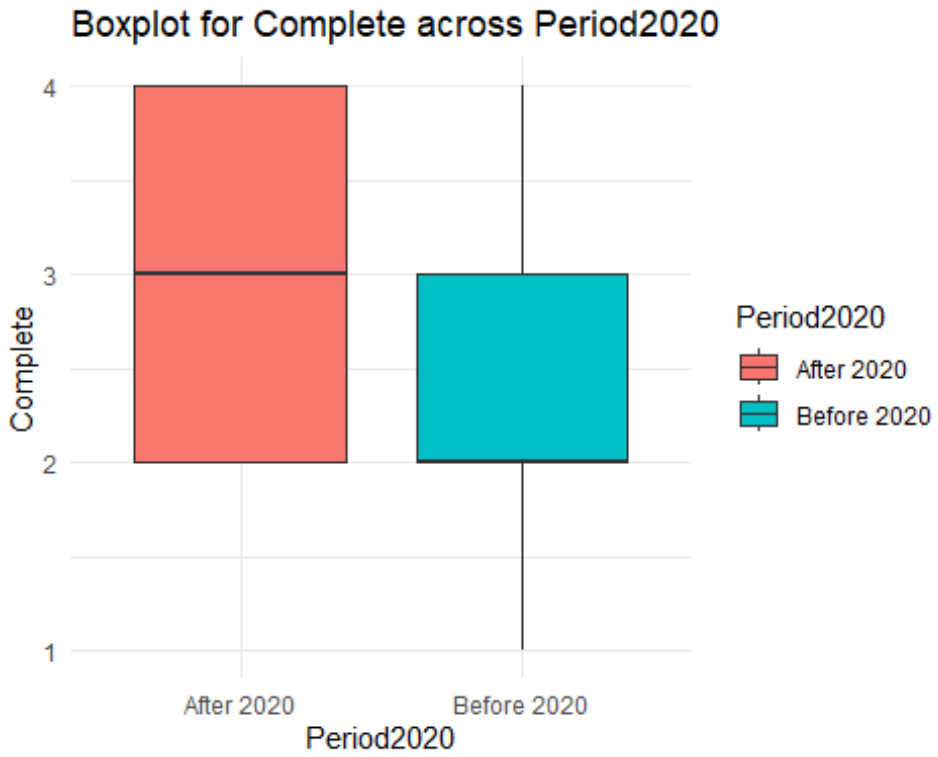
#Create variables for FAIR principles in 2016 and COVID-19 in 2020

```
data <- data %>%
  mutate(Period2020 = ifelse(Year <= 2020, "Before 2020", "After 2020")) %>%
  mutate(Period2016 = ifelse(Year <= 2016, "Before 2016", "After 2016"))#
```

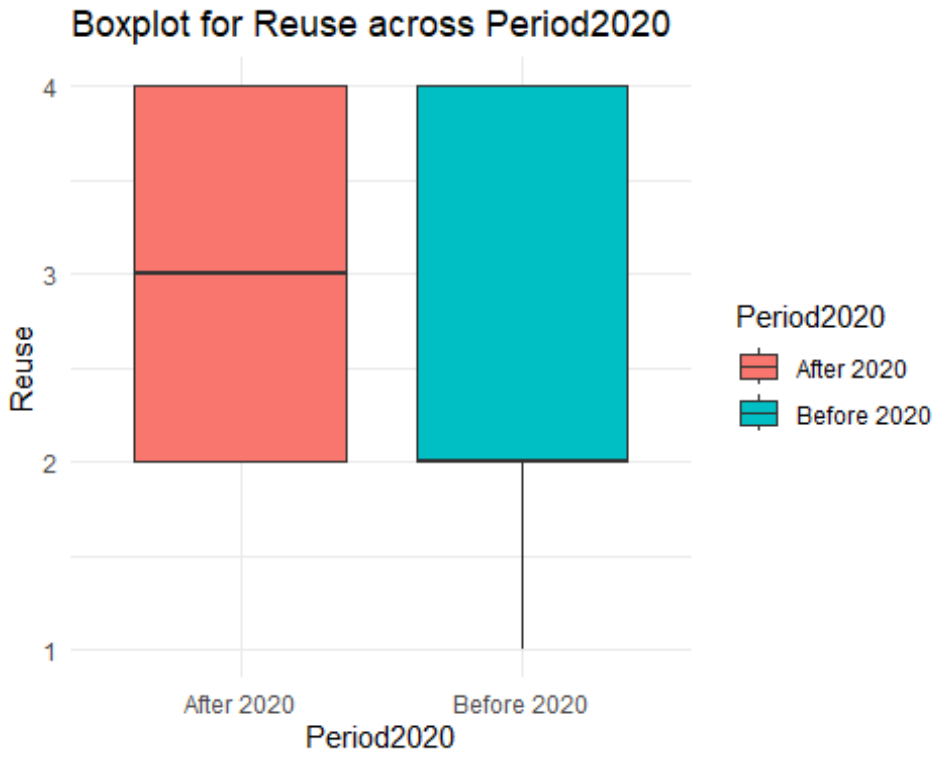
#Study the significant difference before and afterCovid-19 2020 using Median and Mann Whitney U test

```
# Ensure the scoring criteria are numeric
data <- data %>%
  mutate(across(c(Complete, Reuse, Access, Licence), ~as.numeric(as.character
(.))))

# Function to generate box plots for distribution checks
generate_boxplot <- function(data, score_var, period_var) {
  ggplot(data, aes_string(x = period_var, y = score_var, fill = period_var))
+
    geom_boxplot() +
    labs(title = paste("Boxplot for", score_var, "across", period_var),
         x = period_var,
         y = score_var) +
    theme_minimal()
}

# Generate and display box plots for each score with Period2020
lapply(c("Complete", "Reuse", "Access", "Licence"), function(score_var) {
  generate_boxplot(data, score_var, "Period2020")
})

## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
```

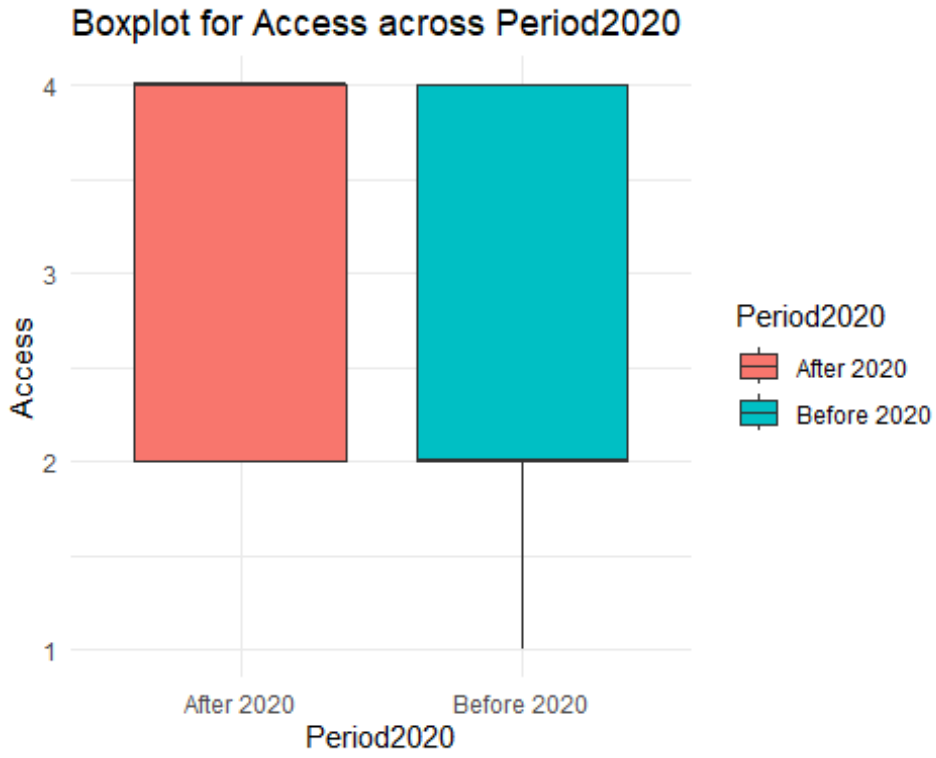
```
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## [[1]]
```



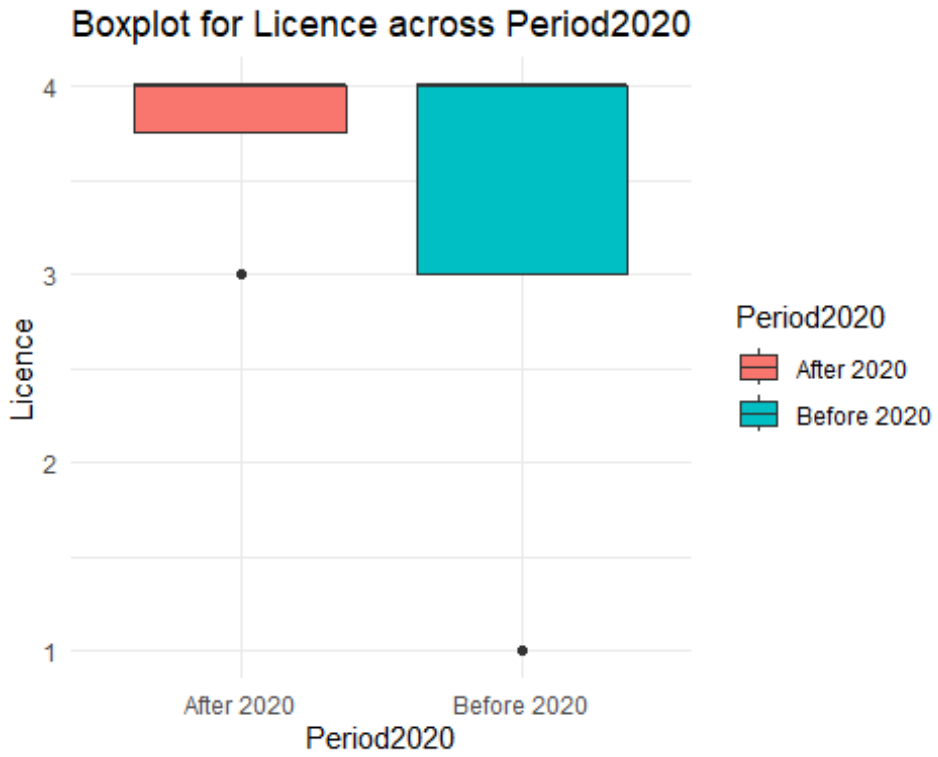

```
##
## [[2]]
```


Boxplot for Reuse across Period2020
Reuse
4
3
2
1
After 2020
Before 2020
Period2020
Period2020
After 2020
Before 2020

```
##
## [[3]]
```


Boxplot for Access across Period2020
Access
4
3
2
1
After 2020
Before 2020
Period2020
Period2020
After 2020
Before 2020

```
##
## [[4]]
```




```
# Function to perform the Mann-Whitney test and calculate medians
perform_analysis <- function(data, score_var, period_var) {
  test_result <- wilcox.test(reformulate(period_var, score_var), data = data,
exact = FALSE)

  medians <- data %>%
    group_by(!!sym(period_var)) %>%
    summarise(median = median(!!sym(score_var), na.rm = TRUE), .groups = 'dro
p')

  list(median = medians, p.value = test_result$p.value)
}

# Analysis for each score for 2020
results_complete_2020 <- perform_analysis(data, "Complete", "Period2020")
results_Reuse_2020 <- perform_analysis(data, "Reuse", "Period2020")
results_access_2020 <- perform_analysis(data, "Access", "Period2020")
results_licence_2020 <- perform_analysis(data, "Licence", "Period2020")

# Print results
print(results_complete_2020)

## $median
## # A tibble: 2 × 2
```

```
##   Period2020  median
##   <chr>        <dbl>
## 1 After 2020       3
## 2 Before 2020      2
##
## $p.value
## [1] 0.0394013

print(results_Reuse_2020)

## $median
## # A tibble: 2 × 2
##   Period2020  median
##   <chr>        <dbl>
## 1 After 2020       3
## 2 Before 2020      2
##
## $p.value
## [1] 0.0639665

print(results_access_2020)

## $median
## # A tibble: 2 × 2
##   Period2020  median
##   <chr>        <dbl>
## 1 After 2020       4
## 2 Before 2020      2
##
## $p.value
## [1] 0.002491228

print(results_licence_2020)

## $median
## # A tibble: 2 × 2
##   Period2020  median
##   <chr>        <dbl>
## 1 After 2020       4
## 2 Before 2020      4
##
## $p.value
## [1] 0.4440382

#The assumptions of the test were met:
### Ordinal Data Check: The scoring criteria are ordinal variables
### Similar Distribution Shapes: For each criterion, it creates a box plot to
visually inspect the distribution shapes. This is crucial to check if the dis
tributions are similar across groups.
### Independence of Observations:the variables are independence observations.
```

#Study the significant difference before and afterCovid-19 2020 using Median and Mann Whitney U test for the IQB3 Papers

```
data_IQB3 <- data %>%
  filter(Type == "IQB3")


perform_analysis <- function(data_IQB3, score_var, period_var) {
  test_result <- wilcox.test(reformulate(period_var, score_var), data = data_
IQB3, exact = FALSE)

  medians <- data_IQB3 %>%
    group_by(!!sym(period_var)) %>%
    summarise(median = median(!!sym(score_var), na.rm = TRUE), .groups = 'dro
p')

  list(median = medians, p.value = test_result$p.value)
}

# Analysis for each score for 2020
results_complete_2020_IQB3 <- perform_analysis(data_IQB3, "Complete", "Period
2020")
results_Reuse_2020_IQB3 <- perform_analysis(data_IQB3, "Reuse", "Period2020")
results_access_2020_IQB3 <- perform_analysis(data_IQB3, "Access", "Period2020
")
results_licence_2020_IQB3 <- perform_analysis(data_IQB3, "Licence", "Period20
20")

# Print results
results_complete_2020_IQB3
```

```
## $median
## # A tibble: 2 × 2
##   Period2020  median
##   <chr>        <dbl>
## 1 After 2020       3
## 2 Before 2020      2
##
## $p.value
## [1] 0.3644481
```

```
results_Reuse_2020_IQB3
```

```
## $median
## # A tibble: 2 × 2
##   Period2020  median
##   <chr>        <dbl>
## 1 After 2020       3
## 2 Before 2020      2
##
```

```
## $p.value
## [1] 0.268142

results_access_2020_IQB3

## $median
## # A tibble: 2 × 2
##   Period2020  median
##   <chr>        <dbl>
## 1 After 2020       4
## 2 Before 2020      2
##
## $p.value
## [1] 0.01209852

results_licence_2020_IQB3

## $median
## # A tibble: 2 × 2
##   Period2020  median
##   <chr>        <dbl>
## 1 After 2020       4
## 2 Before 2020      4
##
## $p.value
## [1] 0.7285709
```

#Study the significant difference before and after Covid-19 2020 using Median and Mann Whitney U test for the Other Papers

```
data_Other <- data %>%
  filter(Type == "Other")


perform_analysis <- function(data_Other, score_var, period_var) {
  test_result <- wilcox.test(reformulate(period_var, score_var), data = data_
Other, exact = FALSE)

  medians <- data_Other %>%
    group_by(!!sym(period_var)) %>%
    summarise(median = median(!!sym(score_var), na.rm = TRUE), .groups = 'dro
p')

  list(median = medians, p.value = test_result$p.value)
}

# Analysis for each score for 2020
results_complete_2020_Other <- perform_analysis(data_Other, "Complete", "Peri
od2020")
results_Reuse_2020_Other <- perform_analysis(data_Other, "Reuse", "Period2020
")
```

```r
results_access_2020_Other <- perform_analysis(data_Other, "Access", "Period2020")
results_licence_2020_Other <- perform_analysis(data_Other, "Licence", "Period2020")

# Print results
results_complete_2020_Other
```

```
## $median
## # A tibble: 2 × 2
##   Period2020  median
##   <chr>        <dbl>
## 1 After 2020     3.5
## 2 Before 2020    2
##
## $p.value
## [1] 0.03066192
```

```r
results_Reuse_2020_Other
```

```
## $median
## # A tibble: 2 × 2
##   Period2020  median
##   <chr>        <dbl>
## 1 After 2020     3
## 2 Before 2020    2.5
##
## $p.value
## [1] 0.1037424
```

```r
results_access_2020_Other
```

```
## $median
## # A tibble: 2 × 2
##   Period2020  median
##   <chr>        <dbl>
## 1 After 2020       4
## 2 Before 2020      2
##
## $p.value
## [1] 0.1016182
```

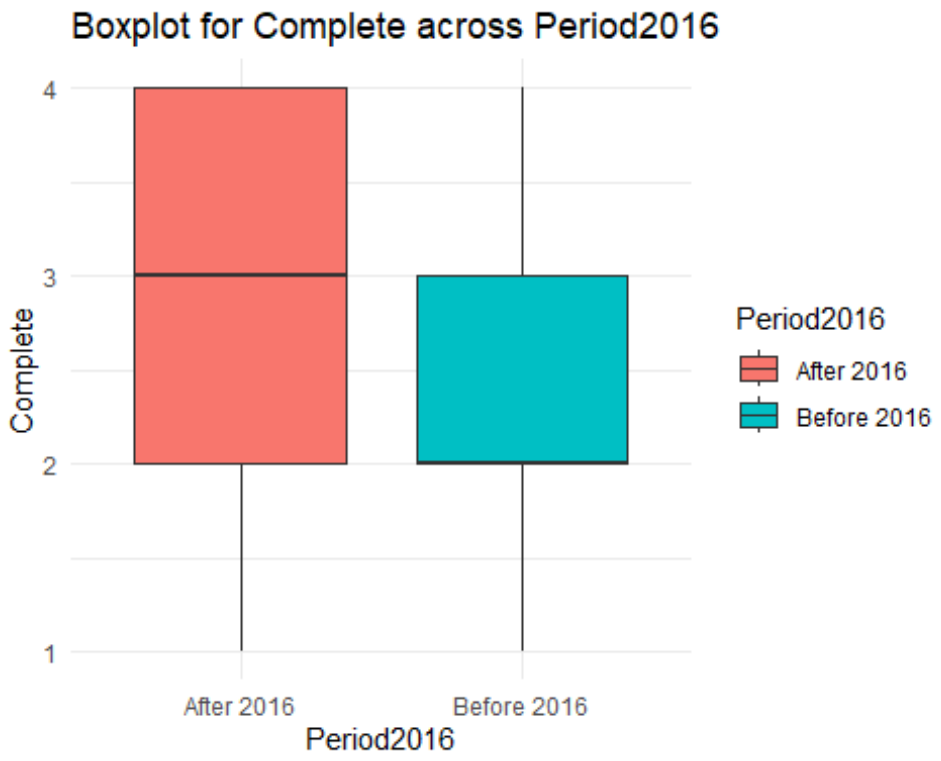
```r
results_licence_2020_Other
```

```
## $median
## # A tibble: 2 × 2
##   Period2020  median
##   <chr>        <dbl>
## 1 After 2020       4
## 2 Before 2020      4
##
```

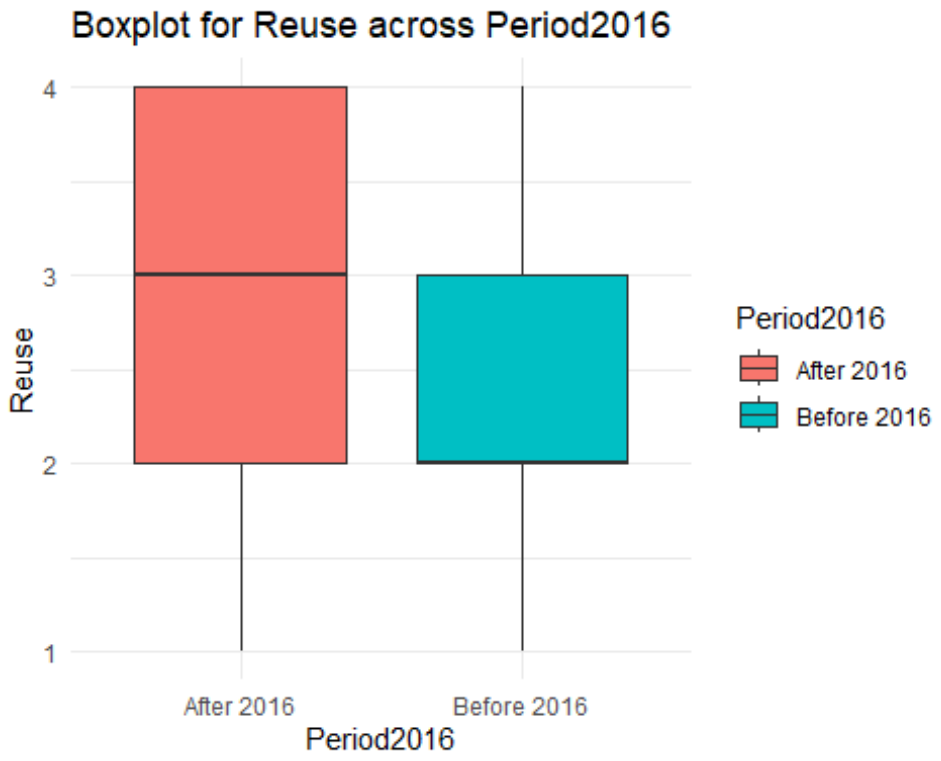
```
## $p.value
## [1] 0.365762
```

#Study the significant difference before and after Fair 2016 using Median and Mann Whitney U for all the papers

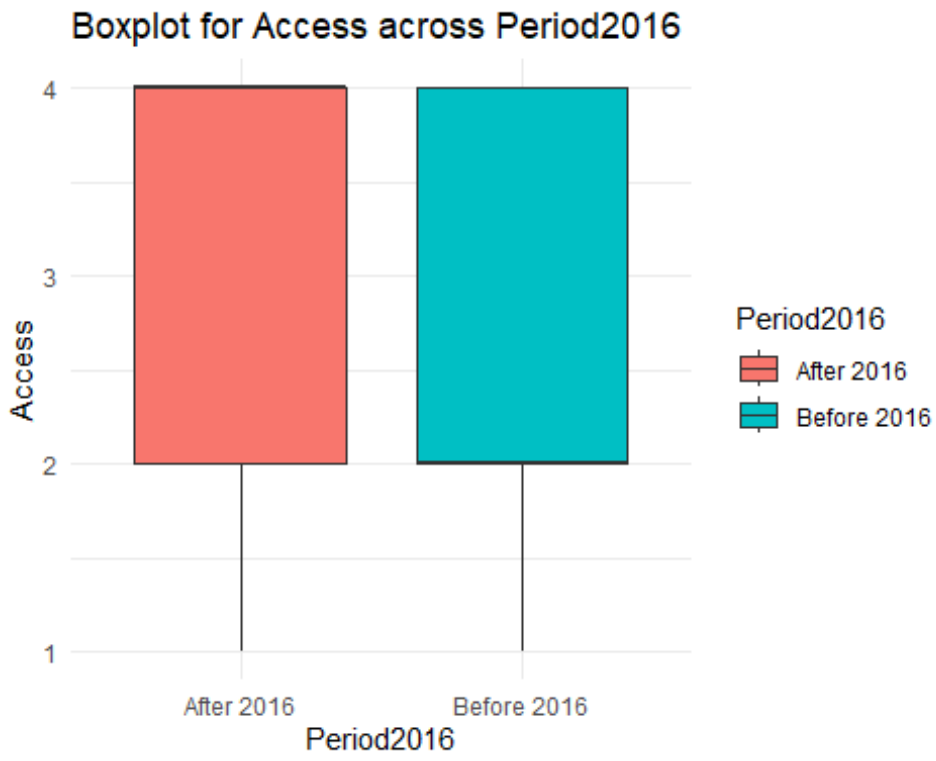
```
data <- data %>%
  mutate(Period2016 = ifelse(Year <= 2016, "Before 2016", "After 2016"))

# Function to perform the Mann-Whitney test, calculate medians, and generate box plots
perform_analysis <- function(data, score_var, period_var) {
  test_result <- wilcox.test(reformulate(period_var, score_var), data = data, exact = FALSE)

  medians <- data %>%
    group_by(!!sym(period_var)) %>%
    summarise(median = median(!!sym(score_var), na.rm = TRUE), .groups = 'drop')

  # Generate a box plot
  box_plot <- ggplot(data, aes_string(x = period_var, y = score_var, fill = period_var)) +
    geom_boxplot() +
    labs(title = paste("Boxplot for", score_var, "across", period_var),
         x = period_var,
         y = score_var) +
    theme_minimal()

  # Print the box plot
  print(box_plot)

  list(median = medians, p.value = test_result$p.value, plot = box_plot)
}

# Analysis for each score for 2016
results_complete_2016 <- perform_analysis(data, "Complete", "Period2016")
```



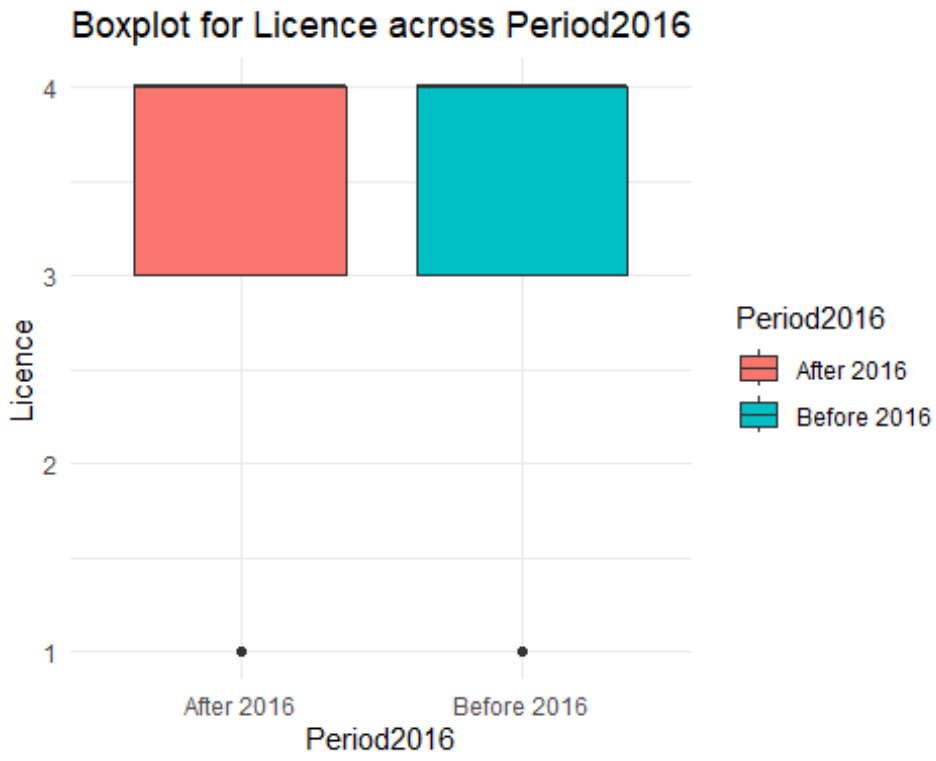

```
results_Reuse_2016 <- perform_analysis(data, "Reuse", "Period2016")
```




```
results_access_2016 <- perform_analysis(data, "Access", "Period2016")
```

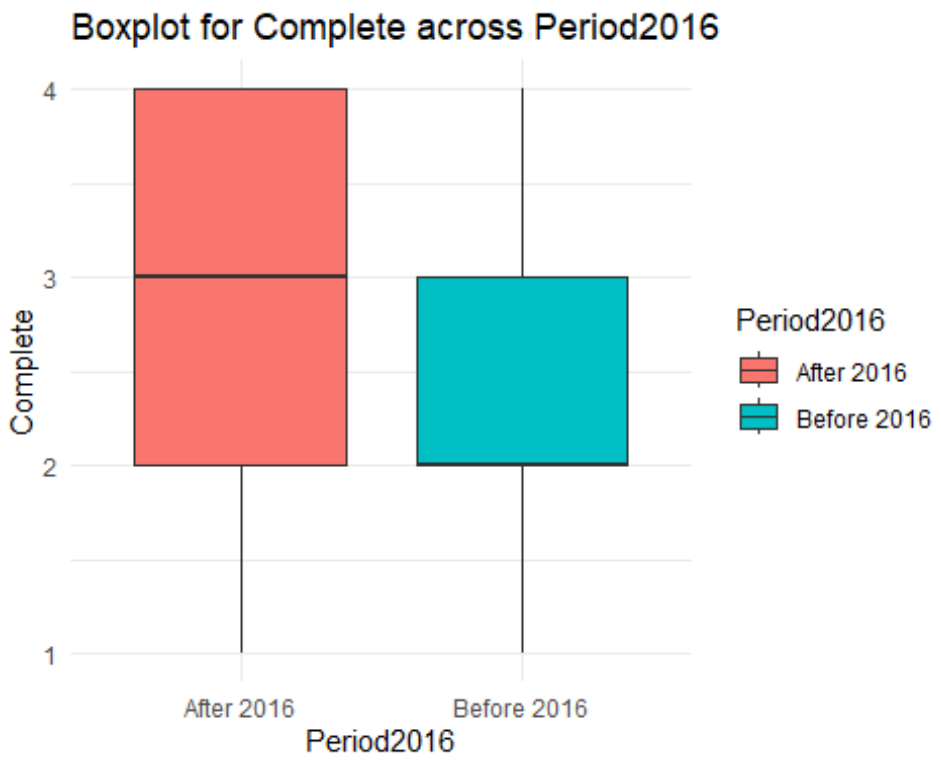



```
results_licence_2016 <- perform_analysis(data, "Licence", "Period2016")
```



```
# Print results
results_complete_2016
```

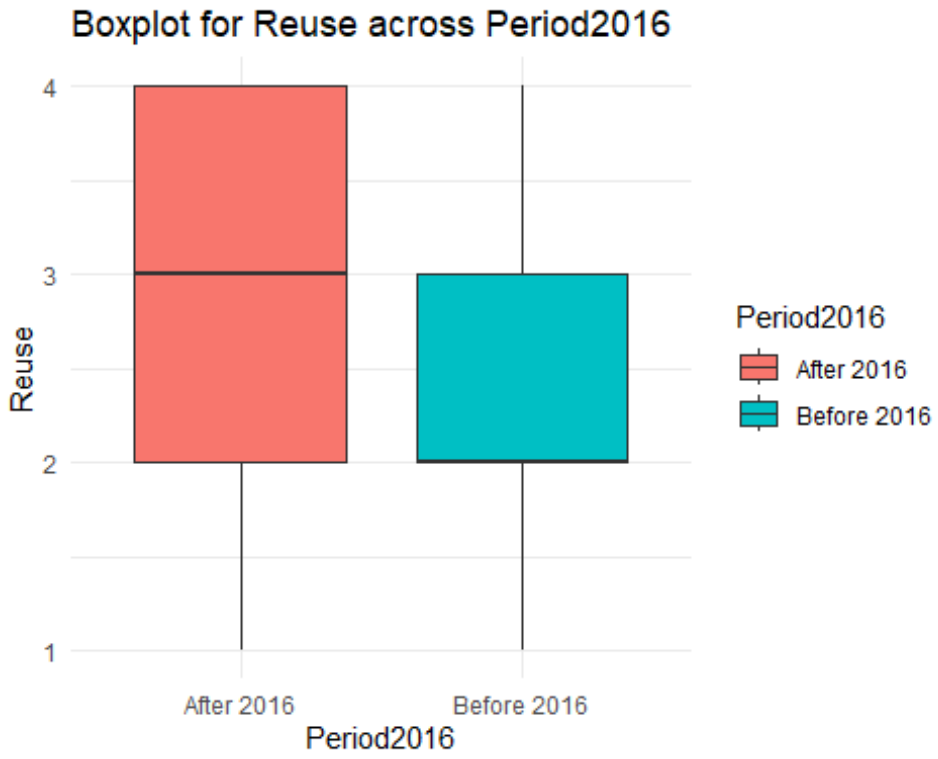
```
## $median
## # A tibble: 2 × 2
##   Period2016  median
##   <chr>        <dbl>
## 1 After 2016       3
## 2 Before 2016      2
##
## $p.value
## [1] 0.01447046
##
## $plot
```



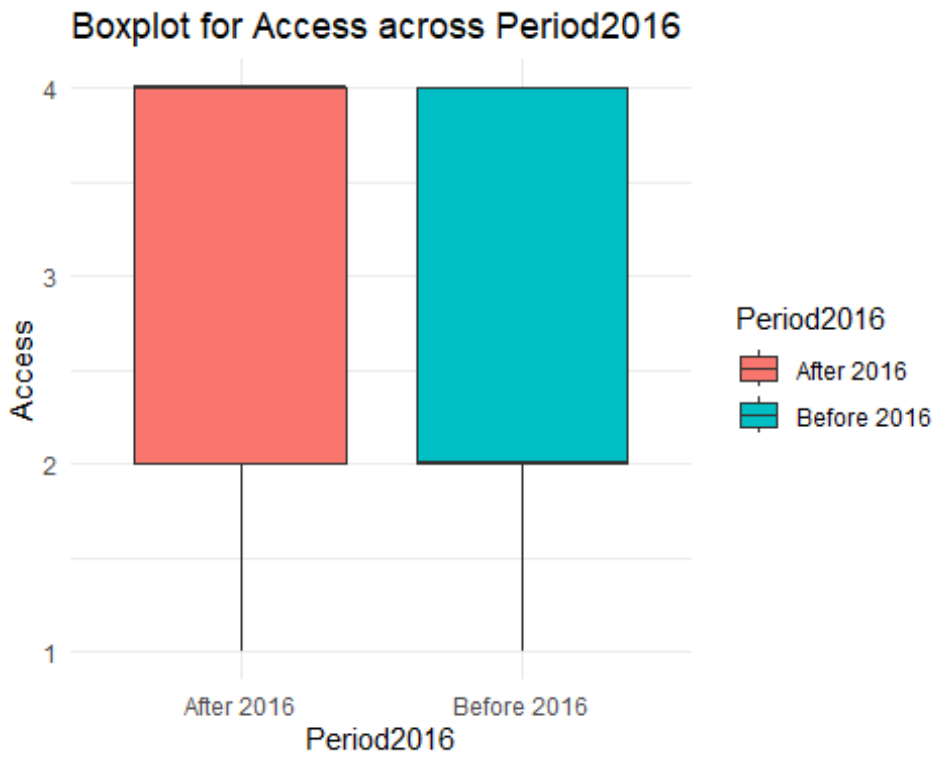

```
results_Reuse_2016
```

```
## $median
## # A tibble: 2 × 2
##   Period2016  median
##   <chr>        <dbl>
## 1 After 2016       3
## 2 Before 2016      2
##
## $p.value
## [1] 0.006953216
```

```
##
## $plot
```

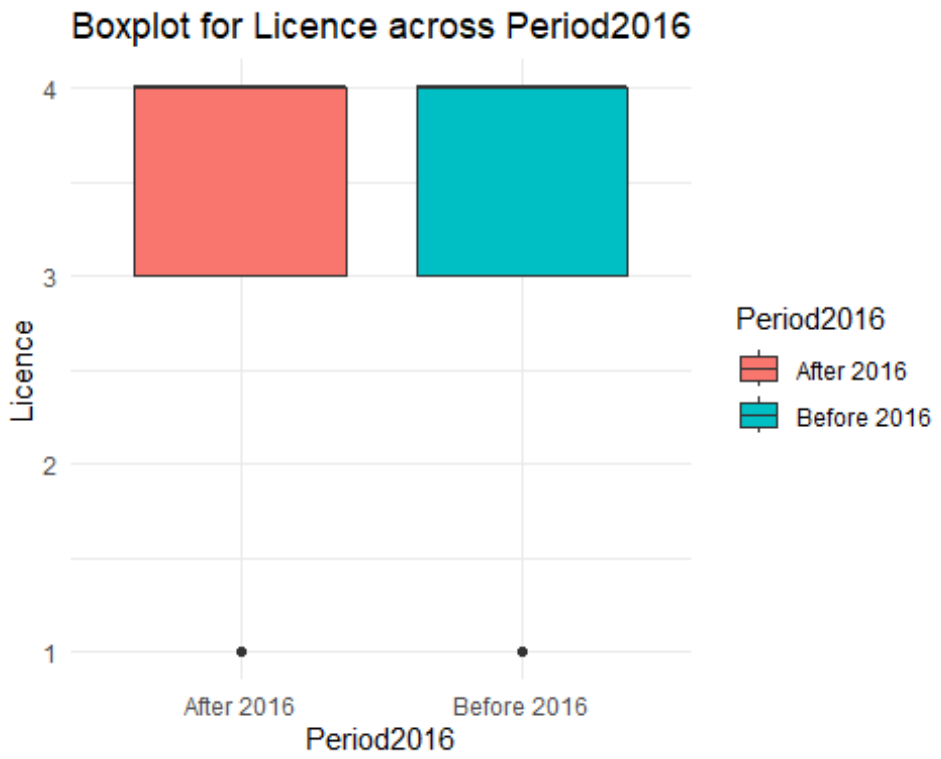



```
results_access_2016

## $median
## # A tibble: 2 × 2
##   Period2016  median
##   <chr>        <dbl>
## 1 After 2016       4
## 2 Before 2016      2
##
## $p.value
## [1] 0.0007181409
##
## $plot
```




```
results_licence_2016
```

```
## $median
## # A tibble: 2 × 2
##   Period2016  median
##   <chr>        <dbl>
## 1 After 2016       4
## 2 Before 2016      4
##
## $p.value
## [1] 0.7055106
##
## $plot
```



#Study the significant difference before and after Fair 2016 using Median and Mann Whitney U test for the IQB3 Papers

```
perform_analysis <- function(data_IQB3, score_var, period_var) {
  test_result <- wilcox.test(reformulate(period_var, score_var), data = data_
IQB3, exact = FALSE)

  medians <- data_IQB3 %>%
    group_by(!!sym(period_var)) %>%
    summarise(median = median(!!sym(score_var), na.rm = TRUE), .groups = 'dro
p')

  list(median = medians, p.value = test_result$p.value)
}
# Analysis for each score for 2016
results_complete_2016_IQB3 <- perform_analysis(data_IQB3, "Complete", "Period
2016")
results_Reuse_2016_IQB3 <- perform_analysis(data_IQB3, "Reuse", "Period2016")
results_access_2016_IQB3 <- perform_analysis(data_IQB3, "Access", "Period2016
")
results_licence_2016_IQB3 <- perform_analysis(data_IQB3, "Licence", "Period20
16")

# Print results
results_complete_2016_IQB3
```

```
## $median
## # A tibble: 2 × 2
##   Period2016  median
##   <chr>        <dbl>
## 1 After 2016       3
## 2 Before 2016      2
##
## $p.value
## [1] 0.3265723
```

```
results_Reuse_2016_IQB3
```

```
## $median
## # A tibble: 2 × 2
##   Period2016  median
##   <chr>        <dbl>
## 1 After 2016       3
## 2 Before 2016      2
##
## $p.value
## [1] 0.1311435
```

```
results_access_2016_IQB3
```

```
## $median
## # A tibble: 2 × 2
##   Period2016  median
##   <chr>        <dbl>
## 1 After 2016       4
## 2 Before 2016      2
##
## $p.value
## [1] 0.004984167
```

```
results_licence_2016_IQB3
```

```
## $median
## # A tibble: 2 × 2
##   Period2016  median
##   <chr>        <dbl>
## 1 After 2016       4
## 2 Before 2016      4
##
## $p.value
## [1] 0.5687401
```

#Study the significant difference before and after Fair 2016 using Median and Mann Whitney U test for the Other Papers

```
# Analysis for each score for 2016
results_complete_2016_Other <- perform_analysis(data_Other, "Complete", "Peri
```

```
od2016")
results_Reuse_2016_Other <- perform_analysis(data_Other, "Reuse", "Period2016
")
results_access_2016_Other <- perform_analysis(data_Other, "Access", "Period20
16")
results_licence_2016_Other <- perform_analysis(data_Other, "Licence", "Period
2016")

# Print results
results_complete_2016_Other
```

```
## $median
## # A tibble: 2 × 2
##   Period2016  median
##   <chr>        <dbl>
## 1 After 2016       3
## 2 Before 2016      2
##
## $p.value
## [1] 0.01351854
```

```
results_Reuse_2016_Other
```

```
## $median
## # A tibble: 2 × 2
##   Period2016  median
##   <chr>        <dbl>
## 1 After 2016       3
## 2 Before 2016      2
##
## $p.value
## [1] 0.01741414
```

```
results_access_2016_Other
```

```
## $median
## # A tibble: 2 × 2
##   Period2016  median
##   <chr>        <dbl>
## 1 After 2016       4
## 2 Before 2016      2
##
## $p.value
## [1] 0.05851632
```

```
results_licence_2016_Other
```

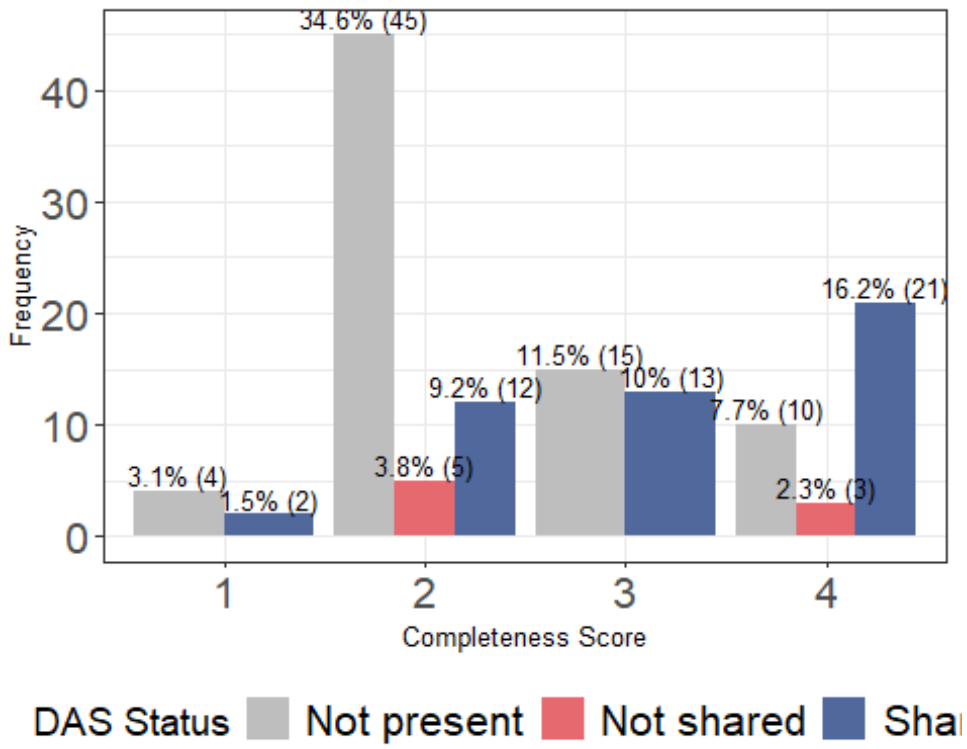
```
## $median
## # A tibble: 2 × 2
##   Period2016  median
##   <chr>        <dbl>
```

```
## 1 After 2016       4
## 2 Before 2016      4
##
## $p.value
## [1] 1
```

#plot the distribution of DAS for each completeness score

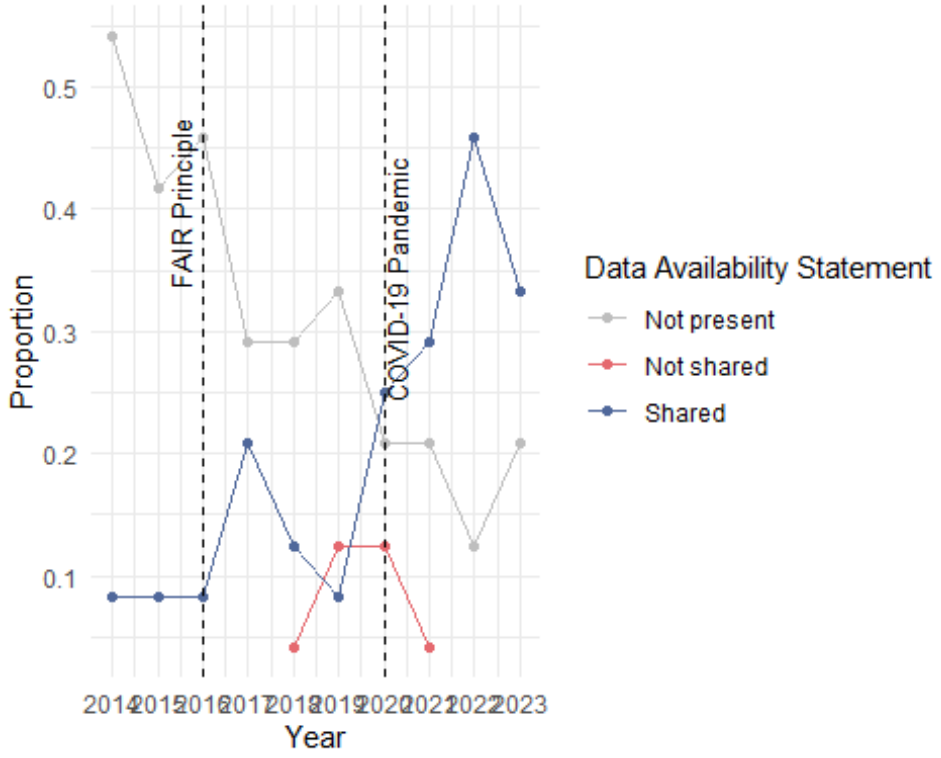
```
# Transform the DAS variable
data <- data %>%
  mutate(NewDAS = case_when(
    is.na(DAS) ~ "Not present",
    DAS == 1 ~ "Shared",
    DAS == 0 ~ "Not shared",
    TRUE ~ as.character(DAS)))  # This line is just a fallback to handle unex
pected values

class(data$NewDAS)

## [1] "character"

# Convert the new DAS variable to a factor for plotting
data$NewDAS <- factor(data$NewDAS, levels = c("Not present", "Not shared", "S
hared"))

# Calculate frequency and percentage for each combination of 'Complete' score
and 'NewDAS' status
das_frequency <- data %>%
  group_by(Complete, NewDAS) %>%
  summarise(Frequency = n(), .groups = 'drop') %>%
  mutate(Percentage = (Frequency / sum(Frequency)) * 100)

# Define specific colors for the new DAS values
colors <- c("Not present" = "gray", "Not shared" = "#E5696F", "Shared" = "#50
689B")

# Create the plot
das_plot <- ggplot(das_frequency, aes(x = as.factor(Complete), y = Frequency,
fill = NewDAS)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = paste0(round(Percentage, 1), "% (", Frequency, ")")),
            position = position_dodge(width = 0.9), vjust = -0.25, size = 3.5
) +
  scale_fill_manual(values = colors, name = "DAS Status") +
  theme_bw() +  # Use theme_bw for a white background
   theme(
    axis.title = element_text(size = 10), # Increase axis titles
    axis.text = element_text(size = 16),  # Increase axis text
    legend.title = element_text(size = 14), # Increase legend title
    legend.text = element_text(size = 16)  # Increase legend text
```

```
  ) +
  labs(x = "Completeness Score", y = "Frequency") +
  theme(legend.position = "bottom")

# Print and save the plot
print(das_plot)
```



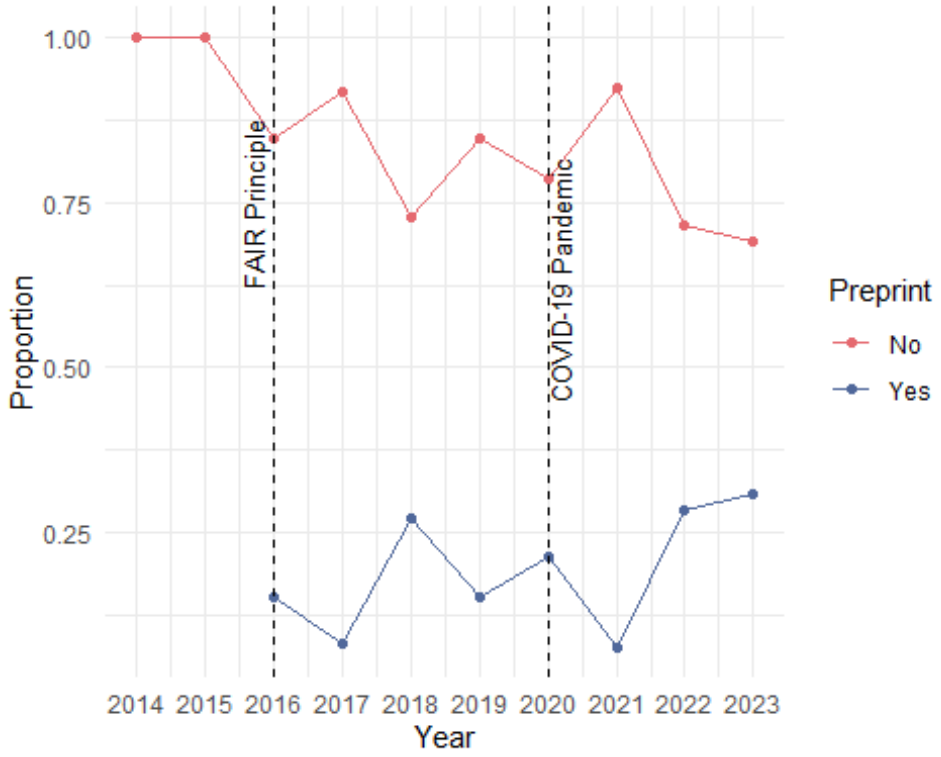

```
ggsave("new_das_distribution_plot.png", das_plot, width = 8, height = 6,  bg
= "white")

# Calculate the total count per year
yearly_totals <- data %>%
  group_by(Year) %>%
  summarise(Total = n(), .groups = 'drop')

# Join the totals back to the original data and calculate proportions
proportion_data <- data %>%
  left_join(yearly_totals, by = "Year") %>%
  group_by(Year, NewDAS) %>%
  summarise(Count = n(), .groups = 'drop') %>%
  mutate(Proportion = Count / n())

# Get all unique years for the x-axis
all_years <- sort(unique(proportion_data$Year))

colors <- c("Not present" = "gray", "Not shared" = "#E5696F", "Shared" = "#50
689B")
```

```r
# Plot with every year on the x-axis
DASyear <- ggplot(proportion_data, aes(x = Year, y = Proportion, color = NewDAS, group = NewDAS)) +
  geom_line() +
  geom_point() +
  geom_vline(xintercept = 2016, linetype = "dashed", color = "black") +
  geom_vline(xintercept = 2020, linetype = "dashed", color = "black") +
  annotate("text", x = 2016, y = max(proportion_data$Proportion), label = "FAIR Principle", angle = 90, hjust = 1.5, vjust = -0.5, size = 3.5) +
  annotate("text", x = 2020, y = max(proportion_data$Proportion), label = "COVID-19 Pandemic", angle = 90, hjust = 1.5, vjust = 1, size = 3.5) +
  scale_x_continuous(breaks = all_years) +
  labs(x = "Year", y = "Proportion", color = "Data Availability Statement") +
  theme(
    axis.title = element_text(size = 10), # Increase axis titles
    axis.text = element_text(size = 16),  # Increase axis text
    legend.title = element_text(size = 14), # Increase legend title
    legend.text = element_text(size = 16)  # Increase legend text
  ) +
  theme_minimal() +
  scale_color_manual(values = colors)

print(DASyear)
```




```r
ggsave("DASyear.png", DASyear, width = 8, height = 6,  bg = "white")
```

```r
 #Convert the Preprint numeric values to factor levels 'Yes' and 'No'
data$Preprint <- factor(ifelse(data$Preprint == 1, "Yes", "No"))

# Calculate the total count per year and proportion for Preprint
proportion_data <- data %>%
  group_by(Year) %>%
  count(Preprint) %>%
  mutate(Total = sum(n),
         Proportion = n / Total)

# Get all unique years for the x-axis
all_years <- sort(unique(proportion_data$Year))

# Define colors for 'Yes' and 'No'
colors <- c("No" = "#E5696F", "Yes" = "#50689B")

# Plot with every year on the x-axis for Preprint
PreprintYear <- ggplot(proportion_data, aes(x = Year, y = Proportion, color =
Preprint, group = Preprint)) +
  geom_line() +
  geom_point() +
  geom_vline(xintercept = 2016, linetype = "dashed", color = "black") +
  geom_vline(xintercept = 2020, linetype = "dashed", color = "black") +
  annotate("text", x = 2016, y = max(proportion_data$Proportion), label = "FA
IR Principle", angle = 90, hjust = 1.5, vjust = -0.5, size = 3.5) +
  annotate("text", x = 2020, y = max(proportion_data$Proportion), label = "CO
VID-19 Pandemic", angle = 90, hjust = 1.5, vjust = 1, size = 3.5) +
  scale_x_continuous(breaks = all_years) +
  labs(x = "Year", y = "Proportion", color = "Preprint") +
  theme(
    axis.title = element_text(size = 10), # Increase axis titles
    axis.text = element_text(size = 16),  # Increase axis text
    legend.title = element_text(size = 14), # Increase legend title
    legend.text = element_text(size = 16)  # Increase legend text
  ) +
  theme_minimal() +
  scale_color_manual(values = colors)

print(PreprintYear)
```




```
# Save the plot for Preprint
ggsave("PreprintYear.png", PreprintYear, width = 8, height = 6, bg = "white")
```

#study if FAIR implementation and Covid19 have an effect on DAS and Preprint # 1- Fair implementataion

```
total_by_period <- data %>%
  group_by(Period2016) %>%
  summarise(Total = n(), .groups = "drop")

# Join this total with the DAS and Preprint summaries and calculate the perce
ntage
Preprint_summary <- data %>%
  group_by(Period2016, Preprint) %>%
  summarise(Frequency = n(), .groups = "drop") %>%
  left_join(total_by_period, by = "Period2016") %>%
  mutate(Percentage = (Frequency / Total) * 100)

# Print the Preprint summary table
print(Preprint_summary)
```

```
## # A tibble: 4 × 5
##   Period2016  Preprint Frequency Total Percentage
##   <chr>       <fct>        <int> <int>      <dbl>
## 1 After 2016  No              72    90         80
## 2 After 2016  Yes             18    90         20
```

```
## 3 Before 2016 No              38    40         95
## 4 Before 2016 Yes              2    40          5
```

```
Preprint_table <- table(data$Preprint, data$Period2016)

# Perform the Chi-square test
result <- chisq.test(Preprint_table)

#Print the chi-square results
print(result)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  Preprint_table
## X-squared = 3.7034, df = 1, p-value = 0.0543
```

```
# Get the expected values
expected_values <- result$expected

# Print the expected values
print(expected_values) ##in order to check the assumption of chi square test: Each cell in the contingency table should have an expected count of 5 or more.
```

```
##
##       After 2016 Before 2016
##   No    76.15385   33.846154
##   Yes   13.84615    6.153846
```

```
#The assumptions of that test were met:

#### Sample Size: Each cell in the contingency table has an expected count of 5 or more.
#### Independence: The observations are  independent of each other. This means that the selection of one observation should not influence or affect the selection of another observation.
### Random Sampling: The data is a random sample.
### Categorical Data: Both variables should be categorical (either nominal or ordinal).

total_by_period <- data %>%
  group_by(Period2016) %>%
  summarise(Total = n(), .groups = "drop")

# Join this total with the NewDAS and Preprint summaries and calculate the percentage
das_summary <- data %>%
  group_by(Period2016, NewDAS)%>%
  summarise(Frequency = n(), .groups = "drop") %>%
  left_join(total_by_period, by = "Period2016") %>%
```

```
  mutate(Percentage = (Frequency / Total) * 100)

# Print the DAS summary table
print(das_summary)
```

```
## # A tibble: 5 × 5
##   Period2016  NewDAS      Frequency Total Percentage
##   <chr>       <fct>           <int> <int>      <dbl>
## 1 After 2016  Not present        40    90      44.4
## 2 After 2016  Not shared          8    90       8.89
## 3 After 2016  Shared             42    90      46.7
## 4 Before 2016 Not present        34    40      85
## 5 Before 2016 Shared              6    40      15
```

```
das_table <- table(data$NewDAS, data$Period2016)

# Perform the Chi-square test
result <- chisq.test(das_table)
```

```
## Warning in chisq.test(das_table): Chi-squared approximation may be incorre
ct
```

```
#Print the chi-square results
print(result)
```

```
##
##  Pearson's Chi-squared test
##
## data:  das_table
## X-squared = 19.078, df = 2, p-value = 7.199e-05
```

```
# Get the expected values
expected_values <- result$expected

# Print the expected values
print(expected_values) ##in order to check the assumption of chi square test:
Each cell in the contingency table should have an expected count of 5 or more
.
```

```
##
##               After 2016 Before 2016
##   Not present  51.230769   22.769231
##   Not shared    5.538462    2.461538
##   Shared       33.230769   14.769231
```

```
#The assumptions of that test were met:

#### Sample Size: Each cell in the contingency table has an expected count of
5 or more.
#### Independence: The observations are  independent of each other. This mean
s that the selection of one observation should not influence or affect the se
```

```
lection of another observation.
### Random Sampling: The data is a random sample.
### Categorical Data: Both variables should be categorical (either nominal or
ordinal).
```

#2- Covid 19 # the impact of COVID-19 on Preprint

```
total_by_period <- data %>%
  group_by(Period2020)%>%
  summarise(Total = n(), .groups = "drop")

# Join this total with the DAS and Preprint summaries and calculate the perce
ntage
Preprint_summary <- data %>%
  group_by(Period2020, Preprint) %>%
  summarise(Frequency = n(), .groups = "drop") %>%
  left_join(total_by_period, by = "Period2020") %>%
  mutate(Percentage = (Frequency / Total) * 100)

# Print the Preprint summary table
print(Preprint_summary)
```

```
## # A tibble: 4 × 5
##   Period2020  Preprint Frequency Total Percentage
##   <chr>       <fct>        <int> <int>      <dbl>
## 1 After 2020  No              31    40       77.5
## 2 After 2020  Yes              9    40       22.5
## 3 Before 2020 No              79    90       87.8
## 4 Before 2020 Yes             11    90       12.2
```

```
Preprint_table <- table(data$Preprint, data$Period2020)

# Perform the Chi-square test
result <- chisq.test(Preprint_table)

#Print the chi-square results
print(result)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  Preprint_table
## X-squared = 1.5269, df = 1, p-value = 0.2166
```

```
# Get the expected values
expected_values <- result$expected

# Print the expected values
print(expected_values) ##in order to check the assumption of chi square test:
Each cell in the contingency table should have an expected count of 5 or more
.
```

```
##
##       After 2020 Before 2020
##   No   33.846154    76.15385
##   Yes   6.153846    13.84615
```

```
#The assumptions of that test were met:

#### Sample Size: Each cell in the contingency table has an expected count of 5 or more.
#### Independence: The observations are  independent of each other. This means that the selection of one observation should not influence or affect the selection of another observation.
### Random Sampling: The data is a random sample.
### Categorical Data: Both variables should be categorical (either nominal or ordinal)

# the impact of COVID-19 on DAS

total_by_period <- data %>%
  group_by(Period2020) %>%
  summarise(Total = n(), .groups = "drop")

# Join this total with the NewDAS and Preprint summaries and calculate the percentage
das_summary <- data %>%
  group_by(Period2020, NewDAS)%>%
  summarise(Frequency = n(), .groups = "drop") %>%
  left_join(total_by_period, by = "Period2020") %>%
  mutate(Percentage = (Frequency / Total) * 100)

# Print the DAS summary table
print(das_summary)
```

```
## # A tibble: 6 × 5
##   Period2020  NewDAS      Frequency Total Percentage
##   <chr>       <fct>           <int> <int>      <dbl>
## 1 After 2020  Not present        13    40      32.5
## 2 After 2020  Not shared          1    40       2.5
## 3 After 2020  Shared             26    40      65
## 4 Before 2020 Not present        61    90      67.8
## 5 Before 2020 Not shared          7    90       7.78
## 6 Before 2020 Shared             22    90      24.4
```

```
das_table <- table(data$NewDAS, data$Period2020)

# Perform the Chi-square test
result <- chisq.test(das_table)
```

```
## Warning in chisq.test(das_table): Chi-squared approximation may be incorrect
```

```r
#Print the chi-square results
print(result)
```

```
##
##  Pearson's Chi-squared test
##
## data:  das_table
## X-squared = 19.644, df = 2, p-value = 5.426e-05
```

```r
# Get the expected values
expected_values <- result$expected

# Print the expected values
print(expected_values) ##in order to check the assumption of chi square test: Each cell in the contingency table should have an expected count of 5 or more.
```

```
##
##               After 2020 Before 2020
##   Not present 22.769231 51.230769
##   Not shared    2.461538 5.538462
##   Shared       14.769231 33.230769
```

```r
#The assumptions of that test were met:

#### Sample Size: Each cell in the contingency table has an expected count of 5 or more.
#### Independence: The observations are  independent of each other. This means that the selection of one observation should not influence or affect the selection of another observation.
### Random Sampling: The data is a random sample.
### Categorical Data: Both variables should be categorical (either nominal or ordinal)

Image_summary <- data %>%
  group_by(Image) %>%
  summarise(count = n(), .groups = "drop")

# Calculate the overall total
overall_total <- sum(Image_summary$count)

# Add a column for the percentage of each DAS category relative to the overal l total
Image_summary <- Image_summary %>%
  mutate(percentage = (count / overall_total) * 100)

print(Image_summary)
```

```
## # A tibble: 3 × 3
##   Image count percentage
##   <int> <int>      <dbl>
```

```
## 1     0    57      43.8
## 2     1    11       8.46
## 3    NA    62      47.7
```

```
data$DAS <- as.factor(data$DAS)

# Frequency and Percentage of the papers with DAS in the publications
data %>%
  group_by(Type, Image) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(Type) %>%
  mutate(total = sum(count),
         percentage = (count/total)*100) %>%
  ungroup() %>%
  arrange(Type, Image)
```

```
## # A tibble: 6 × 5
##   Type  Image count total percentage
##   <chr> <int> <int> <int>      <dbl>
## 1 IQB3      0    32    80       40
## 2 IQB3      1     2    80        2.5
## 3 IQB3     NA    46    80       57.5
## 4 Other     0    25    50       50
## 5 Other     1     9    50       18
## 6 Other    NA    16    50       32
```

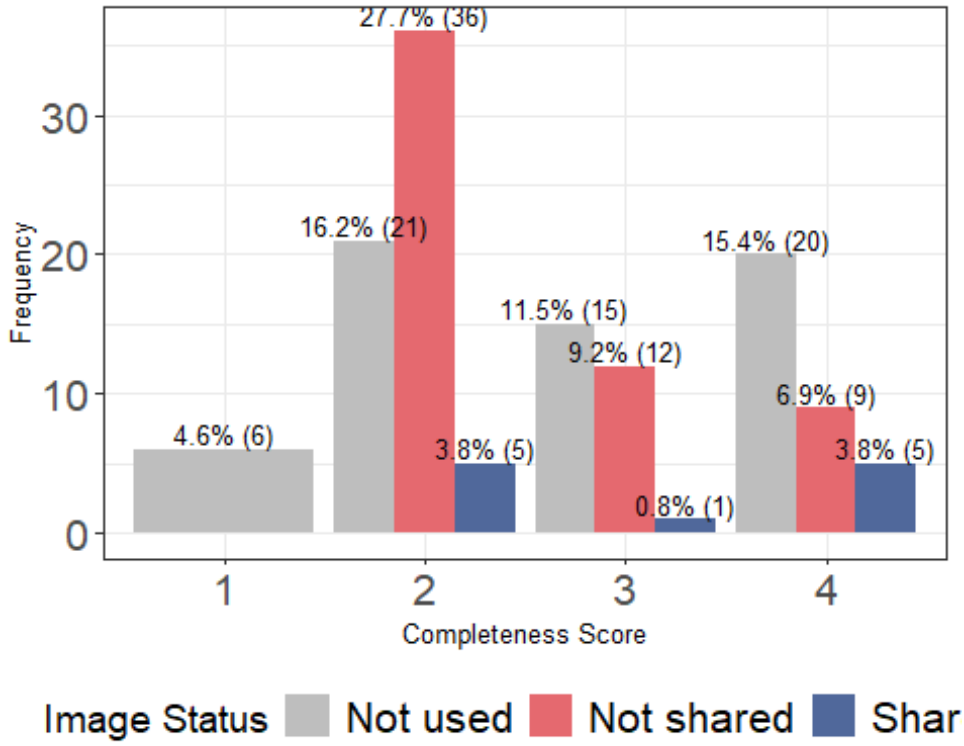
#plot the distribution of Image for each completeness score

```
# Transform the Image variable
data <- data %>%
  mutate(NewImage = case_when(
    is.na(Image) ~ "Not used",
    Image == 1 ~ "Shared",
    Image == 0 ~ "Not shared",
    TRUE ~ as.character(Image)))  # This line is just a fallback to handle un
expected values

class(data$NewImage)
```

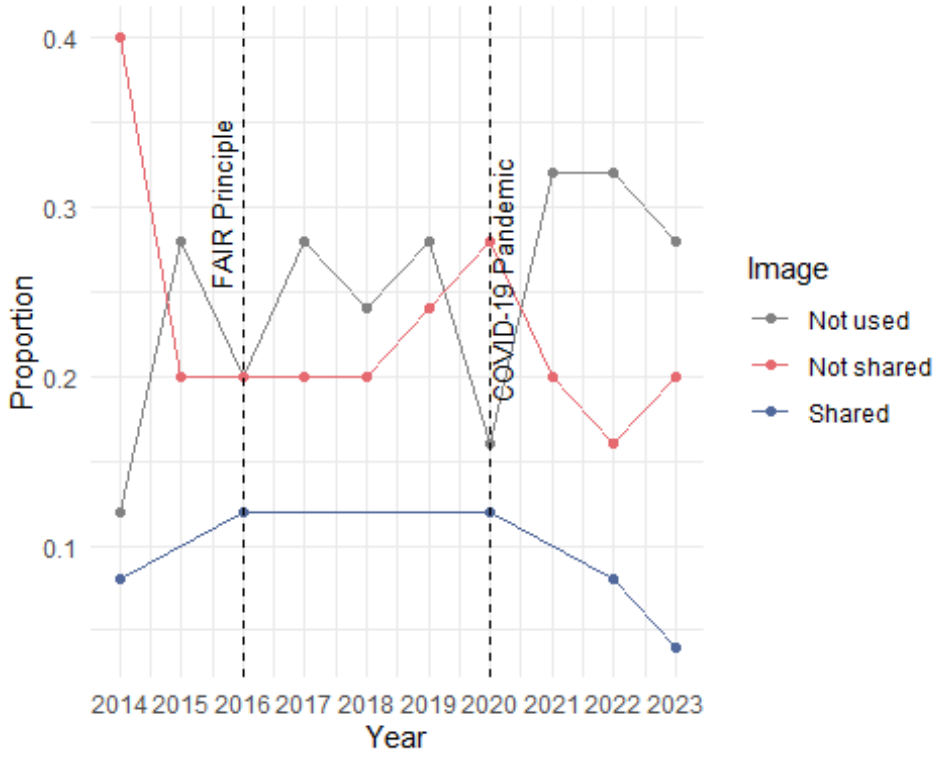
```
## [1] "character"
```

```
# Convert the new DAS variable to a factor for plotting
data$NewImage <- factor(data$NewImage, levels = c("Not used", "Not shared", "
Shared"))

# Calculate frequency and percentage for each combination of 'Complete' score
and 'NewImage' status
Image_frequency <- data %>%
  group_by(Complete, NewImage) %>%
  summarise(Frequency = n(), .groups = 'drop') %>%
  mutate(Percentage = (Frequency / sum(Frequency)) * 100)
```

```r
# Define specific colors for the new Image values
colors <- c("Not used" = "gray", "Not shared" = "#E5696F", "Shared" = "#50689
B")

# Create the plot
Image_plot <- ggplot(Image_frequency, aes(x = as.factor(Complete), y = Freque
ncy, fill = NewImage)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = paste0(round(Percentage, 1), "% (", Frequency, ")")),
            position = position_dodge(width = 0.9), vjust = -0.25, size = 3.5
) +
  scale_fill_manual(values = colors, name = "Image Status") +
  theme_bw() +  # Use theme_bw for a white background
   theme(
    axis.title = element_text(size = 10), # Increase axis titles
    axis.text = element_text(size = 16),  # Increase axis text
    legend.title = element_text(size = 14), # Increase legend title
    legend.text = element_text(size = 16)  # Increase legend text
  ) +
  labs(x = "Completeness Score", y = "Frequency") +
  theme(legend.position = "bottom")

# Print and save the plot
print(Image_plot)
```



```r
ggsave("new_Image_distribution_plot.png", Image_plot, width = 8, height = 6,
bg = "white")

# Calculate the total count per year
yearly_totals <- data %>%
  group_by(Year) %>%
  summarise(Total = n(), .groups = 'drop')

# Join the totals back to the original data and calculate proportions
proportion_data <- data %>%
  left_join(yearly_totals, by = "Year") %>%
  group_by(Year, NewImage) %>%
  summarise(Count = n(), .groups = 'drop') %>%
  mutate(Proportion = Count / n())

# Get all unique years for the x-axis
all_years <- sort(unique(proportion_data$Year))

colors <- c("Not used" = "#828282", "Not shared" = "#E5696F", "Shared" = "#50
689B")

# Plot with every year on the x-axis
Imageyear <- ggplot(proportion_data, aes(x = Year, y = Proportion, color = Ne
wImage, group = NewImage)) +
  geom_line() +
  geom_point() +
  geom_vline(xintercept = 2016, linetype = "dashed", color = "black") +
  geom_vline(xintercept = 2020, linetype = "dashed", color = "black") +
  annotate("text", x = 2016, y = max(proportion_data$Proportion), label = "FA
IR Principle", angle = 90, hjust = 1.5, vjust = -0.5, size = 3.5) +
  annotate("text", x = 2020, y = max(proportion_data$Proportion), label = "CO
VID-19 Pandemic", angle = 90, hjust = 1.5, vjust = 1, size = 3.5) +
  scale_x_continuous(breaks = all_years) +
  labs(x = "Year", y = "Proportion", color = "Image") +
  theme(
    axis.title = element_text(size = 10), # Increase axis titles
    axis.text = element_text(size = 16),  # Increase axis text
    legend.title = element_text(size = 14), # Increase legend title
    legend.text = element_text(size = 16)  # Increase legend text
  ) +
  theme_minimal() +
  scale_color_manual(values = colors)

print(Imageyear)
```




```
ggsave("Imageyear.png", Imageyear, width = 8, height = 6,  bg = "white")
```

#Chi square test to assess if there is any difference in Image between the two types (IQB3, Other)

```
# Create a contingency table
table_Image <- table(data$Type, data$Image)

# Perform the Chi-square test
result <- chisq.test(table_Image)

#Print the chi-square results
print(result)

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table_Image
## X-squared = 3.9043, df = 1, p-value = 0.04816

# Get the expected values
expected_values <- result$expected

# Print the expected values
print(expected_values) ##in order to check the assumption of chi square test: Each cell in the contingency table should have an expected count of 5 or more
.
```

```
##
##            0   1
##   IQB3  28.5 5.5
##   Other 28.5 5.5
```

```
#The assumptions of that test were met:

#### Sample Size: Each cell in the contingency table has an expected count of
5 or more.
#### Independence: The observations are  independent of each other. This mean
s that the selection of one observation should not influence or affect the se
lection of another observation.
### Random Sampling: The data is a random sample.
### Categorical Data: Both variables should be categorical (either nominal or
ordinal).
```

```
Genomics_summary <- data %>%
  group_by(Genomics) %>%
  summarise(count = n(), .groups = "drop")

# Calculate the overall total
overall_total <- sum(Genomics_summary$count)

# Add a column for the percentage of each Genomics category relative to the o
verall total
Genomics_summary <- Genomics_summary %>%
  mutate(percentage = (count / overall_total) * 100)

print(Genomics_summary)
```

```
## # A tibble: 3 × 3
##   Genomics count percentage
##      <int> <int>      <dbl>
## 1        0    27       20.8
## 2        1    27       20.8
## 3       NA    76       58.5
```

```
data$Genomics <- as.factor(data$Genomics)

# Frequency and Percentage of the papers with Genomics in the publications
data %>%
  group_by(Type, Genomics) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(Type) %>%
  mutate(total = sum(count),
         percentage = (count/total)*100) %>%
  ungroup() %>%
  arrange(Type, Genomics)
```

```
## # A tibble: 6 × 5
##   Type  Genomics count total percentage
##   <chr> <fct>    <int> <int>      <dbl>
## 1 IQB3  0           17    80       21.2
## 2 IQB3  1           15    80       18.8
## 3 IQB3  <NA>        48    80       60
## 4 Other 0           10    50       20
## 5 Other 1           12    50       24
## 6 Other <NA>        28    50       56
```

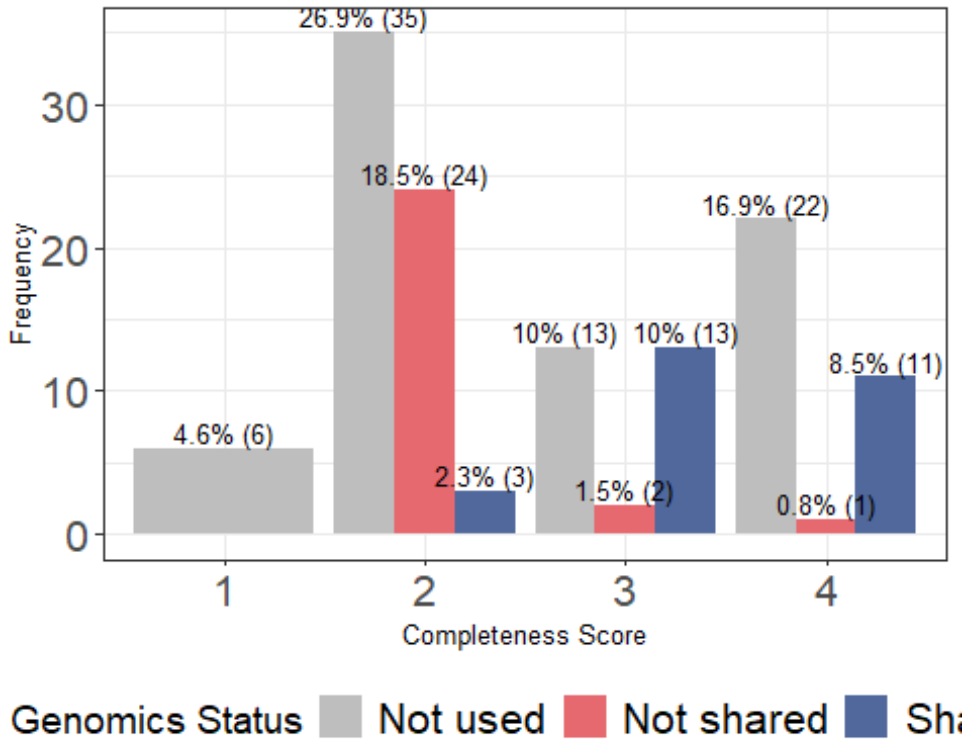
#plot the distribution of Genomics for each completeness score

```
# Transform the Genomics variable
data <- data %>%
  mutate(NewGenomics = case_when(
    is.na(Genomics) ~ "Not used",
    Genomics == 1 ~ "Shared",
    Genomics == 0 ~ "Not shared",
    TRUE ~ as.character(Genomics)))  # This line is just a fallback to handle
unexpected values

class(data$NewGenomics)
```

```
## [1] "character"
```

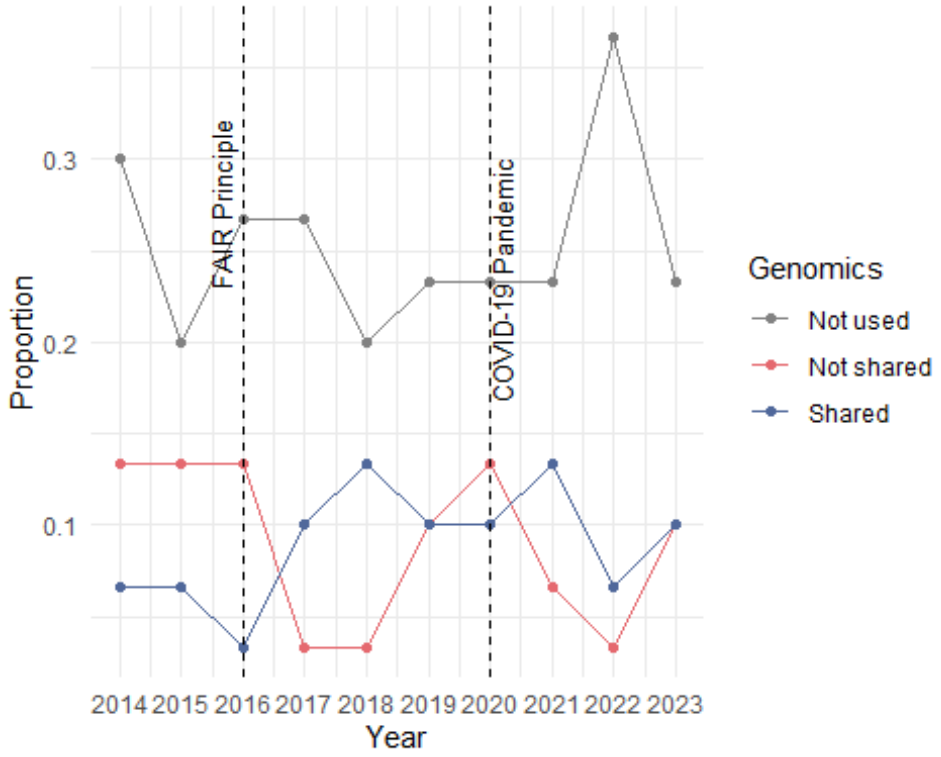
```
# Convert the new Genomic variable to a factor for plotting
data$NewGenomics <- factor(data$NewGenomics, levels = c("Not used", "Not shar
ed", "Shared"))

# Calculate frequency and percentage for each combination of 'Complete' score
and 'NewGenomics' status
Genomics_frequency <- data %>%
  group_by(Complete, NewGenomics) %>%
  summarise(Frequency = n(), .groups = 'drop') %>%
  mutate(Percentage = (Frequency / sum(Frequency)) * 100)

# Define specific colors for the new Genomics values
colors <- c("Not used" = "gray", "Not shared" = "#E5696F", "Shared" = "#50689
B")

# Create the plot
Genomics_plot <- ggplot(Genomics_frequency, aes(x = as.factor(Complete), y =
Frequency, fill = NewGenomics)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = paste0(round(Percentage, 1), "% (", Frequency, ")")),
            position = position_dodge(width = 0.9), vjust = -0.25, size = 3.5
) +
  scale_fill_manual(values = colors, name = "Genomics Status") +
  theme_bw() +  # Use theme_bw for a white background
   theme(
```

```
    axis.title = element_text(size = 10), # Increase axis titles
    axis.text = element_text(size = 16),  # Increase axis text
    legend.title = element_text(size = 14), # Increase legend title
    legend.text = element_text(size = 16)  # Increase legend text
  ) +
  labs(x = "Completeness Score", y = "Frequency") +
  theme(legend.position = "bottom")

# Print and save the plot
print(Genomics_plot)
```




```
ggsave("new_Genomics_distribution_plot.png", Image_plot, width = 8, height =
6,  bg = "white")

# Calculate the total count per year
yearly_totals <- data %>%
  group_by(Year) %>%
  summarise(Total = n(), .groups = 'drop')

# Join the totals back to the original data and calculate proportions
proportion_data <- data %>%
  left_join(yearly_totals, by = "Year") %>%
  group_by(Year, NewGenomics) %>%
  summarise(Count = n(), .groups = 'drop') %>%
  mutate(Proportion = Count / n())

# Get all unique years for the x-axis
```

```r
all_years <- sort(unique(proportion_data$Year))

colors <- c("Not used" = "#828282", "Not shared" = "#E5696F", "Shared" = "#50
689B")

# Plot with every year on the x-axis
Genomicsyear <- ggplot(proportion_data, aes(x = Year, y = Proportion, color =
NewGenomics, group = NewGenomics)) +
  geom_line() +
  geom_point() +
  geom_vline(xintercept = 2016, linetype = "dashed", color = "black") +
  geom_vline(xintercept = 2020, linetype = "dashed", color = "black") +
  annotate("text", x = 2016, y = max(proportion_data$Proportion), label = "FA
IR Principle", angle = 90, hjust = 1.5, vjust = -0.5, size = 3.5) +
  annotate("text", x = 2020, y = max(proportion_data$Proportion), label = "CO
VID-19 Pandemic", angle = 90, hjust = 1.5, vjust = 1, size = 3.5) +
  scale_x_continuous(breaks = all_years) +
  labs(x = "Year", y = "Proportion", color = "Genomics") +
  theme(
    axis.title = element_text(size = 10), # Increase axis titles
    axis.text = element_text(size = 16),  # Increase axis text
    legend.title = element_text(size = 14), # Increase legend title
    legend.text = element_text(size = 16)  # Increase legend text
  ) +
  theme_minimal() +
  scale_color_manual(values = colors)

print(Genomicsyear)
```




```
ggsave("Genomicsyear.png", Genomicsyear, width = 8, height = 6,  bg = "white"
)

# Create a contingency table
table_Genomics <- table(data$Type, data$Genomics)

# Perform the Chi-square test
result <- chisq.test(table_Genomics)

#Print the chi-square results
print(result)

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table_Genomics
## X-squared = 0.076705, df = 1, p-value = 0.7818

# Get the expected values
expected_values <- result$expected

# Print the expected values
print(expected_values) ##in order to check the assumption of chi square test:
Each cell in the contingency table should have an expected count of 5 or more
.
```

```
##
##          0  1
##   IQB3  16 16
##   Other 11 11
```

```
#The assumptions of that test were met:

#### Sample Size: Each cell in the contingency table has an expected count of 5 or more.
#### Independence: The observations are  independent of each other. This means that the selection of one observation should not influence or affect the selection of another observation.
### Random Sampling: The data is a random sample.
### Categorical Data: Both variables should be categorical (either nominal or ordinal).
```

```
Proteometabolomic_summary <- data %>%
  group_by(Proteometabolomic) %>%
  summarise(count = n(), .groups = "drop")

# Calculate the overall total
overall_total <- sum(Proteometabolomic_summary$count)

# Add a column for the percentage of each Proteometabolomic category relative to the overall total
Proteometabolomic_summary <- Proteometabolomic_summary %>%
  mutate(percentage = (count / overall_total) * 100)

print(Proteometabolomic_summary)
```

```
## # A tibble: 3 × 3
##   Proteometabolomic count percentage
##               <int> <int>      <dbl>
## 1                 0    34      26.2
## 2                 1    10       7.69
## 3                NA    86      66.2
```

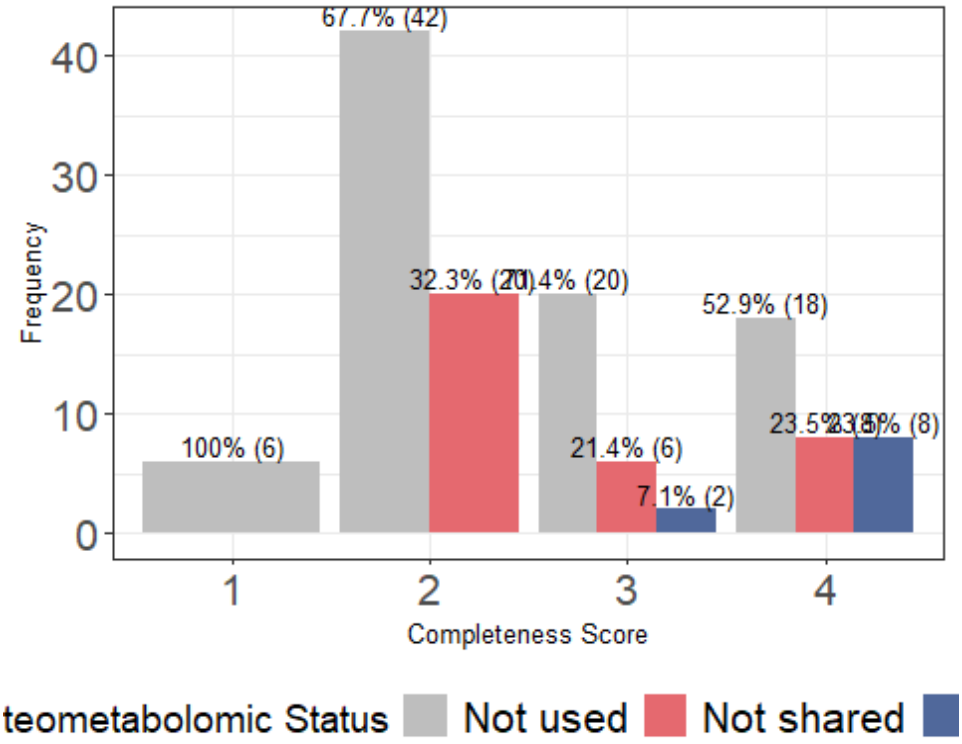
```
data$Proteometabolomic <- as.factor(data$Proteometabolomic)

# Frequency and Percentage of the papers with Proteometabolomic in the publications
data %>%
  group_by(Type, Proteometabolomic) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(Type) %>%
  mutate(total = sum(count),
         percentage = (count/total)*100) %>%
  ungroup() %>%
  arrange(Type, Proteometabolomic)
```

```
## # A tibble: 6 × 5
##   Type  Proteometabolomic count total percentage
##   <chr> <fct>             <int> <int>      <dbl>
## 1 IQB3  0                    25    80       31.2
## 2 IQB3  1                     8    80       10
## 3 IQB3  <NA>                 47    80       58.8
## 4 Other 0                     9    50       18
## 5 Other 1                     2    50        4
## 6 Other <NA>                 39    50       78
```

#plot the distribution of Proteometabolomic for each completeness score

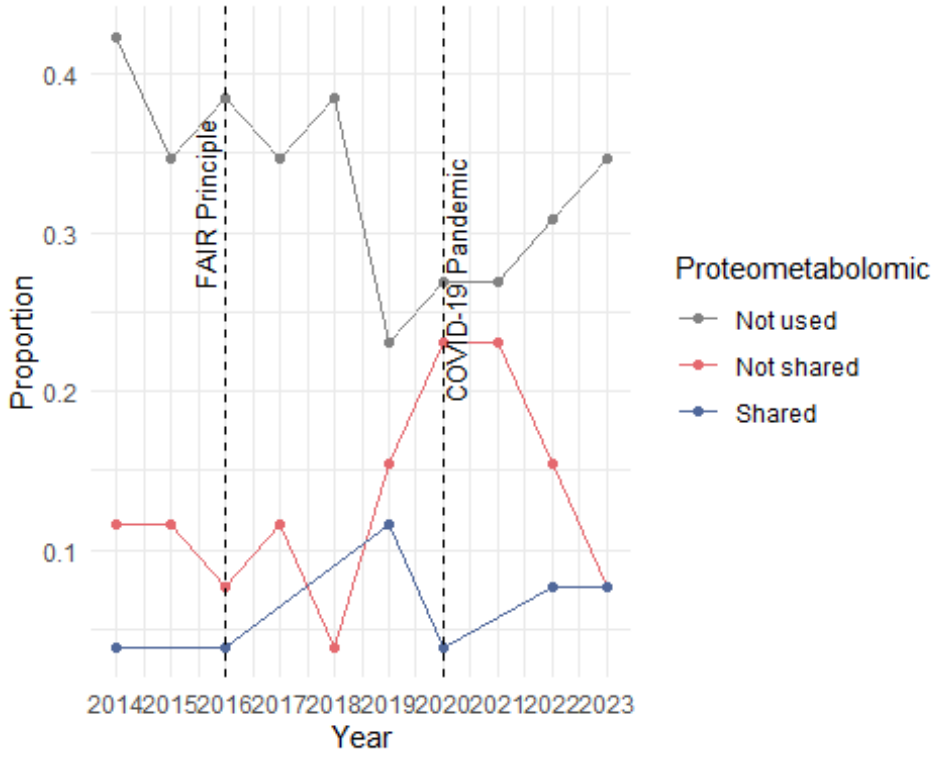
```
library(dplyr)
library(ggplot2)

# Transform the Proteometabolomic variable
data <- data %>%
  mutate(NewProteometabolomic = case_when(
    is.na(Proteometabolomic) ~ "Not used",
    Proteometabolomic == 1 ~ "Shared",
    Proteometabolomic == 0 ~ "Not shared",
    TRUE ~ as.character(Proteometabolomic)))

# Convert the new New Proteometabolomic variable to a factor for plotting
data$NewProteometabolomic <- factor(data$NewProteometabolomic, levels = c("No
t used", "Not shared", "Shared"))

# Calculate frequency and percentage for each combination of 'Complete' score
and 'NewProteometabolomic' status
Proteometabolomic_frequency <- data %>%
  count(Complete, NewProteometabolomic) %>%
  group_by(Complete) %>%
  mutate(Percentage = (n / sum(n)) * 100) %>%
  ungroup()

# Define specific colors for the Proteometabolomic values
colors <- c("Not used" = "gray", "Not shared" = "#E5696F", "Shared" = "#50689
B")

# Create the plot
Proteometabolomic_plot <- ggplot(Proteometabolomic_frequency, aes(x = as.fact
or(Complete), y = n, fill = NewProteometabolomic)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = paste0(round(Percentage, 1), "% (", n, ")")),
            position = position_dodge(width = 0.9), vjust = -0.25, size = 3.5
) +
  scale_fill_manual(values = colors, name = "Proteometabolomic Status") +
  theme_bw() +
  theme(
    axis.title = element_text(size = 10),
```

```
    axis.text = element_text(size = 16),
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 16)
  ) +
  labs(x = "Completeness Score", y = "Frequency") +
  theme(legend.position = "bottom")

# Print and save the plot
print(Proteometabolomic_plot)
```



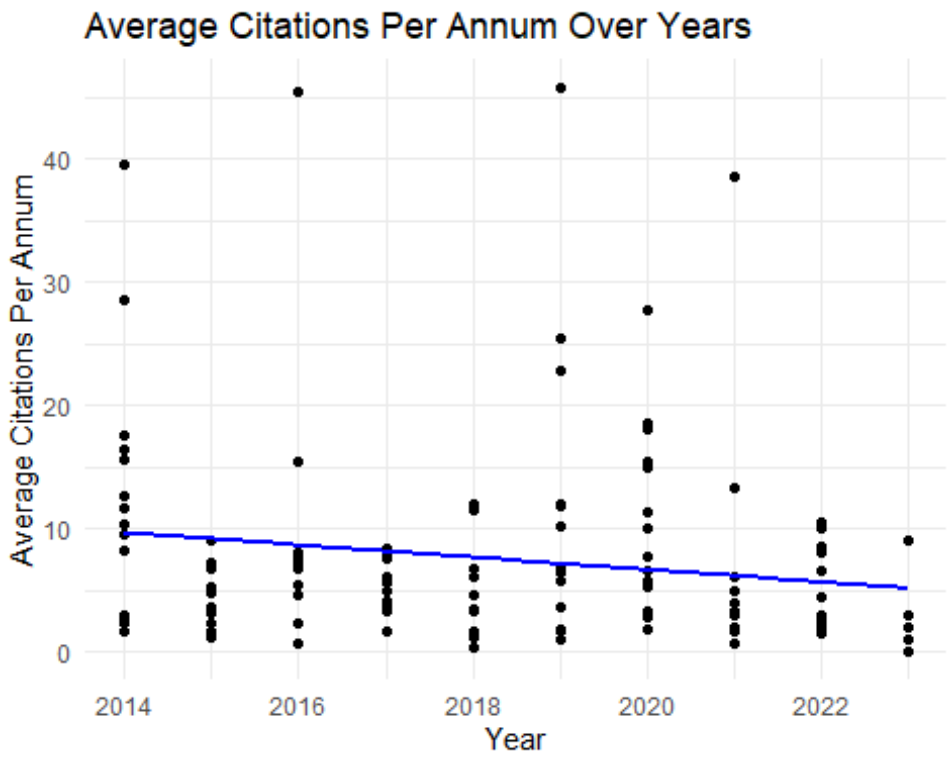

```
ggsave("new_Proteometabolomic_distribution_plot.png", Proteometabolomic_plot,
width = 8, height = 6, bg = "white")

# Calculate the total count per year
yearly_totals <- data %>%
  group_by(Year) %>%
  summarise(Total = n(), .groups = 'drop')

# Join the totals back to the original data and calculate proportions
proportion_data <- data %>%
  left_join(yearly_totals, by = "Year") %>%
  group_by(Year, NewProteometabolomic) %>%
  summarise(Count = n(), .groups = 'drop') %>%
  mutate(Proportion = Count / n())

# Get all unique years for the x-axis
all_years <- sort(unique(proportion_data$Year))
```

```r
colors <- c("Not used" = "#828282", "Not shared" = "#E5696F", "Shared" = "#50
689B")

# Plot with every year on the x-axis
Proteometabolomicyear <- ggplot(proportion_data, aes(x = Year, y = Proportion
, color = NewProteometabolomic, group = NewProteometabolomic)) +
  geom_line() +
  geom_point() +
  geom_vline(xintercept = 2016, linetype = "dashed", color = "black") +
  geom_vline(xintercept = 2020, linetype = "dashed", color = "black") +
  annotate("text", x = 2016, y = max(proportion_data$Proportion), label = "FA
IR Principle", angle = 90, hjust = 1.5, vjust = -0.5, size = 3.5) +
  annotate("text", x = 2020, y = max(proportion_data$Proportion), label = "CO
VID-19 Pandemic", angle = 90, hjust = 1.5, vjust = 1, size = 3.5) +
  scale_x_continuous(breaks = all_years) +
  labs(x = "Year", y = "Proportion", color = "Proteometabolomic") +
  theme(
    axis.title = element_text(size = 10), # Increase axis titles
    axis.text = element_text(size = 16),  # Increase axis text
    legend.title = element_text(size = 14), # Increase legend title
    legend.text = element_text(size = 16)  # Increase legend text
  ) +
  theme_minimal() +
  scale_color_manual(values = colors)

print(Proteometabolomicyear)
```




```
ggsave("Proteometabolomicyear.png", Proteometabolomicyear, width = 8, height
= 6,  bg = "white")

# Create a contingency table
table_Proteometabolomic <- table(data$Type, data$Proteometabolomic)

# Perform the Chi-square test
result <- chisq.test(table_Proteometabolomic)

## Warning in chisq.test(table_Proteometabolomic): Chi-squared approximation
may
## be incorrect

#Print the chi-square results
print(result)

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table_Proteometabolomic
## X-squared = 0, df = 1, p-value = 1

# Get the expected values
expected_values <- result$expected

# Print the expected values
print(expected_values) ##in order to check the assumption of chi square test:
```

```
Each cell in the contingency table should have an expected count of 5 or more
.

##
##            0   1
##   IQB3  25.5 7.5
##   Other  8.5 2.5

#The assumptions of that test were met:

#### Sample Size: Each cell in the contingency table has an expected count of
5 or more.
#### Independence: The observations are  independent of each other. This mean
s that the selection of one observation should not influence or affect the se
lection of another observation.
### Random Sampling: The data is a random sample.
### Categorical Data: Both variables should be categorical (either nominal or
ordinal).

# Load the data
data <- read.csv("datarecord.csv")

# Plot using ggplot2
ggplot(data, aes(x = Year, y = CitationsPA)) +
  geom_point() + # Add points
  geom_smooth(method = "lm", se = FALSE, color = "blue") + # Add a linear tre
nd line without confidence interval
  labs(title = "Average Citations Per Annum Over Years",
       x = "Year",
       y = "Average Citations Per Annum") +
  theme_minimal()

## `geom_smooth()` using formula = 'y ~ x'
```



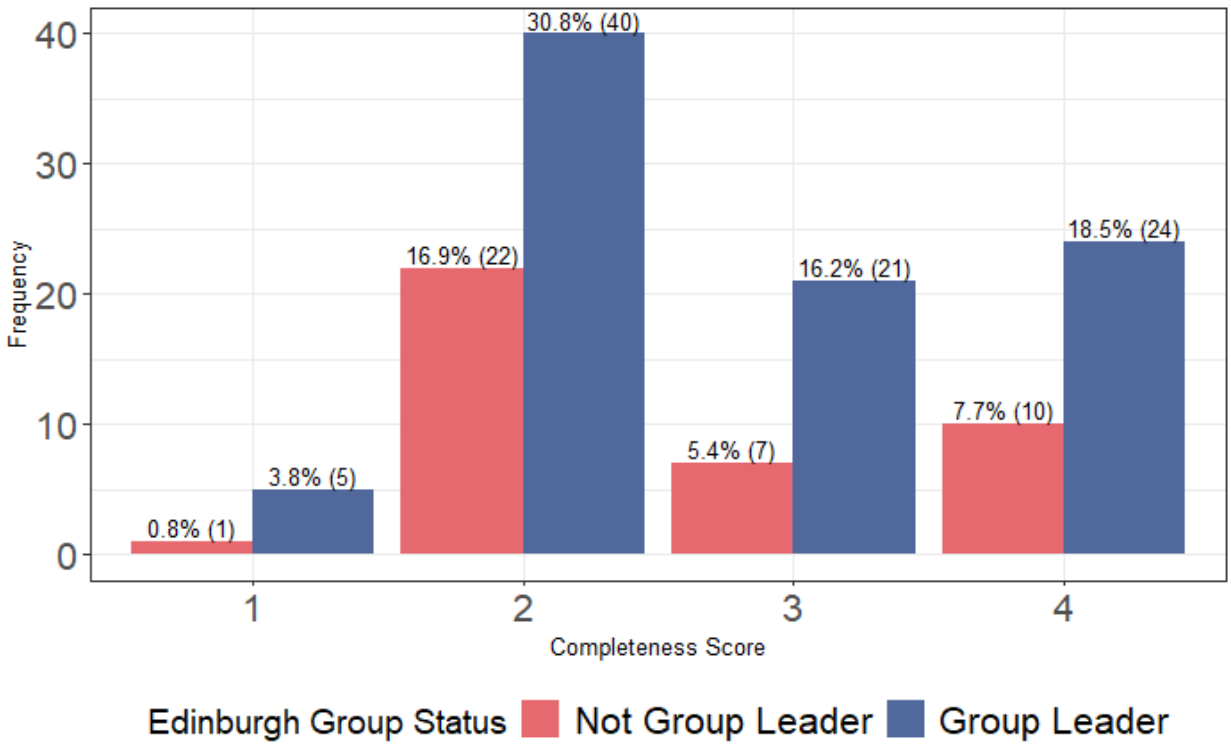

```
## Call: lm(formula = CitationsPA ~ Year, data = data)

Residuals:   Min     1Q Median     3Q

Max -8.045 -4.353 -2.144  0.806 38.656

Coefficients:

      Estimate Std.       Error     t value    Pr(>|t|)

(Intercept) 1016.9684   500.4047   2.032   0.0442 *

Year                    -0.5002     0.2479    -2.018   0.0457 *

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.269 on 128 degrees of freedom

Multiple R-squared:  0.03082, Adjusted R-squared:  0.02325

F-statistic:  4.07 on 1 and 128 DF,  p-value: 0.04573

#add the difference between high and low completeness score

df <- read.csv("datarecord.csv")

# Ensure the 'Complete' and 'CitationsPA' columns are correctly formatted
```

```r
# 'Complete' should be numeric or integer, and 'CitationsPA' should be numeri
c.
# If not, you might need to convert them, for example:
# df$Complete <- as.numeric(df$Complete)
# df$CitationsPA <- as.numeric(df$CitationsPA)

# Step 1: Split the data into two groups based on 'Complete' scores
group1 <- df$CitationsPA[df$Complete %in% c(1,2)]
group2 <- df$CitationsPA[df$Complete %in% c(3,4)]

# Ensure that there are no NA values that could affect the t-test
# This step is optional but recommended
group1 <- na.omit(group1)
group2 <- na.omit(group2)

# Step 2: Conduct a T-test to compare the mean citation counts between the tw
o groups
t_test_result <- t.test(group1, group2)

# Display the t-test results
print(t_test_result)


##  Welch Two Sample t-test
##
## data:  group1 and group2
## t = -0.22511, df = 127.94, p-value = 0.8223
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.232464  2.572094
## sample estimates:
## mean of x mean of y
##  7.229044  7.559229

# Additional: Display mean citations for insight
mean_group1 <- mean(group1)
mean_group2 <- mean(group2)

cat("Mean citation counts for Completeness 1-2:", mean_group1, "\n")

## Mean citation counts for Completeness 1-2: 7.229044

cat("Mean citation counts for Completeness 3-4:", mean_group2, "\n")

## Mean citation counts for Completeness 3-4: 7.559229

# Assuming 'data' is your DataFrame and has been loaded correctly
# Transform the Edinburgh Group Leader variable
data <- data %>%
  mutate(EdinburghGroupLeaderStatus = case_when(
    EdinburghGroupLeader == 1 ~ "Group Leader",
```

```
    EdinburghGroupLeader == 0 ~ "Not Group Leader",
    TRUE ~ as.character(EdinburghGroupLeader)))  # Fallback for unexpected va
lues

# Check the class of the new variable
class(data$EdinburghGroupLeaderStatus)
```

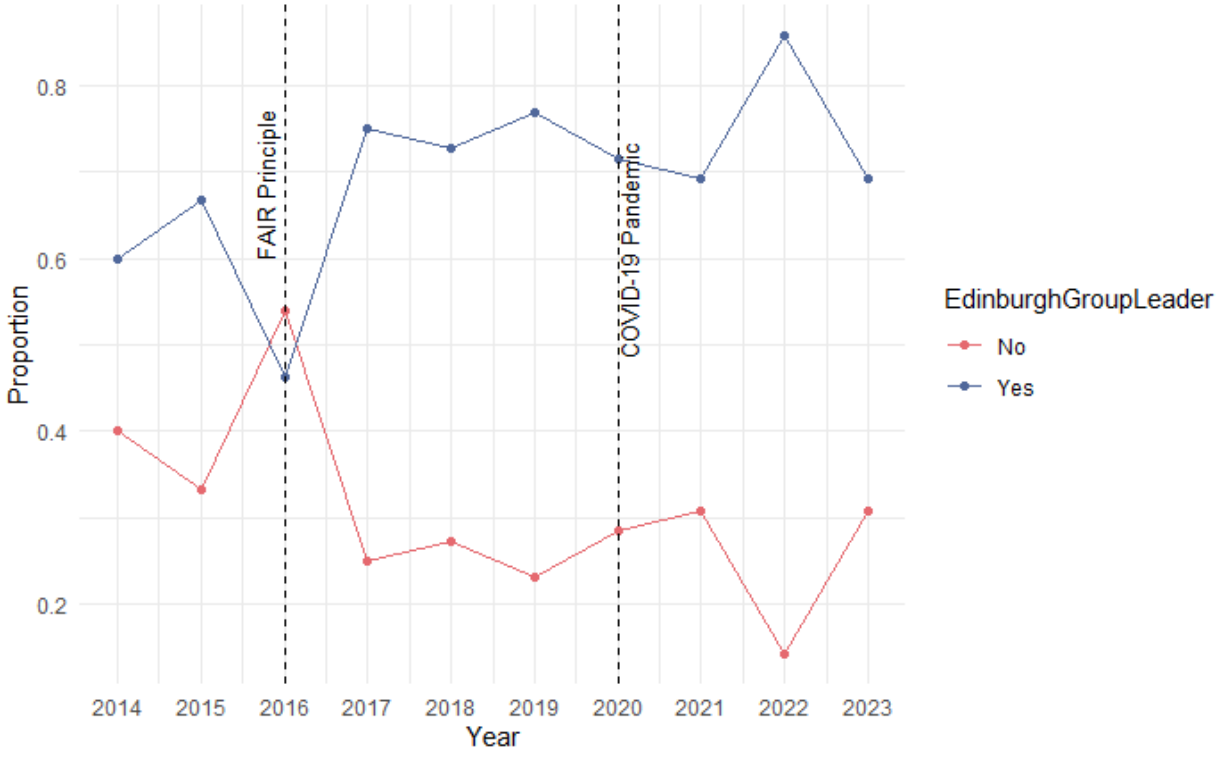
```
## [1] "character"
```

```
# Convert the new Edinburgh Group Leader variable to a factor for plotting
data <- data %>%
  mutate(EdinburghGroupLeaderStatus = factor(EdinburghGroupLeaderStatus, leve
ls = c("Not Group Leader", "Group Leader")))

# Calculate frequency and percentage for each 'Complete' score and 'Edinburgh
GroupLeaderStatus'
Edinburgh_group_frequency <- data %>%
  group_by(Complete, EdinburghGroupLeaderStatus) %>%
  summarise(Frequency = n(), .groups = 'drop') %>%
  mutate(Percentage = (Frequency / sum(Frequency)) * 100)

# Define specific colors for the PI Group Leader status
colors <- c("Not Group Leader" = "#E5696F", " Group Leader" = "#50689B")

# Create the plot
Edinburgh_group_plot <- ggplot(Edinburgh_group_frequency, aes(x = as.factor(C
omplete), y = Frequency, fill = EdinburghGroupLeaderStatus)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = paste0(round(Percentage, 1), "% (", Frequency, ")")),
            position = position_dodge(width = 0.9), vjust = -0.25, size = 3.5
) +
  scale_fill_manual(values = colors, name = "PI Group Status") +
  theme_bw() +
  theme(
    axis.title = element_text(size = 10),
    axis.text = element_text(size = 16),
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 16)
  ) +
  labs(x = "Completeness Score", y = "Frequency") +
  theme(legend.position = "bottom")

# Print the plot
print(Edinburgh_group_plot)
```




```r
# Save the plot to a file
ggsave("Edinburgh_group_status_distribution_plot.png", Edinburgh_group_plot,
width = 8, height = 6, bg = "white")

 #Convert the EdinburghGroupLeader numeric values to factor levels 'Yes' and
'No'
data$EdinburghGroupLeader <- factor(ifelse(data$EdinburghGroupLeader == 1, "Y
es", "No"))

# Calculate the total count per year and proportion for EdinburghGroupLeader
proportion_data <- data %>%
  group_by(Year) %>%
  count(EdinburghGroupLeader) %>%
  mutate(Total = sum(n),
         Proportion = n / Total)

# Get all unique years for the x-axis
all_years <- sort(unique(proportion_data$Year))

# Define colors for 'Yes' and 'No'
colors <- c("No" = "#E5696F", "Yes" = "#50689B")

# Plot with every year on the x-axis for EdinburghGroupLeader
EdinburghGroupLeaderYear <- ggplot(proportion_data, aes(x = Year, y = Proport
ion, color = EdinburghGroupLeader, group = EdinburghGroupLeader)) +
```

```
  geom_line() +
  geom_point() +
  geom_vline(xintercept = 2016, linetype = "dashed", color = "black") +
  geom_vline(xintercept = 2020, linetype = "dashed", color = "black") +
  annotate("text", x = 2016, y = max(proportion_data$Proportion), label = "FA
IR Principle", angle = 90, hjust = 1.5, vjust = -0.5, size = 3.5) +
  annotate("text", x = 2020, y = max(proportion_data$Proportion), label = "CO
VID-19 Pandemic", angle = 90, hjust = 1.5, vjust = 1, size = 3.5) +
  scale_x_continuous(breaks = all_years) +
  labs(x = "Year", y = "Proportion", color = "EdinburghGroupLeader") +
  theme(
    axis.title = element_text(size = 10), # Increase axis titles
    axis.text = element_text(size = 16),  # Increase axis text
    legend.title = element_text(size = 14), # Increase legend title
    legend.text = element_text(size = 16)  # Increase legend text
  ) +
  theme_minimal() +
  scale_color_manual(values = colors)

print(EdinburghGroupLeaderYear)
```




```
# Save the plot for EdinburghGroupLeader
ggsave("EdinburghGroupLeader.png", EdinburghGroupLeaderYear, width = 8, heigh
t = 6, bg = "white")

# Assuming 'data' is your DataFrame and has been loaded correctly
# Transform the completeness score into a binary variable (high vs. low)
```

```r
data <- data %>%
  mutate(CompletenessCategory = case_when(
    Complete %in% 3:4 ~ "High",
    Complete %in% 1:2 ~ "Low",
    TRUE ~ NA_character_  # Handle potential unexpected values
  ))

# Drop rows with NA in CompletenessCategory if any exist due to unexpected va
lues
data <- data %>% filter(!is.na(CompletenessCategory))

# Create a contingency table
contingency_table <- table(data$EdinburghGroupLeader, data$CompletenessCatego
ry)

# Conduct a Chi-square test of independence
chi_square_result <- chisq.test(contingency_table)

# Print the result of the Chi-square test
print(chi_square_result)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 0.35995, df = 1, p-value = 0.5485
```

```r
# Calculate frequency and percentage for each Funding Source  across all type
s
FundingSource_summary <- data %>%
  group_by(FundingSource) %>%
  summarise(count = n(), .groups = "drop")

# Calculate the overall total
overall_total <- sum(FundingSource_summary$count)

# Add a column for the percentage of each FundingSource category relative to
the overall total
FundingSource_summary <- FundingSource_summary %>%
  mutate(percentage = (count / overall_total) * 100)

# Print the summary
print(FundingSource_summary)
```

```
## # A tibble: 2 × 3
##   FundingSource count percentage
##           <int> <int>      <dbl>
## 1             0   124      95.4
## 2             1     6       4.62
```

```r
# Frequency and Percentage of the papers with FundingSource
data %>%
  group_by(Type, FundingSource) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(Type) %>%
  mutate(total = sum(count),
         percentage = (count/total)*100) %>%
  ungroup() %>%
  arrange(Type, FundingSource)
```

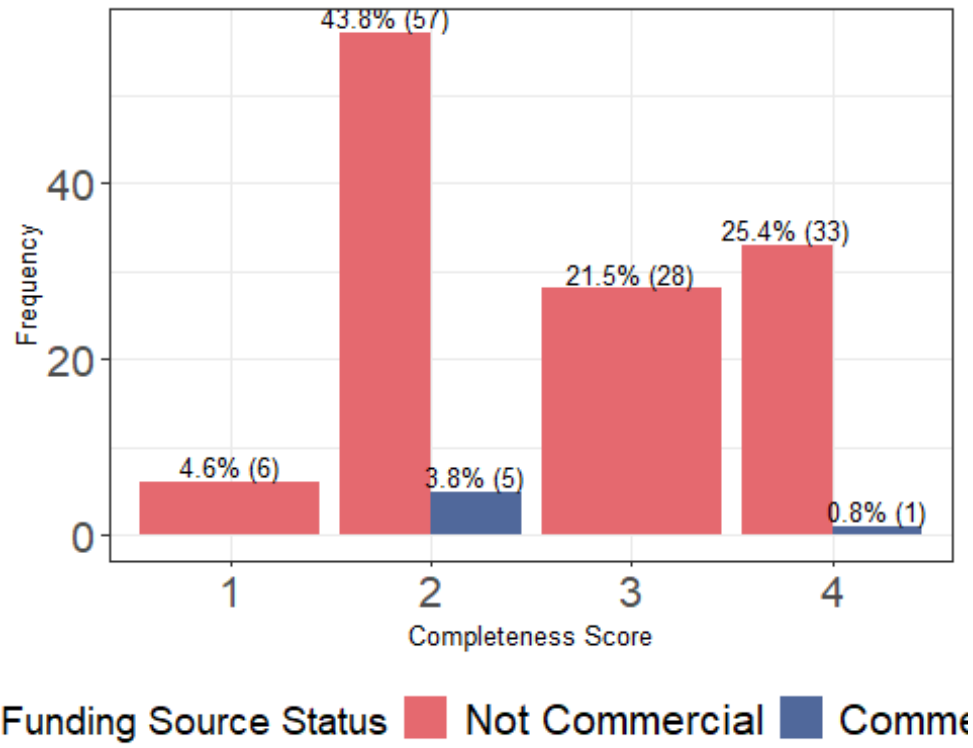
```
## # A tibble: 4 × 5
##   Type  FundingSource count total percentage
##   <chr>         <int> <int> <int>      <dbl>
## 1 IQB3              0    75    80      93.8
## 2 IQB3              1     5    80       6.25
## 3 Other             0    49    50      98
## 4 Other             1     1    50       2
```

```r
# Assuming 'data' is your DataFrame and has been loaded correctly
# Transform the Funding Source variable
data <- data %>%
  mutate(FundingSourceStatus = case_when(
    FundingSource == 1 ~ "Commercial",
    FundingSource == 0 ~ "Not Commercial",
    TRUE ~ as.character(FundingSource)  # Fallback for unexpected values
  ))

# Check the class of the new variable
class(data$FundingSourceStatus)
```

```
## [1] "character"
```

```r
# Convert the new Funding Source Status variable to a factor for plotting
data <- data %>%
  mutate(FundingSourceStatus = factor(FundingSourceStatus, levels = c("Not Co
mmercial", "Commercial")))

# Calculate frequency and percentage for each 'Complete' score and 'Funding S
ource Status'
FundingSource_frequency <- data %>%
  group_by(Complete, FundingSourceStatus) %>%
  summarise(Frequency = n(), .groups = 'drop') %>%
  mutate(Percentage = (Frequency / sum(Frequency)) * 100)

# Define specific colors for the Funding Source status
colors <- c("Not Commercial" = "#E5696F", "Commercial" = "#50689B")  # Correc
t the color names to match factor levels

# Create the plot
FundingSource_plot <- ggplot(FundingSource_frequency, aes(x = as.factor(Compl
```

```
ete), y = Frequency, fill = FundingSourceStatus)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = paste0(round(Percentage, 1), "% (", Frequency, ")")),
            position = position_dodge(width = 0.9), vjust = -0.25, size = 3.5
) +
  scale_fill_manual(values = colors, name = "Funding Source Status") +  # Ens
ure the name matches the legend
  theme_bw() +
  theme(
    axis.title = element_text(size = 10),
    axis.text = element_text(size = 16),
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 16),
    legend.position = "bottom"
  ) +
  labs(x = "Completeness Score", y = "Frequency")

# Print the plot
print(FundingSource_plot)
```




```
# Save the plot to a file
ggsave("FundingSource_status_distribution_plot.png", FundingSource_plot, widt
h = 8, height = 6, bg = "white")

# Assuming 'data' is your DataFrame and has been loaded correctly
library(dplyr)
```

```r
# Transform the Completeness score into a binary variable (high vs. low)
data <- data %>%
  mutate(CompletenessCategory = case_when(
    Complete %in% 3:4 ~ "High",
    Complete %in% 1:2 ~ "Low",
    TRUE ~ NA_character_  # Handle potential unexpected values
  ))

# Assuming FundingSource is already in a format that can be directly categori
zed
# If not, you should transform FundingSource into categorical variable simila
r to CompletenessCategory
# Example transformation is shown in the comment below. Adjust it based on yo
ur data structure.
# data <- data %>%
#   mutate(FundingSourceCategory = case_when(
#     FundingSource == some_condition ~ "Category1",
#     FundingSource == another_condition ~ "Category2",
#     TRUE ~ NA_character_  # Handle potential unexpected values
#   ))

# Drop rows with NA in either category if any exist due to unexpected values
data <- data %>%
  filter(!is.na(CompletenessCategory) & !is.na(FundingSource))

# Create a contingency table
contingency_table <- table(data$CompletenessCategory, data$FundingSource)

# Conduct a Chi-square test of independence
chi_square_result <- chisq.test(contingency_table)

## Warning in chisq.test(contingency_table): Chi-squared approximation may be
## incorrect

# Print the result of the Chi-square test
print(chi_square_result)

# Print the result of the Chi-square test
print(chi_square_result)

##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 1.2984, df = 1, p-value = 0.2545
```

#2 - Ordinal regression models (Scoring Criteria and year)

```
library(ordinal) # ordinal logistic regression: cumulative link mixed models, clm function is in this package; For a detailed explanation of the package and the functions available see: https://cran.r-project.org/web/packages/ordinal/vignettes/clm_article.pdf
```

```
## Warning: package 'ordinal' was built under R version 4.2.3
```

```
##
## Attaching package: 'ordinal'
```

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

```
library(VGAM) # more ordinal regression
```

```
## Warning: package 'VGAM' was built under R version 4.2.3
```

```
## Loading required package: stats4
```

```
## Loading required package: splines
```

```
##
## Attaching package: 'VGAM'
```

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

```
library(dplyr) # Data manipulation
library(chisq.posthoc.test) # If needed
```

```
## Warning: package 'chisq.posthoc.test' was built under R version 4.2.3
```

```
library(gmodels) # For SPSS style chi-sq/ contingency tables
```

```
## Warning: package 'gmodels' was built under R version 4.2.3
```

```
library(emmeans)
```

```
## Warning: package 'emmeans' was built under R version 4.2.3
```

```
library(sure)
```

```
## Warning: package 'sure' was built under R version 4.2.3
```

```
data$Complete <- factor(data$Complete, ordered = TRUE)
data$Reuse <- factor(data$Reuse, ordered = TRUE)
data$Access <- factor(data$Access, ordered = TRUE)
data$Licence <- factor(data$Licence, ordered = TRUE)
data$Year <- as.numeric(data$Year)
```

#Ordinal regression model for the Completeness by year

```
m1a <- clm(Complete ~ Year, data = data)
```

```
## Warning in x$code == 0L || action == "silent": 'length(x) = 2 > 1' in coer
cion
## to 'logical(1)'
```

```
## Warning: (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```
summary_m1a <- summary(m1a)

# Extract coefficients and standard errors
coefs <- summary(m1a)$coefficients
OR <- exp(coefs[,1])
lower_ci <- exp(coefs[,1] - 1.96 * coefs[,2])
upper_ci <- exp(coefs[,1] + 1.96 * coefs[,2])

# Calculate z-values and p-values
z_values <- coefs[,1] / coefs[,2]
p_values <- 2 * (1 - pnorm(abs(z_values)))

# Combine everything into a data frame for easy viewing
results <- data.frame(OR = OR, LowerCI = lower_ci, UpperCI = upper_ci, p_valu
e = p_values)
print(results)
```

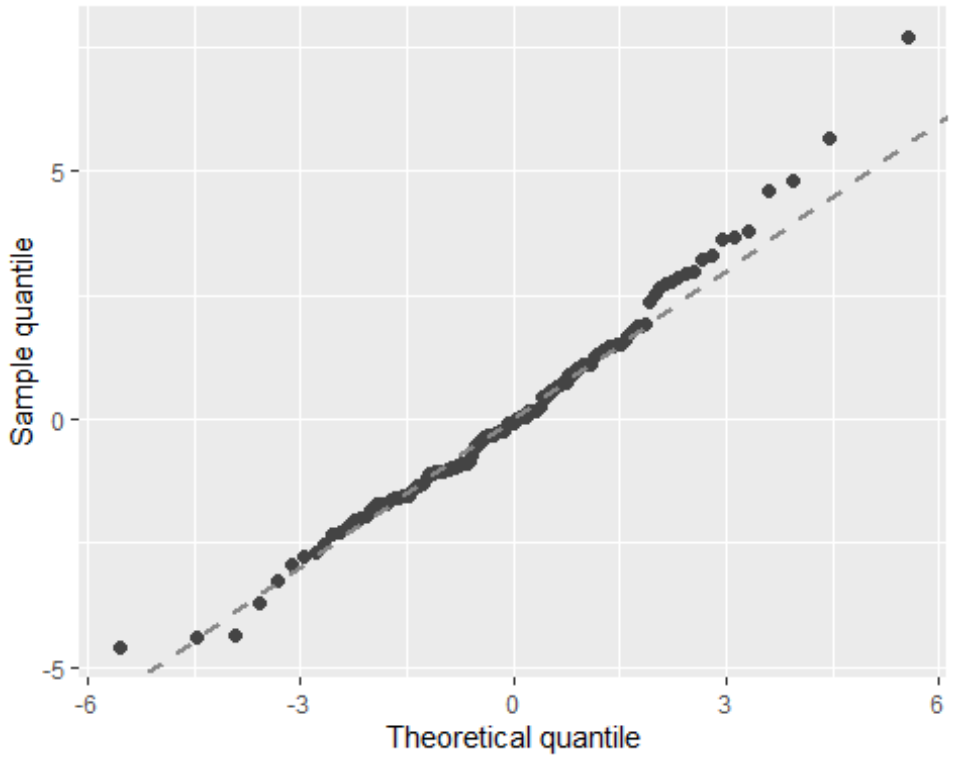
```
##                  OR      LowerCI       UpperCI     p_value
## 1|2  1.622844e+151 2.282840e+50 1.153661e+252 0.003296030
## 2|3  4.125302e+152 5.138138e+51 3.312117e+253 0.003032294
## 3|4  1.117896e+153 1.334844e+52 9.362075e+253 0.002955632
## Year  1.190114e+00 1.060719e+00  1.335293e+00 0.003038852
```

```
#Check the Assumptions
nominal_test(m1a) # This is a test of the proportional odds assumption, odds
are proportional in this case.
```

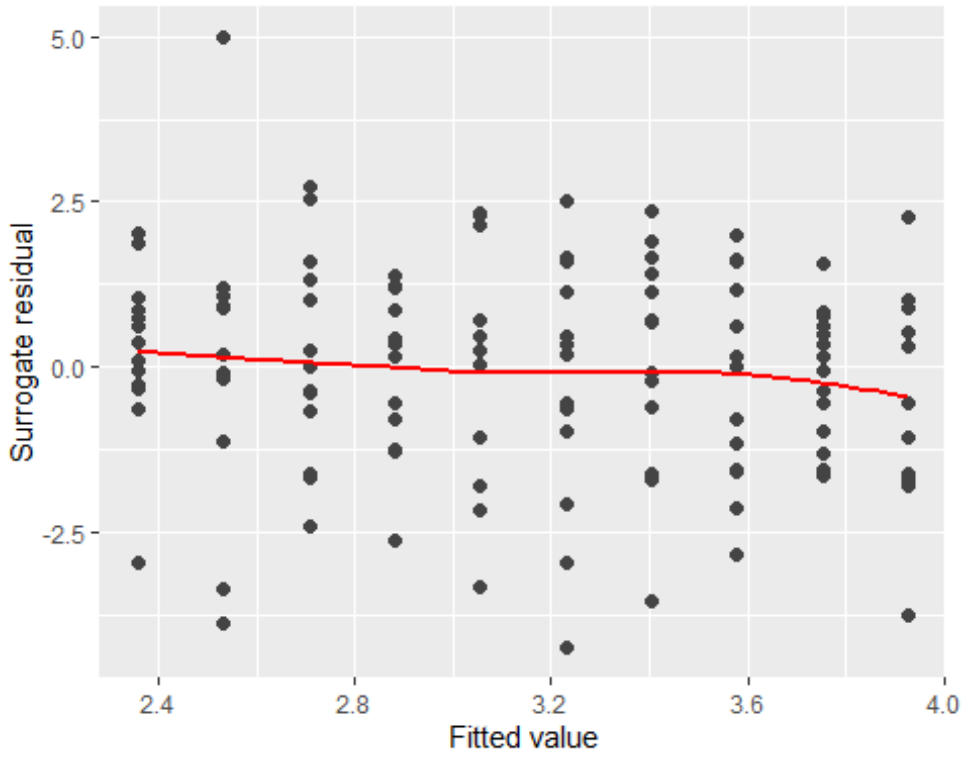
```
## Tests of nominal effects
##
## formula: Complete ~ Year
##        Df  logLik    AIC LRT Pr(>Chi)
## <none>    -148.39 304.78
## Year
```

```
scale_test(m1a) # scale test checks for equal variance/ scale across year, as
sumptions are not violated here because p = 0.8386. If p is less than 0.05 th
en assumptions are violated
```

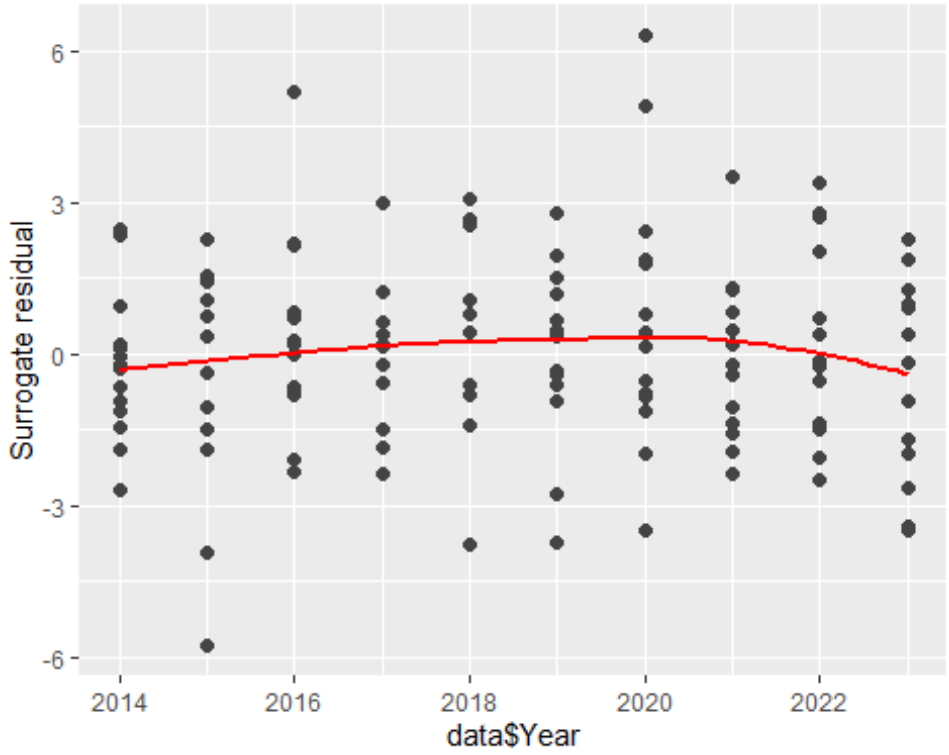
```
## Warning: (-1) Model failed to converge with max|grad| = 154.368 (tol = 1e-
06)
## In addition: iteration limit reached

## Tests of scale effects
##
## formula: Complete ~ Year
##        Df  logLik    AIC LRT Pr(>Chi)
## <none>    -148.39 304.78
## Year

convergence(m1a) # This is another way to assess the model

##  nobs logLik  niter max.grad cond.H  logLik.Error
##  130  -148.39 7(0)  6.62e-10 6.3e+12 <1e-10
##
##      Estimate   Std.Err  Gradient     Error Cor.Dec Sig.Dig
## 1|2   348.175 118.47952 -8.05e-14 -9.96e-11       9      12
## 2|3   351.410 118.54160  4.79e-13 -9.97e-11       9      12
## 3|4   352.407 118.56313 -7.34e-13 -9.97e-11       9      12
## Year    0.174   0.05873  6.62e-10 -4.94e-14      13      13
##
## Eigen values of Hessian:
## 1.484e+08 6.886e+01 8.253e+00 2.371e-05
##
## Convergence message from clm:
## (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## (3) Model is nearly unidentifiable: large eigenvalue ratio
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met

######## Graphically validate proportional odds using the sure package ######
#

autoplot.clm(m1a, what = c("qq")) # most of the points fall along the line, s
o no violation of linearity
```




```
autoplot.clm(m1a, what = c("fitted")) # the residuals do not show a clear pat
tern or trend

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
autoplot.clm(m1a, what = c("covariate"), x = data$Year) # scale test indicate
d no variance issues. This residual, covariate plots seems acceptable as well

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
# N.B. - Interpreting residual plots is largely subjective!

###plotting the model
# Predict probabilities for each level of 'Complete'
new_data <- data.frame(Year = sort(unique(data$Year)))
pred_probs <- predict(m1a, newdata = new_data, type = "prob")

# Add the 'Year' column to the predicted probabilities dataframe
pred_probs_df <- cbind(new_data, as.data.frame(pred_probs))

# Convert the predicted probabilities to a long format for ggplot
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.2.3
```

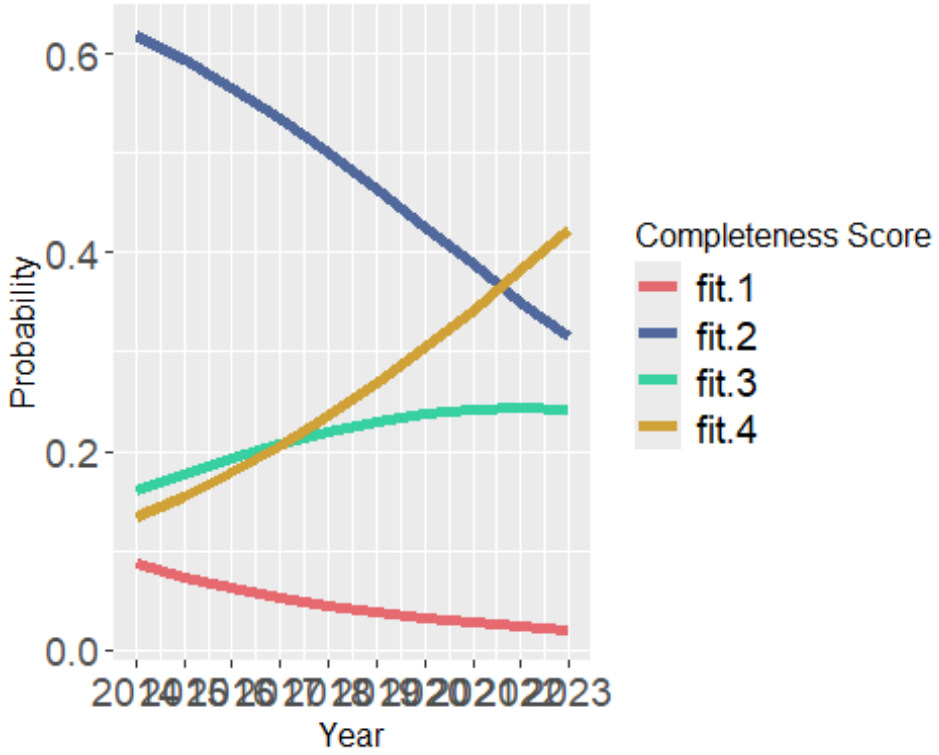
```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
pred_probs_long <- melt(pred_probs_df, id.vars = 'Year', variable.name = 'Com
pleteLevel', value.name = 'PredictedProbability')

# Set the colors
my_colors <- c("#E5696F", "#50689B", "#36D0A1", "#D0A136")
```

```
# Modify ggplot command
og1 <- ggplot(pred_probs_long, aes(x = Year, y = PredictedProbability, group
= CompleteLevel, color = CompleteLevel)) +
  geom_line(size = 2) +  # Set size of the lines to make them thicker
  scale_color_manual(values = my_colors) +
  scale_x_continuous(breaks = unique(pred_probs_long$Year)) +  # Show all yea
rs on x-axis
  labs(x = "Year", y = "Probability", color = "Completeness Score") +
  theme(
    axis.title = element_text(size = 12), # Increase axis titles
    axis.text = element_text(size = 14),  # Increase axis text
    legend.title = element_text(size = 12), # Increase legend title
    legend.text = element_text(size = 14)  # Increase legend text
  )
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
print(og1)
```




```
ggsave("og1.png", og1, width = 15, height = 10, units = "in",  bg = "white")
```

#Ordinal regression model for the Reusibility by year

```r
m1a <- clm(Reuse ~ Year, data = data)
```

```
## Warning in x$code == 0L || action == "silent": 'length(x) = 2 > 1' in coer
cion
## to 'logical(1)'
```

```
## Warning: (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```r
summary_m1a <- summary(m1a)


# Extract coefficients and standard errors
coefs <- summary(m1a)$coefficients
OR <- exp(coefs[,1])
lower_ci <- exp(coefs[,1] - 1.96 * coefs[,2])
upper_ci <- exp(coefs[,1] + 1.96 * coefs[,2])

# Calculate z-values and p-values
z_values <- coefs[,1] / coefs[,2]
p_values <- 2 * (1 - pnorm(abs(z_values)))

# Combine everything into a data frame for easy viewing
results <- data.frame(OR = OR, LowerCI = lower_ci, UpperCI = upper_ci, p_valu
e = p_values)
print(results)
```

```
##                  OR      LowerCI       UpperCI     p_value
## 1|2  1.068091e+156 2.411247e+57 4.731236e+254 0.001934390
## 2|3  2.302380e+157 4.592835e+58 1.154179e+256 0.001778537
## 3|4  5.188970e+157 9.966839e+58 2.701500e+256 0.001739712
## Year  1.196679e+00 1.069257e+00  1.339285e+00 0.001773416
```

```r
#Check the Assumptions
nominal_test(m1a) # This is a test of the proportional odds assumption, odds
are proportional in this case.
```

```
## Tests of nominal effects
##
## formula: Reuse ~ Year
##        Df  logLik    AIC LRT Pr(>Chi)
## <none>    -148.73 305.47
## Year
```

```r
scale_test(m1a) # scale test checks for equal variance/ scale across year, as
sumptions are not violated here
```

```
## Warning: (-1) Model failed to converge with max|grad| = 124.904 (tol = 1e-
06)
## In addition: iteration limit reached
```

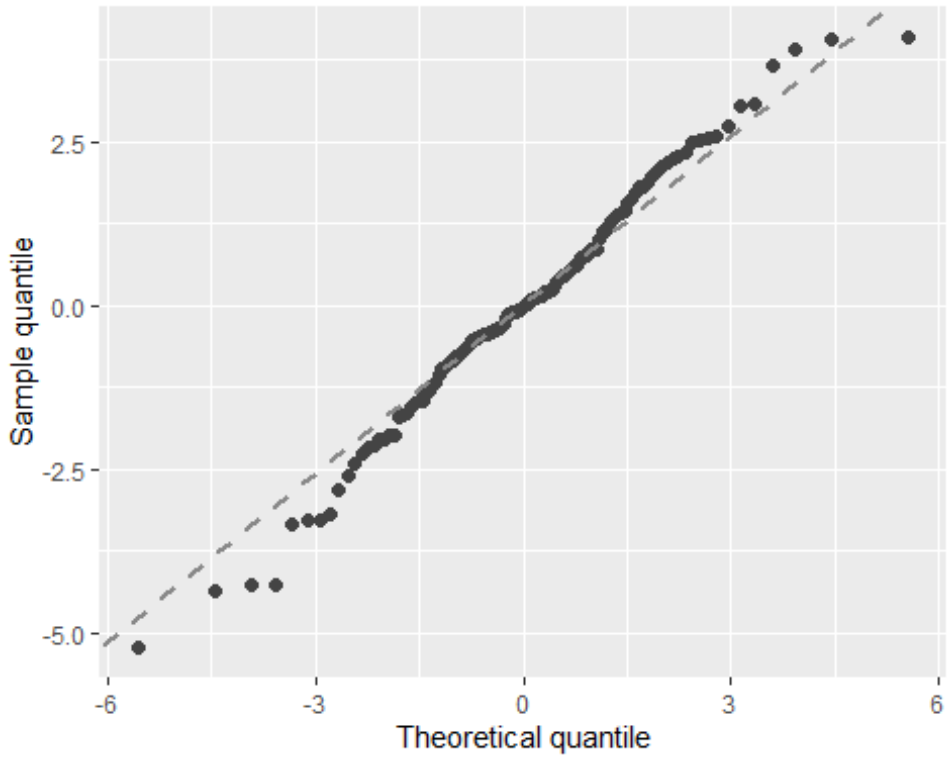
```
## Tests of scale effects
##
## formula: Reuse ~ Year
##        Df  logLik    AIC LRT Pr(>Chi)
## <none>    -148.73 305.47
## Year
```

```
convergence(m1a) # This is another way to assess the model
```

```
##  nobs logLik  niter max.grad cond.H  logLik.Error
##  130  -148.73 7(0)  6.29e-10 6.0e+12 <1e-10
##
##      Estimate   Std.Err  Gradient    Error Cor.Dec Sig.Dig
## 1|2  359.2691 115.88860 -1.73e-13 1.13e-11      10      13
## 2|3  362.3398 115.95172  1.92e-12 1.13e-11      10      13
## 3|4  363.1524 115.97102 -2.06e-12 1.13e-11      10      13
## Year   0.1796   0.05744  6.29e-10 5.60e-15      13      13
##
## Eigen values of Hessian:
## 1.496e+08 8.699e+01 8.406e+00 2.480e-05
##
## Convergence message from clm:
## (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## (3) Model is nearly unidentifiable: large eigenvalue ratio
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

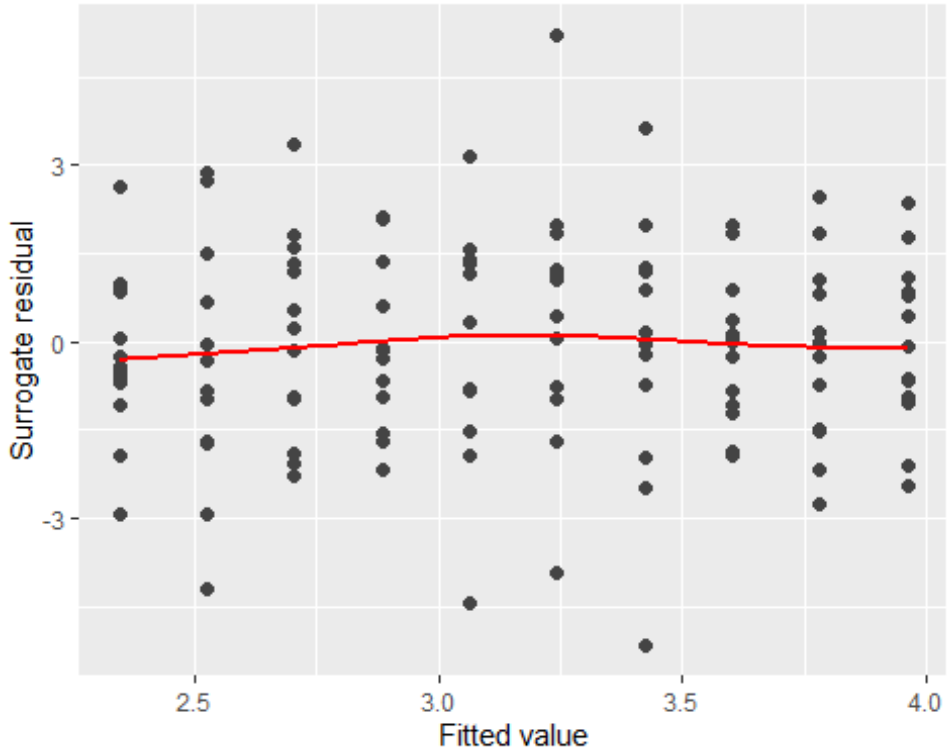
```
######## Graphically validate proportional odds using the sure package ######
#
```

```
autoplot.clm(m1a, what = c("qq")) # most of the points fall along the line, s
o no violation of linearity
```

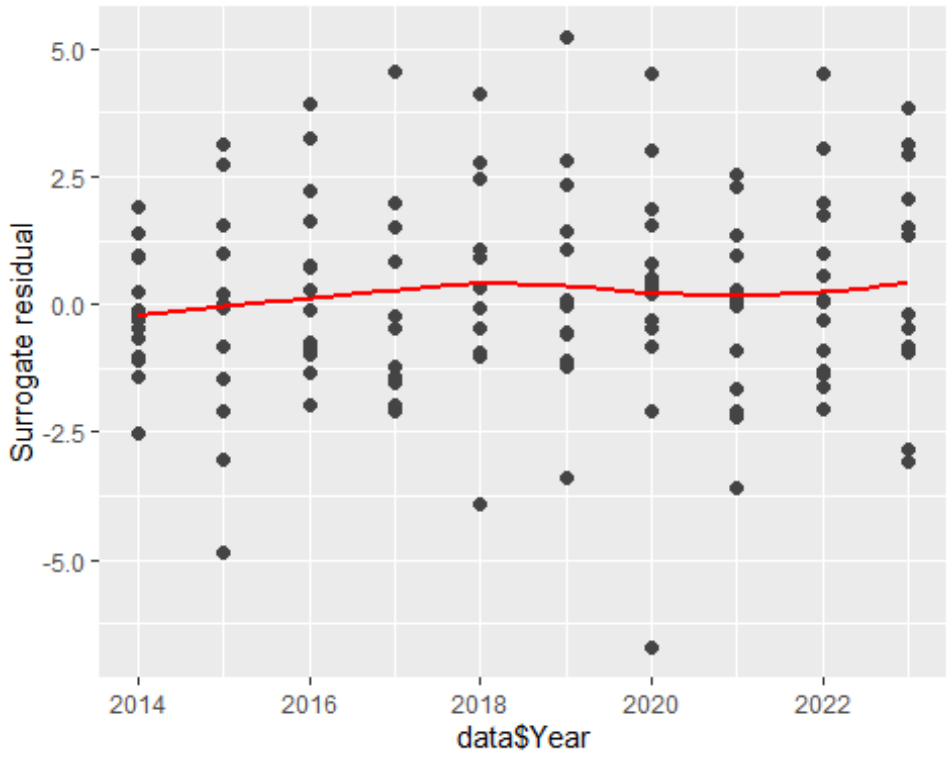



```
autoplot.clm(m1a, what = c("fitted")) # the residuals do not show a clear pat
tern or trend
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

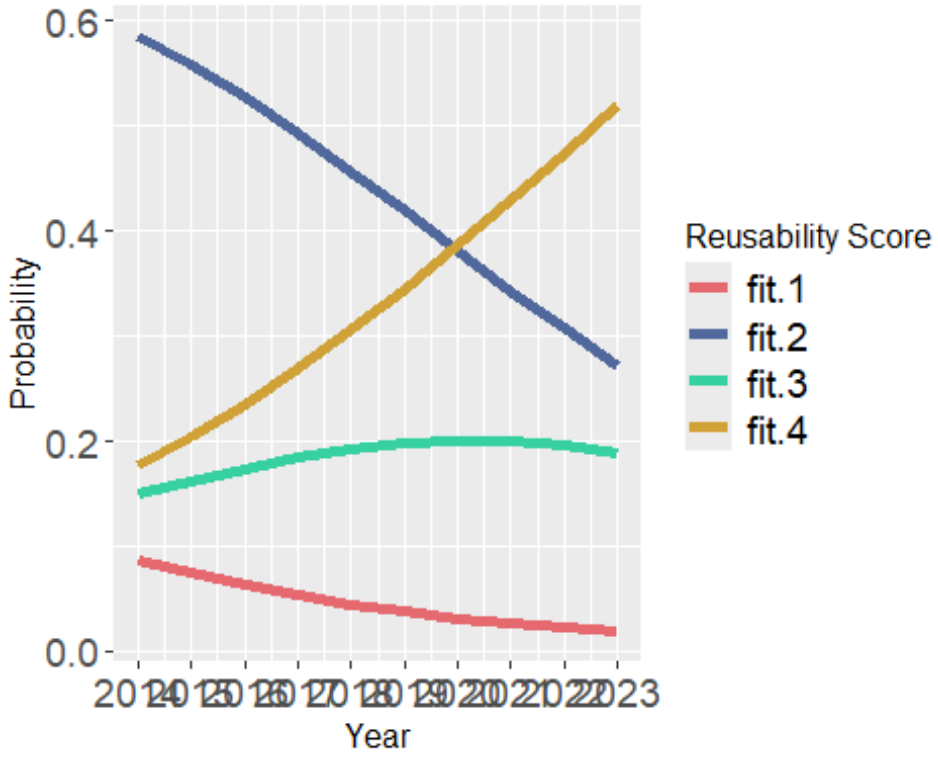



```
autoplot.clm(m1a, what = c("covariate"), x = data$Year) # scale test indicate
d no variance issues. This residual, covariate plots seems acceptable as well

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```r
# N.B. - Interpreting residual plots is largely subjective!

###plotting the model
# Predict probabilities for each level of 'Complete'
new_data <- data.frame(Year = sort(unique(data$Year)))
pred_probs <- predict(m1a, newdata = new_data, type = "prob")

# Add the 'Year' column to the predicted probabilities dataframe
pred_probs_df <- cbind(new_data, as.data.frame(pred_probs))

# Convert the predicted probabilities to a long format for ggplot
library(reshape2)
pred_probs_long <- melt(pred_probs_df, id.vars = 'Year', variable.name = 'Reu
selevel', value.name = 'PredictedProbability')

# Set the colors
my_colors <- c("#E5696F", "#50689B", "#36D0A1", "#D0A136")

# Modify ggplot command
og2 <- ggplot(pred_probs_long, aes(x = Year, y = PredictedProbability, group
= Reuselevel, color = Reuselevel)) +
  geom_line(size = 2) +  # Set size of the lines to make them thicker
  scale_color_manual(values = my_colors) +  # Use colors
  scale_x_continuous(breaks = unique(pred_probs_long$Year)) +  # Show all yea
rs on x-axis
  labs(x = "Year", y = "Probability", color = "Reusability Score") +
```

```
  theme(
    axis.title = element_text(size = 12), # Increase axis titles
    axis.text = element_text(size = 14),  # Increase axis text
    legend.title = element_text(size = 12), # Increase legend title
    legend.text = element_text(size = 14)  # Increase legend text
  )
print(og2)
```




```
ggsave("og2.png", og2, width = 15, height = 10, units = "in",  bg = "white")

m1a <- clm(Access ~ Year, data = data)

## Warning in x$code == 0L || action == "silent": 'length(x) = 2 > 1' in coer
cion
## to 'logical(1)'

## Warning: (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met

summary_m1a <- summary(m1a)


# Extract coefficients and standard errors
coefs <- summary_m1a$coefficients[, 1]  # Coefficients
se_coefs <- summary_m1a$coefficients[, 2]  # Standard errors
```

```r
# Extract coefficients and standard errors
coefs <- summary(m1a)$coefficients
OR <- exp(coefs[,1])
lower_ci <- exp(coefs[,1] - 1.96 * coefs[,2])
upper_ci <- exp(coefs[,1] + 1.96 * coefs[,2])

# Calculate z-values and p-values
z_values <- coefs[,1] / coefs[,2]
p_values <- 2 * (1 - pnorm(abs(z_values)))

# Combine everything into a data frame for easy viewing
results <- data.frame(OR = OR, LowerCI = lower_ci, UpperCI = upper_ci, p_valu
e = p_values)
print(results)
```

```
##                  OR      LowerCI  UpperCI      p_value
## 1|2  2.193927e+208 1.045919e+98      Inf 0.0002143828
## 2|3  4.895981e+209 2.007751e+99      Inf 0.0001967226
## 3|4  5.246755e+209 2.142653e+99      Inf 0.0001963546
## Year  1.270317e+00 1.120009e+00 1.440796 0.0001960819
```

```r
#Check the Assumptions
nominal_test(m1a) # This is a test of the proportional odds assumption, odds
are proportional in this case.
```

```
## Tests of nominal effects
##
## formula: Access ~ Year
##        Df  logLik   AIC LRT Pr(>Chi)
## <none>    -110.95 229.9
## Year
```

```r
scale_test(m1a) # scale test checks for equal variance/ scale across year, as
sumptions are not violated here
```

```
## Warning: (-1) Model failed to converge with max|grad| = 301.489 (tol = 1e-
06)
## In addition: iteration limit reached
```

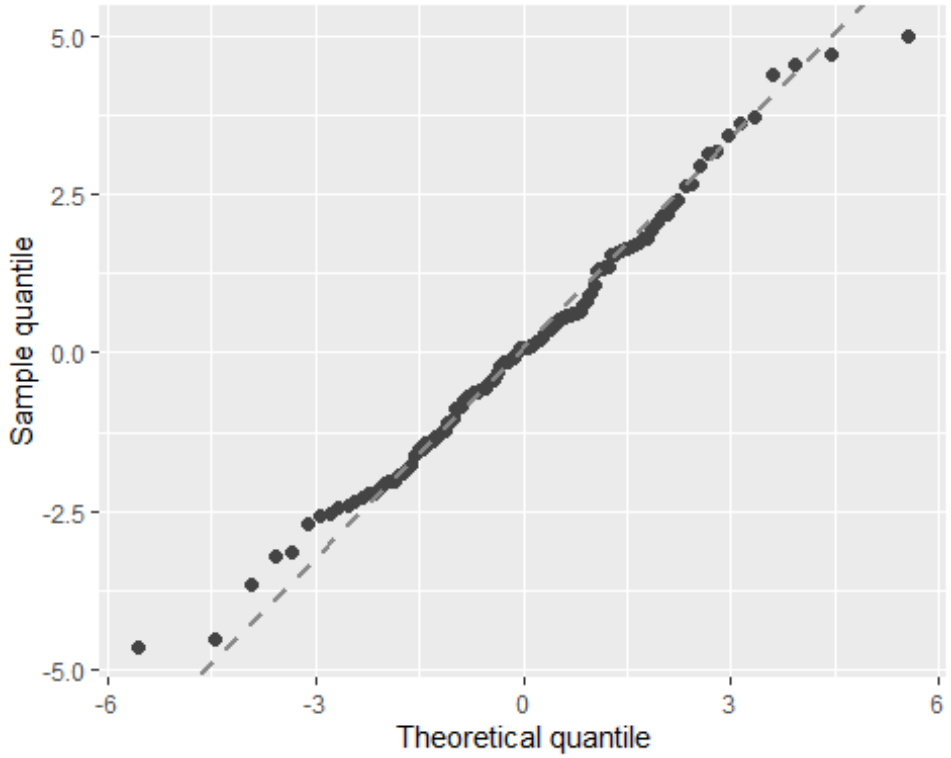
```
## Tests of scale effects
##
## formula: Access ~ Year
##        Df  logLik   AIC LRT Pr(>Chi)
## <none>    -110.95 229.9
## Year
```

```r
convergence(m1a) # This is another way to assess the model
```

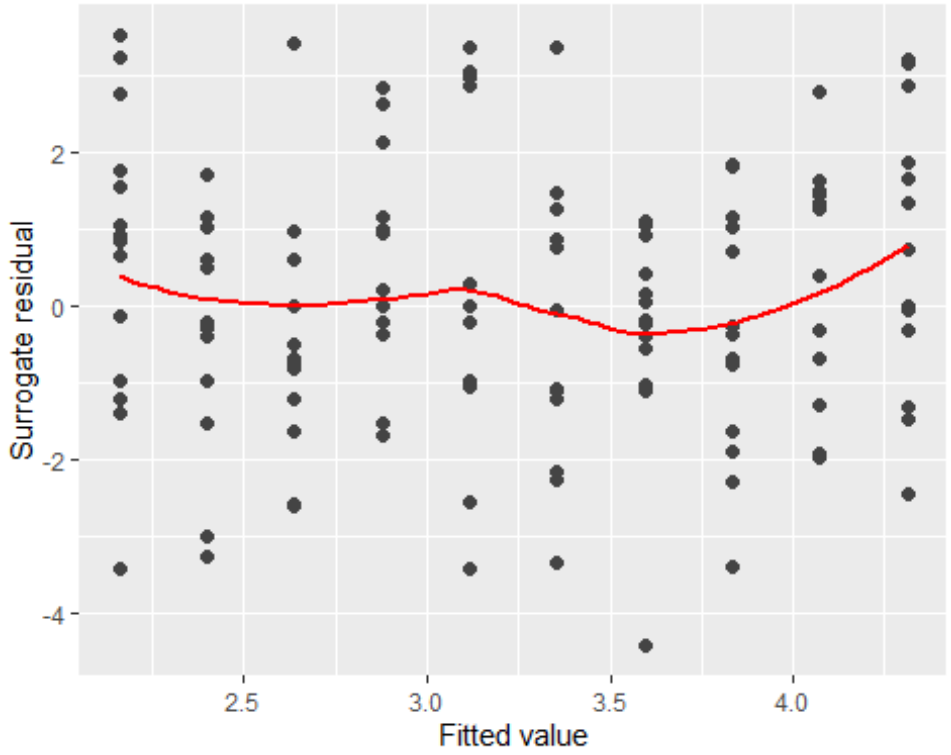
```
##  nobs logLik  niter max.grad cond.H  logLik.Error
##  130  -110.95 8(2)  1.15e-09 6.4e+12 <1e-10
##
```

```
##       Estimate   Std.Err  Gradient    Error Cor.Dec Sig.Dig
## 1|2  479.7234 129.60467 -2.94e-13 2.78e-11      10      13
## 2|3  482.8287 129.68151  9.06e-12 2.79e-11      10      13
## 3|4  482.8979 129.68364 -9.34e-12 2.79e-11      10      13
## Year   0.2393   0.06425  1.15e-09 1.38e-14      13      13
##
## Eigen values of Hessian:
## 1.276e+08 8.504e+02 8.325e+00 1.983e-05
##
## Convergence message from clm:
## (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## (3) Model is nearly unidentifiable: large eigenvalue ratio
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

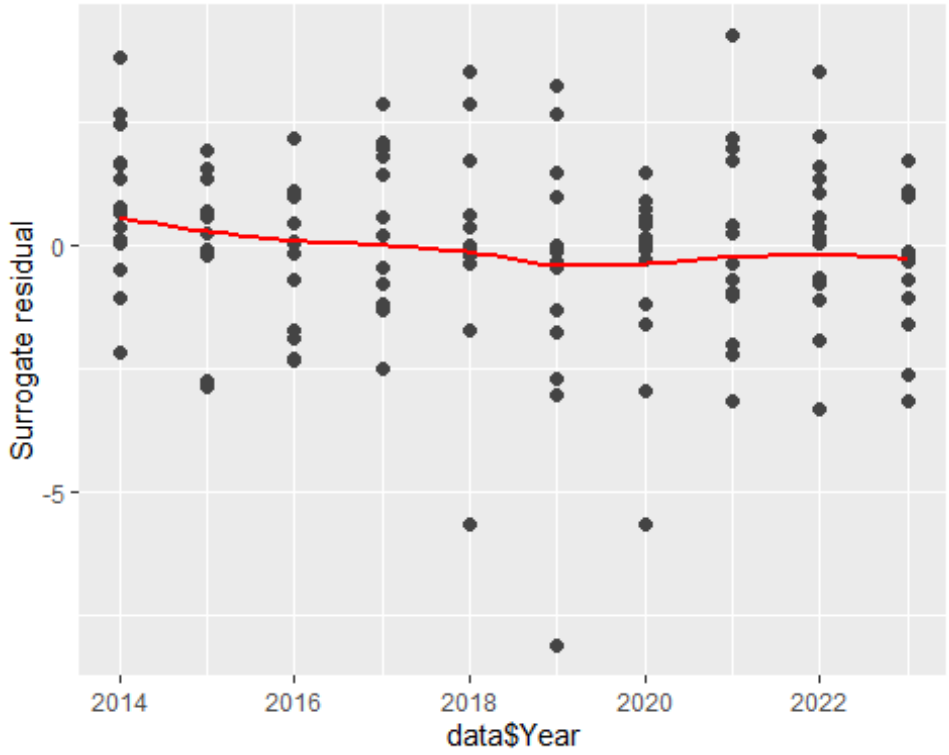
```
######## Graphically validate proportional odds using the sure package ######
#

autoplot.clm(m1a, what = c("qq")) # most of the points fall along the line, s
o no violation of linearity
```




```
autoplot.clm(m1a, what = c("fitted")) # the residuals do not show a clear pat
tern or trend

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

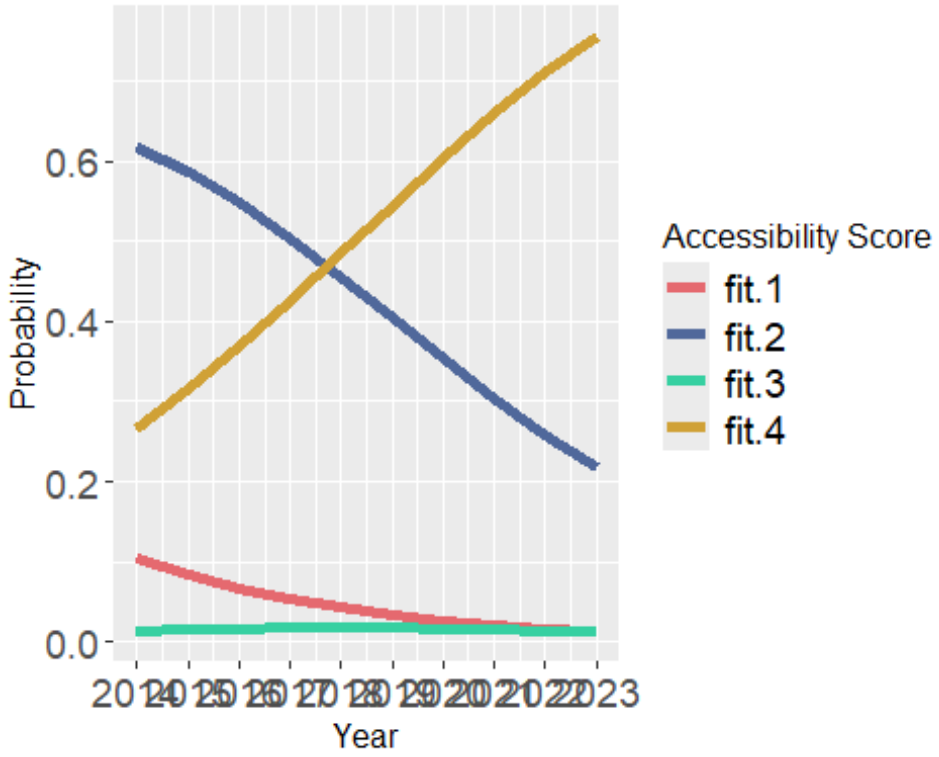



```
autoplot.clm(m1a, what = c("covariate"), x = data$Year) # scale test indicate
d no variance issues. This residual, covariate plots seems acceptable as well

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
# N.B. - Interpreting residual plots is largely subjective!

###plotting the model
# Predict probabilities for each level of 'Complete'
new_data <- data.frame(Year = sort(unique(data$Year)))
pred_probs <- predict(m1a, newdata = new_data, type = "prob")

# Add the 'Year' column to the predicted probabilities dataframe
pred_probs_df <- cbind(new_data, as.data.frame(pred_probs))

# Convert the predicted probabilities to a long format for ggplot
library(reshape2)
pred_probs_long <- melt(pred_probs_df, id.vars = 'Year', variable.name = 'Acc
essLevel', value.name = 'PredictedProbability')

# Set the colors
my_colors <- c("#E5696F", "#50689B", "#36D0A1", "#D0A136")

# Modify ggplot command
og3 <- ggplot(pred_probs_long, aes(x = Year, y = PredictedProbability, group
= AccessLevel, color = AccessLevel)) +
  geom_line(size = 2) +  # Set size of the lines to make them thicker
  scale_color_manual(values = my_colors) +  # Use colors
  scale_x_continuous(breaks = unique(pred_probs_long$Year)) +  # Show all yea
rs on x-axis
  labs(x = "Year", y = "Probability", color = "Accessibility Score") +
```

```
  theme(
    axis.title = element_text(size = 12), # Increase axis titles
    axis.text = element_text(size = 14),  # Increase axis text
    legend.title = element_text(size = 12), # Increase legend title
    legend.text = element_text(size = 14)  # Increase legend text
  )
print(og3)
```




```
ggsave("og3.png", og3, width = 15, height = 10, units = "in",  bg = "white")
```

```
m1a <- clm(Licence ~ Year, data = data)
```

```
## Warning in x$code == 0L || action == "silent": 'length(x) = 2 > 1' in coer
cion
## to 'logical(1)'
```

```
## Warning: (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

```
summary_m1a <- summary(m1a)


# Extract coefficients and standard errors
coefs <- summary_m1a$coefficients[, 1]  # Coefficients
se_coefs <- summary_m1a$coefficients[, 2]  # Standard errors
```

```r
# Extract coefficients and standard errors
coefs <- summary(m1a)$coefficients
OR <- exp(coefs[,1])
lower_ci <- exp(coefs[,1] - 1.96 * coefs[,2])
upper_ci <- exp(coefs[,1] + 1.96 * coefs[,2])

# Calculate z-values and p-values
z_values <- coefs[,1] / coefs[,2]
p_values <- 2 * (1 - pnorm(abs(z_values)))

# Combine everything into a data frame for easy viewing
results <- data.frame(OR = OR, LowerCI = lower_ci, UpperCI = upper_ci, p_valu
e = p_values)
print(results)
```

```
##                OR      LowerCI       UpperCI   p_value
## 1|3  7.570575e+27 7.413337e-86 7.731148e+140 0.6287208
## 3|4  6.239265e+28 6.046581e-85 6.438089e+141 0.6175010
## Year 1.033865e+00 9.088113e-01  1.176126e+00 0.6126309
```

```r
#Check the Assumptions
nominal_test(m1a) # This is a test of the proportional odds assumption, odds
are proportional in this case.
```

```
## Tests of nominal effects
##
## formula: Licence ~ Year
##        Df  logLik    AIC LRT Pr(>Chi)
## <none>    -93.915 193.83
## Year
```

```r
scale_test(m1a) # scale test checks for equal variance/ scale across year, as
sumptions are not violated here
```

```
## Warning: (-1) Model failed to converge with max|grad| = 33798.1 (tol = 1e-
06)
## In addition: iteration limit reached
```

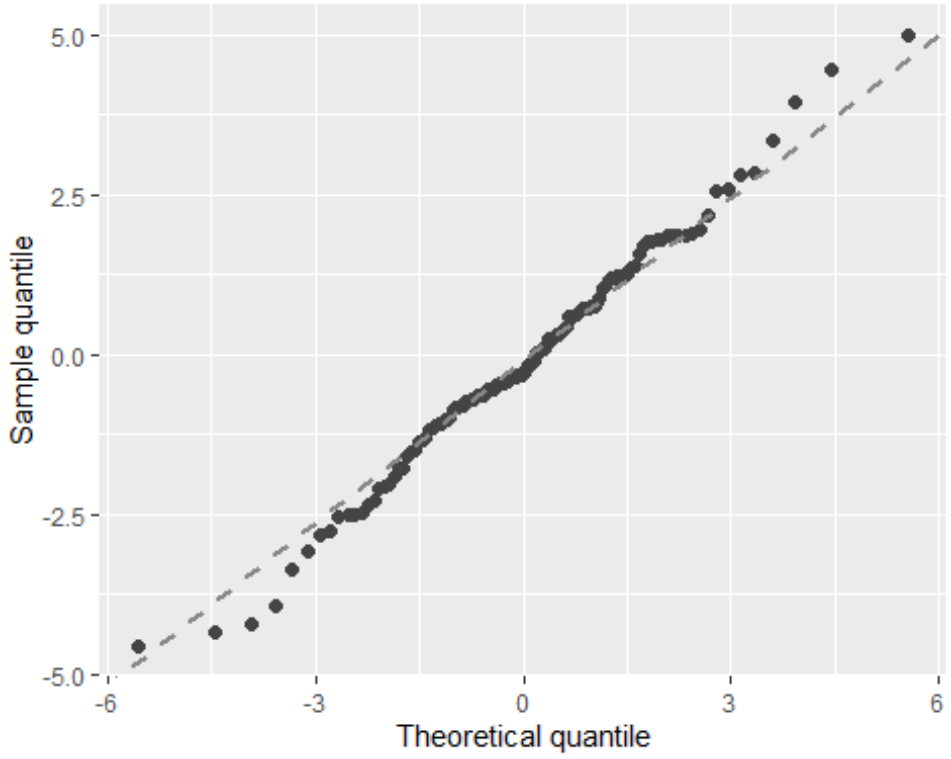
```
## Tests of scale effects
##
## formula: Licence ~ Year
##        Df  logLik    AIC LRT Pr(>Chi)
## <none>    -93.915 193.83
## Year
```

```r
convergence(m1a) # This is another way to assess the model
```

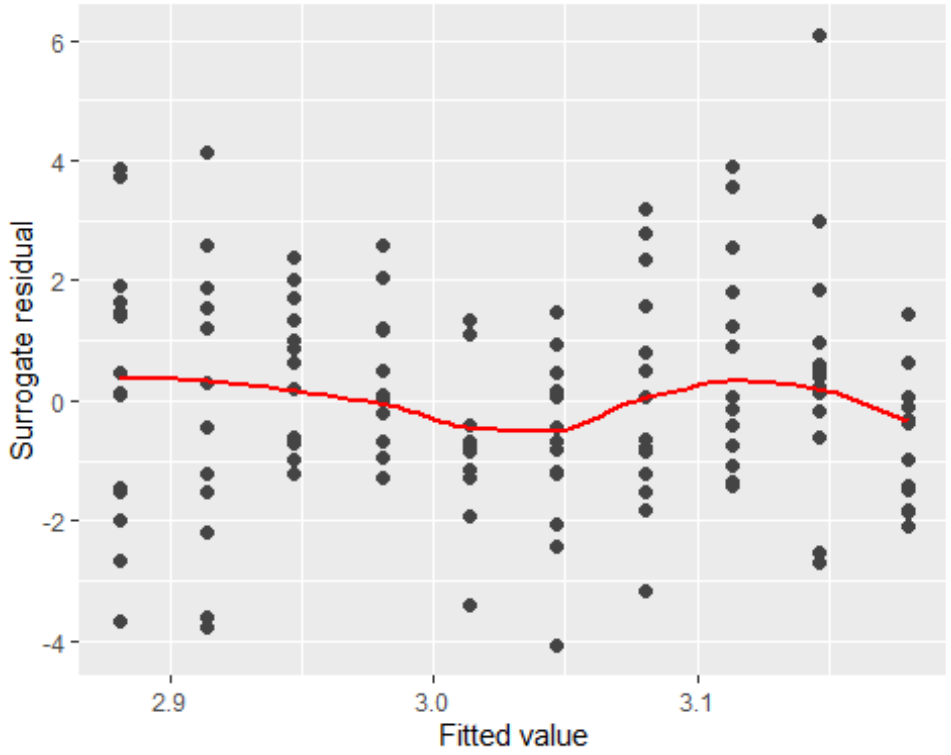
```
##  nobs logLik niter max.grad cond.H  logLik.Error
##  130  -93.92 6(0)  3.37e-10 3.9e+12 <1e-10
##
##      Estimate   Std.Err  Gradient     Error Cor.Dec Sig.Dig
```

```
## 1|3   64.1941 132.76179  3.93e-14 -7.29e-12      10      12
## 3|4   66.3032 132.76708  1.27e-13 -7.29e-12      10      12
## Year   0.0333   0.06578 -3.37e-10 -3.62e-15      14      13
##
## Eigen values of Hessian:
## 1.093e+08 1.282e+01 2.837e-05
##
## Convergence message from clm:
## (2) Model is nearly unidentifiable: very large eigenvalue
##  - Rescale variables?
## (3) Model is nearly unidentifiable: large eigenvalue ratio
##  - Rescale variables?
## In addition: Absolute and relative convergence criteria were met
```

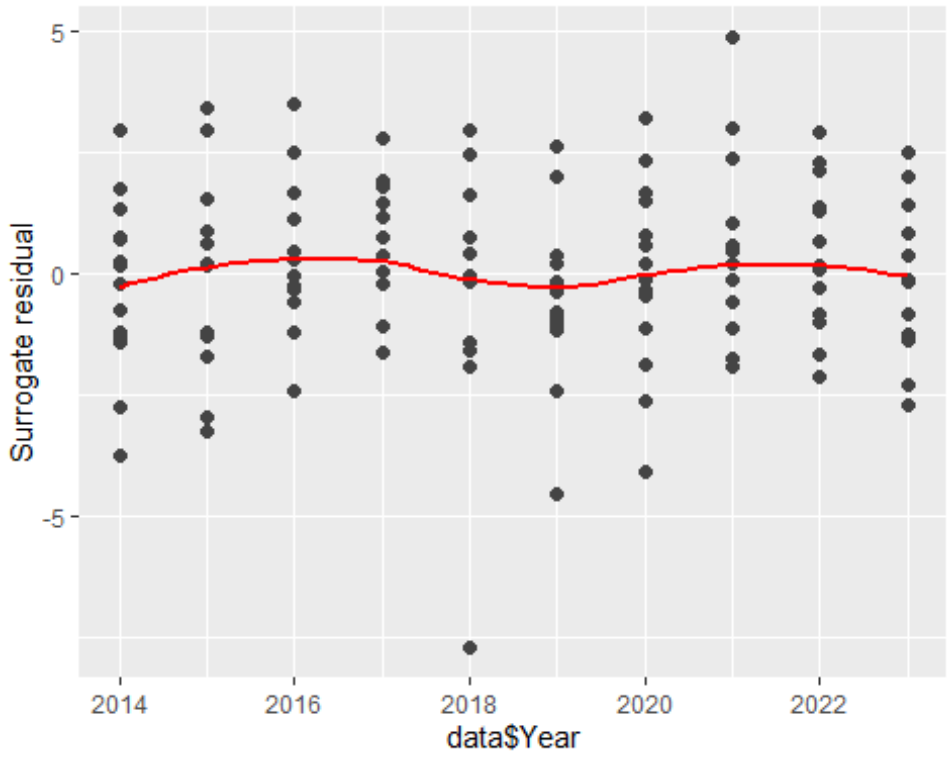
```
######## Graphically validate proportional odds using the sure package ######
#

autoplot.clm(m1a, what = c("qq")) # most of the points fall along the line, s
o no violation of linearity
```




```
autoplot.clm(m1a, what = c("fitted")) # the residuals do not show a clear pat
tern or trend

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

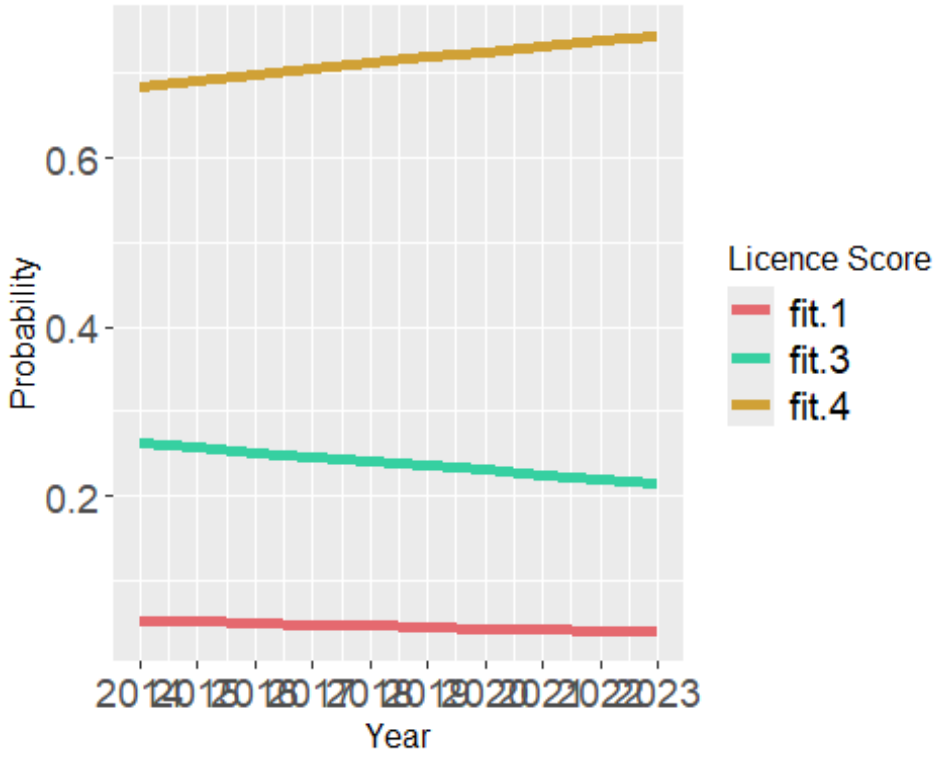



```
autoplot.clm(m1a, what = c("covariate"), x = data$Year) # scale test indicate
d no variance issues. This residual, covariate plots seems acceptable as well

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
# N.B. - Interpreting residual plots is largely subjective!

###plotting the model
# Predict probabilities for each level of 'Complete'
new_data <- data.frame(Year = sort(unique(data$Year)))
pred_probs <- predict(m1a, newdata = new_data, type = "prob")

# Add the 'Year' column to the predicted probabilities dataframe
pred_probs_df <- cbind(new_data, as.data.frame(pred_probs))

# Convert the predicted probabilities to a long format for ggplot
library(reshape2)
pred_probs_long <- melt(pred_probs_df, id.vars = 'Year', variable.name = 'Lic
encelevel', value.name = 'PredictedProbability')

# Set the colors
my_colors <- c("#E5696F", "#36D0A1", "#D0A136")

# Modify ggplot command
og4 <- ggplot(pred_probs_long, aes(x = Year, y = PredictedProbability, group
= Licencelevel, color = Licencelevel)) +
  geom_line(size = 2) +  # Set size of the lines to make them thicker
  scale_color_manual(values = my_colors) +  # Use colors
  scale_x_continuous(breaks = unique(pred_probs_long$Year)) +  # Show all yea
rs on x-axis
  labs(x = "Year", y = "Probability", color = "Licence Score") +
```

```
  theme(
    axis.title = element_text(size = 12), # Increase axis titles
    axis.text = element_text(size = 14),  # Increase axis text
    legend.title = element_text(size = 12), # Increase legend title
    legend.text = element_text(size = 14)  # Increase legend text
  )
print(og4)
```




```
ggsave("og4.png", og4, width = 15, height = 10, units = "in",  bg = "white")
```

```
install.packages("patchwork")
```

```
## Installing package into 'M:/R/Win-Library/4.2'
## (as 'lib' is unspecified)
```

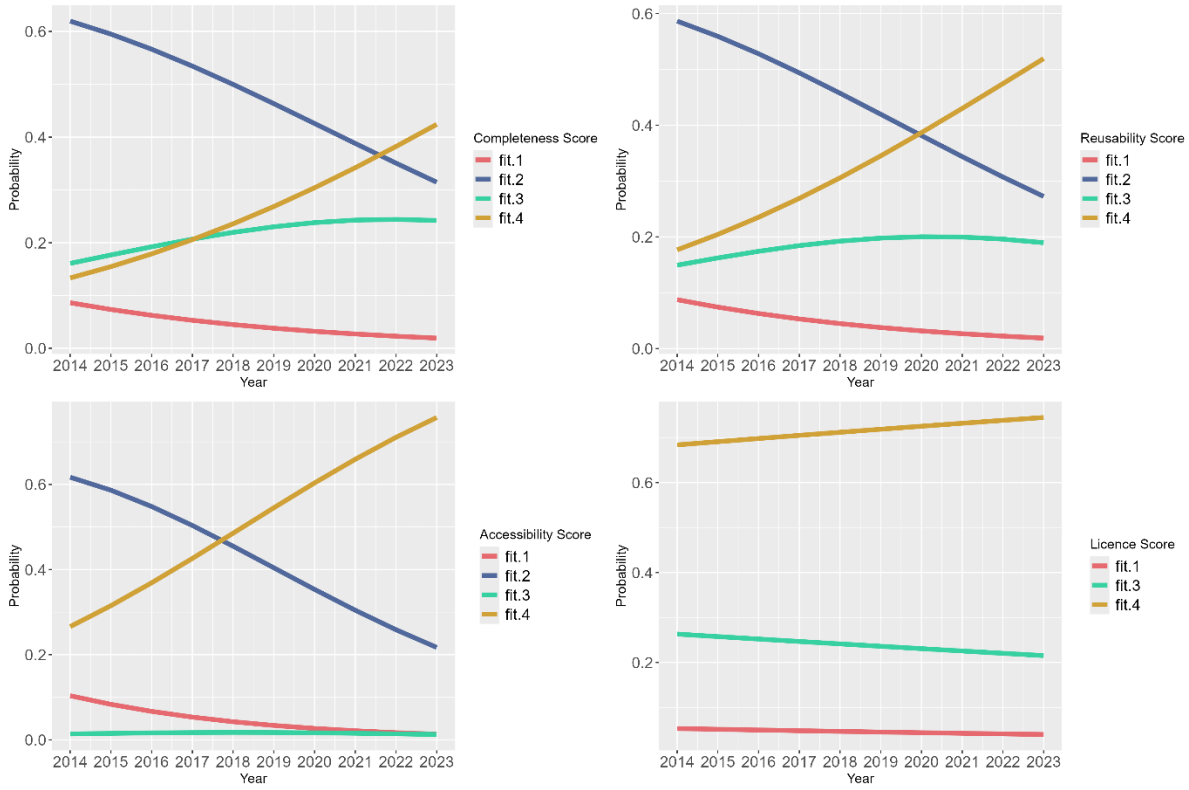
```
## package 'patchwork' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\s2146807\AppData\Local\Temp\RtmpGEDx57\downloaded_packages
```

```
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.2.3
```

```
##
## Attaching package: 'patchwork'
```

```
## The following object is masked from 'package:MASS':
##
##     area

ordinalplot <- og1 + og2 + og3 + og4 +
  plot_layout(
    ncol = 2, heights = c(10, 10), widths = c(10, 10)
  )
print(ordinalplot)
```




```
ggsave("ordinalplot.png", ordinalplot, width = 15, height = 10, units = "in",
bg="white")
```

#Ordinal Regression models depending on the sharing projects

```
m1a <- clm(Complete ~ NewDAS + Preprint, data = data)
summary_m1a <- summary(m1a)

# Extract coefficients and standard errors
coefs <- summary_m1a$coefficients[, 1]  # Coefficients
se_coefs <- summary_m1a$coefficients[, 2]  # Standard errors

# Extract coefficients and standard errors
coefs <- summary(m1a)$coefficients
OR <- exp(coefs[,1])
lower_ci <- exp(coefs[,1] - 1.96 * coefs[,2])
upper_ci <- exp(coefs[,1] + 1.96 * coefs[,2])
```

```r
# Calculate z-values and p-values
z_values <- coefs[,1] / coefs[,2]
p_values <- 2 * (1 - pnorm(abs(z_values)))

# Combine everything into a data frame for easy viewing
results <- data.frame(OR = OR, LowerCI = lower_ci, UpperCI = upper_ci, p_valu
e = p_values)
print(results)
```

```
##                          OR    LowerCI    UpperCI      p_value
## 1|2              0.08040527 0.03446409  0.1875867 5.481457e-09
## 2|3              2.30928285 1.40274308  3.8016850 9.996808e-04
## 3|4              6.98540751 3.86569749 12.6227979 1.201850e-10
## NewDASNot shared 2.22062795 0.50650870  9.7356443 2.900738e-01
## NewDASShared     4.43200155 2.15464581  9.1164115 5.207790e-05
## Preprint         3.11203452 1.21055487  8.0002642 1.844070e-02
```

```r
#Check the Assumptions
nominal_test(m1a) # This is a test of the proportional odds assumption, odds
are proportional in this case.
```

```
## Tests of nominal effects
##
## formula: Complete ~ NewDAS + Preprint
##          Df  logLik    AIC   LRT Pr(>Chi)
## <none>      -141.46 294.91
## NewDAS
## Preprint  2 -140.60 297.20 1.706   0.4261
```

```r
scale_test(m1a) # scale test checks for equal variance/ scale across year, as
sumptions are not violated here because p = 0.8386. If p is less than 0.05 th
en assumptions are violated
```

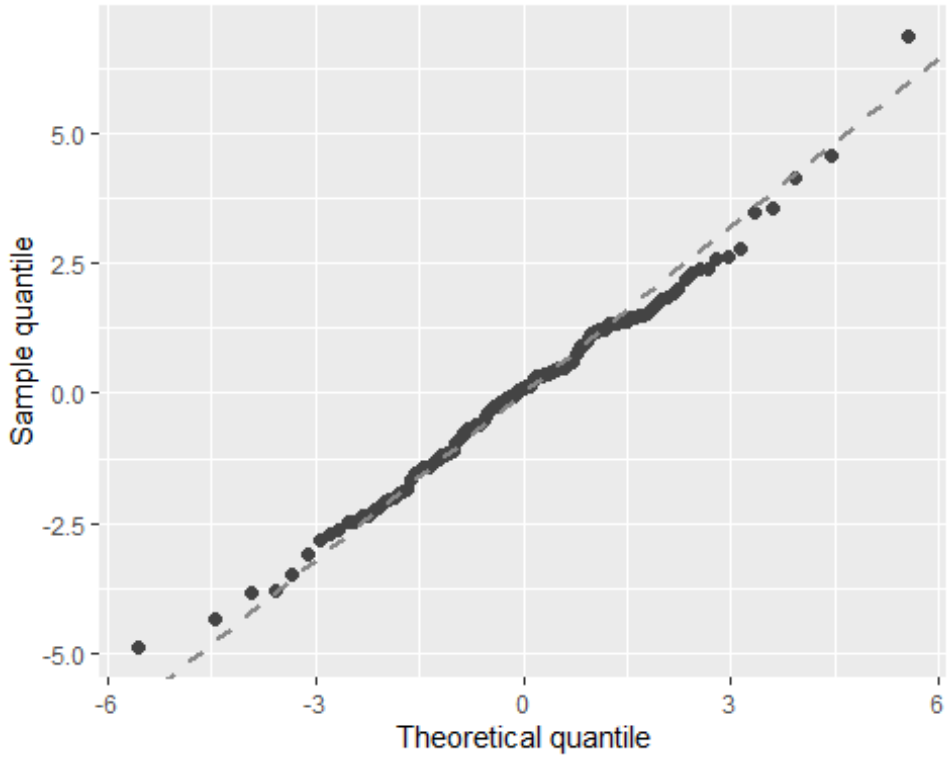
```
## Tests of scale effects
##
## formula: Complete ~ NewDAS + Preprint
##          Df  logLik    AIC     LRT Pr(>Chi)
## <none>      -141.46 294.91
## NewDAS    2 -141.06 298.12 0.78479   0.6754
## Preprint  1 -141.39 296.78 0.12885   0.7196
```

```r
convergence(m1a) # This is another way to assess the model
```

```
##  nobs logLik  niter max.grad cond.H  logLik.Error
##  130  -141.45 6(0)  2.36e-08 3.6e+01 <1e-10
##
##                  Estimate Std.Err  Gradient     Error Cor.Dec Sig.Dig
## 1|2               -2.5207  0.4322  2.36e-08  4.00e-09       8       9
## 2|3                0.8369  0.2543 -1.09e-08 -1.02e-10       9       9
## 3|4                1.9438  0.3019 -5.49e-12 -9.47e-11       9      10
```

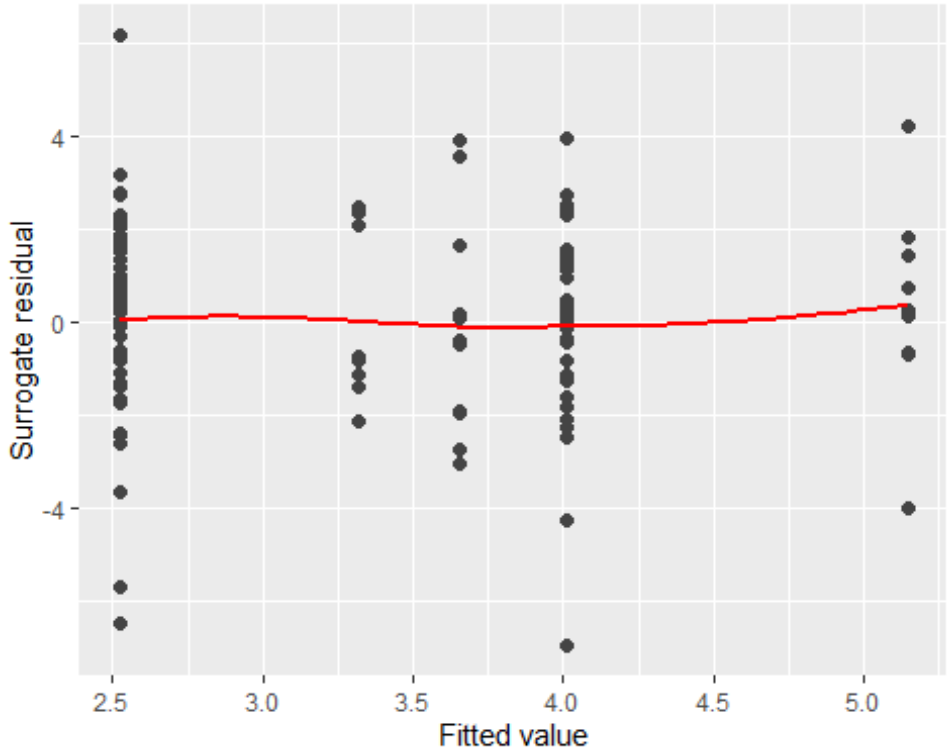
```
## NewDASNot shared   0.7978  0.7541 -6.88e-10 -9.85e-11       9       9
## NewDASShared       1.4889  0.3680 -9.83e-10 -9.04e-11       9      10
## Preprint           1.1353  0.4817 -6.26e-10 -6.25e-11       9      10
##
## Eigen values of Hessian:
## 58.846 25.507  6.644  5.242  3.748  1.627
##
## Convergence message from clm:
## (0) successful convergence
## In addition: Absolute and relative convergence criteria were met
```

```
######## Graphically validate proportional odds using the sure package ######
#

autoplot.clm(m1a, what = c("qq")) # most of the points fall along the line, s
o no violation of linearity
```



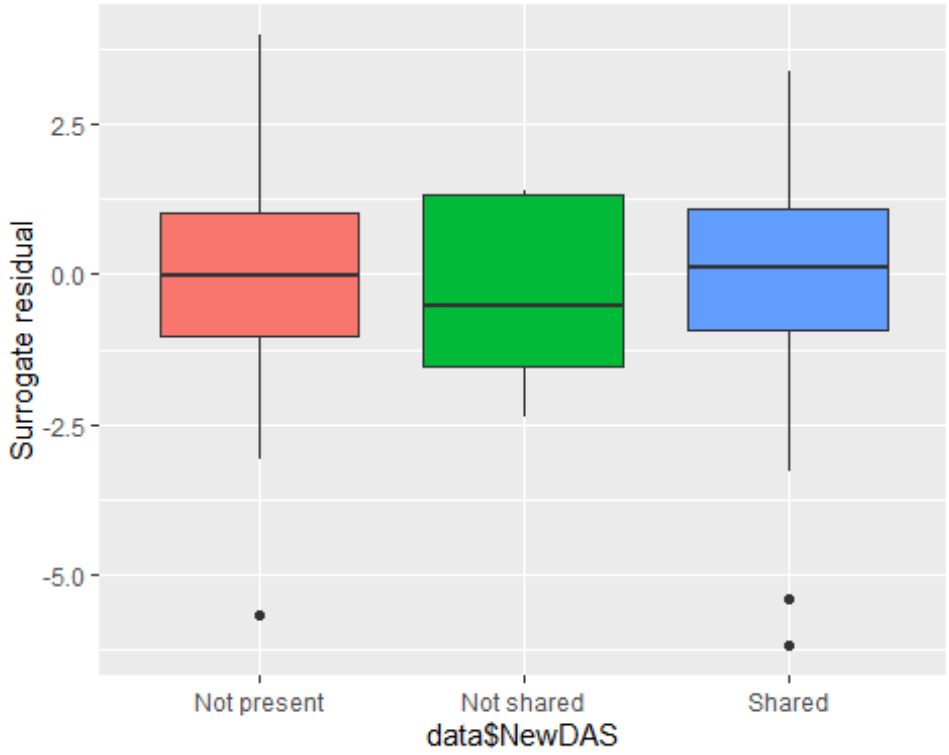

```
autoplot.clm(m1a, what = c("fitted")) # the residuals do not show a clear pat
tern or trend
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
autoplot.clm(m1a, what = c("covariate"), x = data$NewDAS) # scale test indica
ted no variance issues. This residual, covariate plots seems acceptable as we
ll
```

```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none
" instead as
## of ggplot2 3.3.4.
## ℹ The deprecated feature was likely used in the sure package.
##   Please report the issue at <]8;;https://github.com/AFIT-R/sure/issueshtt
ps://github.com/AFIT-R/sure/issues]8;;>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
autoplot.clm(m1a, what = c("covariate"), x = data$Preprint)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : at -0.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : radius 2.5e-05
```

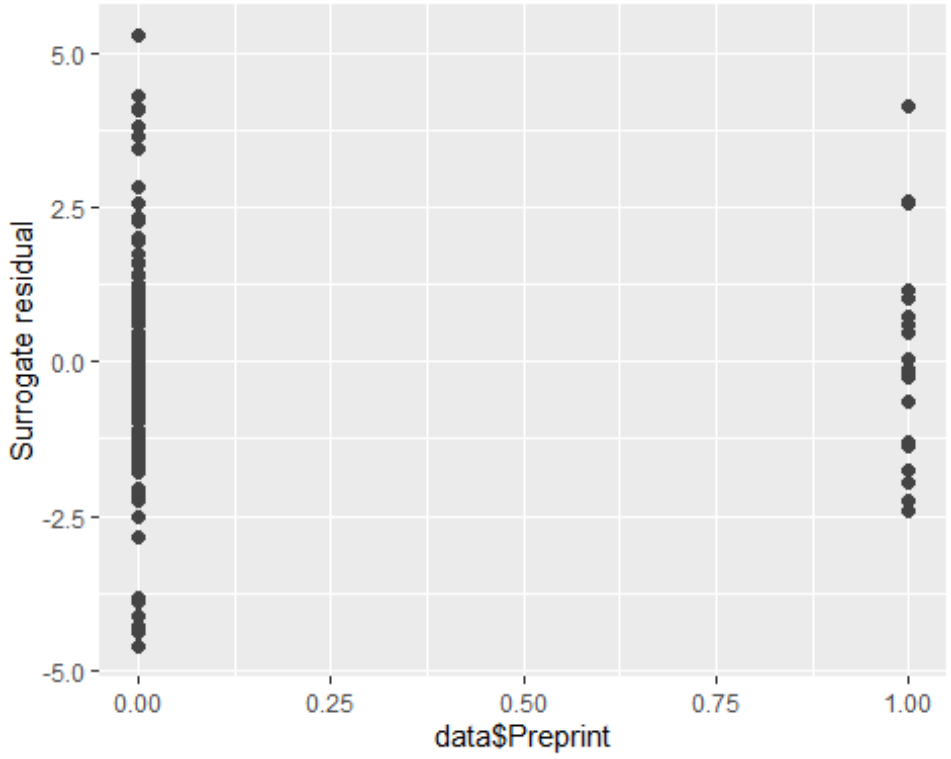
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : all data on boundary of neighborhood. make span bigger
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : pseudoinverse used at -0.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : neighborhood radius 0.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : reciprocal condition number 1
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : There are other near singularities as well. 1.01

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : zero-width neighborhood. make span bigger

## Warning: Failed to fit group -1.
## Caused by error in `predLoess()`:
## ! NA/NaN/Inf in foreign function call (arg 5)
```




```
# N.B. - Interpreting residual plots is largely subjective!

m1a <- clm(Reuse ~ NewDAS + Preprint, data = data)
summary_m1a <- summary(m1a)

# Extract coefficients and standard errors
coefs <- summary_m1a$coefficients[, 1]  # Coefficients
se_coefs <- summary_m1a$coefficients[, 2]  # Standard errors

# Extract coefficients and standard errors
coefs <- summary(m1a)$coefficients
OR <- exp(coefs[,1])
lower_ci <- exp(coefs[,1] - 1.96 * coefs[,2])
upper_ci <- exp(coefs[,1] + 1.96 * coefs[,2])
```

```r
# Calculate z-values and p-values
z_values <- coefs[,1] / coefs[,2]
p_values <- 2 * (1 - pnorm(abs(z_values)))

# Combine everything into a data frame for easy viewing
results <- data.frame(OR = OR, LowerCI = lower_ci, UpperCI = upper_ci, p_valu
e = p_values)
print(results)
```

```
##                           OR    LowerCI    UpperCI      p_value
## 1|2              0.08030323 0.03449122  0.1869638 4.945615e-09
## 2|3              1.94089488 1.19689021  3.1473839 7.173334e-03
## 3|4              4.95212549 2.84354848  8.6242760 1.584018e-08
## NewDASNot shared 3.19798580 0.83340049 12.2715469 9.019262e-02
## NewDASShared     6.37209386 3.00525060 13.5108800 1.367981e-06
## Preprint         1.64953102 0.65229017  4.1713837 2.903525e-01
```

```r
#Check the Assumptions
nominal_test(m1a) # This is a test of the proportional odds assumption, odds
are proportional in this case.
```

```
## Tests of nominal effects
##
## formula: Reuse ~ NewDAS + Preprint
##          Df  logLik    AIC    LRT Pr(>Chi)
## <none>      -139.92 291.85
## NewDAS    4 -138.18 296.36 3.4880   0.4797
## Preprint  2 -139.22 294.43 1.4187   0.4920
```

```r
scale_test(m1a) # scale test checks for equal variance/ scale across year, as
sumptions are not violated here because p = 0.8386. If p is less than 0.05 th
en assumptions are violated
```

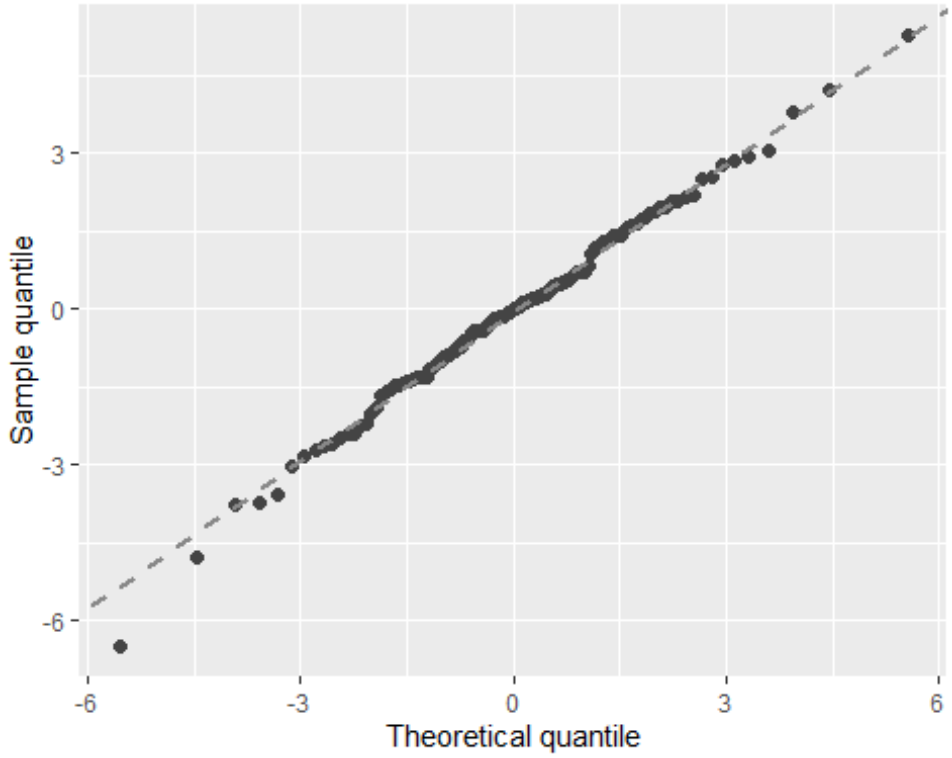
```
## Tests of scale effects
##
## formula: Reuse ~ NewDAS + Preprint
##          Df  logLik    AIC    LRT Pr(>Chi)
## <none>      -139.92 291.85
## NewDAS    2 -138.55 293.11 2.7396   0.2542
## Preprint  1 -139.28 292.57 1.2817   0.2576
```

```r
convergence(m1a) # This is another way to assess the model
```

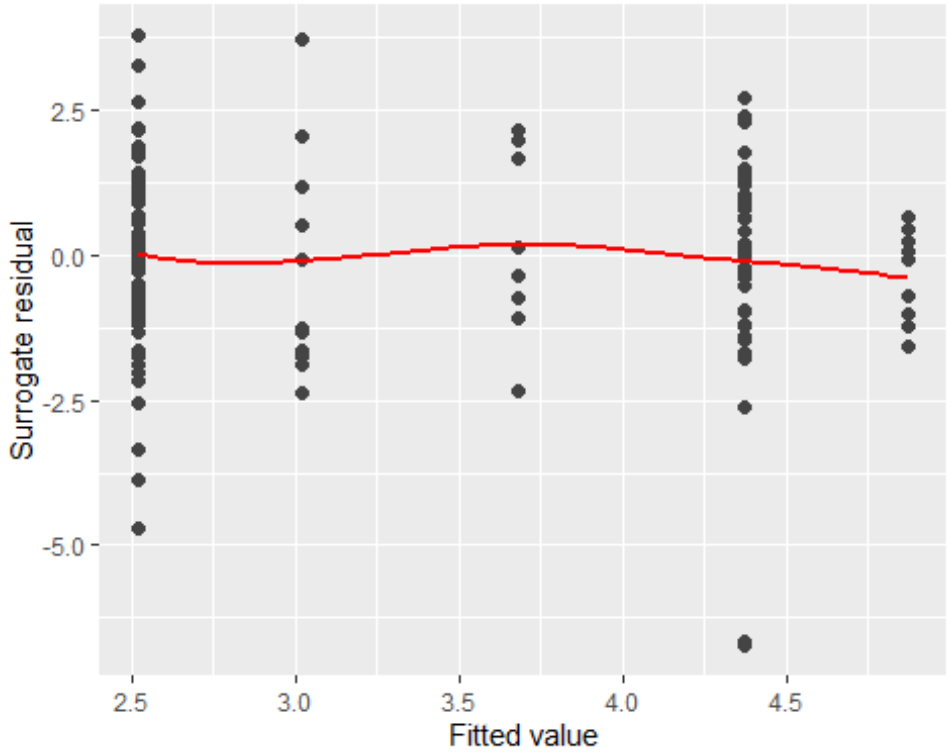
```
##  nobs logLik  niter max.grad cond.H  logLik.Error
##  130  -139.92 6(0)  4.40e-09 3.6e+01 <1e-10
##
##                  Estimate Std.Err  Gradient     Error Cor.Dec Sig.Dig
## 1|2               -2.5219  0.4312  4.40e-09  7.42e-10       8       9
## 2|3                0.6631  0.2466 -2.20e-09 -1.84e-11      10      10
## 3|4                1.5998  0.2830 -1.29e-12 -1.65e-11      10      11
## NewDASNot shared   1.1625  0.6861 -5.46e-11 -1.75e-11      10      11
```

```
## NewDASShared       1.8519  0.3835 -1.05e-10 -1.57e-11      10      11
## Preprint           0.5005  0.4733 -2.18e-10 -7.40e-12      10      10
##
## Eigen values of Hessian:
## 68.197 24.184  6.638  5.094  4.107  1.915
##
## Convergence message from clm:
## (0) successful convergence
## In addition: Absolute and relative convergence criteria were met
```

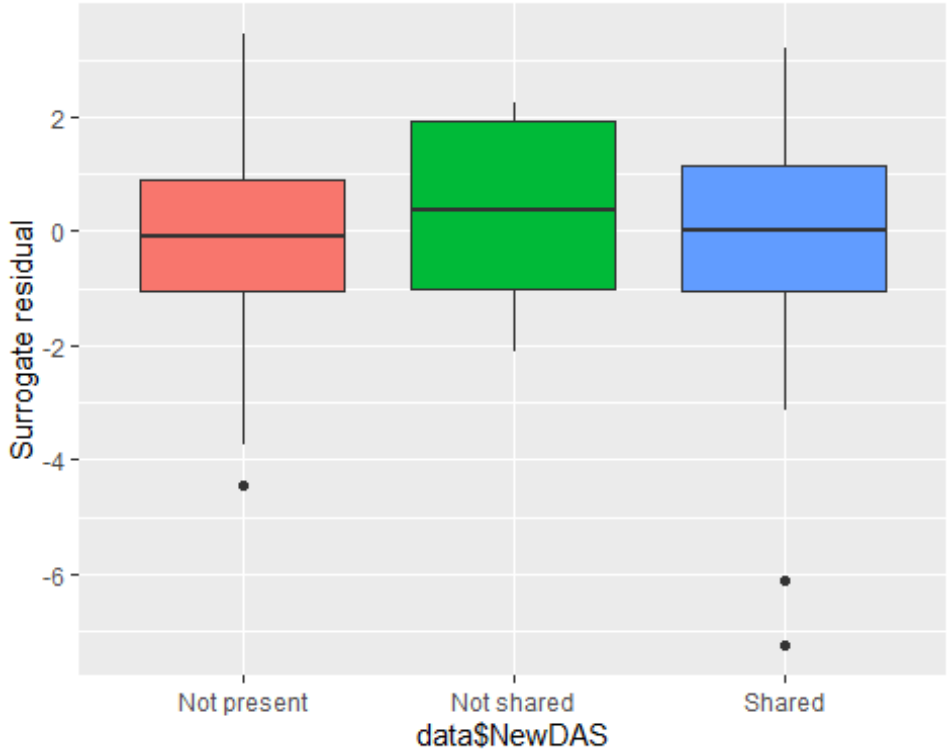
```
######## Graphically validate proportional odds using the sure package ######
#

autoplot.clm(m1a, what = c("qq")) # most of the points fall along the line, s
o no violation of linearity
```




```
autoplot.clm(m1a, what = c("fitted")) # the residuals do not show a clear pat
tern or trend

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
autoplot.clm(m1a, what = c("covariate"), x = data$NewDAS) # scale test indica
ted no variance issues. This residual, covariate plots seems acceptable as we
ll
```



```
autoplot.clm(m1a, what = c("covariate"), x = data$Preprint)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : at -0.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : radius 2.5e-05
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : all data on boundary of neighborhood. make span bigger
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : pseudoinverse used at -0.005
```

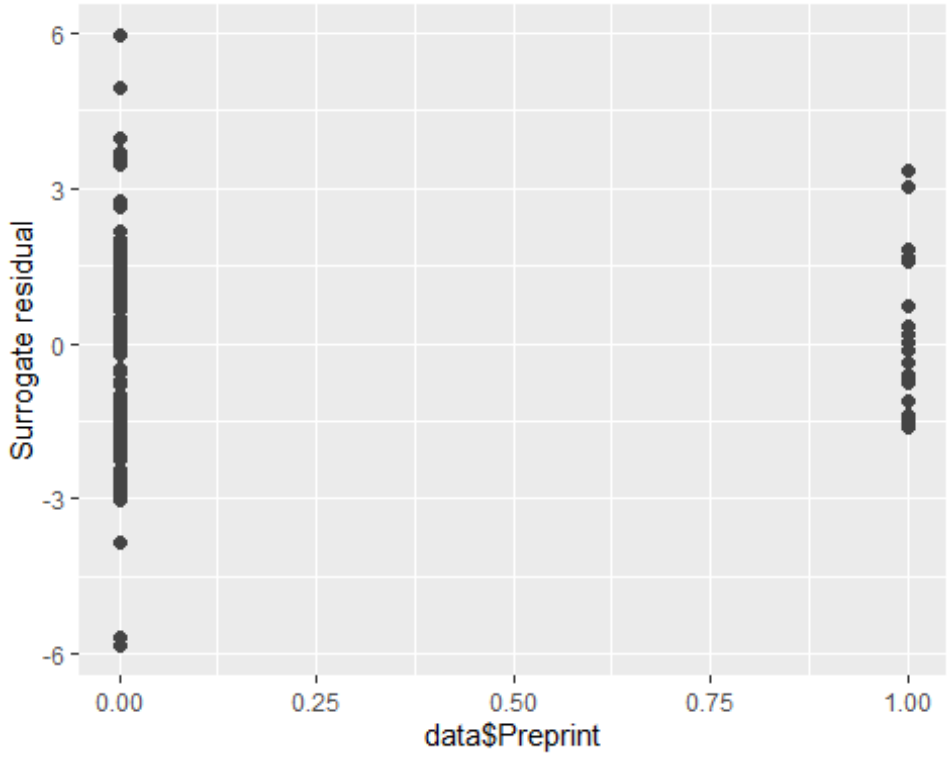
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : neighborhood radius 0.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : reciprocal condition number 1
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : There are other near singularities as well. 1.01
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : zero-width neighborhood. make span bigger
```

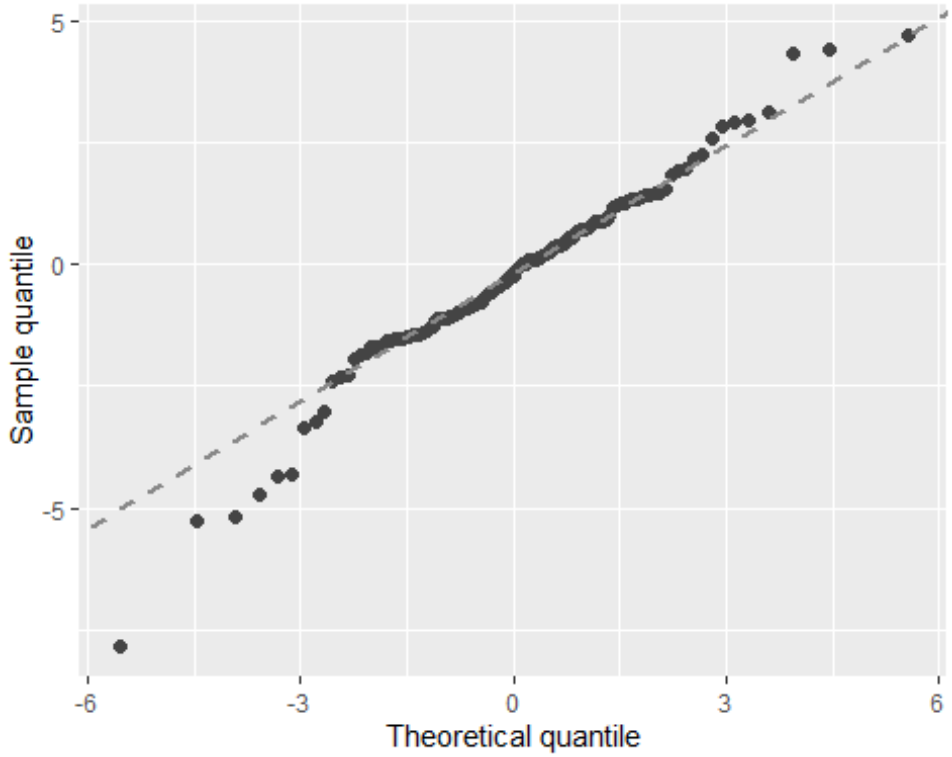
```
## Warning: Failed to fit group -1.
## Caused by error in `predLoess()`:
## ! NA/NaN/Inf in foreign function call (arg 5)
```



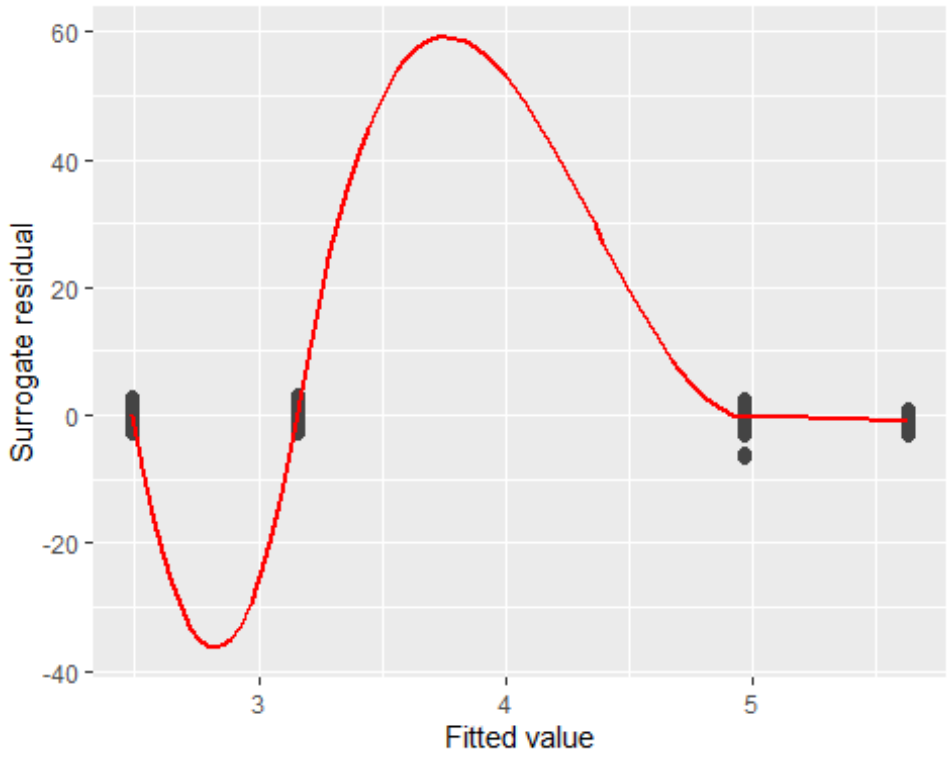

```
# N.B. - Interpreting residual plots is largely subjective!

m1a <- clm(Access ~ NewDAS + Preprint, data = data)
summary_m1a <- summary(m1a)

# Extract coefficients and standard errors
coefs <- summary_m1a$coefficients[, 1]  # Coefficients
se_coefs <- summary_m1a$coefficients[, 2]  # Standard errors

# Extract coefficients and standard errors
coefs <- summary(m1a)$coefficients
OR <- exp(coefs[,1])
lower_ci <- exp(coefs[,1] - 1.96 * coefs[,2])
upper_ci <- exp(coefs[,1] + 1.96 * coefs[,2])

# Calculate z-values and p-values
z_values <- coefs[,1] / coefs[,2]
p_values <- 2 * (1 - pnorm(abs(z_values)))

# Combine everything into a data frame for easy viewing
results <- data.frame(OR = OR, LowerCI = lower_ci, UpperCI = upper_ci, p_valu
e = p_values)
print(results)

##                          OR    LowerCI   UpperCI      p_value
## 1|2              0.08293078 0.03554014  0.193514 8.458039e-09
## 2|3              2.17869399 1.30603472  3.634442 2.857815e-03
```

```
## 3|4                2.37186800 1.41398150  3.978664 1.065660e-03
## NewDASNot shared   1.94745782 0.49961942  7.590962 3.369174e-01
## NewDASShared      11.85644530 4.63730330 30.314018 2.428559e-07
## Preprint           1.94425820 0.64398445  5.869924 2.382481e-01

#Check the Assumptions
nominal_test(m1a) # This is a test of the proportional odds assumption, odds
are proportional in this case.

## Tests of nominal effects
##
## formula: Access ~ NewDAS + Preprint
##          Df   logLik    AIC    LRT  Pr(>Chi)
## <none>      -100.310 212.62
## NewDAS
## Preprint  2  -88.681 193.36 23.258 8.903e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

scale_test(m1a) # scale test checks for equal variance/ scale across year, as
sumptions are not violated here because p = 0.8386. If p is less than 0.05 th
en assumptions are violated

## Tests of scale effects
##
## formula: Access ~ NewDAS + Preprint
##          Df   logLik    AIC    LRT Pr(>Chi)
## <none>      -100.310 212.62
## NewDAS    2  -96.004 208.01 8.6128  0.01348 *
## Preprint  1 -100.232 214.46 0.1562  0.69268
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

convergence(m1a) # This is another way to assess the model

##  nobs logLik  niter max.grad cond.H  logLik.Error
##  130  -100.31 7(2)  6.61e-08 3.2e+02 <1e-10
##
##                  Estimate Std.Err  Gradient     Error Cor.Dec Sig.Dig
## 1|2               -2.4897  0.4323  6.61e-08  1.12e-08       7       8
## 2|3                0.7787  0.2611 -3.06e-08 -3.35e-10       9       9
## 3|4                0.8637  0.2639 -3.75e-11 -3.30e-10       9       9
## NewDASNot shared   0.6665  0.6941 -1.80e-09 -3.29e-10       9       9
## NewDASShared       2.4729  0.4789 -5.62e-10 -3.04e-10       9      10
## Preprint           0.6649  0.5638 -2.73e-09 -2.63e-10       9       9
##
## Eigen values of Hessian:
## 566.901  15.759   5.927   4.135   3.161   1.781
##
## Convergence message from clm:
```

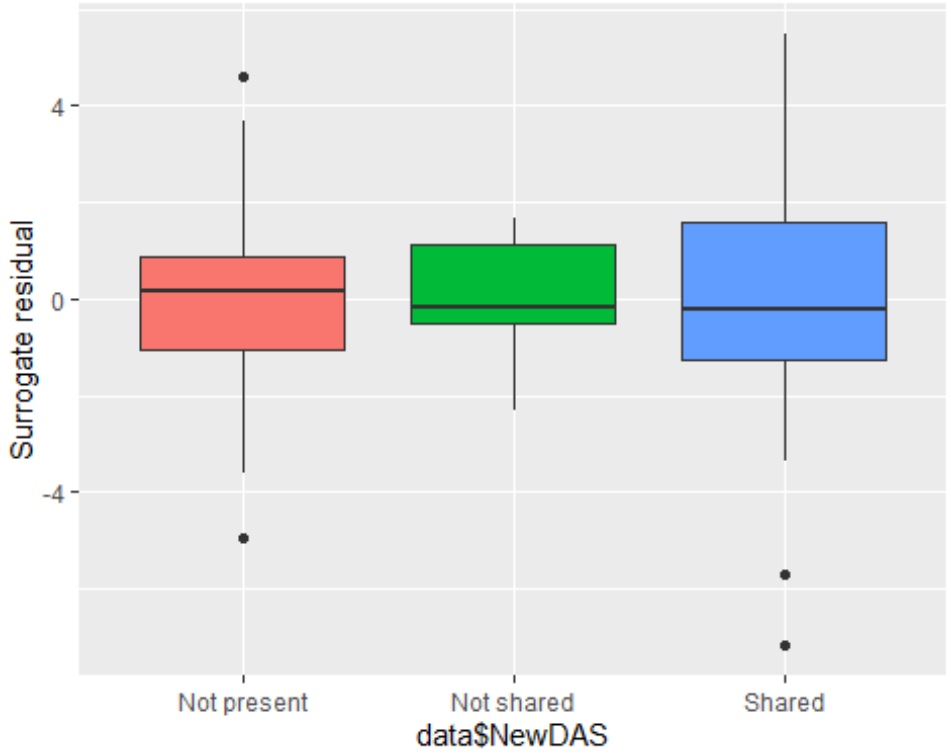
```
## (0) successful convergence
## In addition: Absolute and relative convergence criteria were met

######## Graphically validate proportional odds using the sure package ######
#

autoplot.clm(m1a, what = c("qq")) # most of the points fall along the line, s
o no violation of linearity
```




```
autoplot.clm(m1a, what = c("fitted")) # the residuals do not show a clear pat
tern or trend

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
autoplot.clm(m1a, what = c("covariate"), x = data$NewDAS) # scale test indica
ted no variance issues. This residual, covariate plots seems acceptable as we
ll
```



```
autoplot.clm(m1a, what = c("covariate"), x = data$Preprint)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : at -0.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : radius 2.5e-05
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : all data on boundary of neighborhood. make span bigger
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : pseudoinverse used at -0.005
```

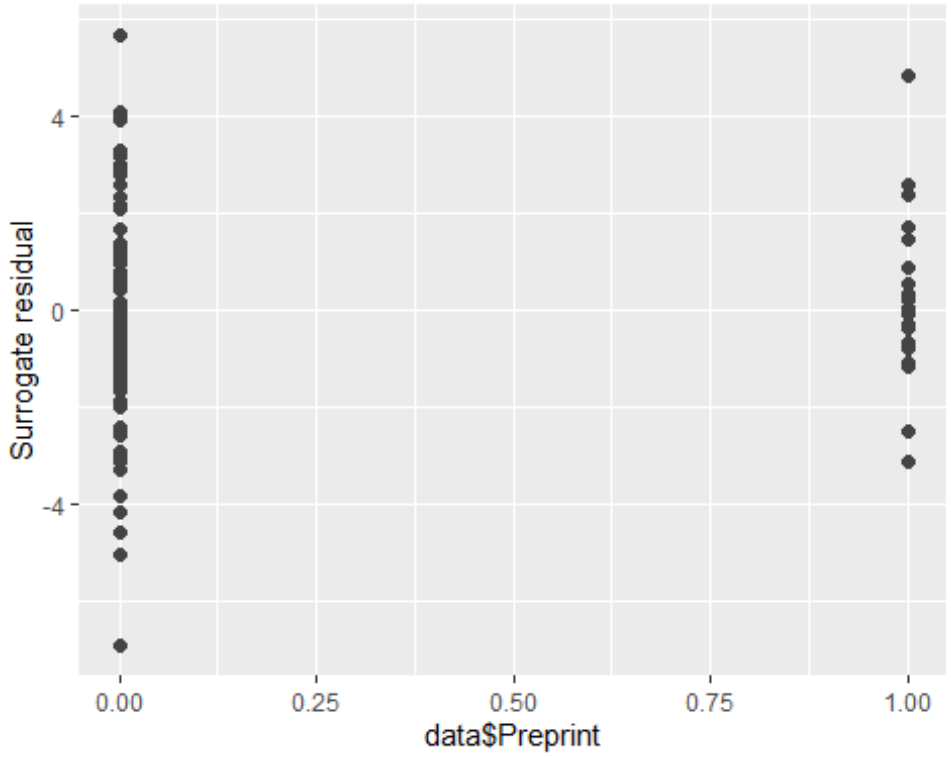
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : neighborhood radius 0.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : reciprocal condition number 1
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : There are other near singularities as well. 1.01
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : zero-width neighborhood. make span bigger
```

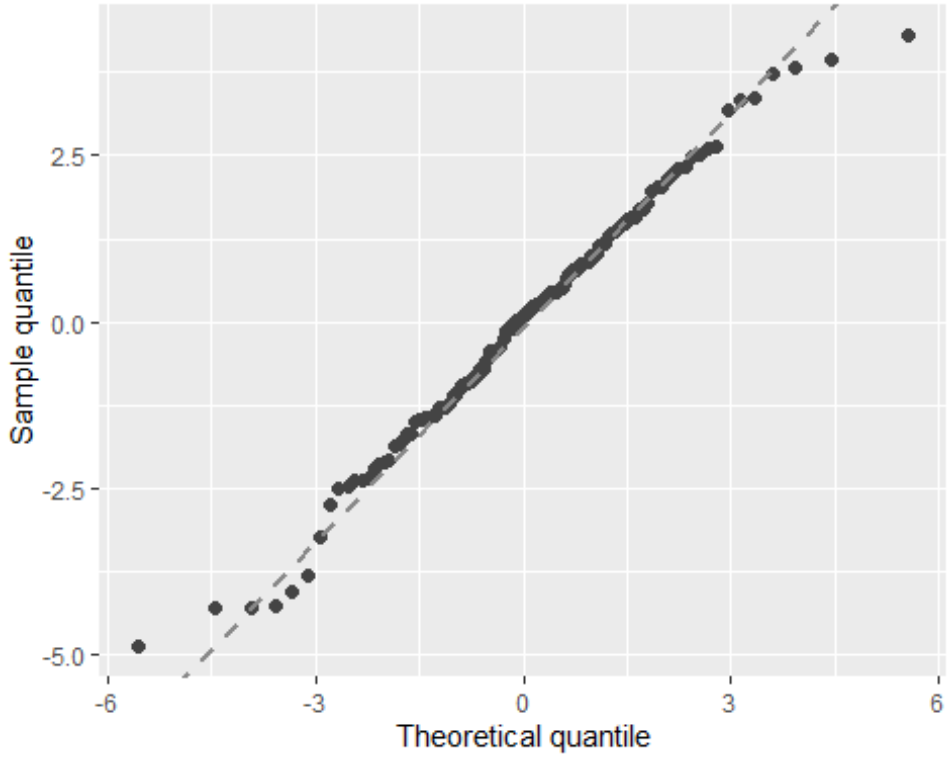
```
## Warning: Failed to fit group -1.
## Caused by error in `predLoess()`:
## ! NA/NaN/Inf in foreign function call (arg 5)
```



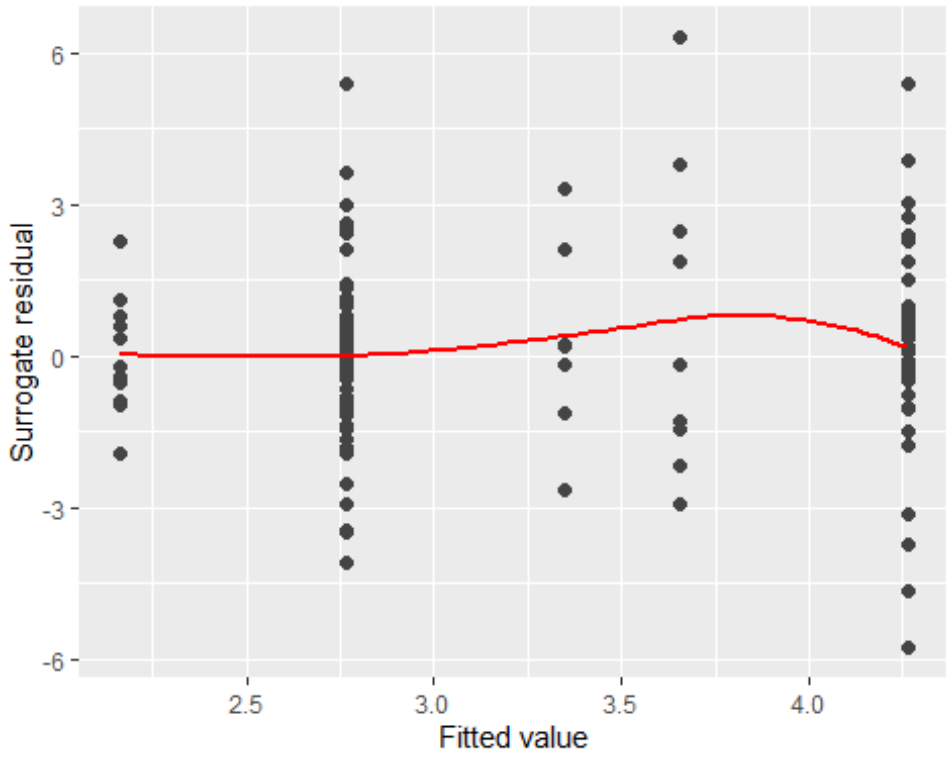

```
# N.B. - Interpreting residual plots is largely subjective!

m1a <- clm(Licence ~ NewDAS + Preprint, data = data)
summary_m1a <- summary(m1a)

# Extract coefficients and standard errors
coefs <- summary_m1a$coefficients[, 1]  # Coefficients
se_coefs <- summary_m1a$coefficients[, 2]  # Standard errors

# Extract coefficients and standard errors
coefs <- summary(m1a)$coefficients
OR <- exp(coefs[,1])
lower_ci <- exp(coefs[,1] - 1.96 * coefs[,2])
upper_ci <- exp(coefs[,1] + 1.96 * coefs[,2])

# Calculate z-values and p-values
z_values <- coefs[,1] / coefs[,2]
p_values <- 2 * (1 - pnorm(abs(z_values)))

# Combine everything into a data frame for easy viewing
results <- data.frame(OR = OR, LowerCI = lower_ci, UpperCI = upper_ci, p_valu
e = p_values)
print(results)

##                          OR    LowerCI    UpperCI      p_value
## 1|3             0.06281708 0.02618925  0.1506720 5.641767e-10
## 3|4             0.56911004 0.34534082  0.9378742 2.699035e-02
```

```
## NewDASNot shared 1.78733704 0.33990304  9.3984853 4.928701e-01
## NewDASShared     4.46125540 1.67179567 11.9050432 2.824748e-03
## Preprint         0.54460152 0.19750155  1.5017138 2.402780e-01

#Check the Assumptions
nominal_test(m1a) # This is a test of the proportional odds assumption, odds
are proportional in this case.

## Tests of nominal effects
##
## formula: Licence ~ NewDAS + Preprint
##          Df  logLik    AIC    LRT Pr(>Chi)
## <none>      -88.207 186.41
## NewDAS    2 -86.914 187.83 2.5854  0.27453
## Preprint  1 -86.238 184.48 3.9372  0.04723 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

scale_test(m1a) # scale test checks for equal variance/ scale across year, as
sumptions are not violated here because p = 0.8386. If p is less than 0.05 th
en assumptions are violated

## Warning: (-1) Model failed to converge with max|grad| = 2.6247e-05 (tol =
1e-06)
## In addition: iteration limit reached

## Tests of scale effects
##
## formula: Licence ~ NewDAS + Preprint
##          Df  logLik    AIC    LRT Pr(>Chi)
## <none>      -88.207 186.41
## NewDAS
## Preprint  1 -87.174 186.35 2.0665   0.1506

convergence(m1a) # This is another way to assess the model

##  nobs logLik niter max.grad cond.H  logLik.Error
##  130  -88.21 5(0)  4.39e-08 2.2e+01 <1e-10
##
##                  Estimate Std.Err  Gradient     Error Cor.Dec Sig.Dig
## 1|3               -2.7675  0.4464  4.39e-08  6.85e-09       7       8
## 3|4               -0.5637  0.2549 -3.38e-08 -1.62e-10       9       9
## NewDASNot shared   0.5807  0.8468 -3.77e-10 -1.08e-10       9       9
## NewDASShared       1.4954  0.5008 -6.23e-10 -2.03e-10       9      10
## Preprint          -0.6077  0.5175 -2.72e-09  2.21e-10       9       9
##
## Eigen values of Hessian:
## 29.924  6.834  3.658  3.311  1.343
##
## Convergence message from clm:
```

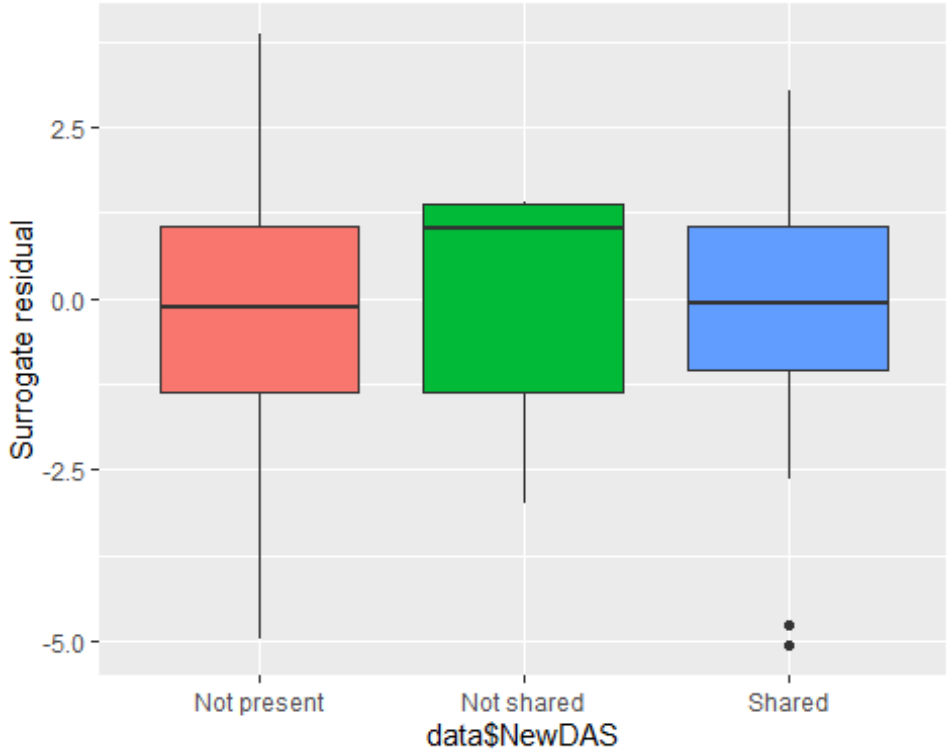
```
## (0) successful convergence
## In addition: Absolute and relative convergence criteria were met

######## Graphically validate proportional odds using the sure package ######
#

autoplot.clm(m1a, what = c("qq")) # most of the points fall along the line, s
o no violation of linearity
```




```
autoplot.clm(m1a, what = c("fitted")) # the residuals do not show a clear pat
tern or trend

## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
autoplot.clm(m1a, what = c("covariate"), x = data$NewDAS) # scale test indica
ted no variance issues. This residual, covariate plots seems acceptable as we
ll
```



```
autoplot.clm(m1a, what = c("covariate"), x = data$Preprint)
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : at -0.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : radius 2.5e-05
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : all data on boundary of neighborhood. make span bigger
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : pseudoinverse used at -0.005
```

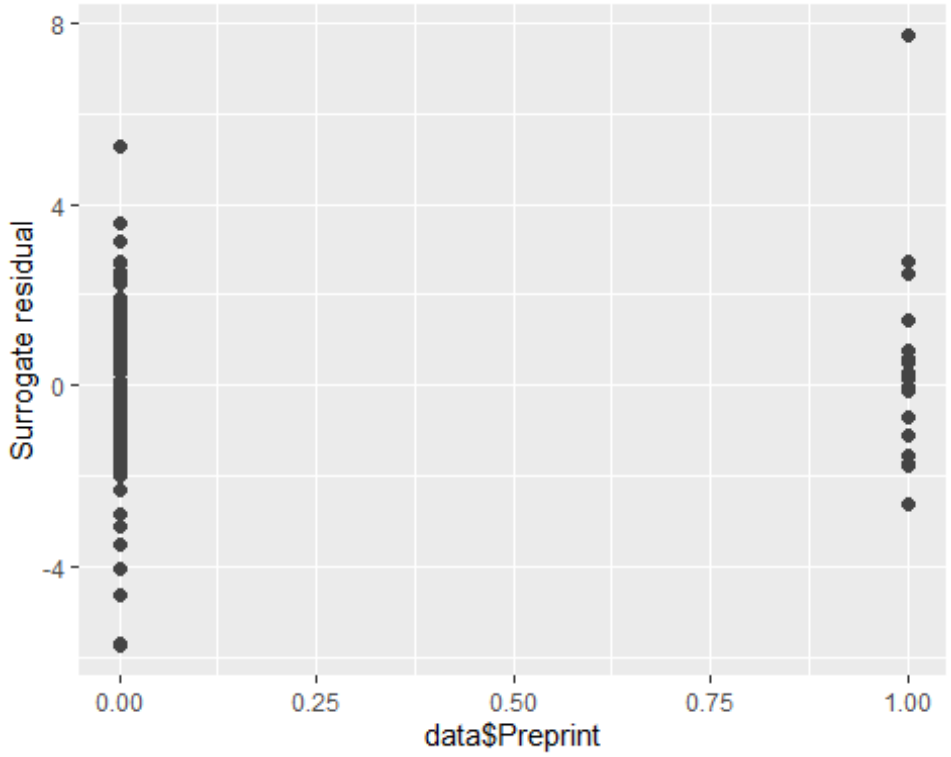
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : neighborhood radius 0.005
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : reciprocal condition number 1
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : There are other near singularities as well. 1.01
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parame
tric,
## : zero-width neighborhood. make span bigger
```

```
## Warning: Failed to fit group -1.
## Caused by error in `predLoess()`:
## ! NA/NaN/Inf in foreign function call (arg 5)
```

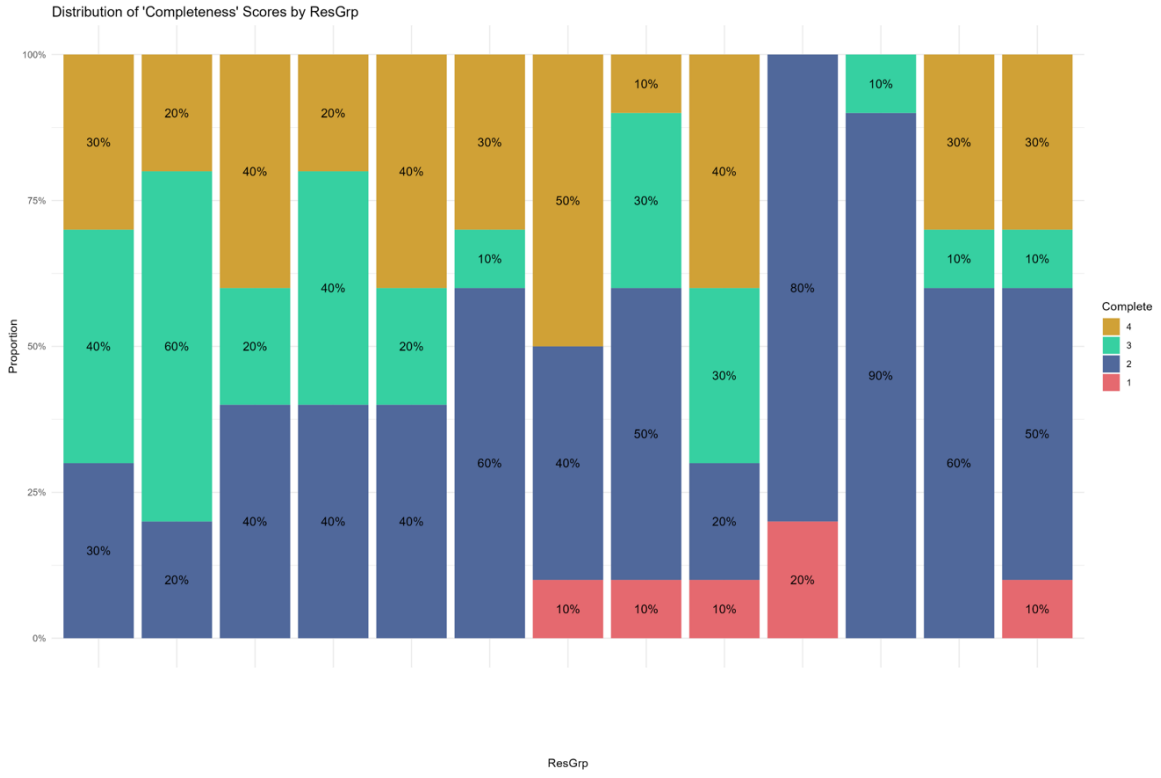



```
# N.B. - Interpreting residual plots is largely subjective
```

#Research Group figures (Internal)

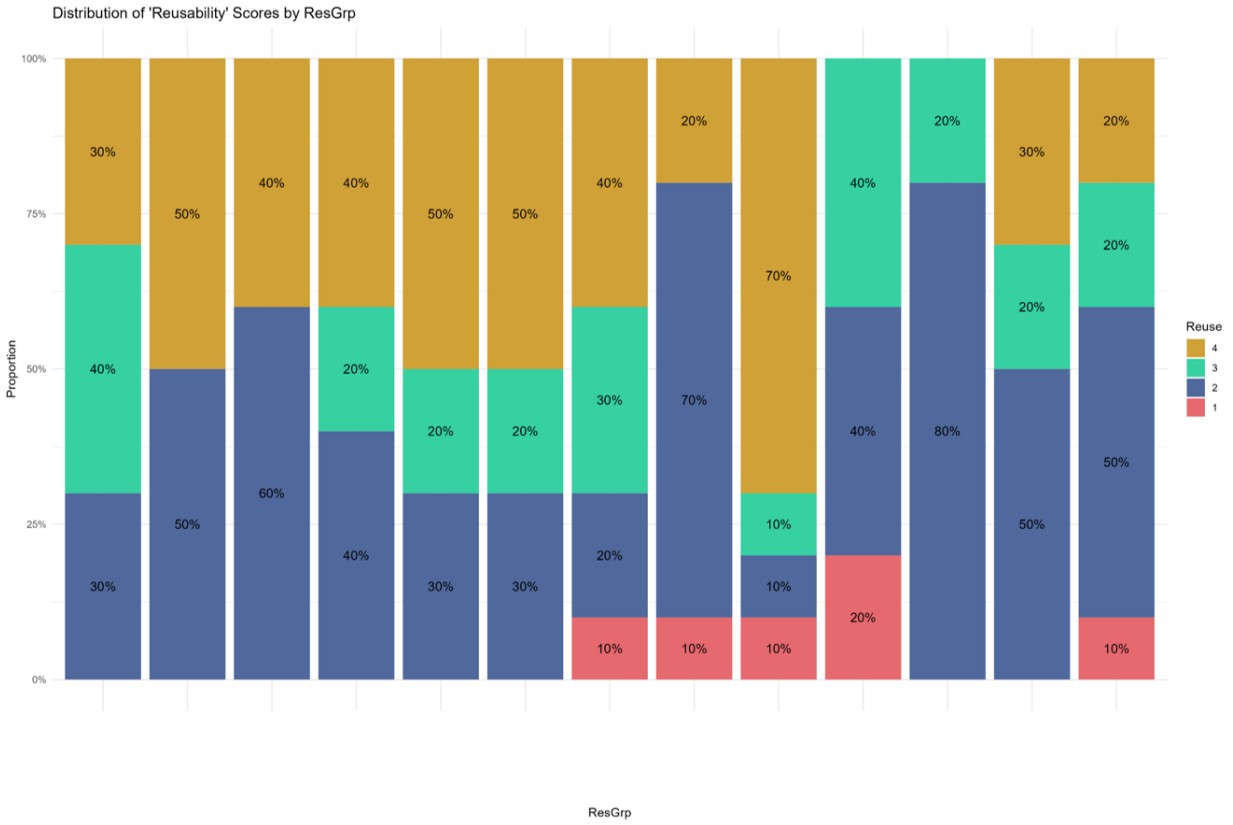
```
# Calculate the frequency and percentage for each 'Complete' score among the
research Group
long_data <- data %>%
  count(ResGrp, Complete) %>%
  group_by(ResGrp) %>%
  mutate(Percentage = n / sum(n) * 100) %>%
  mutate(Label = ifelse(Percentage > 5, paste0(round(Percentage, 1), "%"), ""
)) %>%
  mutate(Cumulative_Percentage = cumsum(Percentage) - (0.5 * Percentage)) %>%
  ungroup()

long_data <- long_data %>%
  mutate(Complete = factor(Complete, levels = c("4", "3", "2", "1")))

# Custom colors
colors <- c("#D0A136", "#36D0A1","#50689B","#E5696F")

# Creating the stacked bar chart with percentage labels
internal1 <- ggplot(long_data, aes(x = as.factor(ResGrp), y = n, fill = as.fa
ctor(Complete))) +
  geom_bar(stat = "identity", position = "fill") +  # Scale the bar heights t
o proportions
  scale_y_continuous(labels = percent_format()) +  # Convert the y-axis to pe
```

```
rcentage
  geom_text(
    aes(label = Label, y = Percentage),
    size = 4,
    color = "black",
    position = position_fill(vjust = 0.5)
  ) +
  scale_fill_manual(values = colors) +
  labs(title = "Distribution of 'Completeness' Scores by ResGrp",
       x = "ResGrp",
       y = "Proportion",
       fill = "Complete") +
  theme_minimal() +
  scale_x_discrete(name = "ResGrp", labels = unique(long_data$ResGrp)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))  # Adjust text ang
le for readability

# Print the plot
print(internal1)
```



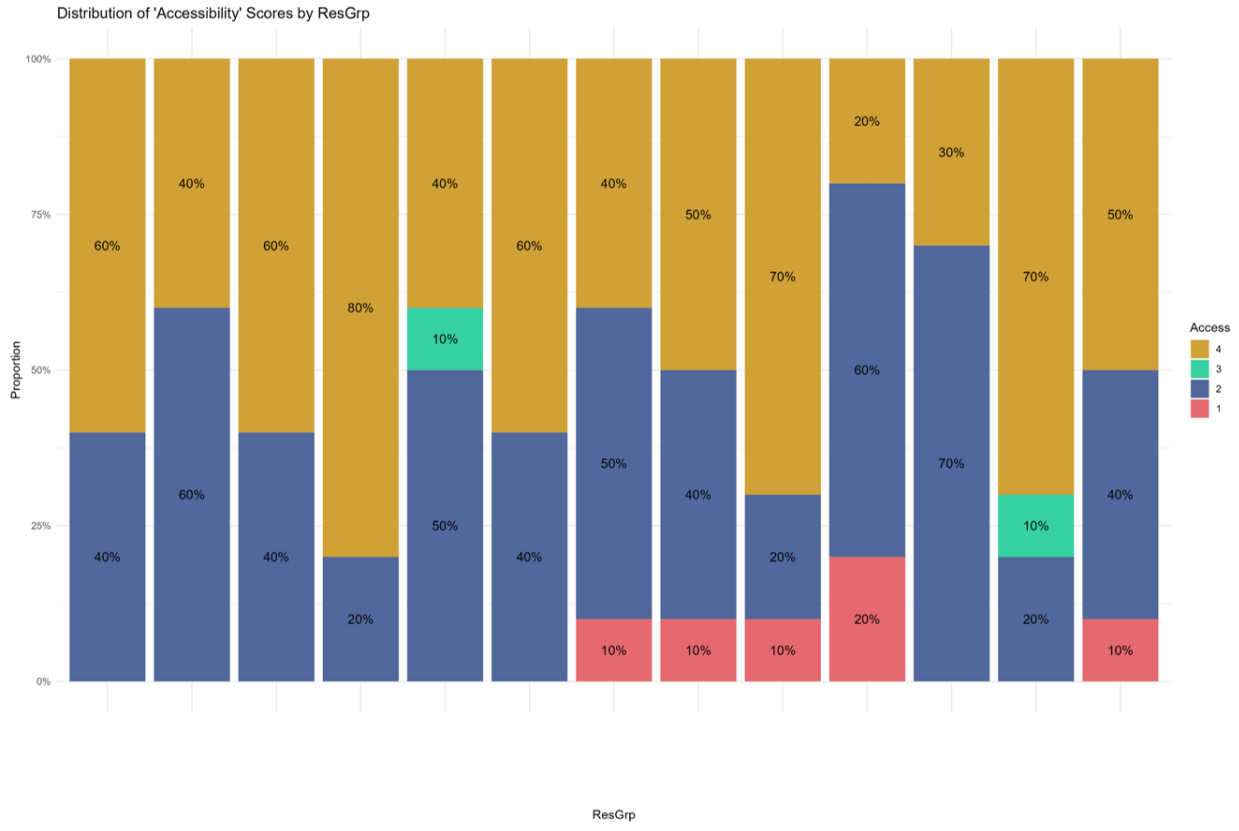

```
ggsave("internal1.png", internal1, width = 15, height = 10, units = "in", bg=
"white")


#these results were used for internal university meetings and presentations
```

```r
# Calculate the frequency and percentage for each 'Reuse' score among the res
earch Group
long_data <- data %>%
  count(ResGrp, Reuse) %>%
  group_by(ResGrp) %>%
  mutate(Percentage = n / sum(n) * 100) %>%
  mutate(Label = ifelse(Percentage > 5, paste0(round(Percentage, 1), "%"), ""
)) %>%
  mutate(Cumulative_Percentage = cumsum(Percentage) - (0.5 * Percentage)) %>%
  ungroup()

long_data <- long_data %>%
  mutate(Reuse = factor(Reuse, levels = c("4", "3", "2", "1")))

# Custom colors
colors <- c("#D0A136", "#36D0A1","#50689B","#E5696F")

# Creating the stacked bar chart with percentage labels
internal2 <- ggplot(long_data, aes(x = as.factor(ResGrp), y = n, fill = as.fa
ctor(Reuse))) +
  geom_bar(stat = "identity", position = "fill") +  # Scale the bar heights t
o proportions
  scale_y_continuous(labels = percent_format()) +  # Convert the y-axis to pe
rcentage
  geom_text(
    aes(label = Label, y = Percentage),
    size = 4,
    color = "black",
    position = position_fill(vjust = 0.5)
  ) +
  scale_fill_manual(values = colors) +
  labs(title = "Distribution of 'Reusability' Scores by ResGrp",
       x = "ResGrp",
       y = "Proportion",
       fill = "Reuse") +
  theme_minimal() +
  scale_x_discrete(name = "ResGrp", labels = unique(long_data$ResGrp)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))  # Adjust text ang
le for readability

# Print the plot
print(internal2)
```



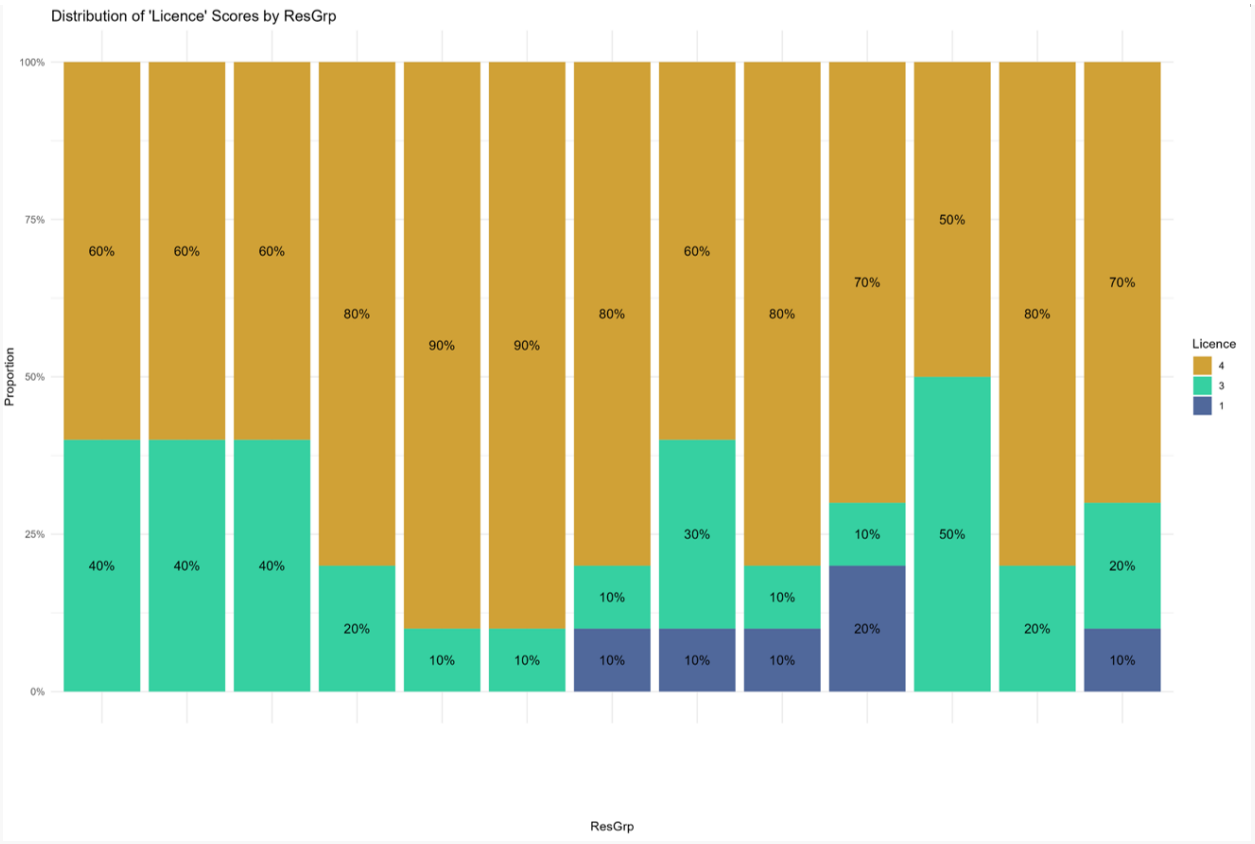

```
ggsave("internal2.png", internal2, width = 15, height = 10, units = "in", bg=
"white")

#these results were used for internal university meetings and presentations

# Calculate the frequency and percentage for each 'Access' score among the re
search Group
long_data <- data %>%
  count(ResGrp, Access) %>%
  group_by(ResGrp) %>%
  mutate(Percentage = n / sum(n) * 100) %>%
  mutate(Label = ifelse(Percentage > 5, paste0(round(Percentage, 1), "%"), ""
)) %>%
  mutate(Cumulative_Percentage = cumsum(Percentage) - (0.5 * Percentage)) %>%
  ungroup()

long_data <- long_data %>%
  mutate(Access = factor(Access, levels = c("4", "3", "2", "1")))

# Custom colors
colors <- c("#D0A136", "#36D0A1","#50689B","#E5696F")

# Creating the stacked bar chart with percentage labels
internal3 <- ggplot(long_data, aes(x = as.factor(ResGrp), y = n, fill = as.fa
ctor(Access))) +
```

```r
  geom_bar(stat = "identity", position = "fill") +  # Scale the bar heights t
o proportions
  scale_y_continuous(labels = percent_format()) +  # Convert the y-axis to pe
rcentage
  geom_text(
    aes(label = Label, y = Percentage),
    size = 4,
    color = "black",
    position = position_fill(vjust = 0.5)
  ) +
  scale_fill_manual(values = colors) +
  labs(title = "Distribution of 'Accessibility' Scores by ResGrp",
       x = "ResGrp",
       y = "Proportion",
       fill = "Access") +
  theme_minimal() +
  scale_x_discrete(name = "ResGrp", labels = unique(long_data$ResGrp)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))  # Adjust text ang
le for readability

# Print the plot
print(internal3)
```




```r
ggsave("internal3.png", internal3, width = 15, height = 10, units = "in", bg=
"white")
```

```r
#these results were used for internal university meetings and presentations

# Calculate the frequency and percentage for each 'Licence' score among the r
esearch Group
long_data <- data %>%
  count(ResGrp, Licence) %>%
  group_by(ResGrp) %>%
  mutate(Percentage = n / sum(n) * 100) %>%
  mutate(Label = ifelse(Percentage > 5, paste0(round(Percentage, 1), "%"), ""
)) %>%
  mutate(Cumulative_Percentage = cumsum(Percentage) - (0.5 * Percentage)) %>%
  ungroup()


long_data <- long_data %>%
  mutate(Licence = factor(Licence, levels = c("4", "3", "2", "1")))

# Custom colors
colors <- c("#D0A136", "#36D0A1","#50689B","#E5696F")

# Creating the stacked bar chart with percentage labels
internal4 <- ggplot(long_data, aes(x = as.factor(ResGrp), y = n, fill = as.fa
ctor(Licence))) +
  geom_bar(stat = "identity", position = "fill") +  # Scale the bar heights t
o proportions
  scale_y_continuous(labels = percent_format()) +  # Convert the y-axis to pe
rcentage
  geom_text(
    aes(label = Label, y = Percentage),
    size = 4,
    color = "black",
    position = position_fill(vjust = 0.5)
  ) +
  scale_fill_manual(values = colors) +
  labs(title = "Distribution of 'Licence' Scores by ResGrp",
       x = "ResGrp",
       y = "Proportion",
       fill = "Licence") +
  theme_minimal() +
  scale_x_discrete(name = "ResGrp", labels = unique(long_data$ResGrp)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))  # Adjust text ang
le for readability

# Print the plot
```




```
print(internal4)
```