MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE RECHERCHE SCIENTIFIQUE

REPUBLIQUE DE COTE D'IVOIRE
UNION-DISCIPLINE-TRAVAIL

PROJET

ECONOMETRIE DES VARIABLES QUANTITATIVES

ETUDIANTE

KOUAHON ESTELLE

PROFESSEUR

AKPOSSO MARTIAL

AVANT-PROPOS

Dans le contexte actuel de croissance rapide du secteur de la livraison de repas, la prédiction précise des délais de livraison est devenue un enjeu crucial pour optimiser les opérations, améliorer la satisfaction client et maîtriser les coûts. Cette étude s'inscrit dans cette problématique en explorant l'application de modèles statistiques pour prédire les délais de livraison en fonction de divers facteurs d'influence.

Ce projet a pour objectif principal de comparer deux approches de modélisation : la régression linéaire multiple et l'analyse de variance (ANOVA), afin de déterminer laquelle est la plus adaptée pour prédire les délais de livraison à partir d'un ensemble de données réelles. Nous avons analysé un jeu de données de 1000 observations, contenant 9 variables pertinentes telles que la distance, les conditions météorologiques, le niveau de trafic, etc.

Je tiens à remercier sincèrement [mentionner les noms des personnes qui ont contribué au projet, par exemple, les professeurs, les mentors, les collaborateurs] pour leur soutien, leurs conseils et leur expertise tout au long de ce projet. Leurs contributions ont été essentielles pour la réalisation de cette étude.

Ce rapport présente les différentes étapes de notre démarche, depuis la préparation des données jusqu'à la comparaison des modèles, en passant par l'analyse descriptive, la modélisation et la validation des résultats. Nous espérons que cette étude apportera des éclaircissements précieux sur la prédiction des délais de livraison et contribuera à l'amélioration des pratiques dans ce domaine en pleine expansion.

SOMMAIRE

INTRODUCTION

Dans le contexte actuel de croissance rapide des services de livraison de repas, la prédiction précise des délais de livraison est devenue un enjeu crucial pour optimiser les opérations et améliorer la satisfaction client. Cette étude se concentre sur le développement et la comparaison de deux modèles statistiques pour prédire les délais de livraison en fonction de divers facteurs d'influence : la régression linéaire multiple et l'analyse de variance (ANOVA).

La question centrale à laquelle nous répondrons est la suivante : lequel de ces deux modèles est le plus approprié pour prédire les délais de livraison en fonction des variables disponibles dans notre jeu de données ?

Objectif Général

Évaluer et comparer les performances des modèles de régression linéaire multiple et d'ANOVA pour prédire les délais de livraison de repas en fonction de facteurs tels que la distance, les conditions météorologiques, le trafic, l'heure de la journée, le type de véhicule, le temps de préparation et l'expérience du coursier.

Objectifs Spécifiques

- **Modélisation par Régression Linéaire Multiple**: Construire un modèle de régression linéaire multiple pour prédire le délai de livraison en utilisant toutes les variables explicatives disponibles.

- **Modélisation par ANOVA**: Développer un modèle ANOVA pour évaluer l'impact des conditions météorologiques sur le délai de livraison.

- **Comparaison des Modèles**: Comparer les performances des deux modèles en termes de précision prédictive et d'adéquation aux données.

- **Validation des Hypothèses**: Vérifier les hypothèses sous-jacentes à chaque modèle (linéarité, normalité des résidus, homoscédasticité, etc.) pour garantir la validité des résultats.

Résultats Attendus

- Un modèle de régression linéaire multiple précis pour prédire les délais de livraison.

- Un modèle ANOVA quantifiant l'impact des conditions météorologiques sur les délais de livraison.
- Une comparaison rigoureuse des deux modèles, identifiant leurs forces et faiblesses respectives.
- Une validation approfondie des hypothèses statistiques pour assurer la robustesse des conclusions.

Méthodologie

- ❖ Collecte et Préparation des Données
- ❖ Analyse Descriptive
- ❖ Régression Linéaire Multiple
- ❖ Analyse de Variance (ANOVA)
- ❖ Comparaison des Modèles
- ❖ Conclusion:

DICTIONNAIRE DE DONNEES

VARIABLES	TYPE	MODALITES	DESCRIPTION
Order_ID	Numérique (entier)	-	Identifiant unique pour chaque commande. Permet d'identifier chaque commande de manière unique.
Distance_km	Numérique (continue)	**Afternoon, Evening, Night, Morning**	La distance de livraison en kilomètres. Distance entre le point de départ et le point de livraison
Weather	Catégoriel (nominale)	**Windy, Clear, Foggy, Rainy, Snowy**	Conditions météorologiques pendant la livraison.
Traffic_Level	Catégoriel (ordinale)	**Low, Medium, High**	Conditions de trafic. Niveau de congestion du trafic lors de la livraison.
Time_of_Day	Catégoriel (nominale)	-	L'heure à laquelle la livraison a eu lieu. Période de la journée à laquelle la livraison a été effectuée.
Vehicle_Type	Catégoriel (nominale)	-	Type de véhicule utilisé pour la livraison.
Preparation_Time_min	Numérique (continue)	-	Temps nécessaire à la préparation de la commande. Durée en minutes nécessaire pour préparer la commande.
Courier_Experience_yrs	Numérique (continue)	-	Expérience du coursier. Nombre d'années d'expérience du coursier.
Delivery_Time_min	Numérique (continue)	-	Délai de livraison total. Durée totale en minutes entre la commande et la livraison (variable cible).

PARTIE I : PRETRAITEMENT DES DONNEES

Le prétraitement est vital pour la qualité de nos prédictions de délais de livraison. Il corrige les données brutes (valeurs manquantes, etc.) et transforme les variables pour optimiser le modèle.

IMPORTATION DES DONNEES

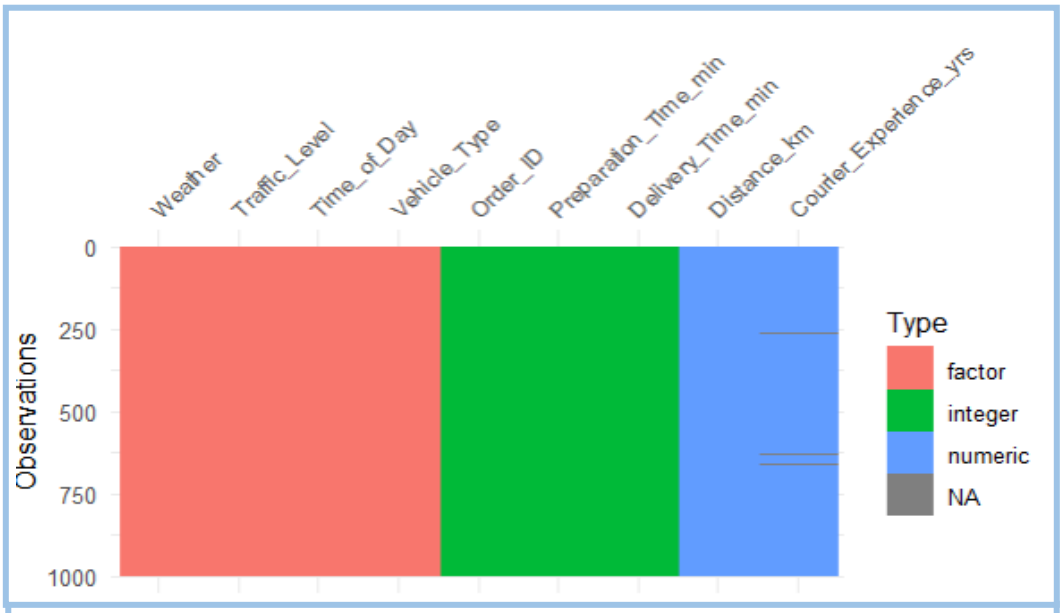
Les données utilisées dans cette étude proviennent de Kaggle, une plateforme de référence pour les data scientists et les data analysts du monde entier. Kaggle offre un large éventail de jeux de données couvrant divers domaines, ce qui en fait une ressource précieuse pour la recherche et l'apprentissage automatique. Notre jeu de données, spécifiquement conçu pour l'analyse des délais de livraison de nourriture, est composé de 1000 observations et 9 variables.

	Order_ID <int>	Distance_km <dbl>	Weather <fctr>	Traffic_Level <fctr>
1	522	7.93	Windy	Low
2	738	16.42	Clear	Medium
3	741	9.52	Foggy	Low
4	661	7.44	Rainy	Medium
5	412	19.03	Clear	Low
6	679	19.40	Clear	Low

TRAITEMENT DES VALEURS MANQUANTES

Dans cette partie, nous expliquons clairement nos choix de méthodes de traitement des valeurs manquantes et nous les justifions en fonction du type de variable, du pourcentage de valeurs manquantes et des avantages des méthodes choisies.

2.1- Visualisation avant traitement des valeurs manquantes




```
'data.frame':   1000 obs. of  9 variables:
 $ Order_ID              : int  522 738 741 661 412 679 627 514 860
137 ...
 $ Distance_km           : num  7.93 16.42 9.52 7.44 19.03 ...
 $ Weather               : Factor w/ 6 levels
"","Clear","Foggy",..: 6 2 3 4 2 2 2 2 5 3 ...
 $ Traffic_Level         : Factor w/ 4 levels "","High","Low",..: 3
4 3 4 3 3 3 4 3 3 ...
 $ Time_of_Day           : Factor w/ 5 levels "","Afternoon",..: 2
3 5 2 4 3 1 3 3 3 ...
 $ Vehicle_Type          : Factor w/ 3 levels
"Bike","Car","Scooter": 3 1 3 3 1 3 1 3 2 3 ...
 $ Preparation_Time_min  : int  12 20 28 5 16 8 12 5 20 29 ...
 $ Courier_Experience_yrs: num  1 2 1 1 5 9 1 6 6 1 ...
 $ Delivery_Time_min     : int  43 84 59 37 68 57 49 46 35 73 ...
```

Sur le graphe de visualisation des valeurs manquantes, on constate que seul la variable Courier_Experience_yrs contient des valeurs manquantes alors que la structure du jeu de données montre clairement qu'il existe, au niveau des variables catégorielles : Weather, Traffic_Level et Time_of_Day de types factor contiennent des niveaux de facteurs vides (" ") Que l'algorithme de R ne reconnait pas comme étant dans NA. C'est pourquoi, dans le graphique de visualisation avant le traitement des valeurs manquantes, nous ne voyez pas de segments gris (NA) pour ces variables.

Les niveaux de facteurs vides sont présents dans les données, mais ils ne sont pas interprétés comme des valeurs manquantes par R. Pour pouvoir traiter correctement ces valeurs manquantes, nous avons transformé les niveaux de facteurs vides (" ") en NA. Cette étape est essentielle, car elle permet à R de les reconnaître et de gérer ces valeurs manquantes de manière appropriée lors des analyses.

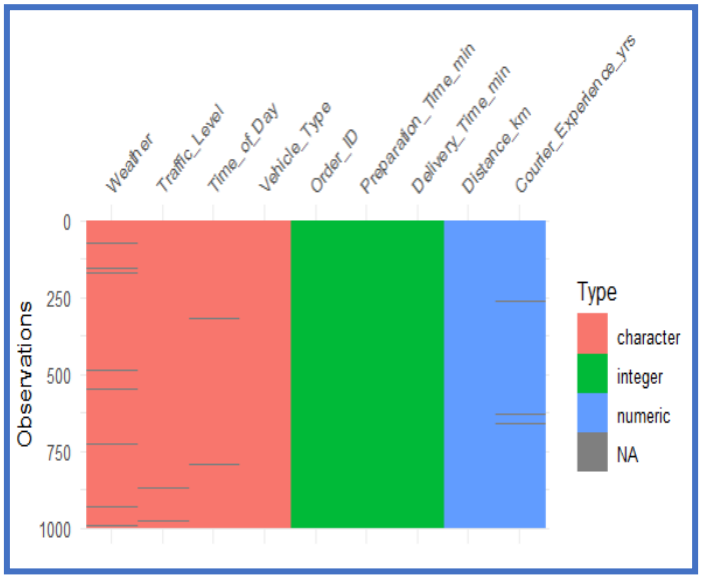
2.2- visualisation après traitement des valeurs manquantes

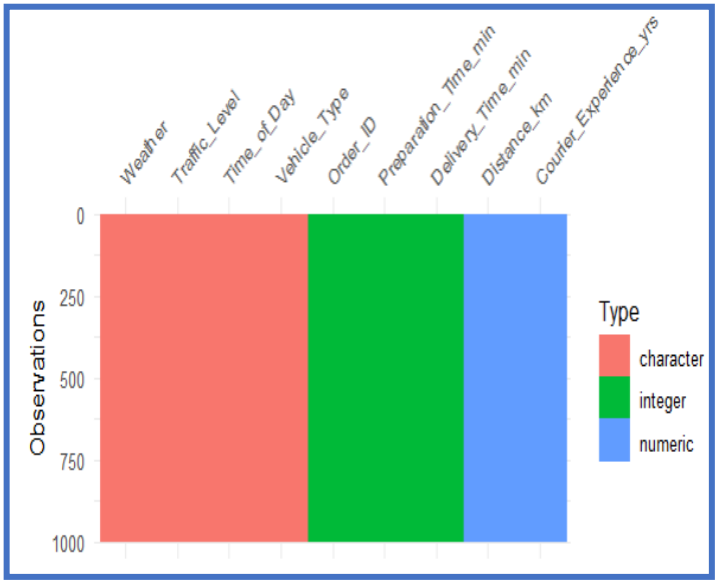
NIVEAU VIDE AVANT TRAITEMENT

```
'data.frame':   1000 obs. of  9 variables:
 $ Order_ID            : int  522 738 741 661 412 679 627 514 860
137 ...
 $ Distance_km         : num  7.93 16.42 9.52 7.44 19.03 ...
 $ Weather             : Factor w/ 6 levels
"","Clear","Foggy",..: 6 2 3 4 2 2 2 2 5 3 ...
 $ Traffic_Level       : Factor w/ 4 levels "","High","Low",..: 3
4 3 4 3 3 3 4 3 3 ...
 $ Time_of_Day         : Factor w/ 5 levels "","Afternoon",..: 2
3 5 2 4 3 1 3 3 3 ...
 $ Vehicle_Type        : Factor w/ 3 levels
"Bike","Car","Scooter": 3 1 3 3 1 3 1 3 2 3 ...
 $ Preparation_Time_min : int  12 20 28 5 16 8 12 5 20 29 ...
 $ Courier_Experience_yrs: num  1 2 1 1 5 9 1 6 6 1 ...
 $ Delivery_Time_min    : int  43 84 59 37 68 57 49 46 35 73 ...
```

MODIFICATION NIVEAU VIDE EN NA

```
'data.frame':   1000 obs. of  9 variables:
 $ Order_ID            : int  522 738 741 661 412 679 627 514 860
137 ...
 $ Distance_km         : num  7.93 16.42 9.52 7.44 19.03 ...
 $ Weather             : chr  "Windy" "Clear" "Foggy" "Rainy" ...
 $ Traffic_Level       : chr  "Low" "Medium" "Low" "Medium" ...
 $ Time_of_Day         : chr  "Afternoon" "Evening" "Night"
"Afternoon" ...
 $ Vehicle_Type        : chr  "Scooter" "Bike" "Scooter" "Scooter"
...
 $ Preparation_Time_min : int  12 20 28 5 16 8 12 5 20 29 ...
 $ Courier_Experience_yrs: num  1 2 1 1 5 9 1 6 6 1 ...
 $ Delivery_Time_min    : int  43 84 59 37 68 57 49 46 35 73 ...
```





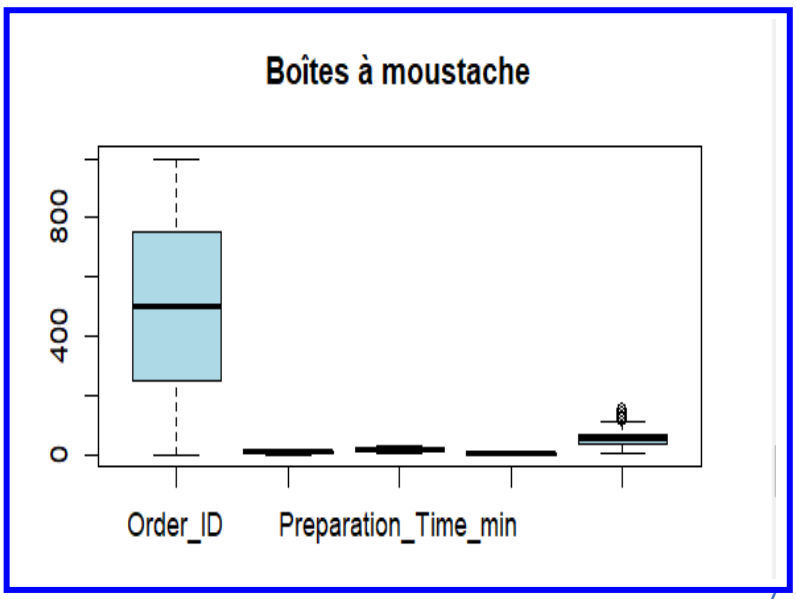
Après avoir transformer les niveaux vides en NA, on visualise clairement maintenant les valeurs manquantes. Bien que le pourcentage total de valeurs manquantes dans votre jeu de données soit faible (1,33%), il est crucial de ne pas supprimer les observations contenant des valeurs manquantes, en particulier dans le contexte d'un projet de prédiction. Dans un projet de prédiction, l'objectif est de prédire le délai de

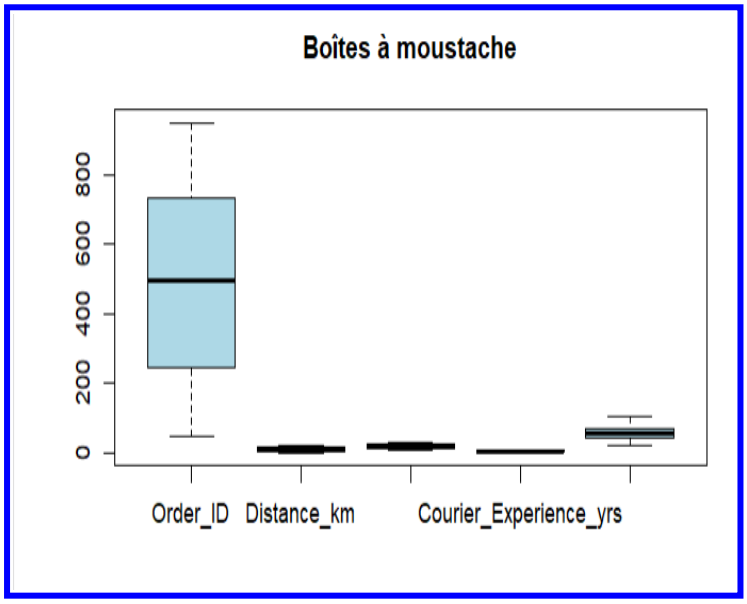
livraison pour toutes les commandes, y compris celles pour lesquelles certaines informations sont manquantes. Par conséquent, il est préférable de traiter les valeurs manquantes plutôt que de les supprimer. Même si le pourcentage est faible, la suppression de lignes pourrait entraîner une perte d'informations précieuses pour la prédiction des délais de livraison. Si les valeurs manquantes ne sont pas aléatoires (c'est-à-dire qu'il y a une raison sous-jacente à leur absence), la suppression des observations pourrait introduire un biais dans votre modèle. L'imputation par le mode est la méthode que nous avons utilisée pour traiter les variables catégorielles de types factor, Cette méthode consiste à remplacer les valeurs manquantes par la catégorie la plus fréquente dans chaque variable. L'imputation par le mode est particulièrement adaptée aux variables catégorielles, car elle préserve la distribution originale des catégories et offre une interprétation simple et intuitive des résultats. En ce qui concerne la variable numérique Courier_Experience_yrs, qui contient 30 valeurs manquantes, nous avons opté pour la méthode KNNImputation. Cette technique d'imputation utilise les valeurs des k plus proches voisins pour estimer les valeurs manquantes, offrant ainsi une plus grande précision que l'imputation par la moyenne ou la médiane.

TRAITEMENT DES DOUBLONS

La visualisation des doublons consiste à identifier les observations (lignes) identiques sur l'ensemble des variables. Les résultats de cette vérification ont confirmé l'absence de doublons, assurant ainsi l'intégrité des données pour la suite de l'étude.

TRAITEMENT DES VALURS ABERRANTES ET EXTREMES





Les boîtes à moustaches montrent que la variable Order_ID a une grande dispersion avec une valeur très éloignée des autres. Cela suggère des valeurs extrêmes ou des erreurs potentielles. Pour Preparation_Time_min, quelques points isolés au-dessus de la boîte indiquent aussi des valeurs potentiellement aberrantes. Il faudra examiner ces valeurs de plus près pour décider comment les traiter avant de construire nos modèles.

Après traitement des valeurs extrêmes et aberrantes, la dispersion des valeurs a considérablement diminué pour Order_ID et Preparation_Time_min. L'échelle de l'axe des ordonnées est plus petite, ce qui suggère que les valeurs extrêmes ont été traitées. Les distributions semblent maintenant plus homogènes et concentrées autour de la médiane.

DIVISION DES DONNEES EN ENSEMBLES D'ENTRAINEMENT ET DE TEST

Dans l'optique de s'assurer de la capacité de notre modèle à faire des prédictions sur de nouvelles données autres que nos données d'origine, nous allons diviser les données en deux catégories, en données test et en données d'entrainement.

```
Taille de l'ensemble d'entraînement : 801
Taille de l'ensemble de test : 199
```

ENCODAGE DES VARIABLES CATEGORIELLES

Dans le cadre de cette étude portant sur la prédiction des délais de livraison, l'encodage des variables catégorielles telles que Weather, Traffic_Level et Time_of_Day est une étape essentielle. Les modèles de régression multiple, que nous utilisons ici, fonctionnent avec des données numériques. Or, ces variables sont de nature catégorielle, c'est-à-dire qu'elles représentent des catégories ou des modalités plutôt que des valeurs numériques. Nous utilisons la méthode d'encodage "one-hot" pour ces variables polytomiques (c'est-à-dire avec plus de deux niveaux).

	WeatherClear	WeatherWindy	WeatherFoggy	WeatherRainy	WeatherSnowy	Traffic_LevelMedium
1	0	1	0	0	0	
6	1	0	0	0	0	
9	0	0	0	0	1	
14	0	0	0	1	0	
42	1	0	0	0	0	

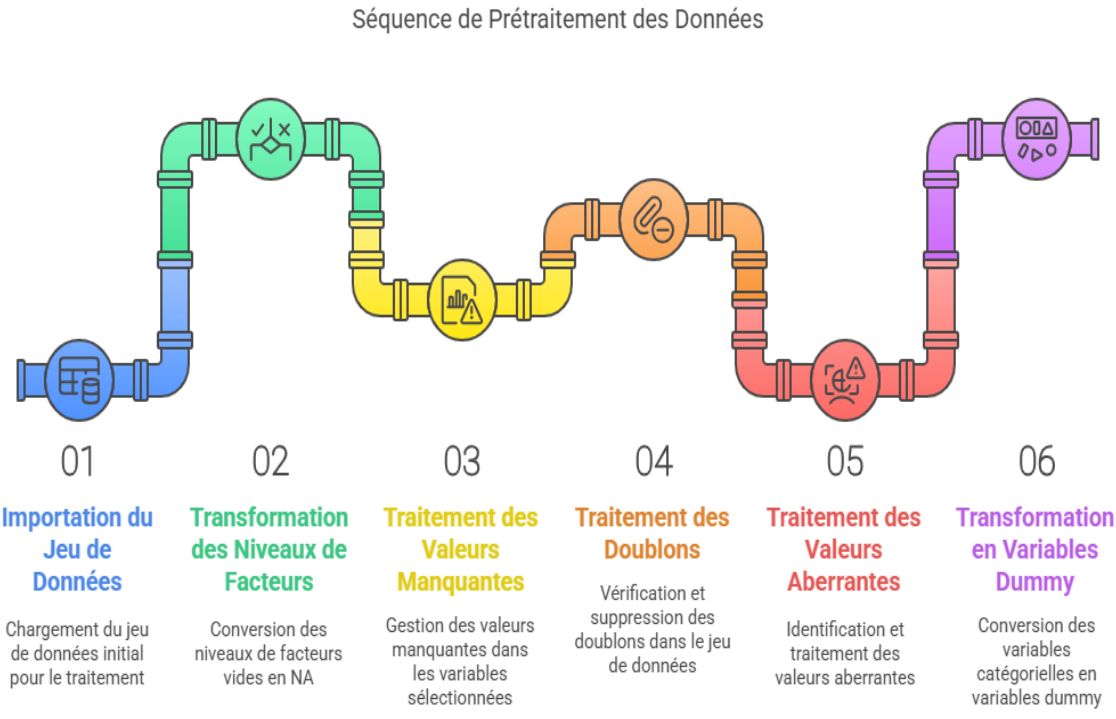
STANDARDISATION OU NORMALISATION DES ENSEMEBLES TEST

6.1- Ensemble Test

	WeatherClear	WeatherWindy	WeatherFoggy	WeatherRainy	WeatherSnowy	Traffic_LevelMedium
1	-1.0056335	3.2295232	-0.3555619	-0.514851	-0.3092567	-0.8670614
6	0.9931566	-0.3092567	-0.3555619	-0.514851	-0.3092567	-0.8670614
9	-1.0056335	-0.3092567	-0.3555619	-0.514851	3.2295232	-0.8670614
14	-1.0056335	-0.3092567	-0.3555619	1.939885	-0.3092567	1.1518810
42	0.9931566	-0.3092567	-0.3555619	-0.514851	-0.3092567	1.1518810

6.2- Ensemble Entrainement

	WeatherClear	WeatherWindy	WeatherFoggy	WeatherRainy	WeatherSnowy	Traffic_LevelMedium
2	0.9931566	-0.3092567	-0.3555619	-0.514851	-0.3092567	1.1518810
3	-1.0056335	-0.3092567	2.8089388	-0.514851	-0.3092567	-0.8670614
4	-1.0056335	-0.3092567	-0.3555619	1.939885	-0.3092567	1.1518810
5	0.9931566	-0.3092567	-0.3555619	-0.514851	-0.3092567	-0.8670614
7	0.9931566	-0.3092567	-0.3555619	-0.514851	-0.3092567	-0.8670614



PARTIE II : MODELISATION EN REGRESSION LINEAIRE MULTIPLE

La RLM modélise la relation linéaire entre nos variables indépendantes et Delivery_Time_min, permettant ainsi d'identifier les facteurs ayant un impact significatif. L'objectif est d'obtenir des prédictions précises et d'améliorer la compréhension des facteurs influençant les délais.

1- VERIFICATION DE L'EXISTENCE DE LA COLINEARITE

1.1- Détermination du VIF

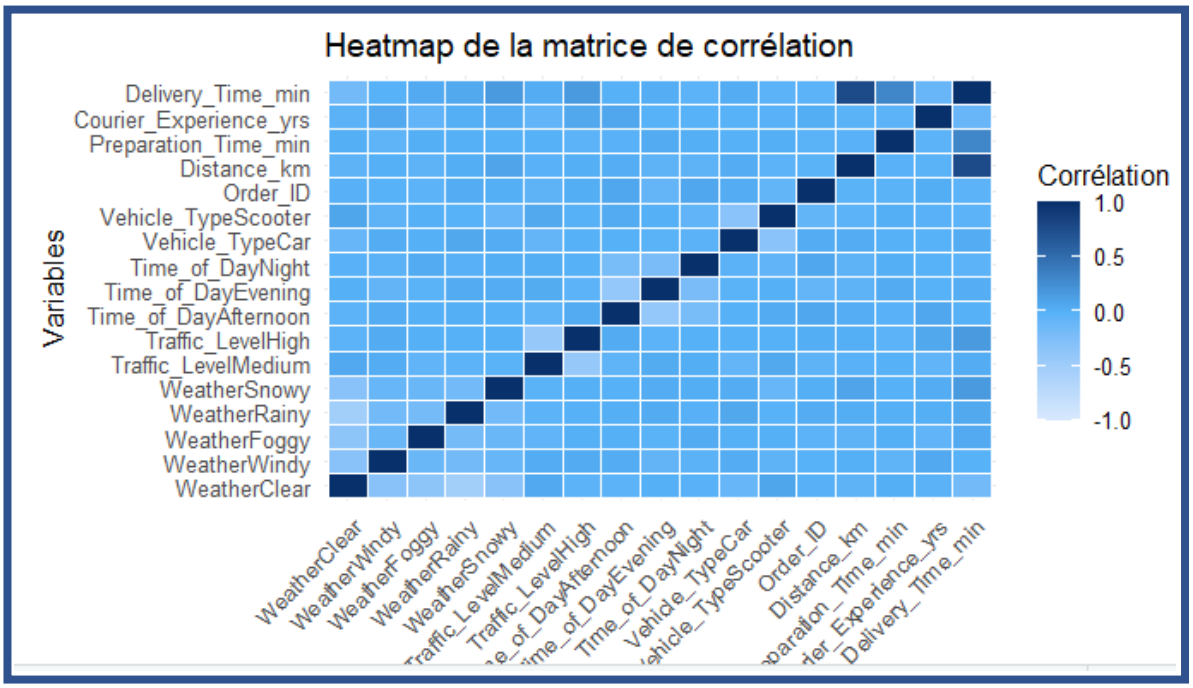
Les VIF (Variance Inflation Factors) sont des mesures statistiques qui permettent de détecter la **colinéarité** (ou multicolinéarité) entre les variables. La colinéarité se produit lorsque deux ou plusieurs variables indépendantes dans un modèle de régression sont fortement corrélées entre elles. La détermination du VIF concerne les variables : Distance_km, Preparation_Time_min, Courier_Experience_yrs. Bien que Order_ID soit numérique, il s'agit d'un identifiant unique pour chaque commande. Il n'est pas pertinent de l'inclure dans le calcul du VIF car il ne représente pas une caractéristique mesurable qui pourrait être corrélée avec d'autres variables. Ce sont les valeurs standardisées de ces variables qui seront utilisés pour calculer le VIF. La standardisation permet de ramener toutes les variables à une échelle commune, ce qui facilite la comparaison et l'interprétation des coefficients de régression.

```
[1] "VIF :"
          Distance_km   Preparation_Time_min Courier_Experience_yrs
             1.000634              1.001303              1.001471
```

Dans le tableau, les VIF pour les trois variables sont très proches de 1. En général, on considère qu'il n'y a pas de colinéarité problématique si les VIF sont inférieurs à 5 ou 10. Ici, les VIF sont de l'ordre de 1, ce qui indique une **absence de colinéarité significative** entre ces variables.

1.2- Matrice de Corrélation

Le heatmap montre une absence de corrélation forte, ce qui confirme nos résultats du VIF et indique qu'il n'y a pas de problème de multicolinéarité. La matrice de corrélation. Il existe une corrélation positive entre Distance_km et Courier_Experience_yrs. Cependant, cette corrélation n'est pas suffisamment forte pour causer un problème de multicolinéarité.



3- CHOIX DE LA METHODE D'ESTIMATION

3.1-Méthode : Régression linéaire multiple (RLM)

La Régression Linéaire Multiple (RLM) modélise la relation linéaire entre une variable dépendante (cible) et plusieurs variables indépendantes (prédicteurs).

3.2-Principe

L'équation de la RLM est :

$$Y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots + \beta nXn + \varepsilon$$

Où :

- Y : variable dépendante (Delivery_Time_min).
- X1, X2, ..., Xn : variables indépendantes.
- β0, β1, ..., βn : coefficients de régression (impact de chaque variable indépendante sur la variable dépendante).
- **ε** : terme d'erreur (variation de Y non expliquée par les variables indépendantes).

3.3 -Objectif

L'objectif de la RLM est de prédire la variable dépendante (Delivery_Time_min) à partir des variables indépendantes.

4- ESTIMATION DES PARAMETRES ET CONSTRUCTION DES INTERVALLES DE CONFIANCE

On utilise la méthode des moindres carrés ordinaires (MCO) pour estimer les coefficients de régression qui minimisent l'erreur entre les valeurs prédites et réelles de la variable dépendante.

4.1- Vérification de la significativité des paramètres par intervalle de confiance.

Un intervalle de confiance est un intervalle de valeurs dans lequel on estime qu'un paramètre (comme un coefficient de régression) a de fortes chances de se trouver.

Si l'intervalle de confiance d'un paramètre **ne contient pas zéro**, cela signifie que ce paramètre est statistiquement significatif. On peut conclure que son impact sur la variable dépendante est peu probable d'être dû au hasard.
Si l'intervalle de confiance d'un paramètre **contient zéro**, cela signifie que ce paramètre n'est pas statistiquement significatif. On ne peut pas affirmer avec certitude qu'il a un impact sur la variable dépendante.

En regardant attentivement le tableau, nous pouvons identifier les variables suivantes dont l'intervalle de confiance contient zéro :

- **(Intercept)**: L'intervalle va de -0.033120665 à 0.03312066, ce qui inclut clairement zéro.
- **WeatherFoggy**: L'intervalle va de -0.077464321 à 0.01780941, incluant également zéro.
- **Time_of_DayAfternoon**: L'intervalle va de -0.028853838 à 0.04798968, contenant zéro.
- **Time_of_DayNight**: L'intervalle va de -0.035769557 à 0.03630434, incluant zéro.
- **Vehicle_TypeCar**: L'intervalle va de -0.033034131 à 0.03750997, contenant zéro.
- **Vehicle_TypeScooter**: L'intervalle va de -0.049495225 à 0.02135940, incluant zéro.
- **Order_ID**: L'intervalle va de -0.037897463 à 0.02910721, incluant zéro.

  Ces variables dont l'intervalle de confiance contient zéro, sont considérées comme **non statistiquement significatives**. Cela signifie que leur impact sur la variable dépendante (délai de livraison) n'est pas clairement établi par le modèle. Il est possible que ces variables n'aient pas d'effet réel sur le délai de livraison, ou que l'effet soit trop faible pour être détecté avec la taille de l'échantillon dont vous disposez.

```
                              2.5 %      97.5 %
(Intercept)            -0.077872157  0.18932670
WeatherClear           -0.567040362 -0.32168038
WeatherWindy           -0.491593628 -0.17268983
WeatherFoggy           -0.244356349  0.05681129
WeatherRainy           -0.360910130 -0.09303426
Traffic_LevelMedium     0.213609512  0.36231050
Traffic_LevelHigh       0.470446543  0.65729345
Time_of_DayAfternoon   -0.064560775  0.10563366
Time_of_DayEvening      0.006981762  0.17620347
Time_of_DayNight       -0.123457731  0.12306270
Vehicle_TypeCar        -0.083615557  0.09473063
Vehicle_TypeScooter    -0.106606896  0.04672983
Distance_km             0.736959476  0.80376145
Preparation_Time_min    0.286598686  0.35304542
Courier_Experience_yrs -0.114571991 -0.04773557

Call:
lm(formula = Delivery_Time_min ~ ., data = entrainement_reduit)
```

4.2- Significativité individuelle et globale du modèle

Les variables non significatives lors de la vérification avec l'intervalle de confiance sont les meme que l'on observe lorsqu'on test la significativité individuelle des paramètres. Le test de significativité individuelle (test t) permet de déterminer si un paramètre est significativement différent de zéro.

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.88800 -0.23053 -0.06444  0.14516  2.77964

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)              0.0557273  0.0680593   0.819  0.41315
WeatherClear            -0.4443604  0.0624966  -7.110 2.61e-12 ***
WeatherWindy            -0.3321417  0.0812292  -4.089 4.78e-05 ***
WeatherFoggy            -0.0937725  0.0767116  -1.222  0.22192
WeatherRainy            -0.2269722  0.0682317  -3.326  0.00092 ***
Traffic_LevelMedium      0.2879600  0.0378762   7.603 8.28e-14 ***
Traffic_LevelHigh        0.5638700  0.0475925  11.848  < 2e-16 ***
Time_of_DayAfternoon     0.0205364  0.0433509   0.474  0.63583
Time_of_DayEvening       0.0915926  0.0431031   2.125  0.03390 *
Time_of_DayNight        -0.0001975  0.0627922  -0.003  0.99749
Vehicle_TypeCar          0.0055575  0.0454273   0.122  0.90266
Vehicle_TypeScooter     -0.0299385  0.0390570  -0.767  0.44359
Distance_km              0.7703605  0.0170154  45.274  < 2e-16 ***
Preparation_Time_min     0.3198221  0.0169249  18.897  < 2e-16 ***
Courier_Experience_yrs -0.0811538  0.0170242  -4.767 2.23e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interprétation

```
p-value: < 2.2e-16
```

Un test F significatif (p-value généralement inférieure à 0.05) indique que le modèle global est statistiquement significatif. En d'autres termes, au moins une des variables indépendantes incluses dans le modèle a un impact significatif sur la variable dépendante (Délai de livraison). Les variables :

WeatherClear,

WeatherWindy,

WeatherRainy,

Traffic_Level_Medium,

Traffic_Level_High,

Time_of_Day_Evening,

Distance_km,

Preparation_Time_min

Courier_Experience_yrs sont statistiquement significatives (p-value < 0.05). Leur impact sur le délai de livraison est donc important.

Il est crucial de ne pas supprimer hâtivement les variables non significatives d'un modèle de régression linéaire, car une variable qui n'est pas significative individuellement peut jouer un rôle important dans la prédiction lorsqu'elle est combinée à d'autres variables. En effet, ces variables peuvent interagir entre elles, ce qui signifie que l'effet de l'une sur la variable cible dépend de la valeur de l'autre. Ainsi, une variable apparemment non significative peut en réalité apporter une information précieuse au modèle en modifiant l'effet d'autres variables. Négliger ces interactions potentielles conduirait à une perte d'information et à un modèle moins performant. Il est donc essentiel d'explorer les interactions possibles avant de décider de supprimer définitivement une variable non significative.

4.3- Modèle de référence

Un modèle de régression linéaire est construit avec toutes les variables potentiellement pertinentes (y compris les variables non significatives). Ce modèle sert de point de départ pour évaluer l'impact des interactions.

4.4- Test des interactions

Le code teste toutes les interactions possibles entre les variables non significatives et les autres variables du modèle. Pour chaque interaction potentielle :

- Une variable d'interaction est créée en multipliant les deux variables concernées.
- Un modèle de régression est ajusté avec cette variable d'interaction.
- Un test ANOVA est effectué pour comparer ce modèle avec le modèle de référence.
- Le R^2 ajusté du modèle avec interaction est calculé.

Ces tests permettent de déterminer si l'interaction est statistiquement significative et si elle améliore la qualité du modèle (mesurée par le R^2 ajusté).

4.5- Sélection de variables stepwise

Une fois les tests d'interaction effectués, une sélection de variables stepwise est appliquée au modèle de référence. Cette méthode permet de sélectionner le sous-ensemble de variables (y compris les interactions) qui contribue le plus à la prédiction.

4.6- Évaluation du modèle final

Le modèle final sélectionné est évalué à l'aide de métriques appropriées (R^2, R^2 ajusté, etc.) et en tenant compte de la validation croisée pour s'assurer qu'il généralise bien à de nouvelles données.

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.90935 -0.23778 -0.06118  0.14389  2.83984

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)          0.005296   0.043983   0.120 0.904186
WeatherClear        -0.394811   0.044646  -8.843  < 2e-16 ***
WeatherWindy        -0.279034   0.068696  -4.062 5.35e-05 ***
WeatherRainy        -0.175131   0.052665  -3.325 0.000924 ***
Traffic_LevelMedium  0.286365   0.037642   7.608 7.94e-14 ***
Traffic_LevelHigh    0.562857   0.047468  11.858  < 2e-16 ***
Time_of_DayEvening   0.084796   0.037252   2.276 0.023096 *
Distance_km          0.772722   0.016886  45.762  < 2e-16 ***
Preparation_Time_min 0.319576   0.016893  18.918  < 2e-16 ***
Courier_Experience_yrs -0.079238   0.016928  -4.681 3.36e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Interprétation des coefficients :

WeatherClear : Le coefficient pour WeatherClear est négatif et significatif (p-value < 2e-16). Cela suggère que lorsque le temps est clair, le temps de livraison a tendance à diminuer en moyenne de 0.39 unités de temps (minutes, probablement), toutes les autres variables étant égales.
WeatherWindy : Le coefficient pour WeatherWindy est également négatif et significatif (p-value = 5.35e-05). Cela indique que lorsque le temps est venteux, le temps de livraison a tendance à diminuer en moyenne de 0.28 unités de temps, toutes les autres variables étant égales. L'impact est moins important qu'avec un temps clair.
WeatherRainy : Le coefficient pour WeatherRainy est négatif et significatif (p-value = 0.000924). Cela suggère que lorsqu'il pleut, le temps de livraison a tendance à diminuer en moyenne de 0.18 unités de temps, toutes les autres variables étant égales. L'impact est moins important qu'avec un temps clair ou venteux.
Traffic_LevelMedium : Le coefficient pour Traffic_LevelMedium est positif et significatif (p-value < 2e-16). Cela suggère que lorsque le niveau de trafic est moyen, le temps de livraison a tendance à augmenter en moyenne de 0.29 unités de temps, toutes les autres variables étant égales.
Traffic_LevelHigh : Le coefficient pour Traffic_LevelHigh est également positif et significatif (p-value < 2e-16). Cela indique que lorsque le niveau de trafic est élevé, le temps de livraison a tendance à augmenter en moyenne de 0.56 unités de temps, toutes les autres variables étant égales. L'impact est plus important qu'avec un niveau de trafic moyen.
Time_of_DayEvening : Le coefficient pour Time_of_DayEvening est positif et significatif (p-value = 0.023096). Cela suggère que le soir, le temps de livraison a tendance à augmenter en moyenne de 0.08 unités de temps, toutes les autres variables étant égales.
Distance_km : Le coefficient pour Distance_km est positif et significatif (p-value < 2e-16). Cela suggère que chaque kilomètre supplémentaire augmente le temps de livraison en moyenne de 0.77 unités de temps, toutes les autres variables étant égales.
Preparation_Time_min : Le coefficient pour Preparation_Time_min est positif et significatif (p-value < 2e-16). Cela suggère que chaque minute de temps de préparation supplémentaire augmente le temps de livraison en moyenne de 0.32 unités de temps, toutes les autres variables étant égales.
Courier_Experience_yrs : Le coefficient pour Courier_Experience_yrs est négatif et significatif (p-value = 3.36e-06). Cela suggère que chaque année d'expérience supplémentaire du livreur diminue le temps de livraison en moyenne de 0.08 unités de temps, toutes les autres variables étant égales.

$$\mathrm{DT} = 0.005296 - 0.394811 * \mathrm{WC} - 0.279034 * \mathrm{WW} - 0.175131 * \mathrm{WR} + 0.286365 * TLM + 0.562857 * \mathrm{TLH} + 0.084796 * \mathrm{TDE} + 0.772722 * \mathrm{D} + 0.319576 * PT - 0.079238 * \mathrm{CE}$$

Avec :

DT : Delivery_Time_min

WC : WeatherClear

WW : WeatherWindy

TLM : Traffic_LevelMedium

THL : Traffic_LevelHigh

D : Distance_km

PT : Preparation_Time

CE : Courier_Experience_yrs

5- VERIFICATION DES HYPOTHESES DU MODELE

5.1- Test de linéarité du modèle

Le test de Rainbow est utilisé pour vérifier si la relation entre les variables indépendantes et la variable dépendante dans votre modèle de régression linéaire est bien linéaire. En d'autres termes, il teste si la forme fonctionnelle de votre modèle est correctement spécifiée.

```
        Rainbow test

data:  modele_final
Rain = 1.1781, df1 = 401, df2 = 390, p-value = 0.05188
```

Formulation de l'hypothèse de Rainbow

- Hypothèse nulle : le modèle est linéaire
- Hypothèse alternative : existence d'une non-linéarité

Interprétation

La p-value est légèrement supérieure au seuil de signification de 0.05. Par conséquent, nous ne rejetons pas l'hypothèse nulle. Selon le test de Rainbow, il n'y a pas de preuve statistique significative de non-linéarité alors l'hypothèse de linéarité est satisfaite.

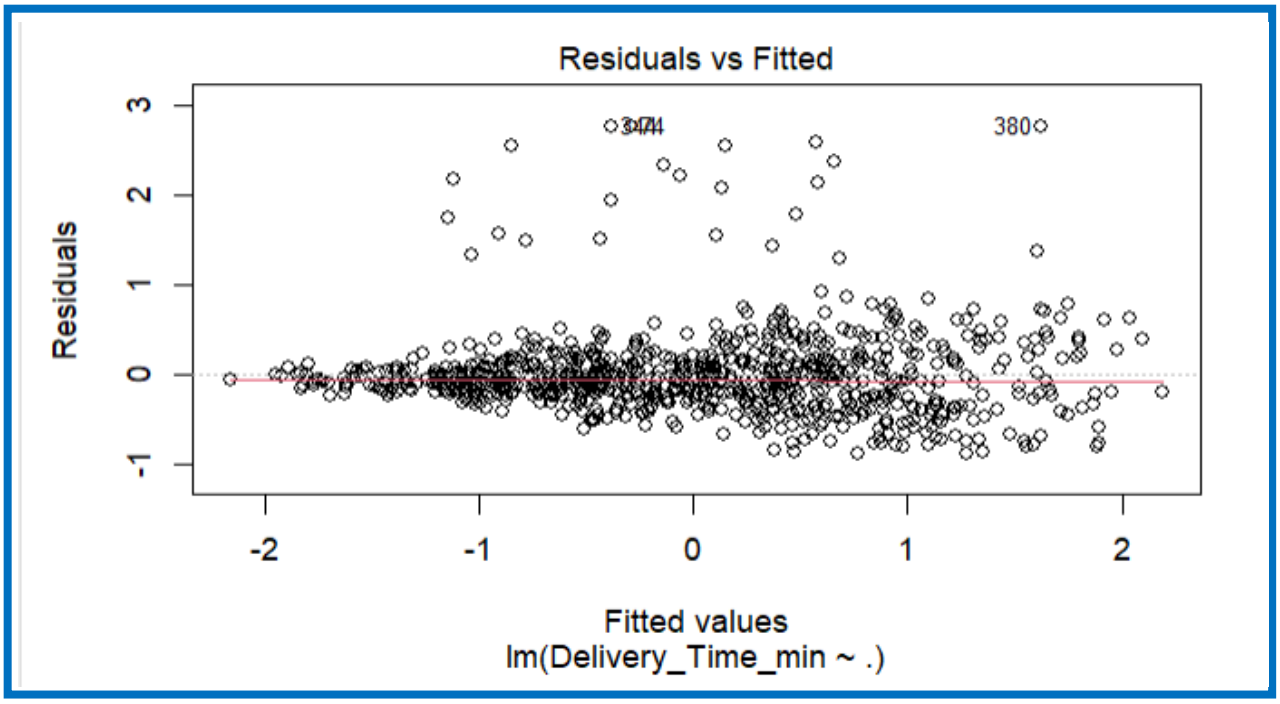
5.2- Test de corrélation entre les variables explicatives (X) et les erreurs (ε) : Cov(X, ε) = 0

L'objectif est de vérifier si la variance des résidus (l'étalement vertical des points) reste constante le long de l'axe des valeurs prédites.

Formulation de l'hypothèse

H_0 : Cov(X, ε) = 0 *il y a homoscédasticité* (Les variables explicatives ne sont pas corrélées avec les erreurs)

H_1 : Cov(X, ε) ≠ 0 *il y a hétéroscédasticité* (Les variables explicatives sont corrélées avec les erreurs)



Interprétation

Les points sont globalement dispersés de manière aléatoire autour de la ligne horizontale à 0. Cela suggère que la variance des erreurs est relativement constante sur l'ensemble des valeurs prédites du temps de livraison.
Il n'y a pas de tendance claire (en entonnoir, en arc de cercle, etc.) qui indiquerait une violation de l'homoscédasticité. L'hypothèse nulle est vérifiée.

5.3- Test d'homoscédasticité des erreurs $V(\varepsilon t) = E(\varepsilon t\ 2) = \sigma\varepsilon\ 2$

Le test d'homoscédasticité des erreurs vise à vérifier si la variance des erreurs d'un modèle de régression est constante pour toutes les observations. En d'autres termes, il teste si la dispersion des erreurs autour de la droite de régression est la même pour toutes les valeurs des variables indépendantes.

Formulation des hypothèses

H_0 : Homoscédasticité (variance constante des erreurs)
H_1 : Hétéroscédasticité (variance non constante des erreurs)

- Test de Breusch-Pagan

```
p-value = 0.1693
```

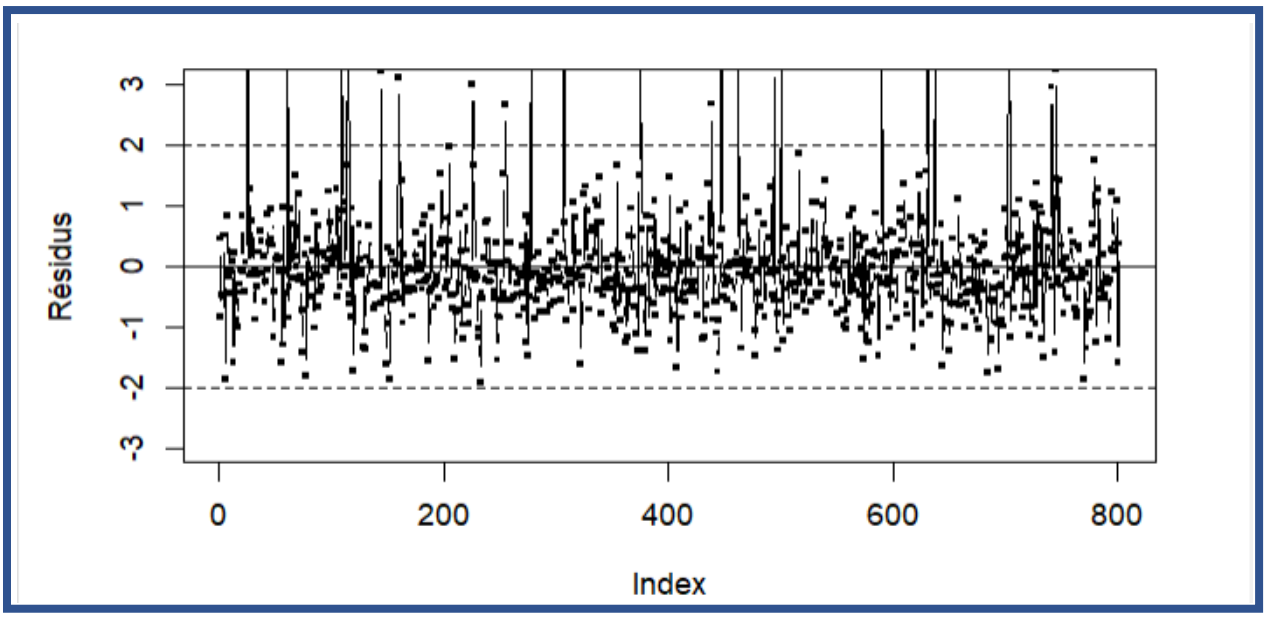
Puisque la p-value est supérieure à 0.05, nous ne rejetons pas l'hypothèse nulle. Cela signifie qu'il n'y a pas de preuve statistique significative d'hétéroscédasticité dans notre modèle. En d'autres termes, le test de Breusch-Pagan suggère que la variance des erreurs est constante.

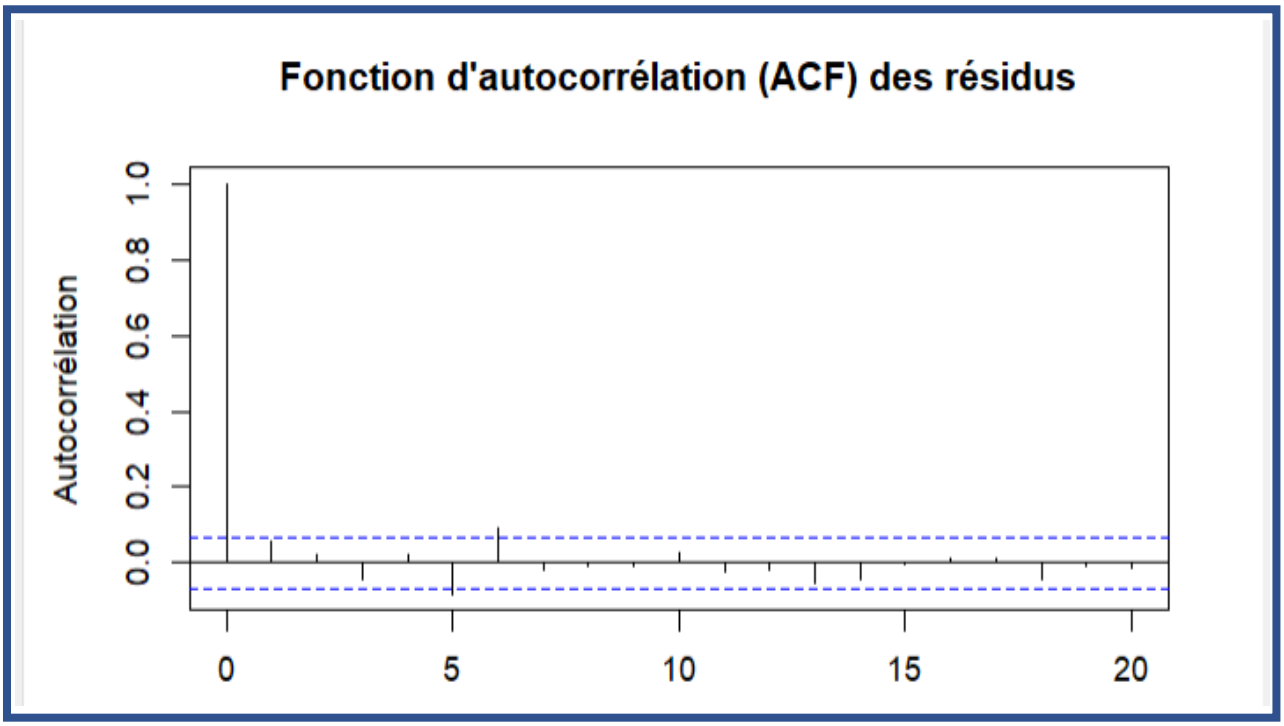
5.4- Test d'autocorrélation des erreurs $Cov(\varepsilon_t, \varepsilon_s) = 0$

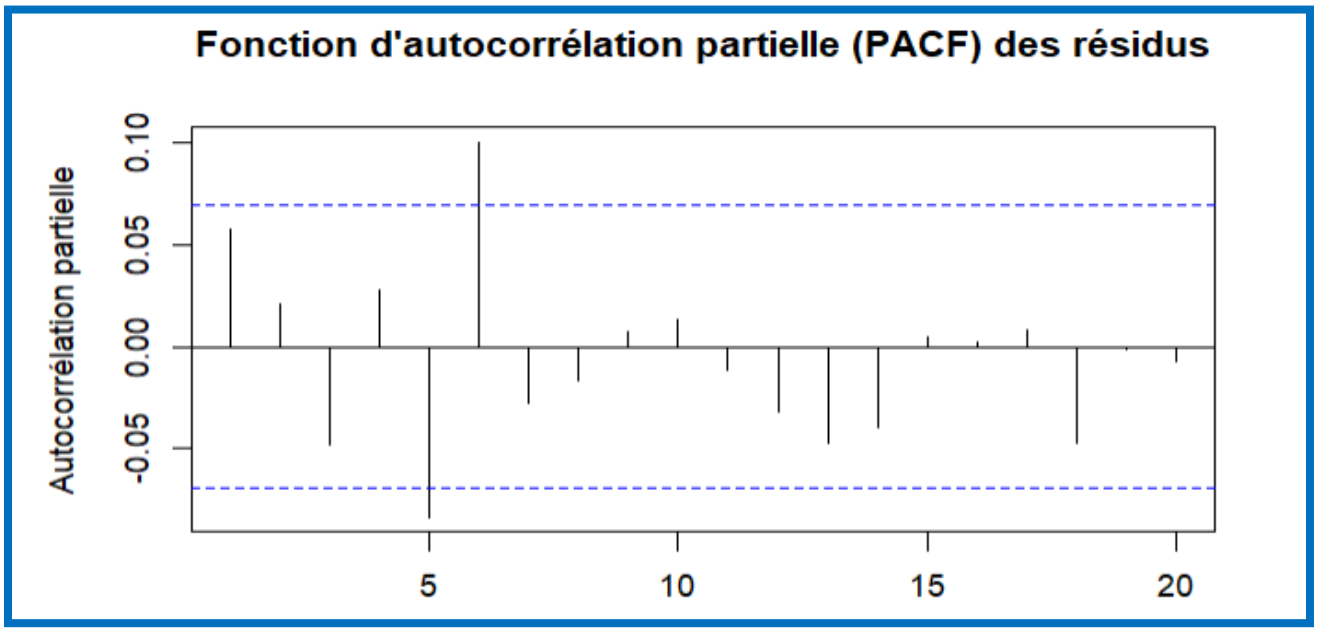
L'autocorrélation des erreurs, également connue sous le nom de corrélation sérielle, est un problème qui survient lorsque les erreurs (ou résidus) d'un modèle de régression linéaire ne sont pas indépendantes les unes des autres.

H_0 : $Cov(\varepsilon_t, \varepsilon_s) = 0$ pour tout $t \neq s$ (Les erreurs ne sont pas autocorrélées).

H_1 : $Cov(\varepsilon_t, \varepsilon_s) \neq 0$ pour au moins une paire $t \neq s$ (Les erreurs sont autocorrélées


Résidus
3
2
1
0
-1
-2
-3
0
200
400
600
800
Index


Fonction d'autocorrélation (ACF) des résidus
Autocorrélation
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20


Fonction d'autocorrélation partielle (PACF) des résidus
Autocorrélation partielle
0.10
0.05
0.00
-0.05
5
10
15
20

```
        Breusch-Godfrey test for serial correlation of order up to 1

data:  modele_final
LM test = 2.6808, df = 1, p-value = 0.1016
```

```
        Durbin-Watson test

data:  modele_final
DW = 1.8848, p-value = 0.05104
alternative hypothesis: true autocorrelation is greater than 0
```

```
        Box-Ljung test

data:  residuals(modele_final)
X-squared = 18.633, df = 10, p-value = 0.04518
```

```
        Runs Test

data:  residus_cat
Standard Normal = -0.25736, p-value = 0.7969
alternative hypothesis: two.sided
```

```
        studentized Breusch-Pagan test

data:  modele_final
BP = 12.855, df = 9, p-value = 0.1693
```

Interprétation

La plupart des tests et les graphiques convergent vers l'absence d'autocorrélation.
Le test de Ljung-Box, qui est le plus puissant, indique une possible autocorrélation,
La p-value est très proche du seuil de 0.05, ce qui suggère une autocorrélation faible qui n'aurait pas un impact majeur sur les résultats.

L'absence de pics prononcés dans les graphiques ACF et PACF suggère que même si une autocorrélation existe, elle est probablement faible et ne justifie pas l'utilisation d'un modèle plus complexe.

5.5- Test de Normalité des erreurs : $\varepsilon_t \sim N(0, \sigma^2)$

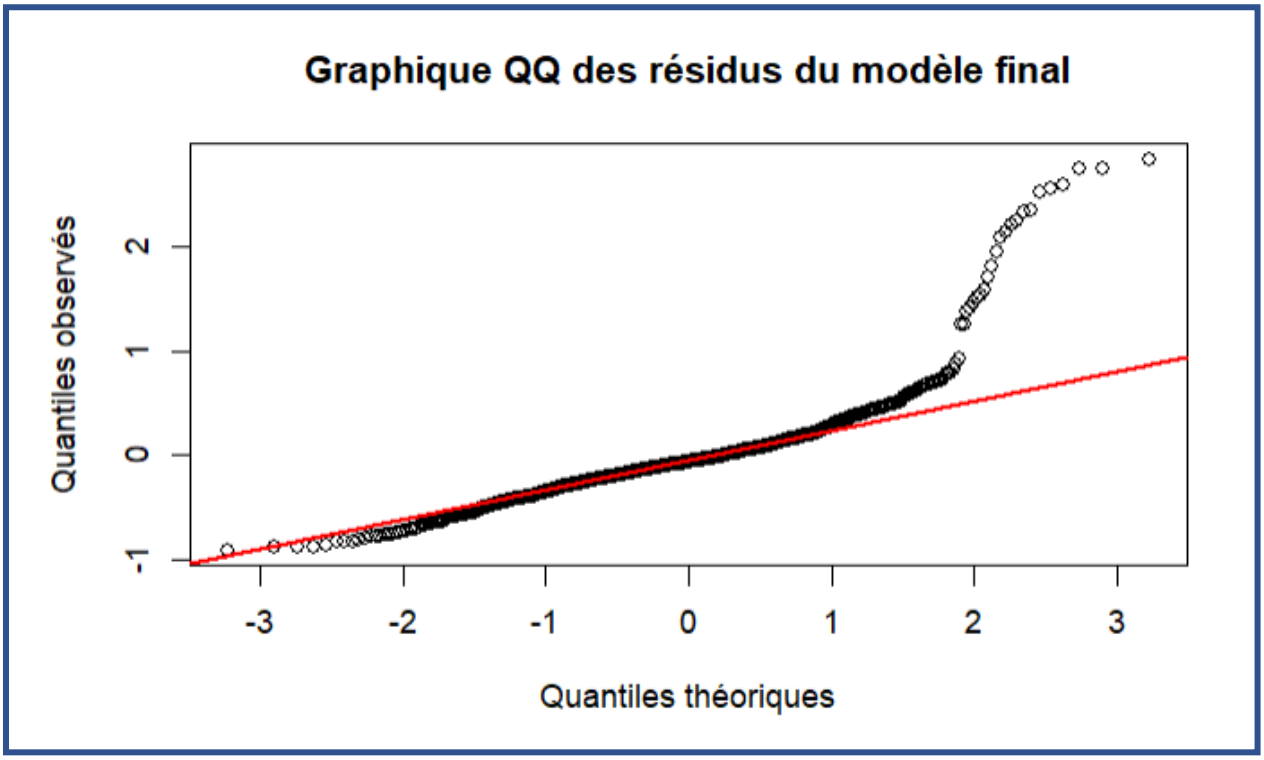
H_0 : Les erreurs suivent une distribution normale $N(0, \sigma^2)$
H_1: Les erreurs ne suivent pas une distribution normale

```
        Shapiro-Wilk normality test

data:  residus_modele_final
W = 0.79824, p-value < 2.2e-16


        Jarque Bera Test

data:  residus
X-squared = 4715.7, df = 2, p-value < 2.2e-16
```



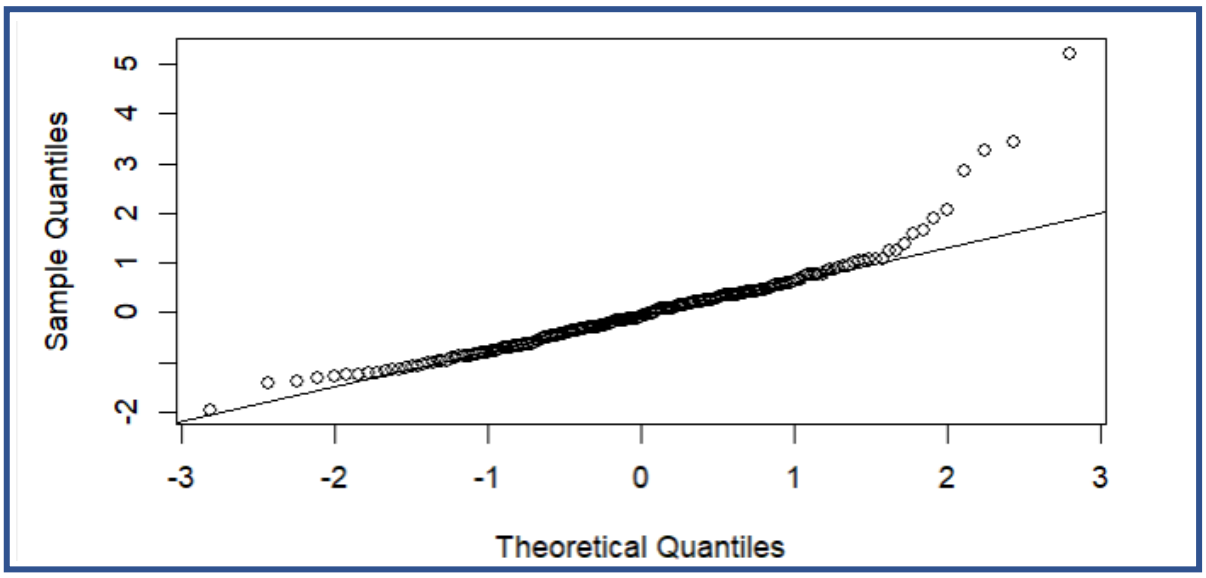
Interprétation

Le graphique QQ suggère que la distribution des résidus n'est pas parfaitement normale, ce qui est confirmé par le test de Shapiro-Wilk. Cependant, l'écart par rapport à la normalité ne semble pas être très important, car la plupart des points suivent la ligne de référence.

Transformation par box cox sur les résidus du modele_final

Trois methodes ont été utilisé pour effecter la transformation, il s'agit de la tranformation, logarithmique, la transformation par box cox et par transformation par racine carree. La meilleure transformation reste la transformation pas box cox.

```
"Test de Shapiro-Wilk : p-value = 6.34140843607756e-11"
```



5.5-Test de la moyenne des erreurs : $\boldsymbol{E(\varepsilon t)} = 0$

L'objectif principal de ce test est de s'assurer que les erreurs (ou résidus) de votre modèle de régression linéaire ont une moyenne de zéro.

H_0 : La moyenne des erreurs est égale à zéro.

H_1 : La moyenne des erreurs n'est pas égale à zéro.

Interprétation

```
    mean of x
3.742714e-18
```

La moyenne des résidus suggèrent que notre modèle est bien spécifié et que les erreurs ont une moyenne proche de zéro. Le modèle semble satisfaire l'hypothèse de moyenne des erreurs nulle.

6- EVALUATION DE LA QUALITE GLOBALE DU MODELE

6.1- Test de Chow

Le test de Chow est un test statistique qui permet de déterminer s'il y a un changement structurel dans un modèle de régression linéaire.

$\boldsymbol{H0}$: il n'y a pas de rupture de structure

$\boldsymbol{H1}$: il a rupture de structure

```
p-value = 0.3267
```

Interprétation

Cela signifie qu'il n'y a pas de preuve statistique significative, au niveau de 5%, d'un changement structurel dans la relation entre votre variable dépendante (**Delivery_Time_min**) et vos variables indépendantes**.** En d'autres termes, le test suggère que la relation entre ces variables est stable sur la période considérée

6.2 - Indicateurs de performance du modèle avant prédictions

```
"MSE : 0.761865885336363"
"AIC : 520.612572501439"
"MAE : 0.621898888082248"
"RMSE : 0.872849291307705"
Multiple R-squared:  0.7755
```

Les valeurs très faibles de MSE, MAE et RMSE suggèrent que le modèle s'ajuste extrêmement bien aux données d'entraînement. Notre modèle explique environ 77,55% de la variance totale du temps de livraison. Ce qui suggère que notre modèle a une capacité explicative raisonnable.

7- PREVISIONS

Prévision sur l'ensemble test et évaluation de la prévison. Visualisation des 5 premières lignes de la prévisions de l'ensemble test.

```
       1        6        9       14       42
45.87371 75.82116 32.98126 33.61515 62.32521
[1] 43 57 35 36 67
```

```
"MSE : 131.697913430745"
"RMSE : 11.4759711323594"
"MAE : 7.5316479935675 5"
"R-carré : 0.744506841501549"
```

Interprétation globale

Les métriques d'erreur (MSE, RMSE, MAE) sont plus élevées que sur les données d'entraînement, Cela indique que le modèle est moins performant sur des données qu'il n'a pas vues pendant l'entraînement. Le R-carré de 0.744506841501549 suggère que le modèle explique une part importante de la variance du temps de livraison sur l'ensemble de test. Les métriques d'erreur indiquent un écart moyen entre les prédictions et les valeurs réelles de l'ordre de 7 à 11 minutes sur l'ensemble de test.Le R-carré suggère que le modèle explique une part importante de la variance du temps de livraison, mais pas la totalité.

PARTIE III : MODELISATION EN ANOVA

Bien que la Régression Linéaire Multiple (RLM) et l'Analyse de la Variance (ANOVA) partagent le même cadre de modélisation linéaire, elles se distinguent par leur objectif principal et le niveau de détail de l'analyse. La RLM offre une vision globale en modélisant les relations entre une variable dépendante continue et divers types de variables indépendantes, qu'elles soient continues ou catégorielles, évaluant ainsi l'impact général de ces dernières. En revanche, l'ANOVA, en tant que composante intégrée de la RLM, se concentre spécifiquement sur l'évaluation des variables catégorielles, allant au-delà de l'effet global pour examiner les différences précises entre les moyennes de la variable dépendante pour chaque sous-groupe défini par ces variables. Elle permet ainsi une analyse plus fine des variables catégorielles, fournissant des informations détaillées sur les variations entre leurs niveaux, ce que la RLM ne fait pas de manière directe.

1-ANALYSE DE LA VARIANCE

```
Analysis of Variance Table

Response: Delivery_Time_min
                       Df Sum Sq Mean Sq   F value    Pr(>F)
WeatherClear            1  21.29   21.29   93.7623 < 2.2e-16 ***
WeatherWindy            1   3.30    3.30   14.5318 0.0001485 ***
WeatherRainy            1   2.95    2.95   13.0070 0.0003298 ***
Traffic_LevelMedium     1   1.68    1.68    7.3955 0.0066822 **
Traffic_LevelHigh       1  30.55   30.55  134.5193 < 2.2e-16 ***
Time_of_DayEvening      1   0.30    0.30    1.3150 0.2518350
Distance_km             1 472.86  472.86 2082.4414 < 2.2e-16 ***
Preparation_Time_min    1  82.49   82.49  363.2883 < 2.2e-16 ***
Courier_Experience_yrs  1   4.98    4.98   21.9116 3.359e-06 ***
Residuals             791 179.61    0.23
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Interprétations

VARIABLES	PR(>F)	INTERPRETATIONS
WeatherClear	Pr(>F) < 2.2e-16 (très faible)	Les jours où le ciel est clair ont un impact statistiquement significatif sur le temps de livraison
WeatherWindy	Pr(>F) = 0.0001485	Les jours venteux ont un impact statistiquement significatif sur le temps de livraison.
WeatherRainy	Pr(>F) = 0.0003298	Les jours de pluie ont un impact statistiquement significatif sur le temps de livraison.
Traffic_LevelMedium	Pr(>F) = 0.0066822	Les niveaux de trafic moyens ont un impact statistiquement significatif sur le temps de livraison.
Traffic_LevelHigh	Pr(>F) < 2.2e-16 (très faible)	Les niveaux de trafic élevés ont un impact statistiquement significatif sur le temps de livraison.
Time_of_DayEvening	Pr(>F) < 2.2e-16 (très faible)	Le moment de la journée (soir) n'a pas un impact statistiquement significatif sur le temps de livraison
Distance_km	Pr(>F) < 2.2e-16 (très faible)	La distance a un impact statistiquement significatif sur le temps de livraison.
Preparation_Time_min	Pr(>F) < 2.2e-16 (très faible)	Le temps de préparation a un impact statistiquement significatif sur le temps de livraison.
Courier_Experience_yr	Pr(>F) = 3.359e-06	L'expérience du coursier a un impact statistiquement significatif sur le temps de livraison.

2- TEST D'EGALITE DES MOYENNES ET TEST DE SIGNIFICATIVITE DES MOYENNES

- Test d'égalité de la moyenne

Variable	P-value
WeatherClear	0.2348
WeatherWindy	0.3711
WeatherRainy	0.5946
Traffic_LevelMedium	0.589
Traffic_LevelHigh	0.5979
Time_of_DayEvening	0.9441

Le test de Levene vérifie si les variances de la variable dépendante (Delivery_Time_min) sont égales entre les groupes définis par chaque variable qualitative. Pour toutes les variables qualitatives (WeatherClear, WeatherWindy, WeatherRainy, Traffic_LevelMedium, Traffic_LevelHigh, Time_of_DayEvening), les valeurs de Pr(>F) (p-values) sont toutes supérieures à 0.05.
Cela signifie que nous ne pouvons pas rejeter l'hypothèse nulle d'égalité des variances.
En d'autres termes, les variances de Delivery_Time_min sont considérées comme égales entre les groupes pour chaque variable qualitative.
En Conclusion, l'hypothèse d'homogénéité des variances est respectée pour toutes les variables qualitatives testées.

- Test Tukey

Variable	P-value ajustée (p adj)
WeatherClear	2.1e-06
WeatherWindy	0.0741323
WeatherRainy	0.1622999
Traffic_LevelMedium	0.1810267
Traffic_LevelHigh	3e-07
Time_of_DayEvening	0.5728457

Le test post hoc de Tukey est utilisé pour effectuer des comparaisons par paires entre les moyennes des groupes lorsque l'ANOVA a révélé des différences significatives. Les variances de Delivery_Time_min sont homogènes entre les groupes pour toutes les variables qualitatives. Seules les variables WeatherClear et Traffic_LevelHigh ont un impact significatif sur les délais de livraison.

CONCLUSION GENERALE

L'analyse de la variance (ANOVA) s'est avérée être un outil précieux pour compléter les capacités de la régression linéaire multiple (RLM) dans notre étude. Alors que le RLM permet d'évaluer l'impact global des variables catégorielles, il ne peut déterminer la significativité des sous-groupes au sein de ces variables. L'ANOVA a comblé cette lacune en confirmant que les variables catégorielles WeatherClear, WeatherWindy, WeatherRainy, Traffic_LevelMedium et Traffic_LevelHigh ont des sous-groupes qui influencent significativement Delivery_Time_min, validant ainsi leur pertinence dans le modèle RLM. Il est important de souligner que l'ANOVA n'est pas une méthode de régression distincte, mais plutôt une composante intégrante de l'analyse de régression. Elle fournit des informations complémentaires essentielles pour une interprétation plus approfondie du modèle RLM. Cependant, l'ANOVA présente une limite : bien qu'elle détermine la significativité des sous-groupes, elle ne quantifie pas directement la taille de ces effets. Pour cela, des mesures de la taille de l'effet et des coefficients standardisés du RLM sont nécessaires. En combinant l'ANOVA et le RLM, nous avons obtenu une analyse plus complète et robuste, permettant une meilleure compréhension des facteurs influençant le temps de livraison.

CODE SOURCE

A. Prétraitement des données

#importation des 5 premières lignes du jeu de données.

```
delai_livraison <- read.csv("C:/Users/hp/Downloads/COURS INSSEDS/PROJETS
INSSEDS/jeu données projets/delai_livraison.csv", stringsAsFactors=TRUE)
head(delai_livraison)

#Vidualisation des valeurs  manquantes
library(visdat)
vis_dat(delai_livraison)
colSums(is.na(delai_livraison))
str(delai_livraison)
# Identifier les variables avec des niveaux vides
empty_levels <- sapply(delai_livraison, function(x) is.factor(x) && any(levels(x)
== ""))
names(empty_levels[empty_levels])

#Visualisation de toutes les valeurs manquantes
# Convertir les niveaux vides en NA pour les colonnes factorielles
library(dplyr)
delai_livraison <- delai_livraison %>%
  mutate(across(where(is.factor), ~ na_if(as.character(.), "")))
str(delai_livraison)
# Vérifier les niveaux vides dans les colonnes factorielles
empty_levels <- sapply(delai_livraison, function(x) is.factor(x) && any(levels(x)
== ""))
empty_levels <- empty_levels[empty_levels]
names(empty_levels)
colSums(is.na(delai_livraison))
vis_dat(delai_livraison)

#Traitement des valeurs manquantes
# Calculer le pourcentage total de valeurs manquantes (NA) dans le dataframe
pourcentage_na <- sum(is.na(delai_livraison)) / (nrow(delai_livraison) *
ncol(delai_livraison)) * 100
cat("Le pourcentage total de valeurs manquantes (NA) dans le jeu de données est
:", pourcentage_na, "%")
# Fonction pour imputer les valeurs manquantes par le mode
impute_mode <- function(x) {
mode_value <- names(sort(table(x), decreasing = TRUE))[1]
x[is.na(x)] <- mode_value
return(x)
      }
# Imputer les valeurs manquantes des niveaux de facteurs par le mode
delai_livraison$Weather <- impute_mode(delai_livraison$Weather)
delai_livraison$Traffic_Level <- impute_mode(delai_livraison$Traffic_Level)
delai_livraison$Time_of_Day <- impute_mode(delai_livraison$Time_of_Day)
colSums(is.na(delai_livraison))


# Fonction pour imputer les variables numeriques par KNNimputation
# Charger les packages nécessaires
library(dplyr)
library(DMwR2)
# Isoler la variable à imputer et les variables explicatives
data_for_imputation <- delai_livraison %>%
```

```
  dplyr::select(Courier_Experience_yrs, Distance_km, Preparation_Time_min)
# Effectuer l'imputation par kNN
data_imputed <- knnImputation(data_for_imputation, k = 10, meth = "median")
# Remplacer les valeurs imputées dans le dataframe original
delai_livraison$Courier_Experience_yrs <- data_imputed$Courier_Experience_yrs
# Afficher le résultat
colSums(is.na(delai_livraison))
vis_dat(delai_livraison)

#Traitement de doublons
delai_livraison = traitement_doublons(delai_livraison)

#Traitement des valeurs extremes et aberrantes
#Visualisation avant traitement des boites à moustaches
afficher_boites_a_moustache(delai_livraison)
#Division des données en ensembles tests et entrainement
# Charger la librairie caret
library(caret)
# Définir la proportion de l'ensemble d'entraînement
p <- 0.8
# Fixer une graine pour la reproductibilité
set.seed(123)
# Créer un échantillon aléatoire de 80% des données
index_entrainement <- createDataPartition(delai_livraison$ Delivery_Time_min, p =
p, list = FALSE)
# Créer les ensembles d'entraînement et de test
entrainement <- delai_livraison[index_entrainement, ]
test <- delai_livraison[-index_entrainement, ]
# Afficher la taille des ensembles d'entraînement et de test
cat("Taille de l'ensemble d'entraînement :", nrow(entrainement), "\n")
cat("Taille de l'ensemble de test :", nrow(test), "\n")
#Traitement des valeurs aberrantes et extremes
# Charger les packages nécessaires
library(dplyr)
library(DescTools)
# Fonction pour afficher les boîtes à moustache
afficher_boites_a_moustache <- function(dataframe) {
  # Sélectionner uniquement les colonnes numériques
  colonnes_numeriques <- sapply(dataframe, is.numeric)

  if (sum(colonnes_numeriques) > 0) {
    # Créer un graphique de boîtes à moustache pour chaque colonne numérique
    boxplot(dataframe[,   colonnes_numeriques],   main="Boîtes   à   moustache",
col="lightblue", border="black")
  } else {
    cat("Aucune colonne numérique à afficher.\n")
  }
}
# Fonction pour traiter les valeurs extrêmes et manquantes
traitement_donnees_extremes <- function(entrainement) {
# Liste des noms de colonnes quantitatives
  colonnes_quantitatives <- sapply(entrainement, is.numeric)
# Winsorisation des variables quantitatives avec des seuils spécifiques
  for (colonne in names(entrainement)[colonnes_quantitatives]) {
    if (colonne %in% c("WeatherWindy", "Traffic_LevelHigh")) {
      # Limites pour WeatherWindy et Traffic_LevelHigh (10% de Winsorisation)
      lower_limit <- quantile(entrainement[[colonne]], 0.05, na.rm = TRUE)
      upper_limit <- quantile(entrainement[[colonne]], 0.95, na.rm = TRUE)
```

```
      entrainement[[colonne]] <- pmax(pmin(entrainement[[colonne]], upper_limit),
lower_limit)
    } else if (colonne %in% c("Delivery_Time_min")) {
      # Limites pour Delivery_Time_min (5% de Winsorisation)
      lower_limit <- quantile(entrainement[[colonne]], 0.025, na.rm = TRUE)
      upper_limit <- quantile(entrainement[[colonne]], 0.975, na.rm = TRUE)
      entrainement[[colonne]] <- pmax(pmin(entrainement[[colonne]], upper_limit),
lower_limit)
    } else {
      # Limites par défaut (10% de Winsorisation) pour les autres variables
      lower_limit <- quantile(entrainement[[colonne]], 0.05, na.rm = TRUE)
      upper_limit <- quantile(entrainement[[colonne]], 0.95, na.rm = TRUE)
      entrainement[[colonne]] <- pmax(pmin(entrainement[[colonne]], upper_limit),
lower_limit)
    }
  }
  return(entrainement)
}
# Exemples d'utilisation
data_traitée <- traitement_donnees_extremes(entrainement)
afficher_boites_a_moustache(data_traitée)
#Encodage des variables catégorielles et standardisation des variables numériques.
# Charger les packages nécessaires
library(dplyr)
library(caret)
library(usdm)
# Recodage des variables catégorielles pour l'ensemble d'entraînement
entrainement$Weather <- factor(entrainement$Weather, levels = c("Clear", "Windy",
"Foggy", "Rainy", "Snowy"))
entrainement$Traffic_Level <- factor(entrainement$Traffic_Level, levels = c("Low",
"Medium", "High"))
entrainement$Time_of_Day <- factor(entrainement$Time_of_Day, levels = c("Morning",
"Afternoon", "Evening", "Night"))
entrainement$Vehicle_Type <- factor(entrainement$Vehicle_Type)
# Encodage One-Hot des variables catégorielles pour l'ensemble d'entraînement
entrainement_encoded <- model.matrix(~ Weather + Traffic_Level + Time_of_Day +
Vehicle_Type - 1, data = entrainement)
entrainement_encoded <- as.data.frame(entrainement_encoded)
# Assurer que les lignes correspondent après encodage One-Hot
if (nrow(entrainement_encoded) != nrow(entrainement)) {
  stop("Le nombre de lignes dans 'entrainement_encoded' et 'entrainement' ne
correspond pas.")
}
# Normalisation/Standardisation des variables numériques pour l'ensemble
d'entraînement
variables_numeriques <- entrainement %>% select_if(is.numeric)
preProc_train <- preProcess(variables_numeriques, method = c("center", "scale"))
variables_numeriques_scaled <- predict(preProc_train, variables_numeriques)
# Assurer que les lignes correspondent après normalisation
if (nrow(variables_numeriques_scaled) != nrow(entrainement)) {
  stop("Le nombre de lignes dans 'variables_numeriques_scaled' et 'entrainement' ne
correspond pas.")
}
# Combiner les variables encodées et standardisées pour l'ensemble d'entraînement
entrainement_scaled <- cbind(entrainement_encoded, variables_numeriques_scaled)
print(entrainement_scaled)
# Recodage des variables catégorielles pour l'ensemble de test
test$Weather <- factor(test$Weather, levels = c("Clear", "Windy", "Foggy", "Rainy",
"Snowy"))
```

```
test$Traffic_Level <- factor(test$Traffic_Level, levels = c("Low", "Medium", "High"))
test$Time_of_Day <- factor(test$Time_of_Day, levels = c("Morning", "Afternoon", "Evening", "Night"))
test$Vehicle_Type <- factor(test$Vehicle_Type)
# Encodage One-Hot des variables catégorielles pour l'ensemble de test
test_encoded <- model.matrix(~ Weather + Traffic_Level + Time_of_Day + Vehicle_Type - 1, data = test)
test_encoded <- as.data.frame(test_encoded)
# Assurer que les lignes correspondent après encodage One-Hot pour l'ensemble de test
if (nrow(test_encoded) != nrow(test)) {
  stop("Le nombre de lignes dans 'test_encoded' et 'test' ne correspond pas.")
}
# Normalisation/Standardisation des variables numériques pour l'ensemble de test
variables_numeriques_test <- test %>% select_if(is.numeric)
test_scaled <- predict(preProc_train, variables_numeriques_test)

# Assurer que les lignes correspondent après normalisation pour l'ensemble de test
if (nrow(test_scaled) != nrow(test)) {
  stop("Le nombre de lignes dans 'test_scaled' et 'test' ne correspond pas.")
}
# Combiner les variables encodées et standardisées pour l'ensemble de test
test_final <- cbind(test_encoded, test_scaled)
print(test_final)

#Determination du VIF pour détecter la multicolinéarité des variables numériques.
# Charger les packages nécessaires
library(usdm)

# Sélectionner les variables d'intérêt
variables_interet <- entrainement_scaled[, c("Distance_km", "Preparation_Time_min", "Courier_Experience_yrs")]
# Calculer le VIF pour les variables sélectionnées
vif_values <- vifcor(variables_interet)
# Afficher les valeurs VIF
print(vif_values)

#Matrice de corrélation de toutes les variables
# Charger les packages nécessaires
library(ggplot2)
library(reshape2)
# Calculer la matrice de corrélation avec toutes les variables de entrainement_scaled
cor_matrix <- cor(entrainement_scaled)
print("Matrice de corrélation :")
print(cor_matrix)
# Visualiser la matrice de corrélation avec un heatmap
cor_matrix_melt <- melt(cor_matrix)
ggplot(data = cor_matrix_melt, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "#D9EAFD", high = "#08306B", mid = "#56B1F7", midpoint = 0, limit = c(-1, 1), space = "Lab", name="Corrélation") +
  theme_minimal() +
  labs(title = "Heatmap de la matrice de corrélation", x = "Variables", y = "Variables") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

#Suppression des variables qui n’apportent aucunes informations
# Charger les packages nécessaires
```

```
library(dplyr)
# Supprimer les variables Order_ID et WeatherSnowy du dataframe
entrainement_scaled
entrainement_reduit <- entrainement_scaled %>%
  dplyr::select(-Order_ID, -WeatherSnowy)
# Afficher les premières lignes du dataframe résultant pour vérifier les
modifications
head(entrainement_reduit)

#Estimation en RLM
reg.fin <- lm(Delivery_Time_min ~ ., data = entrainement_reduit)
# Voir l'intervalle de confiance des coefficients estimés
confint(reg.fin)
# Voir le résumé des résultats du modèle final
summary(reg.fin)

#Selection des variables significatives et vérification des variables non
significatives qui pourrait améliorées la prédiction
# Fonction pour ajuster et évaluer le modèle
ajuster_modele <- function(data) {
  reg <- lm(Delivery_Time_min ~ ., data = entrainement_reduit)
  summary_reg <- summary(reg)
  p_value_test_F <- pf(summary_reg$fstatistic[1], summary_reg$fstatistic[2],
summary_reg$fstatistic[3], lower.tail = FALSE)
  r2_ajuste <- summary_reg$adj.r.squared
  list(model = reg, p_value_test_F = p_value_test_F, r2_ajuste = r2_ajuste)
}
# Vérifier que entrainement_reduit est un dataframe
entrainement_reduit <- as.data.frame(entrainement_reduit)
# Variables non significatives identifiées
variables_non_significatives <- c("WeatherFoggy", "Time_of_DayAfternoon",
"Time_of_DayNight",
                                  "Vehicle_TypeCar", "Vehicle_TypeScooter")
# Variables restantes dans le meilleur modèle
variables_restantes <- c("WeatherClear", "WeatherWindy", "WeatherRainy",
                         "Traffic_LevelMedium", "Traffic_LevelHigh",
                         "Time_of_DayEvening", "Distance_km",
                         "Preparation_Time_min", "Courier_Experience_yrs")
# Construire le modèle de référence avec toutes les variables
modele_reference <- ajuster_modele(entrainement_reduit)
# Fonction pour tester les interactions
tester_interactions <- function(non_sig_vars, autres_vars, data) {
  resultats <- list()
  for (non_sig in non_sig_vars) {
    if (non_sig %in% names(data)) { # Vérifier si la variable existe
      for (autre in autres_vars) {
        if (autre %in% names(data)) { # Vérifier si la variable existe
          # Créer la variable d'interaction
          nom_interaction <- paste0(non_sig, "_", autre)
          data[[nom_interaction]] <- data[[non_sig]] * data[[autre]]

          # Ajuster le modèle avec l'interaction
          data_interaction <- data # Copie pour ne pas modifier l'original
          result_interaction <- ajuster_modele(data_interaction)

          # Comparer le modèle avec interaction et le modèle original
          comparaison <- anova(modele_reference$model, result_interaction$model)
          cat("Test d'interaction :", nom_interaction, "\n")
          print(comparaison)
```

```
          # Analyser les résultats (p-value, R² ajusté)
          cat("R² ajusté du modèle avec interaction :",
result_interaction$r2_ajuste, "\n\n")

          # Enregistrer les résultats
          resultats[[nom_interaction]] <- list(
            comparaison = comparaison,
            r2_ajuste = result_interaction$r2_ajuste,
            p_value_test_F = result_interaction$p_value_test_F
          )

          # Supprimer la variable d'interaction pour le prochain test
          data[[nom_interaction]] <- NULL
        }
      }
    }
  }
  return(resultats)
}

# Tester les interactions avec toutes les autres variables
resultats_interactions <- tester_interactions(variables_non_significatives,
variables_restantes, entrainement_reduit)

# Sélection de variables avec méthode stepwise
modele_final <- step(modele_reference$model, direction = "both", trace = FALSE)
summary(modele_final)

# Évaluation du modèle final
print(summary(modele_final))

#Test d’hypothèse
#Test de linéarité
library(lmtest)
raintest(modele_final)
#Test de corrélation entre variables indépendantes et les erreurs
plot(modele_final, 1)
#Test d’autocorrélation des erreurs
#Test Breusch-Pagan
library(lmtest)
bptest(modele_final)
#Visualisation des residus
res.m <- rstudent(modele_final)
plot(res.m, pch=15, cex=.5, ylab="Résidus", ylim=c(-3,3), type="b")
abline(h=c(-2,0,2), lty=c(2,1,2))
#Test Durbin-Watson
library(lmtest)
dwtest(modele_final
# Box-Ljung test
# Effectuer le test de Ljung-Box sur les résidus du modèle reg.fin
# avec un nombre de décalages (lags) de 10 (vous pouvez ajuster ce nombre)
box_result <- Box.test(residuals(modele_final), lag = 10, type = "Ljung-Box")
# Afficher les résultats du test
print(box_result)
# Runs Test
library(tseries)
# Transformer les résidus en variable catégorielle (signe des résidus)
residus_cat <- factor(ifelse(residuals(modele_final) > 0, "Positif", "Négatif"))
```

```
# Effectuer le test de Runs
runs.test(residus_cat)
#ACF
# Calculer les résidus du modèle reg.fin
residus <- residuals(modele_final)
# Tracer la fonction d'autocorrélation (ACF) des résidus
acf(residus,
    lag.max = 20, # Nombre maximum de décalages à afficher (ajuster si nécessaire)
    main = "Fonction d'autocorrélation (ACF) des résidus",
    xlab = "Décalage (lag)",
    ylab = "Autocorrélation")
« Test de moyenne des erreurs
# Calculer les résidus du modèle final
residus_modele_final <- residuals(modele_final)
# Calculer la moyenne des résidus
moyenne_residus <- mean(residus_modele_final)
# Afficher la moyenne des résidus
print(paste("Moyenne des résidus :", moyenne_residus))
# Effectuer un test t pour vérifier si la moyenne est significativement différente
de zéro
t_test_result <- t.test(residus_modele_final, mu = 0)
# Afficher les résultats du test t
print(t_test_result)
#Test de chow
# Installer le package strucchange si nécessaire
#install.packages("strucchange")
# Charger le package strucchange
library(strucchange)
# Effectuer le test de Chow sur le modèle final
sctest(Delivery_Time_min ~ WeatherClear + WeatherWindy + WeatherRainy +
           Traffic_LevelMedium + Traffic_LevelHigh + Time_of_DayEvening +
           Distance_km + Preparation_Time_min + Courier_Experience_yrs,
       data = entrainement_reduit)
#Test de normalité des erreurs
# Extraire les résidus du modèle final
residus_modele_final <- residuals(modele_final)
# Graphique QQ des résidus
qqnorm(residus_modele_final,
       main = "Graphique QQ des résidus du modèle final",
       xlab = "Quantiles théoriques",
       ylab = "Quantiles observés")
# Ajouter une ligne de référence pour la normalité parfaite
qqline(residus_modele_final, col = "red", lwd = 2)
#Transformation box plot
# Charger les librairies nécessaires
library(MASS) # Pour la transformation de Box-Cox

# 1. Transformation Box-Cox
# Vérifier si les données sont toutes positives (Box-Cox ne fonctionne pas avec
des valeurs négatives ou nulles)
if (all(test$Delivery_Time_min > 0)) {
  boxcox_result <- boxcox(test$Delivery_Time_min ~ 1, plotit = TRUE) # ~ 1 indique
une transformation sans variable indépendante
  lambda <- boxcox_result$x[which.max(boxcox_result$y)] # Obtenir le lambda
optimal
  transformed_boxcox <- (test$Delivery_Time_min^lambda - 1) / lambda # Appliquer
la transformation
  # Créer le modèle linéaire final avec la variable cible transformée par Box-Cox
```

```
  modele_final <- lm(transformed_boxcox ~ Distance_km + Preparation_Time_min +
Courier_Experience_yrs, data = test_final)

  # Afficher le modèle final
  print(summary(modele_final))
  # QQ-plot pour visualiser la normalité des résidus
  residus <- residuals(modele_final)
  qqnorm(residus)
  qqline(residus) # Ajoute la ligne de référence pour la normalité
  # Test de Shapiro-Wilk sur les résidus
  shapiro_test <- shapiro.test(residus)
  print(paste("Test de Shapiro-Wilk : p-value =", shapiro_test$p.value)
} else {
  print("Les données contiennent des valeurs non positives. La transformation de
Box-Cox ne peut pas être appliquée.")
}

« Indicateur de la qualité globale du modèle
#Calcul du MSE
  mse <- mean(residuals(modele_final)^2)
# Calcul de l'AIC
  aic <- AIC(modele_final)
# Calcul du MAE
mae <- mean(abs(residus))
#cal
rmse <- sqrt((mse)^2)
# Affichage des résultats
  print(paste("MSE :", mse))
  print(paste("AIC :", aic))
  print(paste("MAE :", mae))

#Prévision des 5 premières ligne de l’ensemble test
# 1. Assurer la cohérence des niveaux de facteurs
levels(test$Weather) <- levels(entrainement$Weather)
# 2. Sélection des 5 premières lignes de l'ensemble de test PRÉTRAITÉ
test_echantillon <- test_final[1:5, ]
# 3. Faire des prédictions sur l'échantillon de test
predictions_standardisees <- predict(modele_final, newdata = test_echantillon)
# 4. Déstandardiser les prédictions
# Attention : Inverser la transformation de Box-Cox ici !
lambda <- boxcox_result$x[which.max(boxcox_result$y)] # Récupérer le lambda utilisé
dans la transformation
predictions_origines <- (lambda * predictions_standardisees + 1)^(1/lambda)
# 5. Afficher les prédictions déstandardisées
print(predictions_origines)
# 6. Afficher les valeurs réelles correspondantes (de l'ensemble de test ORIGINAL)
print(test$Delivery_Time_min[1:5])
# 7. Calcul des métriques d'évaluation
mse <- mean((predictions_origines - test$Delivery_Time_min[1:5])^2)
rmse <- sqrt(mse)
mae <- mean(abs(predictions_origines - test$Delivery_Time_min[1:5]))
r2 <- 1 - sum((predictions_origines - test$Delivery_Time_min[1:5])^2) /
sum((test$Delivery_Time_min[1:5] - mean(test$Delivery_Time_min))^2)
# 8. Affichage des résultats
print(paste("MSE :", mse))
print(paste("RMSE :", rmse))
print(paste("MAE :", mae))
print(paste("R-carré :", r2))
```

C.ANOVA

```
# Charger les librairies nécessaires
library(dplyr)
library(MASS)  # Pour la fonction stepAIC
# Supposons que `entrainement_scaled` soit votre dataframe avec les variables indiquées
# Modèle initial avec toutes les variables
modele_initial <- lm(Delivery_Time_min ~ ., data = entrainement_scaled)
# Effectuer la sélection par étapes pour obtenir le modèle final
modele_final <- stepAIC(modele_initial, direction = "both")
# Afficher le résumé du modèle final
summary(modele_final)
# Afficher la table ANOVA pour le modèle final
anova(modele_final)

#Test d'égalité de la moyenne et test de significativité des moyennes
# Afficher la table ANOVA pour le modèle final
anova_results <- anova(modele_final)
# Variables qualitatives identifiées à partir des résultats ANOVA
variables_quali <- c("WeatherClear", "WeatherWindy", "WeatherRainy", "Traffic_LevelMedium", "Traffic_LevelHigh", "Time_of_DayEvening")
# Filtrer les résultats ANOVA pour les variables qualitatives uniquement
anova_quali <- anova_results[rownames(anova_results) %in% variables_quali, ]
# Afficher les résultats ANOVA pour les variables qualitatives
print("Résultats ANOVA pour les variables qualitatives:")
print(anova_quali)
# Traiter les variables qualitatives comme des facteurs
entrainement_scaled[variables_quali] <- lapply(entrainement_scaled[variables_quali], factor)
# Effectuer le test de Levene pour chaque variable qualitative
for (var in variables_quali) {
  if (var %in% colnames(entrainement_scaled)) {
    levene_result <- leveneTest(as.formula(paste("Delivery_Time_min ~", var)), data = entrainement_scaled)
    print(paste("Résultats du test de Levene pour l'égalité des variances pour", var, ":"))
    print(levene_result)
  } else {
    print(paste("La variable", var, "n'existe pas dans le dataframe 'entrainement_scaled'."))
  }
}
# Test post hoc de Tukey si ANOVA est significatif
if (any(anova_quali$`Pr(>F)` < 0.05)) {
  tukey_result <- TukeyHSD(aov(Delivery_Time_min ~ WeatherClear + WeatherWindy + WeatherRainy + Traffic_LevelMedium + Traffic_LevelHigh + Time_of_DayEvening, data = entrainement_scaled))
  print("Résultats du test post hoc de Tukey:")
  print(tukey_result)
} else {
  print("Aucune variable qualitative n'est significative selon l'ANOVA.")
}
```