Projet Data Mining: Apprntissage - ALBERI Markus

2025-04-04

EDA - Introduction

1. Introduction

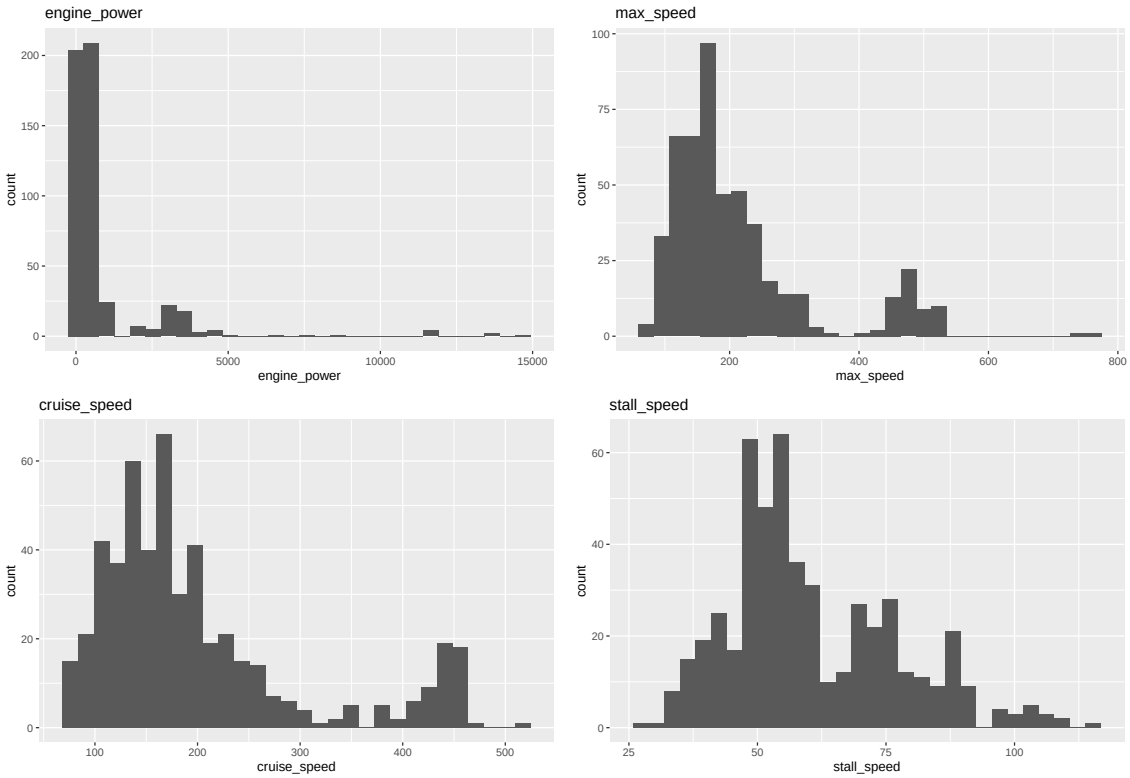
1.1 Problématique

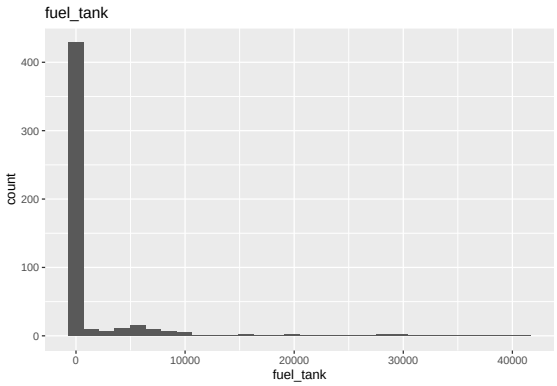
L'objectif de ce projet est de développer des modèles prédictifs pour estimer le prix d'un avion à partir de ses caractéristiques techniques. Ce problème a une forte valeur pratique dans le secteur aéronautique, où la connaissance anticipée de la valeur d'un appareil est cruciale pour les constructeurs, les loueurs et les acheteurs.

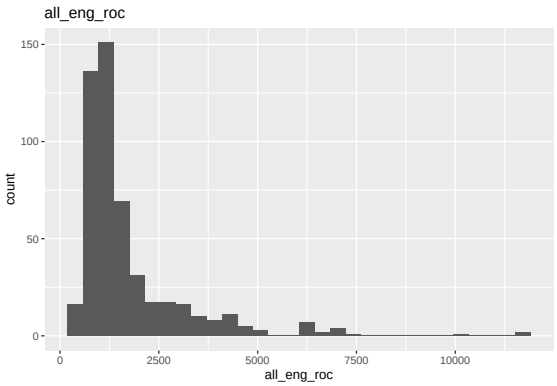
1.2 Données

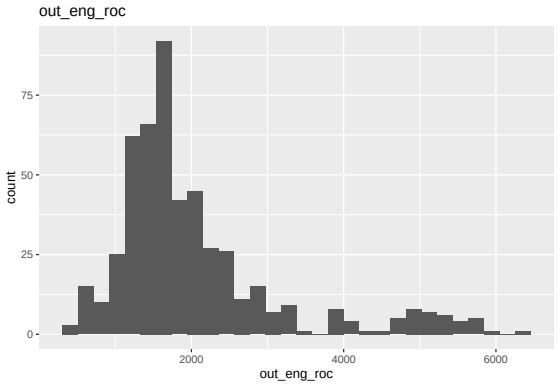
L'étude repose sur un jeu de données composé de 517 observations issues d'un fichier CSV, chacune représentant un avion, avec 16 variables. 14 d'entre elles sont quantitatives, notamment la puissance moteur, la vitesse de croisière, le poids à vide, ou encore la capacité du réservoir, et 2 sont qualitatives, le nom du modèle et le type de moteur. La variable cible est price, exprimée en dollars.

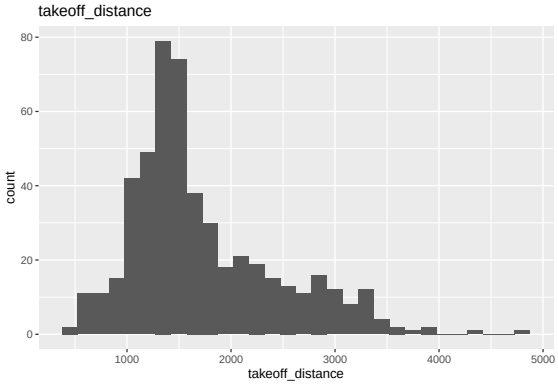
On remarque la présence de 10 valeurs manquantes dans la variable price.

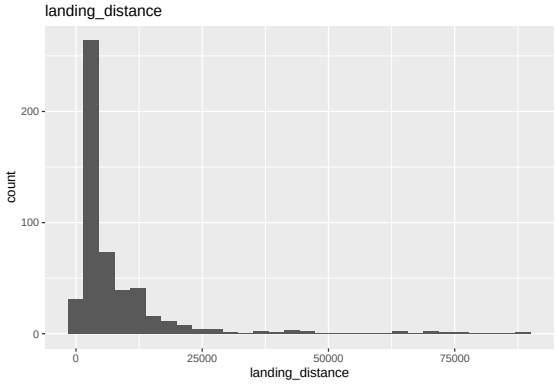


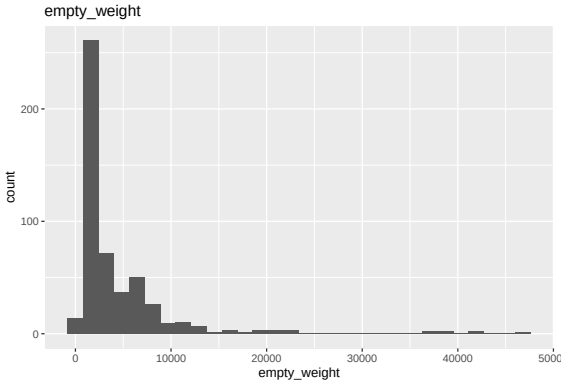

fuel_tank
count
fuel_tank

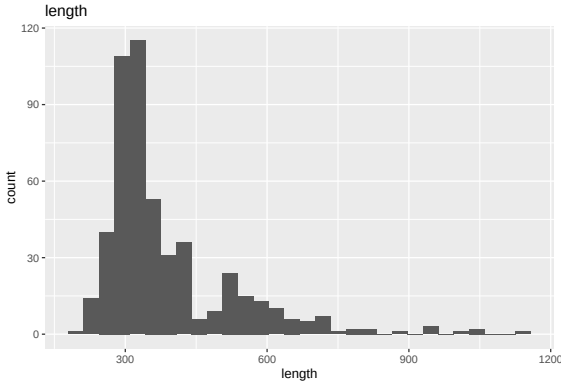

all_eng_roc
count
all_eng_roc

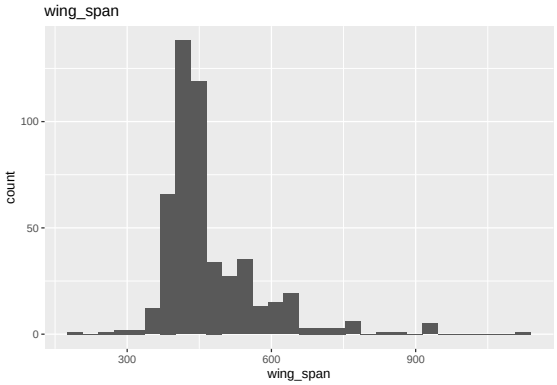

out_eng_roc
count
out_eng_roc

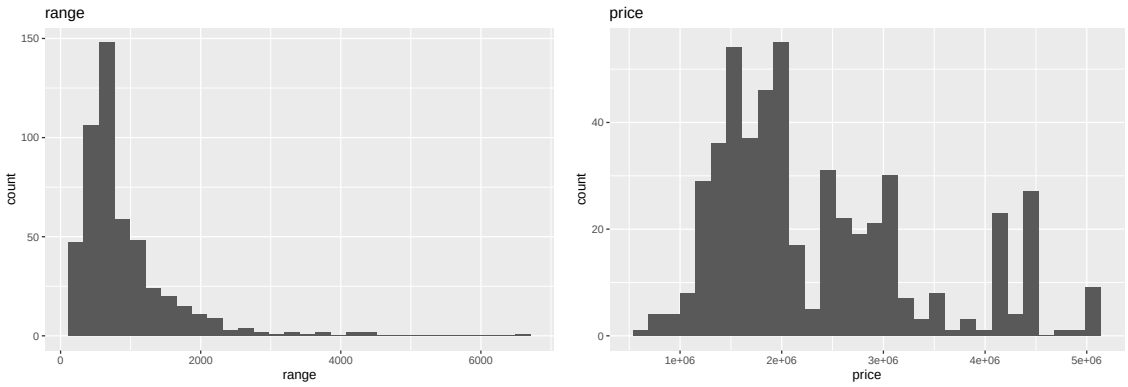

takeoff_distance
count
takeoff_distance


landing_distance
count
landing_distance

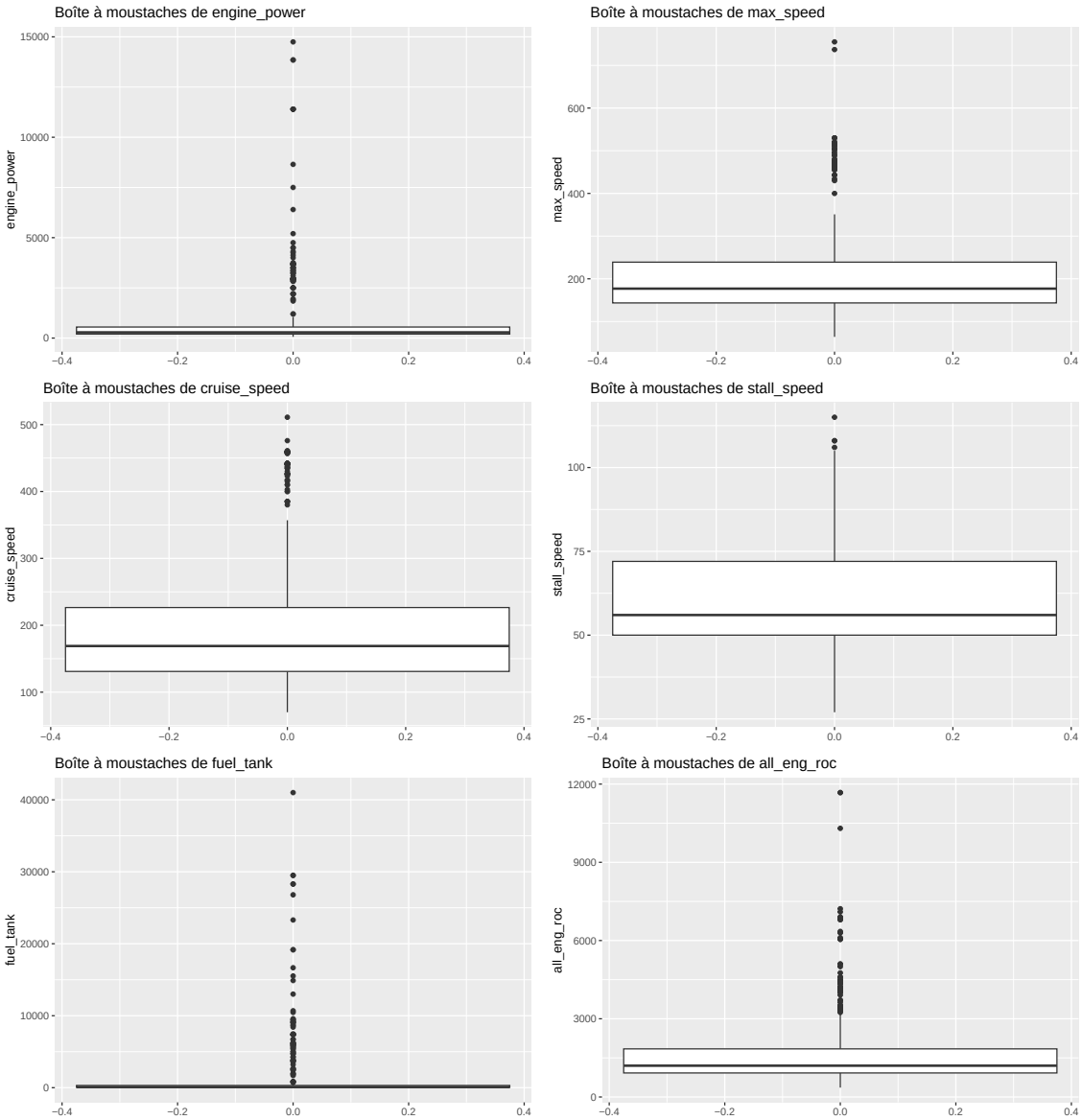

empty_weight
count
empty_weight

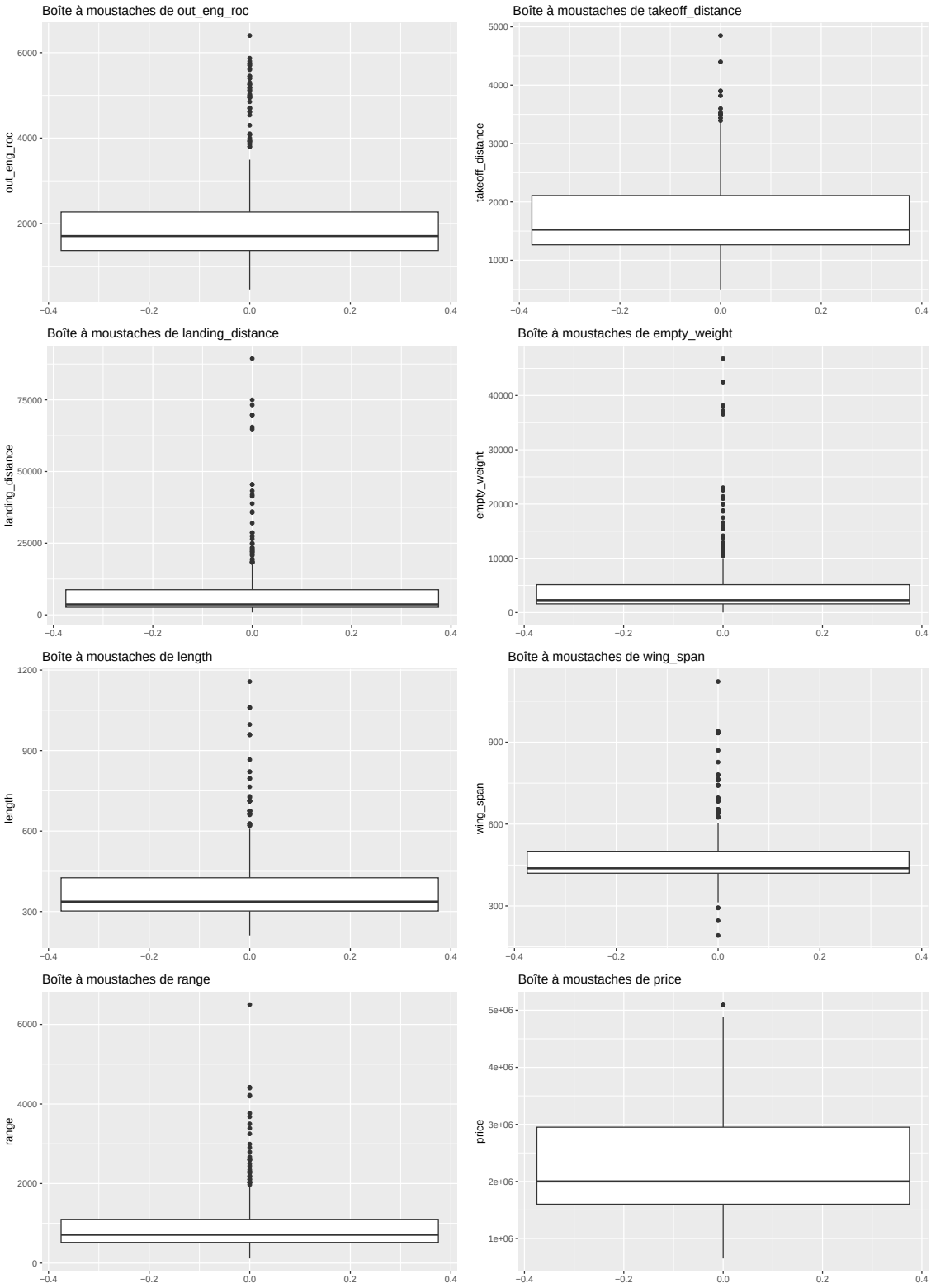

length
count
length


wing_span
count
wing_span



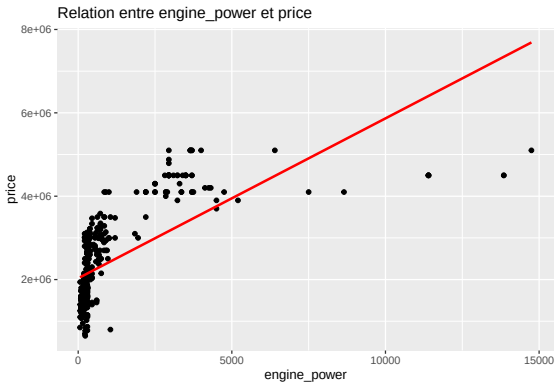
Aucune des variables ne présentent une distribution normale.



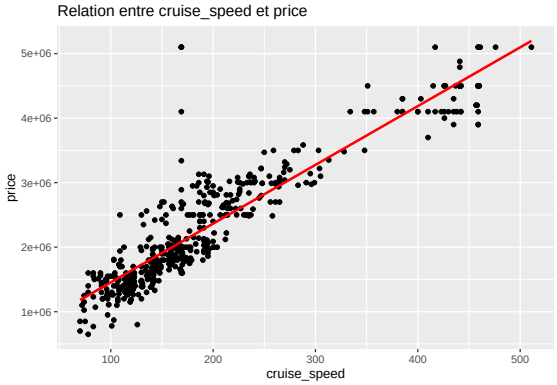


Certaines variables présentent de nombreux outliers.

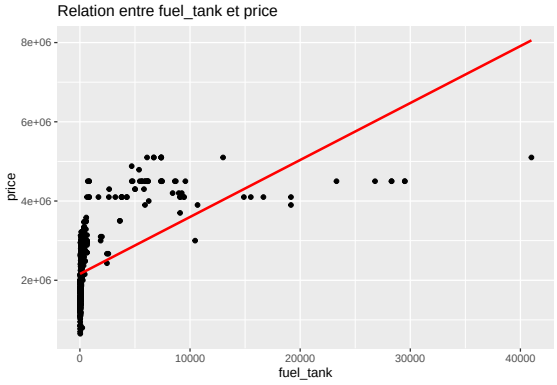
D'après l'IQR, une méthode utiliser afin d'identifier les observations grandement différentes des autres, seuls 9 le sont dans le jeu de données, assez peu pour être ignorées.

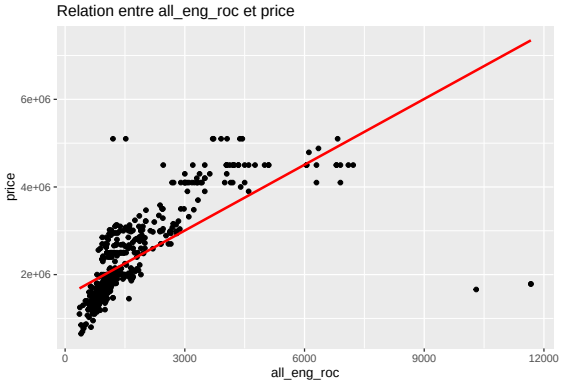

Relation entre engine_power et price
price
engine_power
8e+06
6e+06
4e+06
2e+06
0
5000
10000
15000


Relation entre max_speed et price
price
max_speed
6e+06
4e+06
2e+06
200
400
600

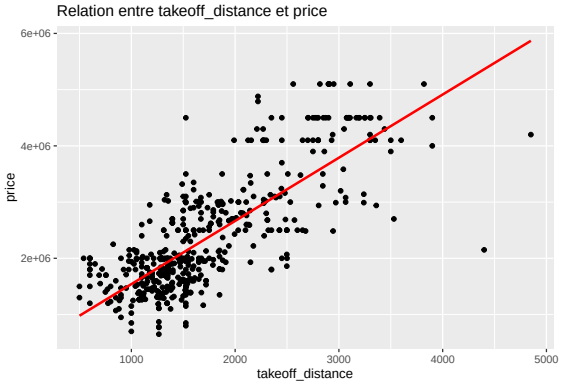

Relation entre cruise_speed et price
price
cruise_speed
5e+06
4e+06
3e+06
2e+06
1e+06
100
200
300
400
500

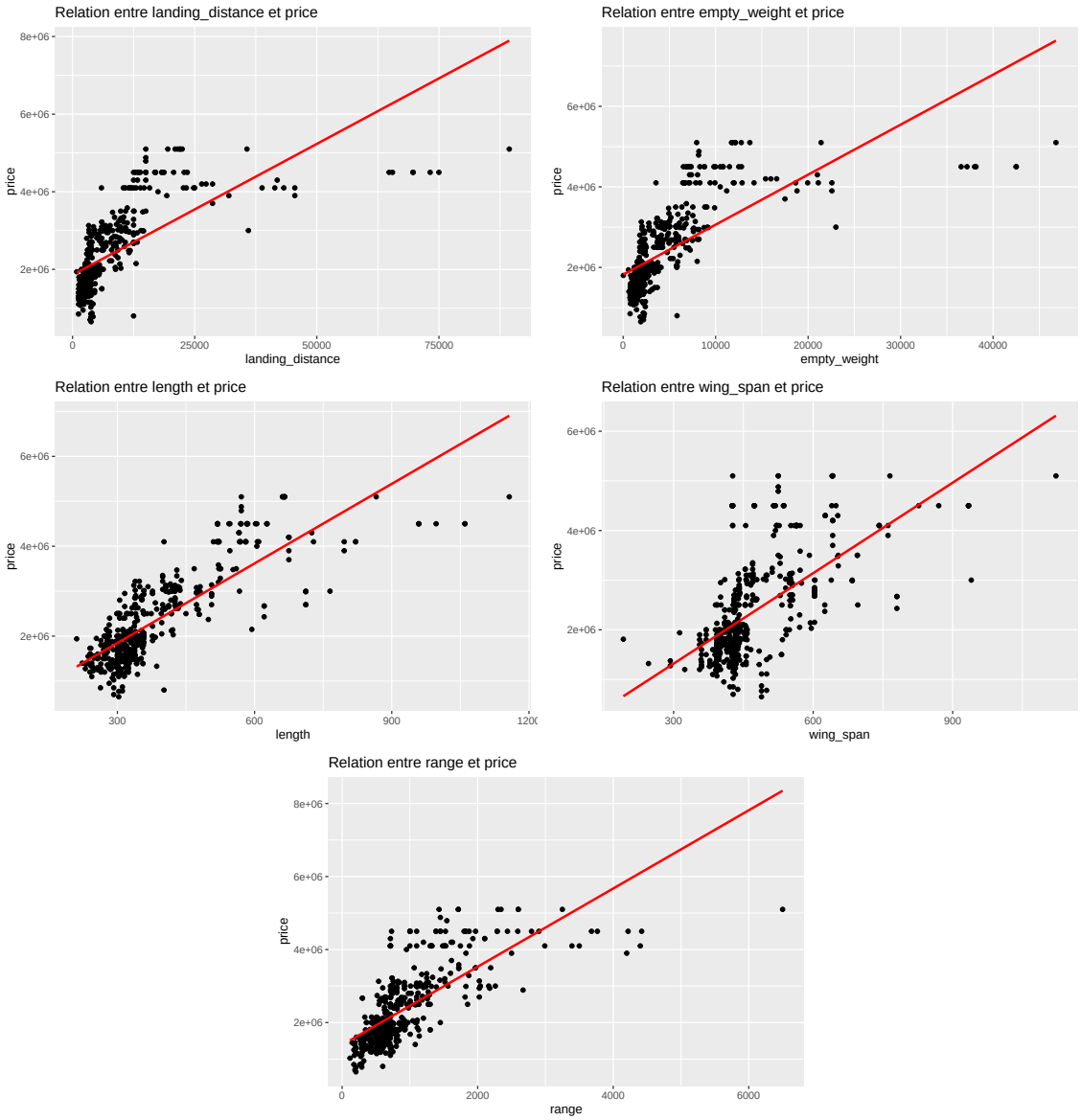

Relation entre stall_speed et price
price
stall_speed
5e+06
4e+06
3e+06
2e+06
1e+06
25
50
75
100


Relation entre fuel_tank et price
price
fuel_tank
8e+06
6e+06
4e+06
2e+06
0
10000
20000
30000
40000


Relation entre all_eng_roc et price
price
all_eng_roc
6e+06
4e+06
2e+06
0
3000
6000
9000
12000


Relation entre out_eng_roc et price
price
out_eng_roc
5e+06
4e+06
3e+06
2e+06
1e+06
2000
4000
6000


Relation entre takeoff_distance et price
price
takeoff_distance
6e+06
4e+06
2e+06
1000
2000
3000
4000
5000



On note une relation linéaire entre certaines variables et le prix notamment celles concernant la vitesse de l'appareil.

Table 1: Matrice des corrélations

	engn_	mx_sp	crs_s	stll_	fl_tn	all___	ot_n_	tkff_	lndn_	empt_	lngth	wng_s	range	price
engine_power	1.00	0.70	0.73	0.56	0.96	0.53	0.76	0.61	0.96	0.95	0.86	0.72	0.79	0.67
max_speed	0.70	1.00	0.87	0.76	0.65	0.71	0.76	0.74	0.72	0.72	0.79	0.56	0.71	0.85
cruise_speed	0.73	0.87	1.00	0.80	0.67	0.76	0.81	0.80	0.74	0.74	0.82	0.59	0.74	0.92
stall_speed	0.56	0.76	0.80	1.00	0.49	0.63	0.71	0.79	0.63	0.65	0.76	0.59	0.64	0.79
fuel_tank	0.96	0.65	0.67	0.49	1.00	0.44	0.73	0.55	0.95	0.94	0.83	0.73	0.80	0.61
all_eng_roc	0.53	0.71	0.76	0.63	0.44	1.00	0.61	0.60	0.50	0.50	0.60	0.35	0.49	0.72
out_eng_roc	0.76	0.76	0.81	0.71	0.73	0.61	1.00	0.83	0.76	0.76	0.83	0.60	0.73	0.77
takeoff_distance	0.61	0.74	0.80	0.79	0.55	0.60	0.83	1.00	0.65	0.66	0.77	0.61	0.68	0.78
landing_distance	0.96	0.72	0.74	0.63	0.95	0.50	0.76	0.65	1.00	1.00	0.92	0.83	0.84	0.69

	engn_	mx_spcrs_s	stll_	fl_tn	all___	ot_n_tkff_	lndn_	empt_lngth	wng_srange	price
empty_weight	0.95	0.72	0.74	0.65	0.94	0.50	0.76	0.66	1.00	1.00	0.92	0.85	0.84	0.69
length	0.86	0.79	0.82	0.76	0.83	0.60	0.83	0.77	0.92	0.92	1.00	0.87	0.85	0.81
wing_span	0.72	0.56	0.59	0.59	0.73	0.35	0.60	0.61	0.83	0.85	0.87	1.00	0.75	0.61
range	0.79	0.71	0.74	0.64	0.80	0.49	0.73	0.68	0.84	0.84	0.85	0.75	1.00	0.73
price	0.67	0.85	0.92	0.79	0.61	0.72	0.77	0.78	0.69	0.69	0.81	0.61	0.73	1.00

Les cinq variables présentant les corrélations les plus fortes avec le prix, cruise_speed, max_speed, length, stall_speed, et takeoff_distance, présentent des corrélations fortes entre elles. Ceci indique une possible multicolinéarité dans le modèle.

1.3 Méthodologie

Pour répondre à la problématique, une méthodologie rigoureuse a été définie, articulée en trois grandes étapes.

Tout d'abord, on prétraitera les données a été réalisé afin de garantir la qualité des analyses : les valeurs manquantes seront supprimées, ainsi que les variables purement descriptives et les variables catégorielles seront encodées. On séparera aussi les données en un échantillon d'apprentissage et en un échantillon de test.

Ensuite, plusieurs modèles prédictifs seront développer pour modéliser la relation entre les caractéristiques techniques et le prix des avions : des régressions linéaires, avec sélection manuelle, via le critère AIC et Cp de Mallows, une régression Lasso, une régression Ridge, un arbre de décision et un Random Forest.

Enfin, les modèles seront évalués, à l'aide d'un bootstrap, en s'appuyant sur deux métriques de performance classiques : le RMSE, Root Mean Square Error, pour mesurer la précision des prédictions, et le R^2 pour évaluer la qualité de l'ajustement.

2. Calibrage des modèles

2.1 Régression linéaire avec Élimination descendante manuelle basée sur les p-values.

Le premier modèle est calibré par la méthode d'élimination descendante manuelle.

Après avoir estimer un modèle de régression linéaire incluant toutes les variables explicatives sur l'échantillon d'apprentissage. On vise à sélectionner un modèle parcimonieux en retirant itérativement les variables non significatives. Ici on a à chaque itération retirer la variable présentant la p-value la plus élevée supérieur au seuil de significativité 5%. Le processus étant répété jusqu'à ce que toutes les variables retenues atteignent une p-value inférieure à 0.05, celui-ci garantie ainsi leurs contributions à l'explication significative de la variable cible. Le modèle final met ici en avant les variables engine, max_speed, cruise_speed, landing_distance, empty_weight, wing_span et range.

2.2 Régression linéaire avec sélection automatique avec AIC descendante.

On a ensuite calibré un second modèle de régression linéaire, celui-ci à l'aide d'une procédure de sélection automatique basée sur le critère d'information d'Akaike, AIC. Dans cette approche, l'algorithme débute par l'estimation d'un modèle complet qui inclut toutes les variables explicatives sur l'échantillon d'apprentissage. Ensuite, il procède à une élimination descendante itérative : à chaque étape, la variable dont la suppression conduit à la plus faible augmentation ou une diminution de l'AIC est retirée du modèle. Ce processus est répété jusqu'à ce qu'il ne soit plus possible d'améliorer le critère en supprimant une nouvelle variable, créeant ainsi un modèle parcimonieux.

Le modèle final obtenu retient sept variables clés : engine_type, max_speed, cruise_speed, landing_distance, empty_weight, wing_span et range. Parmi celles-ci, engine_type et cruise_speed se distinguent par leur contribution majeure.

2.3 Régression linéaire avec Sélection automatique avec AIC ascendante.

On a ensuite calibré un troisième modèle de régression linéaire en utilisant une procédure de sélection automatique ascendante basée sur le critère AIC. Dans cette approche, on débute avec un modèle minimal qui ne comporte que l'intercept, puis on ajoute progressivement les variables explicatives, en sélectionnant à chaque étape celle qui améliore le plus le critère AIC. Ce processus se poursuit jusqu'à ce qu'aucun ajout ne permette d'optimiser davantage le modèle. Le résultat final est un modèle parcimonieux intégrant les variables suivantes : cruise_speed, max_speed, wing_span, landing_distance, range, empty_weight et engine_type. Parmi elles, cruise_speed se distingue particulièrement par son impact significatif sur la prédiction du prix.

2.4 Régression linéaire minimisant le critère Cp de Mallows.

Pour le quatrième modèle, on a opté pour une approche basée sur la minimisation du critère Cp de Mallows. Dans cette approche, on commence par évaluer l'ensemble de tous les sous-modèles possibles et on compare les modèles selon leur Cp. Ensuite on identifie le sous-modèle ayant la valeur minimale du Cp. Ici, le modèle final a pour variables : engine_type, engine_power, max_speed, cruise_speed, stall_speed, landing_distance, empty_weight, length, wing_span et range. Parmi celles-ci, les variables liées au type de moteur, ainsi que max_speed et cruise_speed, se distinguent par leur contribution significative à la prédiction du prix, confirmant leur importance dans l'explication de la variable cible.

2.5 Lasso et Ridge

Pour calibrer les cinquième et sixième modèles , les modèles régularisés Lasso et Ridge, nous avons d'abord préparé les données d'entrée en créant une matrice de variables explicatives en excluant l'intercept. Les jeux de données X_train et y_train ont ainsi été construits à partir des données d'entraînement, tandis que X_test a été généré pour les prédictions futures.

L'entraînement des modèles a ensuite été réalisé en effectuant une validation croisée.

Pour le modèle Lasso, le paramètre alpha a été fixé à 1, ce qui favorise une sélection stricte des variables en appliquant une pénalisation L1. Le paramètre de régularisation optimal, lambda.min, a été automatiquement déterminé en minimisant l'erreur de validation croisée. De la même manière, le modèle Ridge a été calibré en fixant alpha = 0, ce qui induit une pénalisation L2 favorisant des coefficients plus stables mais non nuls. Là aussi, la meilleure valeur de pénalisation a été extraite à partir de la validation croisée.

2.6 Arbre de décision.

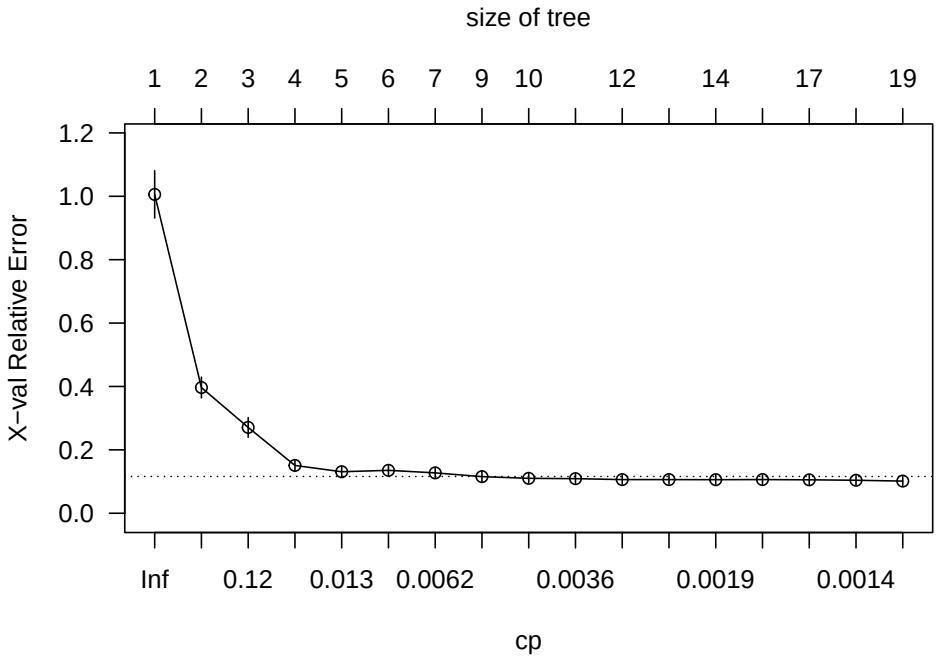
Le septième modèle de régression, un abre de décision, a été calibré en deux étapes. Tout d'abord, on a construit un arbre initial en utilisant toutes les variables que l'on a entrainé sur l'échantillon d'apprentissage. Celui-ci est défini par les paramètres:

1) Cp, le seuil de complexité définit le seuil minimal de réduction de l'erreur requise pour que la division d'un nœud soit justifiée. Ici la valeur 0.001 a été choisie pour permettre une exploration initiale des divisions potentielles tout en évitant un arbre trop complexe.

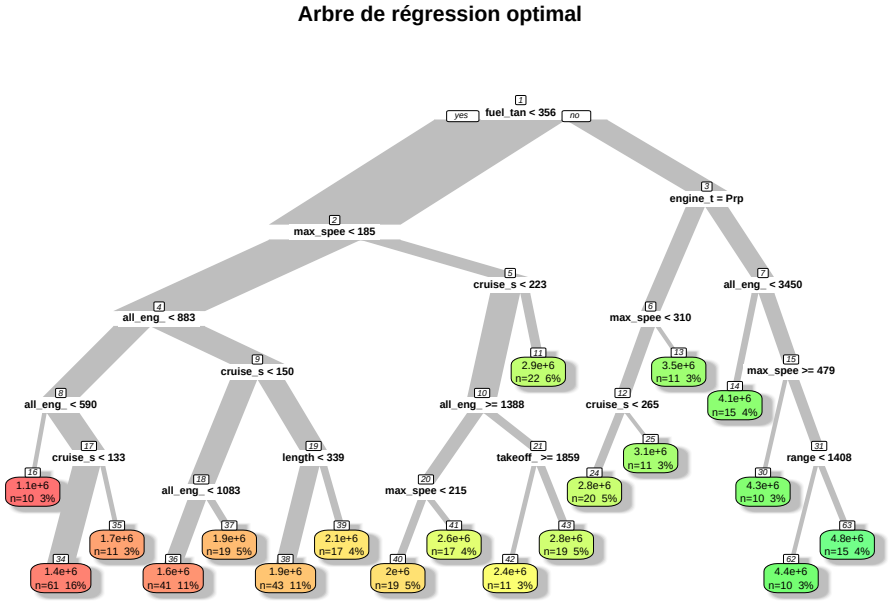
2) Minsplit,le nombre minimal d'observations pour qu'un nœud puisse être divisé. Ici la valeur 20 a été choisie valeurafin d'éviter la création de nœuds trop petits qui pourraient conduire au surapprentissage.

3) Minbucket, le nombre minimal d'observations dans une feuille. Ici la valeur 10 a été choisie pour garantir que chaque feuille contient suffisamment d'observations pour être statistiquement significative.

4) Xval, le nombre de fold pour validation croisée. Ici la valeur 10 a été choisie pour permettre d'estimer la performance du modèle de manière robuste.



Ensuite, nous avons procédé à l'élagage de l'arbre afin de simplifier sa structure en éliminant les divisions non informatives. Le paramètre de complexité optimal, cp.opt, ici choisi à 0.01, a été déterminé en sélectionnant la valeur qui minimisait l'erreur de validation croisée, xerror, 0.11.



Le modèle final, obtenu après élagage, met en lumière les variables les plus influentes, telles que fuel_tank et engine_type , pour prédire le prix.

2.7 Random Forest

Le huitième modèle de régression, un modèle Random Forest, a lui aussi été calibré en deux étapes.

Tout d'abord, on a construit un modèle initial sur l'échantillon d'apprentissage en utilisant toutes les variables explicatives. Ce modèle initial est défini par les paramètres suivants :

1) ntree, le nombre d'arbres, initialement fixé à 500. Cette valeur a été choisie afin d'assurer une diversité suffisante dans la forêt pour capturer des relations complexes dans les données.

2) mtry, le nombre de variables candidates par division, est ici défini comme la racine carrée du nombre total de variables. Ce choix permet de limiter la corrélation entre les arbres individuels et ainsi d'améliorer la robustesse du modèle.

3) nodesize, la taille minimale des feuilles, ici fixée à 5. Le paramètre permet de garantir que chaque feuille contient un nombre d'observations suffisant pour être statistiquement significative.

4) replace, l'option d'échantillonnage avec remise, est ici activée pour favoriser la variabilité entre les arbres et enrichir l'ensemble des modèles individuels.

Ensuite, nous avons affiné le modèle à l'aide d'une validation croisée à 10 fold pour déterminer le nombre optimal d'arbres. Cette étape d'optimisation nous a permis de trouver le nombre d'arbres qui donnait les meilleurs résultats tout en gardat le modèle le plus simple possible. On a ensuite reconstruit le modèle en utilisant ce nombre optimal d'arbres, en gardant tous les autres réglages inchangés.

3. Comparaison des modèles et analyse de l'importance des variables.

Dans cette partie on étudiera les performances des modèles ainsi que l'influence des variables sur le prix.

Pour évaluer les modèles, nous avons utilisé un jeu de données de test indépendant, distinct de celui utilisé pour l'entraînement. Chaque modèle a été appliqué pour prédire le prix des avions sur ce jeu de test, et nous avons ensuite comparé ces prédictions aux valeurs réelles observées. Nous avons principalement utilisé deux indicateurs de performance :

1) la RMSE, Root Mean Square Error. Cette métrique mesure l'erreur moyenne des prédictions. Une RMSE plus faible indique que les prédictions du modèle sont, en moyenne, plus proches des valeurs réelles.

2) Le R^2, Coefficient de Détermination. Cette mesure indique la proportion de la variance du prix qui est expliquée par le modèle. Un R^2 plus élevé signifie que le modèle capture mieux la variabilité des données.

En complément, pour vérifier la stabilité des résultats, nous avons appliqué une méthode Bootstrap avec 999 répétitions pour estimer des intervalles de confiance autour de ces indicateurs. Ceux-ci nous permettent de mieux évaluer la stabilité et la robustesse des performances de chaque modèle sur le jeu de test.

Table 2: Comparaison des performances des modèles traditionnels avec IC Bootstrap

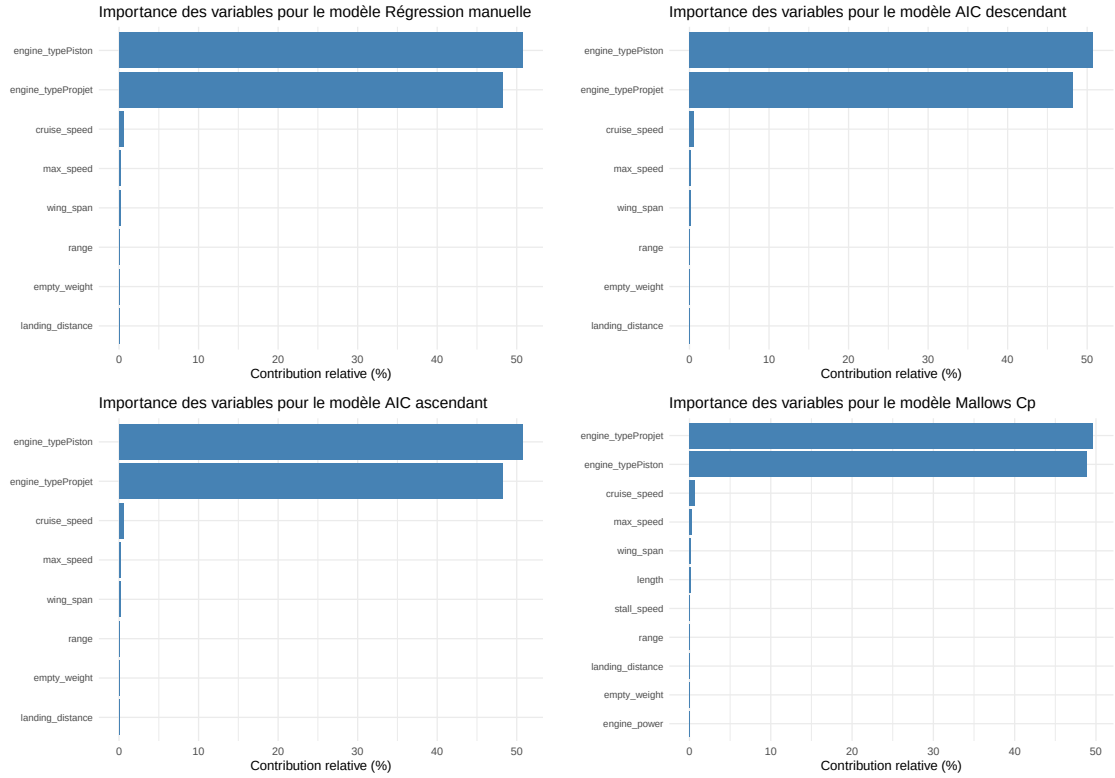
Modèle	RMSE Bootstrap	RMSE IC Inf	RMSE IC Sup	R^2 Bootstrap	R^2 IC Inf	R^2 IC Sup
AIC Ascendant	403593.0	335489.9	470763.3	0.85	0.80	0.886
AIC Descendant	405147.9	339334.1	472916.7	0.85	0.80	0.891
Arbre de Régression	331947.5	284290.6	383671.6	0.90	0.85	0.931
Mallows Cp	393699.6	330893.0	463650.8	0.85	0.81	0.896

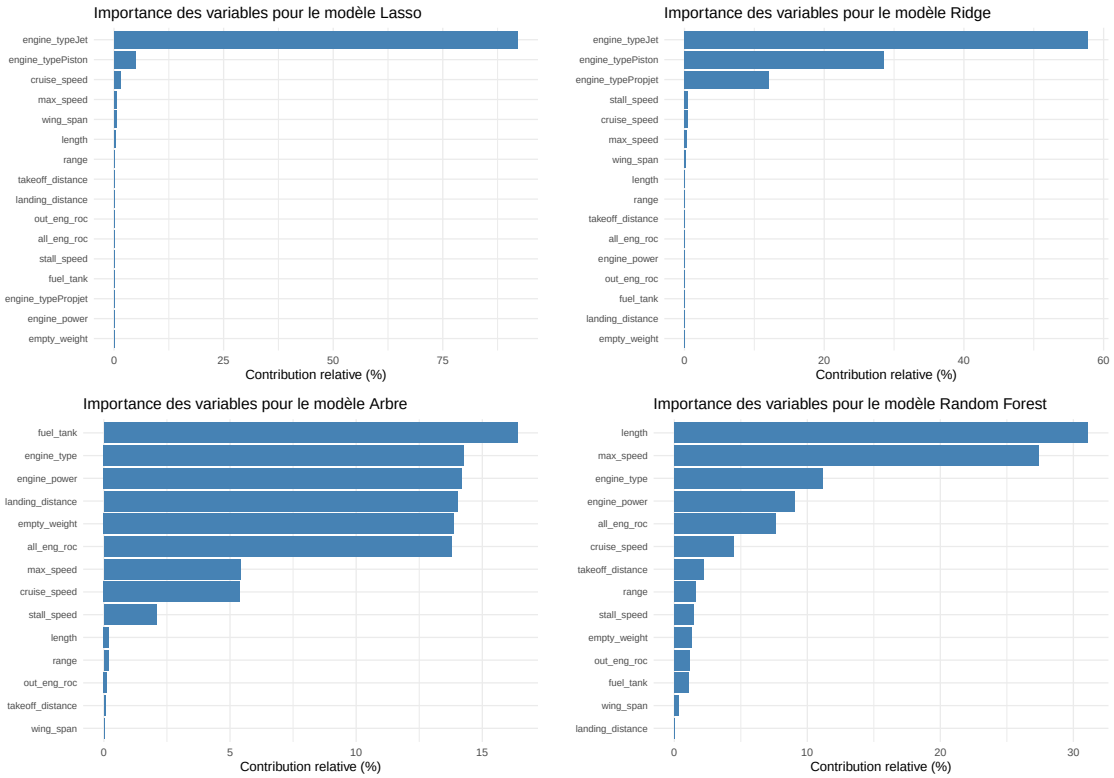
Modèle	RMSE Bootstrap	RMSE IC Inf	RMSE IC Sup	R^2 Bootstrap	R^2 IC Inf	R^2 IC Sup
Manuelle (p-values)	403168.9	337731.5	469132.4	0.85	0.80	0.889
Random Forest	350046.6	284567.9	417059.9	0.88	0.82	0.929

Table 3: Comparaison des performances des modèles régularisés avec IC Bootstrap

Modèle	RMSE Bootstrap	IC Inf	IC Sup	R^2 Bootstrap	IC Inf	IC Sup
Lasso	392071.1	330073.3	467608.7	0.86	0.81	0.895
Ridge	385395.2	333139.7	442208.2	0.86	0.82	0.896

D'après les résultats, l'arbre de régression se distingue comme le modèle le plus performant. En effet, il présente un RMSE moyen de 331947.5, avec un intervalle de confiance allant de 284290.6 à 383671.6, et un R^2 moyen de 0.90 avec un IC de 0.85 à 0.931. Ces valeurs indiquent que ce modèle prédit le prix des avions avec une précision supérieure à celle des autres modèles. Ainsi on considère que l'arbre de régression offre le meilleur compromis entre précision et robustesse, et constitue donc le choix optimal.





L'analyse de l'importance des variables met en évidence des différences notables selon les modèles.

Pour les modèles de régression traditionnelle qu'il s'agisse de la régression manuelle, de l'AIC descendant, de l'AIC ascendant ou encore de la méthode de sélection via le critère Cp de Mallows, la variable engine_typePiston domine largement, représentant à elle seule environ 50 % de l'importance totale. À côté, les variables cruise_speed, max_speed et wing_span n'apportent qu'une contribution très marginale, tandis que toutes les autres variables se révèlent négligeables.

Le modèle Lasso met davantage l'accent sur la variable engine_typeJet, qui représente plus de 75 % de l'importance, loin devant engine_typePiston, qui représente environ 5 %, tandis que le reste des variables, y compris les vitesses et les dimensions, n'ont qu'un impact très faible voire nul.

Le modèle Ridge, quant à lui, répartit un peu plus l'importance : engine_typeJet reste prépondérante, un peu moins de 60 %, suivie de engine_typePiston, environ 30 %, et de engine_typePropjet, environ 12 %. Les autres variables, notamment cruise_speed, stall_speed, max_speed, wing_span et length, n'interviennent que de façon marginale.

Avec l'arbre de décision, l'importance des variables se répartit plus équitablement : fuel_tank dépasse légèrement les 15 %, tandis que engine_type, engine_power, landing_distance, empty_weight et all_eng_roc gravitent autour des 15 %. Max_speed et cruise_speed se situent autour de 5 %, stall_speed en dessous de 2,5 %, et des variables comme length, range, out_eng_roc, takeoff_distance ou wing_span ont une contribution très faible.

Enfin, le modèle Random Forest présente une hiérarchie encore différente : length ressort comme la variable la plus influente, avec un peu plus de 30 %, suivie par max_speed, un peu moins de 30 %. Engine_type arrive ensuite avec environ 11 %, tandis que engine_power et all_eng_roc tournent autour des 10 %. Les autres variables telles que cruise_speed, takeoff_distance, range, stall_speed, empty_weight, out_eng_roc, fuel_tank et wing_span ont toutes une importance inférieure à 5 %.

Code_Projet_Data_Mining_app

2025-04-06

```
knitr::opts_chunk$set(echo = TRUE)
options(repos = c(CRAN = "https://cran.r-project.org"))
install.packages("randomForest")
```

```
##
## The downloaded binary packages are in
##  /var/folders/hy/c9wjv6px46b1z_1cz_kgxbx80000gn/T//RtmpXLnKyR/downloaded_packages
```

```
install.packages("Metrics")
```

```
##
## The downloaded binary packages are in
##  /var/folders/hy/c9wjv6px46b1z_1cz_kgxbx80000gn/T//RtmpXLnKyR/downloaded_packages
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(knitr)
library(dplyr)
```

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

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

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

```
library(MASS)
```

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

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

```r
library(Metrics)
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following objects are masked from 'package:Metrics':
##
##     precision, recall
```

```r
library(leaps)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

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

```r
library(caret)
library(ggplot2)
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.2
```

```r
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

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

```r
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:car':
##
##     logit

## The following object is masked from 'package:lattice':
##
##     melanoma
```

```r
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
# Définition des fonctions de performance
rmse = function(obs, pred) sqrt(mean((obs - pred)^2))
r2 = function(obs, pred) 1 - sum((obs - pred)^2)/sum((obs - mean(obs))^2)
```

```r
tinytex::is_tinytex()
```

```
## [1] TRUE
```

```r
tinytex::tinytex_root()
```

```
## [1] "/Users/markus/Library/TinyTeX"
```

```r
data = read.csv("/Users/markus/Downloads/aircraft_price.csv")
head(data)
```

```
##                          model_name engine_type engine_power max_speed
## 1      100 Darter (S.L. Industries)      Piston          145       104
## 2                      7 CCM Champ      Piston           85        89
## 3      100 Darter (S.L. Industries)      Piston           90        90
## 4                       7 AC Champ      Piston           85        88
## 5      100 Darter (S.L. Industries)      Piston           65        83
## 6 PA-60-700P Aerostar (preliminary)      Piston           65        78
##   cruise_speed stall_speed fuel_tank all_eng_roc out_eng_roc takeoff_distance
## 1           91          46        36         450         900              1300
## 2           83          44        15         600         720               800
## 3           78          37        19         650         475               850
## 4           78          37        19         620         500               850
## 5           74          33        14         370         632               885
## 6           72          33        15         360         583               880
##   landing_distance empty_weight length wing_span range   price
```

```
## 1             2050         1180    303       449   370 1300000
## 2             1350          820    247       433   190 1230000
## 3             1300          810    257       420   210 1600000
## 4             1300          800    257       420   210 1300000
## 5             1220          740    257       420   175 1250000
## 6             1250          786    244       433   180 1100000
```

```
#Infos Dataset
str(data)
```

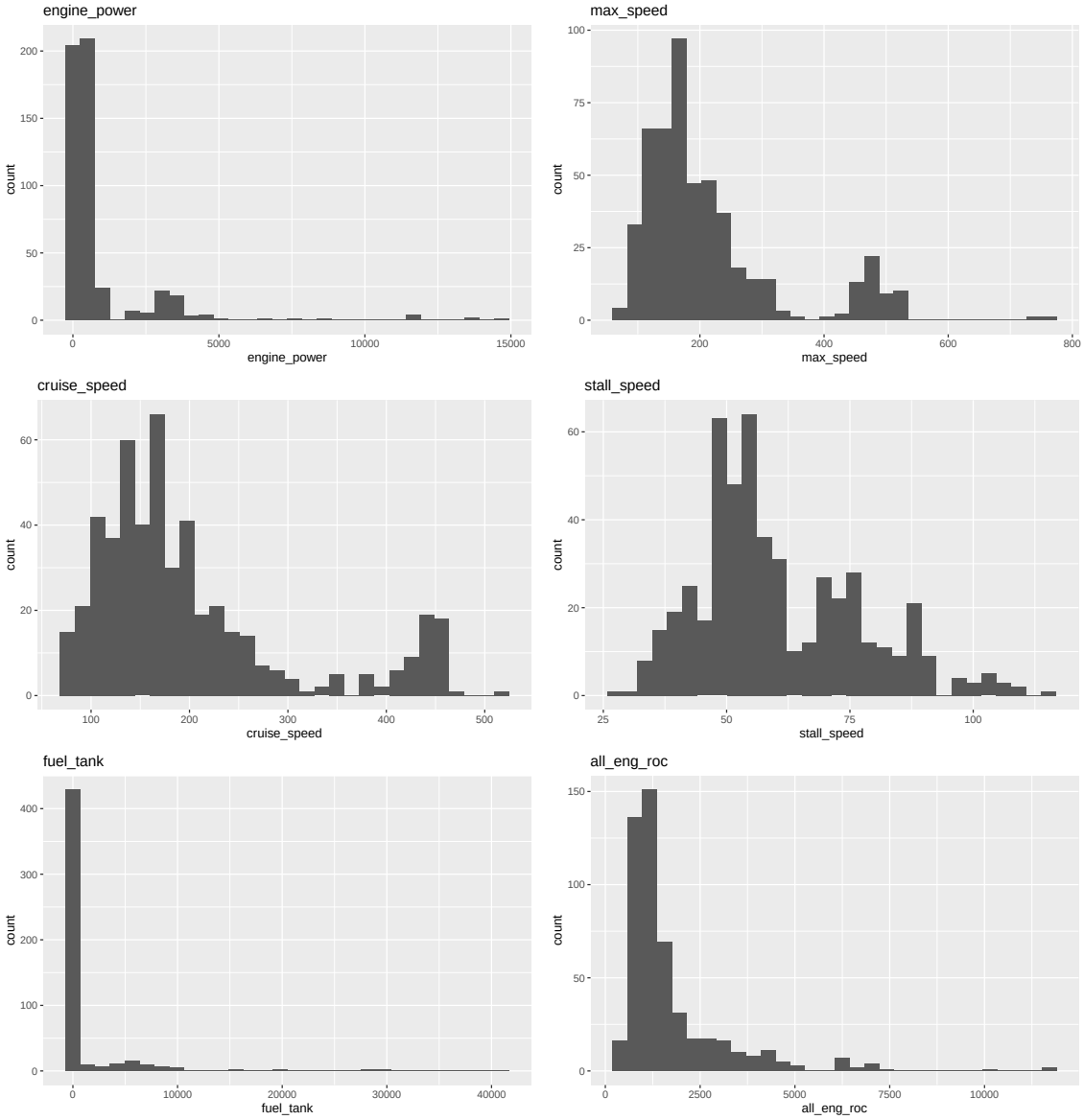
```
## 'data.frame':    517 obs. of  16 variables:
##  $ model_name      : chr  "100 Darter (S.L. Industries)" "7 CCM Champ" "100 Darter (S.L. Industries)"
##  $ engine_type     : chr  "Piston" "Piston" "Piston" "Piston" ...
##  $ engine_power    : num  145 85 90 85 65 65 350 290 290 290 ...
##  $ max_speed       : num  104 89 90 88 83 78 264 262 257 257 ...
##  $ cruise_speed    : num  91 83 78 78 74 72 230 247 235 237 ...
##  $ stall_speed     : num  46 44 37 37 33 33 80 77 77 77 ...
##  $ fuel_tank       : num  36 15 19 19 14 15 165 165 165 165 ...
##  $ all_eng_roc     : num  450 600 650 620 370 ...
##  $ out_eng_roc     : num  900 720 475 500 632 583 3080 2250 2490 2490 ...
##  $ takeoff_distance: num  1300 800 850 850 885 ...
##  $ landing_distance: num  2050 1350 1300 1300 1220 ...
##  $ empty_weight    : num  1180 820 810 800 740 ...
##  $ length          : num  303 247 257 257 257 244 418 417 418 418 ...
##  $ wing_span       : num  449 433 420 420 420 433 440 439 440 440 ...
##  $ range           : num  370 190 210 210 175 ...
##  $ price           : num  1300000 1230000 1600000 1300000 1250000 1100000 2500000 2800000 2500000 30
```

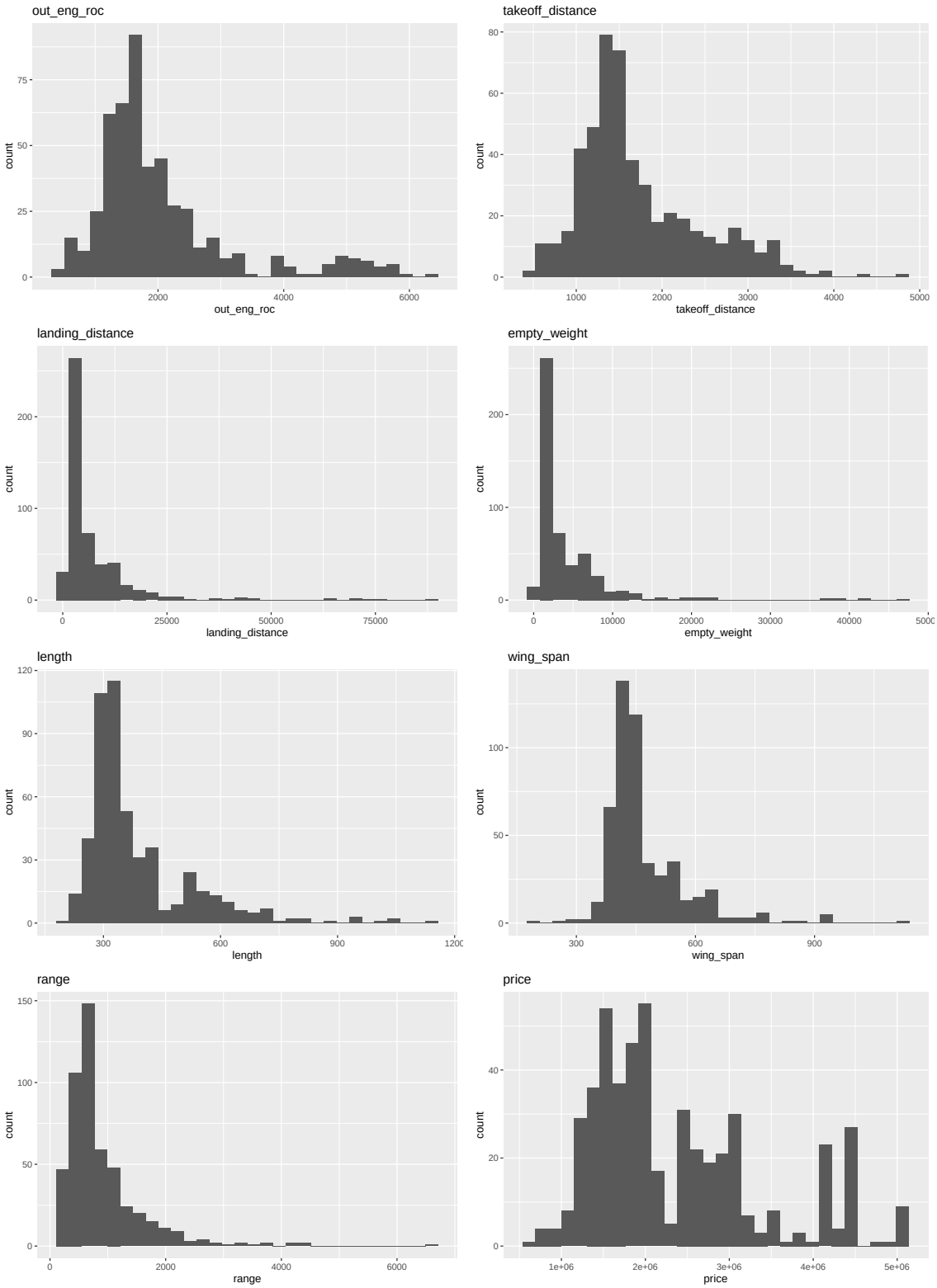
```
#Résumé statistique
summary(data)
```

```
##   model_name        engine_type         engine_power       max_speed
##  Length:517         Length:517         Min.   :   60.0   Min.   : 64.0
##  Class :character   Class :character   1st Qu.:  200.0   1st Qu.:143.0
##  Mode  :character   Mode  :character   Median :  285.0   Median :177.0
##                                        Mean   :  869.3   Mean   :212.8
##                                        3rd Qu.:  550.0   3rd Qu.:238.0
##                                        Max.   :14750.0   Max.   :755.0
##
##   cruise_speed    stall_speed       fuel_tank      all_eng_roc
##  Min.   : 70.0   Min.   : 27.00   Min.   :   12   Min.   :  360
##  1st Qu.:131.0   1st Qu.: 50.00   1st Qu.:   50   1st Qu.:  924
##  Median :169.0   Median : 56.00   Median :   89   Median : 1200
##  Mean   :200.2   Mean   : 60.66   Mean   : 1419   Mean   : 1718
##  3rd Qu.:229.0   3rd Qu.: 73.00   3rd Qu.:  335   3rd Qu.: 1861
##  Max.   :511.0   Max.   :115.00   Max.   :41000   Max.   :11673
##
##   out_eng_roc   takeoff_distance landing_distance  empty_weight
##  Min.   : 457   Min.   : 500     Min.   :  567    Min.   :   23
##  1st Qu.:1365   1st Qu.:1265     1st Qu.: 2650    1st Qu.: 1575
##  Median :1706   Median :1525     Median : 3625    Median : 2286
##  Mean   :2047   Mean   :1733     Mean   : 7485    Mean   : 4377
##  3rd Qu.:2280   3rd Qu.:2110     3rd Qu.: 8800    3rd Qu.: 5164
```

```
##  Max.   :6400   Max.   :4850     Max.   :89400    Max.   :46800
##
##      length         wing_span         range            price
##  Min.   : 211.0   Min.   : 192.0   Min.   : 117.0   Min.   : 650000
##  1st Qu.: 302.0   1st Qu.: 420.0   1st Qu.: 517.0   1st Qu.:1600000
##  Median : 337.0   Median : 438.0   Median : 713.0   Median :2000000
##  Mean   : 387.2   Mean   : 472.5   Mean   : 911.4   Mean   :2362673
##  3rd Qu.: 426.0   3rd Qu.: 507.0   3rd Qu.:1100.0   3rd Qu.:2950000
##  Max.   :1157.0   Max.   :1122.0   Max.   :6500.0   Max.   :5100000
##                                                     NA's   :10
```

```
#suppression valeurs manquantes
data = na.omit(data)
```



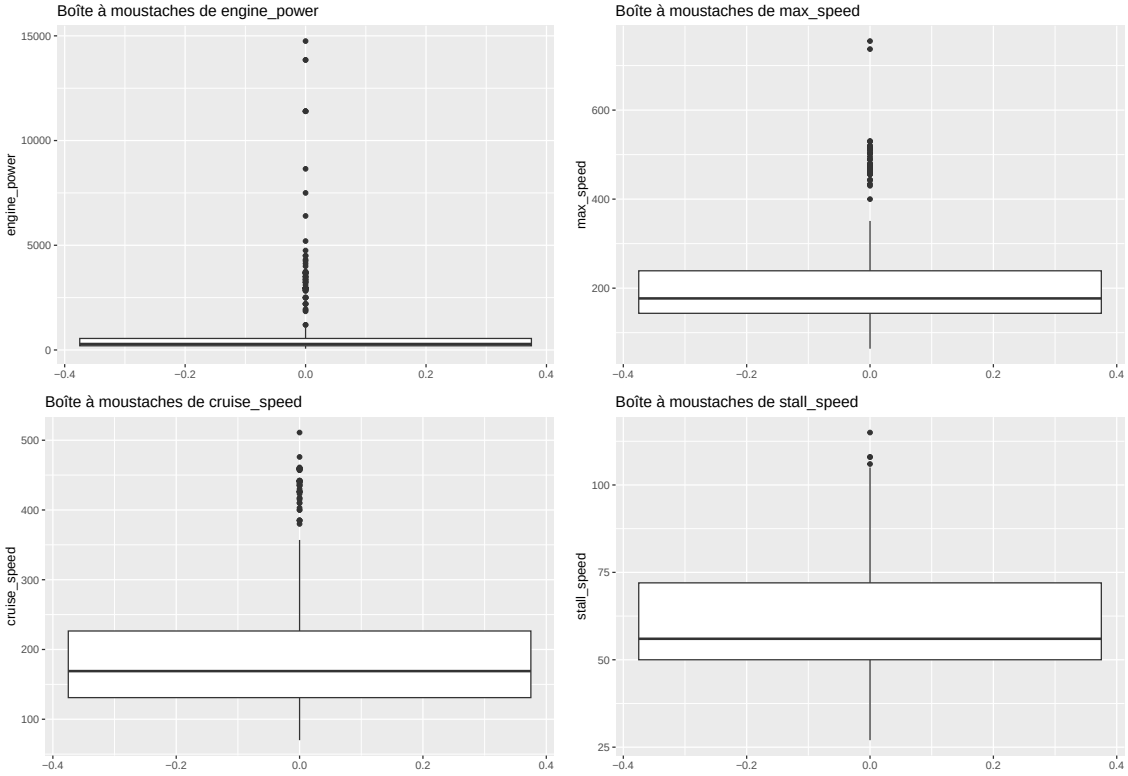


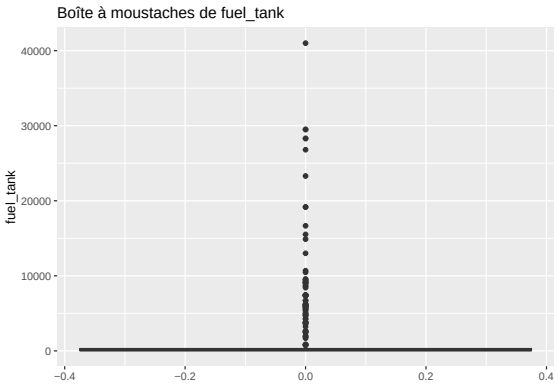

```
knitr::kable(summary(num_vars %>% as.data.frame(.) %>% dplyr::mutate_if(is.numeric, round, 2)))
```

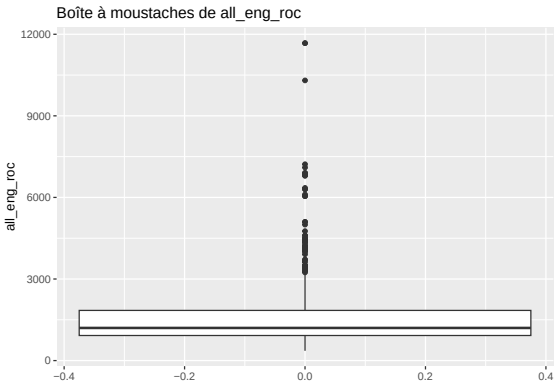
```
## Warning: 'xfun::attr()' is deprecated.
```

```
## Use 'xfun::attr2()' instead.
## See help("Deprecated")

## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

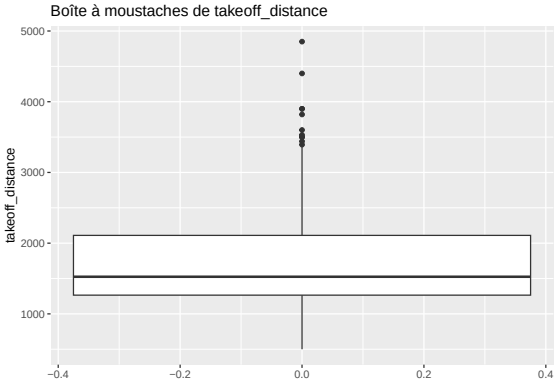
engine_power	max_speed	cruise_speed	stall_speed	fuel_tank	engine_count	takeoff_distance	landing_distance	empty_weight	wing_span	range	price
Min. : 60.0	Min. : 64.0	Min. : 70.0	Min. : 27.00	Min. : 12	Min. : 360	Min. : 457	Min. : 500	Min. : 850	Min. : 23	Min. : 211.0	Min. : 192.0	Min. : 117.0	Min. : 650000
1st Qu.: 200.0	1st Qu.:143.0	1st Qu.:131.0	1st Qu.: 50.00	1st Qu.: 50	1st Qu.: 922	1st Qu.:1368	1st Qu.:1265	1st Qu.: 2650	1st Qu.: 1600	1st Qu.: 302.0	1st Qu.: 420.0	1st Qu.: 520.0	1st Qu.:1600000
Median : 285.0	Median :177.0	Median :169.0	Median : 56.00	Median : 90	Median : 1200	Median :1706	Median :1525	Median : 3650	Median : 2286	Median : 337.0	Median : 438.0	Median : 713.0	Median :2000000
Mean : 863.6	Mean :214.2	Mean :199.8	Mean : 60.68	Mean : 1413	Mean : 1715	Mean :2048	Mean :1731	Mean : 7503	Mean : 4381	Mean : 387.4	Mean : 472.0	Mean : 911.3	Mean :2362673
3rd Qu.: 550.0	3rd Qu.:239.0	3rd Qu.:226.0	3rd Qu.: 72.00	3rd Qu.: 310	3rd Qu.: 1845	3rd Qu.:2270	3rd Qu.:2110	3rd Qu.: 8775	3rd Qu.: 5142	3rd Qu.: 426.0	3rd Qu.: 500.5	3rd Qu.:1100.0	3rd Qu.:2950000
Max. :14750.0	Max. :755.0	Max. :511.0	Max. :115.00	Max. :41000	Max. :11673	Max. :6400	Max. :4850	Max. :89400	Max. :46800	Max. :1157.0	Max. :1122.0	Max. :6500.0	Max. :5100000

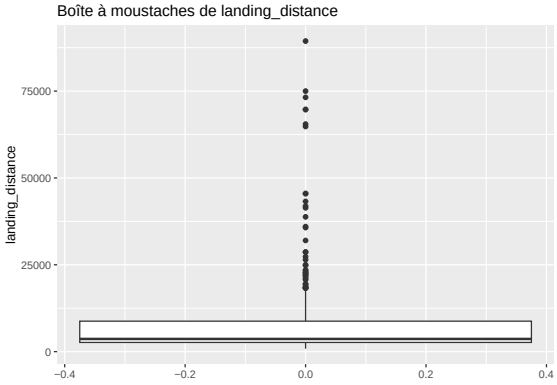


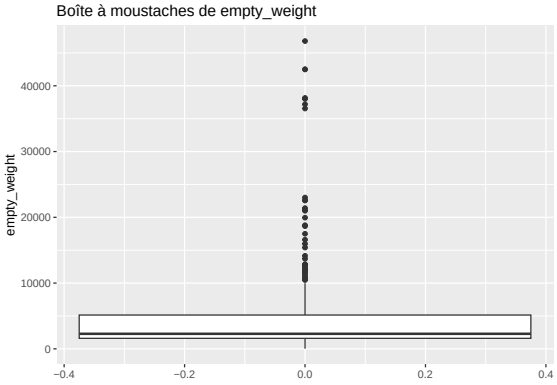

Boîte à moustaches de fuel_tank
fuel_tank
40000
30000
20000
10000
0
−0.4
−0.2
0.0
0.2
0.4

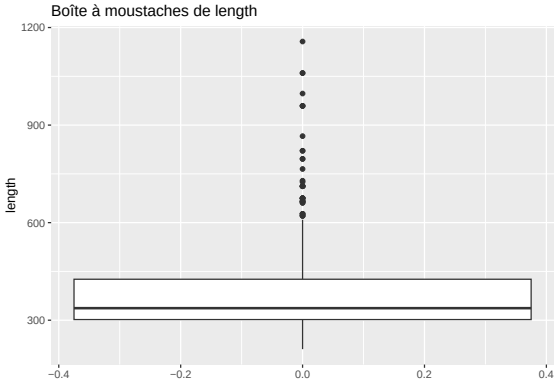

Boîte à moustaches de all_eng_roc
all_eng_roc
12000
9000
6000
3000
0
−0.4
−0.2
0.0
0.2
0.4

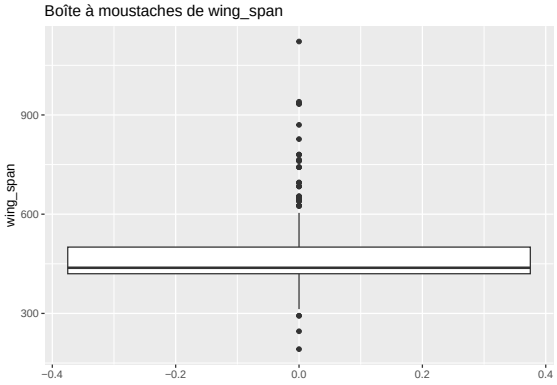

Boîte à moustaches de out_eng_roc
out_eng_roc
6000
4000
2000
−0.4
−0.2
0.0
0.2
0.4

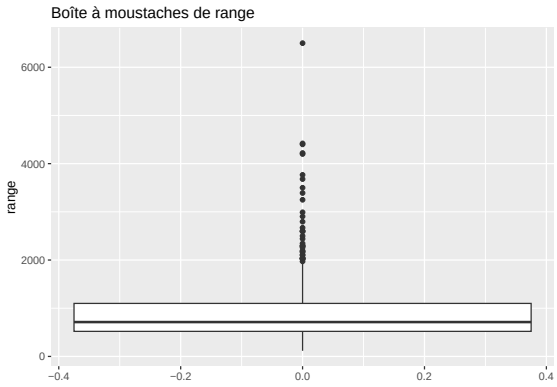

Boîte à moustaches de takeoff_distance
takeoff_distance
5000
4000
3000
2000
1000
−0.4
−0.2
0.0
0.2
0.4

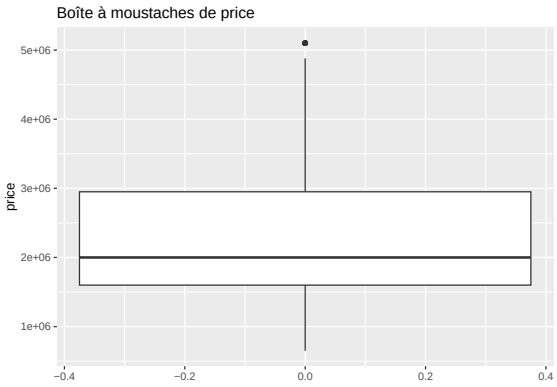

Boîte à moustaches de landing_distance
landing_distance
75000
50000
25000
0
−0.4
−0.2
0.0
0.2
0.4


Boîte à moustaches de empty_weight
empty_weight
40000
30000
20000
10000
0
−0.4
−0.2
0.0
0.2
0.4


Boîte à moustaches de length
length
1200
900
600
300
−0.4
−0.2
0.0
0.2
0.4


Boîte à moustaches de wing_span
wing_span
900
600
300
−0.4
−0.2
0.0
0.2
0.4

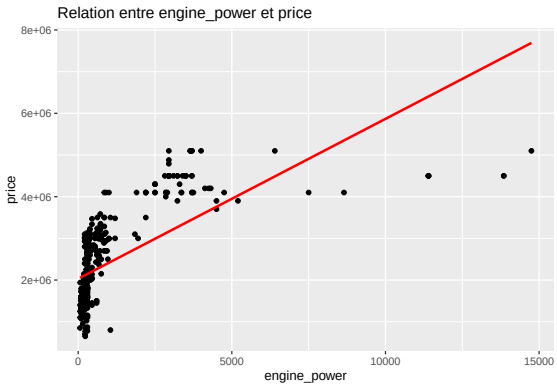


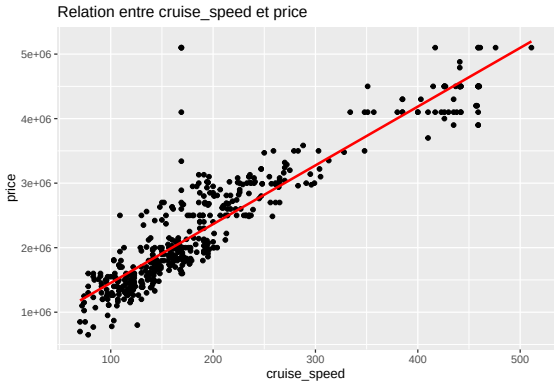


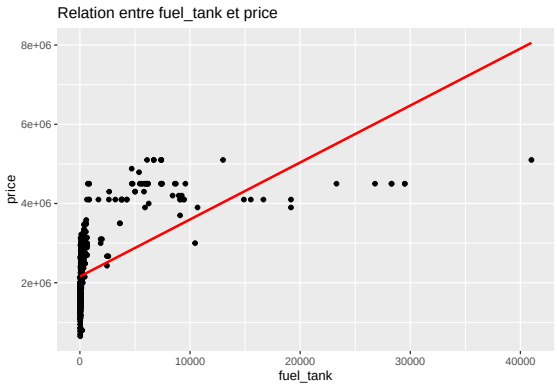

```r
# Calcul des quartiles et de l'IQR
Q1 = quantile(data$price, 0.25)
Q3 = quantile(data$price, 0.75)
IQR_val = IQR(data$price)

# Limites pour les anomalies
lower_bound = Q1 - 1.5 * IQR_val
upper_bound = Q3 + 1.5 * IQR_val

# Id les anomalies
anomalies = data[data$price < lower_bound | data$price > upper_bound, ]
anomalies
```

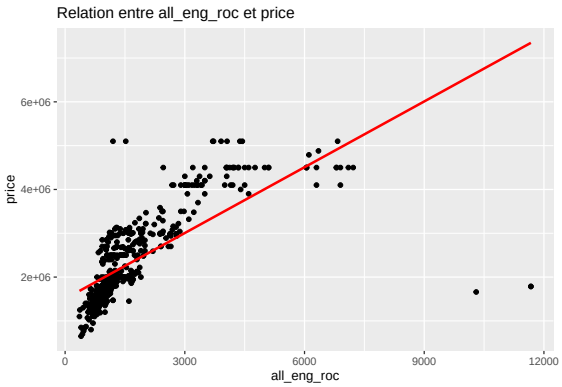
```
##                                model_name engine_type engine_power max_speed
## 153                  402C Business Liner II         Jet         3700       455
## 154                    402,-A turbocharged          Jet         3700       457
## 163 T 337 G-P II,H-P, Skymaster pressurized         Jet         2950       464
## 175                          208 Caravan-675         Jet         6400        92
## 176            100 Darter (S.L. Industries)          Jet         4000        85
## 177            100 Darter (S.L. Industries)          Jet         3650       473
## 178    T 210 K,L Turbo (K=9 mph less speed)          Jet         3650       472
## 179            100 Darter (S.L. Industries)          Jet         3650       472
## 423             PA-30 B Turbo Twin Comanche          Jet        14750       516
##     cruise_speed stall_speed fuel_tank all_eng_roc out_eng_roc takeoff_distance
## 153          417          56      6707        4059        5600             3300
## 154          169          56      6707        4380        4540             3109
## 163          459          97      6098        6830        3937             2817
## 175          511          56     13000        3720        5710             3820
## 176          476          97      7385        4442        4690             2910
## 177          169          97      7385        1520        5150             2900
## 178          461          97      7384        3699        5030             2900
## 179          169          89      7384        3909        4710             2560
## 423          459          56     41000        1200        5870             2950
##     landing_distance empty_weight length wing_span range   price
## 153            21500        12135    661       525  1715 5100000
## 154            19500        12130    661       525  1715 5100000
## 163            15000         7950    571       427  1431 5100000
## 175            35700        21400    866       765  3250 5100000
## 176            22450        13700    665       641  2300 5100000
## 177            22000        12775    665       641  2345 5100000
```

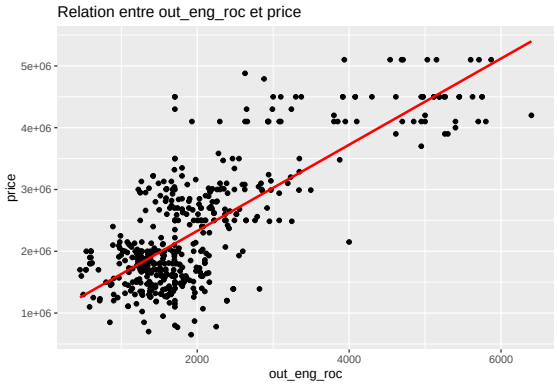
```
## 178          22000          11811     666        642  2600 5100000
## 179          21000          11720     666        642  2600 5100000
## 423          89400          46800    1157       1122  6500 5100000
```

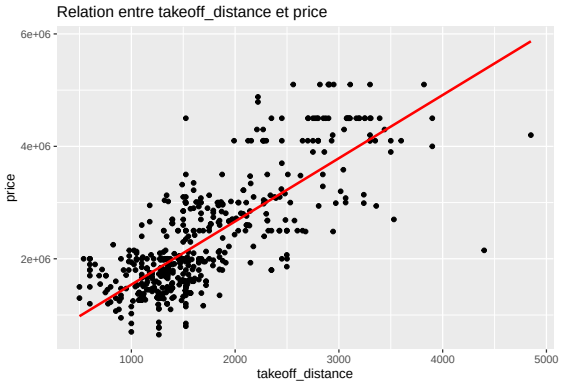












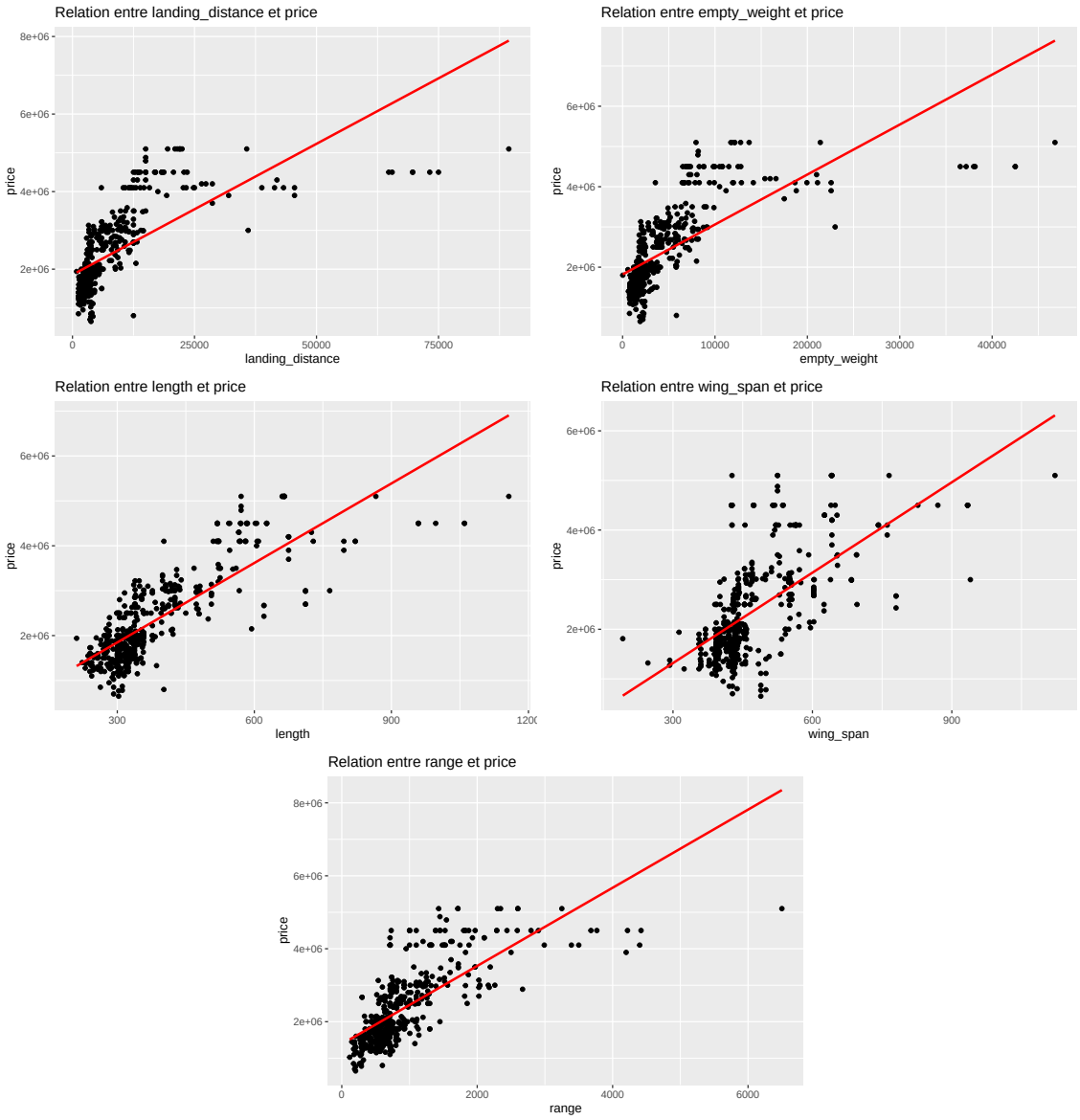


Table 2: Matrice des corrélations

	engn_	mx_sp	crs_s	stll_	fl_tn	all___	ot_n_	tkff_	lndn_	empt_	lngth	wng_s	range	price
engine_power	1.00	0.70	0.73	0.56	0.96	0.53	0.76	0.61	0.96	0.95	0.86	0.72	0.79	0.67
max_speed	0.70	1.00	0.87	0.76	0.65	0.71	0.76	0.74	0.72	0.72	0.79	0.56	0.71	0.85
cruise_speed	0.73	0.87	1.00	0.80	0.67	0.76	0.81	0.80	0.74	0.74	0.82	0.59	0.74	0.92
stall_speed	0.56	0.76	0.80	1.00	0.49	0.63	0.71	0.79	0.63	0.65	0.76	0.59	0.64	0.79
fuel_tank	0.96	0.65	0.67	0.49	1.00	0.44	0.73	0.55	0.95	0.94	0.83	0.73	0.80	0.61
all_eng_roc	0.53	0.71	0.76	0.63	0.44	1.00	0.61	0.60	0.50	0.50	0.60	0.35	0.49	0.72
out_eng_roc	0.76	0.76	0.81	0.71	0.73	0.61	1.00	0.83	0.76	0.76	0.83	0.60	0.73	0.77
takeoff_distance	0.61	0.74	0.80	0.79	0.55	0.60	0.83	1.00	0.65	0.66	0.77	0.61	0.68	0.78
landing_distance	0.96	0.72	0.74	0.63	0.95	0.50	0.76	0.65	1.00	1.00	0.92	0.83	0.84	0.69
empty_weight	0.95	0.72	0.74	0.65	0.94	0.50	0.76	0.66	1.00	1.00	0.92	0.85	0.84	0.69
length	0.86	0.79	0.82	0.76	0.83	0.60	0.83	0.77	0.92	0.92	1.00	0.87	0.85	0.81
wing_span	0.72	0.56	0.59	0.59	0.73	0.35	0.60	0.61	0.83	0.85	0.87	1.00	0.75	0.61

	engn_	mx_spcrs_s	stll_	fl_tn	all___	ot_n_tkff_	lndn_	empt_lngth	wng_srange	price
range	0.79	0.71	0.74	0.64	0.80	0.49	0.73	0.68	0.84	0.84	0.85	0.75	1.00	0.73
price	0.67	0.85	0.92	0.79	0.61	0.72	0.77	0.78	0.69	0.69	0.81	0.61	0.73	1.00

```r
# Suppression de la colonne 'model_name'
data = subset(data, select = -c(model_name))

# Convertion de 'engine_type' en facteur
data$engine_type = as.factor(data$engine_type)

# Vérification de la structure
str(data)
```

```
## 'data.frame':    507 obs. of  15 variables:
##  $ engine_type     : Factor w/ 3 levels "Jet","Piston",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ engine_power    : num  145 85 90 85 65 65 350 290 290 290 ...
##  $ max_speed       : num  104 89 90 88 83 78 264 262 257 257 ...
##  $ cruise_speed    : num  91 83 78 78 74 72 230 247 235 237 ...
##  $ stall_speed     : num  46 44 37 37 33 33 80 77 77 77 ...
##  $ fuel_tank       : num  36 15 19 19 14 15 165 165 165 165 ...
##  $ all_eng_roc     : num  450 600 650 620 370 ...
##  $ out_eng_roc     : num  900 720 475 500 632 583 3080 2250 2490 2490 ...
##  $ takeoff_distance: num  1300 800 850 850 885 ...
##  $ landing_distance: num  2050 1350 1300 1300 1220 ...
##  $ empty_weight    : num  1180 820 810 800 740 ...
##  $ length          : num  303 247 257 257 257 244 418 417 418 418 ...
##  $ wing_span       : num  449 433 420 420 420 433 440 439 440 440 ...
##  $ range           : num  370 190 210 210 175 ...
##  $ price           : num  1300000 1230000 1600000 1300000 1250000 1100000 2500000 2800000 2500000 30
```

```r
#Création des échantillons d'apprentissage et de test
set.seed(1)
indxdata = createDataPartition(data$price, p = 0.75, list = FALSE)
datatrain = data[indxdata, ]
datatest = data[-indxdata, ]
```

```r
#Élimination descendante manuelle basée sur les p-values

# Modèle initial avec toutes les variables
lm_manuel = lm(price ~ ., data = datatrain)
summary(lm_manuel)
```

```
##
## Call:
## lm(formula = price ~ ., data = datatrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -965729 -224122  -54097  180861 1678536
##
## Coefficients:
```

```
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -15215.59  267104.25  -0.057 0.954604
## engine_typePiston  -390758.36  142745.54  -2.737 0.006494 **
## engine_typePropjet -393371.20  142938.54  -2.752 0.006218 **
## engine_power            33.55      57.07   0.588 0.557033
## max_speed             2439.52     372.69   6.546 2.00e-10 ***
## cruise_speed          5255.42     540.82   9.718  < 2e-16 ***
## stall_speed           -624.13    2264.17  -0.276 0.782970
## fuel_tank               16.98      20.71   0.820 0.412810
## all_eng_roc             17.09      19.95   0.857 0.392069
## out_eng_roc            -64.68      39.13  -1.653 0.099206 .
## takeoff_distance        89.07      56.27   1.583 0.114343
## landing_distance      -100.70      20.85  -4.829 2.02e-06 ***
## empty_weight            96.79      34.56   2.801 0.005371 **
## length                1202.62     557.12   2.159 0.031529 *
## wing_span             1721.83     469.05   3.671 0.000278 ***
## range                  153.46      56.61   2.711 0.007023 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 337400 on 366 degrees of freedom
## Multiple R-squared:  0.893,  Adjusted R-squared:  0.8887
## F-statistic: 203.7 on 15 and 366 DF,  p-value: < 2.2e-16
```

```r
# Boucle d'élimination
while(TRUE) {
  p_values = summary(lm_manuel)$coefficients[-1, 4]
  if (all(p_values <= 0.05)) break
  var_to_remove = names(which.max(p_values))
  formula_new = as.formula(paste("price ~ . -", var_to_remove))
  lm_manuel = update(lm_manuel, formula_new)
}

# Modèle final après élimination
summary(lm_manuel)
```

```
##
## Call:
## lm(formula = price ~ engine_type + max_speed + cruise_speed +
##     landing_distance + empty_weight + wing_span + range, data = datatrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1059656  -220273   -48436   172877  1823928
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         200049.78  218994.26   0.913 0.361573
## engine_typePiston  -527361.98  115335.41  -4.572 6.57e-06 ***
## engine_typePropjet -500882.18  109095.16  -4.591 6.03e-06 ***
## max_speed             2543.78     352.53   7.216 3.02e-12 ***
## cruise_speed          5485.21     471.56  11.632  < 2e-16 ***
## landing_distance       -89.84      18.17  -4.946 1.15e-06 ***
## empty_weight           104.69      33.23   3.150 0.001762 **
```

```
## wing_span             2185.14     362.33   6.031 3.93e-09 ***
## range                  187.39      51.06   3.670 0.000278 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 338300 on 373 degrees of freedom
## Multiple R-squared:  0.8904, Adjusted R-squared:  0.888
## F-statistic: 378.7 on 8 and 373 DF,  p-value: < 2.2e-16
```

```
#Sélection automatique avec AIC descendante

mod_aic_desc = step(lm_manuel, trace = TRUE)
```

```
## Start:  AIC=9735.94
## price ~ engine_type + max_speed + cruise_speed + landing_distance +
##     empty_weight + wing_span + range
##
##                    Df  Sum of Sq        RSS    AIC
## <none>                           4.2692e+13 9735.9
## - empty_weight      1 1.1360e+12 4.3828e+13 9744.0
## - range             1 1.5415e+12 4.4234e+13 9747.5
## - engine_type       2 2.6828e+12 4.5375e+13 9755.2
## - landing_distance  1 2.7995e+12 4.5492e+13 9758.2
## - wing_span         1 4.1630e+12 4.6855e+13 9769.5
## - max_speed         1 5.9596e+12 4.8652e+13 9783.9
## - cruise_speed      1 1.5487e+13 5.8179e+13 9852.2
```

```
summary(mod_aic_desc)
```

```
##
## Call:
## lm(formula = price ~ engine_type + max_speed + cruise_speed +
##     landing_distance + empty_weight + wing_span + range, data = datatrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1059656  -220273   -48436   172877  1823928
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         200049.78  218994.26   0.913 0.361573
## engine_typePiston  -527361.98  115335.41  -4.572 6.57e-06 ***
## engine_typePropjet -500882.18  109095.16  -4.591 6.03e-06 ***
## max_speed             2543.78     352.53   7.216 3.02e-12 ***
## cruise_speed          5485.21     471.56  11.632  < 2e-16 ***
## landing_distance       -89.84      18.17  -4.946 1.15e-06 ***
## empty_weight           104.69      33.23   3.150 0.001762 **
## wing_span             2185.14     362.33   6.031 3.93e-09 ***
## range                  187.39      51.06   3.670 0.000278 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 338300 on 373 degrees of freedom
```

```
## Multiple R-squared:  0.8904, Adjusted R-squared:  0.888
## F-statistic: 378.7 on 8 and 373 DF,  p-value: < 2.2e-16
```

```r
#Sélection automatique avec AIC Ascendante

mod_aic_asc = lm(price ~ 1, data = datatrain)
step(mod_aic_asc, scope = formula(lm_manuel), direction = "forward", trace = TRUE)
```

```
## Start:  AIC=10564.43
## price ~ 1
##
##                     Df  Sum of Sq        RSS     AIC
## + cruise_speed       1 3.2902e+14 6.0431e+13  9854.7
## + max_speed          1 2.9947e+14 8.9980e+13 10006.7
## + engine_type        2 2.8221e+14 1.0724e+14 10075.8
## + range              1 2.0114e+14 1.8831e+14 10288.8
## + empty_weight       1 1.8487e+14 2.0458e+14 10320.5
## + landing_distance   1 1.8253e+14 2.0692e+14 10324.9
## + wing_span          1 1.3189e+14 2.5756e+14 10408.5
## <none>                            3.8945e+14 10564.4
##
## Step:  AIC=9854.68
## price ~ cruise_speed
##
##                     Df  Sum of Sq        RSS    AIC
## + max_speed          1 6.8171e+12 5.3614e+13 9811.0
## + engine_type        2 4.0091e+12 5.6422e+13 9832.5
## + wing_span          1 3.3234e+12 5.7108e+13 9835.1
## + range              1 2.2421e+12 5.8189e+13 9842.2
## <none>                            6.0431e+13 9854.7
## + empty_weight       1 2.8377e+11 6.0148e+13 9854.9
## + landing_distance   1 8.3941e+10 6.0347e+13 9856.1
##
## Step:  AIC=9810.95
## price ~ cruise_speed + max_speed
##
##                     Df  Sum of Sq        RSS    AIC
## + wing_span          1 2.3040e+12 5.1310e+13 9796.2
## + engine_type        2 2.3758e+12 5.1239e+13 9797.6
## + range              1 9.5091e+11 5.2663e+13 9806.1
## <none>                            5.3614e+13 9811.0
## + landing_distance   1 9.6061e+10 5.3518e+13 9812.3
## + empty_weight       1 2.3741e+09 5.3612e+13 9812.9
##
## Step:  AIC=9796.17
## price ~ cruise_speed + max_speed + wing_span
##
##                     Df  Sum of Sq        RSS    AIC
## + landing_distance   1 4.4344e+12 4.6876e+13 9763.6
## + empty_weight       1 3.5355e+12 4.7775e+13 9770.9
## + engine_type        2 1.1981e+12 5.0112e+13 9791.1
## <none>                            5.1310e+13 9796.2
## + range              1 1.6141e+10 5.1294e+13 9798.1
##
```

```
## Step:  AIC=9763.65
## price ~ cruise_speed + max_speed + wing_span + landing_distance
##
##                Df  Sum of Sq        RSS    AIC
## + engine_type   2 1.5922e+12 4.5284e+13 9754.4
## + range         1 1.0072e+12 4.5869e+13 9757.3
## + empty_weight  1 5.4894e+11 4.6327e+13 9761.1
## <none>                       4.6876e+13 9763.6
##
## Step:  AIC=9754.44
## price ~ cruise_speed + max_speed + wing_span + landing_distance +
##     engine_type
##
##                Df  Sum of Sq        RSS    AIC
## + range         1 1.4553e+12 4.3828e+13 9744.0
## + empty_weight  1 1.0498e+12 4.4234e+13 9747.5
## <none>                       4.5284e+13 9754.4
##
## Step:  AIC=9743.97
## price ~ cruise_speed + max_speed + wing_span + landing_distance +
##     engine_type + range
##
##                Df Sum of Sq        RSS    AIC
## + empty_weight  1 1.136e+12 4.2692e+13 9735.9
## <none>                      4.3828e+13 9744.0
##
## Step:  AIC=9735.94
## price ~ cruise_speed + max_speed + wing_span + landing_distance +
##     engine_type + range + empty_weight
```

```
##
## Call:
## lm(formula = price ~ cruise_speed + max_speed + wing_span + landing_distance +
##     engine_type + range + empty_weight, data = datatrain)
##
## Coefficients:
##        (Intercept)        cruise_speed           max_speed           wing_span
##          200049.78             5485.21             2543.78             2185.14
##   landing_distance   engine_typePiston  engine_typePropjet               range
##              -89.84          -527361.98          -500882.18              187.39
##       empty_weight
##              104.69
```

```
#Modèle AIC Ascendant
mod_aic_asc = lm(price ~ cruise_speed + max_speed + wing_span + landing_distance + range + empty_weight
    data = datatrain)
summary(mod_aic_asc)
```

```
##
## Call:
## lm(formula = price ~ cruise_speed + max_speed + wing_span + landing_distance +
##     range + empty_weight + engine_type, data = datatrain)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1059656  -220273   -48436   172877  1823928
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         200049.78  218994.26   0.913 0.361573
## cruise_speed          5485.21     471.56  11.632  < 2e-16 ***
## max_speed             2543.78     352.53   7.216 3.02e-12 ***
## wing_span             2185.14     362.33   6.031 3.93e-09 ***
## landing_distance       -89.84      18.17  -4.946 1.15e-06 ***
## range                  187.39      51.06   3.670 0.000278 ***
## empty_weight           104.69      33.23   3.150 0.001762 **
## engine_typePiston  -527361.98  115335.41  -4.572 6.57e-06 ***
## engine_typePropjet -500882.18  109095.16  -4.591 6.03e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 338300 on 373 degrees of freedom
## Multiple R-squared:  0.8904, Adjusted R-squared:  0.888
## F-statistic: 378.7 on 8 and 373 DF,  p-value: < 2.2e-16
```

```r
#Sélection du meilleur modèle avec le critère Cp de Mallows

# Création d'une matrice de prédicteurs
model_matrix = model.matrix(price ~ ., data = data)[, -1]  # Sans intercept

# Recherche des meilleurs modèles
data.choix = leaps(model_matrix, data$price, method = "Cp", nbest = 1)

# Meilleur modèle
best_model_idx = which.min(data.choix$Cp)
variables_incluses = colnames(model_matrix)[data.choix$which[best_model_idx, ]]
print(variables_incluses)
```

```
##  [1] "engine_typePiston"  "engine_typePropjet" "engine_power"
##  [4] "max_speed"          "cruise_speed"       "stall_speed"
##  [7] "landing_distance"   "empty_weight"       "length"
## [10] "wing_span"          "range"
```

```r
# Meilleur modèle Cp de Mallows
lm_mallows = lm(price ~ engine_type +  engine_power + max_speed + cruise_speed + stall_speed + landing_
summary(lm_mallows)
```

```
##
## Call:
## lm(formula = price ~ engine_type + engine_power + max_speed +
##     cruise_speed + stall_speed + landing_distance + empty_weight +
##     length + wing_span + range, data = datatrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1022970  -225796   -43998   188755  1740479
```

```
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -54525.53  262740.76  -0.208  0.83571
## engine_typePiston  -384929.26  138145.94  -2.786  0.00560 **
## engine_typePropjet -391068.50  135884.75  -2.878  0.00424 **
## engine_power            46.27      51.90   0.892  0.37322
## max_speed             2491.24     367.65   6.776 4.88e-11 ***
## cruise_speed          5436.17     521.40  10.426  < 2e-16 ***
## stall_speed           -204.11    2099.50  -0.097  0.92261
## landing_distance       -97.12      20.32  -4.780 2.53e-06 ***
## empty_weight            95.74      34.54   2.772  0.00585 **
## length                1008.64     530.57   1.901  0.05807 .
## wing_span             1870.41     455.74   4.104 5.00e-05 ***
## range                  163.72      52.45   3.121  0.00194 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 337500 on 370 degrees of freedom
## Multiple R-squared:  0.8918, Adjusted R-squared:  0.8885
## F-statistic: 277.1 on 11 and 370 DF,  p-value: < 2.2e-16
```

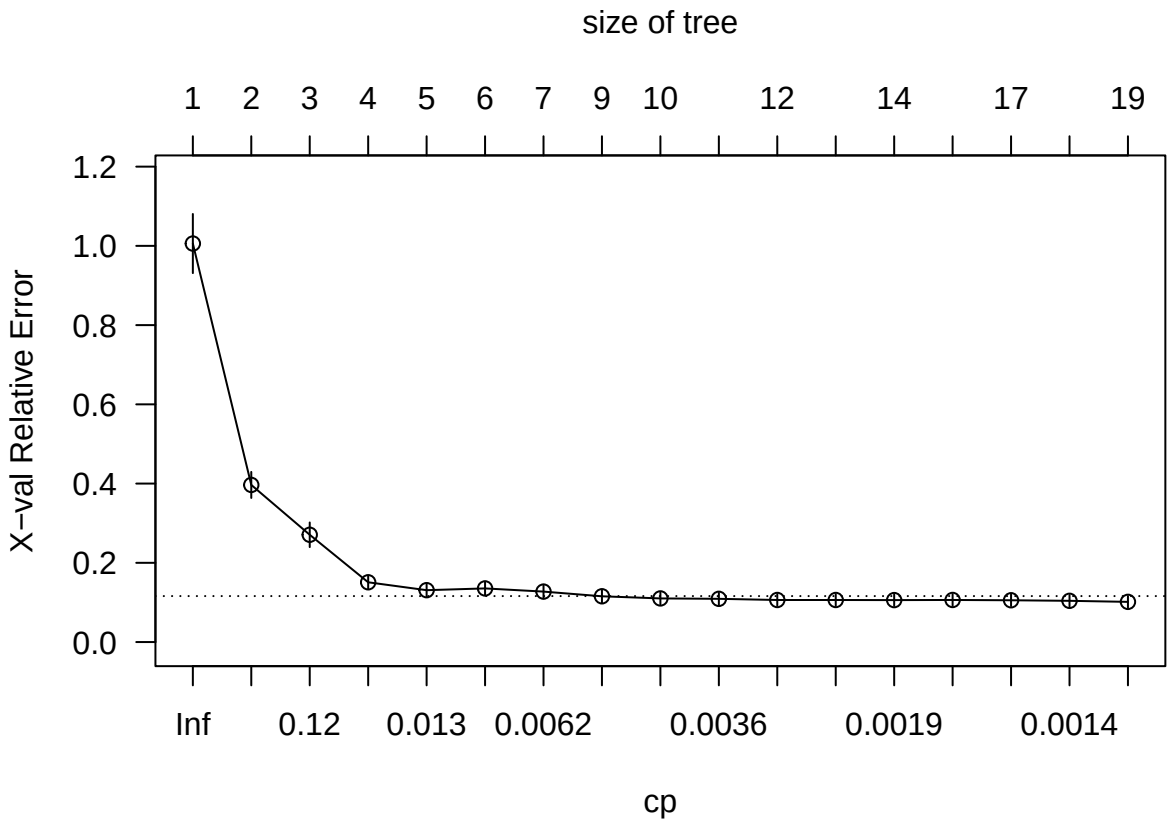
```r
# Préparation des données
X_train = model.matrix(price ~ . -1, data = datatrain)
y_train = datatrain$price
X_test = model.matrix(price ~ . -1, data = datatest)

# Entraînement des modèles avec validation croisée
set.seed(123)

# Lasso
cv_lasso = cv.glmnet(X_train, y_train, alpha = 1)
lambda_lasso = cv_lasso$lambda.min

# Ridge
cv_ridge = cv.glmnet(X_train, y_train, alpha = 0)
lambda_ridge = cv_ridge$lambda.min
```

```r
#Tree

# Création de l'arbre initial avec toutes les variables

set.seed(1)
tree = rpart(price ~ .,
             data = datatrain,
             method = "anova",
             control = rpart.control(
               cp = 0.001,
               minsplit = 20,
               minbucket = 10,
               xval = 10
             ))

# Analyse de la complexité optimale
```

```r
# Courbe d'erreur CV
printcp(tree)
```

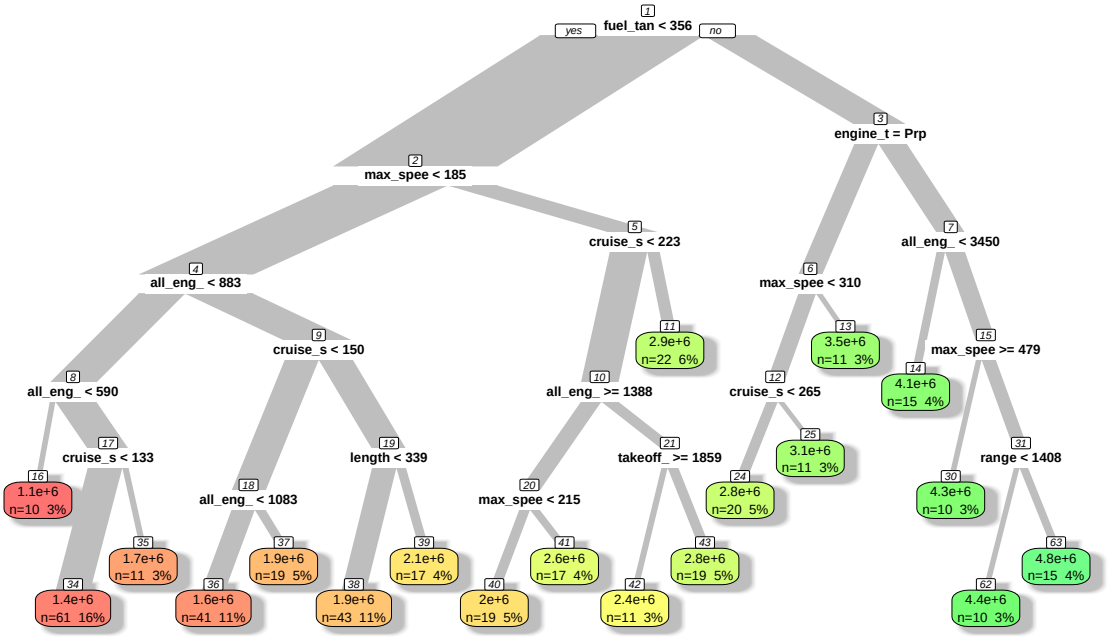
```
##
## Regression tree:
## rpart(formula = price ~ ., data = datatrain, method = "anova",
##     control = rpart.control(cp = 0.001, minsplit = 20, minbucket = 10,
##         xval = 10))
##
## Variables actually used in tree construction:
## [1] all_eng_roc      cruise_speed     engine_type      fuel_tank
## [5] length           max_speed        range            takeoff_distance
##
## Root node error: 3.8945e+14/382 = 1.0195e+12
##
## n= 382
##
##           CP nsplit rel error  xerror     xstd
## 1  0.6349976      0  1.000000 1.00588 0.074635
## 2  0.1292592      1  0.365002 0.39656 0.032898
## 3  0.1071935      2  0.235743 0.27076 0.031115
## 4  0.0249910      3  0.128550 0.15087 0.017296
## 5  0.0071477      4  0.103559 0.13096 0.017267
## 6  0.0064392      5  0.096411 0.13536 0.017394
## 7  0.0060314      6  0.089972 0.12737 0.017344
## 8  0.0050838      8  0.077909 0.11553 0.017382
## 9  0.0043076      9  0.072825 0.11012 0.015262
## 10 0.0029977     10  0.068518 0.10914 0.015066
## 11 0.0019867     11  0.065520 0.10616 0.014926
## 12 0.0018826     12  0.063533 0.10615 0.014784
## 13 0.0018422     13  0.061651 0.10587 0.014787
## 14 0.0016775     14  0.059808 0.10620 0.014759
## 15 0.0015765     16  0.056454 0.10545 0.014733
## 16 0.0013063     17  0.054877 0.10408 0.014611
## 17 0.0010000     18  0.053571 0.10146 0.014557
```

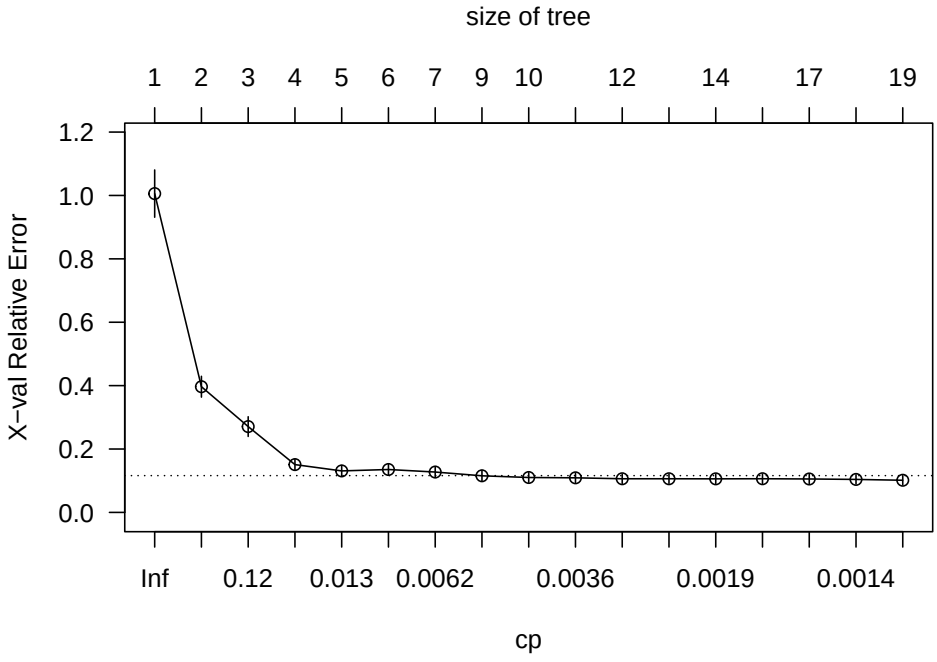
```r
plotcp(tree, las = 1)
```



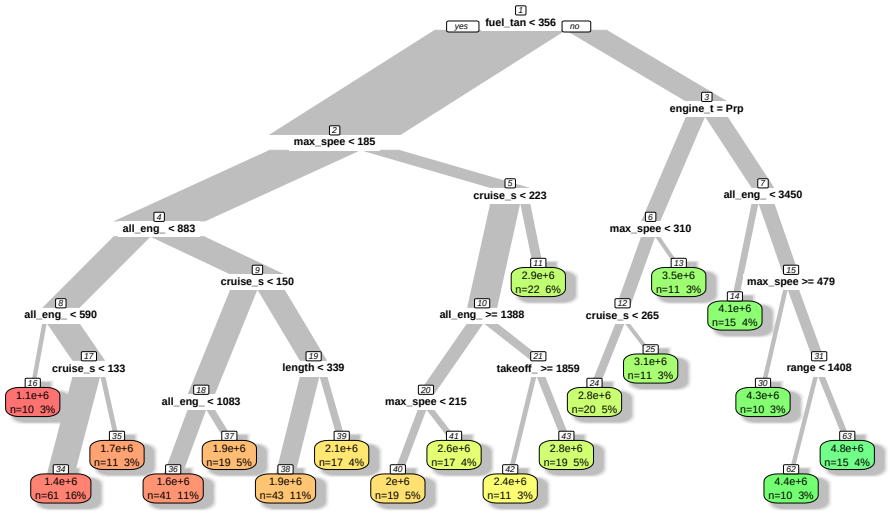

```r
# Sélection du cp optimal
cp_table = as.data.frame(tree$cptable)
cp.opt = cp_table[which.min(cp_table$xerror), "CP"]


# Élagage de l'arbre optimal

tree.opt = prune(tree, cp = cp.opt)

# Visualisation de l'arbre
prp(tree.opt,
    branch.type = 5,
    box.palette = "RdYlGn",
    shadow.col = "gray",
    nn = TRUE,
    extra = 101,
    main = "Arbre de régression optimal")
```

Arbre de régression optimal





Arbre de régression optimal




```
# Random Forest

# Création du modèle Random Forest initial

rf_model = randomForest(price ~ .,
                        data = datatrain,
                        ntree = 500,
                        mtry = sqrt(ncol(datatrain) - 1),
                        importance = TRUE,
                        nodesize = 5,
                        replace = TRUE)


# Validation croisée pour déterminer le nombre optimal d'arbres

set.seed(1)#123
control = trainControl(method = "cv", number = 10)
tuning_grid = expand.grid(.mtry = sqrt(ncol(datatrain) - 1))

rf_tuned = train(price ~ .,
                 data = datatrain,
                 method = "rf",
                 trControl = control,
                 tuneGrid = tuning_grid,
                 ntree = 500)


# Nombre optimal d'arbres
ntree_optimal = which.min(rf_tuned$results$RMSE)

# Réentraînement avec le nombre optimal d'arbres
```

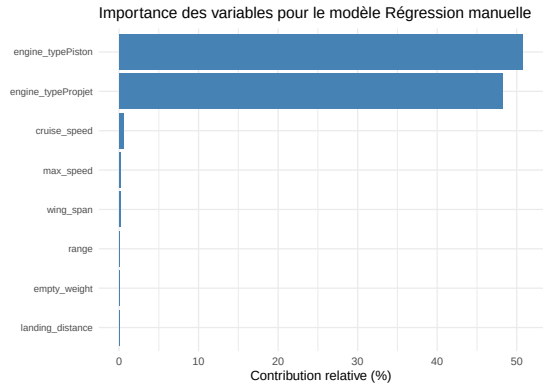
```r
rf_optimal = randomForest(price ~ .,
                          data = datatrain,
                          ntree = ntree_optimal,
                          mtry = sqrt(ncol(datatrain) - 1),
                          importance = TRUE,
                          nodesize = 5,
                          replace = TRUE)
```

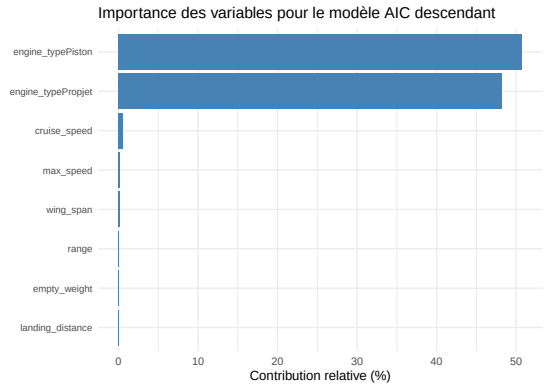
Table 3: Comparaison des performances des modèles traditionnels avec IC Bootstrap

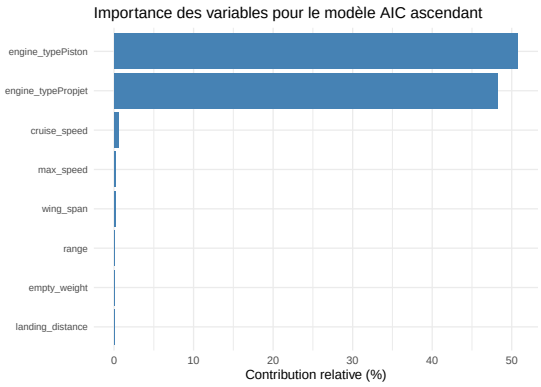
Modèle	RMSE Bootstrap	RMSE IC Inf	RMSE IC Sup	R^2 Bootstrap	R^2 IC Inf	R^2 IC Sup
AIC Ascendant	403593.0	335489.9	470763.3	0.85	0.80	0.886
AIC Descendant	405147.9	339334.1	472916.7	0.85	0.80	0.891
Arbre de Régression	331947.5	284290.6	383671.6	0.90	0.85	0.931
Mallows Cp	393699.6	330893.0	463650.8	0.85	0.81	0.896
Manuelle (p-values)	403168.9	337731.5	469132.4	0.85	0.80	0.889
Random Forest	350046.6	284567.9	417059.9	0.88	0.82	0.929

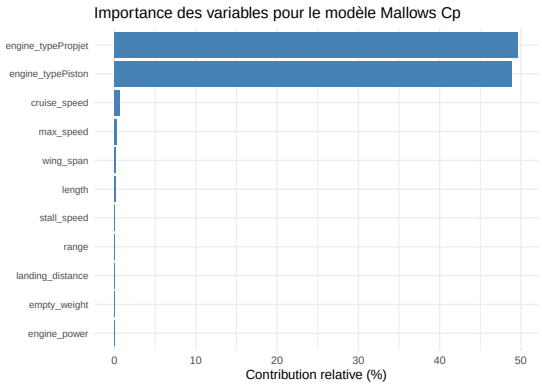
Table 4: Comparaison des performances des modèles régularisés avec IC Bootstrap

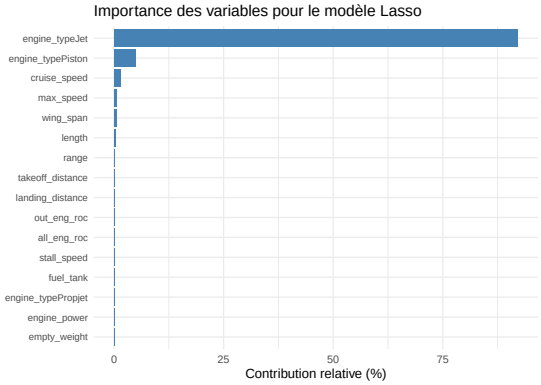
Modèle	RMSE Bootstrap	IC Inf	IC Sup	R^2 Bootstrap	IC Inf	IC Sup
Lasso	392071.1	330073.3	467608.7	0.86	0.81	0.895
Ridge	385395.2	333139.7	442208.2	0.86	0.82	0.896

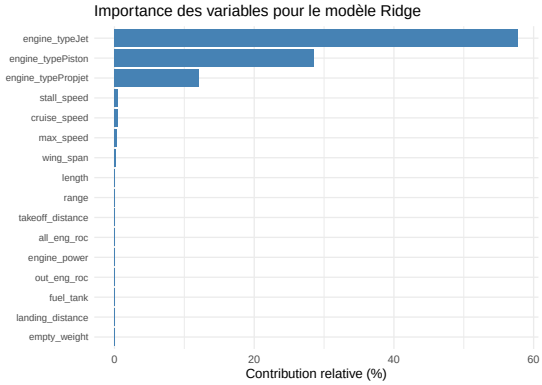


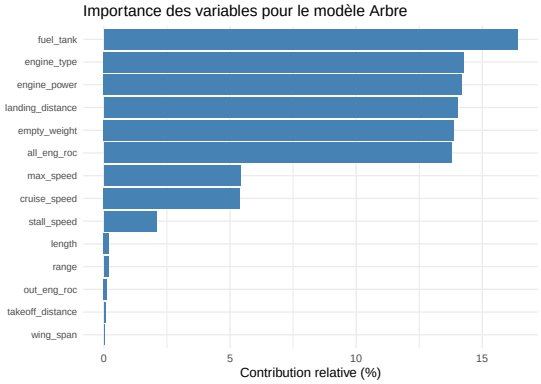


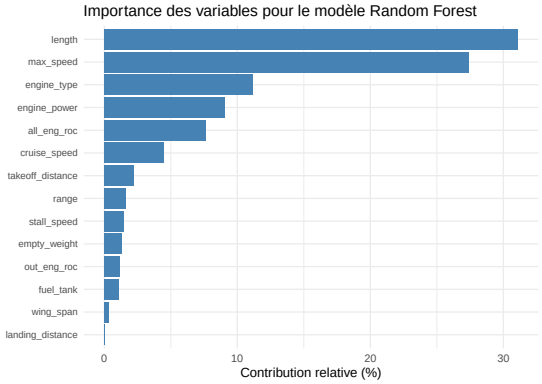

Importance des variables pour le modèle AIC ascendant
engine_typePiston
engine_typePropjet
cruise_speed
max_speed
wing_span
range
empty_weight
landing_distance
0
10
20
30
40
50
Contribution relative (%)


Importance des variables pour le modèle Mallows Cp
engine_typePropjet
engine_typePiston
cruise_speed
max_speed
wing_span
length
stall_speed
range
landing_distance
empty_weight
engine_power
0
10
20
30
40
50
Contribution relative (%)


Importance des variables pour le modèle Lasso
engine_typeJet
engine_typePiston
cruise_speed
max_speed
wing_span
length
range
takeoff_distance
landing_distance
out_eng_roc
all_eng_roc
stall_speed
fuel_tank
engine_typePropjet
engine_power
empty_weight
0
25
50
75
Contribution relative (%)


Importance des variables pour le modèle Ridge
engine_typeJet
engine_typePiston
engine_typePropjet
stall_speed
cruise_speed
max_speed
wing_span
length
range
takeoff_distance
all_eng_roc
engine_power
out_eng_roc
fuel_tank
landing_distance
empty_weight
0
20
40
60
Contribution relative (%)


Importance des variables pour le modèle Arbre
fuel_tank
engine_type
engine_power
landing_distance
empty_weight
all_eng_roc
max_speed
cruise_speed
stall_speed
length
range
out_eng_roc
takeoff_distance
wing_span
0
5
10
15
Contribution relative (%)


Importance des variables pour le modèle Random Forest
length
max_speed
engine_type
engine_power
all_eng_roc
cruise_speed
takeoff_distance
range
stall_speed
empty_weight
out_eng_roc
fuel_tank
wing_span
landing_distance
0
10
20
30
Contribution relative (%)