ACT6100 - TP

Khechman Ayman , Bogdan Vasile,Chan Edmen ,Berjaoui Karim, Bilke Ram

2020-04-26

0.1 Installer les packages nécessaires

0.2 Charger les packages

Partie 1 : Statistique descriptive

On télécharge la base de données

```
data <- read_csv("https://raw.githubusercontent.com/nmeraihi/ACT6100/8f29c17d9ddd25713585fad6d1dc2296a0
  filter(!is.na(nb_sinistre))
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   name = col_character(),
##   id = col_character()
## )

## See spec(...) for full column specifications.
```

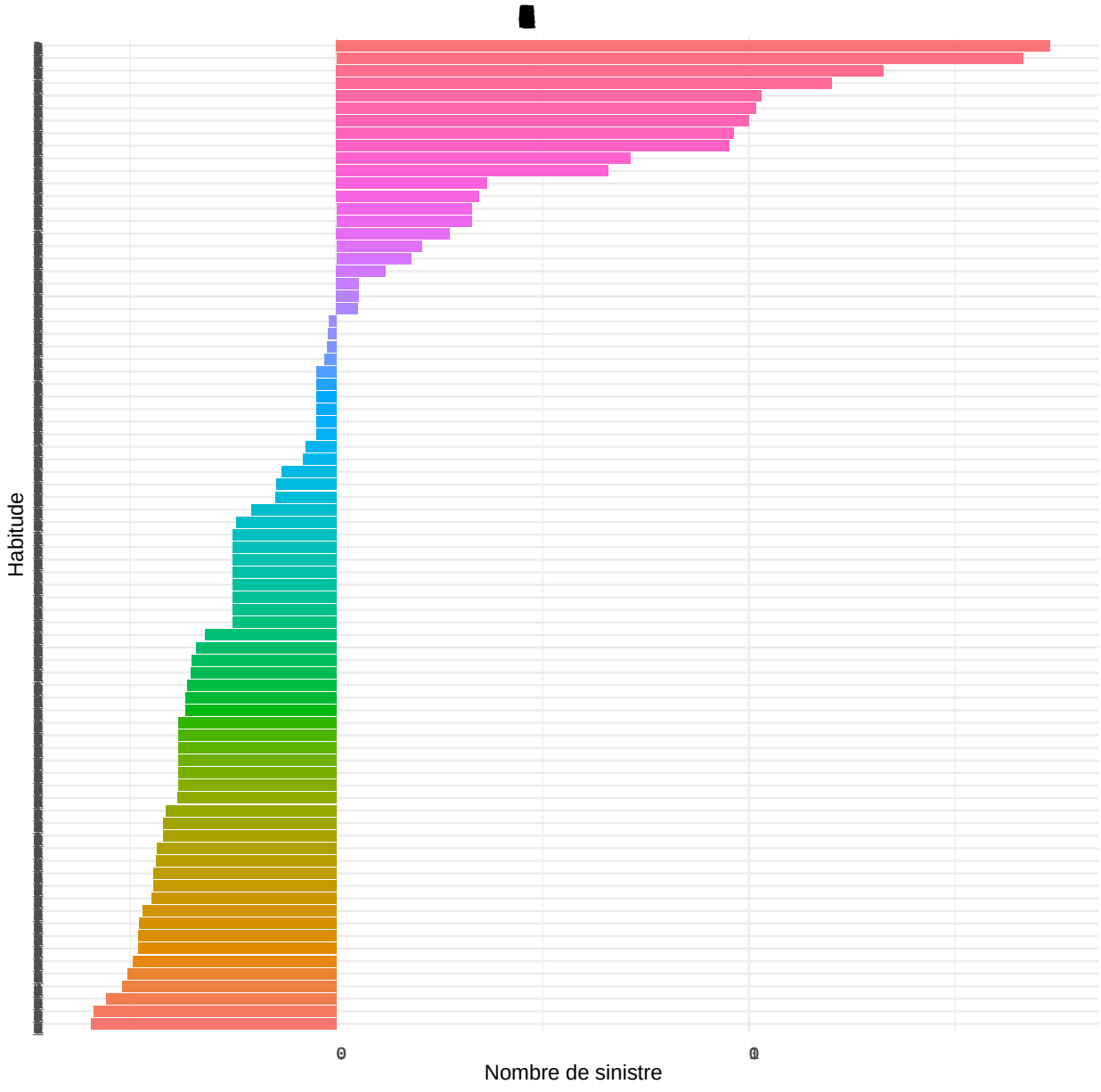
```
liste <- names(data)[str_detect(names(data),"variable")]
```

1.1 Corrélation entre nos variables

Vérifions la corrélation entre la variable habitudes et le nombre de sinistres.

```
data %>%
  select(nb_sinistre,starts_with("variable"))%>%
  correlate() %>%
  filter(!is.na(nb_sinistre))%>%
  select(nb_sinistre,rowname)%>%
  mutate(rowname = fct_reorder(rowname,nb_sinistre))%>%
  ggplot(mapping = aes(x = rowname , y = nb_sinistre,fill = rowname))+
  geom_col()+
  coord_flip()+
  theme(legend.position = "",
        plot.title = element_text(size = 16 , family = "Arial Nova Light",hjust = 0.45),
        axis.text = element_text(family = "Arial Nova Cond Light"))+
  labs(title = "Corrélation entre le NB de sinistres et les habitudes des conducteurs",x = "Habitude" ,
```

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```



1.1.2 Variables ayant une corrélation de 0

```
data %>%
  select(nb_sinistre,starts_with("variable"))%>%
  correlate() %>%
  filter(!is.na(nb_sinistre))%>%
  pull(rowname) ->liste_correle
```

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

```
liste_correle <- liste_correle[-1]
liste
```

```
##  [1] "variable_1"  "variable_2"  "variable_3"  "variable_4"  "variable_5"
##  [6] "variable_6"  "variable_7"  "variable_8"  "variable_9"  "variable_10"
## [11] "variable_11" "variable_12" "variable_13" "variable_14" "variable_15"
## [16] "variable_16" "variable_17" "variable_18" "variable_19" "variable_20"
## [21] "variable_21" "variable_22" "variable_23" "variable_24" "variable_25"
## [26] "variable_26" "variable_27" "variable_28" "variable_29" "variable_30"
## [31] "variable_31" "variable_32" "variable_33" "variable_34" "variable_35"
## [36] "variable_36" "variable_37" "variable_38" "variable_39" "variable_40"
## [41] "variable_41" "variable_42" "variable_43" "variable_44" "variable_45"
## [46] "variable_46" "variable_47" "variable_48" "variable_49" "variable_50"
## [51] "variable_51" "variable_52" "variable_53" "variable_54" "variable_55"
## [56] "variable_56" "variable_57" "variable_58" "variable_59" "variable_60"
## [61] "variable_61" "variable_62" "variable_63" "variable_64" "variable_65"
## [66] "variable_66" "variable_67" "variable_68" "variable_69" "variable_70"
## [71] "variable_71" "variable_72" "variable_73" "variable_74" "variable_75"
## [76] "variable_76" "variable_77" "variable_78" "variable_79" "variable_80"
## [81] "variable_81" "variable_82" "variable_83" "variable_84" "variable_85"
## [86] "variable_86" "variable_87" "variable_88" "variable_89" "variable_90"
## [91] "variable_91" "variable_92" "variable_93" "variable_94" "variable_95"
## [96] "variable_96" "variable_97"
```

```
variables_correlations_nulle <-liste[!liste %in% liste_correle]
```

1.2 Base de données finale

Voici notre base de donnée finale sans les NA et nous avons supprimé les variables avec une corrélation de 0.

```
data <- read_csv("https://raw.githubusercontent.com/nmeraihi/ACT6100/8f29c17d9ddd25713585fad6d1dc2296a0
  filter(!is.na(nb_sinistre))%>%
  select(-variables_correlations_nulle)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   name = col_character(),
##   id = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(variables_correlations_nulle)` instead of `variables_correlations_nulle` to silence thi
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

1.3 Vérifier les doublons au niveau des noms

```
data %>%
  count(name,sort = TRUE)%>%
  filter(n>1)
```

```
## # A tibble: 627 x 2
##    name                  n
##    <chr>             <int>
##  1 Salvador Blackwell     3
##  2 Sid Wyatt              3
##  3 Abel Tate              2
##  4 Adaline Parrish        2
##  5 Adam Hendrix           2
##  6 Adam Solis             2
##  7 Adolfo Richardson      2
##  8 Adolfo Stone           2
##  9 Adolph Gates           2
## 10 Agustin Brooks         2
## # ... with 617 more rows
```

```
data %>%
  filter(name =="Adaline Parrish")
```

```
## # A tibble: 2 x 100
##   name  id    statut_marital  sexe  zone code_territoire age_aconducteur
##   <chr> <chr>          <dbl> <dbl> <dbl>           <dbl>           <dbl>
## 1 Adal~ 96cd~              1     1     0              16              57
## 2 Adal~ b6b5~              3     1     0              20              22
## # ... with 93 more variables: type_vehicule <dbl>, marque_vehhicule <dbl>,
## #   model_vehicule <dbl>, age_vehicule <dbl>, code_vehicule <dbl>,
## #   annee_vehicule <dbl>, utilisation_vehicule <dbl>, nb_annees_permis <dbl>,
## #   nb_ann_sans_accidents <dbl>, taux_territoire <dbl>,
## #   exposition_en_jour <dbl>, exposition_temps <dbl>,
## #   nb_klms_declare_annee <dbl>, distance_conduite <dbl>, variable_2 <dbl>,
## #   variable_3 <dbl>, variable_4 <dbl>, variable_5 <dbl>, variable_6 <dbl>,
## #   variable_7 <dbl>, variable_8 <dbl>, variable_9 <dbl>, variable_10 <dbl>,
## #   variable_11 <dbl>, variable_12 <dbl>, variable_13 <dbl>, variable_14 <dbl>,
## #   variable_15 <dbl>, variable_16 <dbl>, variable_17 <dbl>, variable_18 <dbl>,
## #   variable_19 <dbl>, variable_20 <dbl>, variable_21 <dbl>, variable_22 <dbl>,
## #   variable_23 <dbl>, variable_24 <dbl>, variable_25 <dbl>, variable_26 <dbl>,
## #   variable_27 <dbl>, variable_28 <dbl>, variable_29 <dbl>, variable_30 <dbl>,
## #   variable_31 <dbl>, variable_32 <dbl>, variable_33 <dbl>, variable_34 <dbl>,
## #   variable_35 <dbl>, variable_36 <dbl>, variable_37 <dbl>, variable_38 <dbl>,
## #   variable_39 <dbl>, variable_40 <dbl>, variable_47 <dbl>, variable_48 <dbl>,
## #   variable_49 <dbl>, variable_50 <dbl>, variable_51 <dbl>, variable_52 <dbl>,
## #   variable_59 <dbl>, variable_60 <dbl>, variable_61 <dbl>, variable_62 <dbl>,
## #   variable_63 <dbl>, variable_64 <dbl>, variable_71 <dbl>, variable_72 <dbl>,
## #   variable_73 <dbl>, variable_74 <dbl>, variable_75 <dbl>, variable_76 <dbl>,
## #   variable_77 <dbl>, variable_78 <dbl>, variable_79 <dbl>, variable_80 <dbl>,
## #   variable_81 <dbl>, variable_82 <dbl>, variable_83 <dbl>, variable_84 <dbl>,
## #   variable_85 <dbl>, variable_86 <dbl>, variable_87 <dbl>, variable_88 <dbl>,
```

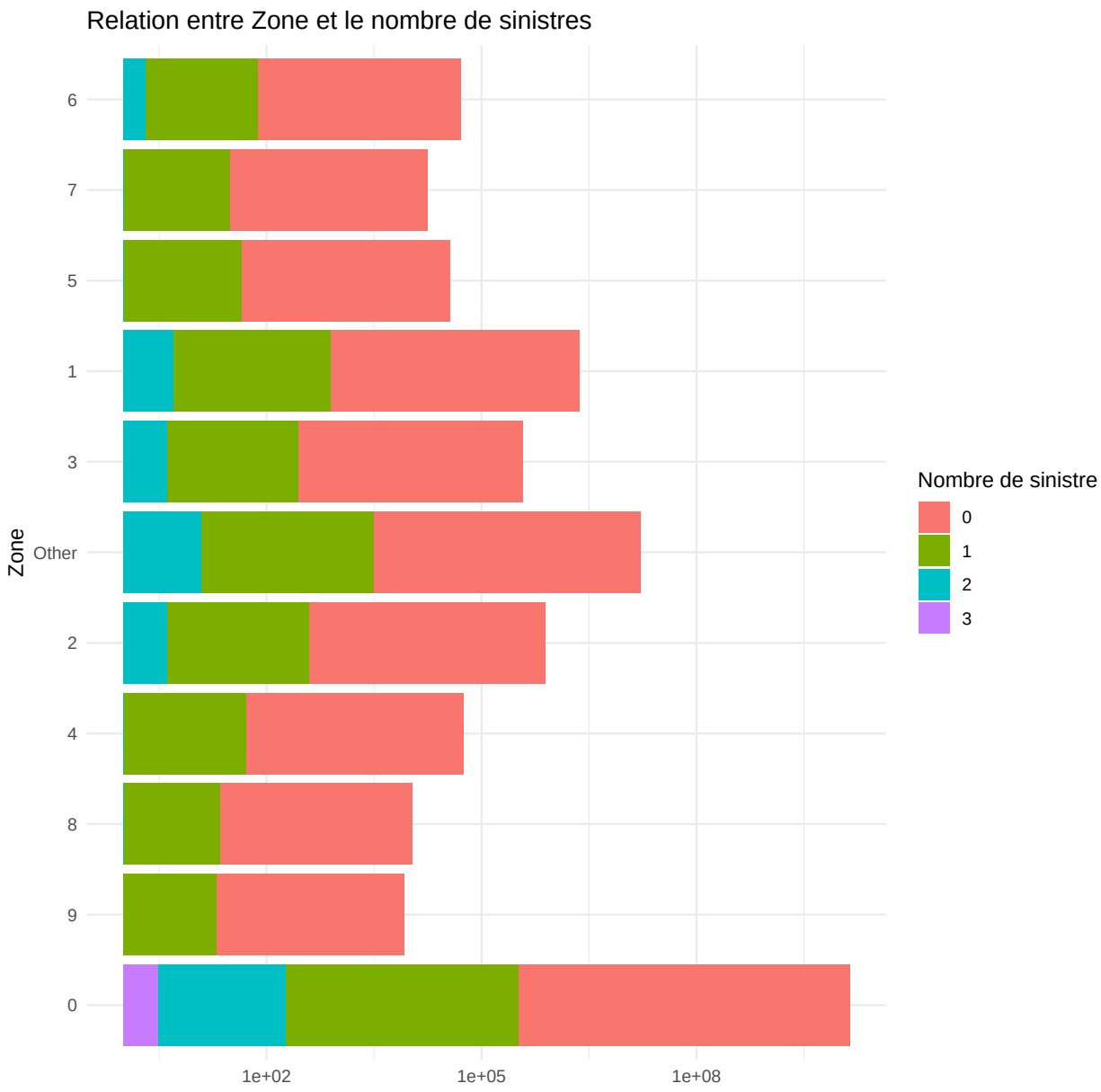
```
## #   variable_89 <dbl>, variable_90 <dbl>, variable_91 <dbl>, variable_92 <dbl>,
## #   variable_93 <dbl>, variable_94 <dbl>, variable_95 <dbl>, variable_96 <dbl>,
## #   variable_97 <dbl>, nb_sinistre <dbl>
```

```
## Aucun doublons, les personnes sont tous différents
```

On remarque que certaines personnes ont le même nom, mais ce sont des personnes différentes.

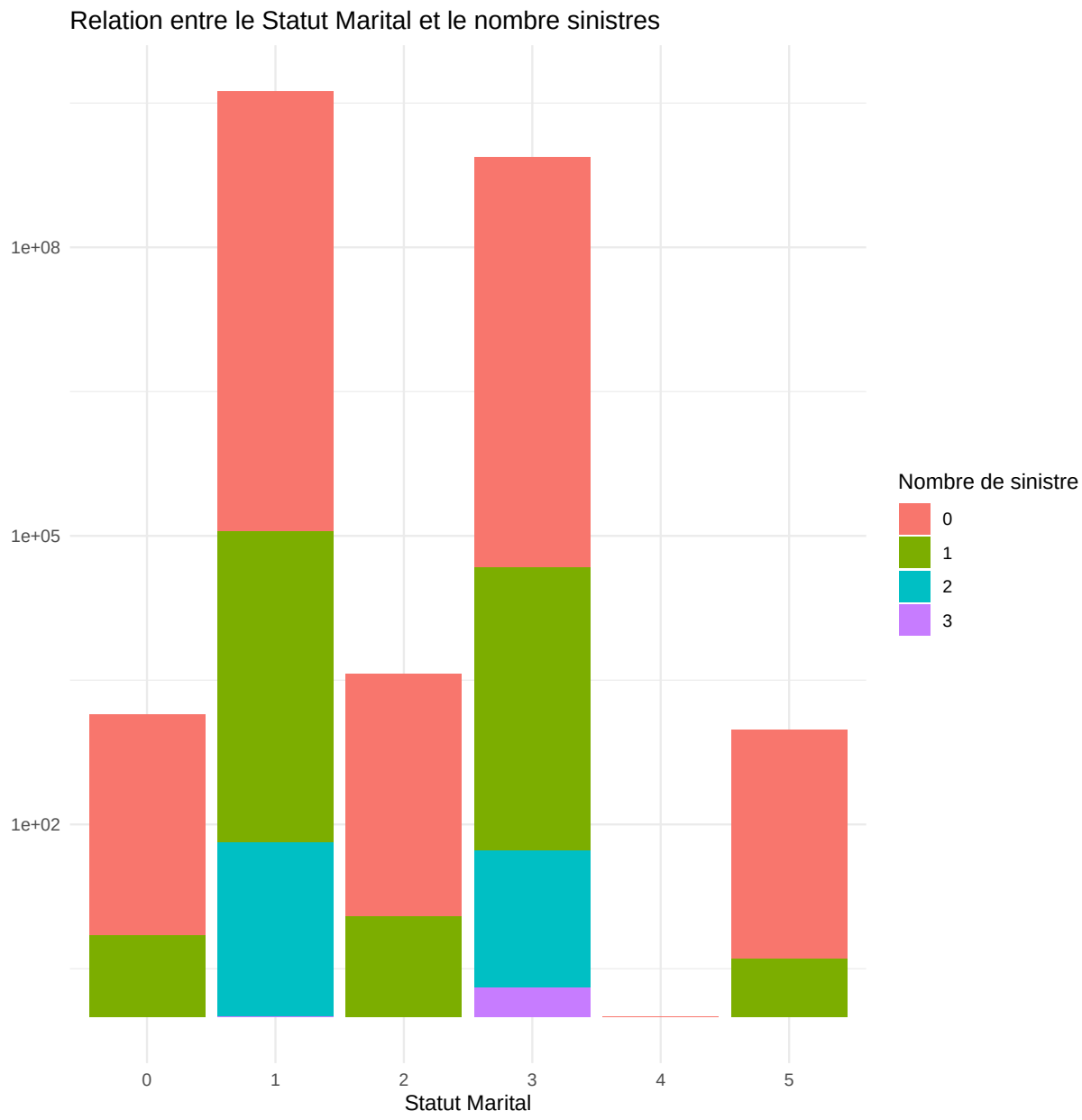
1.4 Zone

```
data %>%
  mutate(zone = as.factor(zone),
         zone = fct_lump(zone,10),
         zone = fct_reorder(zone,nb_sinistre,mean))%>%
  ggplot(mapping = aes(x = zone , fill = as.factor(nb_sinistre)))+
  geom_bar(stat = "count")+
  coord_flip()+
  scale_y_log10()+
  labs(title = "Relation entre Zone et le nombre de sinistres" , x="Zone" , y ="" , fill = "Nombre de si
```



1.5 Statut Marital

```
data %>%
  mutate(statut_marital = as.factor(statut_marital))%>%
  ggplot(mapping = aes(x = statut_marital,fill = as.factor(nb_sinistre)))+
  geom_bar(stat = "count")+
  scale_y_log10()+
  labs(title = "Relation entre le Statut Marital et le nombre sinistres" , fill = "Nombre de sinistre",
```



1.6 Créer une nouvelle colonne

Cette colonne nous aidera à diviser le problème de prédiction en deux, ceux qui ont fait aucun accident auront un nombre de sinistre égale à 0 et ceux qui ont fait des accidents, on fera une regression pour prédire leur nombre de sinistres.

```
data <- data %>%
  mutate(nb_statut = ifelse(nb_sinistre == 0 , 0 , 1 ))
```

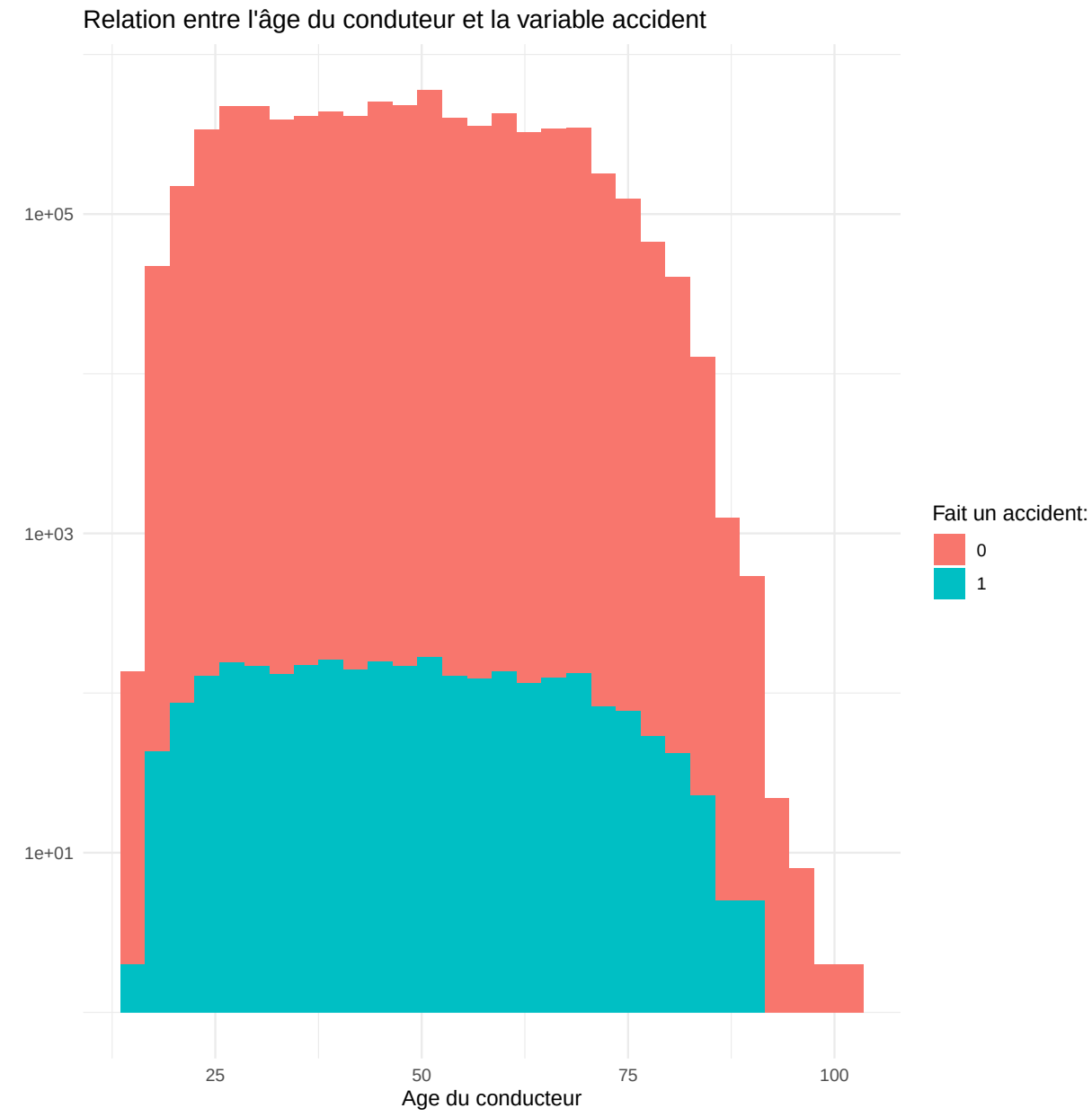
Voyons quelles sont les variables qui déterminent si un conducteur fera un accident ou non.

Feature enginneering Nous allons ajouter de nouvelles variables afin que de nous aider à prédire le nombre de sinistres.

```
data %>%
  mutate(feature_1 = nb_klms_declare_annee - distance_conduite,
         feature_2 = nb_annees_permis - nb_ann_sans_accidents,
         feature_3 = nb_ann_sans_accidents*utilisation_vehicule) ->data_pred
```

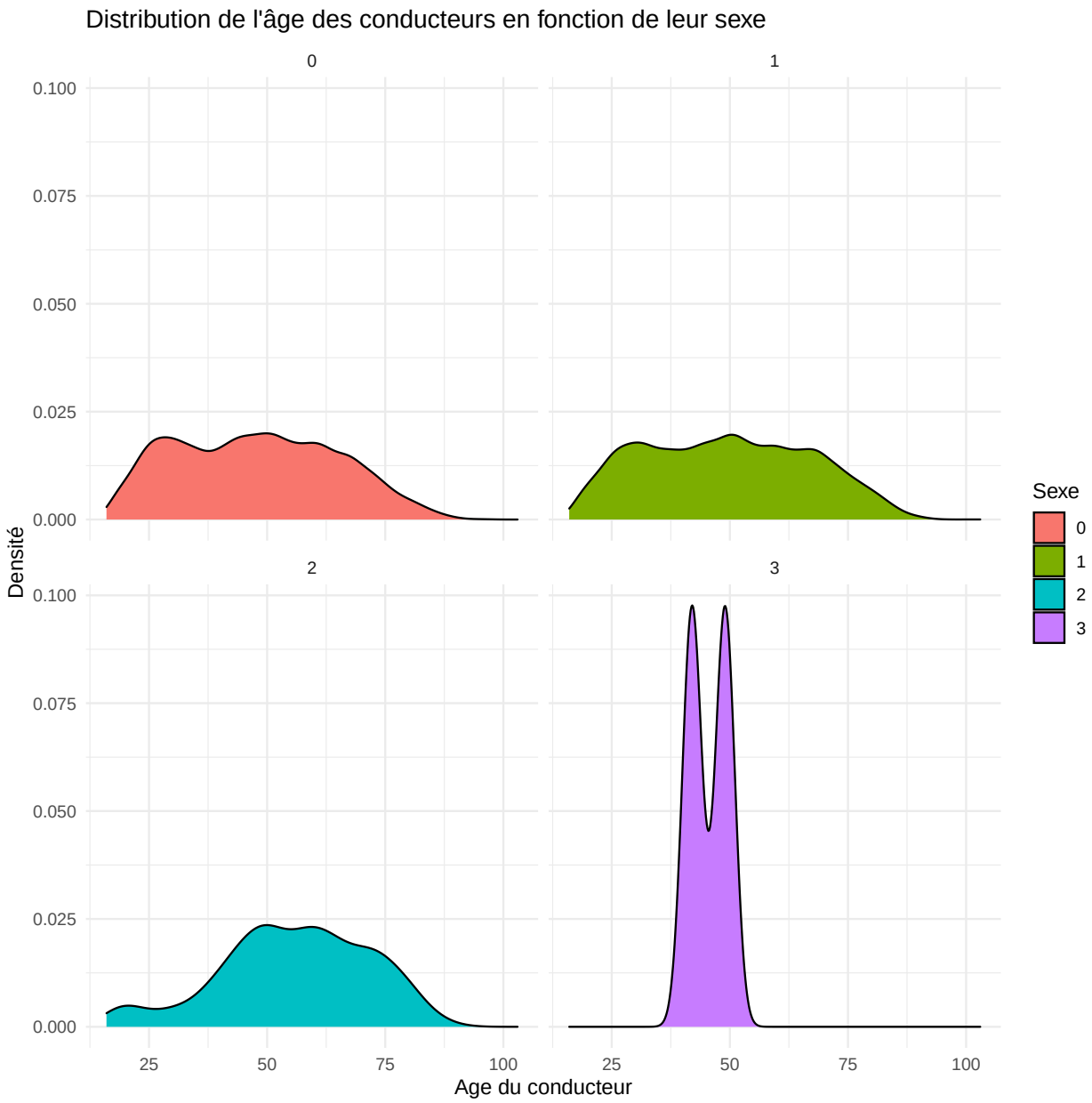
```
##Nombre de Statut
data %>%
  ggplot(mapping = aes(x = age_aconducteur , fill =as.factor(nb_statut)))+
  geom_histogram()+
  scale_y_log10()+
    labs(title = "Relation entre l'âge du conduteur et la variable accident" , fill = "Fait un accident
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Regardons maintenant la distribution de l'âge des conducteurs en fonction de leur sexe.

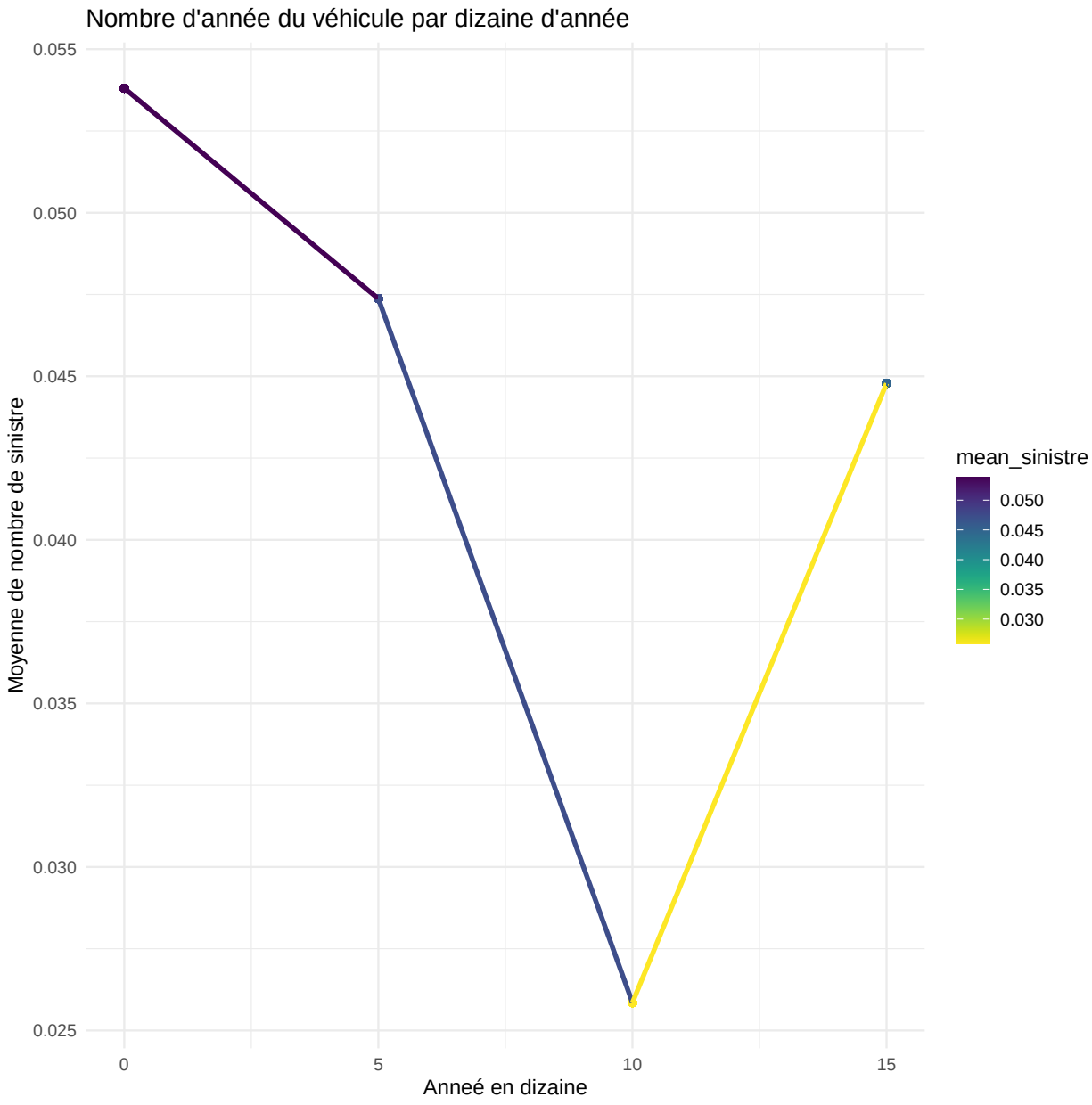
```
data %>%
  ggplot(mapping = aes(x = age_aconducteur,fill = as.factor(sexe)))+
  geom_density()+
  facet_wrap(~sexe)+
  labs(title = "Distribution de l'âge des conducteurs en fonction de leur sexe" , fill = "Sexe" , x  ="
```



1.7 Utilisation du véhicule

Jetons un coup d'oeil sur la variable Année Vehicule

```r
data %>%
    mutate(utilisation_vehicule = 5 * (utilisation_vehicule %/% 5))%>%
    group_by(utilisation_vehicule)%>%
    mutate(mean_sinistre = mean(nb_sinistre))%>%
    ggplot(mapping = aes(x = utilisation_vehicule , y = mean_sinistre,color = mean_sinistre))+
    geom_point(size = 1.5)+
    geom_line(size = 1.2)+
    scale_color_viridis(direction = -1  ,option = "F")+
    labs(y = "Moyenne de nombre de sinistre" , title = "Nombre d'année du véhicule par dizaine d'année"
```



Ce graphique explique deux phénomènes :

Le premier phénomène est dû au fait que les personnes qui ont peu d'expérience vont certainement avoir plus d'accidents. De plus, au fur et à mesure que l'utilisation du véhicule augmente, leur expérience augmente ce

qui fait diminuer le nombre de sinsitre moyen.

Cependant, on voit dans le graphique au temps 10 ans, le nombre de sinistre moyen augmente lorsque le minimum est atteint. Ceci est dû au fait que les sinsitres peuvent être causé par d'autres conducteurs non expérimentés et aussi au fait que notre risque d'avoir un sinistre augmente avec le temps.

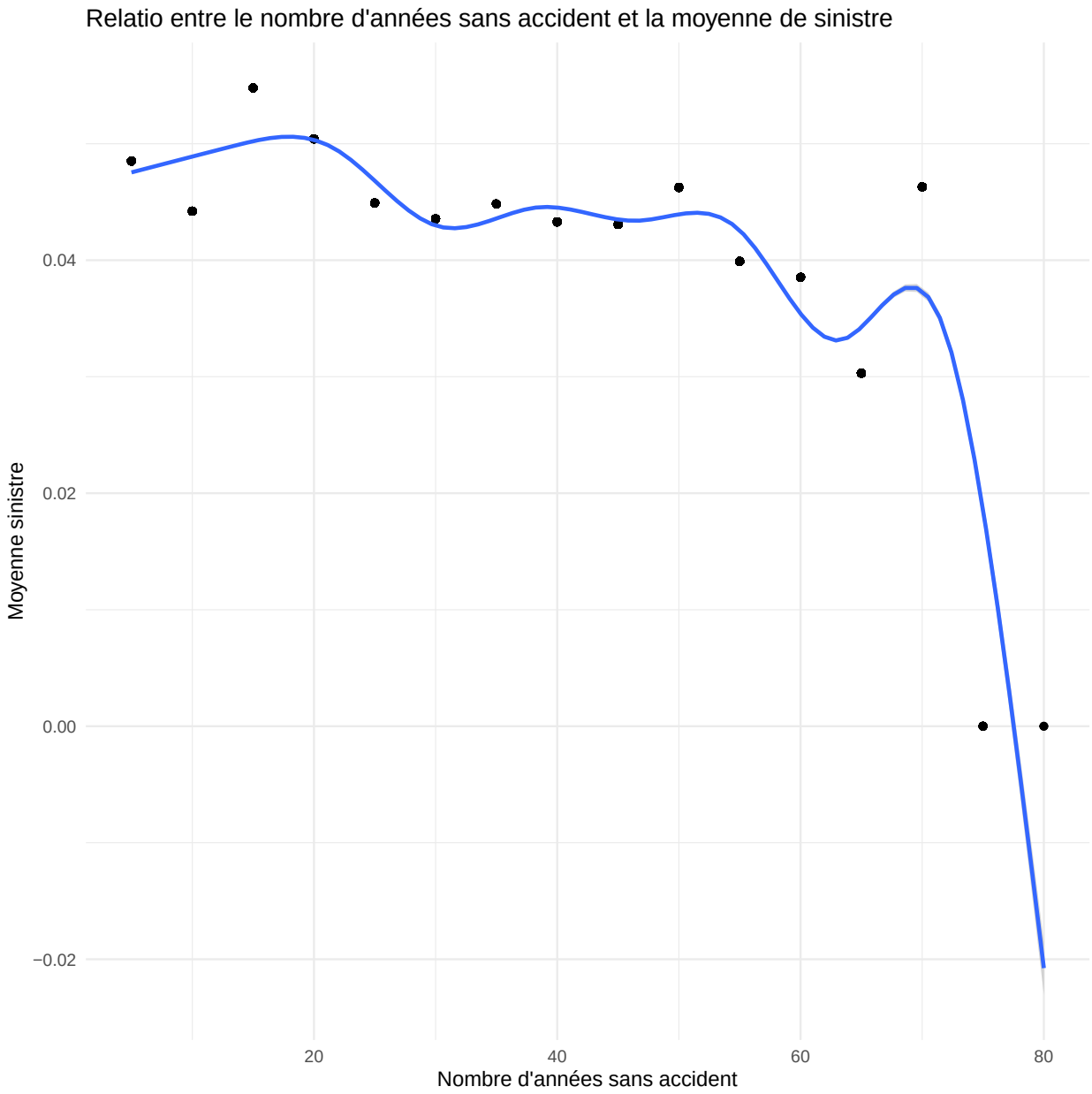
PS : Ou tout simplement le manque de personne qui ont utilisé leur vehicule plus que 15 ans.

1.8 Nombre d'année sans accident

Nous allons maintenant jeter un coup d'oeil sur cette variable, à priori elle devrait être intéressante.

```
data %>%
  filter(nb_ann_sans_accidents >= 5 )%>%
  mutate(nb_ann_sans_accidents = 5 * (nb_ann_sans_accidents %/% 5))%>%
  group_by(nb_ann_sans_accidents)%>%
  mutate(moyenne_sinistre = mean(nb_sinistre))%>%
  ggplot(mapping = aes(x = nb_ann_sans_accidents , y = moyenne_sinistre))+
  geom_point()+
  geom_smooth()+
  labs(title = "Relatio entre le nombre d'années sans accident et la moyenne de sinistre" , x = "Nombre
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Exactement ce qu'on avait prédit, le nombre d'accident diminue en fonction du nombre d'année sans accident des conducteurs.

Cette variable devrait être intéressante dans nos modèles.

1.9 Les variables sur les caractéristiques des voitures des assurés

Regardons combien de modèle nous avons dans notre base de données

```
data %>%
  count(model_vehicule,type_vehicule,marque_vehhicule,sort = TRUE)
```

```
## # A tibble: 619 x 4
```

```
##    model_vehicule type_vehicule marque_vehhicule     n
##             <dbl>         <dbl>            <dbl> <int>
##  1            571             6               47  8688
##  2            129             3                7  1952
##  3            164             4               44  1924
##  4            141             4               14  1648
##  5            217             5                9  1335
##  6            340             4               28  1198
##  7            571             4               14  1134
##  8            126             4               44  1079
##  9            208             3                9  1054
## 10            404             3               44  1020
## # ... with 609 more rows
```

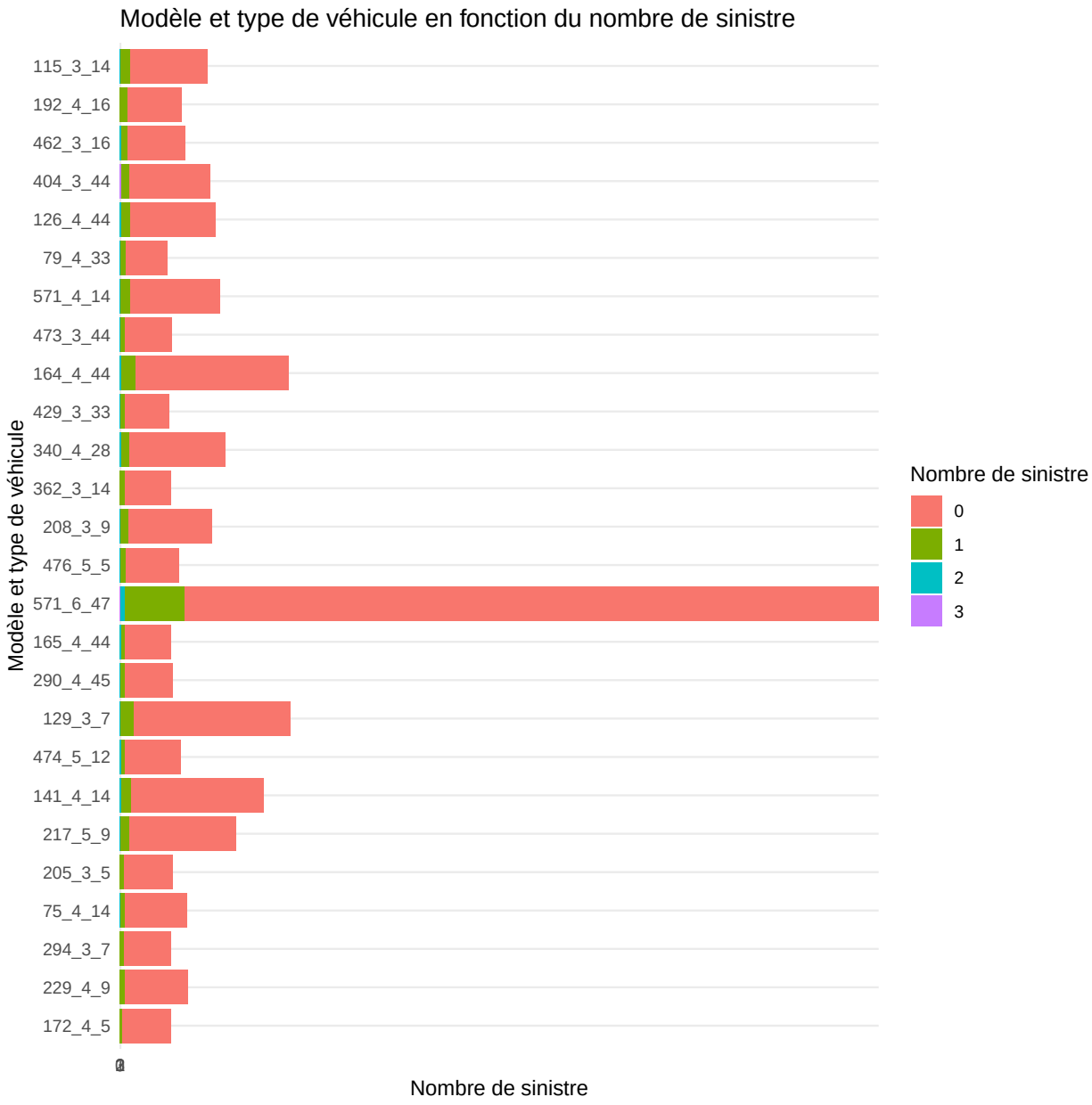
WOW , nous avons 590 modèles avec un type de véhicule différents, cette variable n'est pas intéressante. En effet, si nous utilisons un modèle à arbre de décision cette variable pourrait créer plusieurs branches différentes donc ce modèle ne sera pas très flexible.

On remarque cependant qu'on une voiture de modèle '571' et de type '6' qui est la plus conduite par nos conducteurs.

Regardons celles qui sont les plus présentes dans notre base de données.

Nous allons créer une nouvelle colonne qui regroupe le modèle du véhicule et son type.

```
data %>%
  mutate(type_mod = paste0(model_vehicule,"_",type_vehicule,"_",marque_vehhicule),
         type_mod = fct_reorder(type_mod,nb_sinistre,mean))%>%
  group_by(type_mod)%>%
  mutate(nb = n())%>%
  filter(nb > 500)%>%
  ggplot(mapping = aes(x = type_mod , y = as.factor(nb_sinistre), fill = as.factor(nb_sinistre)))+
  geom_bar(stat = "identity")+
  coord_flip()+
  labs(title = "Modèle et type de véhicule en fonction du nombre de sinistre" , fill = "Nombre de sinis
```

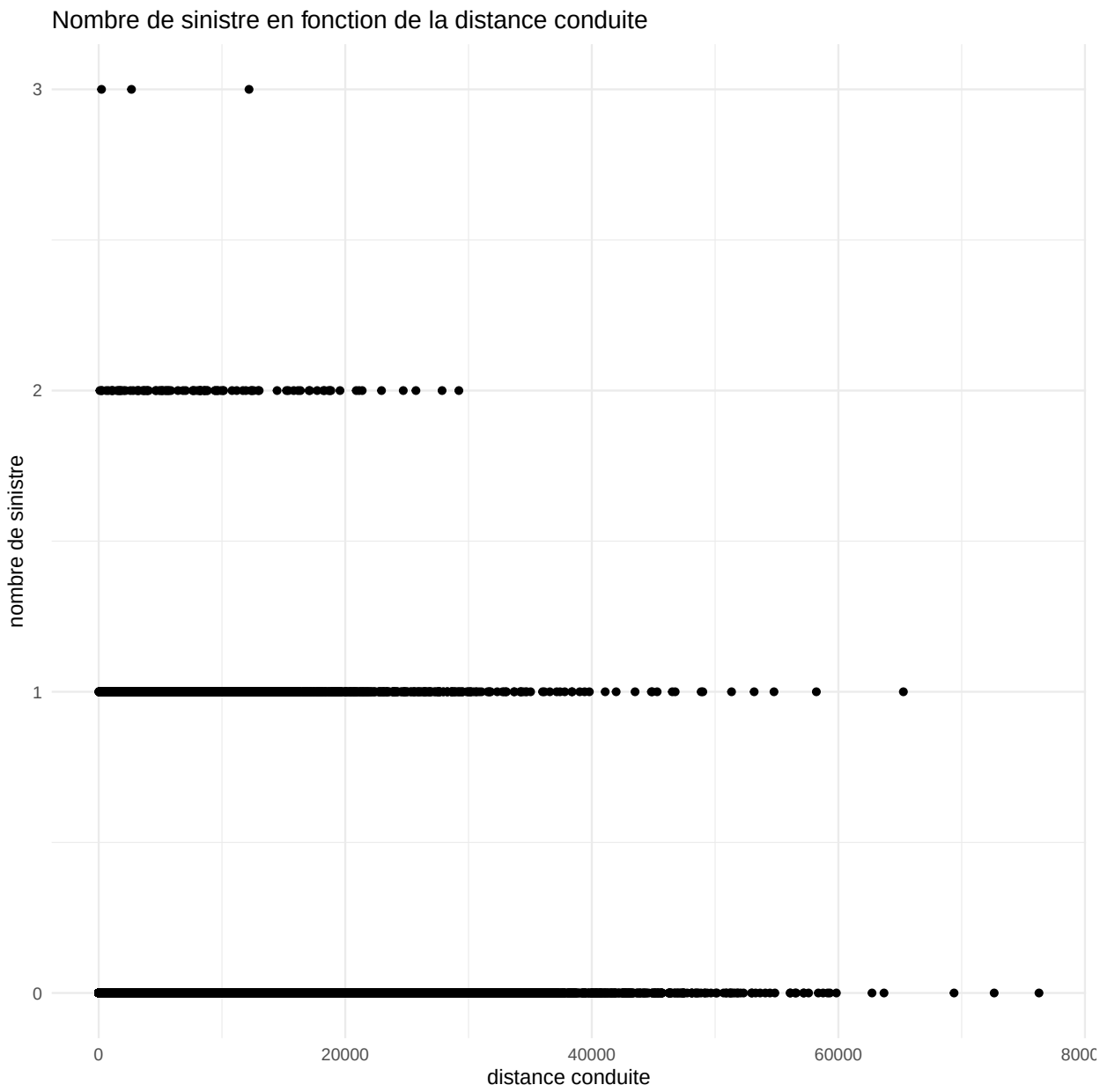


Au sommet, nous avons le modèle 115_3_14 qui a le plus grand *taux* de sinistre.

Dans ce graphique, nous avons choisi de classer les types de véhicule et les modèles en fonction du nombre de sinistre moyen.

2.0 Nombre de sinistre selon les distances parcourru

```
data %>%
  ggplot(mapping = aes(x = distance_conduite , y = nb_sinistre))+
  geom_point()+
  labs(x = "distance conduite" , y = "nombre de sinistre", title = "Nombre de sinistre en fonction de la
```



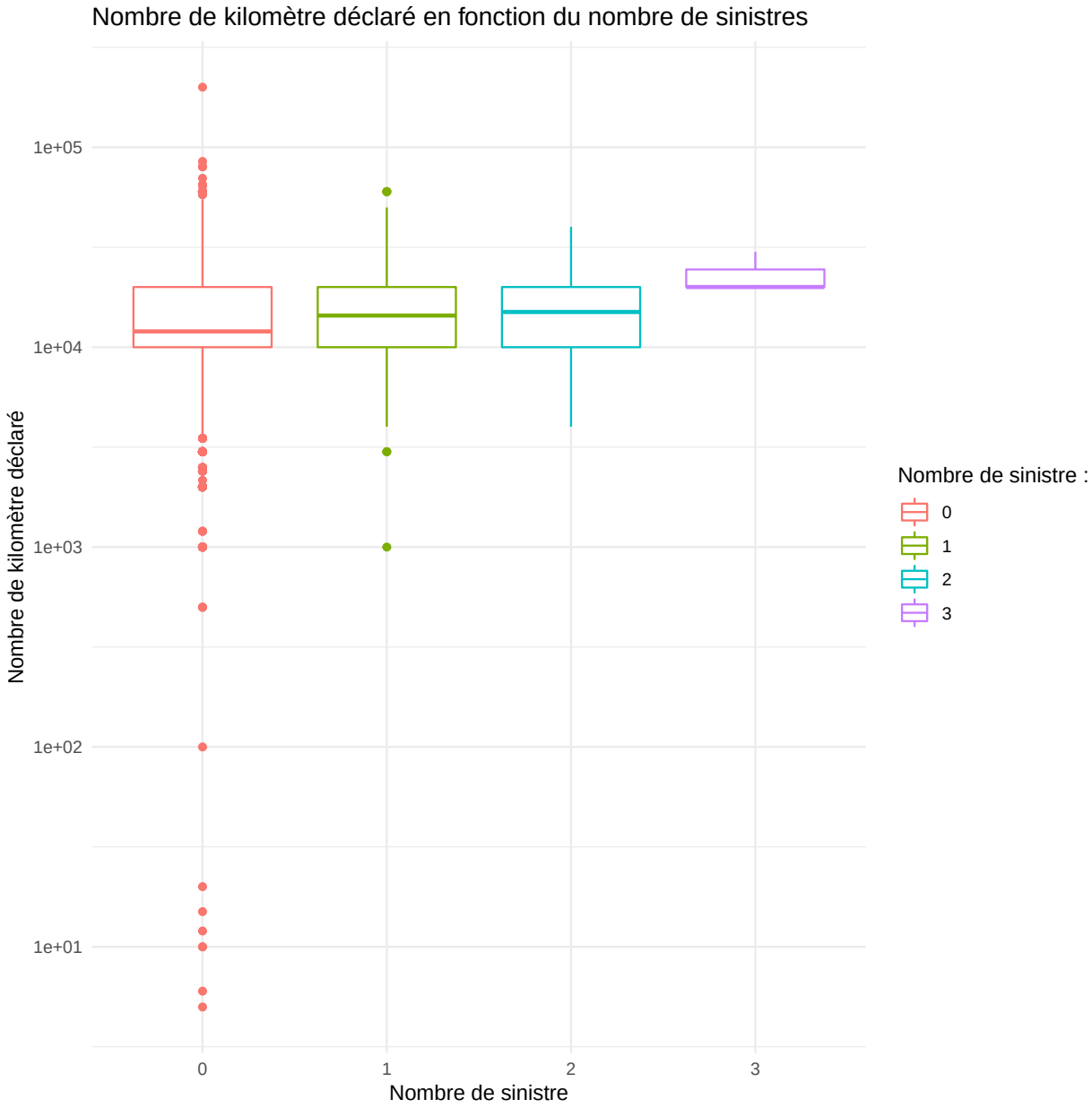
Comme nous pouvons constater avec le graphique ci-dessus, la distance conduite ne serait pas la principale cause des sinistres. En effet, on peut voir que les personnes ayant eu le plus de sinistres sont celles ayant conduit une petite distance. Donc, les sinistres sont plus relier à la qualité du conducteur qu'à la distance parcourrue.

2.1 Nombre de kilomètre

Nous allons traiter regrouper le nombre de sinistre en plusieurs catégorie.En effet, cela nous permettrait de faire des boxplo afin de mieux voir la distribution. On peut voir le chiffre "0" comme un conducteur non risqué et le chiffre 3 étant un conducteur risqué.

```
data %>%
  ggplot(mapping = aes(x = as.factor(nb_sinistre), y = nb_klms_declare_annee,color = as.factor(nb_sinis
```

```
  geom_boxplot()+
  scale_y_log10()+
  labs(title = "Nombre de kilomètre déclaré en fonction du nombre de sinistres" , y = "Nombre de kilomè
```



On remarque que les deux catégories qui se distinguent le plus sont ceux qui ne font pas d'accident et ceux qui ont fait 3 accidents. La médiane de ceux qui ont fait 3 accidents est assez élevé par rapport aux autres catégories.

Regardons un peu les classes :

```
table(data$nb_statut)
```

```
##
##     0     1
```

```
## 57327  2672
```

Nous pouvons remarquer qu'environ 5 % des personnes ont fait un accident.

Combinaisons lineaire de certaines variables

```
data <- data[,-c(1:2)]

linear_combination <- findLinearCombos(data)


data <- data[,-linear_combination$remove]

data_pred <- data_pred[,-linear_combination$remove]
```

Transformons les variables numériques en facteur

```
facteurs_liste <- c("statut_marital","sexe","zone","code_territoire","type_vehicule","marque_vehhicule"

data_pred[facteurs_liste] <- lapply(data_pred[facteurs_liste], function(x) factor(x))
```

Nous allons maintenant diviser notre base données en train et test

```
set.seed(129) ##Pour préserver l'exactitude nos réponses

sample<- sample(1:nrow(data),nrow(data)*0.7,replace = F)

data_pred <- data_pred[,-c(1:2)]


levels(data_pred$nb_statut) <- c("non","oui")


train <- data_pred[sample,]
test <- data_pred[-sample,]
```

Vérifions les proportions afin de voir si elles sont proches.

```
prop.table(table(train$nb_statut))
```

```
##
##        non        oui
## 0.95545132 0.04454868
```

```
prop.table(table(test$nb_statut))
```

```
##
##    non    oui
## 0.9555 0.0445
```

Partie 2

1.1 Clustering

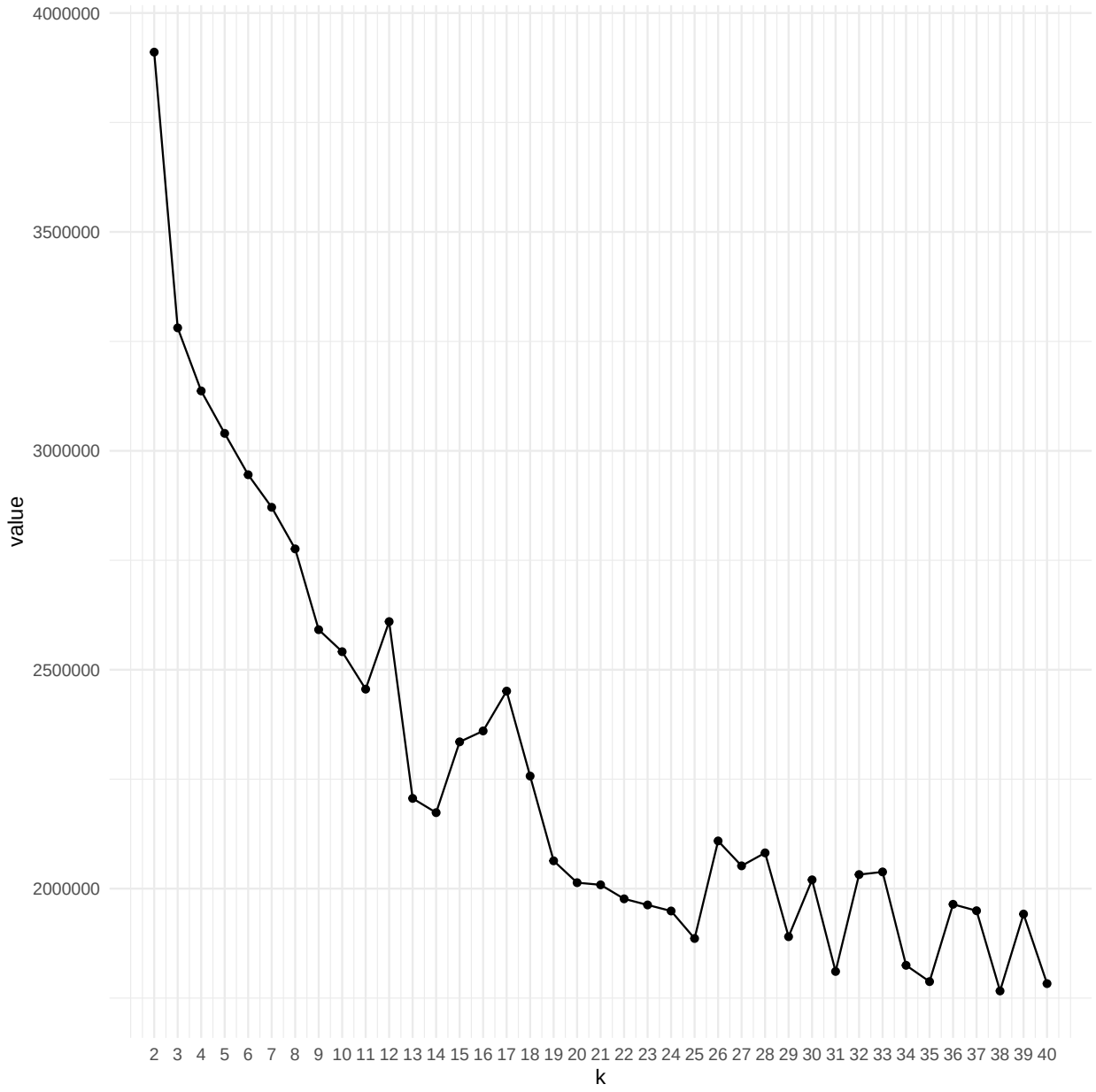
Nous allons utiliser du clustering pour différencier les conducteurs avec sinistre des conducteurs sans sinistre dans notre base de données.

On sélectionnera seulement les variables numériques et on va "Scale" notre base de données, car les variables ne sont pas toutes sur la même échelle de mesure.

```
data_pred %>%
   select_if(is.numeric) %>%
  mutate_if(is.numeric,scale) ->data_pred_cluster
```

Pour le clustering, nous allons performer un K-mean. De plus, vu que cet algorithm demande qu'on donne une valeur à k. Nous allons créer une fonction qui fera du clustering sur notre base de données. Ensuite, nous allons effectuer un graphique des résultats et on utilisera la méthode "elbow".

```
kmeans_function <- function(k) {
    cluster <- kmeans(data_pred_cluster, k)
    return (cluster$tot.withinss)
}

df_kmeans <- sapply(2:40, kmeans_function)
elbow <-data.frame(k = 2:40,value = df_kmeans)


ggplot(elbow, aes(x = k, y = value)) +
    geom_point() +
    geom_line() +
    scale_x_continuous(breaks = seq(2, 40, by = 1))
```



On essayera avec K=20 , pour préserver les résultats on utilisera un seed. Veuillez noter que si vous utiliser cet algorithm sur une autre base de données les résultats peuvent être différent.

```
set.seed(129)
k2_clust <- kmeans(data_pred_cluster,20)

table(k2_clust$cluster)
```

```
##
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2158   27 5252  794 5251 2891 3192 7574 1404 6113   55 4358   24   17 1574 7289
##   17   18   19   20
##  654 3790  625 6957
```

```r
table(data_pred$nb_statut,k2_clust$cluster)
```

```
##
##          1    2    3    4    5    6    7    8    9   10   11   12   13   14
##   non 2098   27 5246    0 5248 2824 3173 7574 1369 6113   55 4356   24   15
##   oui   60    0    6  794    3   67   19    0   35    0    0    2    0    2
##
##         15   16   17   18   19   20
##   non    0 7267  624 3746  611 6957
##   oui 1574   22   30   44   14    0
```

COOL ! Si on regarde bien notre tableau, on voit que ceux qui ont fait des accidents sont majoritairement dans la classe *4* et *15*. De plus, tous les indivius qui n'ont fait aucun ne sont pas dans ces classes.

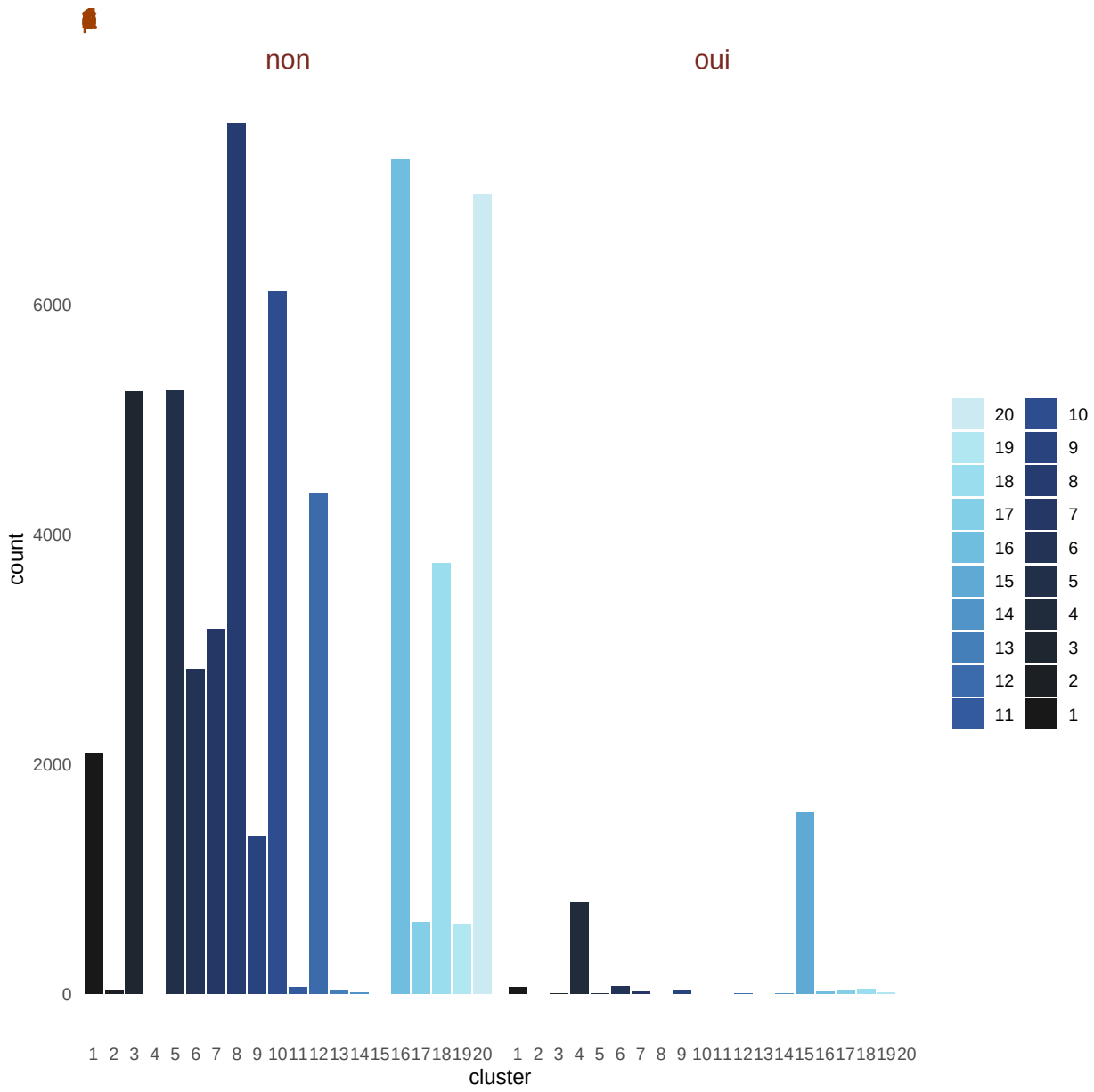
Donc, on associera la réponse "oui" (Ayant eu un accident) à ceux de la classe *15* et *4* et "non" aux autres classes.

```r
data_pred$cluster <- k2_clust$cluster

data_pred %>%
  mutate(nb_statut_pred = ifelse(cluster == 15 | cluster == 4 ,"oui","non")) ->data_pred


prop.table(table(data_pred$nb_statut_pred,data_pred$nb_statut))
```

```
##
##                non         oui
##   non 0.955465924 0.005066751
##   oui 0.000000000 0.039467324
```

Donc, pour les assurés auxquels nous avons accordé une valeur "non" auront automatiquement une prédiction de *0* pour le nombre de *sinistre*.

```r
data_pred %>%
  ggplot(mapping = aes(x = as.factor(cluster) , fill = as.factor(cluster)))+
  geom_bar(stat = "count")+
  facet_wrap(~nb_statut)+
  labs(fill=NULL)+
  scale_fill_fish_d(option = "Ostracion_whitleyi")+
  guides(fill = guide_legend(reverse = TRUE , ncol = 2))+
  theme(
    panel.grid = element_blank(),
    strip.text = element_text(size = 14,color = "#78281f"),
    plot.title = element_text(size = 14 , color = "#a04000",family = "Arial Nova")
  )+
  labs(x = "cluster" , title = "Séparation des assurés avec sinistre et sans sinistre")
```



Update Train et Test

```
train <- data_pred[sample,]
test <- data_pred[-sample,]
```

```
test %>%
  mutate(predictions_nb_sinistre = ifelse(nb_statut_pred=="non",0,NA))->test_1


test_1 %>%
    filter(is.na(predictions_nb_sinistre)) ->test_1_avec_sinistre
```

On garde notre base de données *test* avec les valeurs non prédit qui sont les personnes qui ont fait un accident ou encore les personnes de la classe *4* et *15*.

Nous allons effectuer les mêmes étapes pour train et on entrainera des modèles sur cette base de données. Puis, on prédira la base test_1_avec_sinistre.

```
train %>%
   filter(nb_sinistre>0) ->train_avec_sinistre
```

1.2 XGBOOST Linéaire

Tuning parameter, notez que nous avons performé plusieurs fois le modèle afin de sélectionner les meilleurs paramètres possibles.

Tout d'abord, nous allons exécuter notre premier algorithme : *xgboost linéaire*.

```
tun_grid <- expand.grid(
  nrounds = seq(from = 200 , to = 300,by = 50),
  eta = c(0.001,0.002,0.03),
  alpha = c(0.05,0.1),
  lambda = c(0.2,0.25,0.3)
)
```

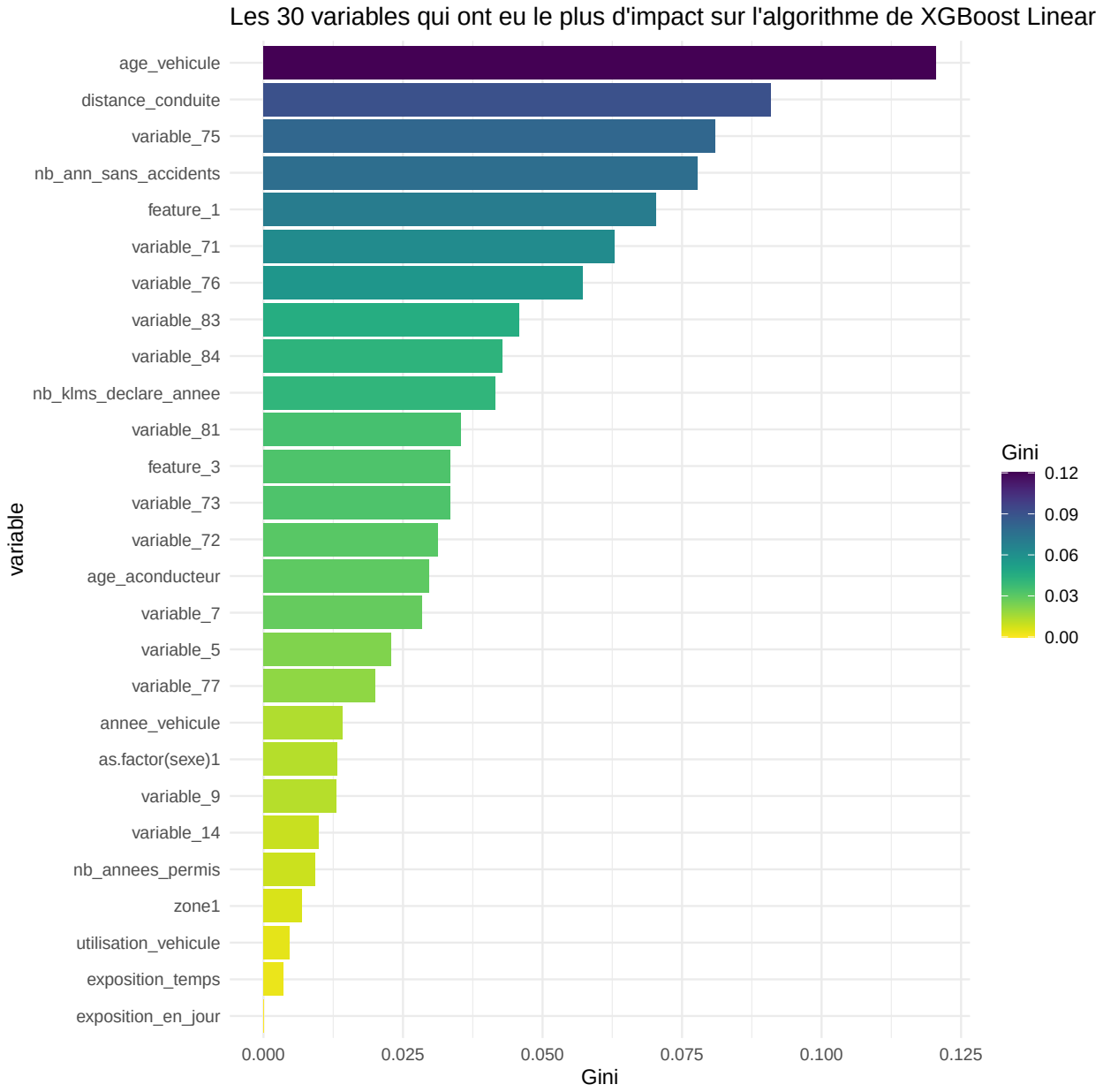
```
model_xgbL <- train(nb_sinistre ~ age_vehicule+distance_conduite+variable_75+variable_71+exposition_en_
```

Le modèle finale a été selectionné selon la performance du RMSE. Les paramètres qui minimisent le RMSE sont les suivants : **nrounds= 200** , **lambda=0.3** , **alpha=0.1** et **eta=0.001**

Regardons, quelles sont les variables importantes pour cet algorithme.

```
importance <- varImp(model_xgbL,scale = F)

importance <- importance$importance

data.frame(variable = row.names(importance), Gini = importance$Overall) %>%
  top_n(30)%>%
  mutate(variable = fct_reorder(variable,Gini))%>%
  ggplot(mapping = aes(x = variable , y = Gini,fill = Gini ))+
  geom_bar(stat = 'identity')+
  scale_fill_viridis(direction = -1,option = 'F')+
  theme_minimal()+
  coord_flip()+
  labs(title = "Les 30 variables qui ont eu le plus d'impact sur l'algorithme de XGBoost Linear")
```

```
## Selecting by Gini
```



On remarque que la 4e valeur la plus importante est : *feature_1*. En effet, c'est une variable que nous avons créé.

Passons aux prédictions, nous allons vérifier la performance de notre modèle.

```
predictions_xgbL <- round(predict(model_xgbL,test_1_avec_sinistre))



prop.table(table(predictions_xgbL,test_1_avec_sinistre$nb_sinistre)
)
```

```
##
## predictions_xgbL           1           2           3
```

```
##                 1 0.952973721 0.038727524 0.002766252
##                 2 0.005532503 0.000000000 0.000000000
```

Notre modèle prédit que la majorité des assurés avec sinistre ont seulement eu un sinistre durant cette année avec une précision de 95.2973%.

XGBOOST Tree

Comme dit le nom, on perfomera maintenant un xgboost, mais sur un arbre. Voici les paramètres pour tuner notre algorithm.

```
grid_default <- expand.grid(
  nrounds = seq(from = 100 , to = 200,by = 50),
  max_depth = c(5,6,7,8,9,10),
  eta = c(0.003,0.004,0.05),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)
```

```
model_xgbT <- train(nb_sinistre ~ age_vehicule+distance_conduite+variable_75+variable_71+exposition_en_
                      annee_vehicule+nb_ann_sans_accidents+nb_annees_permis+variable_72+zone+exposition_
                        nb_klms_declare_annee+variable_83+feature_3+variable_81+age_aconducteur+variable_
                    trControl = trainControl(method = "cv",number = 2))
```

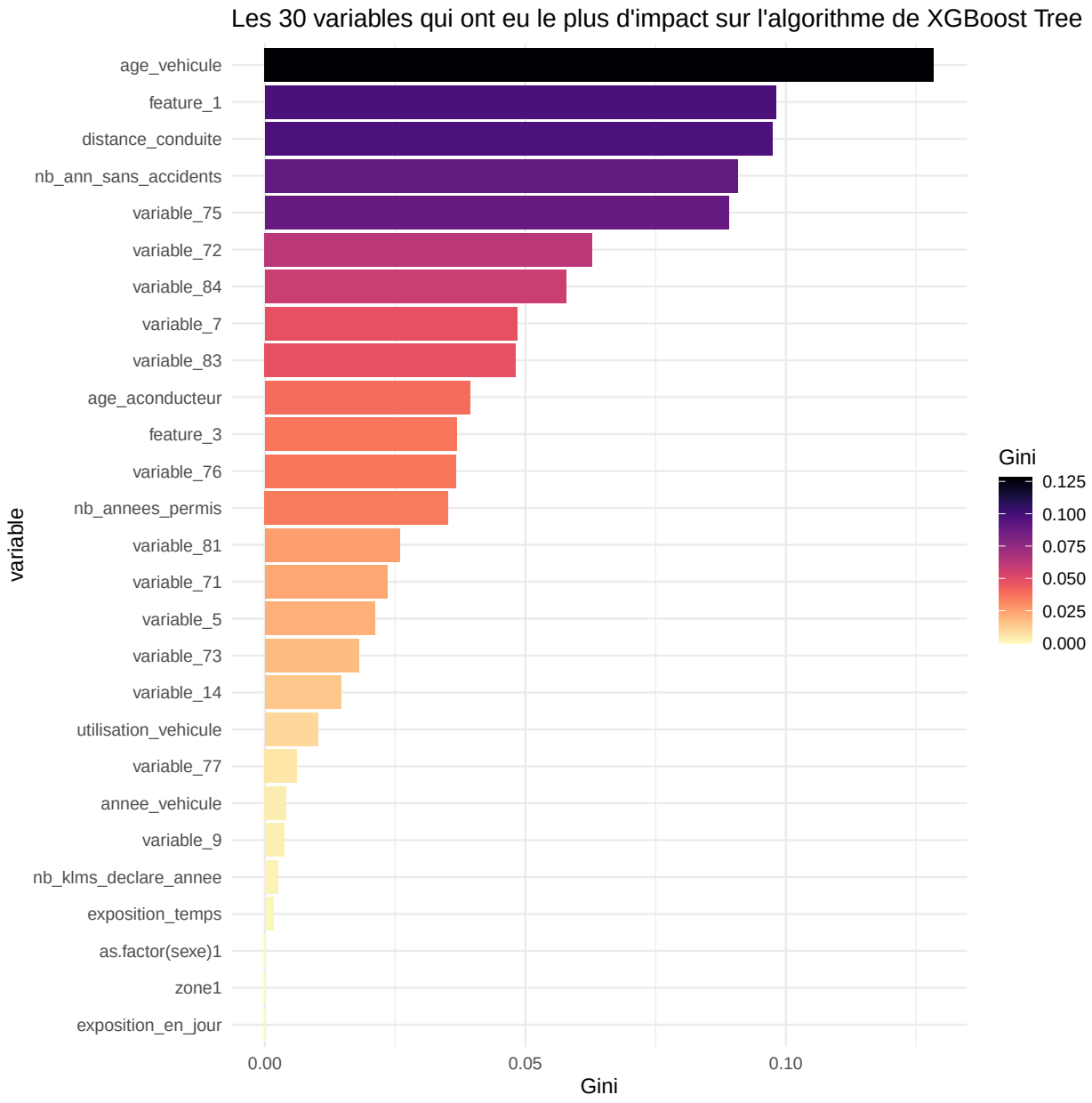
Encore une fois, le modèle a choisi les paramètres qui minimisent le RMSE et les paramètres sont :

nrounds=100 , **eta=0.05** , **max_depth=5** , **gamma=0** , **colsample_bytree=1** , **min_child_weight=1** et **subsample=1**

Observons l'importance des variables pour cet algorithme

```
importance <- varImp(model_xgbT,scale = F)

importance <- importance$importance

data.frame(variable = row.names(importance), Gini = importance$Overall) %>%
  top_n(30)%>%
  mutate(variable = fct_reorder(variable,Gini))%>%
  ggplot(mapping = aes(x = variable , y = Gini,fill = Gini ))+
  geom_bar(stat = 'identity')+
  scale_fill_viridis(direction = -1,option = 'A')+
  theme_minimal()+
  coord_flip()+
  labs(title = "Les 30 variables qui ont eu le plus d'impact sur l'algorithme de XGBoost Tree")
```

```
## Selecting by Gini
```



Ici, on voit que notre feature_1 est en 2e position. Vérifions les prédictions jusqu'à présent.

```
predictions_xgbT <- round(predict(model_xgbT,test_1_avec_sinistre))



prop.table(table(predictions_xgbT,test_1_avec_sinistre$nb_sinistre)
)
```

```
##
## predictions_xgbT           1           2           3
##                1 0.955739972 0.038727524 0.002766252
##                2 0.002766252 0.000000000 0.000000000
```

On obtient les même résultats.

1.3 Random Forest

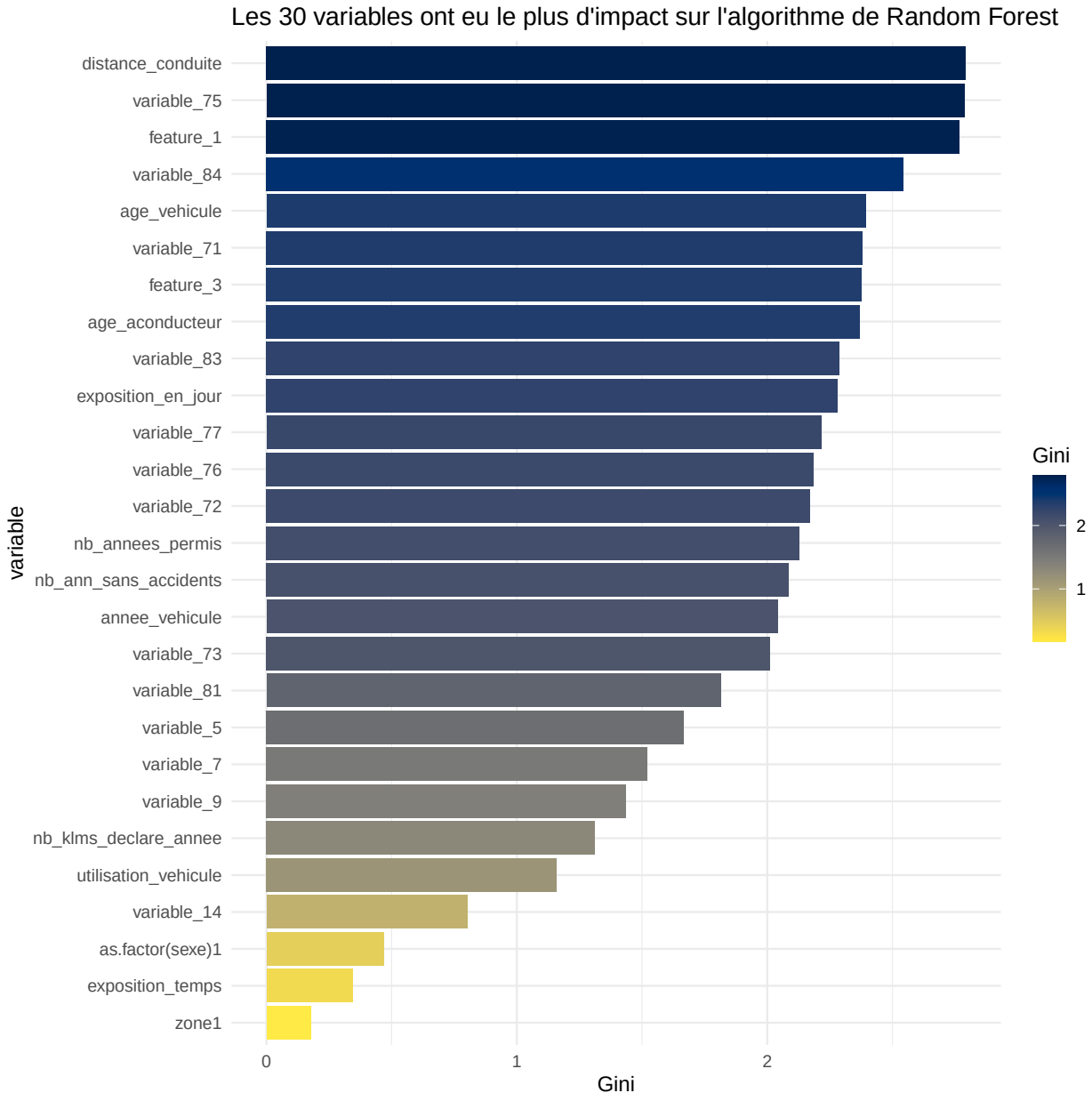
```
model_rf <- train(nb_sinistre ~ age_vehicule+distance_conduite+variable_75+variable_71+exposition_en_jou
                    annee_vehicule+nb_ann_sans_accidents+nb_annees_permis+variable_72+zone+exposition_
                  nb_klms_declare_annee+variable_83+feature_3+variable_81+age_aconducteur+variable_76
                 trControl = trainControl(method = "cv",number = 2))
```

L'algorithme choisira **mtry=2**, car il minimise le RMSE.

Vérifions l'importance des variables.

```
importance <- varImp(model_rf,scale = F)

importance <- importance$importance

data.frame(variable = row.names(importance), Gini = importance$Overall) %>%
  top_n(30)%>%
  mutate(variable = fct_reorder(variable,Gini))%>%
  ggplot(mapping = aes(x = variable , y = Gini,fill = Gini ))+
  geom_bar(stat = 'identity')+
  scale_fill_viridis(direction = -1,option = 'E')+
  theme_minimal()+
  coord_flip()+
  labs(title = "Les 30 variables ont eu le plus d'impact sur l'algorithme de Random Forest")
```

```
## Selecting by Gini
```




```
predictions_rf <- round(predict(model_rf,test_1_avec_sinistre))

prop.table(table(predictions_rf,test_1_avec_sinistre$nb_sinistre)
)
```

```
##
## predictions_rf           1           2           3
##              1 0.958506224 0.038727524 0.002766252
```

Random forest ne prédit que des **1**. Ceci augmente notre précision, mais l'algorithme pourrait être moins flexible sur une autre base de données par rapport aux deux autres algorithme précédent.

Finalement, nous avons choisi l'algorithme *XGBOOST Lineaire* .

1.4 Prédictions Finale

```
test_1_avec_sinistre$predictions_nb_sinistre <- predictions_xgbL

test_1 %>%
  mutate(predictions_nb_sinistre = ifelse(nb_statut_pred=="oui",predictions_xgbL,predictions_nb_sinistre
```

Proportion correcte de nos prédictions.

```
table(test_finale$predictions_nb_sinistre,test$nb_sinistre)
```

```
##
##         0     1     2     3
##   0 17199    78     0     0
##   1     0   689    27     2
##   2     0     4     1     0
```

```
out <- prop.table(table(test_finale$predictions_nb_sinistre,test$nb_sinistre))
print(out)
```

```
##
##                0           1           2           3
##   0 9.555000e-01 4.333333e-03 0.000000e+00 0.000000e+00
##   1 0.000000e+00 3.827778e-02 1.500000e-03 1.111111e-04
##   2 0.000000e+00 2.222222e-04 5.555556e-05 0.000000e+00
```

```
Accuracy <- out[1,1] + out[2,2] + out[3,3]
print(Accuracy)
```

```
## [1] 0.9938333
```

Partie 3

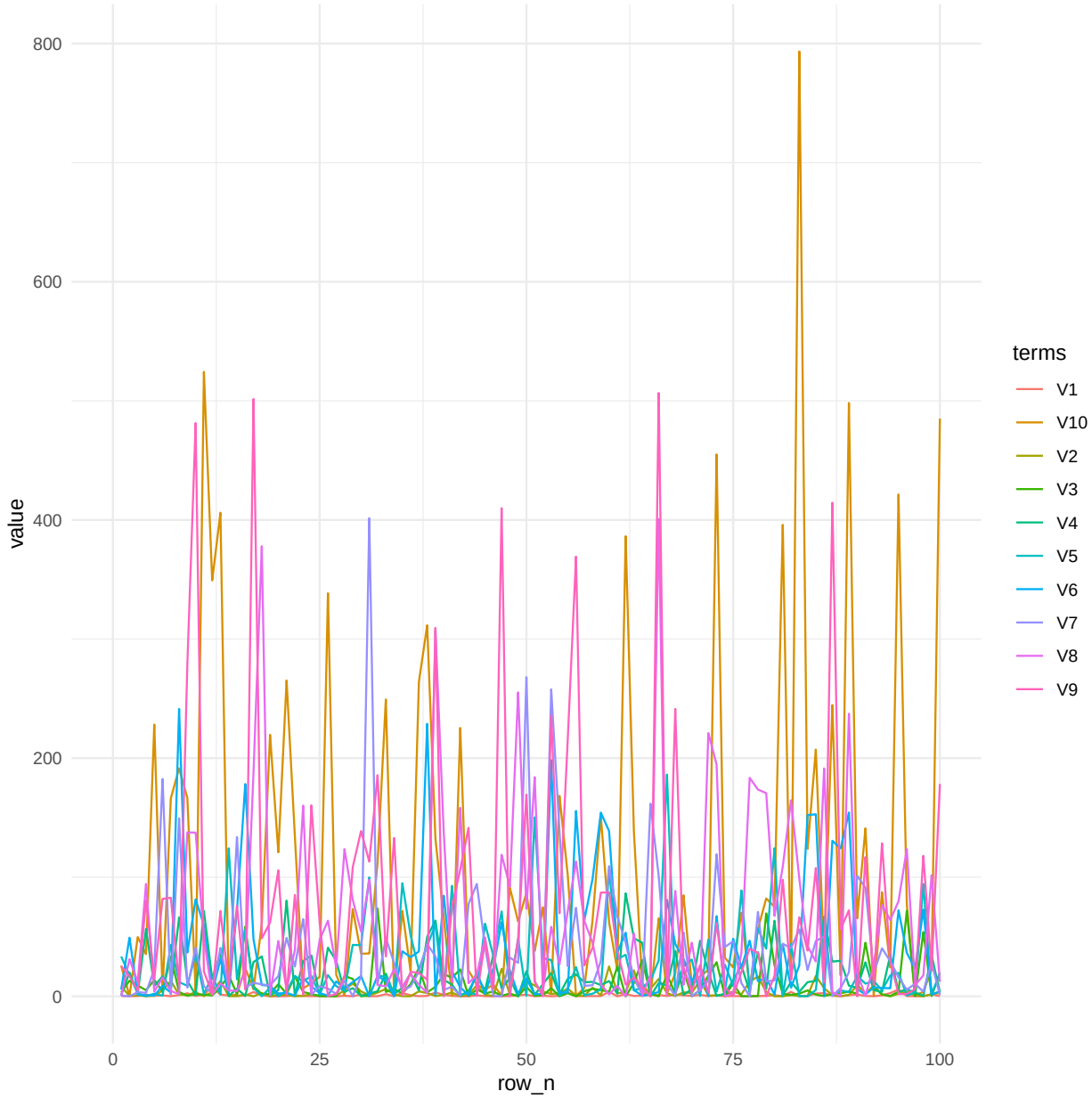
Selection de modèle

Vérifions pourquoi le RMSE est un mauvais prédicteur pour sélectionner un modèle et essayons de voir quel autre “metric” choisir.

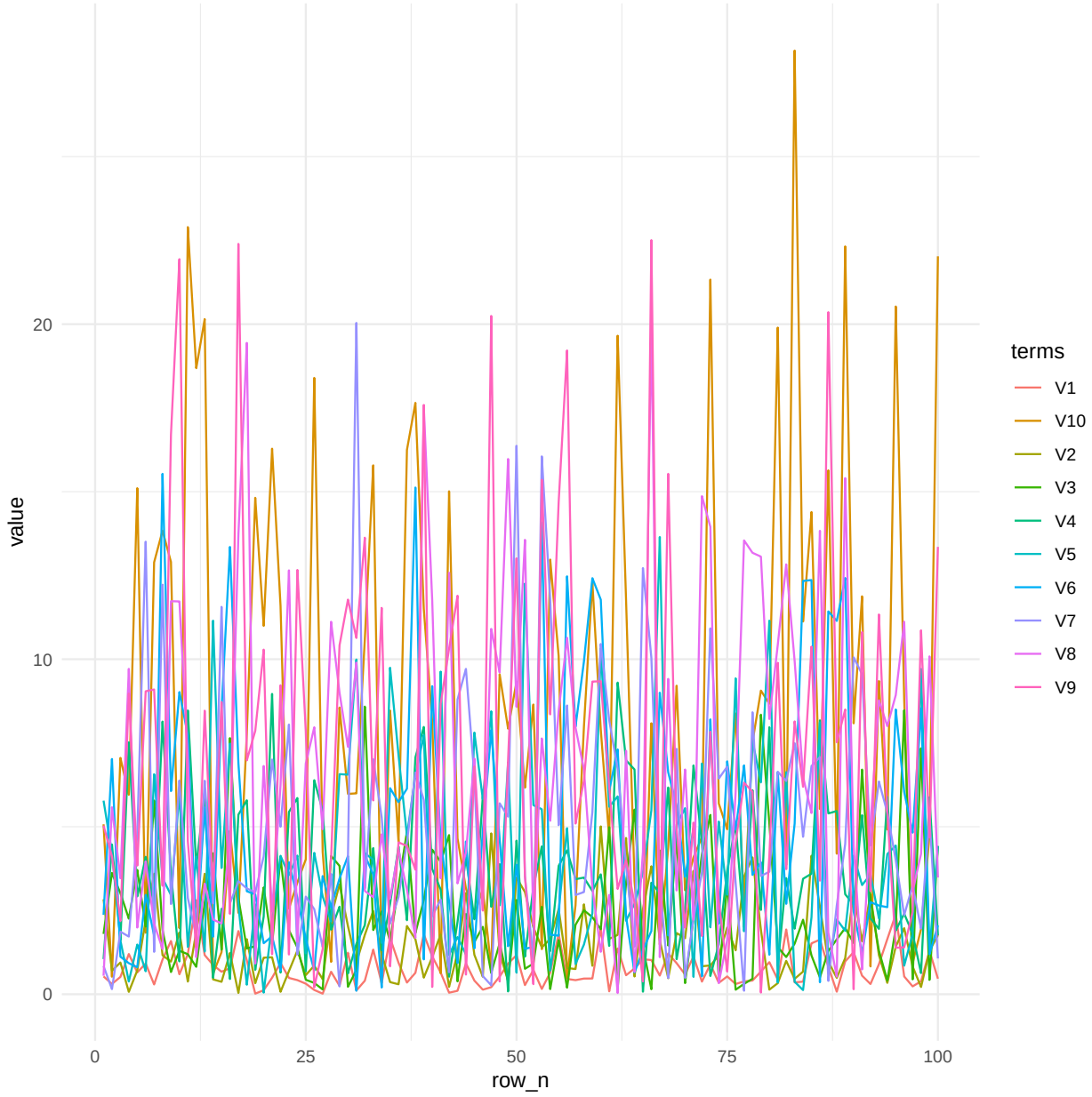
```
normaux_liste <- matrix(ncol = 10,nrow = 100)

for(i in 1:10){
normaux_liste[,i]<- rnorm(100,0,i)
}

normaux_df <- as.data.frame(normaux_liste)

normaux_df_2 <- normaux_df^2

names(normaux_df_2)
```

```
##  [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10"
```

```
normaux_df_2 %>%
   mutate(row_n = row_number())%>%
   pivot_longer(-row_n,names_to = "terms",values_to = "value") ->d_1



   d_1 %>%
   ggplot(mapping = aes(x = row_n , y = value,group = terms, color = terms))+
   geom_line()+
  expand_limits(x = 0 , y = 0)
```



```r
normaux_df <- abs(normaux_df)
normaux_df %>%
   mutate(row_n = row_number())%>%
   pivot_longer(-row_n,names_to = "terms",values_to = "value") ->d_2



   d_2%>%
   ggplot(mapping = aes(x = row_n , y = value,group = terms, color = terms))+
   geom_line()
```



Ceci montre pourquoi utiliser le RMSE avec un dataset qui a une grande dispersion est une mauvaise idée . Ici, le *0* est la vraie *valeur* et on change la variance qui est l'erreur, en d'autres mots on suppose que l'erreur suit une loi normale. On remarque le *RMSE* amplifie les erreurs vu qu'il met au carré les erreurs, ce qui

résulte un RMSE très élevé.

Ceci est différent pour le MAE.

Pour notre choix de modèle, nous allons regarder pour chaque modèle quels sont les *3 plus importantes variables* qu'ils utilisent pour prédire la variable *nb_sinistre*.

Ensuite, on calculera *la variance* de ces 3 variables et on choisira le *minimum* de ces valeurs.

L'idée est simple, si ces 3 variables varient trop, il y a des chances que si on prend une autre base de données l'échantillon ne sera pas représentatif à notre base de données.

Cependant si on prend un très grand échantillon , il n'y aura pas de problème .

En effet, on pourrait se demander si on utilise la fonction *sample* et on mesure la *variance* de chaque sample pour certaines variables quel sera la différence .

On essayera de vérifier ceci tout de suite , nous allons vérifier les 4 variables les plus importantes en générale.

```
var_liste <- matrix(ncol = 1 , nrow = 20)
var_liste_2 <- matrix(ncol = 1 , nrow = 20)
var_liste_3 <- matrix(ncol = 1 , nrow = 20)
var_liste_4 <- matrix(ncol = 1 , nrow = 20)


for(i in 1:20){
  var_smple <- sample(1:nrow(data_pred),5000,replace = F)
  var_df <- data_pred[var_smple,]
  var_liste[i,1] <- var(var_df$age_vehicule)
}


for(i in 1:20){
  var_smple <- sample(1:nrow(data_pred),5000,replace = F)
  var_df <- data_pred[var_smple,]
  var_liste_2[i,1] <- var(var_df$feature_1)
}

for(i in 1:20){
  var_smple <- sample(1:nrow(data_pred),5000,replace = F)
  var_df <- data_pred[var_smple,]
  var_liste_3[i,1] <- var(var_df$distance_conduite)
}

for(i in 1:20){
  var_smple <- sample(1:nrow(data_pred),5000,replace = F)
  var_df <- data_pred[var_smple,]
  var_liste_4[i,1] <- var(var_df$variable_75)
}
```

```
var_df <- data.frame(term = paste0("s","_",1:20) , age_vehicule = var_liste[,1] ,feature_1 = var_liste_
                     distance_conduite = var_liste_3[,1] , variable_75 = var_liste_4[,1])

var_df %>%
  pivot_longer(-term , names_to = "variable" , values_to = "valeur_variance")%>%
  ggplot(mapping = aes(x = term , y = valeur_variance , color =term , group = 1))+
  geom_line(size = 1.1)+
```

```
facet_wrap(~variable,scales = "free")+
labs(x = "" , y = "" , title = "Variance des variables importantes choisi par les modèles")+
expand_limits(y = 0)+
theme(plot.title = element_text(size = 16 , hjust = 0.5 , color  = "#1a5276"),
      legend.position = "",
      strip.background = element_rect(fill = "#d98880"),
      strip.text = element_text(size = 12))
```

Déjà on voit que la variable qui varie le moins est la variable *age_vehicule*, donc on choisira un des deux modèles *xgboost*. En effet, dans les deux xgboost la variable *age_vehicule* est celle qui varie le moins.

Ensuite, pour les deux xgboost entre *Linear* et *Tree*, nous allons choisir le *Linear*, car la différence est négligeable.