fréquence

2025-12-10

1) Chargement et préparation des données

```
df_accidents <- read.csv("../data/ONISR-2021.csv")
```

```
library(dplyr)
```

```
##
## Attachement du package : 'dplyr'
```

```
## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag
```

```
## Les objets suivants sont masqués depuis 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
df_frequence <- df_accidents %>%
  group_by(dep) %>%  # On regroupe tout par département
  summarise(
    #NOTRE VARIABLE Y (réponse)
    Nb_Accidents = n(),
    #Variables explicatives X

    Part_Pluie = mean(atm == "pluie forte" | atm == "pluie legere", na.rm = TRUE),

    Part_Autoroute = mean(catr == "autoroute", na.rm = TRUE),

    Part_Departementale = mean(catr == "departementale", na.rm = TRUE),


    Part_Hors_Agglo = mean(agg == "hors agglo", na.rm = TRUE),

    Part_Virage = mean(plan != "rectiligne", na.rm = TRUE),


    VMA_Moyenne = mean(as.numeric(vma), na.rm = TRUE),

    Taux_Mortalite = mean(nb_tue > 0, na.rm = TRUE)
  )
head(df_frequence)
```

```
## # A tibble: 6 x 9
##   dep   Nb_Accidents Part_Pluie Part_Autoroute Part_Departementale
##   <chr>        <int>      <dbl>          <dbl>               <dbl>
## 1 01             422     0.154          0.121               0.692
## 2 02             199     0.131          0.0653              0.578
```

```
## 3 03             230     0.130          0.0652               0.683
## 4 04             202     0.0396         0.0149               0.733
## 5 05             250     0.068          0                    0.472
## 6 06             941     0.0490         0.0584               0.294
## # i 4 more variables: Part_Hors_Agglo <dbl>, Part_Virage <dbl>,
## #   VMA_Moyenne <dbl>, Taux_Mortalite <dbl>
```

2) Le CSV pour l'exposition : Population

On a pris les données provenant de l'INSEE 2021 via cette adresse : https://www.insee.fr/fr/statistiques/7739582?sommaire=7728826

```r
df_pop_brut <- read.csv("donnees_departements_INSEE_2021.csv", sep = ";", stringsAsFactors = FALSE)

#On nettoie les données
df_pop <- df_pop_brut %>%
  dplyr::select(DEP, PTOT) %>%
  rename(dep = DEP, Population = PTOT) %>%
  mutate(dep = as.character(dep))
#On conserve uniquement la population et le code département,
#que l'on renomme et convertit en texte pour permettre la jointure.
```

3) Fusion des deux CSV

On crée un nouveau CSV qui contient toutes les données de base plus une colonne Population

```r
df_final <- inner_join(df_frequence, df_pop, by = "dep")
```

4) Tests : Modèle linéaire généarlisé

Premièrement, on essaye avec une Loi de poisson parce que La loi de Poisson est pertinente pour décrire le nombre d'événements dans d'autres types d'intervalles, spatiaux plutôt que temporels, comme des segments, surfaces ou volumes.

Nb_Accidents agit comme un "compteur"

De plus, on prend en compte l'exposition, d'après une formule qu'on a trouvé sur https://perso.univ-rennes1.fr/valerie.monbet/GLM/GLMpharma.pdf, il mentionne notamment (diapo 146/203) la présence d'un offset qui dans notre cas est la population (il nous dit exactement que : "*L'offset est fréquent dans les modèles log-linéaires. Il représente souvent le log d'une mesure d'exposition*"

(Cela explique pourquoi on a rajouté un CSV avec les données de la population en 2021 de l'INSEE

(*PS: le pdf mis ci-dessus nous a été très utile notamment toute la partie sur Régression de Poisson, dans lequel on retrouve notamment des propriétés notamment le calcule de la déviance permettant de mesurer l'écart entre le modèle et l'observation*)

4.1) Test pour une loi de Poisson

Avant toute chose, pour utiliser la régression de Poisson :

L'offset permet de modéliser un **taux d'accidents par habitant**, et non un nombre brut. Sans offset, les départements très peuplés auraient automatiquement plus d'accidents, ce qui biaiserait complètement le modèle

```r
# --- LE MODÈLE NUL (Référence) ---
#Il ne prend en compte que l'Offset de la population
modele_nul <- glm(Nb_Accidents ~ 1 + offset(log(Population)),
```

```
                  family = poisson,
                  data = df_final)
modele_nul
```

```
##
## Call:  glm(formula = Nb_Accidents ~ 1 + offset(log(Population)), family = poisson,
##     data = df_final)
##
## Coefficients:
## (Intercept)
##      -7.114
##
## Degrees of Freedom: 99 Total (i.e. Null);  99 Residual
## Null Deviance:       15670
## Residual Deviance: 15670     AIC: 16440
```

CI-dessous, notre modèle avec les variables explicatives

```
#Voici notre modèle
modele_poisson <- glm(Nb_Accidents ~ Part_Pluie + Part_Autoroute +
                      Part_Departementale + Part_Hors_Agglo +
                      Part_Virage + VMA_Moyenne +
                      offset(log(Population)),
                      family = poisson,
                      data = df_final)
modele_poisson
```

```
##
## Call:  glm(formula = Nb_Accidents ~ Part_Pluie + Part_Autoroute + Part_Departementale +
##     Part_Hors_Agglo + Part_Virage + VMA_Moyenne + offset(log(Population)),
##     family = poisson, data = df_final)
##
## Coefficients:
##         (Intercept)           Part_Pluie       Part_Autoroute
##            -3.74756              0.38972              2.54099
## Part_Departementale      Part_Hors_Agglo          Part_Virage
##            -0.06115             -0.04387             -0.43556
##         VMA_Moyenne
##            -0.05856
##
## Degrees of Freedom: 99 Total (i.e. Null);  93 Residual
## Null Deviance:       15670
## Residual Deviance: 6696  AIC: 7482
```

```
summary(modele_poisson)
```

```
##
## Call:
## glm(formula = Nb_Accidents ~ Part_Pluie + Part_Autoroute + Part_Departementale +
##     Part_Hors_Agglo + Part_Virage + VMA_Moyenne + offset(log(Population)),
##     family = poisson, data = df_final)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -3.74756    0.10332 -36.270  < 2e-16 ***
## Part_Pluie           0.38972    0.12890   3.024   0.0025 **
```

```
## Part_Autoroute       2.54099    0.07097  35.802  < 2e-16 ***
## Part_Departementale -0.06115    0.02901  -2.108   0.0350 *
## Part_Hors_Agglo     -0.04387    0.10326  -0.425   0.6709
## Part_Virage         -0.43556    0.10496  -4.150 3.33e-05 ***
## VMA_Moyenne         -0.05856    0.00234 -25.022  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 15668.2  on 99  degrees of freedom
## Residual deviance:  6695.8  on 93  degrees of freedom
## AIC: 7482.2
##
## Number of Fisher Scoring iterations: 4
```

$$D(y, \hat{\mu}) = 2 \sum_{i=1}^{n} \left[y_i \ln \left(\frac{y_i}{\hat{\mu}_i} \right) - (y_i - \hat{\mu}_i) \right]$$

Voici la formule de la Déviance, où

- y_i est le nb d'accidents **observé** (Réel) dans le département i

-$\hat{\mu}$ est le nb d'accident **estimé** (Prédit) par le modèle pour un département i

-n est le nb total d'observations

De plus, si on regarde le summary => On va trouver notamment le AIC : C'est un score qui arbitre entre la précision du modèle et sa simplicité, en pénalisant l'ajout de variables pour éviter le surapprentissage.

```
Deviance <- deviance(modele_poisson)

#Degré de liberté
Df <- df.residual(modele_poisson)

#Superposition

Phi <- Deviance / Df

round(Deviance,2)
```

```
## [1] 6695.8
```

```
round(Phi,2)
```

```
## [1] 72
```

```
print(paste("Déviance du Modèle Nul :", round(deviance(modele_nul), 2)))
```

```
## [1] "Déviance du Modèle Nul : 15668.18"
```

```
print(paste("Déviance de notre Modèle   :", round(deviance(modele_poisson), 2)))
```

```
## [1] "Déviance de notre Modèle   : 6695.8"
```

Ou bien on peut raisonner de la façon suivante :

on pose H_0 : le modèle permet de reproduire les obs ;

H_1 le modèle ne permet pas de reproduire les observations

On peut utiliser la statistique de Pearson qui vérifie une loi du Chi2 à n-p degré de liberté

```
# 1. Calcul de la statistique de Pearson (
res_pearson <- sum(residuals(modele_poisson, type = "pearson")^2)

# 2. Degrés de liberté
ddl <- df.residual(modele_poisson)

p_value <- pchisq(res_pearson, df = ddl, lower.tail = FALSE)

print(paste("Statistique de Pearson :", round(res_pearson, 2)))
```

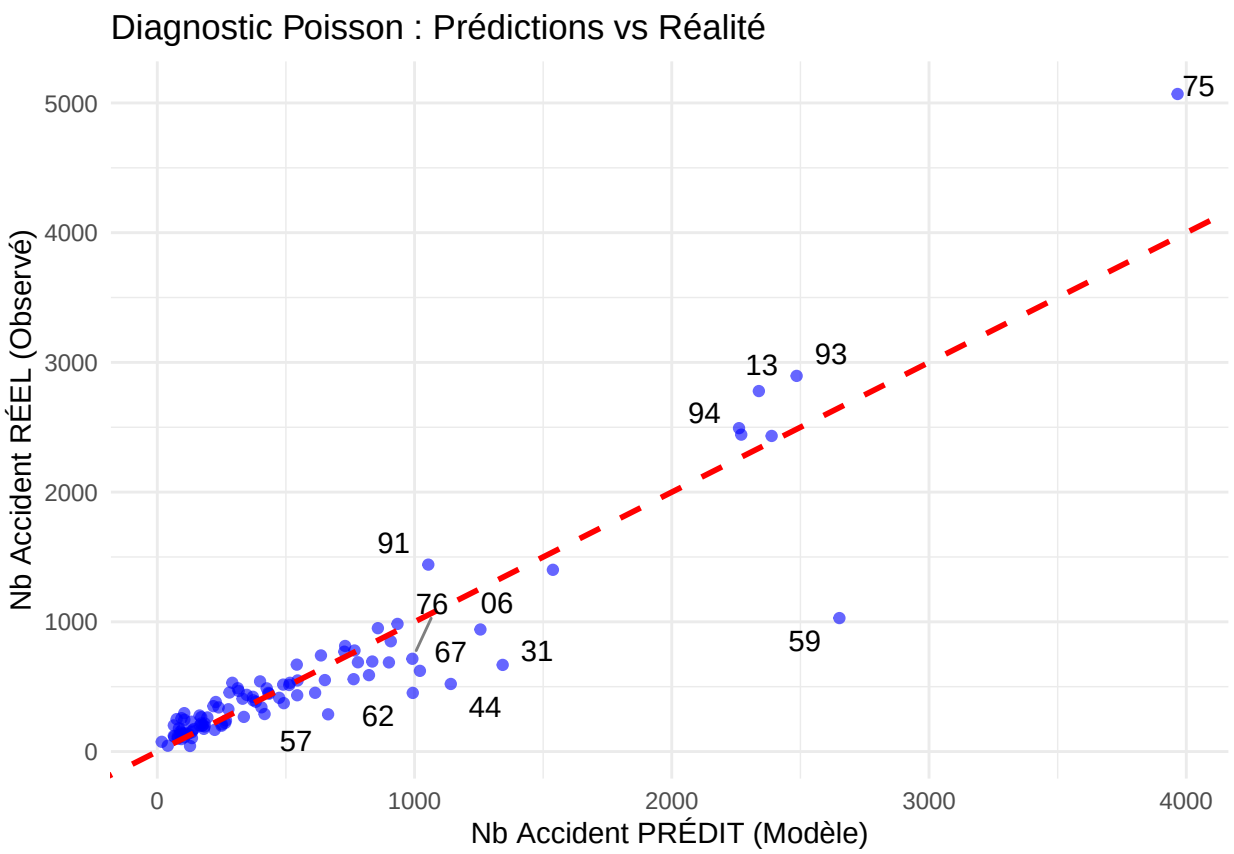
```
## [1] "Statistique de Pearson : 6887.66"
```

Ici, un apercu graphique de notre modèle de Poisson par rapport à la réalité:

```
library(ggplot2)
library(ggrepel) # Pour afficher les noms des départements sans chevauchement


df_final$Prediction_Poisson <- fitted(modele_poisson)

# 2. Le Graphe
ggplot(df_final, aes(x = Prediction_Poisson, y = Nb_Accidents)) +
  geom_point(alpha = 0.6, color = "blue") +

  # La ligne de perfection (y = x)
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed", size = 1) +

  geom_text_repel(aes(label = ifelse(abs(Nb_Accidents - Prediction_Poisson) > 200, dep, "")),
                  box.padding = 0.35, point.padding = 0.5, segment.color = 'grey50') +

  labs(title = "Diagnostic Poisson : Prédictions vs Réalité",
       x = "Nb Accident PRÉDIT (Modèle)",
       y = "Nb Accident RÉEL (Observé)") +
  theme_minimal()
```

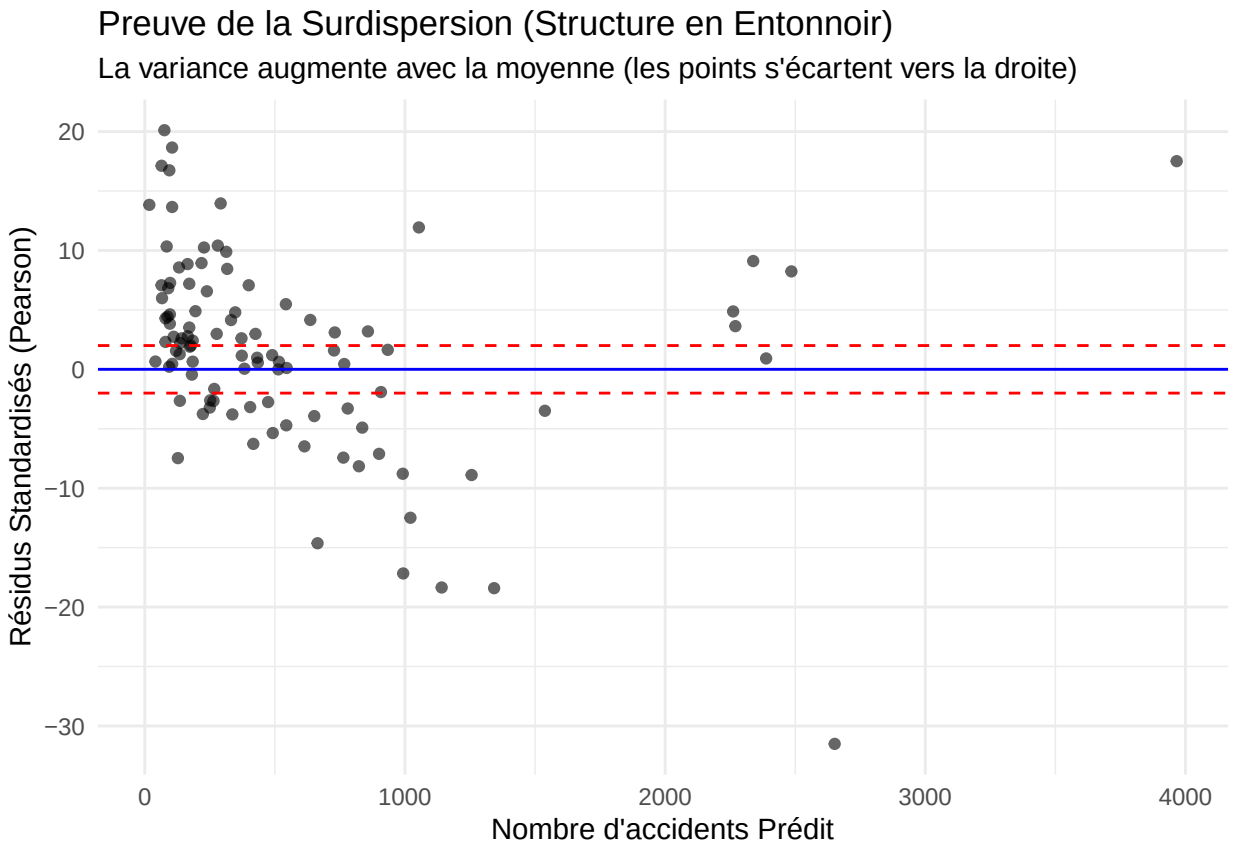
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: ggrepel: 4 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```r
library(dplyr)
```

```r
#Visuel des résidus
df_final$Residus_Pearson <- residuals(modele_poisson, type = "pearson")

ggplot(df_final, aes(x = Prediction_Poisson, y = Residus_Pearson)) +
  geom_point(alpha = 0.6) +

  # Lignes de tolérance théorique (-2 et +2 écarts-types)
  geom_hline(yintercept = c(-2, 2), color = "red", linetype = "dashed") +
  geom_hline(yintercept = 0, color = "blue") +

  # Si on voit des points très loin (ex: > 5 ou < -5), c'est de la surdispersion
  labs(title = "Preuve de la Surdispersion (Structure en Entonnoir)",
       subtitle = "La variance augmente avec la moyenne (les points s'écartent vers la droite)",
       x = "Nombre d'accidents Prédit",
       y = "Résidus Standardisés (Pearson)") +
  theme_minimal()
```



Ainsi, on voit que le modèle de Poisson ne prédit pas correctement la fréquence des accidents la défiance étant très élevé le Chi-test étant pas en adéquation avec ce que l'on recherche.

On a donc décidé de continuer la démarche établi par le PDF en utilisant un modèle de Régression binomiale négative. En effet, le modèle binomiale négative va surement permettre de réduire cet effet de surdispersion

4.2) Test pour modèle binomiale négative

```
library(MASS) # Pour utiliser glm.nb()
```

```
##
## Attachement du package : 'MASS'
```

```
## L'objet suivant est masqué depuis 'package:dplyr':
##
##     select
```

```
modele_nb <- glm.nb(Nb_Accidents ~ Part_Pluie + Part_Autoroute +
                    Part_Departementale + Part_Hors_Agglo +
                    Part_Virage + VMA_Moyenne +
                    offset(log(Population)),
                    data = df_final)

summary(modele_nb)
```

```
##
## Call:
## glm.nb(formula = Nb_Accidents ~ Part_Pluie + Part_Autoroute +
##     Part_Departementale + Part_Hors_Agglo + Part_Virage + VMA_Moyenne +
```

```
##     offset(log(Population)), data = df_final, init.theta = 7.852170144,
##     link = log)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -5.56931    0.67989  -8.192 2.58e-16 ***
## Part_Pluie          -0.73535    1.03665  -0.709  0.47811
## Part_Autoroute       1.79415    0.73895   2.428  0.01518 *
## Part_Departementale -0.35670    0.30645  -1.164  0.24443
## Part_Hors_Agglo     -1.28635    0.69241  -1.858  0.06320 .
## Part_Virage          1.81674    0.68701   2.644  0.00818 **
## VMA_Moyenne         -0.02169    0.01539  -1.410  0.15868
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(7.8522) family taken to be 1)
##
##     Null deviance: 159.09  on 99  degrees of freedom
## Residual deviance: 102.09  on 93  degrees of freedom
## AIC: 1278
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  7.85
##           Std. Err.:  1.12
##
##  2 x log-likelihood:  -1262.037
```

A présent, comparons nos deux modèles. On va comparer les AIC et extraire les coefficents du modèle le plus efficace.

```
#On compare les AIC

aic_poisson <- AIC(modele_poisson)
aic_nb      <- AIC(modele_nb)

cat("AIC Poisson            :", round(aic_poisson, 2), "\n")
```

```
## AIC Poisson            : 7482.19
```

```
cat("AIC Binomiale Négative :", round(aic_nb, 2), "\n")
```

```
## AIC Binomiale Négative : 1278.04
```

On voit que le modèle Binomiale Négative est plus intéressant car l'AIC est nettement plus petit

```
coefficients_binomiaux_négative <- exp(coef(modele_nb))

print(round(coefficients_binomiaux_négative, 3))
```

```
##         (Intercept)          Part_Pluie      Part_Autoroute Part_Departementale
##               0.004               0.479               6.014               0.700
##     Part_Hors_Agglo         Part_Virage         VMA_Moyenne
##               0.276               6.152               0.979
```

AInsi, on peut interpréter cela de la façon suivante :

Virage & Autoroute : Facteurs critiques majeurs, ils **multiplient par 6** le nombre d'accidents.

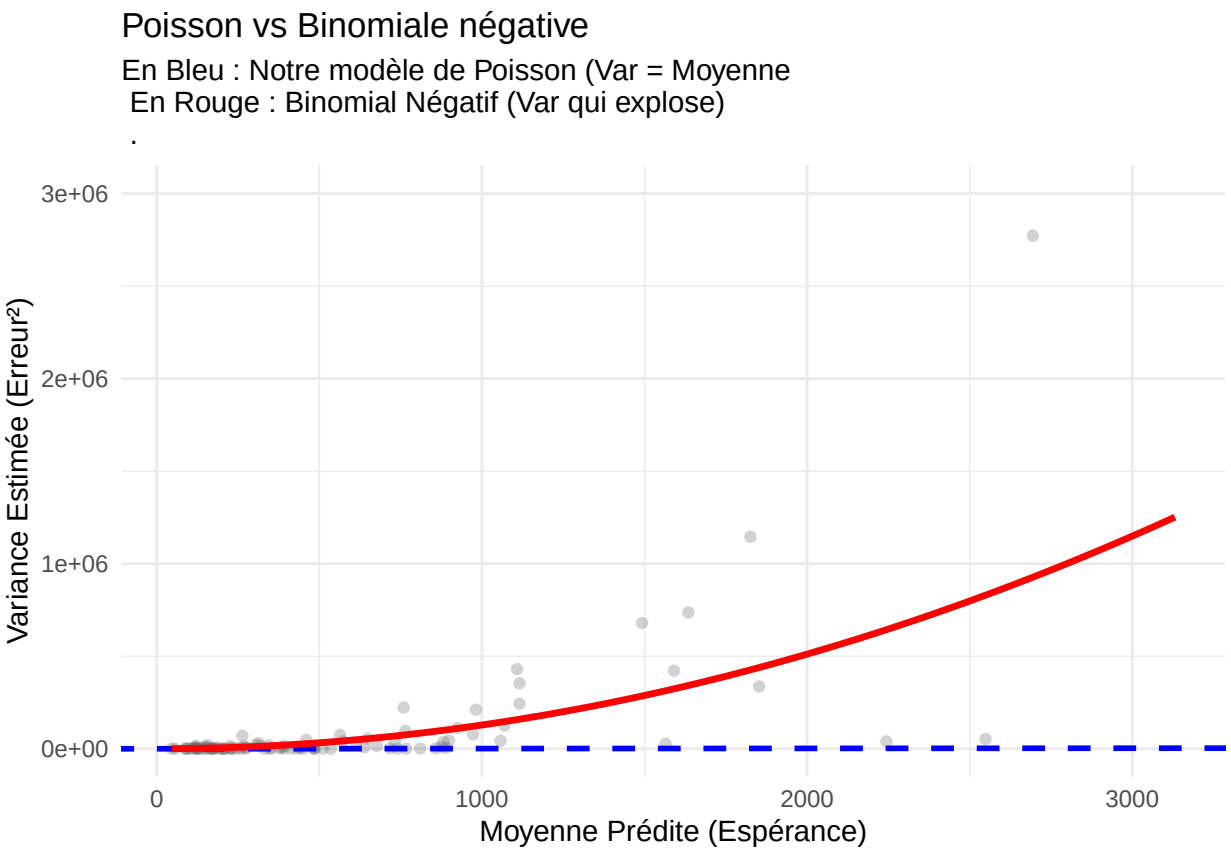
Hors-Agglo : Le plus protecteur, il **divise par 4** le risque par rapport à la ville.

Pluie : Protecteur paradoxal (prudence accrue), elle **divise par 2** les accidents.

Départementale : Réduit modérément le risque (**-30%**).

VMA : Impact négligeable (**-2%**).

```r
theta <- modele_nb$theta

mu <- fitted(modele_nb)

# On estime la variance observée "brute" par le carré de l'erreur
var_obs <- (df_final$Nb_Accidents - mu)^2

data_comparaison <- data.frame(Moyenne = mu, Variance_Obs = var_obs)

# 3. Le Graphique du Duel
ggplot(data_comparaison, aes(x = Moyenne, y = Variance_Obs)) +
  # A. Les points réels (L'erreur observée)
  geom_point(alpha = 0.3, color = "grey40") +

  # Poisson
  geom_abline(intercept = 0, slope = 1, color = "blue", size = 1, linetype = "dashed") +

  # Binomiale Négative
  stat_function(fun = function(x) x + (x^2 / theta),
                color = "red", size = 1.2) +

  labs(title = "Poisson vs Binomiale négative",
       subtitle = "En Bleu : Notre modèle de Poisson (Var = Moyenne\n En Rouge : Binomial Négatif (Var
       x = "Moyenne Prédite (Espérance)",
       y = "Variance Estimée (Erreur²)") +
  theme_minimal() +
  coord_cartesian(ylim = c(0, max(var_obs)*0.8)) # Uniquement pour zoomer pour rien oublier
```



Ainsi, graphiquement, on constate que cela a plus de sens car nos petits points gris (les erreurs de prédiction au carré pour chaque département) suivent plus la droite rouge que la droite bleu et donc c'est cohérent que le modèle Binomiale Négative