



2024

Rapport de projet

Mener une étude statistique dans un domaine
d'application

Donneur d'ordre :

Aurelie BARCQ

Laurent BUREAU

Axel DAGNAUD

Noah DE GUNST

Victor CHASSÉ

SOMMAIRE

Introduction	3
Audit des données	5
• Analyse descriptive de la base de données	
• Identification des ruptures d'activité et traitement proposé	
• Sélection du segment d'activité pour la modélisation	
Data Prep	9
• Processus d'agrégation des données	
• Extrait de la base de données de travail	
Ajout de variables	11
• Variables complémentaires ajoutées	
• Intérêt et impact des nouvelles variables sur l'analyse	
• Préparation pour la modélisation	
Modélisation	14
• Test de significativité des variables	
• Présentation des deux méthodes de modélisation	
• Comparaison des modèles et choix du modèle recommandé	
• Résultats et interprétation	
Power BI	23
Conclusion	25
Annexe	26

INTRODUCTION

INTRODUCTION

Dans le cadre de la “**SAÉ - Mener une étude statistique dans un domaine d'application**”, nous avons été mandatés par **Inter Mutuelles Assistance** pour réaliser une étude statistique visant à prévoir l’activité quotidienne sur leurs plateaux d’assistance. Cette prévision est cruciale pour assurer un **dimensionnement optimal du personnel**, afin de garantir une **réponse rapide et efficace** aux besoins des automobilistes confrontés à des situations imprévues telles que les **pannes** ou les **accidents**.

Le projet s’appuie sur une **base de données (BDD)** contenant les **volumes quotidiens de dossiers depuis 2013** pour deux segments d’activité propres à l’entreprise. Notre mission s’articule autour de plusieurs **étapes clés** : l’audit des données, leur **préparation**, l’ajout de variables pertinentes, la **modélisation statistique** et la **création d’un outil de suivi sur Power BI**.

Chaque étape a pour objectif de répondre à des questions spécifiques :

- Comment **interpréter** et **exploiter** les données existantes pour en extraire des informations pertinentes ?
- Quels facteurs additionnels pourraient **améliorer la qualité des prévisions** ?
- Quelle **modélisation** est **la plus adaptée** aux besoins de l’entreprise ?
- Comment **traduire** ces analyses en **visualisations claires et opérationnelles** ?

Le présent rapport détaille notre démarche, depuis l’**analyse préliminaire des données** jusqu’à la **proposition d’un modèle prédictif robuste** et d’un **outil de suivi interactif**. Nous mettons un point d’honneur à respecter les **exigences formulées**, tout en apportant une **valeur ajoutée** grâce à une **approche personnalisée et innovante**.

Ainsi, ce rapport se structure autour des sections suivantes :

- **Audit des données** : analyse descriptive, identification des ruptures, et segmentation.
- **Data Prep** : préparation et agrégation des données pour une exploitation optimale.
- Ajout de variables : enrichissement de la BDD avec des données contextuelles et calculs pertinents.
- **Modélisation** : comparaison et sélection de modèles prédictifs pour l’activité quotidienne.
- **Conclusion** : synthèse des résultats et recommandations.
- **Annexes** : extraits de la BDD, copies d’écran, et codes R commentés.

Ce travail vise à fournir des **outils décisionnels fiables et adaptés** aux besoins stratégiques d’**Inter Mutuelles Assistance**, dans une perspective d’amélioration continue de leurs services.

Audit des données

Audit des données

Ce document analyse les données d'activité d'Inter Mutuelles Assistance pour mieux comprendre leur structure et identifier les tendances clés, **cette analyse a été effectué avec R des détails sur le code se trouve dans les annexes.**

Après une exploration des données et des corrections sur les anomalies, un segment principal est sélectionné pour la modélisation, en tenant compte de sa pertinence pour la prévision des volumes d'appels.

Analyse descriptive de la base de données

L'analyse descriptive révèle les caractéristiques principales des **3 995 263 enregistrements** de la base de données. Les colonnes principales incluent des informations sur le segment d'activité (**lb_segment**), la date d'ouverture des dossiers (**dateouv**), les causes détaillées (**lb_detail_cause**) et le nombre de dossiers ouverts (**RL_DOSS**).

Statistiques clés :

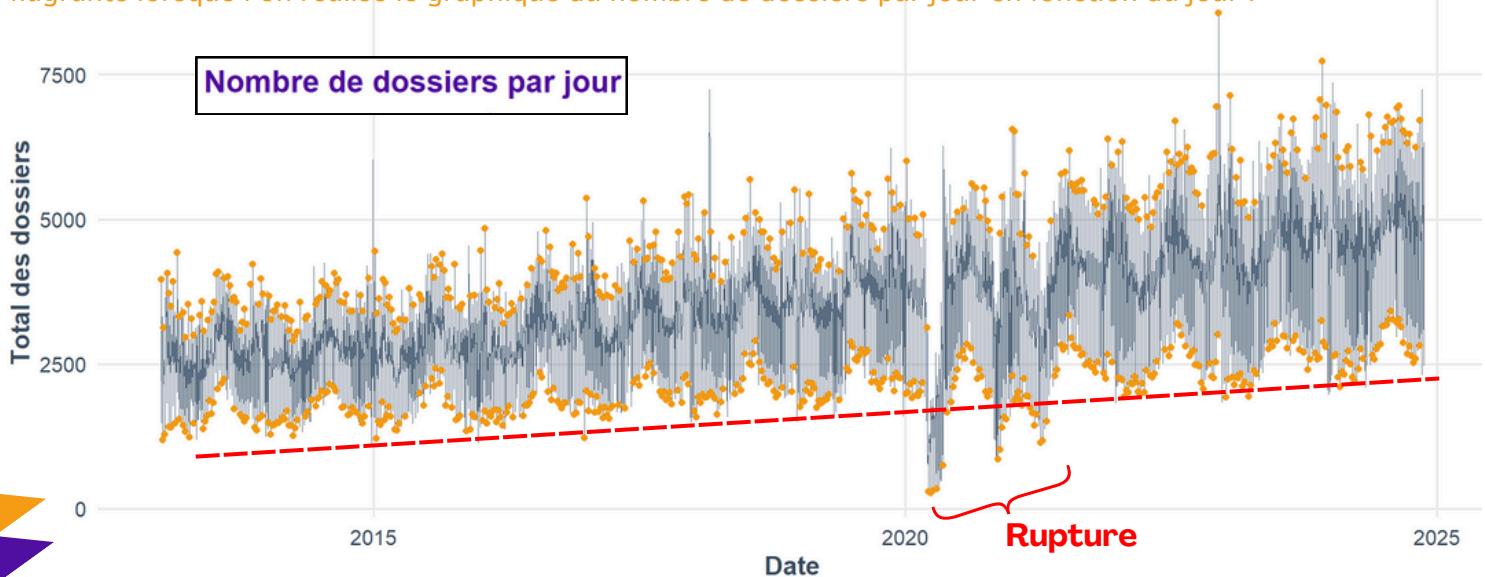
- Deux segments principaux identifiés :
 - Assistance à moins de 50 km (**11 413 998 dossiers**)
 - Assistance à plus de 50 km (**3 499 401 dossiers**)

Distribution temporelle : Une **croissance globale** est observée **entre 2013 et 2023**, avec un pic en 2023 à 1 593 685 dossiers. L'année 2024 semble continuer sur la même lancée. Cependant, on observe une **rupture significative en 2020** sans doute du au COVID, cette rupture sera traité dans la prochaine partie.

Valeurs manquantes : La colonne **NBKMSORI** présente **259 959 valeurs manquantes**, soit **6,5 %** des données. Aucune autre colonne n'est affectée par des valeurs absentes.

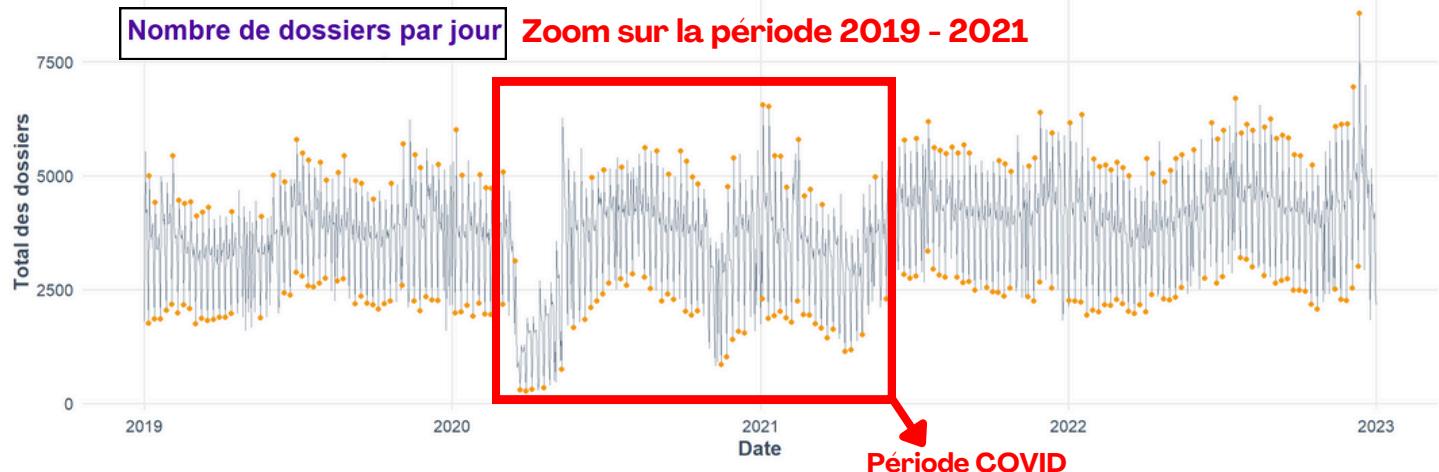
Identification des ruptures d'activité et traitement proposé

Pendant l'analyse, une rupture liée au mouvement saisonnier habituel a été observée, cette rupture est flagrante lorsque l'on réalise le graphique du nombre de dossiers par jour en fonction du jour :



Audit des données

Afin de pouvoir traiter cette rupture, il est important de zoomer sur la zone de rupture pour définir la période à remplacer :



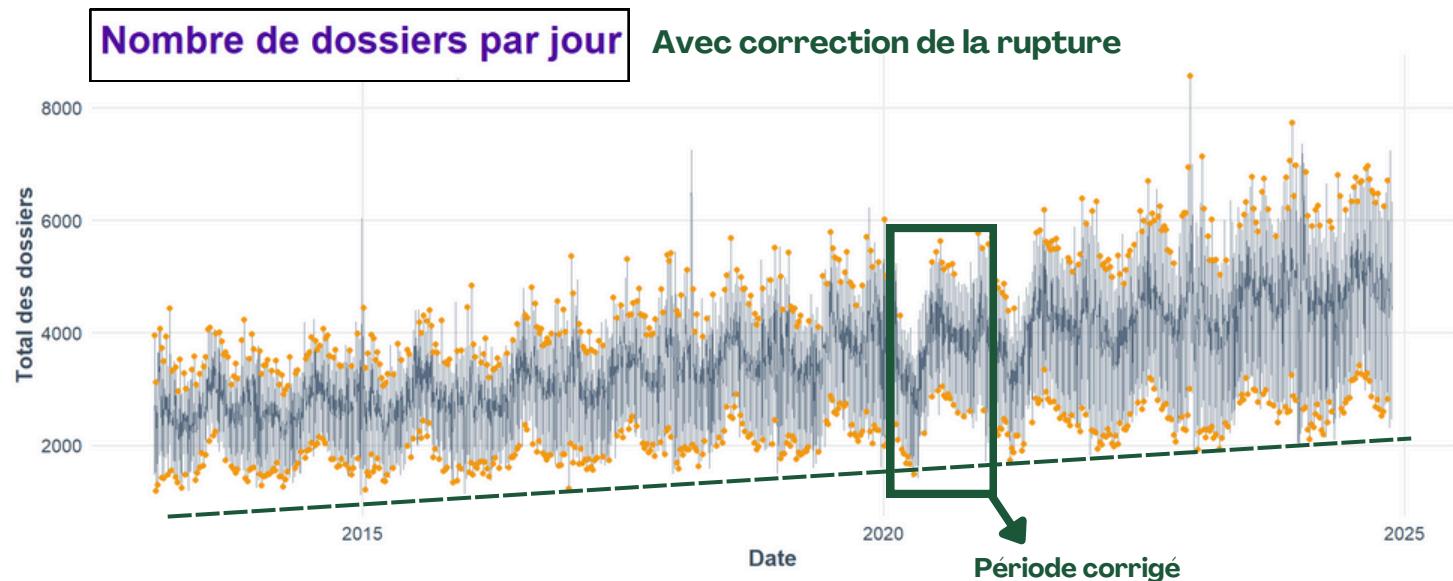
Cette rupture est probablement causée par la **pandémie de COVID-19**. Pour pallier cette irrégularité et lisser les données, voici une **proposition de solution** :

Nous pourrions **remplacer la période concernée** par la moyenne du nombre de dossiers pour **le même jour des années n-1 et n+1**. Cependant, étant donné la **croissance observée** du nombre de dossiers au fil des années, il serait pertinent de **pondérer ces valeurs**. Ainsi, il serait judicieux d'accorder un poids plus important à **la valeur de l'année n+1**.

Après analyse graphique, la rupture semble s'étaler de **février 2020 à juin 2021**. Par itérations, nous avons défini les **coefficients suivants** pour réaliser une **moyenne pondérée** :

- **20 %** des données issues de l'année précédente (**n-1**),
- **80 %** des données issues de l'année suivante (**n+1**).

Enfin, un **nouveau graphique** est réalisé avec ces **corrections** :



Audit des données

Sélection du segment d'activité pour la modélisation

Afin de sélectionner un segment d'activité pour la modélisation, nous nous sommes poser les questions suivantes :

- Quel est le segment avec le plus d'observations et donc le plus de ligne dans la base de données ?
- Quel segment a la plus grande volumétrie et est donc le plus important à bien prédire afin de bien gérer la présence d'assistant sur le plateau ?

On obtient alors les résultats suivant avec la base de données de 2018 jusqu'à aujourd'hui :

#	lb_segment	nb_observations	nb_dossiers
1	Assistance à moins de 50km	2016245	11413998
2	Assistance à plus de 50km	1979018	3499401

Avec plus de 11 millions de dossiers, l'assistance dans un rayon de moins de 50 kilomètres constitue le segment au plus grand volume. Il est donc essentiel de comprendre les raisons de cette ampleur et d'explorer les moyens de mieux anticiper ces demandes. C'est donc ce segment que nous avons pris pour la modelisation.

Data Prep

Data Prep

Processus d'agrégation des données

Pour préparer la base de données de travail, nous avons suivi les étapes suivantes :

1. Chargement des données :

- Les données principales ont été importées à partir du fichier bdd_projet.csv et celles des données calendaires à partir de TP3 - df_calendaires_France.csv. Ces fichiers ont été encodés en format latin1 pour assurer une lecture correcte.

2. Nettoyage des colonnes :

- La colonne site, ne contenant que la France, a été supprimée.

3. Agrégation des données :

- Les données ont été regroupées par les colonnes segment et dateouv afin de calculer le nombre total de dossiers pour chaque couple (segment, date). La somme de la variable RL_DOSS a été utilisée à cet effet, en ignorant les valeurs manquantes. Cela fait perdre la granularité des données mais permet des meilleures prédictions

4. Filtrage spécifique :

- L'analyse étant restreinte au segment DD01 et donc à moins de 30 kilomètres, les autres segments ont été supprimés.

5. Simplification finale :

- La colonne segment a été supprimée, puisque l'ensemble des données de la base de travail correspond exclusivement au segment DD01.

```
data <- read.csv2("bdd_projet.csv", encoding = "latin1")  
  
# Supprime la colonne site car pas nécessaire  
data$site <- NULL
```

Voici le code utilisé :

```
# Calcule nb dossiers pour chaque couple (segment,date)  
data <- data %>% group_by(segment, dateouv) %>%  
  summarise(nb_dossier = sum(RL_DOSS, na.rm = TRUE))  
  
# Choix du segment DD01 car c'est celui avec le plus de données  
data <- data[data$segment == 'DD01', ]  
  
# Supprime la colonne segment car pas nécessaire  
data$segment <- NULL
```

Extrait de la base de données de travail

Voici un extrait de la base de données de travail :

```
> head(data,15)  
# A tibble: 15 x 3  
# Groups:   segment [1]  
  segment dateouv    nb_dossier  
  <chr>   <chr>        <int>  
1 DD01    2013-01-01     815  
2 DD01    2013-01-02    2995  
3 DD01    2013-01-03    2094  
4 DD01    2013-01-04    1901  
5 DD01    2013-01-05    1290  
6 DD01    2013-01-06    2013-01-06    701  
7 DD01    2013-01-07    2013-01-07    2655  
8 DD01    2013-01-08    2013-01-08    1936  
9 DD01    2013-01-09    2013-01-09    2037  
10 DD01   2013-01-10    2013-01-10    2001  
11 DD01   2013-01-11    2013-01-11    2044  
12 DD01   2013-01-12    2013-01-12    1519  
13 DD01   2013-01-13    2013-01-13    834  
14 DD01   2013-01-14    2013-01-14    3085  
15 DD01   2013-01-15    2013-01-15    3143
```

Ajout de variables

Ajout de variables

Variables complémentaires ajoutées

Pour enrichir la base de données initiale, nous avons intégré des **données calendaires**. Ces données offrent des **informations contextuelles** qui peuvent avoir un impact significatif sur l'activité observée. Voici les **principales variables** intégrées et leur intérêt :

- **FL_FERIE** : Indicateur des jours fériés. Cela permet de mesurer l'impact des jours fériés sur le volume d'activité.
- **FL_PONT** : Indicateur des ponts (jours ouvrés entre un jour férié et un week-end). Ces jours spécifiques peuvent influencer l'activité économique.
- **FL_VACA, FL_VACB, FL_VACC** : Indicateurs des périodes de vacances scolaires selon les zones A, B et C. Ces variables permettent de modéliser l'impact des vacances sur l'activité, notamment dans les secteurs liés à l'éducation ou aux familles.
- **FL_WEEKEND** : Indicateur des week-ends. Cela capture les variations liées à l'activité réduite en fin de semaine.
- **LB_SAISON** : Libellé de la saison (hiver, printemps, été, automne). Les comportements d'activité peuvent varier selon les saisons.

Approche et méthodologie

1. Les données calendaires ont été filtrées pour ne conserver que les dates postérieures au 1er janvier 2013, afin de correspondre à la période d'étude de la base initiale.
2. Une jointure a été réalisée entre la base initiale et les données calendaires

Limites et perspectives

Bien que seules les données calendaires aient été ajoutées dans cette phase, d'autres types de données pourraient enrichir davantage l'analyse, notamment :

- Données météo : Les conditions climatiques peuvent influencer les comportements, notamment dans des secteurs spécifiques.
- Données de trafic : Les flux de transport pourraient donner des indications précieuses sur les déplacements et les activités associées.

Nous avons décidé de ne pas utiliser ces données, car nous ne pouvons pas garantir leur fiabilité ni leur conformité aux règles d'utilisation. Par ailleurs, intégrer des données incertaines pourrait introduire des biais dans le modèle, ce qui risquerait de provoquer des erreurs de prédiction importantes, un scénario que nous souhaitons absolument éviter.

Ajout de variables

Préparation pour la modélisation

Pour rendre les données exploitables dans le cadre de modèles prédictifs, certaines transformations ont été effectuées :

- Suppression des colonnes non nécessaires :

- La variable LB_JOUR a été supprimée car elle était déjà encodée sous forme one-hot.
- Les colonnes LB_JOURAG, LB_JOURAG_Jm1, et LB_JOURAG_Jp1, bien que fournissant des informations temporelles contextuelles, ont également été retirées pour éviter une redondance ou une complexité inutile.

- Encodage one-hot :

- Les variables catégoriques restantes telles que LB_SAISON, LB_JOURAG, et LB_FERIE ont été transformées en plusieurs colonnes binaires via un encodage one-hot.
- L'encodage a été réalisé en utilisant la fonction dummyVars de R, suivie de leur intégration dans la base de données, tout en supprimant les colonnes d'origine.

- Normalisation des données :

- La colonne année (NU_AN) a été centrée en enlevant l'année 2013
- La variable cible nb_dossier a été centrée et réduite

- Le code utilisé pour ces étapes est le suivant :

```
data_modele <- na.omit(data_modele) # Suppression des lignes contenant des valeurs manquantes

# Encodage des variables qualitatives sélectionnées
modele_encodeage <- dummyVars(~ LB_SAISON + LB_JOURAG + LB_FERIE, data = data_modele)

# Transformation des données avec l'encodage
encodage_data <- as.data.frame(predict(modele_encodeage, newdata = data_modele))

# Ajout des colonnes encodées aux données d'origine
data_modele <- cbind(data_modele, encodage_data)

# Suppression des colonnes originales encodées
data_modele <- data_modele %>% select(-LB_SAISON, -LB_JOURAG, -LB_FERIE,
                                         -LB_JOURAG_Jm1, -LB_JOURAG_Jp1, -DT_JOUR)

# Ajustement de l'année
data_modele$NU_AN <- data_modele$NU_AN - 2013

# Normalisation de la cible
moyenne_y <- mean(data_modele$nb_dossier, na.rm = TRUE)
ec_type_y <- sd(data_modele$nb_dossier, na.rm = TRUE)
data_modele$nb_dossier_normalise <- (data_modele$nb_dossier - moyenne_y) / ec_type_y
```

Les Modèles

Les modèles

Analyse de la significativité des variables

Avant d'entamer la phase de modélisation, il est essentiel d'identifier les **variables** ayant un lien **statistiquement significatif** avec l'activité quotidienne, mesurée par le nombre de dossiers (nb_dossier). Cette étape permet de réduire la complexité du modèle, d'améliorer ses performances et d'éviter l'introduction de bruit dans les prédictions.

Approche méthodologique

Identification des variables à tester :

Les variables directement liées à la temporalité, comme DT_JOUR ainsi que la variable cible nb_dossier, ont été exclues de l'analyse.

Les variables restantes, **principalement binaires** ou catégoriques, ont été testées pour déterminer leur significativité.

Méthode statistique utilisée :

Un test t de Student a été effectué entre la variable cible nb_dossier et chaque variable à tester.

Ce test mesure si la moyenne de nb_dossier diffère significativement en fonction des catégories ou niveaux de chaque variable explicative.

Critère de significativité :

Une **p-valeur inférieure à 0,01** (seuil de significativité de 1 %) a été retenue pour sélectionner les variables pertinentes.

Automatisation du processus :

Un script R a été développé pour tester automatiquement chaque variable et extraire celles qui satisfont le critère de significativité. **Voici le code utilisé :**

```
# les colonnes à tester statistiquement avec la colonne nb_dossier
columns_FL <- colnames(data)
columns_FL <- columns_FL[!columns_FL %in% c("nb_dossier", "NU_AN", "DT_JOUR", "NU_SEMN_SERIE")]
#recupere les variables statistiquement signifcatif avec le nombre de dossiers
perform_statistical_test <- function(df, numeric_col, columns_to_test) {
  significant_columns <- c()

  for (col in columns_to_test) {
    # Effectuer le test t
    test_result <- tryCatch({
      t.test(df[[numeric_col]], df[[col]], na.rm = TRUE)
    }, error = function(e) NULL)

    # Extraire la p-valeur si le résultat du test est valide
    if (!is.null(test_result)) {
      p_value <- test_result$p.value
    } else {
      p_value <- NA
    }

    # Vérifier si la p-valeur est inférieure au seuil de signification
    if (!is.na(p_value) && p_value < 0.01) {
      significant_columns <- c(significant_columns, col)
    }
  }

  return(significant_columns)
}

# les colonnes significatifs
signif <- perform_statistical_test(data, 'nb_dossier', columns_FL)

data_signif <- data[, signif] # garde uniquement les colonnes significatifs à prédire
data_signif$nb_dossier <- data$nb_dossier
```

Les modèles

```
data <- read.csv2("bdd_projet.csv", encoding = "latin1")
data_calendaires <- read.csv2("TP3 - df_calendaires_France.csv", encoding ="latin1")
data$site <- NULL
data <- data %>% group_by(segment, dateouv) %>%
  summarise(nb_dossier = sum(RL_DOSS, na.rm = TRUE))
data <- data[data$segment == 'DD01', ]
data$segment <- NULL
data_calendaires <- data_calendaires[data_calendaires$DT_JOUR >= as.Date("2013-01-01"), ]
data_joint <- merge(x = data_calendaires, y=data, by.x = 'DT_JOUR', by.y = 'dateouv', all.x=TRUE)
```

Avec ce code, nous lisons les fichiers, bdd_projet (données des dossiers quotidiennes) ainsi que des données calendaires contenant les jours de la semaine et fériés notamment, permettant d'affiner l'analyse. Ensuite nous regroupons par segment et date puis nous sélectionnons seulement le segment DD01 (dossiers à moins de 50km). Nous finissons par joindre les deux fichiers

Résultats préliminaires

Parmi les variables analysées, **83** ont été identifiées comme **statistiquement significatives**. Ces variables présentent un lien potentiel avec l'activité quotidienne et seront utilisées dans les phases ultérieures pour entraîner et valider les modèles de prévision.

Préparation des données pour la modélisation

Les **observations** sans valeur pour la variable cible nb_dossier (**postérieures au 28/11/2024**) ont été isolées dans un sous-ensemble distinct (data_a_predire) pour être **prédictes ultérieurement**. Le reste des données, où nb_dossier est connu, a été conservé pour former l'ensemble d'entraînement et de validation des modèles (data_signif_train_test_val).

Code utilisé pour cette étape :

```
# La ou on a pas le nb de dossier (post 28/11/2024)
data_a_predire <- data_signif[is.na(data_signif$nb_dossier),]

# Le reste est utilisé pour entraîner le modèle
data_signif_train_test_val <- data_signif[!is.na(data_signif$nb_dossier),]
```

Perspectives

Cette sélection préliminaire réduit la dimensionnalité des données tout en conservant les variables les plus pertinentes pour expliquer l'activité quotidienne. Ces variables serviront de base à la création et à la comparaison des modèles dans la phase suivante.

Les modèles

Séparation des données en ensemble train, test et validation

Comme dans tout projet de modélisation, il est essentiel de disposer d'au moins deux ensembles de données : l'un pour entraîner le modèle, et l'autre pour tester ses performances et vérifier sa capacité de généralisation. Dans notre cas, en raison du nombre limité d'observations et de la difficulté à prédire la variable cible, nous avons alloué **90 % des données à l'entraînement**. Les 10 % restants ont été divisés de manière égale entre les ensembles de test et de validation. De plus, nous avons **fixé un seed** (graine) pour garantir la reproductibilité des ensembles de données, quel que soit le nombre d'exécutions.

Code utilisé pour cette étape :

```
# Division des données en ensembles d'entraînement, de validation et de test
set.seed(666)
split1 <- initial_split(data_signif_train_test_val, prop = 0.9)
trainData <- training(split1)
remainingData <- testing(split1)

split2 <- initial_split(remainingData, prop = 0.5)
valData <- training(split2)
testData <- testing(split2)

# Préparation des matrices pour les modèles
X_train <- as.matrix(trainData %>% select(-nb_dossier, -nb_dossier_normalise))
Y_train <- trainData$nb_dossier_normalise

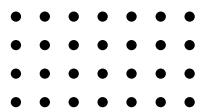
X_val <- as.matrix(valData %>% select(-nb_dossier, -nb_dossier_normalise))
Y_val <- valData$nb_dossier_normalise

X_test <- as.matrix(testData %>% select(-nb_dossier, -nb_dossier_normalise))
Y_test <- testData$nb_dossier_normalise
```

Perspectives

Cette étape est essentielle pour garantir une modélisation rigoureuse et fiable. La séparation des ensembles (entraînement, test, validation) permet d'évaluer la généralisabilité du modèle, d'ajuster ses hyperparamètres et d'éviter le surapprentissage. La fixation du seed assure la reproductibilité des résultats, favorisant une comparaison équitable des modèles. Enfin, elle peut révéler des limitations dans les données, orientant d'éventuelles améliorations.

Les modèles



La modélisation

Choix des modèles

Plusieurs modèles ont été testés pour la régression lors de cet étape mais uniquement deux ont été retenus. Voici les modèles que nous avons testés :



Une régression linéaire basique, ce modèle était assez satisfaisant mais faisait quelques grandes erreurs de prédiction



Une régression linéaire de type ridge, ce modèle était plus satisfaisant que la régression linéaire normale et a été retenue comme premier modèle.



L'algorithme des k plus proches voisins était très décevant, même en prenant des mesures de distances adapté à des variables binaires tel que la distance de Jaccard ou de Hamming.



Une régression par random forest. Ce modèle a mis beaucoup de temps à s'entraîner et cela sans beaucoup de résultats positifs.



Un réseau de neurones de type dense a été très efficace. Grâce à beaucoup de neurones, le modèle a réussi à bien prédire le nombre de dossier quotidien sans faire d'erreurs non acceptables. Nous avons donc retenu ce modèle.

Perspectives

Cette étape permet d'identifier les types de modèles pouvant prévoir au mieux l'activité d'IMA.

La régression ridge

La régression Ridge est une extension de la régression linéaire qui introduit une pénalité sur la magnitude des coefficients afin de limiter leur amplitude. Cette régularisation réduit le risque de surajustement en empêchant le modèle d'attribuer un poids excessif à des variables qui n'expliquent que partiellement le nombre de dossiers traités quotidiennement. Cela améliore la robustesse et la capacité de généralisation du modèle, en particulier lorsque les données présentent des corrélations importantes entre les variables explicatives ou un grand nombre de variables par rapport aux observations.

Code utilisé pour cette étape :

```
##### Ridge #####
lambda_seq <- 10^seq(3, -3, by = -0.1)
ridge_modele <- cv.glmnet(x_train, Y_train, alpha = 0, lambda = lambda_seq)

# Évaluation des performances sur les ensembles d'entraînement, de validation et de test
pred_train <- predict(ridge_modele, newx = X_train, s = "lambda.min")
pred_val <- predict(ridge_modele, newx = X_val, s = "lambda.min")
pred_test <- predict(ridge_modele, newx = X_test, s = "lambda.min")

# Calcul des R2 et des erreurs moyennes
r2_train <- calc_r2(Y_train,pred_train)
erreur_moyenne_train <- mean(abs(Y_train - pred_train))

r2_val <- calc_r2(Y_val,pred_val)
erreur_moyenne_val <- mean(abs(Y_val - pred_val))

r2_test <- calc_r2(Y_test,pred_test)
erreur_moyenne_test <- mean(abs(Y_test - pred_test))
```

Ce code permet de déterminer à quel point il faut punir la magnitude des coefficients en itérant sur des valeurs différentes dans une échelle logarithmique entre 3 et -3. Ensuite, pour évaluer la qualité du modèle, nous calculons pour chacun des trois ensembles train, test et validation :

- le R² qui permet de donner le pourcentage de la variance que le modèle est capable d'expliquer.
- L'erreur moyenne en valeur absolue.

Résultats :

Performance du modèle Ridge:

```
> cat("R2 Entraînement:", round(r2_train, 4), "\n")
R2 Entraînement: 0.8708
> cat("Erreur Moyenne Entraînement:", round(erreur_moyenne_train, 4), "\n")
Erreur Moyenne Entraînement: 2889.835
> cat("R2 Validation:", round(r2_val, 4), "\n")
R2 Validation: 0.8838
> cat("Erreur Moyenne Validation:", round(erreur_moyenne_val, 4), "\n")
Erreur Moyenne Validation: 2866.456
> cat("R2 Test:", round(r2_test, 4), "\n")
R2 Test: 0.8848
> cat("Erreur Moyenne Test:", round(erreur_moyenne_test, 4), "\n")
Erreur Moyenne Test: 2888.791
```



Les résultats obtenus sont globalement satisfaisants, car le modèle parvient à capturer une part significative de la variance. Cependant, l'erreur de prédiction reste élevée en raison de certaines prévisions aberrantes. Ces erreurs, bien que rares, peuvent avoir un impact significatif et potentiellement critique pour une entreprise comme IMA, qui dépend de prévisions précises pour son fonctionnement quotidien. Il est donc essentiel de développer un modèle plus robuste, capable de limiter ces écarts extrêmes, afin de garantir des estimations fiables et de renforcer la prise de décision opérationnelle.

Réseau de neurones

Notre deuxième modèle est un réseau de neurones de type dense et qui constitue en quelques sortes la partie ‘calculatoire’ d’une architecture de réseau de neurones. D’autres tentatives ont été réalisés en utilisant notamment une structure LSTM qui utilise la mémoire à court et long terme mais les résultats ont été très décevants.

Voici la structure du modèle :

```
model <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = 'relu', input_shape = c(ncol(x_train))) %>%
  layer_dense(units = 256, activation = 'relu') %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dense(units = 64, activation = 'relu') %>%
  layer_dense(units = 32, activation = 'relu') %>%
  layer_dense(units = 16, activation = 'relu') %>%
  layer_dense(units = 1)

# Compilation du modèle
model %>% compile(
  loss = loss_huber(delta = 1.5),
  optimizer = optimizer_adam(learning_rate = 0.01),
  metrics = c("mean_squared_error", "mean_absolute_error")
)

# Callback pour ajuster le taux d'apprentissage
reduce_lr <- callback_reduce_lr_on_plateau(
  monitor = "val_loss",
  factor = 0.5,
  patience = 5,
  min_lr = 1e-6
)
```

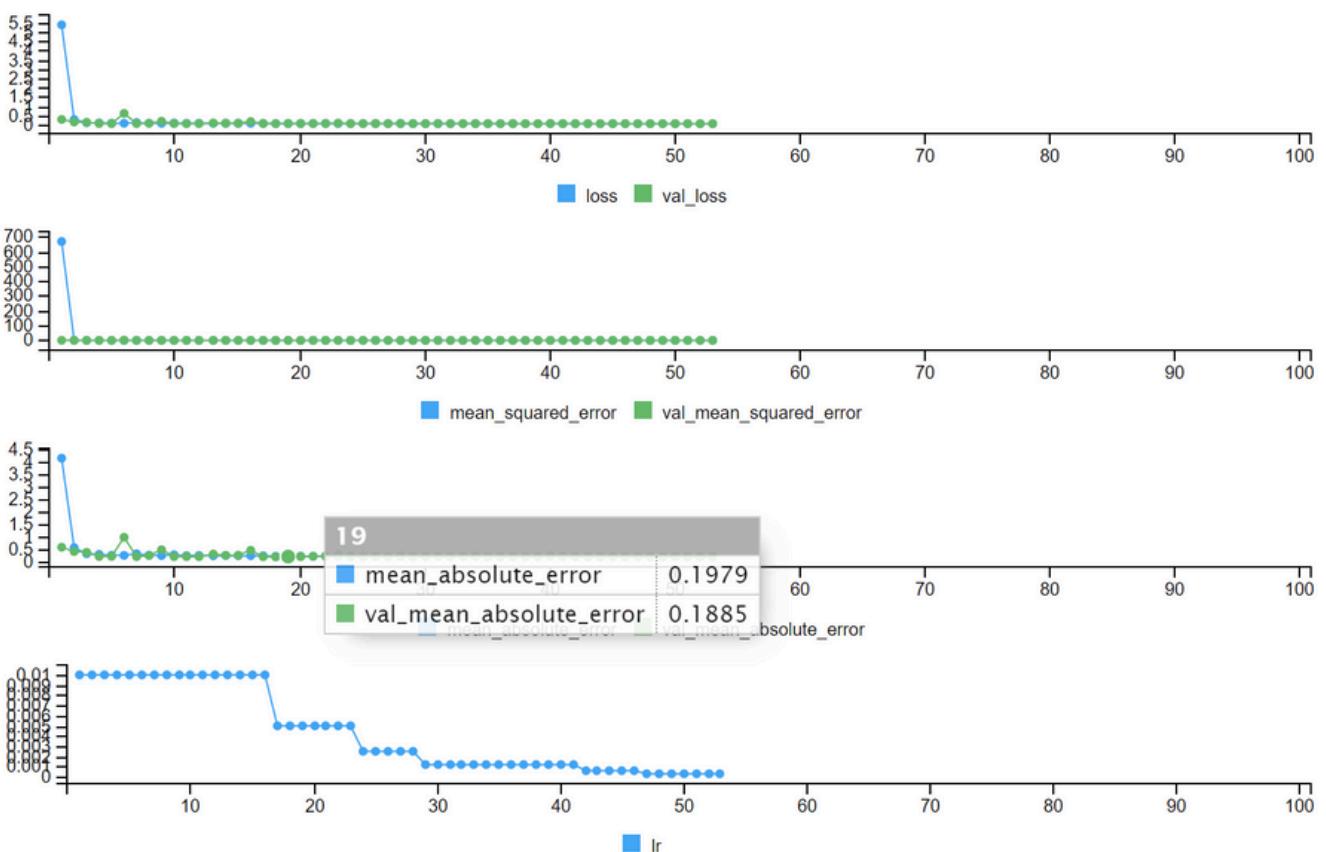


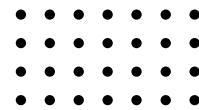
Ce modèle est construit à l'aide de **Keras** en R qui s'appuie sur Python. Il comprend **sept couches entièrement connectées** (Dense) organisées en une architecture de **taille décroissante** : la première couche comporte 512 neurones et reçoit comme entrée le nombre de caractéristiques de l'ensemble d'entraînement (X_train), suivie par des couches successives avec respectivement 256, 128, 64, 32, et 16 neurones. La dernière couche, avec un seul neurone, fournit une sortie adaptée à la prédiction de valeurs continues et donc du nombre de dossiers. Toutes les couches, sauf la dernière, utilisent la fonction d'activation **ReLU**.

Le modèle est compilé avec la fonction de **perte Huber**, qui permet de **bien prédire les valeurs aberrantes** qu'on peut observer lors de certains jours fériés ou la période COVID grâce à une transition entre l'erreur quadratique moyenne (MSE) et l'erreur absolue moyenne (MAE), contrôlée par un paramètre delta fixé à 1,5. L'optimisation est réalisée à l'aide de l'algorithme Adam, bien adapté aux réseaux de neurones, avec un taux d'apprentissage initial défini à 0,01. Les performances du modèle sont évaluées pendant l'entraînement à l'aide des métriques MSE et MAE.

Un **callback** est utilisé pour **ajuster dynamiquement le taux d'apprentissage** en fonction de la perte sur les données de validation (val_loss). Si cette métrique ne s'améliore pas après 5 époques/itérations consécutives (patience = 5), le taux d'apprentissage est réduit de moitié (factor = 0.5). Ce mécanisme permet d'**améliorer la convergence** du modèle en ajustant automatiquement l'apprentissage lorsque la perte atteint un plateau.

Voici à quoi ressemble les graphiques d'entraînement du modèle :





La tache étant très complexe, le modèle a parfois **du mal à converger vers une solution correcte**. C'est pour cela qu'une boucle a été mis en place qui crée un réseau de neurones jusqu'à ce qu'on soit en dessous d'un certain seuil d'erreur défini par l'utilisateur :

```
pourcentage_erreur_max <- 1
# Initialiser une variable pour stocker le pourcentage d'erreur
pourcentage_erreur <- 100 # Commencez avec un pourcentage d'erreur élevé

# Boucle jusqu'à ce que le pourcentage d'erreur soit inférieur à 1%
while (pourcentage_erreur >= pourcentage_erreur_max | pourcentage_erreur_max <=
pourcentage_erreur) {
```

En effet, la variable pourcentage_erreur_max définit un seuil d'erreur qu'on ne souhaite pas dépasser sur l'ensemble test. Ici ce seuil est de 1 %. Si on est en dessous, la boucle s'arrete les performances sont calculées et le modèle est ensuite enrégistrées.

Cela est fait grâce au code suivant :

```
# Évaluation du modèle
results <- model %>% evaluate(X_test, Y_test)
cat("Loss sur le test ACP :", results$loss, "\n")

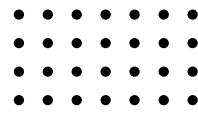
# Prédictions et calculs du R²
train_predictions <- model %>% predict(X_train)
val_predictions <- model %>% predict(X_val)
test_predictions <- model %>% predict(X_test)

train_r2 <- calc_r2(Y_train, train_predictions)
val_r2 <- calc_r2(Y_val, val_predictions)
test_r2 <- calc_r2(Y_test, test_predictions)

cat("R² sur l'entraînement ACP:", train_r2, "\n")
cat("R² sur la validation ACP:", val_r2, "\n")
cat("R² sur le test ACP:", test_r2, "\n")

# Sauvegarde du modèle de neurones
model %>% save_model("modele_neurones.keras")
```

Nous arrivons avec ce code à un R^2 de 0.94 ce qui est très élevé. Le reste du code est dédié à préparer la base de données des modèles pour l'exploitation du power BI.



D'abord, l'ensemble des données entre 2013 jusqu'en 2026 subissent les mêmes transformations que les données utilisées pour l'entraînement du modèle :

```
##### Prédiction sur l'ensemble des données #####
data_a_predire <- data

# Suppression des colonnes inutiles dans le dataset à prédire
data_a_predire$LB_JOUR <- NULL
data_a_predire$x <- NULL

# Ajustement de l'année (comme pour les données d'entraînement)
data_a_predire$NU_AN <- data_a_predire$NU_AN - 2013

# Appliquer les mêmes transformations d'encodage
encodage_data <- as.data.frame(predict(modele_encodage, newdata = data_a_predire))
data_a_predire <- cbind(data_a_predire, encodage_data)

# Suppression des colonnes d'origine après encodage
data_a_predire <- data_a_predire %>% select(-LB_SAISON, -LB_JOURAG, -LB_FERIE, -LB_JOURAG_Jm1, -LB_JOURAG_Jp1, -DT_JOUR)

# Appliquer la normalisation sur la cible (nb_dossier) si nécessaire
data_a_predire$nb_dossier_normalise <- (data_a_predire$nb_dossier - moyenne_y) / ec_type_y

# Préparation des matrices pour les prédictions
X_data <- as.matrix(data_a_predire %>% select(-nb_dossier, -nb_dossier_normalise))
```

Puis nous appliquons les deux modèles à ce jeu de données en chargeant le modèle de réseau de neurones déjà entraîné :

```
# Prédiction Ridge sur les nouvelles données
predictions_ridge <- predict(ridge_modele, newx = X_data, s = "lambda.min")

# Charger le modèle de neurones sauvegardé et faire des prédictions
loaded_model <- load_model("modele_neurones.keras")
predictions_reseau <- predict(loaded_model, X_data)
```

Pour finalement ajouter au jeu de données entier les prédictions des deux modèles (dénormalisés) puis nous sélectionnons les colonnes utiles pour ensuite l'exporter en format csv.

```
# Réversal de la normalisation pour obtenir les valeurs prédites originales de nb_dossier
data$nb_dossier_m1 <- as.integer(predictions_ridge * ec_type_y + moyenne_y)
data$nb_dossier_m2 <- as.integer(predictions_reseau * ec_type_y + moyenne_y)

# Ajouter une colonne pour segmenter les résultats
data$LB_SEGMENT <- "DD01"

# Sélectionner les colonnes finales pour l'export
data_final <- data %>%
  select(DT_JOUR, LB_SAISON, LB_JOUR, LB_SEGMENT, NU_SEMN_MOIS, LB_FERIE, NU_SEMN, nb_dossier, nb_dossier_m1, nb_dossier_m2)

# Sauvegarde des résultats dans un fichier csv
write.csv2(data_final, "data_predite.csv")
```

Tout les fichiers sont désormais prêts pour l'exploitation sous Power-BI.

Power BI

Le tableau de bord réalisé sur **Power BI** comporte **3 pages** avec des fonctionnalités et des enjeux différents, voici une brève explication du contenu de chaque page :

Page 1 - Analyse de l'activité par segment

Cette page fournit une analyse détaillée de la répartition de l'activité par segment d'assistance (**Moins de 50 et plus de 50 km**).

- Le **graphique principal** montre la répartition de l'activité entre les différents segments sur la période de **janvier 2013 à novembre 2024**.
- Les **tableaux à droite de la page** listent les **5 principaux clients** ainsi que les **5 périodes et jours fériés les plus chargés**.
- Les **filtres en haut** permettent de sélectionner l'**année, la saison, le mois, le jour de la semaine** ainsi que le **segment et le client**.

Page 2 - Tendances temporelles

Cette page se concentre sur l'analyse des **tendances saisonnières, mensuelles et journalières** de l'activité.

- Le **graphique en arbre** en haut à gauche montre la décomposition des dossiers par période temporelle (**année, mois, jours, jours fériés**). Il permet d'identifier des jours où l'activité était importante.
- Le **graphique en barre** en haut montre l'évolution du **nombre de dossiers moyen par semaine** sur les années **2013 à 2024**, mettant en évidence les variations saisonnières et les jours les plus chargés.
- La **courbe du bas** présente le **nombre moyen de dossiers par mois**, pour identifier les mois les plus chargés.
- Le **tableau de données** montre le **nombre de dossiers moyen par mois et par jour**, localisant des mois ou certains jours particulièrement chargés.
- Les **filtres en haut** offrent la possibilité de se concentrer sur l'**année, la saison, le mois, le jour de la semaine** ainsi que le **segment et le client**.

Page 3 - Suivi de l'activité d'assistance et des prévisions

Ce tableau de bord offre une vue d'ensemble de l'activité réelle et la comparaison des prédictions des deux modèles sur la période de **janvier 2013 à janvier 2026**.

- Le **graphique principal** montre l'évolution du **volume réel, des volumes prévus par le modèle 1 et 2** au fil du temps, permettant de visualiser les écarts entre les données réelles et les prévisions.
- Les **indicateurs clés** à gauche de la page affichent les **volumes de dossiers réels, la précision et l'erreur moyenne** des modèles 1 et 2.
- Les **filtres en haut** offrent la possibilité de se concentrer sur l'**année, la saison, le mois, le jour de la semaine** ainsi que le **segment et le client**.

Conclusion

À l'issue de cette étude, nous avons **atteint les objectifs** fixés par Inter Mutuelles Assistance : fournir un **modèle statistique robuste** permettant de **prévoir l'activité quotidienne** sur les plateaux d'assistance avec un excellent niveau de précision. Grâce à une **démarche rigoureuse** comprenant l'audit des données, leur préparation et l'enrichissement par des variables pertinentes, ainsi qu'une modélisation optimisée, nous avons obtenu un coefficient de détermination **R² de 94%**.

Ce résultat témoigne de la **qualité de notre approche** et de la capacité du modèle à expliquer la variabilité des volumes d'activité observés. En pratique, il permettra à l'entreprise **d'anticiper les besoins en personnel** et de **garantir une réactivité accrue** face aux demandes des automobilistes, répondant ainsi à la mission critique de l'organisation.

En complément, l'outil de suivi interactif développé sur **Power BI** offre une **interface intuitive** pour **visualiser les prévisions** et **ajuster les ressources** en temps réel. Il s'agit d'un levier essentiel pour **favoriser une prise de décision rapide et informée** par les équipes opérationnelles.

Nous recommandons cependant de **maintenir une vigilance** quant à la mise à jour des données et à l'intégration continue d'éventuels facteurs externes (par exemple, les conditions climatiques ou les nouvelles tendances de mobilité) afin de **préserver la performance et la pertinence du modèle à long terme**.

Ce projet a permis de **démontrer le rôle stratégique des outils statistiques et décisionnels** dans l'optimisation des services. Nous remercions Inter Mutuelles Assistance pour leur confiance et restons disponibles pour les accompagner dans l'évolution et le déploiement des solutions proposées.

ANNEXE

Annexe

Audit des données

```
# 1. Charger les données
file_path <- "bdd_projet.csv"
data <- read.csv(file_path, sep = ";", na.strings = c("", "NA", "NULL"), fileEncoding = "Latin1")

# Conversion de la colonne 'dateouv' au format Date
data$dateouv <- as.Date(data$dateouv, format = "%Y-%m-%d")

# Vérifier si 'RL_DOSS' est correctement interprété comme numérique
data$RL_DOSS <- as.numeric(data$RL_DOSS)

# 2. Aperçu des données
print("Aperçu des données :")
glimpse(data)

# 3. Statistiques descriptives globales
print("Résumé statistique des données :")
summary(data)
> # 2. Aperçu des données
> print("Aperçu des données :")
[1] "Aperçu des données :"
> glimpse(data)
Rows: 3,995,263
Columns: 8
$ site           <lg> FALSE, FALSE, FALSE
$ dateouv        <date> 2013-01-01, 2013-0
$ client          <chr> "ZLT", "ZLT", "ZNZ"
$ segment         <chr> "DD01", "DD01", "DD
$ lb_segment      <chr> "Assistance à moins
$ lb_detail_cause <chr> "Accident matériel"
$ NBKMSORI       <int> NA, NA, NA, NA, NA,
$ RL_DOSS         <dbl> 1, 1, 7, 2, 1, 4, 1

> # 3. Statistiques descriptives globales
> print("Résumé statistique des données :")
[1] "Résumé statistique des données :"
> summary(data)
   site          dateouv            client      segment
  Mode :logical  Min.   :2013-01-01 Length:3995263
  FALSE:3995263  1st Qu.:2019-06-17 Class  :character
                           Median :2021-06-23 Mode   :character
                           Mean   :2021-03-06
                           3rd Qu.:2023-03-28
                           Max.   :2024-11-18

   lb_detail_cause    NBKMSORI      RL_DOSS
  Length:3995263  Min.   : 0.0  Min.   : 0.000
  Class  :character  1st Qu.: 13.0  1st Qu.: 1.000
  Mode   :character  Median : 50.0  Median : 1.000
                           Mean   : 169.6  Mean   : 3.733
                           3rd Qu.: 221.0  3rd Qu.: 2.000
                           Max.   :9798.0   Max.   :1175.000
                           NA's   :259959

# 4. Vérification des valeurs manquantes
print("Nombre de valeurs manquantes par colonne :")
missing_values <- colSums(is.na(data))
print(missing_values)

# 5. Vérification des doublons
print("Nombre de doublons dans les données :")
duplicates <- nrow(data) - nrow(distinct(data))
print(duplicates)

# 6. Identification des anomalies dans les colonnes clés
# Vérifier les anomalies dans lb_segment et lb_detail_ca
print("Valeurs uniques dans lb_segment :")
print(unique(data$lb_segment))

print("Valeurs uniques dans lb_detail_cause :")
print(unique(data$lb_detail_cause))

# 4. Vérification des valeurs manquantes
> print("Nombre de valeurs manquantes par colonne :")
[1] "Nombre de valeurs manquantes par colonne :"
> missing_values <- colSums(is.na(data))
> print(missing_values)
   site          dateouv            client      segment
  0             0                 0             0
  NBKMSORI      RL_DOSS
  259959         0

# 5. Vérification des doublons
> print("Nombre de doublons dans les données :")
[1] "Nombre de doublons dans les données :"
> duplicates <- nrow(data) - nrow(distinct(data))
> print(duplicates)
[1] 0

# 6. Identification des anomalies dans les colonnes clés
# Vérifier les anomalies dans lb_segment et lb_detail_cause
print("Valeurs uniques dans lb_segment :")
print(unique(data$lb_segment))
[1] "Assistance à moins de 50km" "Assistance à plus de 50km"
> print("Valeurs uniques dans lb_detail_cause :")
[1] "Valeurs uniques dans lb_detail_cause :"
> print(unique(data$lb_detail_cause))
[1] "Accident matériel"          "clés(Enfermement/Dysfonct" "C
[4] "Incendie véhicule"          "Panne mécanique"          "V
[7] "Tenta.Vol Vandali.Veh"      "Vol de véhicule"          "B
[10] "Carbur(panne/erreur)"       "clés(Perte/Vol)"          "ai
```