Visualising Trends in Apps on the Google Play Store

Lewis Higgins - Student ID 22133848

CMP5352 - Data Visualisation

Word count: XXXX

Abstract

This report delves into a dataset containing information on various apps from the Google Play Store. By leveraging Exploratory Data Analysis (EDA) and data visualization techniques, the report aims to uncover insights and trends within the Play Store ecosystem based on data such as review scores, install counts and app categories.

Contents

Introduction

Data visualisation is a field of data science wherein large datasets are parsed using code (most commonly written in Python or R) to produce clear visualisations interpretable to a wide audience, even if they do not have in-depth knowledge of the dataset.

This report specifically aims to produce visualisations based on an analysis of a large dataset based on the content available on the Google Play Store, where users of Android mobile devices can download a wide variety of apps to suit their needs. The mobile app industry is an enormous market to be involved in, with over 27 billion downloads occurring on the Play Store in Q1 2023 alone (Ceci, 2023a). This report delves into a comprehensive dataset of apps available on the Play store, aiming to uncover valuable insights and trends. Through exploratory data analysis, the composition of the store will be examined, including factors such as app reviews in comparison to their install counts, as well as additional insights such as app pricing and how it affects user counts. This exploration will also provide potential indicators of current and future trends within the Play Store and the broader mobile market.

This report is split across three sections:

- The **motivation and objectives** of this report.
- The **results from experiments** on the dataset.
- A **summary** of overall findings.

This report was compiled as a .Rnw file, meaning that all R code used in this report is embedded into the PDF and readable within the report.

Motivation and objectives

The Google Play Store is a massive market to tap into, hosting over 3.5 million apps as of Q3 2022 (Ceci, 2023b), so it is therefore crucial to understnad current and future trends to ensure that content released on the store can be properly optimised to maximise install count and review scores, which will ultimately increase profit.

The dataset in use is sourced from Kaggle, a public dataset-sharing website. It contains 9660 unique values with 13 columns of data:

- App - The name of the app.
- Category - The specific category of the app.
- Rating - A number between 1 and 5, representing the "star" rating of the app.
- Reviews - The amount of reviews for the app.
- Size - The size of the app in kilobytes or megabytes.
- Installs - Number of user installs of the app.
- Type - Whether the app costs money or is free.
- Price - The price of the app in dollars, or 0 if it is free.
- Content.Rating - The general age rating of the app.
- Genres - The genres associated with the app.
- Last.Updated - The most recent date when the app received an update.
- Current.Ver - The version number of the app.
- Android.Ver - The required Android version of the app.

1.1 Key questions and expected answers conerning the data

- Is there a positive correlation between the average review score of an app and its install count?
  - Expected: Yes, as it is harder to weigh the average down when the volume of installs increases.
- Which category of app has the highest review score on average?
  - Expected: Games are likely to have the highest review scores from users enjoying their experiences.
- What is the distribution of prices? Are there certain prices used by many apps?
  - Expected: If an app is not free, it would likely be under $3 to ensure people buy it.
- Do paid apps receive higher or lower review score on average?
  - Expected: No, as users may be more harsh with their reviews if they paid for the app.
- Is there a correlation between an app's content rating and it's review score?
  - Expected: Yes, with adult apps likely being higher rated because children would be unlikely to have accounts.
- Does app size play a role in install count?
  - Expected: Yes, as users would likely not want to download very large apps due to internet speeds and device storage.

Experimental results

2.1 Data wrangling

To be able to analyse the data, it is essential that the data is first in a state that can be analysed effectively. To do so, the data must be explored to identify any potential issues, and any that are found must be cleaned, which is referred to as data wrangling. This process is vital to ensure that any products of this data are accurate, correct and not misleading to viewers as a result of incomplete or inconsistent data.

Another good step to take in the initial exploration of the data is to identify if there are any missing values in certain columns. To do so, we can convert any instance of a blank string into R's recognised "NA" type and then use is.na() to identify how many there are.

```r
# Convert blank strings or NaN to NA.
dataDf[(dataDf == "" | dataDf == "NaN")] <- NA

# Identify rows containing NA.
naRows <- dataDf[rowSums(is.na(dataDf) > 0),]
nrow(naRows)

[1] 1481

nrow(naRows)/nrow(dataDf)*100

[1] 13.66
```

We can identify that this dataset has 1481 rows where at least one column has missing data, equating to 13.66% of the dataset. To analyse this in further detail to see specifically which columns contain missing data, the naniar package can be used to generate a tibble of where the data is missing.

```r
kable(miss_var_summary(dataDf),
     caption = "Missing variable summary of the dataset", "latex") %>%
kable_styling(latex_options = "HOLD_position")
```

Table 2.1: Missing variable summary of the dataset

variable	n_miss	pct_miss
Rating	1474	13.6
Current.Ver	8	0.0738
Android.Ver	3	0.0277
Type	1	0.00922
Content.Rating	1	0.00922
App	0	0
Category	0	0
Reviews	0	0
Size	0	0
Installs	0	0
Price	0	0
Genres	0	0
Last.Updated	0	0

```
str(dataDf, width = 80, strict.width = "cut")

'data.frame': 10841 obs. of  13 variables:
 $ App           : chr  "Photo Editor & Candy Camera & Grid & ScrapBook" "Col"..
 $ Category      : chr  "ART_AND_DESIGN" "ART_AND_DESIGN" "ART_AND_DESIGN" "A"..
 $ Rating        : num  4.1 3.9 4.7 4.5 4.3 4.4 3.8 4.1 4.4 4.7 ...
 $ Reviews       : chr  "159" "967" "87510" "215644" ...
 $ Size          : chr  "19M" "14M" "8.7M" "25M" ...
 $ Installs      : chr  "10,000+" "500,000+" "5,000,000+" "50,000,000+" ...
 $ Type          : chr  "Free" "Free" "Free" "Free" ...
 $ Price         : chr  "0" "0" "0" "0" ...
 $ Content.Rating: chr  "Everyone" "Everyone" "Everyone" "Teen" ...
 $ Genres        : chr  "Art & Design" "Art & Design;Pretend Play" "Art & Des"..
 $ Last.Updated  : chr  "January 7, 2018" "January 15, 2018" "August 1, 2018""..
 $ Current.Ver   : chr  "1.0.0" "2.0.0" "1.2.4" "Varies with device" ...
 $ Android.Ver   : chr  "4.0.3 and up" "4.0.3 and up" "4.0.3 and up" "4.2 and"..
```

From this brief summary, we can clearly identify certain categorical data that should be treated as factors in the dataset, such as:

- Installs - Uses thresholds for amounts of installs.
- Category - An app has one category.
- Type - An app is free or paid.
- Price - Most apps have specific prices shared by other apps.
- Content.Rating - Has categories of age ratings.
- Android.Ver - Categorised by the Android version.

After converting them and reproducing the summary, the snippet of the data is even clearer.

```
cols <- c("Installs","Category","Type","Price","Content.Rating","Android.Ver")
dataDf <- dataDf %>% mutate(across(all_of(cols), as.factor))

# Also convert Last.Updated to a date.
dataDf$Last.Updated <- mdy(dataDf$Last.Updated)

str(dataDf, width = 80, strict.width = "cut")

'data.frame': 10841 obs. of  13 variables:
 $ App           : chr  "Photo Editor & Candy Camera & Grid & ScrapBook" "Col"..
 $ Category      : Factor w/ 34 levels "1.9","ART_AND_DESIGN",..: 2 2 2 2 2 2 ..
 $ Rating        : num  4.1 3.9 4.7 4.5 4.3 4.4 3.8 4.1 4.4 4.7 ...
 $ Reviews       : chr  "159" "967" "87510" "215644" ...
 $ Size          : chr  "19M" "14M" "8.7M" "25M" ...
 $ Installs      : Factor w/ 22 levels "0","0+","1,000,000,000+",..: 8 20 13 1..
 $ Type          : Factor w/ 3 levels "0","Free","Paid": 2 2 2 2 2 2 2 2 2 2 ...
 $ Price         : Factor w/ 93 levels "$0.99","$1.00",..: 92 92 92 92 92 92 9..
 $ Content.Rating: Factor w/ 6 levels "Adults only 18+",..: 2 2 2 5 2 2 2 2 2 ..
 $ Genres        : chr  "Art & Design" "Art & Design;Pretend Play" "Art & Des"..
 $ Last.Updated  : Date, format: "2018-01-07" "2018-01-15" ...
 $ Current.Ver   : chr  "1.0.0" "2.0.0" "1.2.4" "Varies with device" ...
 $ Android.Ver   : Factor w/ 33 levels "1.0 and up","1.5 and up",..: 16 16 16 ..
```

However, this new snippet of the data reveals that there are unexpected values in both the category and type columns, with a number being present in category and in type, which is strongly inconsistent. Additionally, by viewing the factor levels of Android.Ver, we can see that some apps appear to have a maximum version.

```
levels(dataDf$Android.Ver)

 [1] "1.0 and up"         "1.5 and up"         "1.6 and up"
 [4] "2.0 and up"         "2.0.1 and up"       "2.1 and up"
 [7] "2.2 - 7.1.1"        "2.2 and up"         "2.3 and up"
[10] "2.3.3 and up"       "3.0 and up"         "3.1 and up"
[13] "3.2 and up"         "4.0 and up"         "4.0.3 - 7.1.1"
[16] "4.0.3 and up"       "4.1 - 7.1.1"        "4.1 and up"
[19] "4.2 and up"         "4.3 and up"         "4.4 and up"
[22] "4.4W and up"        "5.0 - 6.0"          "5.0 - 7.1.1"
[25] "5.0 - 8.0"          "5.0 and up"         "5.1 and up"
[28] "6.0 and up"         "7.0 - 7.1.1"        "7.0 and up"
[31] "7.1 and up"         "8.0 and up"         "Varies with device"

nrow(dataDf)

[1] 10841

# Remove the rows that were detected as inconsistent.
# Filter to where conditions are NOT met.
dataDf <- dataDf %>%
    filter(!(Category == "1.9" | Type == "0" | str_detect(Android.Ver,
        "[-]+")))

# Removed 13 rows.
nrow(dataDf)

[1] 10828

# Drop the factor levels that no longer exist as a result
# of that row's removal.
dataDf <- droplevels(dataDf)

nrow(dataDf) - length(unique(dataDf$App))

[1] 1181
```

We can also observe that the dataset has 10,828 rows without these inconsistent rows. This is 1181 more rows that there are unique values, which suggests there are a considerable amount of duplicate rows. While we clean these duplicates, we can also remove data that won't be relevant to our analyses, as well.

```
# Immediately identifies 483 completely duplicated rows
# which will be removed.
dataDf <- dataDf[!duplicated(dataDf),]

# Only keep apps with more than 0 reviews, as 0 review apps are
# worthless to our analysis. Then, group all the apps with the
# same name, leaving only the one with the MOST REVIEWS,
# rather than the one that occurs first in the dataset.
dataDf <- dataDf %>%
    group_by(App) %>%     # If 2 occurrences, keep one with highest reviews.
    filter(Reviews > 0 & Reviews == max(Reviews)) %>%
    ungroup() # Ungroup for cleaner output later.

# 9061, meaning the data is much cleaner.
nrow(dataDf)

[1] 9061
```

Now that the data has been cleared of duplicate and irrelevant values, we can analyse why some data may be missing.

```
kable(miss_var_summary(dataDf),
      caption = "Missing variable summary of the modified dataset", "latex") %>%
kable_styling(latex_options = "HOLD_position")
```

Table 2.2: Missing variable summary of the modified dataset

variable	n_miss	pct_miss
Rating	868	9.58
Current.Ver	7	0.0773
App	0	0
Category	0	0
Reviews	0	0
Size	0	0
Installs	0	0
Type	0	0
Price	0	0
Content.Rating	0	0
Genres	0	0
Last.Updated	0	0
Android.Ver	0	0

The most commonly missing value according to Table 2.1 is that of the rating. However, it is not possible for us to impute this data as user reviews cannot be entirely predicted as there are many more factors than those available in this dataset involved in user rating, so rows with missing ratings must be removed.

```
noRatings <- dataDf %>%
    filter(is.na(Rating))

nrow(noRatings)

[1] 868

# Because there is no way to source non-existent rating
# data, these will have to be removed.
dataDf <- dataDf %>%
    filter(!is.na(Rating))

nrow(dataDf)

[1] 8193

naRows <- dataDf[rowSums(is.na(dataDf) > 0), ]
nrow(naRows)

[1] 4

# Drop the factor levels that no longer exist as a result
# of this.
dataDf <- droplevels(dataDf)
```

Four NA values remain and also six more apps with duplicated names but different categories, though due to how small of a proportion these make up of the data, these 12 rows can be safely removed.

```
dataDf <- dataDf[complete.cases(dataDf), ]
nrow(dataDf)

[1] 8189

# 6 apps still remain with duplicated names but different categories, but due
# to the small proportion, we will remove them and keep the first occurrence.
dataDf <- subset(dataDf, !duplicated(App))
nrow(dataDf)

[1] 8183

str(dataDf, width = 80, strict.width = "cut")

tibble [8,183 x 13] (S3: tbl_df/tbl/data.frame)
 $ App           : chr [1:8183] "Photo Editor & Candy Camera & Grid & ScrapBo"..
 $ Category      : Factor w/ 33 levels "ART_AND_DESIGN",..: 1 1 1 1 1 1 1 1 1 ..
 $ Rating        : num [1:8183] 4.1 4.7 4.5 4.3 4.4 3.8 4.1 4.4 4.7 4.4 ...
 $ Reviews       : chr [1:8183] "159" "87510" "215644" "967" ...
 $ Size          : chr [1:8183] "19M" "8.7M" "25M" "2.8M" ...
 $ Installs      : Factor w/ 19 levels "1,000,000,000+",..: 6 11 14 9 15 15 2 ..
 $ Type          : Factor w/ 2 levels "Free","Paid": 1 1 1 1 1 1 1 1 1 1 ...
 $ Price         : Factor w/ 73 levels "$0.99","$1.00",..: 73 73 73 73 73 73 7..
 $ Content.Rating: Factor w/ 6 levels "Adults only 18+",..: 2 2 5 2 2 2 2 2 2 ..
 $ Genres        : chr [1:8183] "Art & Design" "Art & Design" "Art & Design" "..
 $ Last.Updated  : Date[1:8183], format: "2018-01-07" "2018-08-01" ...
 $ Current.Ver   : chr [1:8183] "1.0.0" "1.2.4" "Varies with device" "1.1" ...
 $ Android.Ver   : Factor w/ 26 levels "1.0 and up","1.5 and up",..: 14 14 16 ..
```

Furthermore, the "Current.Ver" column is not useful to our analysis, as version numbers are not relevant in comparison to when the app was last updated, which is a much more useful data source that also exists in the dataset. Additionally, version numbers are entirely at the developer's discretion and follow no standardisation, meaning they do not reveal anything to us. The "Genres" column also often repeats the Category column, which specialises apps much more. Therefore, these columns will be dropped.

```
dataDf <- subset(dataDf, select = -c(Current.Ver, Genres))
dim(dataDf) # Now 11 columns instead of 13.

[1] 8183   11
```

Now that we are working with a cleaner dataset free of duplicate and inconsistent data, we can begin to perform Exploratory Data Analysis.

2.2 Exploratory Data Analysis

2.2.1 Feature engineering

To produce some of the upcoming visualisations in both this section of the report and Section 2.3, we will need to add some additional columns based on the data we have, as well as modifying existing column data types so that comparisons can be performed on them.

```r
# Function to convert size values to numeric and set
# "Varies with device" rows to 0.
convertSize <- function(size) {
    if (size == "Varies with device") {
        # Make it 0.
        return(0)
    } else if (grepl("M", size)) {
        # Remove the M, make it numeric.
        return(as.numeric(sub("M", "", size)))
    } else if (grepl("k", size)) {
        # Remove the k, convert it to MB by dividing it, then make it numeric.
        return(as.numeric(sub("k", "", size)) / 1024)
    }
}

dataDf <- dataDf %>%
# Make an integer column for Installs by removing commas and pluses.
mutate(intInstalls = as.integer(gsub("[,+]+", "", dataDf$Installs))) %>%
# Make Category sentence case and replace underscores with
# spaces for better visualisations.
mutate(Category = gsub("[_]+", " ", str_to_sentence(dataDf$Category))) %>%
# Make a numeric column for Size using the function defined earlier.
mutate(numSize = sapply(Size, convertSize))

# Make a categorical factor column for size, grouping
# numerical sizes into thresholds.
dataDf$sizeThresholds <- cut(dataDf$numSize,
# Because all "Varies with device" sizes were set to 0, it
# allows for a label to be set especially for them.
# By being inbetween -1 and 0.001, they meet the first break.
breaks = c(-1, 0.001, 1, 5, 10, 25, 50, 75, 100, 150, Inf),
labels = c("Varies with device", "Under 1MB", "1 - 5MB", "5 - 10MB",
            "10 - 25MB", "25 - 50MB", "50 - 75MB",
            "75 - 100MB", "100 - 150MB", "150MB+"))


nrow(filter(dataDf, Content.Rating == "Adults only 18+"))

[1] 3

# Because there's only three of these, and there is already a "Mature 17+"
# category, we can upgrade all Mature apps to 18+ instead, as it is still
# technically true.
dataDf <- dataDf %>%
mutate(Content.Rating = if_else(Content.Rating == "Mature 17+",
                        "Adults only 18+", Content.Rating))
```

```
nrow(filter(dataDf, Content.Rating == "Unrated"))

[1] 1

# There's only one of these, so just remove it.

dataDf <- dataDf %>%
filter(Content.Rating != "Unrated")
```

There is also another column that needs modification, which is the Android version. By reviewing the factor levels of the column, we can see that it reads "[number] and up", though we only wish to preserve the number for our visualisations.

```
levels(dataDf$Android.Ver)

 [1] "1.0 and up"          "1.5 and up"          "1.6 and up"
 [4] "2.0 and up"          "2.0.1 and up"        "2.1 and up"
 [7] "2.2 and up"          "2.3 and up"          "2.3.3 and up"
[10] "3.0 and up"          "3.1 and up"          "3.2 and up"
[13] "4.0 and up"          "4.0.3 and up"        "4.1 and up"
[16] "4.2 and up"          "4.3 and up"          "4.4 and up"
[19] "4.4W and up"         "5.0 and up"          "5.1 and up"
[22] "6.0 and up"          "7.0 and up"          "7.1 and up"
[25] "8.0 and up"          "Varies with device"

# Remove " and up" from the rows.
dataDf <- dataDf %>%
mutate(numVer = as.factor(str_replace_all(dataDf$Android.Ver, " and up", "")))

levels(dataDf$numVer)

 [1] "1.0"                 "1.5"                 "1.6"
 [4] "2.0"                 "2.0.1"               "2.1"
 [7] "2.2"                 "2.3"                 "2.3.3"
[10] "3.0"                 "3.1"                 "3.2"
[13] "4.0"                 "4.0.3"               "4.1"
[16] "4.2"                 "4.3"                 "4.4"
[19] "4.4W"                "5.0"                 "5.1"
[22] "6.0"                 "7.0"                 "7.1"
[25] "8.0"                 "Varies with device"

# Check how many rows have a varying version by device.
varyingSize <- dataDf %>%
filter(numVer == "Varies with device")

nrow(varyingSize)

[1] 947
```

A considerable portion of apps have a varying Android version requirement by device, so these apps are still valuable data that should be kept.

2.2.2 Data distribution visualisations

It is important to identify the distribution of our data across certain variables, especially ones such as review count and install count by various other factors, so that trends within the Play Store can be observed and followed by aspiring developers or companies to maximise their userbase.

```r
# The existing levels for the installs are far too specific,
# especially those such as 1+, 5+, 50+ and 100+, which could
# come under a wider label.

dataDf$installThresholds <- cut(dataDf$intInstalls,
# Set the new levels.
breaks = c(0, 500, 1000, 10000, 50000, 100000, 500000, 1000000,
           5000000, 10000000, 50000000, 100000000, 500000000,
           1000000000, Inf),
# Set the labels for each level.
labels = c("1 - 500", "500 - 1K", "1K - 10K", "10K - 50K",
           "50K - 100K", "100K - 500K",
           "500K - 1M", "1M - 5M", "5M - 10M",
           "10M - 50M", "50M - 100M", "100M - 500M",
           "500M - 1B", "1B+"))

installCounts <- dataDf %>%
group_by(installThresholds) %>%  # Group by the new install thresholds
summarise(count = n())    # Count the number of apps in each group

ggplot(installCounts, aes(x = installThresholds, y = count,
                          fill = installThresholds)) +
geom_bar(stat = "identity") +
geom_text(aes(label = count),
        # In the bar
        position = position_stack(vjust = 0.5), size = 4) +
labs(title = "Distribution of overall app installs",
     x = "Amount of installs", y = "Number of Apps",
     fill = "Install thresholds") +
theme(axis.text.x = element_text(angle =45, hjust = 1))
```

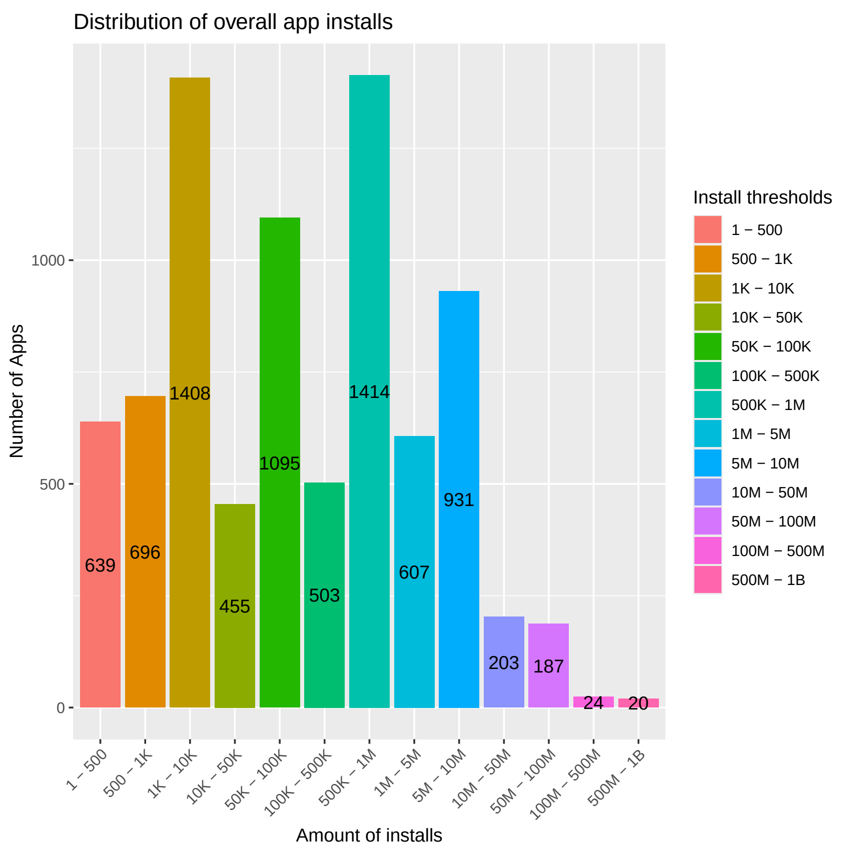

Figure 2.1: Bar chart of install counts.

We can see that the majority of the dataset contains apps between 1 and 1,000,000 installs. The most common install threshold for apps to be in is between 1 and 50,000 installs, followed by 500,000 to 1,000,000. This shows that the market is highly active, and users download a lot of various apps. There are even 20 apps with between 500,000,000 and 1,000,000,000 installs, showing how widely used the Play Store is, furthering just how vital of a market it is to place apps on.

```
installCounts <- dataDf %>%
group_by(Category) %>%  # Group by the category
summarise(count = n())   # Count the number of apps in each group

ggplot(installCounts, aes(x = count, y = Category, fill = Category)) +
geom_bar(stat = "identity") +
geom_text(aes(label = count),
        # At the end of the bar
        position = position_stack(), hjust = 1, size = 4) +
labs(title = "Distribution of apps by category",
     x = "Amount of apps", y = "Categories") +
theme(axis.text.x = element_text(angle =45, hjust = 1),
      legend.position = "none")
```

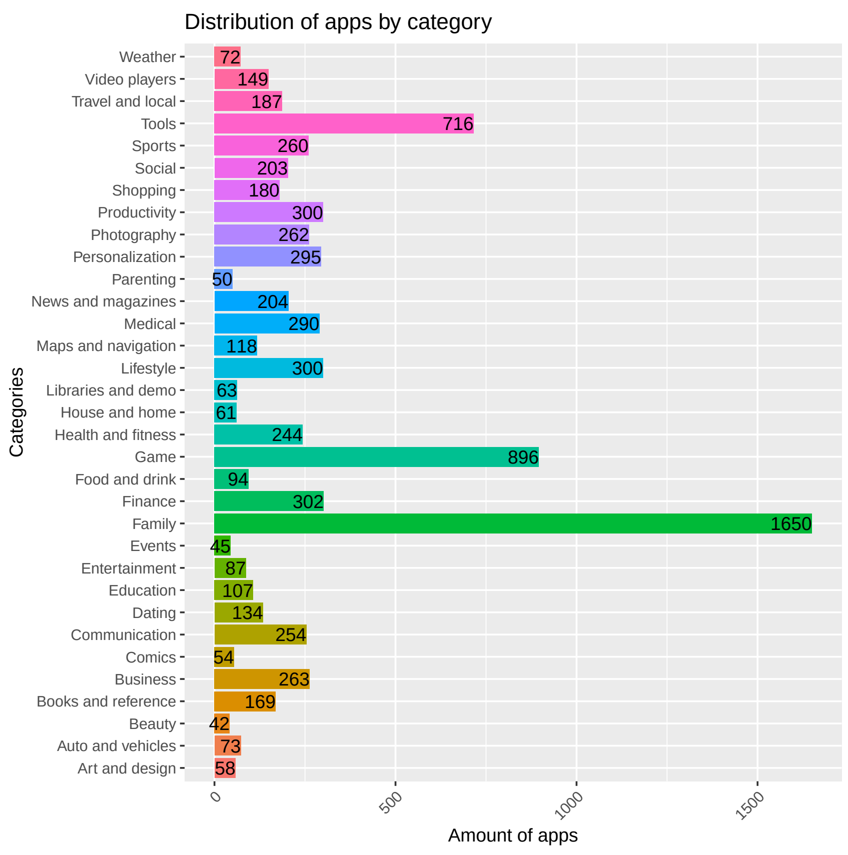

Figure 2.2: Bar chart of the amount of apps in each category.

The market is mostly dominated by apps categorised as "Family" apps, followed by games and tools. This suggests that these apps are successful due to the sheer quantity of them in relation to other available categories, which will be analysed further by checking the ratings of apps by category in Section 2.3

```
installCounts <- dataDf %>%
group_by(numVer) %>%  # Group by the version
summarise(count = n())   # Count the number of apps in each group

ggplot(installCounts, aes(x = count, y = numVer, fill = numVer)) +
geom_bar(stat = "identity") +
geom_text(aes(label = count),
        # Above the bar
        position = position_stack(vjust = 0.75), size = 4) +
labs(title = "Distribution of apps by minimum Android version",
     x = "Amount of apps", y = "Minimum version required") +
theme(axis.text.x = element_text(angle =45, hjust = 1),
      legend.position = "none")
```

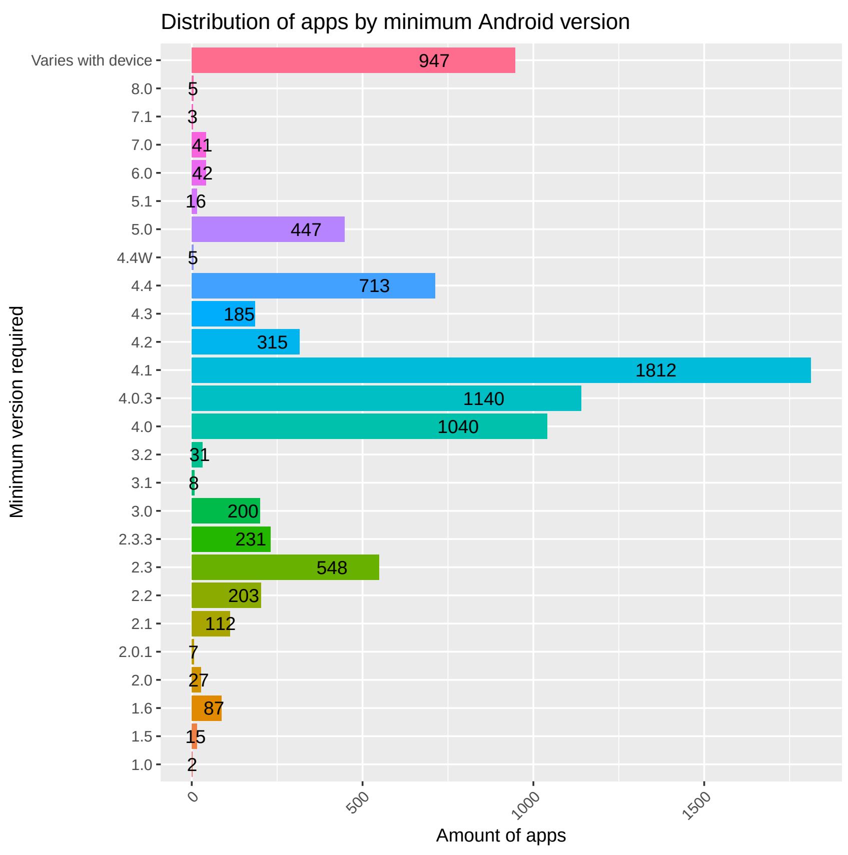

Figure 2.3: Bar chart of the amount of apps by minimum required Android version.

We can see that a large majority of the apps in the dataset require at least Android version 4.0, which was released in 2011. It is ill advised to develop apps below this version, as observed by the trends in Figure 2.3 due to their low market share and also how outdated they are, even relative to the time of this dataset's publication in 2018.

```r
notVarying <- dataDf %>%
# Have to remove this level because we can't make a string numeric.
filter(numVer != "Varies with device") %>%

# Convert values greater than 4 to 4, and values less than 4 to 1.
mutate(numVer = if_else(unfactor(numVer) >= 4.0, ">= 4.0", "< 4.0")) %>%
group_by(numVer) %>%
summarise(count = n())

ggplot(notVarying, aes(x = "", y = count, fill = numVer,
                       label = scales::percent(count / sum(count)))) +
    geom_bar(stat = "identity", width = 1) +
    geom_text(position = position_stack(vjust = 0.5)) +
    coord_polar("y", start = 0) +
    labs(title = "Minimum required Android versions",
        fill = "Required version") +
# Change the default blue colours
    scale_fill_manual(values=c("#9933FF", "#33FFFF"),
                        labels = c("Less than 4.0", "4.0 or higher")) +
    theme_void() # Hide the axes
```

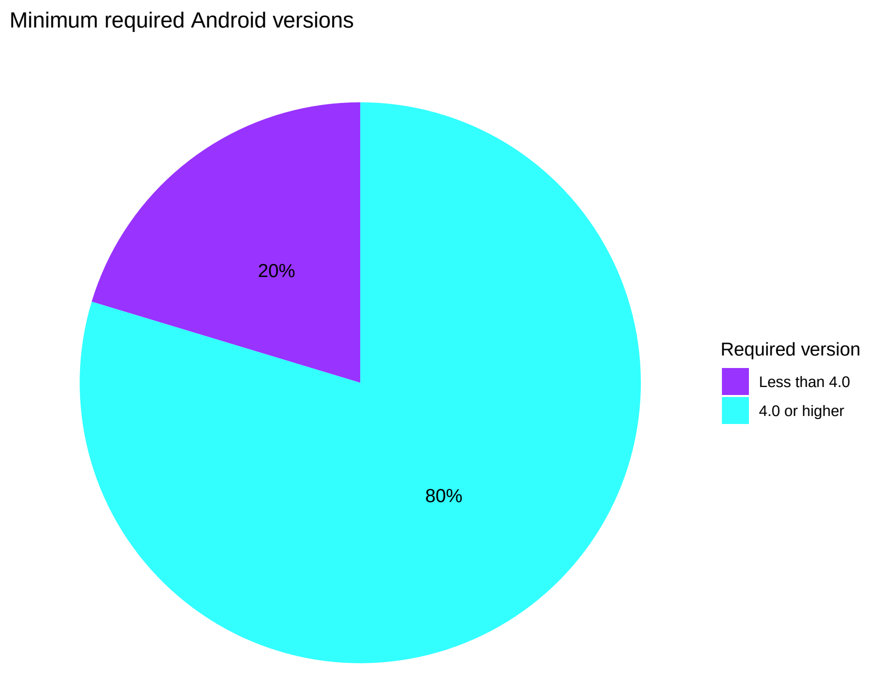

Figure 2.4: Summarisation of required Android versions

```
kable(notVarying, caption = "Required Android versions", "latex") %>%
kable_styling(latex_options = "HOLD_position")
```

Table 2.3: Required Android versions

numVer	count
< 4.0	1471
>= 4.0	5764

Figure 2.4 and Table 2.3 show how much the market is dominated by apps of more recent Android versions. Only 20% of all apps in this dataset have a minimum version below Android 4.0, likely due to the lack of key features and security maintenance. It is therefore essential to join the remaining 80% of the market in developing apps for newer Android versions.

```
pieData <- dataDf %>%
group_by(Type) %>%
summarise(count = n())

ggplot(pieData, aes(x = "", y = count, fill = Type,
                label = scales::percent(count / sum(count)))) +
        geom_bar(stat = "identity", width = 1) +
        geom_text(position = position_stack(vjust = 0.5)) +
        coord_polar("y", start = 0) +
        labs(title = "App type distribution",
            fill = "Type") +
    # Change the default blue colours to 2 random ones I searched for.
        scale_fill_manual(values=c("#A05195", "#FFA600"),
                          labels = c("Free", "Paid")) +
        theme_void() # Hide the axes
```

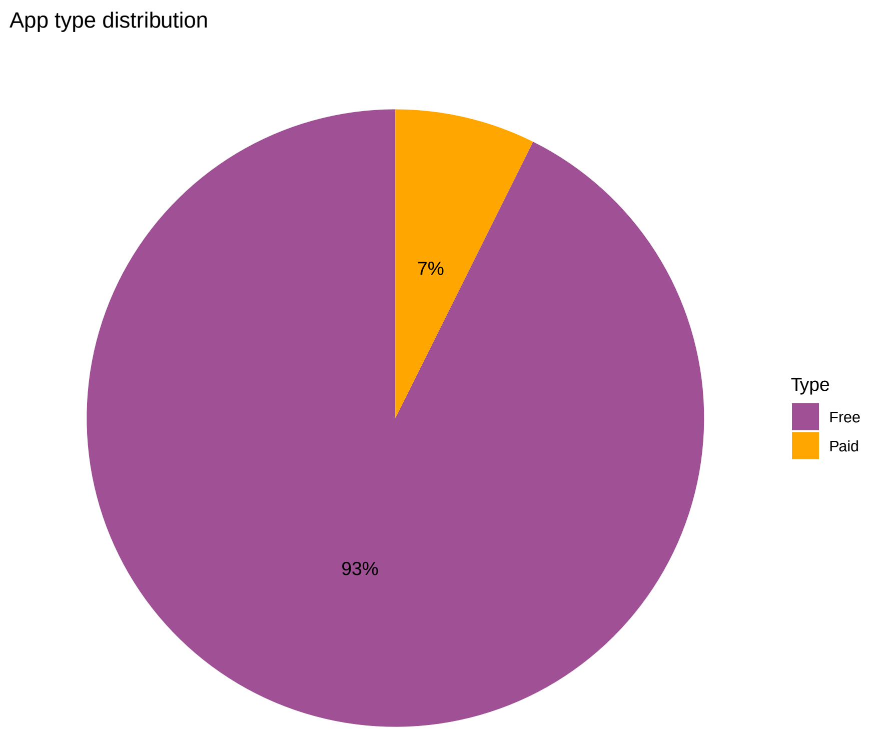

Figure 2.5: Paid and free app distribution

```
kable(pieData, caption = "Paid and free app distribution", "latex") %>%
kable_styling(latex_options = "HOLD_position")
```

Table 2.4: Paid and free app distribution

Type	count
Free	7580
Paid	602

Figure 2.5 and TAble 2.4 shows us the types of apps in the dataset, meaning whether the apps are free to download or whether they cost money. We can see that free apps have a 93% market share as opposed to paid apps that account for only 7%.

```
doesNotVary <- dataDf %>%
# Grab non-varying sizes by filtering 0.
filter(numSize != "0")

ggplot(doesNotVary, aes(x = numSize)) +
    geom_histogram(binwidth = 2, fill = "red", color = "black") +
    labs(x = "App size (in MB)", y = "Number of apps",
         title = "App size distribution") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

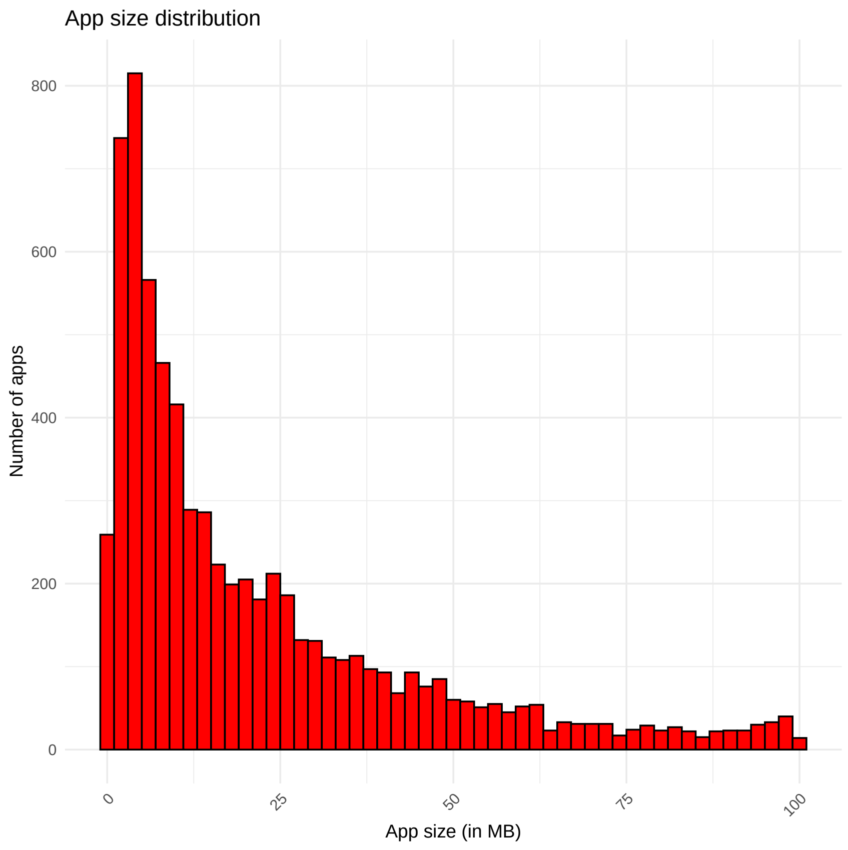

Figure 2.6: App size distribution histogram

It can be seen that the vast majority of apps in the dataset are all under 20 megabytes in size, likely due to storage capacities on mobile devices. The correlation between size and install counts will be analysed in Section 2.3

```r
ggplot(dataDf, aes(x = month(Last.Updated, label = TRUE))) +
    geom_histogram(fill = "skyblue", color = "black",
                   stat = "count") +
    labs(x = "Date", y = "Frequency",
         title = "Histogram of overall last update dates") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    facet_wrap(~year(Last.Updated))
```

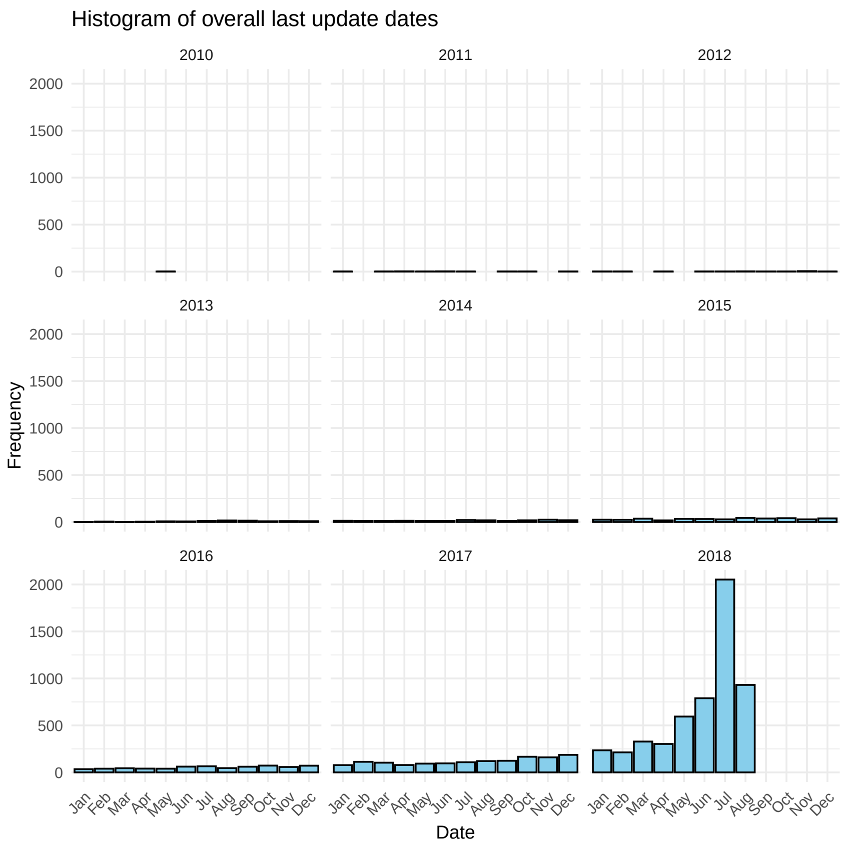

Figure 2.7: Overall histogram of when apps were last updated

This histogram reveals that the vast majority of the apps were last updated in 2018, correlating with the date that this dataset was scraped and released. We can analyse this in further detail by producing a histogram exclusively of apps that were updated in 2018.

```r
updated2018 <- dataDf %>%
filter(year(Last.Updated) == 2018)
```

```r
ggplot(updated2018, aes(x = Last.Updated)) +
    geom_histogram(binwidth = 3, fill = "orange", color = "black") +
    labs(x = "Date", y = "Frequency",
         title = "Histogram of updates in 2018") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

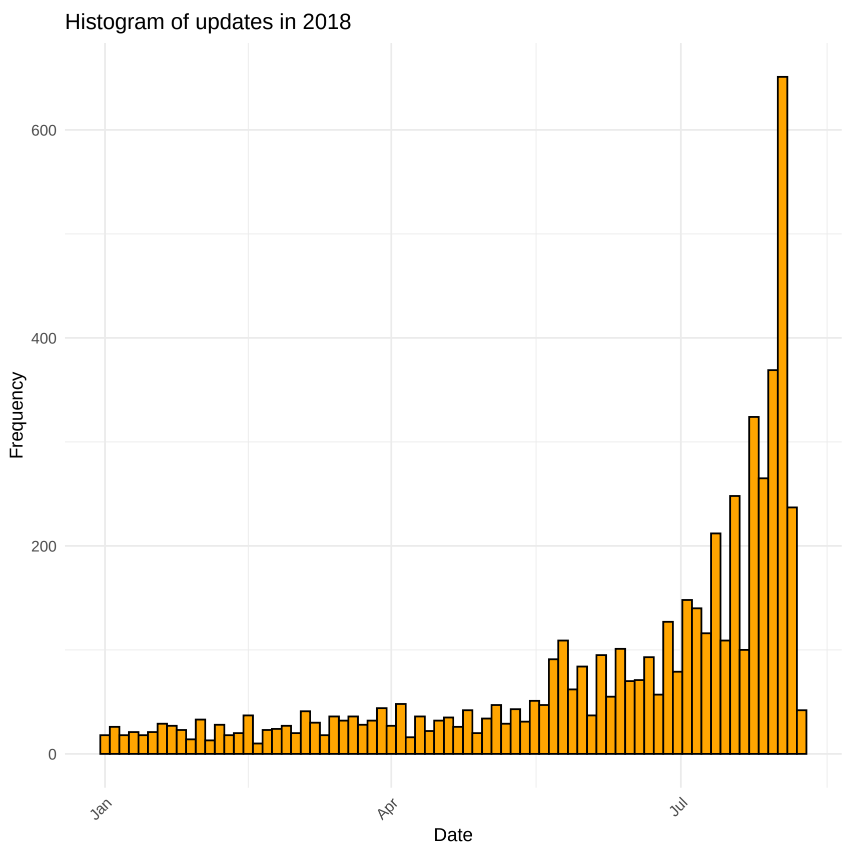

Figure 2.8: Histogram of app update dates in 2018

We can see from the distribution of Figure 2.8 that the majority of apps on the store are updated frequently, as the highest concentration of recent updates is at the time this data was scraped. This shows the trend that frequent and constant additions of new features or fixing of any bugs is a key element in having an app on the Play Store.

```r
updateDays <- dataDf %>%
# label makes the numeric day into the actual day.
# (0 -> Sunday, 1 -> Monday, etc)
    mutate(Day = wday(Last.Updated, label = TRUE)) %>%
    group_by(Day) %>%
    summarise(count = n())
```

```
ggplot(updateDays, aes(x = "", y = count, fill = Day,
      label = scales::percent(round(count / sum(count), 3)))) +
    geom_bar(stat = "identity", width = 1) +
    geom_text(position = position_stack(vjust = 0.5)) +
    coord_polar("y", start = 0) +
    labs(title = "Most common days for updates across all years") +
# Change the default blue colours
    scale_fill_brewer(palette = "Dark2") +
    theme_void() # Hide the axes
```

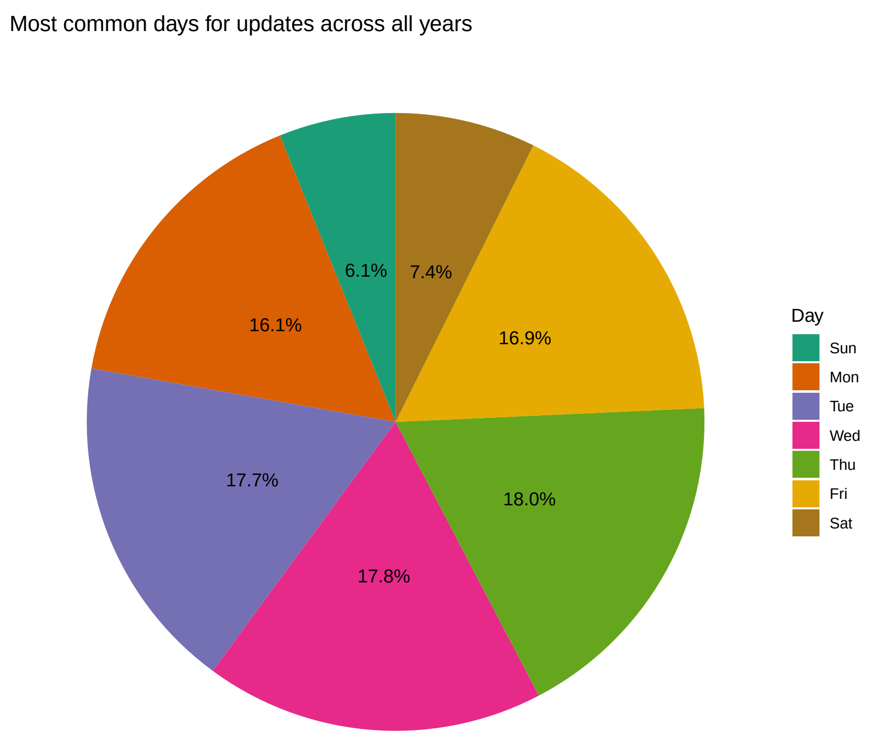

Figure 2.9: Days of the week when apps are updated in all documented years

It can be seen that apps are most commonly updated on weekdays, with Thursday being the most common by a slight margin. Apps are not commonly updated on weekends, likely due to developers not working these days.

```
contentRatings <- dataDf %>%
group_by(Content.Rating) %>%
summarise(count = n())

ggplot(contentRatings, aes(x = "", y = count, fill = Content.Rating,
                           label = scales::percent(count / sum(count))) +
        geom_bar(stat = "identity", width = 1) +
        geom_text(position = position_stack(vjust = 0.5)) +
        coord_polar("y", start = 0) +
        labs(title = "App content rating distribution",
            fill = "Type") +
    # Change the default blue colours
        scale_fill_brewer(palette = "Pastel1") +
        theme_void() # Hide the axes
```

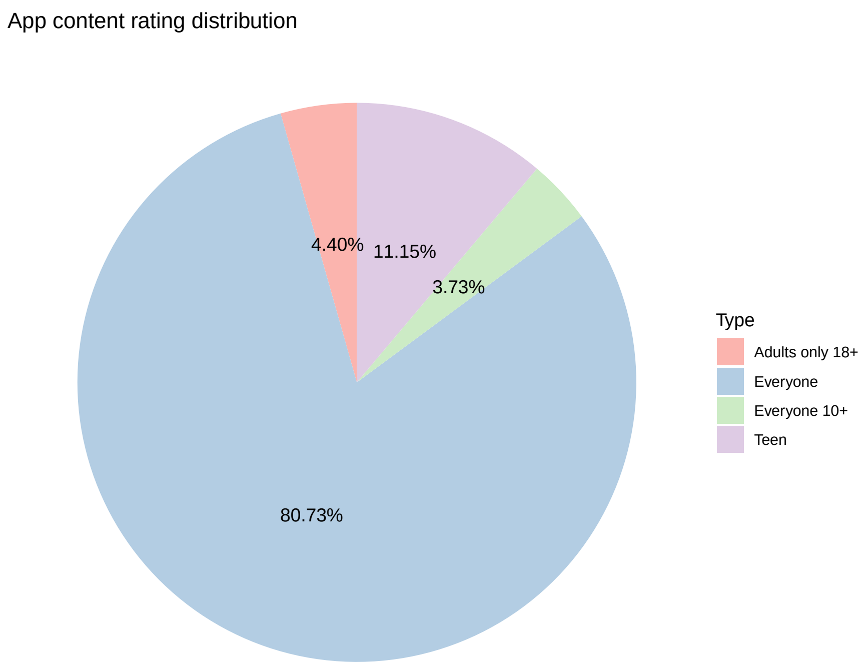

Figure 2.10: Distribution of content ratings

Apps targeted at users of all ages are a substantial portion of the dataset showing their high prevalence in the market, which is then followed by apps which target teens, apps that target adults only, and apps that target everyone aged 10 and above.

2.3 Visualisations for question answers

In this section, the questioned posed in Section 1.1 will be answered using the data that has been analysed throughout this report.

2.3.1 Is there a correlation between the average review score of an app and its install count?

```
ratingInstalls <- dataDf %>%
group_by(installThresholds) %>%
summarise(avgRating = mean(Rating))

ggplot(ratingInstalls, aes(x = installThresholds, y = avgRating,
                           fill = installThresholds)) +
    geom_bar(stat = "identity") +
    geom_text(aes(label = round(avgRating, 3)),
              position = position_stack(vjust = 0.5)) +
    labs(x = "Install Threshold", y = "Average Rating",
         title = "Average Rating by Install Threshold") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          legend.position = "none")
```

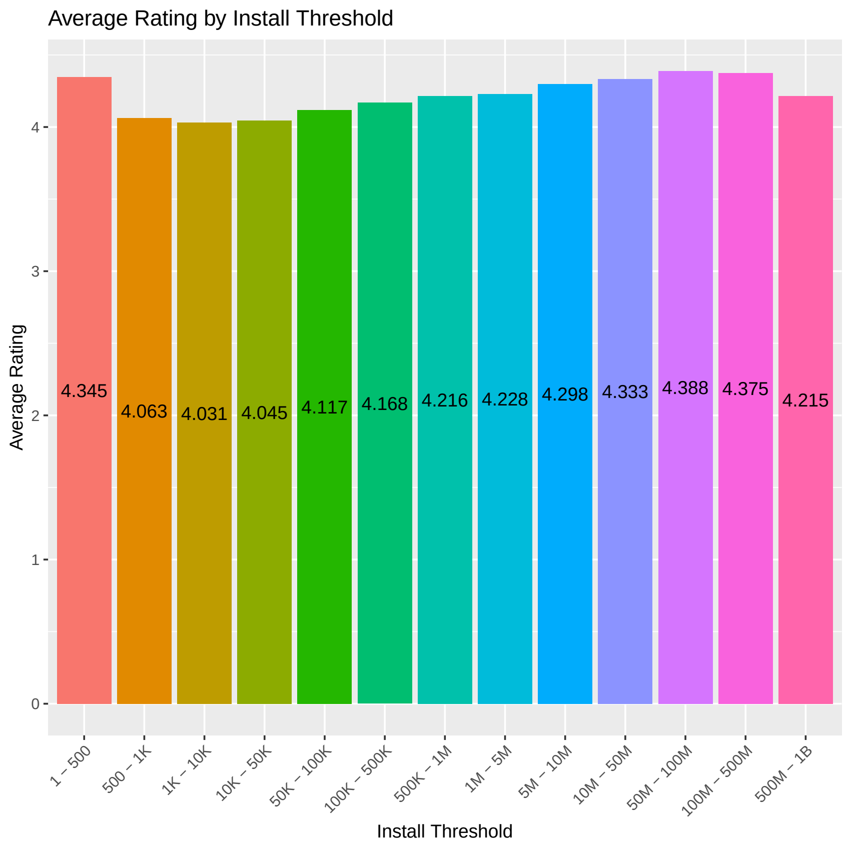

Figure 2.11: Average rating for apps in each install threshold

From this general bar chart, it can be observed that the average rating across all install thresholds does not subceed 4. Therefore, this data will be much easier to view in a more zoomed-in format as a line chart, where a linear regression line can be added to show the trend between installs and rating in closer detail.

```
ggplot(ratingInstalls, aes(x = installThresholds, y = avgRating,
                           group = 1)) +
    geom_point(aes(color = installThresholds), size = 2) +
    geom_line(aes(group = 1), color = "skyblue") +
    geom_smooth(formula = y ~ x, method = lm, alpha = 0.1,
                color = "orange") +
    labs(x = "Install Threshold", y = "Average Rating",
         title = "Average Rating by Install Threshold") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          legend.position = "none")
```

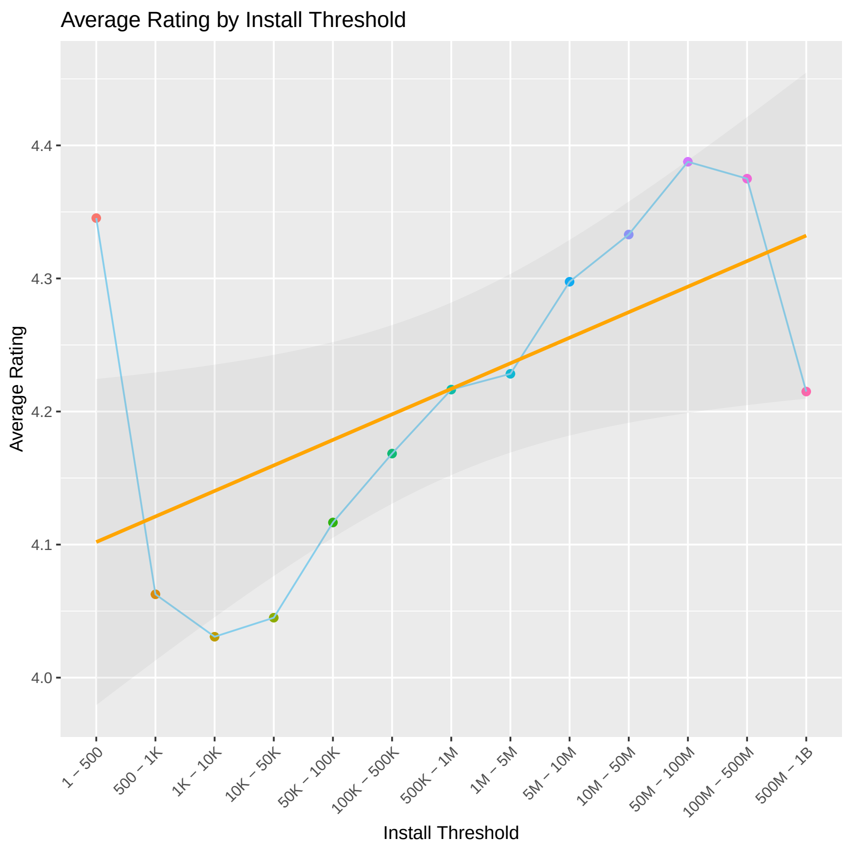

Figure 2.12: Zoomed in average rating for apps in each install threshold

From this line graph, the differences in average rating are much clearer. We can observe a mostly positive correlation between the amount of installs and the average rating of an app. Apps with between 1 and 500 installs have a very high review score on average due to the low amount of reviews they would have, where a single review could skew their overall rating. This is evident immediately with apps between 500 and 50,000 installs suffering from lower average ratings, which likely causes a loop that makes users not want to install these apps. Past this point, the average rating increases at a mostly linear rate, showing the strong link between installs and review score. Apps between 500 million and 1 billion reviews have a lower average review score than the trend would otherwise suggest, which may be due to the specific apps in this threshold rather than an overall trend.

Overall, this matches the expected answer from Section 1.1.

2.3.2 Which category of app has the highest review score on average?

```
filterReviews <- dataDf %>%
    group_by(Category) %>%
    summarise(avgRating = mean(Rating), count = n())

ggplot(filterReviews, aes(x = avgRating, y = Category, fill = Category)) +
    geom_bar(stat = "identity") +
    geom_text(aes(label = round(avgRating, 3)),
              position = position_stack(vjust = 0.5)) +
    labs(title = "Average Rating by App Category", x = "App Category",
         y = "Average Rating") +
    # Too many categories for a legend to reasonably fit.
    theme(legend.position = "none")
```

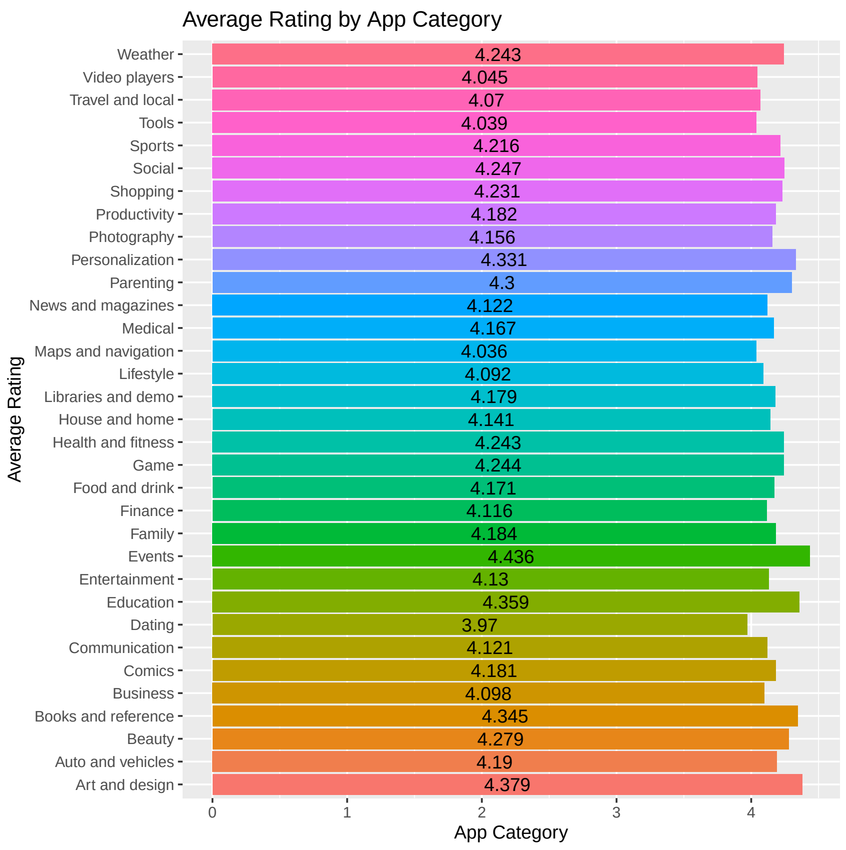

Figure 2.13: Average rating by app category

Figure 2.13 depicts that the highest rated category is the Events category, followed by Art and Design and Education. There are 33 categories of app, and the only one to have an average rating below 4 is the Dating category, suggesting that developers of this category of app must prepare for potential lower ratings overall than they would see in other categories. In Table 2.5 below, the entire data frame used for this graph is shown, with the average rating and amount of occurrences of each category in the dataset in descending order.

```r
# Sort by average rating (descending)
filterReviews <- arrange(filterReviews, desc(avgRating))

kable(filterReviews, caption = "Categories by review score", "latex") %>%
# Large table, so make the font small.
kable_styling(font_size = 8, latex_options = c("HOLD_position"))
```

Table 2.5: Categories by review score

Category	avgRating	count
Events	4.436	45
Art and design	4.379	58
Education	4.359	107
Books and reference	4.345	169
Personalization	4.331	295
Parenting	4.300	50
Beauty	4.279	42
Social	4.247	203
Game	4.244	896
Weather	4.243	72
Health and fitness	4.243	244
Shopping	4.231	180
Sports	4.216	260
Auto and vehicles	4.190	73
Family	4.184	1650
Productivity	4.182	300
Comics	4.181	54
Libraries and demo	4.179	63
Food and drink	4.171	94
Medical	4.167	290
Photography	4.156	262
House and home	4.141	61
Entertainment	4.130	87
News and magazines	4.122	204
Communication	4.121	254
Finance	4.116	302
Business	4.098	263
Lifestyle	4.092	300
Travel and local	4.069	187
Video players	4.045	149
Tools	4.039	716
Maps and navigation	4.036	118
Dating	3.970	134

Table 2.5 shows that the most frequent categories (seen in the "count" column) were the Family, Game and Tools. Of these most frequent categories, Game has the highest average rating, but not of the overall categories, meaning the assumption made in Section 1.1 was incorrect.

2.3.3 What is the distribution of prices? Are there certain prices used by many apps?

To properly analyse the distribution of prices, the column will need some modification in order to categorise prices and remove apps that do not have a price (free apps) in order to produce readable visualisations.

```
# Price has 73 factor levels.
nlevels(dataDf$Price)

[1] 73

# To show the most frequent ones, we can create
# an ordered list of the most common ones, and consider all the others
# to be in an "other" category.

# From the previous graphs, we know that the vast majority (93%) of
# our dataset is comprised of free apps. Therefore, these will be
# filtered out to not pollute the graphs with the price point of 0.
paidApps <- dataDf %>%
filter(Price != "0")

levels(paidApps$Price)

 [1] "$0.99"   "$1.00"   "$1.20"   "$1.29"   "$1.49"   "$1.50"   "$1.59"
 [8] "$1.61"   "$1.70"   "$1.75"   "$1.76"   "$1.97"   "$1.99"   "$10.00"
[15] "$10.99"  "$11.99"  "$12.99"  "$13.99"  "$14.00"  "$14.99"  "$15.46"
[22] "$15.99"  "$16.99"  "$17.99"  "$18.99"  "$19.40"  "$19.99"  "$2.00"
[29] "$2.49"   "$2.50"   "$2.56"   "$2.59"   "$2.90"   "$2.95"   "$2.99"
[36] "$24.99"  "$29.99"  "$299.99" "$3.02"   "$3.04"   "$3.08"   "$3.28"
[43] "$3.49"   "$3.88"   "$3.90"   "$3.95"   "$3.99"   "$33.99"  "$37.99"
[50] "$379.99" "$389.99" "$39.99"  "$399.99" "$4.29"   "$4.49"   "$4.59"
[57] "$4.60"   "$4.77"   "$4.84"   "$4.99"   "$400.00" "$5.49"   "$5.99"
[64] "$6.49"   "$6.99"   "$7.49"   "$7.99"   "$79.99"  "$8.49"   "$8.99"
[71] "$9.00"   "$9.99"   "0"

# Calculate frequency of each price
priceCounts <- table(paidApps$Price)
# Sort the price counts in descending order
priceCounts <- sort(priceCounts, decreasing = TRUE)
# Extract the top 15 levels
mostCommon <- names(priceCounts)[1:15]
# Count of other prices combined
others <- sum(priceCounts[16:length(priceCounts)])
# Create a new factor combining top 15 levels and others
paidApps$priceFactor <- factor(paidApps$Price, levels = c(mostCommon, "Others"))
# Replace levels not in top 15 with "Others"
paidApps$priceFactor[!(paidApps$priceFactor %in% mostCommon)] <- "Others"
# Recalculate frequency of each price after combining levels
priceCounts <- table(paidApps$priceFactor)

# Convert to data frame for ggplot
pricePlot <- data.frame(Price = names(priceCounts),
                        Frequency = as.numeric(priceCounts))
```

```
ggplot(pricePlot, aes(x = Price, y = Frequency, fill = Price)) +
    geom_bar(stat = "identity") +
    geom_text(aes(label = Frequency), position = position_stack(vjust = 0.5)) +
    labs(x = "Prices", y = "Frequency", title = "App price point frequencies") +
    # $399.99 and $14.99 would not show in price order normally, so they can
    # be manually adjusted.
    scale_x_discrete(limits = c("$0.99","$1.49","$1.99","$2.49","$2.99","$3.49",
                                "$3.99","$4.49","$4.99","$5.99","$6.99","$7.99",
                                "$9.99","$14.99","$399.99","Others")) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          legend.position = "none")
```

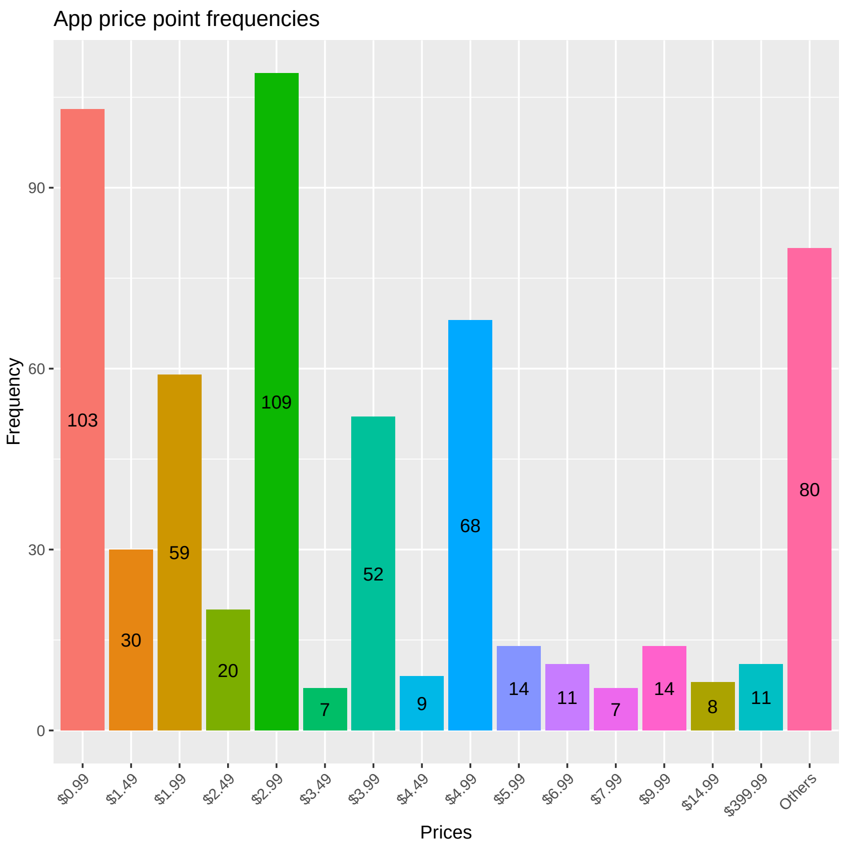

Figure 2.14: Frequency of app prices

```
# Sort by average rating (descending)
pricePlot <- arrange(pricePlot, desc(Frequency))
# Move "Others" to the bottom of the table.
pricePlot <- rbind(pricePlot[!pricePlot$Price == "Others",],
                   pricePlot[pricePlot$Price == "Others",])
# Convert back to a tibble to remove the index column.
pricePlot <- tibble(pricePlot)

# Split over two columns.
```

```
kable(list(pricePlot[1:8,], pricePlot[9:16,]), "latex") %>%
kable_styling(latex_options = "HOLD_position")
```

Price	Frequency	Price	Frequency
$2.99	109	$9.99	14
$0.99	103	$399.99	11
$4.99	68	$6.99	11
$1.99	59	$4.49	9
$3.99	52	$14.99	8
$1.49	30	$3.49	7
$2.49	20	$7.99	7
$5.99	14	Others	80

As per Table 2.4, we know that there are only 602 paid apps, which is why Figure 2.14 is on a much smaller scale. We can see that 3 of the 5 most common price points for apps are under $3, meaning the expectation in 1.1 was mostly correct. We can determine that the most common price points used in the dataset are $2.99, $0.99 and $4.99, suggesting that paid apps should mostly be kept under $5. However, this graph yields some additional surprising information, most notably in the amount of apps that are priced at $399.99. While it is still a low number at 11, it does still rank within the top fifteen price points in this dataset.

2.3.4 Do paid apps receive higher or lower review score on average?

```r
filterReviews <- dataDf %>%
group_by(Type) %>%
    summarise(avgRating = mean(Rating), count = n())

ggplot(filterReviews, aes(x = avgRating, y = Type, fill = Type)) +
    geom_bar(stat = "identity") +
    geom_text(aes(label = round(avgRating, 3)),
              position = position_stack(vjust = 0.5)) +
    labs(title = "Average rating by app type", x = "Average rating",
         y = "App type")
```

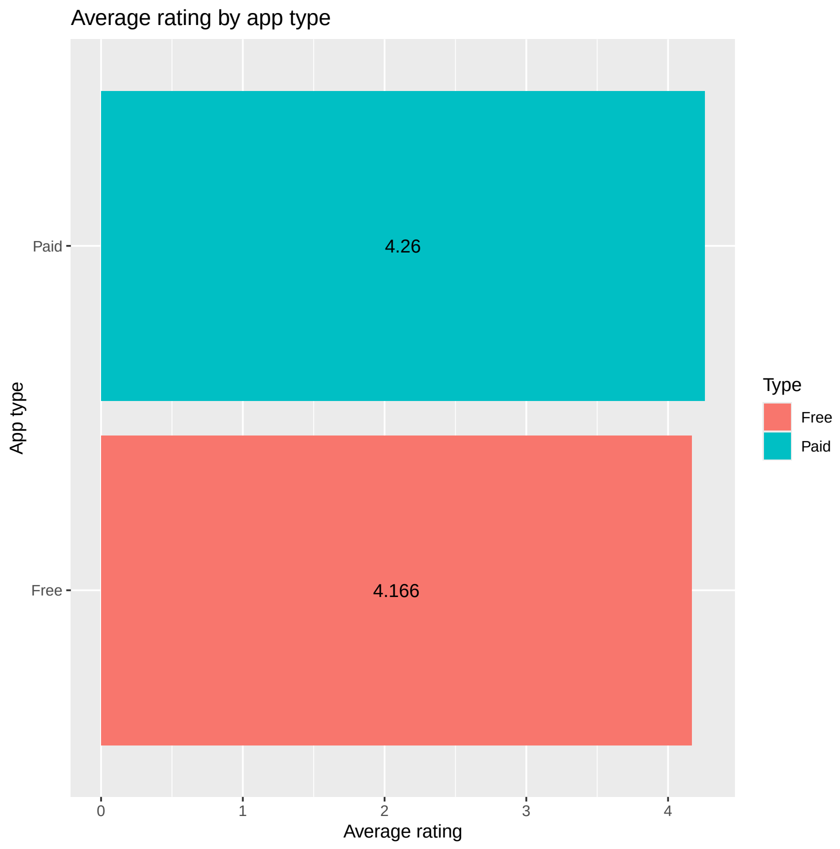

Figure 2.15: Average rating of paid and free apps

Surprisingly, Figure 2.15 shows us that paid apps often have a higher review score than free apps. This could be due to several factors such as that users who pay for an app might be more invested in its success, leading them to be more forgiving of minor issues and more likely to leave a positive review. Furthermore, the cost might act as a filter, deterring users who wouldn't like the app from downloading it in the first place, leading to a more engaged user base with a higher likelihood of leaving positive ratings.

2.3.5 Is there a correlation between an app's content rating and it's review score?

Because Figure 2.10 shows us that a significant portion of the data has "Everyone" as its content rating, we can first observe the ratings specifically of that rating, as it would inflate the Y axis of the visualisations if paired with the others.

```
everyone <- dataDf %>%
filter(Content.Rating == "Everyone")

ggplot(everyone, aes(x = Rating)) +
    geom_histogram(fill = "pink", color = "black", binwidth = 0.1) +
    labs(x = "Score", y = "Amount of apps",
         title = "Review scores across 'Everyone'") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

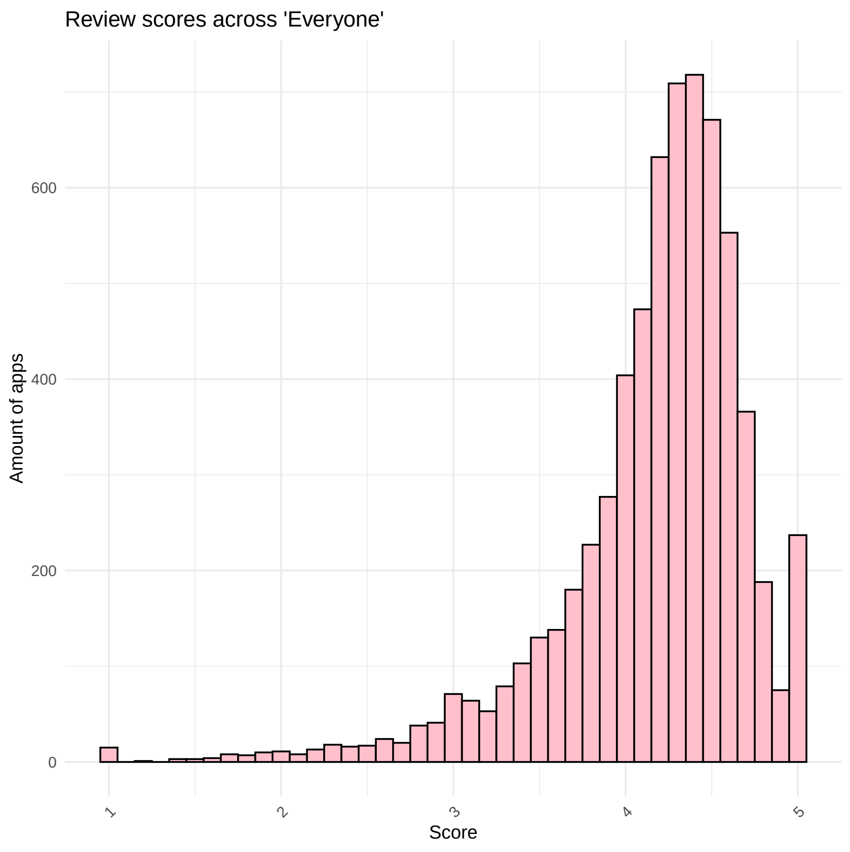

Figure 2.16: Review score distributions across the 'Everyone' rating

We can see that most apps with the "Everyone" content rating have a review score of over 4, though there is still a distribution across all scores from 1 to 5. We can see how this compares against the other content ratings in Figure 2.17.

```r
notEveryone <- dataDf %>%
filter(Content.Rating != "Everyone")

# binwidth is 0.1 because our ratings are to 1 decimal place.
ggplot(notEveryone, aes(x = Rating)) +
    geom_histogram(fill = "pink", color = "black", binwidth = 0.1) +
    labs(x = "Score", y = "Amount of apps",
         title = "Review scores across all other content ratings") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    facet_wrap(~Content.Rating, ncol = 3)
```

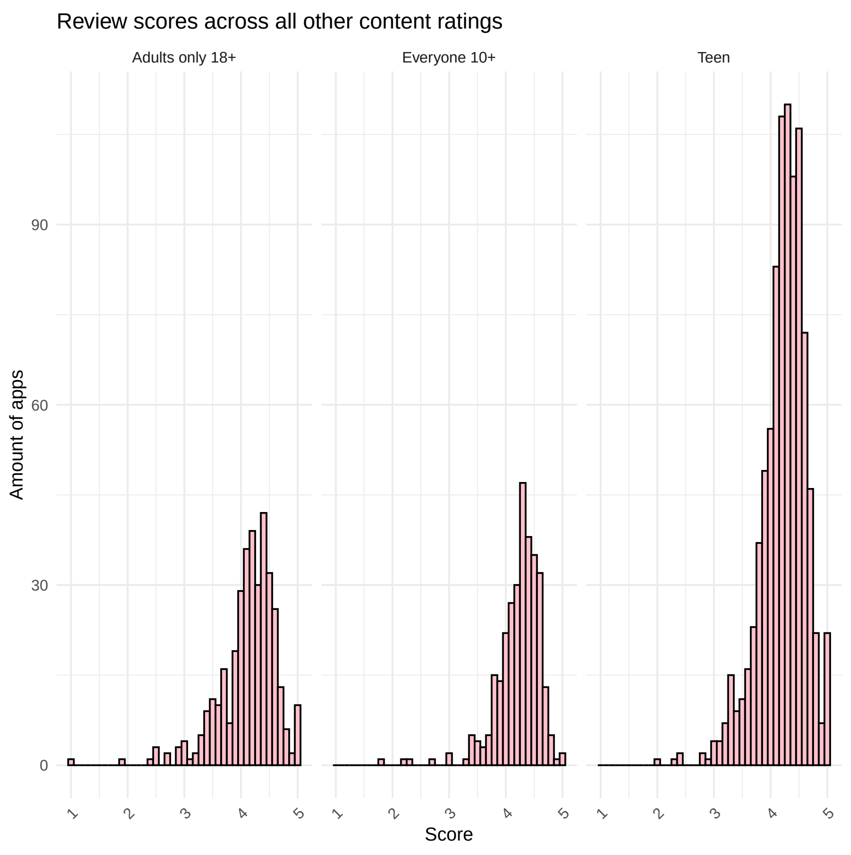

Figure 2.17: Review score distributions across other content ratings

We can see that this pattern mostly repeats across all content ratings, though it can also be observed that apps with a content rating of "Adults only 18+" have a higher concentration of reviews lower than 4. This can be analysed further with a bar chart as shown in Figure 2.18.

```r
ratingInstalls <- dataDf %>%
    group_by(Content.Rating) %>%
    summarise(avgRating = mean(Rating))
```

```
ggplot(ratingInstalls, aes(x = Content.Rating, y = avgRating,
                           fill = Content.Rating)) +
    geom_bar(stat = "identity") +
    geom_text(aes(label = round(avgRating, 3)),
              position = position_stack(vjust = 0.5)) +
    labs(x = "Content age rating", y = "Average review score",
         title = "Average review score by content age rating") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          legend.position = "none")
```

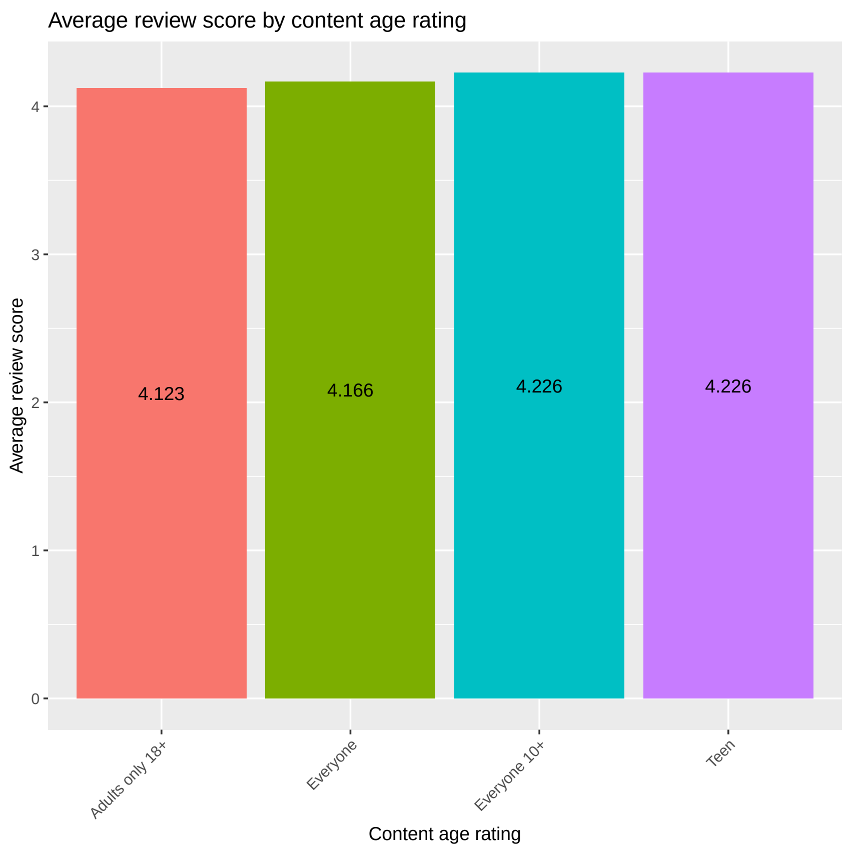

Figure 2.18: Average review score for each content rating

"Adults only" is indeed the lowest rated category. This subverts the expectation stated in Section 1.1. This is likely due to the trends in Figure 2.13 where Dating apps were rated lower than all others. Dating apps are all 18+ by nature, so they would bring down the average of their associated content rating.

2.3.6 Does app size play a role in install count?

```
# Filter out apps that vary in size.
# Because "Varies with device" was changed to 0 earlier, remove 0.
installsSize <- dataDf %>%
filter(numSize != 0) %>%
group_by(installThresholds, numSize)

ggplot(installsSize, aes(x=installThresholds, y=numSize)) +
    geom_boxplot(fill='skyblue') +
    labs(title = "App installs by size", x = "Installs",
         y = "Size in MB") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

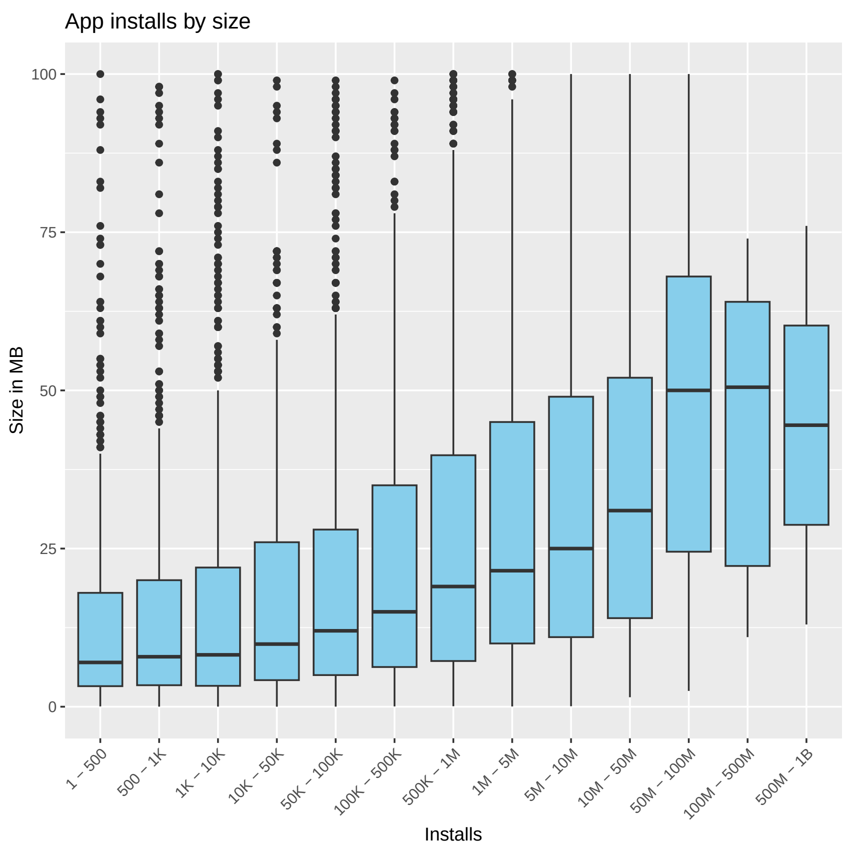

Figure 2.19: Boxplot of app size distributions across install thresholds

```
doesNotVary <- dataDf %>%
# Grab non-varying sizes by filtering 0 out,
# then grab the first eight factor levels.
filter(numSize != "0" & installThresholds %in% c("1 - 500",
"500 - 1K", "1K - 10K",
"10K - 50K", "50K - 100K", "100K - 500K",
"500K - 1M", "1M - 10M"))

ggplot(doesNotVary, aes(x = numSize)) +
    geom_histogram(fill = "purple", color = "black", binwidth = 4) +
    labs(title = "Histogram of size distribution (1 - 1M installs)",
         x = "Size (in MB)", y = "Amount of apps") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    facet_wrap(~installThresholds, ncol = 3)
```

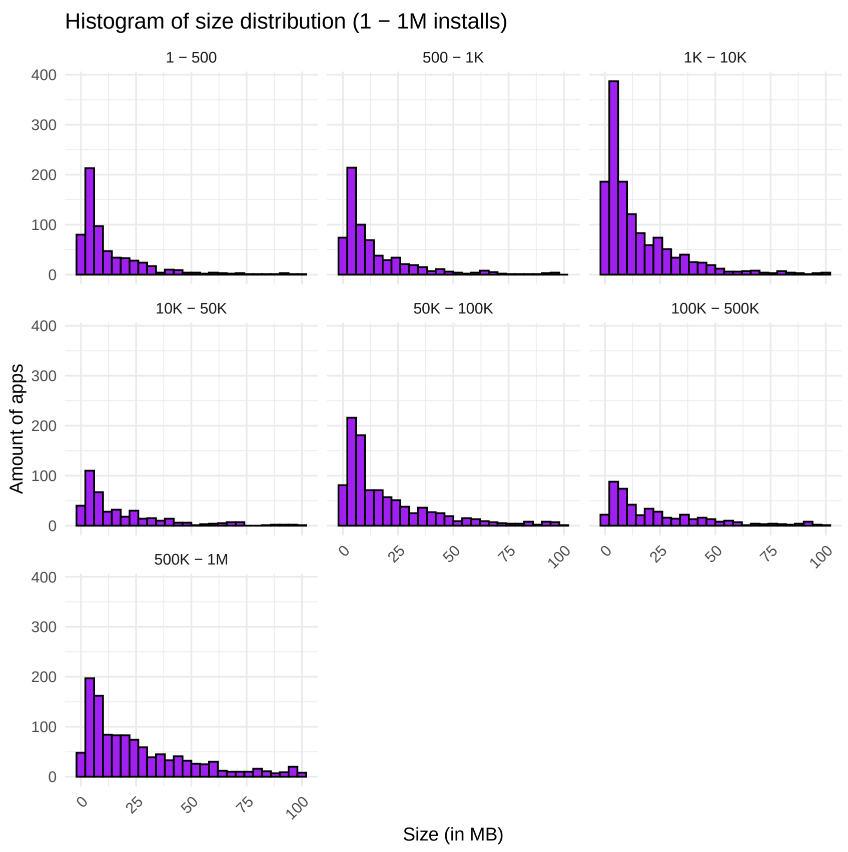

Figure 2.20: Size distribution across apps between 1 and 1 million installs

```
doesNotVary <- dataDf %>%
# Grab the other factor levels by filtering to those NOT in this list.
filter(numSize != "0" & !installThresholds %in% c("1 - 500",
"500 - 1K", "1K - 10K",
"10K - 50K", "50K - 100K", "100K - 500K",
"500K - 1M", "1M - 10M"))

ggplot(doesNotVary, aes(x = numSize)) +
    geom_histogram(fill = "purple", color = "black", binwidth = 4) +
    labs(x = "Size (in MB)", y = "Amount of apps",
        title = "Histogram of size distribution (1M - 1B installs)") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    facet_wrap(~installThresholds, ncol = 2)
```

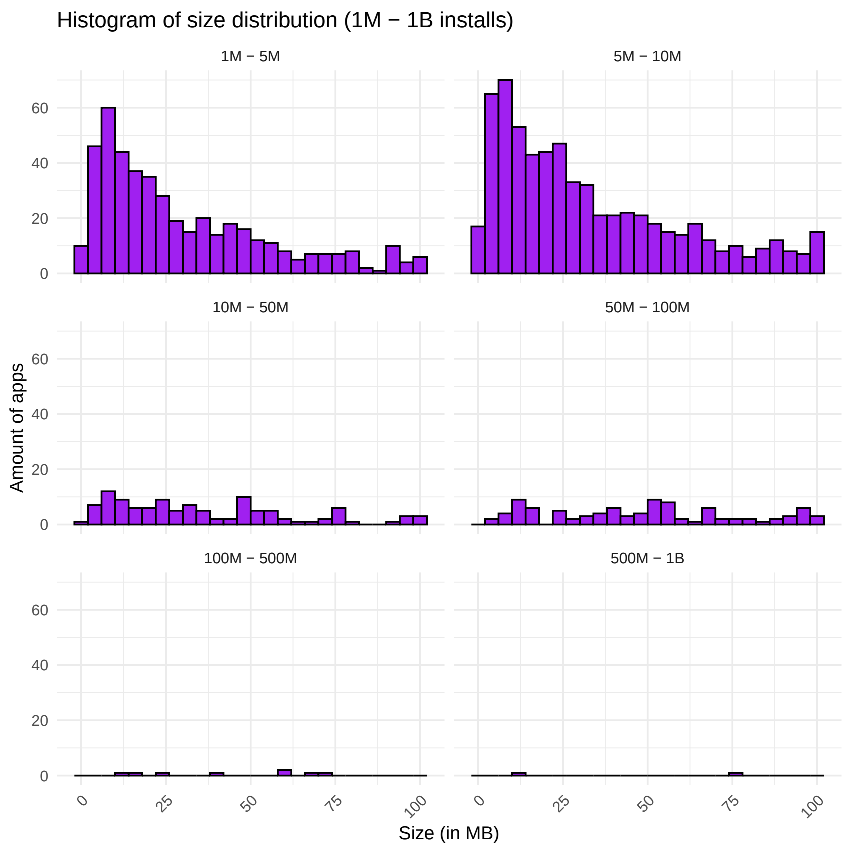

Figure 2.21: Size distribution across apps between 1 million and 1 billion installs

```
installsSize <- installsSize %>%
    group_by(installThresholds) %>%
    summarise(avgSize = mean(numSize))

ggplot(installsSize, aes(x = avgSize, y = installThresholds,
                         fill = installThresholds)) +
    geom_bar(stat = "summary", fun = mean) +
    geom_text(aes(label = round(avgSize, 3)),
              position = position_stack(vjust = 0.5)) +
    labs(title = "Average size of apps by install threshold",
        x = "Size in MB", y = "Installs") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          legend.position = "none")
```

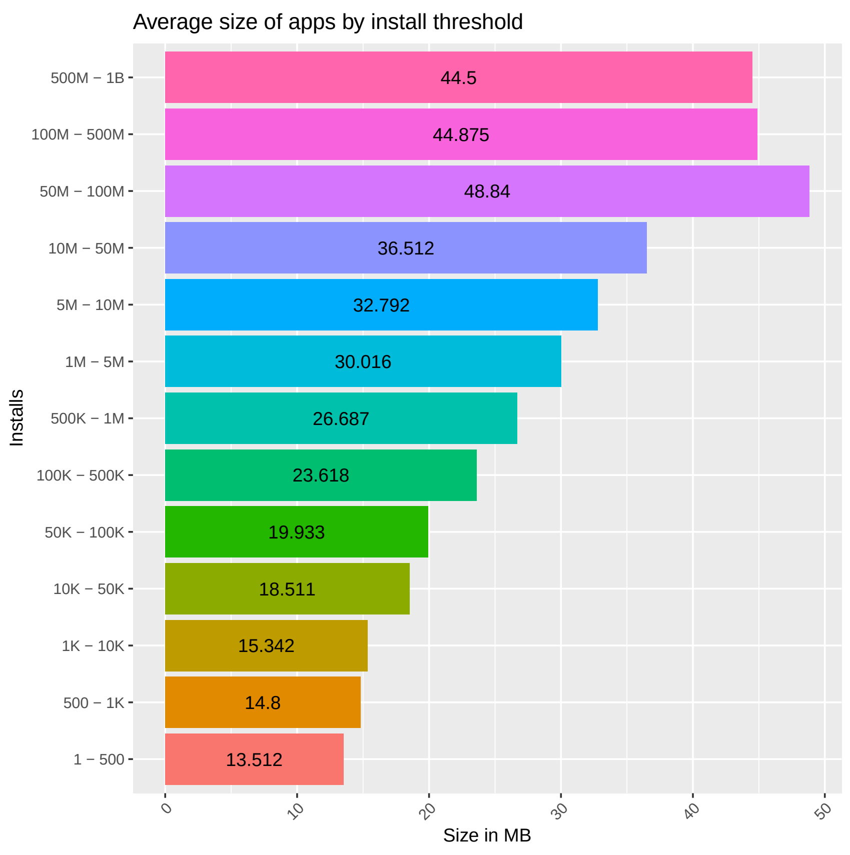

Figure 2.22: Average size of apps by install threshold

Interestingly, we can observe a mostly positive correlation between app size and install count, with the average sizes of apps mostly increasing by small amounts, except for apps with between 100 million and 1 billion downloads, which both sit around the 44/45 megabyte size on average. This suggests that it is important to ensure that while apps do have a variety of features that would increase their size, it is important to optimise it where possible due to the limited storage space available on mobile devices that could cause the user to uninstall apps in favour of others.

Summary and conclusion

3.1 Summary

- Is there a positive correlation between the average review score of an app and its install count?
  - Expected: Yes, as it is harder to weigh the average down when the volume of installs increases.
  - Actual: Yes, there is a positive correlation between the two.
- Which category of app has the highest review score on average?
  - Expected: Games are likely to have the highest review scores from users enjoying their experiences.
  - Actual: Events apps have the highest review score on average due to a low sample size.
  - Of the top five categories, however, Games are rated the highest.
- What is the distribution of prices? Are there certain prices used by many apps?
  - Expected: If an app is not free, it would likely be under $3 to ensure people buy it.
  - Actual: The majority of apps are priced at $2.99, $0.99 or $4.99.
- Do paid apps receive higher or lower review score on average?
  - Expected: No, as users may be more harsh with their reviews if they paid for the app.
  - Actual: Paid apps actually do receive higher average ratings, perhaps due to users being more forgiving of products that they paid money for.
- Is there a correlation between an app's content rating and it's review score?
  - Expected: Yes, with adult apps likely being higher rated because children would be unlikely to have accounts.
  - Actual: Yes, though it is somewhat minor. Expectations were defied due to "Adults only" apps actually being the lowest rated on average.
- Does app size play a role in install count?
  - Expected: Yes, as users would likely not want to download very large apps due to internet speeds and device storage.
  - Actual: The data showed a positive correlation between average size and install count to a plateau of approximately 49MB.

3.2 Conclusion

This report has analyzed a dataset of Google Play Store apps, revealing valuable insights into app characteristics, user ratings, and trends. We observed the distributions of data across the dataset, revealing trends in review scores and install counts with categories, prices, age ratings and app size. Overall, this exploration of the Google Play Store data highlights valuable trends for app developers and users alike. Developers can leverage these insights to identify key focus points in the development process, such as tailoring their apps to be for certain age groups or pricing them at certain costs. For users, understanding these trends can inform their app selection process, leading them towards apps that better meet their needs.

Rename variables to be relevant to the graph they are for. Word count troubles; you're within 10% over, but that still puts you at 3300/3000.

Bibliography

Ceci, Laura (Aug. 29, 2023a). *Number of Apple App Store and Google Play mobile app downloads worldwide from 3rd quarter 2016 to 1st quarter 2023*. URL: https://www.statista.com/statistics/695094/quarterly-number-of-mobile-app-downloads-store/ (visited on 05/10/2024).

Ceci, Laura (Dec. 8, 2023b). *Number of available apps in the Google Play Store from 2nd quarter 2015 to 3rd quarter 2022*. URL: https://www.statista.com/statistics/289418/number-of-available-apps-in-the-google-play-store-quarter/ (visited on 05/10/2024).