Projet Phare Google Data Analytics

Étude de cas Cyclistic Chicago

Mustapha Rherrad

24/10/2024

Contents

Introduction . . . 2
Le scénario . . . 2
La Phase de la Demande . . . 2
Énoncé de la tâche commerciale : . . . 2
Mes principaux partenaires : . . . 3
La Phase de Préparation . . . 3
Source des données: . . . 3
Organisation et description des données: . . . 3
Créditabilité des données: . . . 3
Sécurité des données: . . . 3
Limitations des données : . . . 4
La Phase du Traitement: . . . 4
Mise en place de l'environnement . . . 4
Importation des données . . . 6
La Phase d'Analyse : . . . 13
Aperçu sur les modalités d'utilisation des vélos . . . 13
Manipulation des dates pour une analyse détaillée des trajets (durée, jour de la semaine, heure de la journée..) . . . 15
analyse descriptive de la durée du trajet . . . 18
Pourcentage de trajets effectués par type client dans les 7 jours de la semaine . . . 20
Pourcentage de trajets effectués par type client au fil des heures de la journée . . . 25
Durée moyenne du parcours en fonction du type de cycliste et du nombre de chaque type de cycliste . . . 26
Analyse des effets de la saisonnalité . . . 27
Phase de Partage . . . 34

Introduction

Pour le projet de fin d'études du certificat Google Data Analytics, j'ai choisi de travailler sur les données de l'utilsation des vélos Cyclistic en libre-service. Pour l'étude de cas, j'effectuerai des tâches réelles d'analyste de données junior pour l'équipe marketing de Cyclistic, une société fictive de partage de vélos à Chicago. Pour répondre aux questions commerciales clés, je vais suivre les six étapes du processus d'analyse des données:

- demander,
- préparer,
- traiter,
- analyser,
- partager
- agir.

Le scénario

La directrice marketing de Cyclistic, Lily Moreno, estime que la croissance future de l'entreprise dépend de la maximisation du nombre d'adhésions annuelles. Par conséquent, l'équipe d'analystes marketing souhaite comprendre comment les cyclistes occasionnels et les membres annuels utilisent différemment les vélos Cyclistic. À partir de ces informations, l'équipe d'analystes pourrait être en mesure de concevoir une nouvelle stratégie marketing pour convertir les cyclistes occasionnels en membres annuels.

Trois questions guideront la future campagne marketing :

1. Comment les membres annuels et les cyclistes occasionnels utilisent-ils différemment les vélos Cyclistic ?
2. Pourquoi les cyclistes occasionnels achèteraient-ils des adhésions annuelles à Cyclistic ?
3. Comment Cyclistic peut-il utiliser les médias numériques pour inciter les cyclistes occasionnels à devenir membres ?

Moreno m'a assigné la première question.

La Phase de la Demande

Énoncé de la tâche commerciale :

Cyclistic a conclu que les membres annuels sont beaucoup plus rentables que les cyclistes occasionnels. Nous voulons donc concevoir une stratégie marketing et une campagne qui nous aident à convertir les cyclistes occasionnels en membres annuels.

Mes principaux partenaires :

1. Lily Moreno : la directrice du marketing et ma responsable directe. Moreno a lancé cette stratégie. La première partie prenante à qui la livrer.
2. L'équipe de direction : pour que l'idée de Moreno fonctionne, l'équipe de direction doit approuver nos recommandations, elles doivent donc être étayées par des informations convaincantes et des visualisations de données professionnelles.

La Phase de Préparation

Source des données:

Les données originales concernant les 12 derniers mois sur l'utilisation des vélos en libre-service, du 01/10/2023 au 30/09/2024, ont été extraits sous forme de 12 fichiers .csv zippés. Les données sont mises à disposition et sous licence par Motivate International Inc. en vertu de cette licence.

Organisation et description des données:

- ***Convention de dénomination des fichiers***: YYYYMM-divvy-tripdata
- ***Type de fichier***: format csv
- ***Contenu du fichier***: chaque fichier csv se compose de 13 colonnes contenant des informations relatives à l'identifiant du trajet, au type de passager, à l'heure de début et de fin du trajet, au lieu de début et de fin, etc. Le nombre de lignes varie entre 224.073 et 821.276 dans les différents fichiers csv.
- ***Conventions des données***: Les dates → yyyy-MM-dd HH:mm:ss ; Longitude et Lattitude → ±DD.D (ISO 6709)

Créditabilité des données:

L'ensemble de données est fiable, les données sont complètes et exactes pour la période choisie.(Quelques valeurs nulles et quelques erreurs qui vont être ajustées dans la phase du traitement)

Les données sont originales, il s'agit d'informations de première main.

Les données sont complètes, l'ensemble de données contient toutes les informations nécessaires pour répondre à la question.

Les données sont actuelles, les données des passagers des 12 derniers mois ont été utilisées.

Les données sont citées et vérifiées par le département des transports de Chicago.

Sécurité des données:

Les informations personnelles identifiables des utilisateurs ne figurent pas dans les données.

Les fichiers originaux sont sauvegardés dans un dossier séparé.

Limitations des données :

Comme les informations personnelles identifiables des cyclistes sont absents, il ne sera pas possible de savoir si les cyclistes occasionnels vivent dans la zone de service cycliste ou s'ils ont acheté plusieurs pass simples.

La Phase du Traitement:

J'ai utilisé R pour la vérification , le nettoyage des données et aussi pour l'analyse et la visualisation. J'ai travaillé en local avec RStudio Desktop.

- **Version**: RStudio 2024.09.0+375 "Cranberry Hibiscus" Release (c8fc7aee6dc218d5687553f9041c6b1e5ea268ff, 2024-09-16) pour windows Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) RStudio/2024.09.0+375 Chrome/124.0.6367.243 Electron/30.4.0 Safari/537.36, Quarto 1.5.57
- ***Raisons***: Les 12 ensembles de données combinés contiendront plus de 5 millions de lignes de données (5.854.544 lignes). Ce qui dépasse largement la limite de la feuille de calcul Excel, qui est de 1 048 576 lignes. De plus, certains fichiers csv n'ont pas pu être téléchargés sur BigQuery en raison de problèmes de taille de fichier. Ainsi, R est utilisé pour effectuer toutes les tâches d'organisation, de nettoyage, d'analyse et de visualisation des données. J'aurais pu travailler ,en amont, en local avec SQLite Studio pour faire traiter, nettoyer et analyser mes données. Puis, afficher mes visualisations sur Tableau Public.

Mise en place de l'environnement

Ici, j'utilise plusieurs bibliothèques qui aident à lire, nettoyer, organiser et analyser les données.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(janitor)
```

```
##
## Attachement du package : 'janitor'
##
## Les objets suivants sont masqués depuis 'package:stats':
##
##     chisq.test, fisher.test
```

```r
library(skimr)
library(here)
```

```
## here() starts at C:/Users/musta/Documents/R_Projects/Cyclistic Case Study
```

```r
library(hablar)
```

```
##
## Attachement du package : 'hablar'
##
## L'objet suivant est masqué depuis 'package:skimr':
##
##     n_unique
##
## L'objet suivant est masqué depuis 'package:forcats':
##
##     fct
##
## L'objet suivant est masqué depuis 'package:dplyr':
##
##     na_if
##
## L'objet suivant est masqué depuis 'package:tibble':
##
##     num
```

```r
library(readxl)
library(data.table)
```

```
##
## Attachement du package : 'data.table'
##
## Les objets suivants sont masqués depuis 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## Les objets suivants sont masqués depuis 'package:dplyr':
##
##     between, first, last
##
## L'objet suivant est masqué depuis 'package:purrr':
##
##     transpose
```

```r
library(chron)
```

```
##
## Attachement du package : 'chron'
##
## Les objets suivants sont masqués depuis 'package:lubridate':
##
##     days, hours, minutes, seconds, years
```

```r
library(readr)
library(lubridate)
library(magrittr)
```

```
##
## Attachement du package : 'magrittr'
##
## L'objet suivant est masqué depuis 'package:purrr':
##
##     set_names
##
## L'objet suivant est masqué depuis 'package:tidyr':
##
##     extract
```

```r
library(DescTools)
```

```
##
## Attachement du package : 'DescTools'
##
## L'objet suivant est masqué depuis 'package:data.table':
##
##     %like%
```

```r
library(metR)
```

```
##
## Attachement du package : 'metR'
##
## L'objet suivant est masqué depuis 'package:purrr':
##
##     cross
```

Importation des données

```r
# 1. Télécharger le fichier ZIP
url <- "https://divvy-tripdata.s3.amazonaws.com/202310-divvy-tripdata.zip"
destfile <- "202310-divvy-tripdata.zip"
download.file(url, destfile)

# 2. Lister les fichiers présents dans l'archive ZIP
fichiers_dans_zip <- unzip(destfile, list = TRUE)$Name

# 3. Filtrer les fichiers à exclure (ceux dans _MACOSX)
fichiers_a_extraire <- fichiers_dans_zip[!grepl("^__MACOSX", fichiers_dans_zip)]

# 4. Extraire uniquement les fichiers filtrés
unzip(destfile, files = fichiers_a_extraire, exdir = "Target")
```

```r
# 5. Lister et charger le fichier CSV
data_a_inspecter <- read.csv("Target/202310-divvy-tripdata.csv")

# 6. Lire les 10 premières lignes du dataframe
head(data_a_inspecter, 10)
```

Importation d'un seul fichier csv en vue de l'inspecter

```
##             ride_id rideable_type          started_at            ended_at
## 1  4449097279F8BBE7  classic_bike 2023-10-08 10:36:26 2023-10-08 10:49:19
## 2  9CF060543CA7B439 electric_bike 2023-10-11 17:23:59 2023-10-11 17:36:08
## 3  667F21F4D6BDE69C electric_bike 2023-10-12 07:02:33 2023-10-12 07:06:53
## 4  F92714CC6B019B96  classic_bike 2023-10-24 19:13:03 2023-10-24 19:18:29
## 5  5E34BA5DE945A9CC  classic_bike 2023-10-09 18:19:26 2023-10-09 18:30:56
## 6  F7D7420AFAC53CD9 electric_bike 2023-10-04 17:10:59 2023-10-04 17:25:21
## 7  870B2D4CD112D7B7 electric_bike 2023-10-31 17:32:20 2023-10-31 17:44:20
## 8  D9179D36E32D456C  classic_bike 2023-10-02 18:51:51 2023-10-02 18:57:09
## 9  F8E131281F722FEF  classic_bike 2023-10-17 08:28:18 2023-10-17 08:50:03
## 10 91938B71748FA405  classic_bike 2023-10-17 19:17:38 2023-10-17 19:32:23
##                      start_station_name start_station_id
## 1  Orleans St & Chestnut St (NEXT Apts)              620
## 2             Desplaines St & Kinzie St     TA1306000003
## 3  Orleans St & Chestnut St (NEXT Apts)              620
## 4             Desplaines St & Kinzie St     TA1306000003
## 5             Desplaines St & Kinzie St     TA1306000003
## 6  Orleans St & Chestnut St (NEXT Apts)              620
## 7  Orleans St & Chestnut St (NEXT Apts)              620
## 8             Desplaines St & Kinzie St     TA1306000003
## 9                 Calumet Ave & 18th St            13102
## 10                Wolcott Ave & Polk St     TA1309000064
##               end_station_name end_station_id start_lat start_lng  end_lat
## 1  Sheffield Ave & Webster Ave   TA1309000033  41.89820 -87.63754 41.92154
## 2  Sheffield Ave & Webster Ave   TA1309000033  41.88864 -87.64441 41.92154
## 3        Franklin St & Lake St   TA1307000111  41.89807 -87.63751 41.88584
## 4        Franklin St & Lake St   TA1307000111  41.88872 -87.64445 41.88584
## 5        Franklin St & Lake St   TA1307000111  41.88872 -87.64445 41.88584
## 6  Sheffield Ave & Webster Ave   TA1309000033  41.89812 -87.63753 41.92154
## 7  Sheffield Ave & Webster Ave   TA1309000033  41.89818 -87.63755 41.92154
## 8        Franklin St & Lake St   TA1307000111  41.88872 -87.64445 41.88584
## 9          Morgan St & Polk St   TA1307000130  41.85762 -87.61941 41.87174
## 10         Morgan St & Polk St   TA1307000130  41.87126 -87.67369 41.87174
##      end_lng member_casual
## 1  -87.65382        member
## 2  -87.65382        member
## 3  -87.63550        member
## 4  -87.63550        member
## 5  -87.63550        member
## 6  -87.65382        member
## 7  -87.65382        member
## 8  -87.63550        casual
## 9  -87.65103        member
## 10 -87.65103        member
```

```r
# 7. Lister les colonnes du dataframe
colnames(data_a_inspecter)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"
##  [4] "ended_at"           "start_station_name" "start_station_id"
##  [7] "end_station_name"   "end_station_id"     "start_lat"
## [10] "start_lng"          "end_lat"            "end_lng"
## [13] "member_casual"
```

Inspection des données du fichier en question On remarque que l'identifiant des trajets "ride_id" est sous forme d'UUID (Universally Unique Identifier) composé de 16 caractères presque 6 millions de fois. Ce qui risque d'alourdir le dataframe final. Cette colonne est sans intérêt pour notre analyse. De même, les colonnes "start_station_id et"end_station_id" ne présentent pas d'utilité. Nous allons donc supprimer ces 3 colonnes.

```r
# Supprimer les colonnes "ride_id", "start_station_id" et "end_station_id"
data_a_inspecter <- data_a_inspecter[ , !(names(data_a_inspecter) %in% c("ride_id", "start_station_id",
head(data_a_inspecter)
```

```
##   rideable_type          started_at            ended_at
## 1  classic_bike 2023-10-08 10:36:26 2023-10-08 10:49:19
## 2 electric_bike 2023-10-11 17:23:59 2023-10-11 17:36:08
## 3 electric_bike 2023-10-12 07:02:33 2023-10-12 07:06:53
## 4  classic_bike 2023-10-24 19:13:03 2023-10-24 19:18:29
## 5  classic_bike 2023-10-09 18:19:26 2023-10-09 18:30:56
## 6 electric_bike 2023-10-04 17:10:59 2023-10-04 17:25:21
##                     start_station_name            end_station_name start_lat
## 1 Orleans St & Chestnut St (NEXT Apts) Sheffield Ave & Webster Ave  41.89820
## 2            Desplaines St & Kinzie St Sheffield Ave & Webster Ave  41.88864
## 3 Orleans St & Chestnut St (NEXT Apts)       Franklin St & Lake St  41.89807
## 4            Desplaines St & Kinzie St       Franklin St & Lake St  41.88872
## 5            Desplaines St & Kinzie St       Franklin St & Lake St  41.88872
## 6 Orleans St & Chestnut St (NEXT Apts) Sheffield Ave & Webster Ave  41.89812
##   start_lng  end_lat   end_lng member_casual
## 1 -87.63754 41.92154 -87.65382        member
## 2 -87.64441 41.92154 -87.65382        member
## 3 -87.63751 41.88584 -87.63550        member
## 4 -87.64445 41.88584 -87.63550        member
## 5 -87.64445 41.88584 -87.63550        member
## 6 -87.63753 41.92154 -87.65382        member
```

On voudrait afficher les valeurs uniques des variables catégorielles "rideable_type" (type de vélo) et "member_casual" (nature du client). On affichera également le nombre de valeurs uniques pour les stations de départ et d'arrivée.

```r
# Afficher les valeurs uniques des colonnes "rideable_type" et "member_casual"
cat("Les valeurs uniques du type des vélos sont :", unique(data_a_inspecter$rideable_type), "\n")
```

```
## Les valeurs uniques du type des vélos sont : classic_bike electric_bike
```

```r
cat("Les valeurs uniques des types d'utilisateurs sont :", unique(data_a_inspecter$member_casual), "\n"
```

```
## Les valeurs uniques des types d'utilisateurs sont : member casual
```

```r
# Obtenir et afficher le nombre de stations uniques
cat("Le nombre de stations de départ uniques est :", length(unique(data_a_inspecter$start_station_name)
```

```
## Le nombre de stations de départ uniques est : 1190
```

```r
cat("Le nombre de stations d'arrivée uniques est :", length(unique(data_a_inspecter$end_station_name)),
```

```
## Le nombre de stations d'arrivée uniques est : 1175
```

Importation de tous les fichiers, chargement dans des dataframes puis fusion en un seul dataframe final Les données Cyclistic à compter du 10/2023 jusqu'au 09/2024 sont importées et chargées. J'ai créé une fonction d'automatisation puisque les fichiers csv ont les mêmes suffixes dans leurs dénominations.

```r
library(readr)

# Créer une liste des noms de fichiers
file_names <- c("202310", "202311", "202312", "202401", "202402", "202403",
                "202404", "202405", "202406", "202407", "202408", "202409")

# Chemin vers le répertoire contenant les fichiers
file_path <- "Data/"

# Charger les fichiers dans des dataframes individuels
for (i in seq_along(file_names)) {
  # Créer le nom du dataframe dynamiquement
  dataframe_name <- paste0("data_month_", sprintf("%02d", i))

  # Lire le fichier CSV et assigner le dataframe à un nom dynamique
  assign(dataframe_name, read_csv(paste0(file_path, file_names[i], "-divvy-tripdata.csv")), envir = .Gl
}
```

```
## Rows: 537113 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 362518 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
```

```
## dttm (2): started_at, ended_at
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## Rows: 224073 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## Rows: 144873 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## Rows: 223164 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## Rows: 301687 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## Rows: 415025 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## Rows: 609493 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
```

```
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 710721 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 748962 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 755639 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 821276 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Toujours avec une fonction d'automatisation, je compare les noms des colonnes de chacun des dataframes. Même si les noms ne doivent pas nécessairement être dans le même ordre, ils DOIVENT correspondre parfaitement avant d'être fusionnés dans un seul dataframe merged_data.

```r
# Créer une liste des noms de dataframes générés
dataframes <- lapply(1:12, function(i) get(paste0("data_month_", sprintf("%02d", i)), envir = .GlobalEnv

# Vérifier que les noms de colonnes sont identiques pour tous les dataframes
column_names_equal <- sapply(dataframes, colnames)

# Vérifier si toutes les colonnes sont identiques pour tous les dataframes
identical_columns <- all(apply(column_names_equal, 2, function(col) all(col == colnames(dataframes[[1]]]
```

```r
if (identical_columns) {
  # Si les colonnes sont identiques, fusionner tous les dataframes en un seul
  merged_data <- do.call(rbind, dataframes)
  print("Tous les dataframes ont des colonnes identiques. Fusion réussie.")
} else {
  # Afficher les colonnes pour chaque dataframe en cas de non-conformité
  print("Les colonnes diffèrent entre les dataframes. Vérifiez les colonnes ci-dessous :")
  col_diff <- sapply(dataframes, colnames)
  print(col_diff)
}
```

```
## [1] "Tous les dataframes ont des colonnes identiques. Fusion réussie."
```

Tous les dataframes ont des colonnes identiques.La fusion a été faite sans problème.

```r
# Calcul du nombre d'observations
nombre_observations <- nrow(merged_data)
print(nombre_observations)
```

```
## [1] 5854544
```

Les 12 datasets sont alors fusionnés dans un large Dataframe de 5854544 observations.

```r
# Supprimer les colonnes "ride_id", "start_station_id" et "end_station_id"
merged_data <- merged_data[ , !(names(merged_data) %in% c("ride_id", "start_station_id", "end_station_i
head(merged_data)
```

```
## # A tibble: 6 x 10
##   rideable_type started_at          ended_at            start_station_name
##   <chr>         <dttm>              <dttm>              <chr>
## 1 classic_bike  2023-10-08 10:36:26 2023-10-08 10:49:19 Orleans St & Chestnut S~
## 2 electric_bike 2023-10-11 17:23:59 2023-10-11 17:36:08 Desplaines St & Kinzie ~
## 3 electric_bike 2023-10-12 07:02:33 2023-10-12 07:06:53 Orleans St & Chestnut S~
## 4 classic_bike  2023-10-24 19:13:03 2023-10-24 19:18:29 Desplaines St & Kinzie ~
## 5 classic_bike  2023-10-09 18:19:26 2023-10-09 18:30:56 Desplaines St & Kinzie ~
## 6 electric_bike 2023-10-04 17:10:59 2023-10-04 17:25:21 Orleans St & Chestnut S~
## # i 6 more variables: end_station_name <chr>, start_lat <dbl>, start_lng <dbl>,
## #   end_lat <dbl>, end_lng <dbl>, member_casual <chr>
```

```r
# Voir la structure du dataframe final
str(merged_data)
```

```
## tibble [5,854,544 x 10] (S3: tbl_df/tbl/data.frame)
##  $ rideable_type     : chr [1:5854544] "classic_bike" "electric_bike" "electric_bike" "classic_bike"
##  $ started_at        : POSIXct[1:5854544], format: "2023-10-08 10:36:26" "2023-10-11 17:23:59" ...
##  $ ended_at          : POSIXct[1:5854544], format: "2023-10-08 10:49:19" "2023-10-11 17:36:08" ...
##  $ start_station_name: chr [1:5854544] "Orleans St & Chestnut St (NEXT Apts)" "Desplaines St & Kinzi
##  $ end_station_name  : chr [1:5854544] "Sheffield Ave & Webster Ave" "Sheffield Ave & Webster Ave" "
##  $ start_lat         : num [1:5854544] 41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num [1:5854544] -87.6 -87.6 -87.6 -87.6 -87.6 ...
##  $ end_lat           : num [1:5854544] 41.9 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num [1:5854544] -87.7 -87.7 -87.6 -87.6 -87.6 ...
##  $ member_casual     : chr [1:5854544] "member" "member" "member" "member" ...
```

J'ai supprimé les 3 colonnes inutiles pour alléger la table des données

Pour assurer une bonne analyse, on va vérifier s'il existe encore des valeurs nulles parmi nos données

```
# Compter les valeurs nulles pour chaque colonne du dataframe
na_counts <- merged_data %>%
  summarise_all(~sum(is.na(.)))

print(na_counts)
```

```
## # A tibble: 1 x 10
##   rideable_type started_at ended_at start_station_name end_station_name
##           <int>      <int>    <int>              <int>            <int>
## 1             0          0        0            1056535          1091792
## # i 5 more variables: start_lat <int>, start_lng <int>, end_lat <int>,
## #   end_lng <int>, member_casual <int>
```

```
# Supprimer les lignes avec des NA dans end_lat ou end_lng
merged_data <- merged_data %>%
  filter(!is.na(end_lat) & !is.na(end_lng))

# Calcul du nombre d'observations
nombre_observations <- nrow(merged_data)
print(nombre_observations)
```

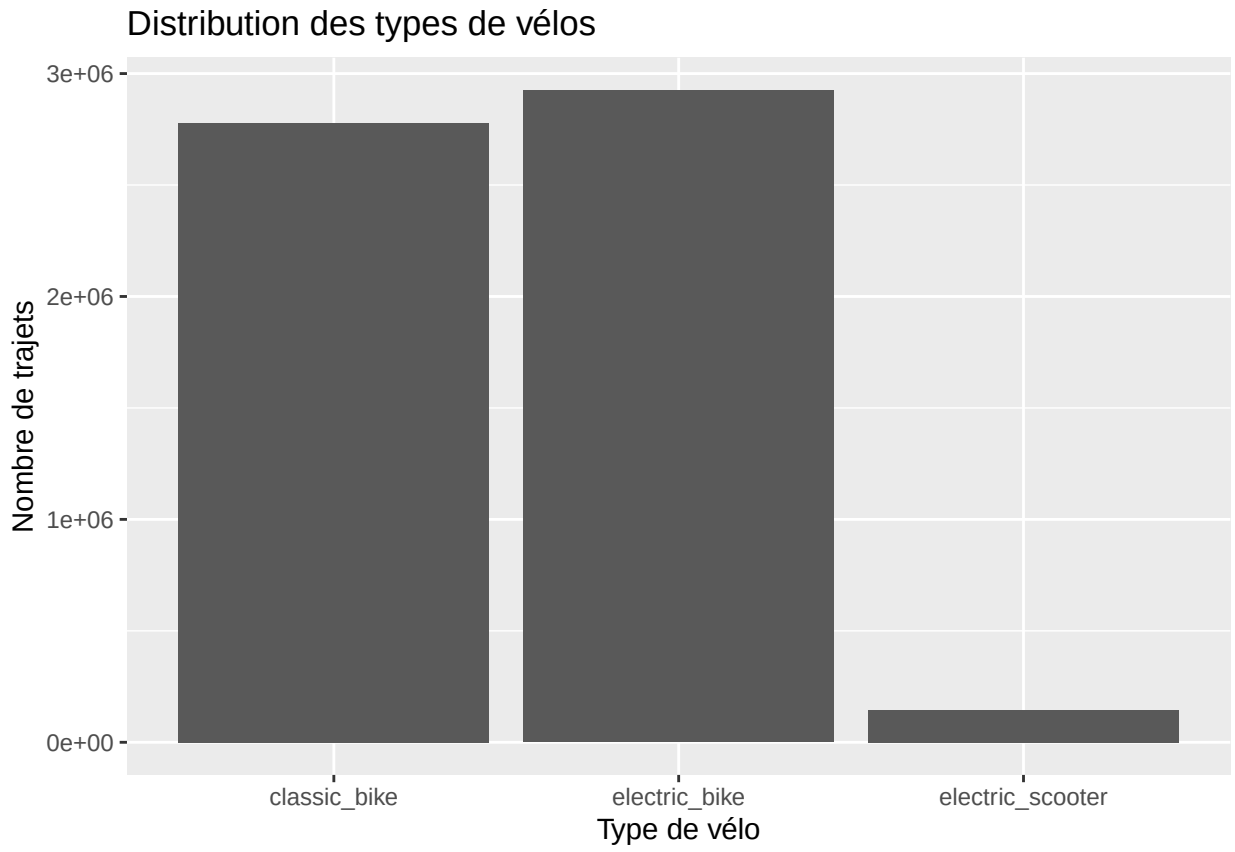
```
## [1] 5847103
```

La Phase d'Analyse :

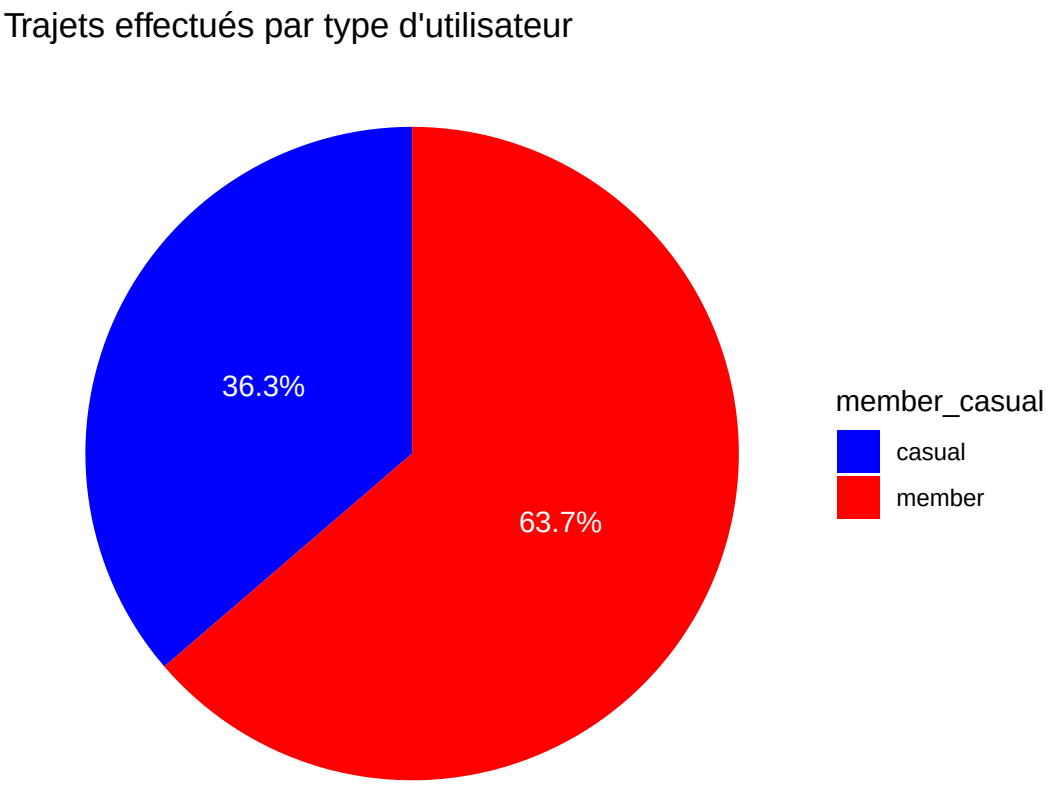
Aperçu sur les modalités d'utilisation des vélos

On commence la phase d'analyse par deux petites visualisations pour voir la distribution des variables "rideable_type" et "member_casual". Ce qui nous donnera pour ce début d'analyse une idée sur le pourcentage de type de vélos utilisés et le pourcentage de trajets effectués par chacun des deux types d'utilisateurs.

```
# Installation et chargement de ggplot2
#install.packages("ggplot2")
library(ggplot2)

# Diagramme en barres pour "rideable_type"
ggplot(merged_data, aes(x = rideable_type)) +
  geom_bar() +
  labs(x = "Type de vélo", y = "Nombre de trajets", title = "Distribution des types de vélos")
```




```r
# Calculer les pourcentages
counts <- table(merged_data$member_casual)
percentages <- round(prop.table(counts) * 100, 1)

# Créer un dataframe pour ggplot et renommer les colonnes correctement
data_pie <- data.frame(
  member_casual = names(percentages),
  percentage = as.numeric(percentages)
)

# Créer le diagramme circulaire
ggplot(data = data_pie, aes(x = "", y = percentage, fill = member_casual)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start = 0) +
  labs(title = "Trajets effectués par type d'utilisateur") +
  theme_void() +
  geom_text(aes(label = paste0(percentage, "%")), position = position_stack(vjust = 0.5), color = "white
  scale_fill_manual(values = c("blue", "red"))
```



On remarque que les deux tiers des trajets effectués durant la période étudiée sont faits par des abonnés.

Manipulation des dates pour une analyse détaillée des trajets (durée, jour de la semaine, heure de la journée..)

Trajets s'étendant sur plusieurs dates Je vais vérifier si certaines utilisations des vélos comprennent des trajets qui s'étendent sur plusieurs dates.

```
# Convertir les colonnes "started_at" et "ended_at" en format date-heure
merged_data$started_at <- as.POSIXct(merged_data$started_at, format = "%Y-%m-%d %H:%M:%S")
merged_data$ended_at <- as.POSIXct(merged_data$ended_at, format = "%Y-%m-%d %H:%M:%S")

# Extraire uniquement les dates (sans les heures) pour les deux colonnes
merged_data$start_date <- as.Date(merged_data$started_at)
merged_data$end_date <- as.Date(merged_data$ended_at)

# Vérifier si les dates de début et de fin sont différentes
multiple_dates <- merged_data$start_date != merged_data$end_date

# Compter le nombre de trajets qui s'étendent sur plusieurs dates
trajets_multi_dates <- sum(multiple_dates)

# Afficher le nombre de trajets avec des dates différentes
cat("Nombre de trajets s'étendant sur plusieurs dates :", trajets_multi_dates, "\n")
```

```
## Nombre de trajets s'étendant sur plusieurs dates : 22946
```

```r
# Optionnel : afficher les lignes concernées
data_multi_dates <- merged_data[multiple_dates, ]
head(data_multi_dates)
```

```
## # A tibble: 6 x 12
##   rideable_type started_at          ended_at            start_station_name
##   <chr>         <dttm>              <dttm>              <chr>
## 1 classic_bike  2023-10-08 23:55:35 2023-10-09 00:21:30 Western Ave & Division ~
## 2 classic_bike  2023-10-06 23:53:13 2023-10-07 00:03:15 Wells St & Concord Ln
## 3 classic_bike  2023-10-19 17:19:58 2023-10-20 02:31:58 Broadway & Argyle St
## 4 classic_bike  2023-10-02 23:17:41 2023-10-03 00:20:07 Damen Ave & Cortland St
## 5 electric_bike 2023-10-26 23:55:21 2023-10-27 00:04:24 Aberdeen St & Monroe St
## 6 classic_bike  2023-10-03 19:51:11 2023-10-04 12:38:37 Desplaines St & Kinzie ~
## # i 8 more variables: end_station_name <chr>, start_lat <dbl>, start_lng <dbl>,
## #   end_lat <dbl>, end_lng <dbl>, member_casual <chr>, start_date <date>,
## #   end_date <date>
```

Environ 0.5% des trajets d'étendent sur plusieurs dates (généralement deux dates). Il s'agit d'usages nocturnes des vélos.Je vais donc créer une nouvelle colonne (multi_date_trip) pour marquer les trajets qui s'étendent sur plusieurs dates. J'ajouterai également une colonne pour calculer la durée de chaque trajet

```r
# Ajouter une colonne "multi_date_trip" pour indiquer si un trajet s'étend sur plusieurs dates
merged_data$multi_date_trip <- as.Date(merged_data$started_at) != as.Date(merged_data$ended_at)

# Calculer la durée de chaque trajet en secondes
merged_data$trip_duration_sec <- as.numeric(difftime(merged_data$ended_at, merged_data$started_at, units

# Convertir la durée en heures, minutes et secondes
merged_data$trip_duration_hms <- format(as.POSIXct(merged_data$trip_duration_sec, origin = "1970-01-01"

# Afficher les premières lignes pour vérifier les nouvelles colonnes
head(merged_data[, c("started_at", "ended_at",  "trip_duration_sec", "trip_duration_hms")])
```

```
## # A tibble: 6 x 4
##   started_at          ended_at            trip_duration_sec trip_duration_hms
##   <dttm>              <dttm>                          <dbl> <chr>
## 1 2023-10-08 10:36:26 2023-10-08 10:49:19               773 00:12:53
## 2 2023-10-11 17:23:59 2023-10-11 17:36:08               729 00:12:09
## 3 2023-10-12 07:02:33 2023-10-12 07:06:53               260 00:04:20
## 4 2023-10-24 19:13:03 2023-10-24 19:18:29               326 00:05:26
## 5 2023-10-09 18:19:26 2023-10-09 18:30:56               690 00:11:30
## 6 2023-10-04 17:10:59 2023-10-04 17:25:21               862 00:14:22
```

Trajets à durée négative Je vais vérifier s'il existe des anomalies relatives à des erreurs dans les données sous forme de durée négative. S'il en existe, on les supprime en créant un nouveau dataframe merged_data_v1.

```r
# Vérifier s'il existe des trajets avec une durée négative
negative_duration_trips <- merged_data[merged_data$trip_duration_sec < 0, ]

# Afficher un aperçu des trajets ayant une durée négative, s'il y en a
```

```r
if (nrow(negative_duration_trips) > 0) {
  cat("Nombre de trajets avec une durée négative :", nrow(negative_duration_trips), "\n")
  print(head(negative_duration_trips))

  # Créer un nouveau dataframe sans trajets à durée négative
  merged_data_v1 <- merged_data[merged_data$trip_duration_sec >= 0, ]
  cat("Nouveau dataframe créé : merged_data_v1 avec des trajets à durée valide.\n")
} else {
  cat("Aucun trajet avec une durée négative détecté. Le dataframe merged_data_v1 ne sera pas créé.\n")
}
```

```
## Nombre de trajets avec une durée négative : 294
## # A tibble: 6 x 15
##   rideable_type started_at          ended_at            start_station_name
##   <chr>         <dttm>              <dttm>              <chr>
## 1 electric_bike 2023-10-01 00:15:12 2023-10-01 00:15:11 Leavitt St & Archer Ave
## 2 electric_bike 2023-10-19 13:35:27 2023-10-19 13:35:26 Federal St & Polk St
## 3 electric_bike 2023-10-08 17:38:12 2023-10-08 17:38:09 Wilton Ave & Belmont Ave
## 4 electric_bike 2023-10-27 13:18:23 2023-10-27 13:18:22 Sedgwick St & Huron St
## 5 electric_bike 2023-10-11 07:30:33 2023-10-11 07:30:31 Wabash Ave & Grand Ave
## 6 electric_bike 2023-10-14 22:26:30 2023-10-14 22:26:29 Millennium Park
## # i 11 more variables: end_station_name <chr>, start_lat <dbl>,
## #   start_lng <dbl>, end_lat <dbl>, end_lng <dbl>, member_casual <chr>,
## #   start_date <date>, end_date <date>, multi_date_trip <lgl>,
## #   trip_duration_sec <dbl>, trip_duration_hms <chr>
## Nouveau dataframe créé : merged_data_v1 avec des trajets à durée valide.
```

On va donc éliminer tous les trajets à durée négative en créant une nouvelle dataframe merged_data_v1.

Création de nouvelles colonnes relatives aux mois, jour et l'année Je vais créer des nouvelles colonnes dans mon dataframe merged_data pour analyser les données de manière plus détaillée. Ainsi, je pourrais filtrer, grouper et visualiser les données en fonction du mois, du jour, de l'année et du jour de la semaine.

```r
# Créer une nouvelle colonne "year" pour l'année
merged_data_v1$year <- format(merged_data_v1$started_at, "%Y")

# Créer une nouvelle colonne "month" pour le mois
merged_data_v1$month <- format(merged_data_v1$started_at, "%m")

# Créer une nouvelle colonne "day" pour le jour du mois
merged_data_v1$day <- format(merged_data_v1$started_at, "%d")

# Créer une nouvelle colonne "weekday" pour le jour de la semaine (lundi, mardi, etc.)
# La fonction "weekdays" retourne le jour de la semaine complet
merged_data_v1$weekday <- weekdays(merged_data_v1$started_at)

# Créer une nouvelle colonne "week" pour le numéro de la semaine dans l'année
merged_data_v1$week <- format(merged_data_v1$started_at, "%U")

# Créer une nouvelle colonne "hour" pour l'heure du début du trajet
merged_data_v1$hour <- format(merged_data_v1$started_at, "%H")
```

```r
# Vérifier les nouvelles colonnes
head(merged_data_v1[, c("started_at", "year", "month", "day", "weekday", "week", "hour")])
```

```
## # A tibble: 6 x 7
##   started_at          year  month day   weekday  week  hour
##   <dttm>              <chr> <chr> <chr> <chr>    <chr> <chr>
## 1 2023-10-08 10:36:26 2023  10    08    dimanche 41    10
## 2 2023-10-11 17:23:59 2023  10    11    mercredi 41    17
## 3 2023-10-12 07:02:33 2023  10    12    jeudi    41    07
## 4 2023-10-24 19:13:03 2023  10    24    mardi    43    19
## 5 2023-10-09 18:19:26 2023  10    09    lundi    41    18
## 6 2023-10-04 17:10:59 2023  10    04    mercredi 40    17
```

analyse descriptive de la durée du trajet

```r
library(dplyr)
merged_data_v1 %>%
  summarise(max(trip_duration_sec),min(trip_duration_sec),mean(trip_duration_sec))
```

```
## # A tibble: 1 x 3
##   `max(trip_duration_sec)` `min(trip_duration_sec)` `mean(trip_duration_sec)`
##                      <dbl>                    <dbl>                     <dbl>
## 1                    90562                        0                      926.
```

J'ai remarqué qu'ils existent des trajets à durée nulle. Les durées nulles (trip_duration_sec = 0) peuvent indiquer plusieurs choses :

- Erreurs de saisie ou problèmes de collecte de données : Les trajets avec une durée nulle peuvent être le résultat d'une erreur technique ou d'une saisie incorrecte des informations.
- Trajets annulés ou non commencés : Les utilisateurs ont peut-être réservé un vélo, mais ont annulé leur trajet immédiatement, ou n'ont pas réussi à démarrer correctement le trajet.
- Retours immédiats : Les vélos peuvent avoir été pris et retournés sans aucun déplacement réel.

Je supprime, alors, les trajets avec une durée nulle:

```r
# Compter le nombre de trajets avec une durée nulle
count_null_duration <- merged_data_v1 %>%
  filter(trip_duration_sec == 0) %>%
  summarise(count = n())

print(count_null_duration)
```

```
## # A tibble: 1 x 1
##   count
##   <int>
## 1   749
```

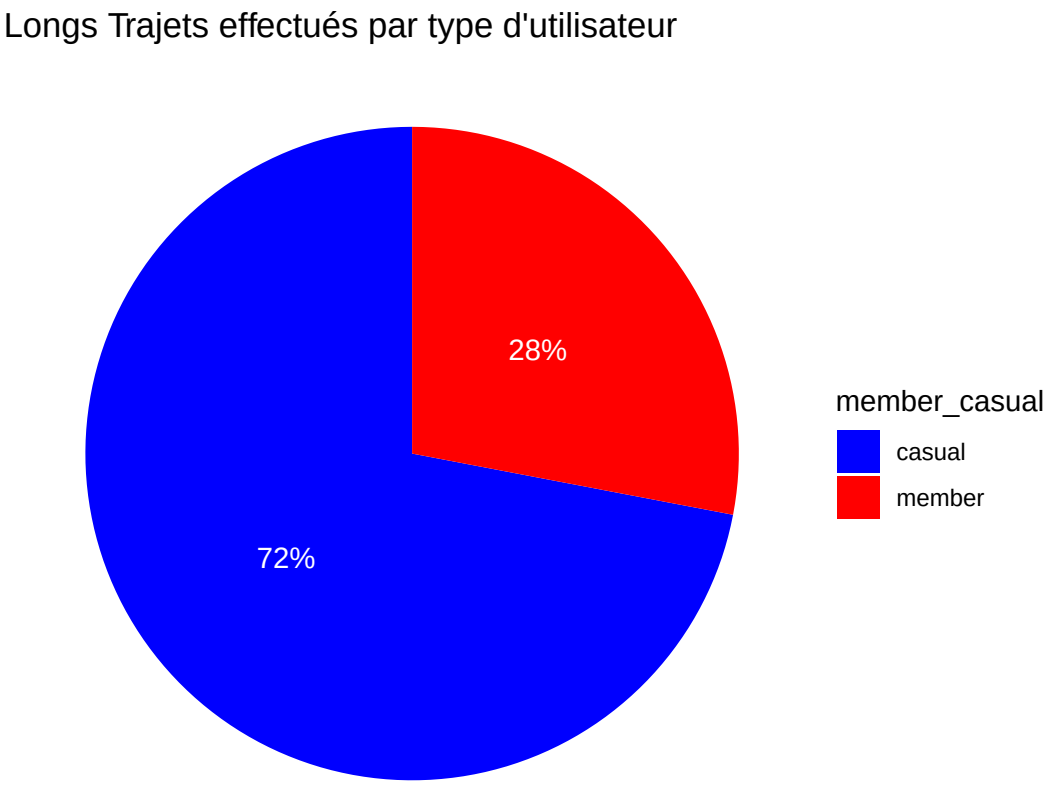
```r
# Supprimer les trajets avec une durée nulle
merged_data_v1 <- merged_data_v1 %>%
  filter(trip_duration_sec > 0)
```

Je vais extraire les trajets dépassant 12 heures de mon nouveau dataframe merged_data_v1. Pour cela, je crée un sous-dataframe en filtrant les lignes où la colonne trip_duration_sec dépasse le seuil de 12 heures, soit 43 200 secondes (12 * 3600).

```r
# Filtrer les trajets dépassant 12 heures (12 * 3600 = 43200 secondes)
long_trips <- merged_data_v1 %>%
  filter(trip_duration_sec > 43200)

# Afficher les premiers trajets qui dépassent 12 heures pour vérifier
head(long_trips)
```

```
## # A tibble: 6 x 21
##   rideable_type started_at          ended_at            start_station_name
##   <chr>         <dttm>              <dttm>              <chr>
## 1 classic_bike  2023-10-03 19:51:11 2023-10-04 12:38:37 Desplaines St & Kinzie ~
## 2 classic_bike  2023-10-29 11:07:48 2023-10-30 07:15:11 Wells St & Concord Ln
## 3 classic_bike  2023-10-29 11:17:42 2023-10-30 07:15:11 Kingsbury St & Kinzie St
## 4 classic_bike  2023-10-21 00:56:56 2023-10-21 18:15:36 Elston Ave & Wabansia A~
## 5 classic_bike  2023-10-21 12:41:45 2023-10-22 10:52:32 Clark St & Newport St
## 6 classic_bike  2023-10-23 02:54:33 2023-10-23 23:37:43 Glenwood Ave & Morse Ave
## # i 17 more variables: end_station_name <chr>, start_lat <dbl>,
## #   start_lng <dbl>, end_lat <dbl>, end_lng <dbl>, member_casual <chr>,
## #   start_date <date>, end_date <date>, multi_date_trip <lgl>,
## #   trip_duration_sec <dbl>, trip_duration_hms <chr>, year <chr>, month <chr>,
## #   day <chr>, weekday <chr>, week <chr>, hour <chr>
```

```r
# Calculer les pourcentages pour les longs trajets
counts_lt <- table(long_trips$member_casual)
percentages_lt <- round(prop.table(counts_lt) * 100, 1)

# Créer un dataframe pour ggplot et renommer les colonnes correctement
data_pie_lt <- data.frame(
  member_casual = names(percentages_lt),
  percentage_lt = as.numeric(percentages_lt)
)

# Créer le diagramme circulaire
ggplot(data = data_pie_lt, aes(x = "", y = percentage_lt, fill = member_casual)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start = 0) +
  labs(title = "Longs Trajets effectués par type d'utilisateur") +
  theme_void() +
  geom_text(aes(label = paste0(percentage_lt, "%")), position = position_stack(vjust = 0.5), color = "wh
  scale_fill_manual(values = c("blue", "red"))
```



On remarque que les longs trajets sont effectués plus souvent par les utilisateurs occasionnels. Ce qui laisse entendre qu'il s'agit probablement de touristes ou d'habitants de Chicago souscrivants à des formules d'abonnements pour une journée entière.

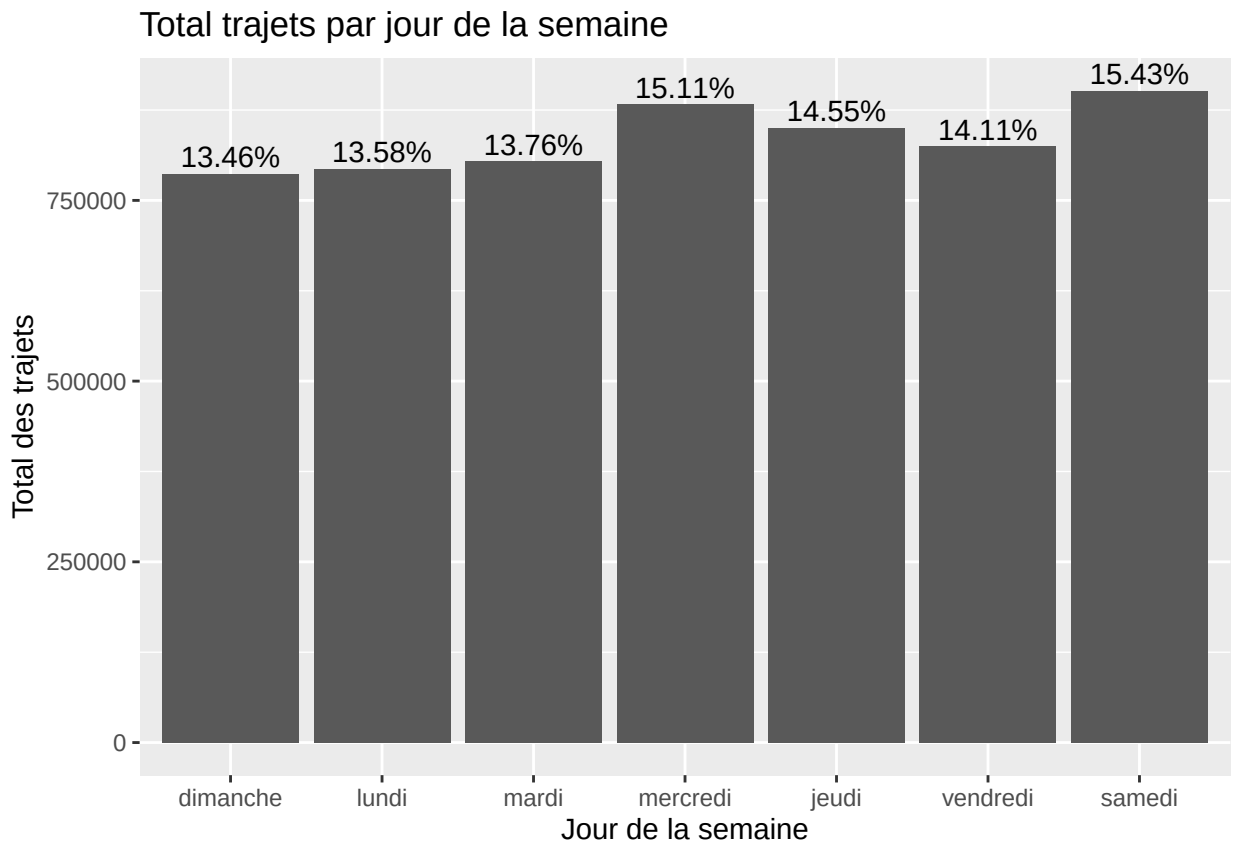
```
# Calculer et afficher le nombre de trajets dépassant 12 heures
cat("Le nombre de trajets dépassant 12 heures est :", nrow(long_trips), "\n")
```

```
## Le nombre de trajets dépassant 12 heures est : 2998
```

Pourcentage de trajets effectués par type client dans les 7 jours de la semaine

```
# Ordre souhaité des jours de la semaine
jours_semaine <- c("dimanche", "lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi")

# Compter le nombre de trajets par jour de la semaine et ordonner
day_counts <- merged_data_v1 %>%
  mutate(weekday = factor(weekday, levels = jours_semaine)) %>%  # Ordonner les niveaux du facteur
  group_by(weekday) %>%
  summarise(total_rides = n()) %>%
  arrange(weekday)  # Trier par ordre croissant des niveaux du facteur

# Calculer les pourcentages
day_counts$percent <- prop.table(day_counts$total_rides) * 100
```

```r
# Créer le graphique
ggplot(day_counts, aes(x = weekday, y = total_rides)) +
  geom_col() +
  geom_text(aes(label = paste0(round(percent,2), "%")), vjust = -0.3) +
  labs(title = "Total trajets par jour de la semaine",
       x = "Jour de la semaine",
       y = "Total des trajets")
```



On remarque qu'il y a une certaine régularité quant au nombre total de trajet par jour de la semaine, avec un léger dépassement pour les jours de Semedi et Mercredi. Cette légère augmentation pour le samedi peut s'expliquer par le fait que c'est un jour de repos pour la majorité, il est logique de voir une augmentation de l'utilisation des vélos pour des activités de loisirs, des balades, des sorties en famille, etc. Alors que le mercredi, étant le jour du milieu de la semaine, peut représenter un moment de pause pour certains, favorisant ainsi l'utilisation du vélo pour des déplacements plus courts ou des activités physiques.

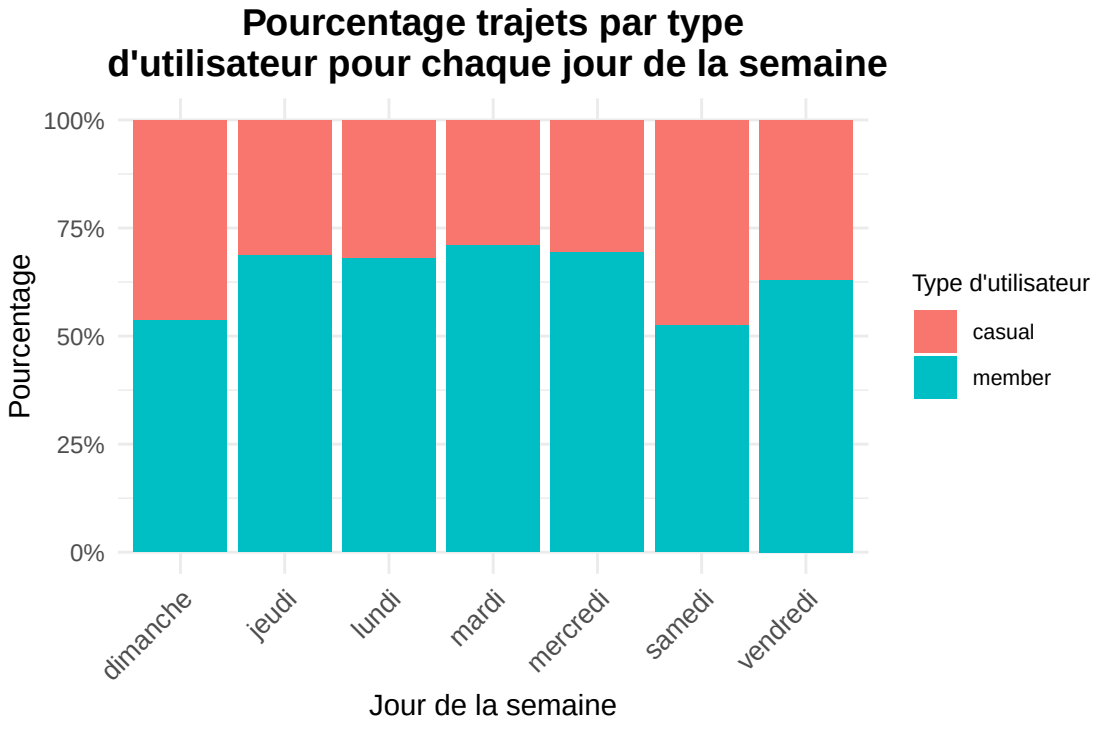
Après avoir visualiser la distribution de la totalité des trajets par les jours de la semaine, on va affiner cette distribution par type d'utilsateur:

```r
# Charger les bibliothèques nécessaires
library(dplyr)
library(lubridate)
library(ggplot2)


#  Calculer le pourcentage d'utilisation par type d'utilisateur pour chaque jour de la semaine
data_summary <- merged_data_v1 %>%
```

```r
  group_by(weekday, member_casual) %>%
  summarise(count = n()) %>%
  mutate(percentage = (count / sum(count)) * 100)
```

```
## `summarise()` has grouped output by 'weekday'. You can override using the
## `.groups` argument.
```

```r
# Créer les graphiques pour chaque jour de la semaine
ggplot(data_summary, aes(x = weekday, y = percentage, fill = member_casual)) +
  geom_bar(stat = "identity", position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  labs(title = "Pourcentage trajets par type\n d'utilisateur pour chaque jour de la semaine",
       x = "Jour de la semaine", y = "Pourcentage", fill = "Type d'utilisateur") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1, size = 10),  # Taille plus grande pour les labels
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5),  # Taille du titre augmentée
    legend.title = element_text(size = 9),
    legend.text = element_text(size = 8),
    plot.margin = margin(1, 1, 1, 1, "cm")  # Marges plus homogènes
  )
```

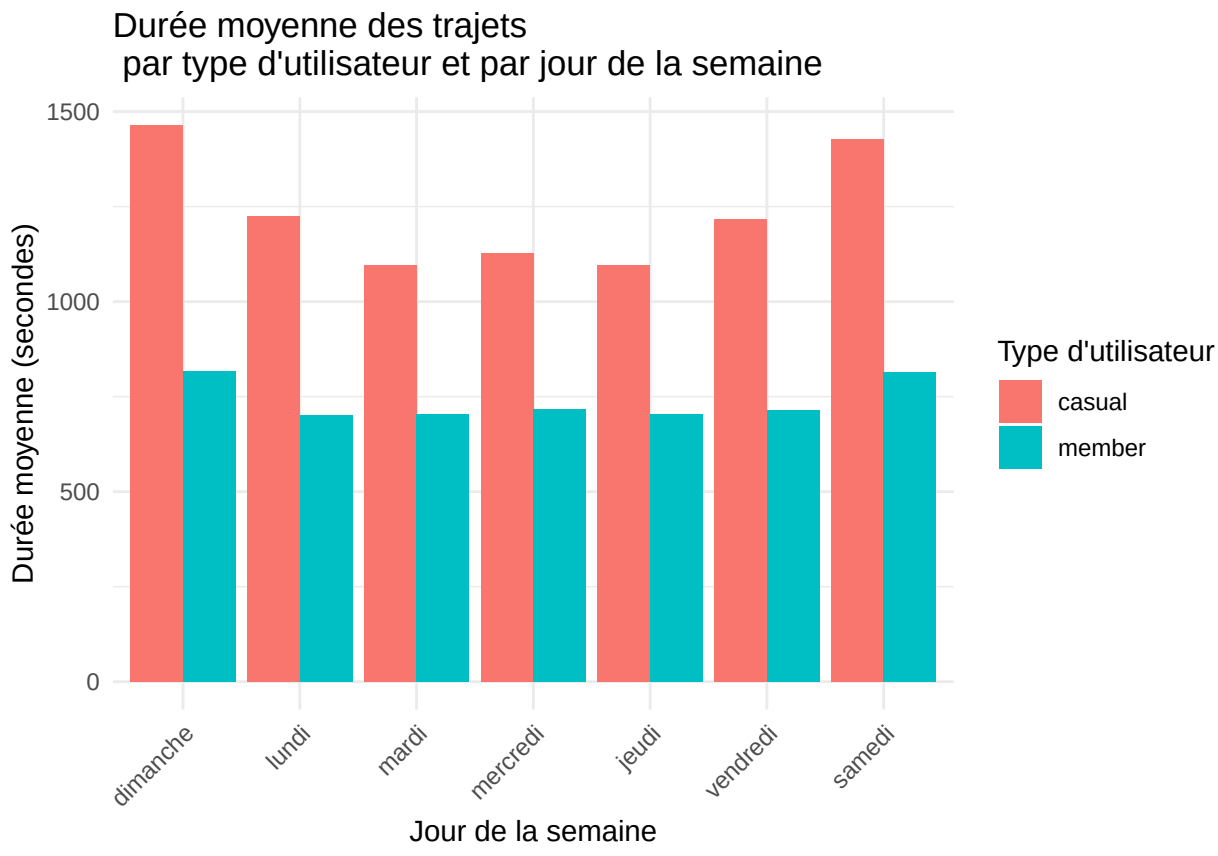


On remarque bien que les trajets sont effectués à hauteur d'environ 30 % par les clients occasionnels au cours des jours de la semaine. Alors que la proportion de trajets pour ce type d'utilisateur augmente à presque 50% au cours des jours du week-end.

```r
# Calculer la durée moyenne des trajets par jour de la semaine et par type d'utilisateur
avg_duration_per_day <- merged_data_v1 %>%
  mutate(weekday = factor(weekday, levels = jours_semaine)) %>%
  group_by(weekday, member_casual) %>%
  summarise(average_duration = mean(trip_duration_sec, na.rm = TRUE)) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'weekday'. You can override using the
## `.groups` argument.
```

```r
# Créer le graphique
ggplot(avg_duration_per_day, aes(x = weekday, y = average_duration, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(title = "Durée moyenne des trajets\n par type d'utilisateur et par jour de la semaine",
       x = "Jour de la semaine",
       y = "Durée moyenne (secondes)",
       fill = "Type d'utilisateur") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

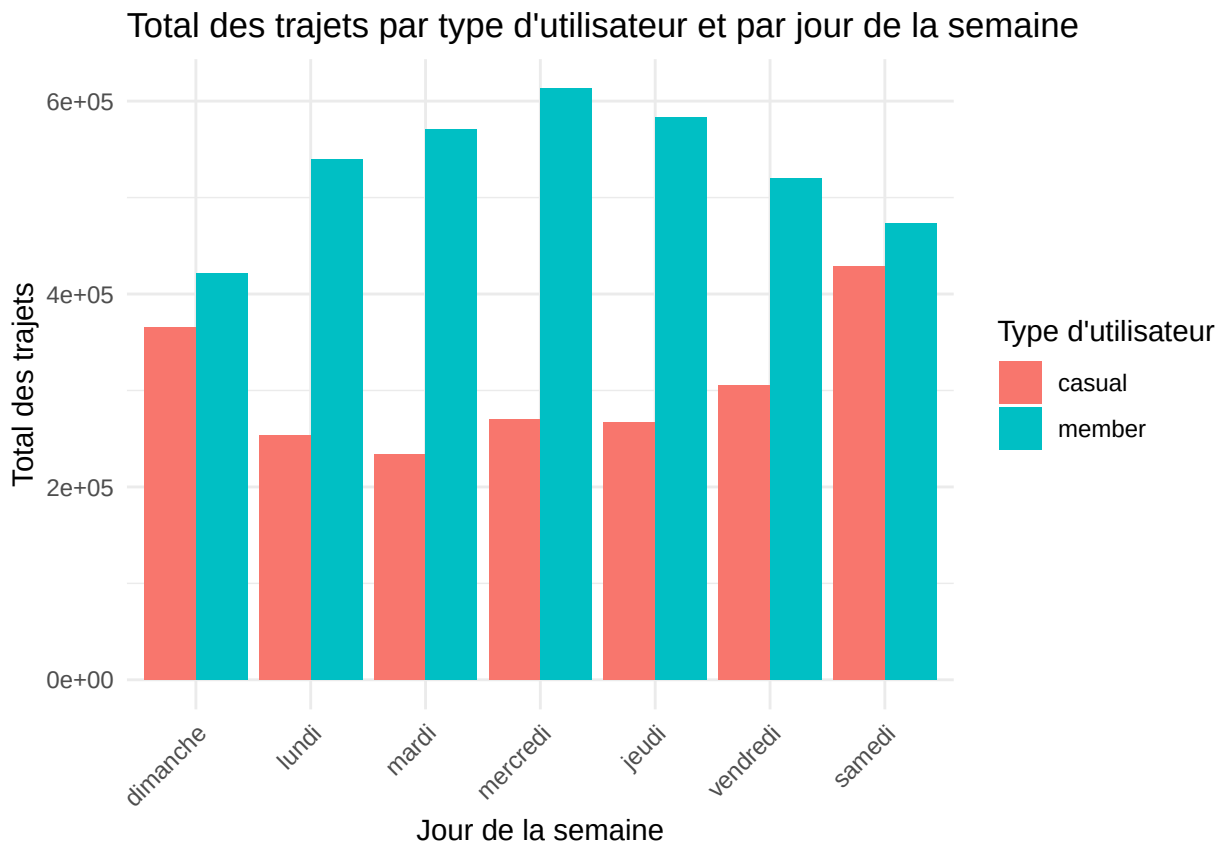


Le graphique montre que les cyclistes occasionnels louent des vélos pour des durées plus longues, notamment le dimanche, le samedi, puis le vendredi et le lundi. Les membres affichent un comportement d'utilisation régulier, et ils ont également tendance à rouler un peu plus longtemps le week-end.

Ici, le nombre de trajets par jour pour chaque type de cycliste est représenté.

```r
merged_data_v1 %>%
  mutate(weekday = factor(weekday, levels = jours_semaine)) %>%
  group_by(member_casual, weekday) %>%
  summarise(total_rides = n()
            ,average_duration = mean(trip_duration_sec)) %>%
  arrange(member_casual, weekday)  %>%
  ggplot(aes(x = weekday, y = total_rides, fill = member_casual)) +
  geom_col(position = "dodge")+
  labs(title = "Total des trajets par type d'utilisateur et par jour de la semaine",
       x = "Jour de la semaine",
       y = "Total des trajets",
       fill = "Type d'utilisateur") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## `summarise()` has grouped output by 'member_casual'. You can override using the
## `.groups` argument.
```



Étonnamment, et contrairement au scénario précédent, les membres effectuent plus de trajets et ont donc un nombre plus élevé de trajets chaque jour de la semaine. On remarque que le samedi et le dimanche connaissent un certain équilibre dans le nombre de trajets effectués par les deux types d'utilisateurs (abonnés, occasionnels).

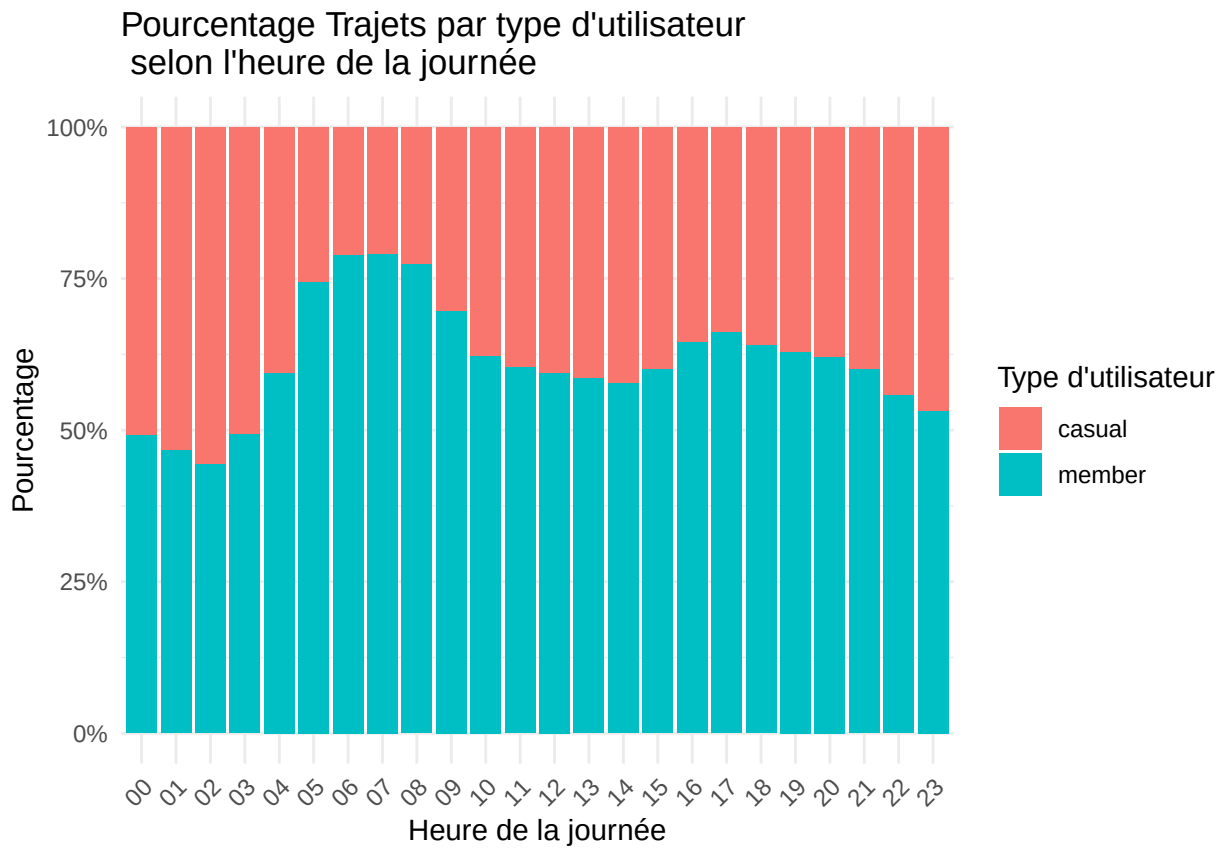
Pourcentage de trajets effectués par type client au fil des heures de la journée

Pour visualiser le pourcentage de clients par type d'utilisateur au fil des heures de la journée, nous allons créer un graphique qui affiche la répartition des utilisateurs membres et occasionnels pour chaque heure de la journée.

```
# Calculer le pourcentage d'utilisation par type d'utilisateur pour chaque heure de la journée
data_summary_hour <- merged_data_v1 %>%
  group_by(hour, member_casual) %>%
  summarise(count = n()) %>%
  mutate(percentage = (count / sum(count)) * 100)
```

```
## `summarise()` has grouped output by 'hour'. You can override using the
## `.groups` argument.
```

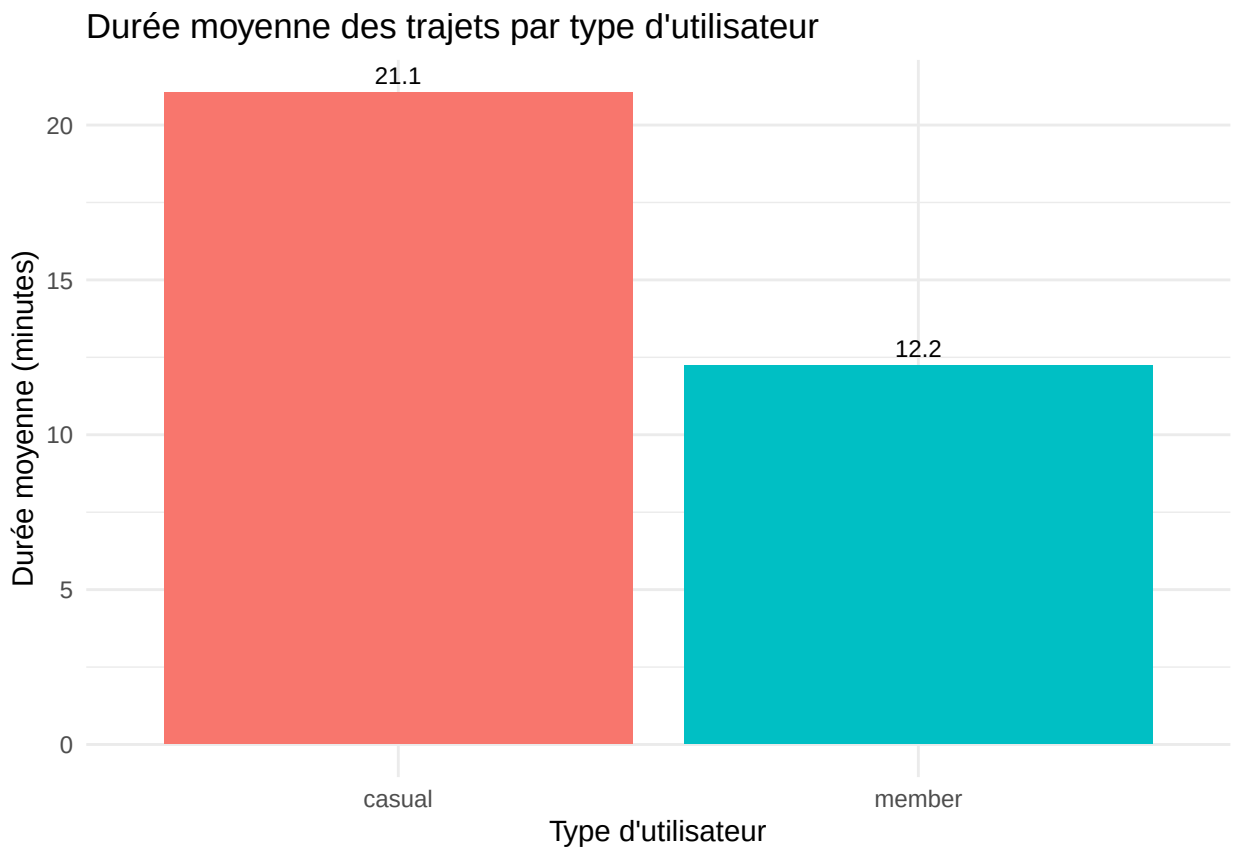
```
# Créer un graphique avec ggplot2
ggplot(data_summary_hour, aes(x = hour, y = percentage, fill = member_casual)) +
  geom_bar(stat = "identity", position = "fill") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(title = "Pourcentage Trajets par type d'utilisateur\n selon l'heure de la journée",
       x = "Heure de la journée", y = "Pourcentage", fill = "Type d'utilisateur") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



On remarque que les abonnés ont tendance à utiliser plus les vélos dans les plages horaires correspondant aux horaires d'ouverture et de fermeture des bureaux (de 5 heures à 9 heures le matin et de 16 heures à 18 heures l'après-midi).

Durée moyenne du parcours en fonction du type de cycliste et du nombre de chaque type de cycliste

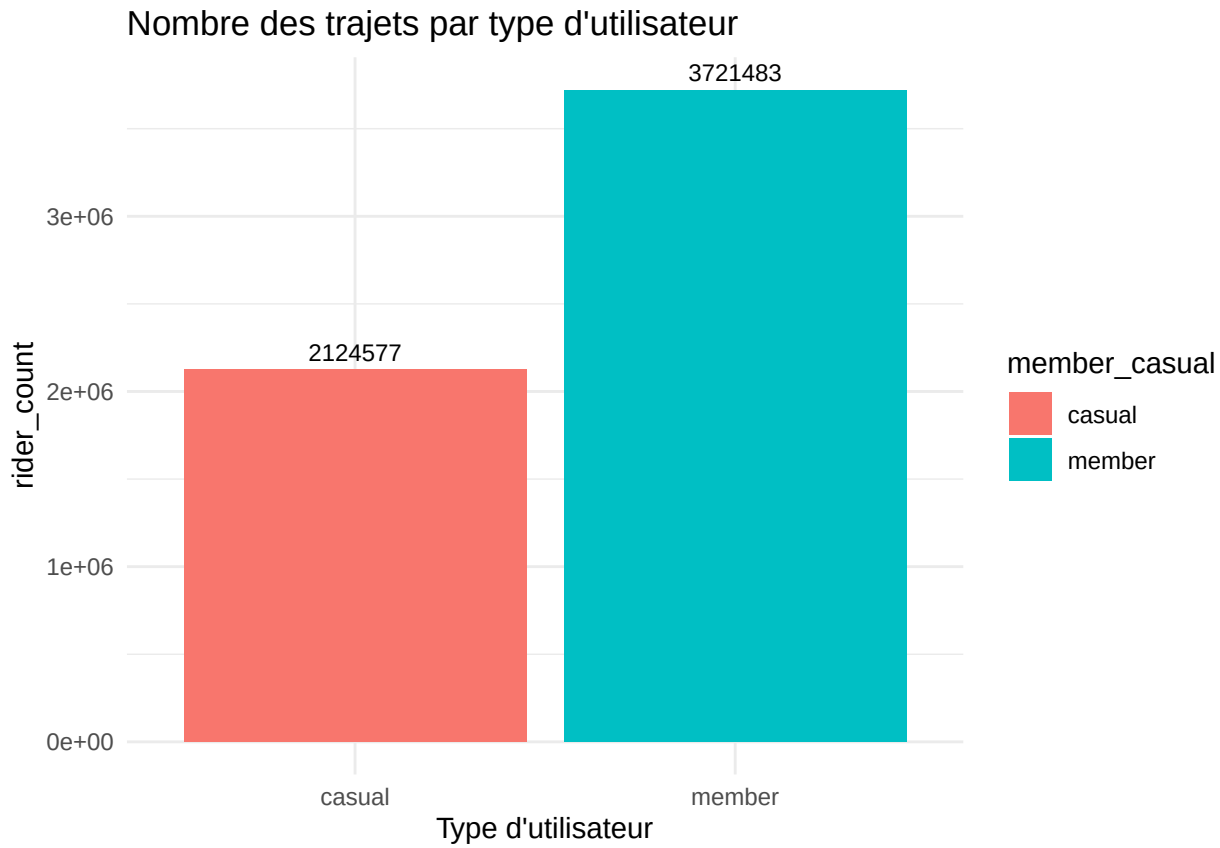
```r
merged_data_v1 %>%
  group_by(member_casual) %>%
  summarise(max_trip_duration = max(trip_duration_sec, na.rm = TRUE),
            min_trip_duration = min(trip_duration_sec, na.rm = TRUE),
            avg_trip_duration = mean(trip_duration_sec, na.rm = TRUE)) %>%
  mutate(avg_trip_duration_min = avg_trip_duration / 60) %>%  # Conversion en minutes
  ggplot(aes(x = member_casual, y = avg_trip_duration_min, fill = member_casual)) +
  geom_col(show.legend = FALSE) +
  scale_y_continuous(name = "Durée moyenne (minutes)", breaks = seq(0, 40, by = 5)) +  # Ajustement des
  labs(title = "Durée moyenne des trajets par type d'utilisateur",
       x = "Type d'utilisateur") +
  theme_minimal() +
  geom_text(aes(label = round(avg_trip_duration_min, 1)), vjust = -0.5, size = 3)  # Affichage de la du
```



Ainsi, le résultat montre que les cyclistes occasionnels ont tendance à louer des vélos pour des durées moyennes plus longues que les membres (21 min à 12 min), conformément au graphique 2. Les membres utilisent probablement des vélos pour se déplacer, tandis que les cyclistes occasionnels peuvent, entre autres, faire de l'exercice, visiter la ville ou assister à des événements spéciaux.

Ici, le nombre total de cyclistes en fonction du type de cycliste est représenté:

```r
merged_data_v1 %>%
  group_by(member_casual) %>%
  summarise(rider_count = n()) %>%
  ggplot(aes(x = member_casual, y = rider_count,fill=member_casual )) +
  geom_col()+
  labs(title = "Nombre des trajets par type d'utilisateur",
       x = "Type d'utilisateur") +
  theme_minimal() +
  geom_text(aes(label = rider_count), vjust = -0.5, size = 3)  # Affichage du nombre des trajets sur le
```



On a vu dans un graphique précédent, avec plus de précision que les membres representent les deux tiers de la clientèle, alors que le tiers restant est constitué d'utilisateurs occasionnels.

Analyse des effets de la saisonnalité

On crée une fonction « season » pour attribuer la saison aux mois:

Les mois de Décembre, Janvier et Février pour l'hiver

Les mois de Mars, Avril et MAi pour le printemps

Les mois de Juin, Juillet et Août pour l'été

Les mois de Septembre, Octobre et Novembre pour l'automne

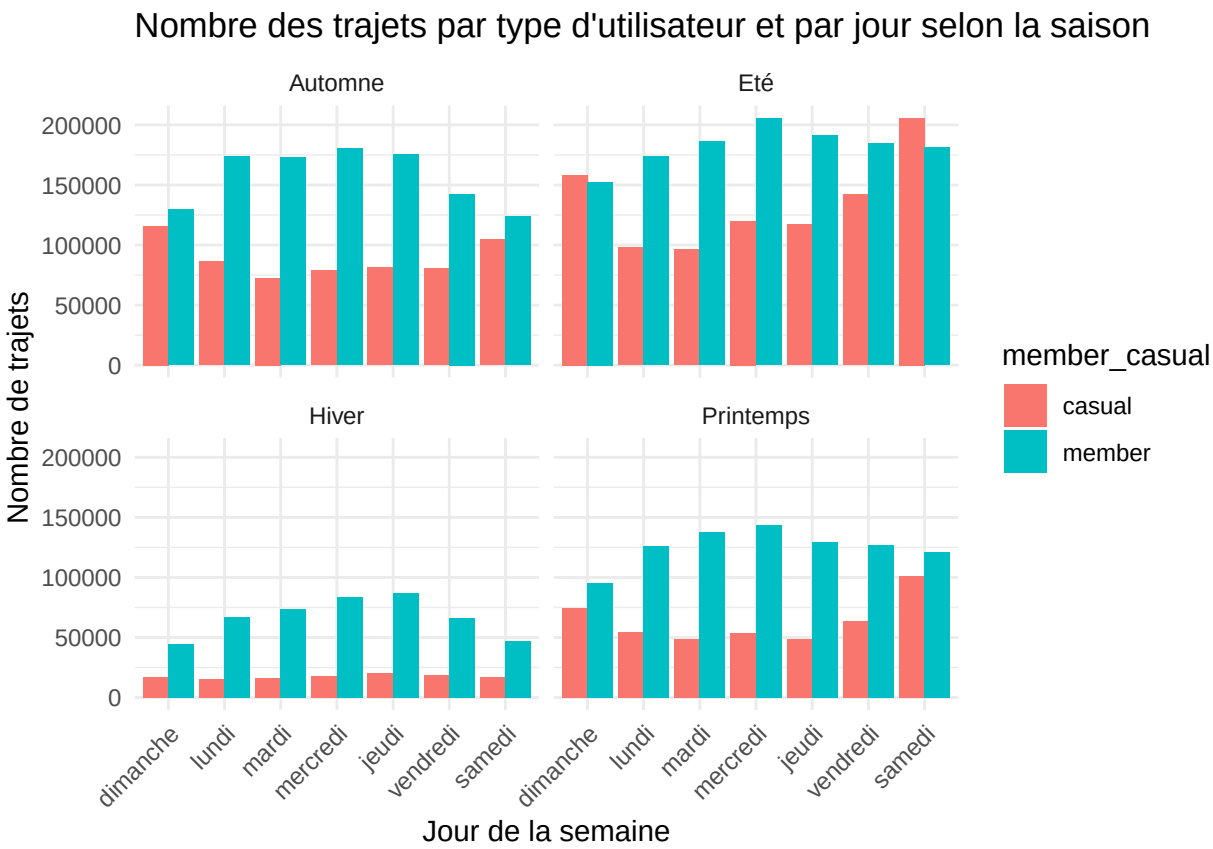
```r
#  S'assurer que les mois sont bien des entiers
merged_data_v1 <- merged_data_v1 %>%
  mutate(month = as.integer(month))

#  Définir la fonction de saison correctement
season <- function(month) {
  ifelse(month %in% c(12, 1, 2), "Hiver",
         ifelse(month %in% c(3, 4, 5), "Printemps",
                ifelse(month %in% c(6, 7, 8), "Eté", "Automne")))
}
```

```r
#  Créer la colonne `season` après vérification des mois
merged_data_v1 <- merged_data_v1 %>%
  filter(!is.na(month)) %>%  # Filtrer les valeurs `NA` pour les mois
  mutate(season = season(month))
```

Tout d'abord, considérons le nombre de trajets et la durée du trajet par jour de la semaine à chaque saison.

```r
merged_data_v1 %>%
  mutate(weekday = factor(weekday, levels = jours_semaine)) %>%
  group_by(season, weekday, member_casual) %>%
  summarise(
    rider_count = n(),
    avg_trip_duration_min = mean(trip_duration_sec, na.rm = TRUE) / 60  # Durée moyenne en minutes
  ) %>%
  ggplot() +
  geom_col(
    mapping = aes(x = weekday, y = rider_count, fill = member_casual),
    position = "dodge"
  ) +
  facet_wrap(~ season) +
  scale_y_continuous(
    name = "Nombre de trajets",
    breaks = seq(0, 800000, by = 50000)
  ) +
  labs(
    title = "Nombre des trajets par type d'utilisateur et par jour selon la saison",
    x = "Jour de la semaine",
    y = "Nombre de trajets"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

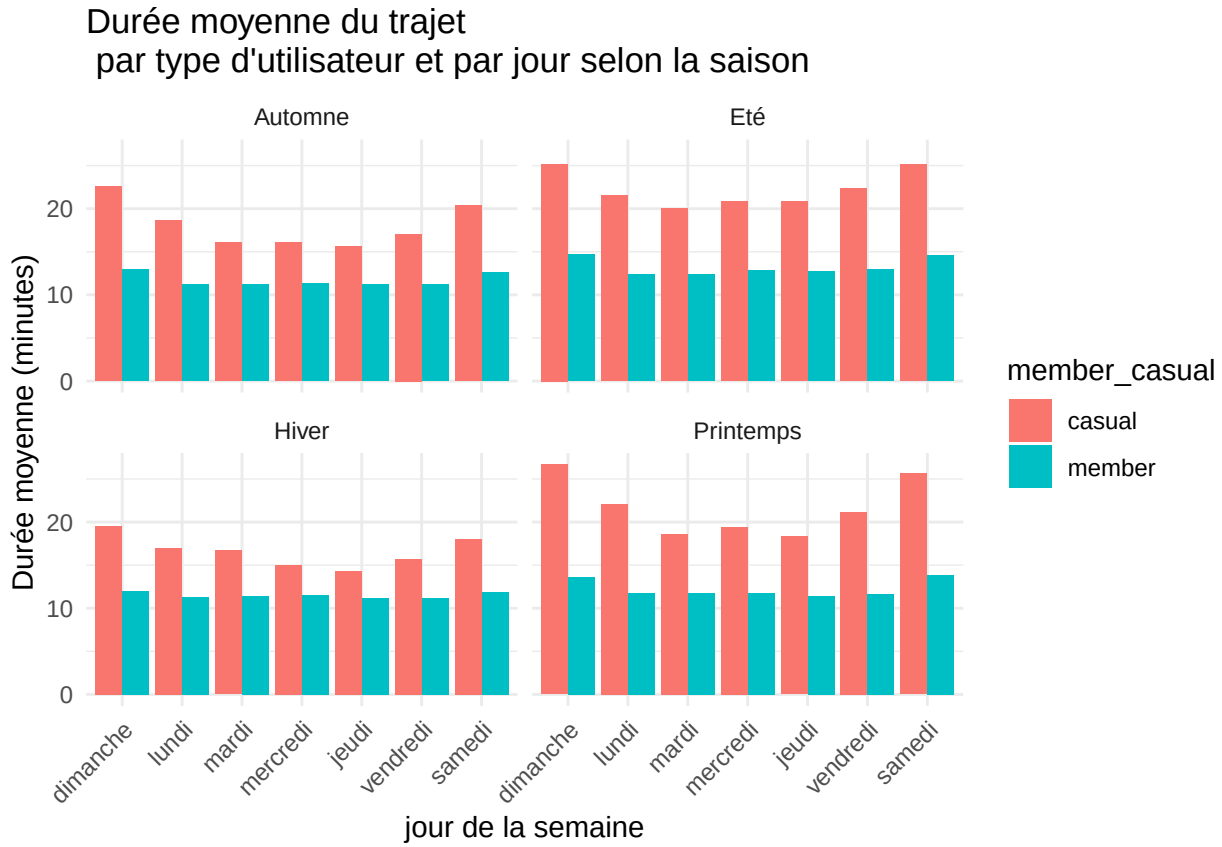
```
## `summarise()` has grouped output by 'season', 'weekday'. You can override using
## the `.groups` argument.
```



Ce graphique nous indique que le nombre de déplacements des membres est toujours plus élevé que celui des cyclistes occasionnels chaque jour ouvrable et en toute saison.Sauf pour les samedis et les dimanches de l'été où les occasionnels font plus de trajets que les abonnés. En Hiver, le nombre de trajets chutent de plus de la moitié, ce qui est tout à fait normal vu que la pluie et le froid découragent les balades en vélo.

```
merged_data_v1 %>%
  mutate(weekday = factor(weekday, levels = jours_semaine)) %>%
  group_by(season, weekday, member_casual) %>%
  summarise(rider_count = n(),
            avg_trip_duration_min = mean(trip_duration_sec)/60) %>%
  ggplot() +
  geom_col(mapping = aes(x = weekday, y = avg_trip_duration_min, fill = member_casual), position = "dod
  facet_wrap(~season) +
  scale_y_continuous(name = "Durée moyenne (minutes)", breaks = seq(0, 50, by = 10)) +
  labs(title = "Durée moyenne du trajet\n par type d'utilisateur et par jour selon la saison",
       x = "jour de la semaine", y = "Durée moyenne (minutes)") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## `summarise()` has grouped output by 'season', 'weekday'. You can override using
## the `.groups` argument.
```

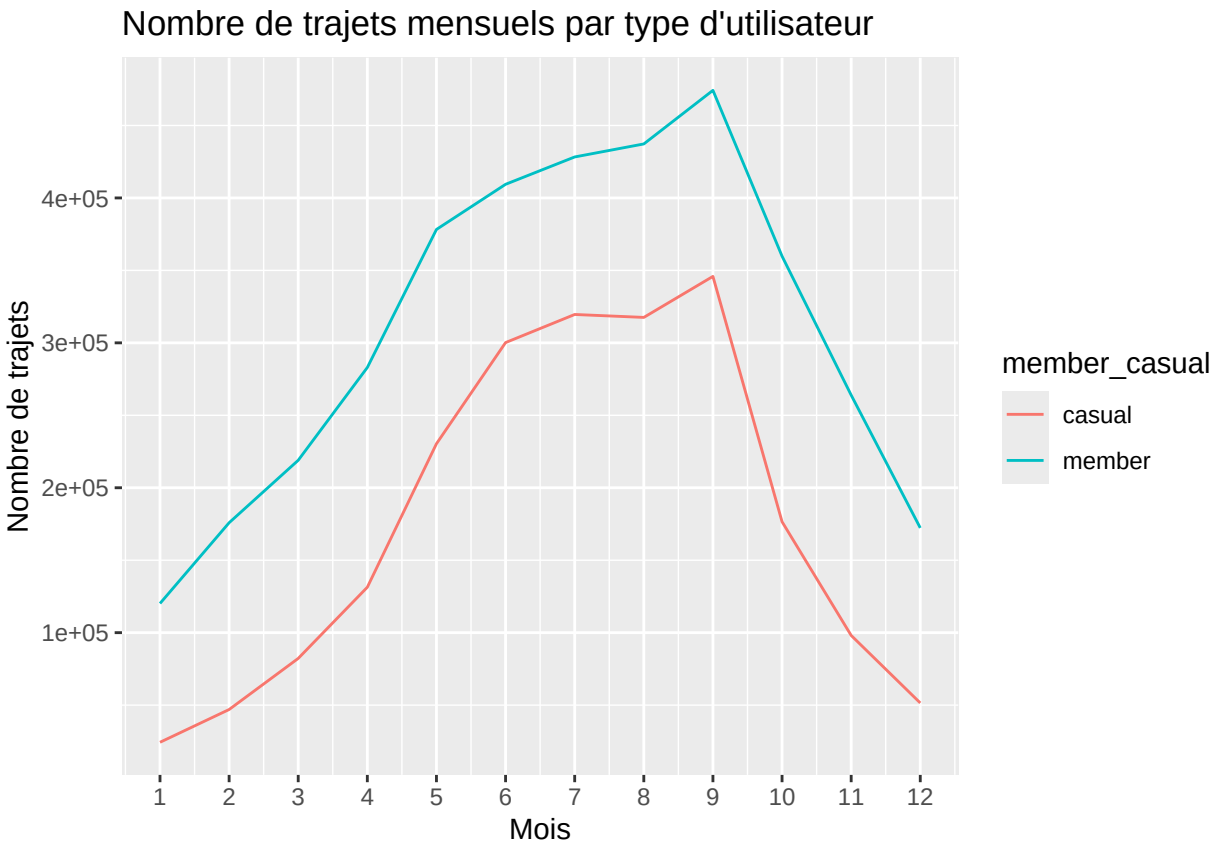


On remarque bien que le groupe de membres utilise le vélo pendant toute l'année pendant une durée moyenne d'environ 12 minutes. Les cyclistes occasionnels utilisent le vélo pendant environ 25 minutes tous les jours au printemps et en été. En hiver et en automne, la durée moyenne des trajets devient pratiquement inférieure à 20 minutes.

Enfin, générons un graphique linéaire pour l'évolution continue du nombre de sorties tout au long de l'année pour les deux types de cyclistes.

```
merged_data_v1 %>%
  mutate(month = as.numeric(month)) %>%  # Convertir les mois en nombres
  group_by(month, member_casual) %>%
  summarise(
    rider_count = n(),
    avg_trip_duration_min = mean(trip_duration_sec, na.rm = TRUE) / 60  # Moyenne des durées de trajet
  ) %>%
  ggplot(aes(x = month, y = rider_count, color = member_casual)) +
  geom_line() +
  scale_x_continuous(breaks = seq(1, 12, by = 1)) +  # Afficher les mois de 1 à 12
  labs(
    title = "Nombre de trajets mensuels par type d'utilisateur",
    x = "Mois",
    y = "Nombre de trajets"
  )
```

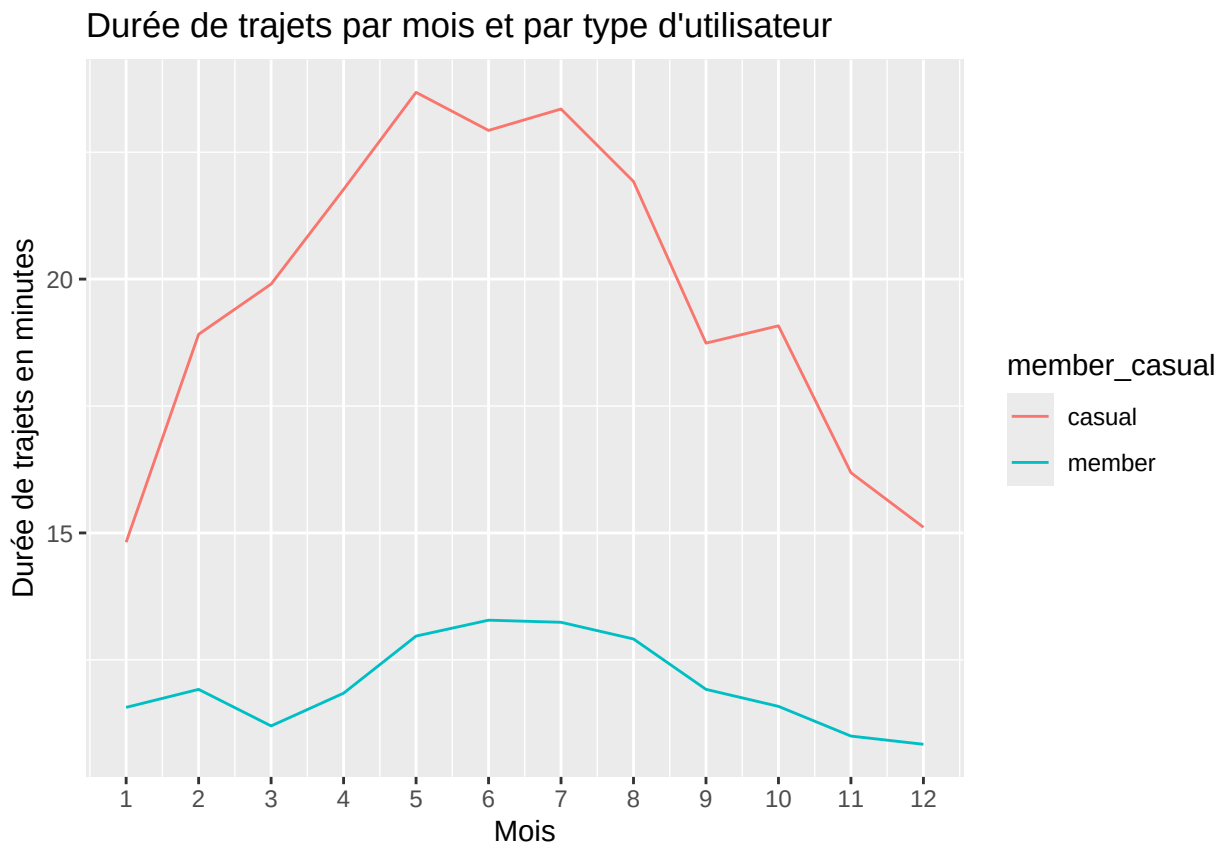
```
## `summarise()` has grouped output by 'month'. You can override using the
## `.groups` argument.
```



Le graphique indique que, pour tous les usagers qu'ils soient membres ou occasionnels, le nombre de trajets augmente considérablement pour les membres au mois de mai et pour les occasionnels au mpois de juin. Ce nombre atteint un pic vers juillet/août/Septembre (les mois d'été étant le point tournant) et a atteint son plus bas niveau en février avant de rebondir rapidement et continuellement.

```
merged_data_v1 %>%
  group_by(month, member_casual) %>%
  summarise(
    rider_count = n(),
    avg_trip_duration_min = mean(trip_duration_sec, na.rm = TRUE) / 60  # Moyenne des durées de trajet
  ) %>%
  ggplot(aes(x = month, y = avg_trip_duration_min, color = member_casual)) +
  geom_line() +
  scale_x_continuous(breaks = seq(1, 12, by = 1)) +  # Afficher les mois de 1 à 12
  labs(
    title = "Durée de trajets par mois et par type d'utilisateur",
    x = "Mois",
    y = "Durée de trajets en minutes"
  )
```

```
## `summarise()` has grouped output by 'month'. You can override using the
## `.groups` argument.
```



Encore une fois, on constate que la durée des trajets est presque régulière à partir du mois d'avril jusqu'au mois de septembre. Avec une chute notable pour les mois de l'automne et de l'hiver. Alors que la durée moyenne des trajets effectués par les occasionnels connait un pic à partir du mois de mai jusqu'au mois de juillet.

Astuce pour contourner les limites de nos données. Sans un identifiant unique pour chaque utilisateur, on ne peut pas distinguer les différentes utilisations par un même individu. Les données ne permettent donc pas de :

- Identifier le nombre d'utilisations multiples par une même personne : Par exemple, un abonné qui utilise le service cinq fois dans une journée ou sur une semaine sera compté cinq fois sans que l'on puisse savoir qu'il s'agit de la même personne.
- Faire la distinction entre les utilisateurs occasionnels récurrents et les visiteurs uniques : Un utilisateur occasionnel peut être un habitant de Chicago qui utilise le service sans abonnement ou un touriste qui ne fait qu'une seule utilisation. Les données ne permettent pas de déterminer la nature de l'utilisateur occasionnel (habitant régulier, touriste, etc.).

Pour répondre à ces limitations, on peut faire des regroupements sur une base temporelle (par exemple, par journée) et estimer des motifs de réutilisation. Par exemple, un pic dans les trajets répétés sur une même station au cours d'une journée peut suggérer des allers-retours fréquents par un même abonné, mais ce n'est qu'une estimation. On peut également croiser les données géographiques et temporelles pour avoir une meilleure estimation. Par exemple, les utilisateurs occasionnels qui partent et arrivent souvent des points stratégiques touristiques pourraient être identifiés comme des touristes potentiels, mais encore une fois, c'est une inférence.

```r
# définir un miroir
options(repos = c(CRAN = "https://cloud.r-project.org/"))

# Installer les bibliothèques nécessaires
install.packages("leaflet")
```

Visualiser les stations les plus fréquentées de Chicago

```
## Installation du package dans 'C:/Users/musta/AppData/Local/R/win-library/4.4'
## (car 'lib' n'est pas spécifié)

## le package 'leaflet' a été décompressé et les sommes MD5 ont été vérifiées avec succés
##
## Les packages binaires téléchargés sont dans
##  C:\Users\musta\AppData\Local\Temp\RtmpOAO6MO\downloaded_packages
```

```r
# Charger les bibliothèques
library(leaflet)
library(dplyr)  # Pour la manipulation de données
```

```r
# Charger les bibliothèques nécessaires
library(dplyr)
library(leaflet)

# 1. Identifier les trois stations les plus fréquentées pour chaque jour (lundi et samedi)

# Convertir 'weekday' en facteur avec les jours dans le bon ordre
merged_data_v1 <- merged_data_v1 %>%
  mutate(weekday = factor(weekday,
                          levels = c("dimanche", "lundi", "mardi", "mercredi", "jeudi", "vendredi", "sa

# Calculer le nombre de trajets par station d'arrivée pour le lundi
top_monday_stations <- merged_data_v1 %>%
  filter(weekday == "lundi", !is.na(end_station_name), !is.na(end_lat), !is.na(end_lng)) %>%
  group_by(end_station_name, end_lat, end_lng) %>%
  summarise(traffic_count = n(), .groups = "drop") %>%
  arrange(desc(traffic_count)) %>%
  slice_max(order_by = traffic_count, n = 5)


# Calculer le nombre de trajets par station d'arrivée pour le samedi
top_saturday_stations <- merged_data_v1 %>%
  filter(weekday == "samedi", !is.na(end_station_name), !is.na(end_lat), !is.na(end_lng)) %>%
  group_by(end_station_name, end_lat, end_lng) %>%
  summarise(traffic_count = n(), .groups = "drop") %>%
  arrange(desc(traffic_count)) %>%
  slice_max(order_by = traffic_count, n = 5)

nrow(top_saturday_stations)
```

```
## [1] 5
```

```r
leaflet() %>%
  # Ajouter les stations du lundi en rouge
  addTiles() %>%
  addCircleMarkers(data = top_monday_stations,
                   lat = ~end_lat,
                   lng = ~end_lng,
                   color = "red",
                   fillOpacity = 0.8,
                   radius = 8,
                   label = ~paste(end_station_name, "<br>", "Trafic :", traffic_count),
                   popup = ~paste("<strong>Station :</strong> ", end_station_name,
                                  "<br><strong>Trafic le lundi :</strong> ", traffic_count)) %>%
  # Ajouter les stations du samedi en bleu
  addCircleMarkers(data = top_saturday_stations,
                   lat = ~end_lat,
                   lng = ~end_lng,
                   color = "blue",
                   fillOpacity = 0.8,
                   radius = 8,
                   label = ~paste(end_station_name, "<br>", "Trafic :", traffic_count),
                   popup = ~paste("<strong>Station :</strong> ", end_station_name,
                                  "<br><strong>Trafic le samedi :</strong> ", traffic_count)) %>%
  # Ajouter une légende
  addLegend(position = "bottomright",
            colors = c("red", "blue"),
            labels = c("Top 5 lundi", "Top 5 samedi"),
            title = "Stations avec le plus de trafic")
```

Pour une analyse sommaire des comportements des utilisateurs des vélos selon les jours de la semaine (lundi et samledi), on peut déduire ce qui suit:

- Usage récréatif et touristique le week-end :

Le samedi, les stations les plus fréquentées incluent des lieux emblématiques de loisirs et de détente comme le Chicago Yacht Club, Monroe Station, le Janne Adams Memorial Park, et Lincoln Park. Cela peut indiquer une tendance des utilisateurs à pratiquer des activités de plein air ou de loisir le week-end, profitant des espaces verts, des attractions touristiques et des vues sur le lac Michigan.

- Déplacements quotidiens et professionnels en semaine :

Le lundi, les stations les plus fréquentées se trouvent autour de West Kinzie Street et North Clinton Street. Ces stations se situent probablement dans des zones avec des bureaux, entreprises ou infrastructures de transport (comme les gares). Cela suggère que les vélos Divvy sont utilisés comme moyen de transport quotidien par les navetteurs pour se rendre à leur travail.

Phase de Partage

Conclusions et Résumé des observations Les membres et les cyclistes occasionnels diffèrent dans la durée et la fréquence d'utilisation des vélos, ainsi que dans les jours de la semaine où chaque groupe atteint son pic:

- Les trajets occasionnels atteignent leur pic le week-end . Il est fort probable qu'il s'agisse de touristes visitant la ville ou de résidents ordinaires de Chicago qui font du vélo pendant leur temps libre pendant le week-end. Le temps de trajet moyen plus long pour les cyclistes occasionnels , qui culmine également le week-end, en est la preuve.

- La durée du trajet des membres est relativement plus courte que celle des cyclistes occasionnels. Cela pourrait être clarifié comme suit: la plupart des membres utilisent le vélo pour se déplacer les jours ouvrables. Cette clarification expliquerait également les courtes durées de trajet des membres. Ils roulent du point A au point B, à savoir à peu près toujours les mêmes longueurs de trajet et la même distance. Encore faut-il préciser qu'en l'absence d'informations personnelles des utilisateurs, on n'est pas en mesure de conformer ou d'infirmer cette hypothèse.

- Le nombre de trajets commence à augmenter à partir de février (du printemps à l'été) et commence à diminuer en octobre (de l'automne à l'hiver). Cette corrélation est due aux changements saisonniers. Lorsque le temps commence à se réchauffer et à devenir plus agréable en février (début du printemps), de plus en plus de personnes commencent à faire du vélo, et inversement lorsque le temps devient moins chaud et froid vers septembre (début de l'automne).

- Plus de 63 % des trajets sont effectués par des membres annuels , ce qui suggère que l'entreprise a déjà atteint un certain niveau de fidélité parmi ses utilisateurs de vélo. Cela indique un message positif, à savoir que l'entreprise va pouvoir convaincre de nombreux cyclistes occasionnels de devenir membres et de garder les nouveaux membres satisfaits.

Recommandations Offrir des réductions pour les trajets plus longs pour les membres abonnés. Les trajets plus longs peuvent bénéficier d'un programme de récompenses lorsqu'ils deviennent membres.

- La campagne marketing serait plus optimisée si elle est lancée entre avril et septembre, car le nombre de trajets effectués par les cyclistes occasionnels atteint son maximum à cette période de l'année.

- Comme l'utilisation des cyclistes occasionnels atteint son point culminant le week-end, la campagne marketing peut inclure un abonnement annuel "Spécial Week-End" à un prix raisonnable. Cela pourrait inciter les cyclistes occasionnels à devenir membres.

- La campagne pourrait inclure un plan tarifaire pour les non abonnés basé sur la durée du trajet et la station d'arrivée (peut-être seulement le week-end) : Si vous faites plus de vélo pour arriver à telles stations, vous allez payez plus ! Cela inciterait davantage les occasionnels à adhérer aux formules d'abonnement proposées.

- Faire pousser l'analyse plus loin en incluant les informations personnelles des usagers ou au moins en incluant un identifiant pour les vélos pour un traçage plus précis des trajets. Celà permettrait de distinguer les trajets d'habitude des trajets ponctuels.