Filtre tes connaissances

Conception de filtres pour le réduction du bruit du capteur de distance

Daniel Lavallée

Mathieu Sévégny

Tristan Lafontaine

Émile Bois

Vincent Kilaknowvski

2024-10-31

Table of Contents

[Configuration de l’environnement 1](#_Toc181476882)

[Lecture des données 1](#_Toc181476883)

[Nettoyage des données 2](#_Toc181476884)

[Visualisation des données 2](#_Toc181476885)

[Conception d’un filtre Butterworth 4](#_Toc181476886)

[Conception d’un filtre Chebyshev de type 1 8](#_Toc181476887)

[Conception d’un filtre Chebyshev de type 2 13](#_Toc181476888)

[Conception d’un filtre elliptique 18](#_Toc181476889)

[Création d’un modèle de régression linéaire sans filtre 23](#_Toc181476890)

[Création d’un modèle de moyenne mobile simple d’order 3 25](#_Toc181476891)

[Création d’un modèle de moyenne mobile simple d’ordre 5 27](#_Toc181476892)

[Création d’un modèle Knn 30](#_Toc181476893)

[Comparaison des modèles 32](#_Toc181476894)

# Configuration de l’environnement

## Lecture des données

## Fonction qui vérifie si un packet est installé et qui l'installe avant   
## de le charger au besoin.  
loadPackage <- function(package) {  
 if (!require(package, character.only = TRUE)) {  
 install.packages(package, quiet = TRUE)  
 library(package, character.only = TRUE, quietly = TRUE)  
 }  
 else library(package, character.only = TRUE, quietly = TRUE)  
}

loadPackage("tidyverse")  
loadPackage("ggplot2")  
loadPackage("readxl")  
loadPackage("Metrics")  
loadPackage("signal")  
loadPackage("TTR")  
loadPackage("caret")

Lecture des données du capteur de distance (données réelle vs données bruitées).

## Chemin vers les données  
path\_donnees\_obstacle <- "../data/donnees\_detecteur\_obstacle(more).csv"  
  
## Lecture des données  
donnees\_detecteur\_obstacle <- read\_csv(path\_donnees\_obstacle)  
donnees\_detecteur\_obstacle <- donnees\_detecteur\_obstacle %>%  
 rename(real\_distance\_cm = `Sample Name`)  
  
## dt moyen  
dt\_moyen <- mean(donnees\_detecteur\_obstacle$dt)

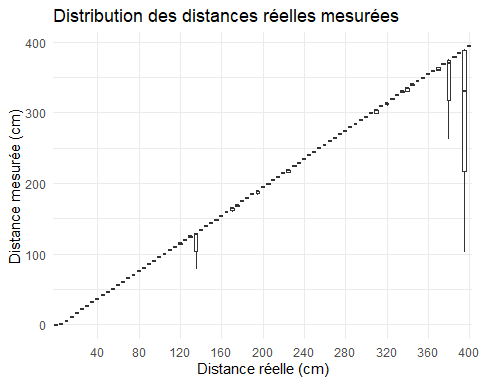
# Nettoyage des données

## Suppression des colonnes inutiles et transformation des données  
donnees\_detecteur\_obstacle <- donnees\_detecteur\_obstacle %>%  
 pivot\_longer(cols = starts\_with("Sample"), names\_to = "Sample\_No", values\_to = "measured\_distance\_cm") %>%  
 select(., -dt, -Sample\_No)  
  
## Aggrégation des données par la moyenne des distances mesurées  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle %>%  
 group\_by(real\_distance\_cm) %>%  
 summarise(measured\_distance\_cm = mean(measured\_distance\_cm, na.rm = TRUE))

# Visualisation des données

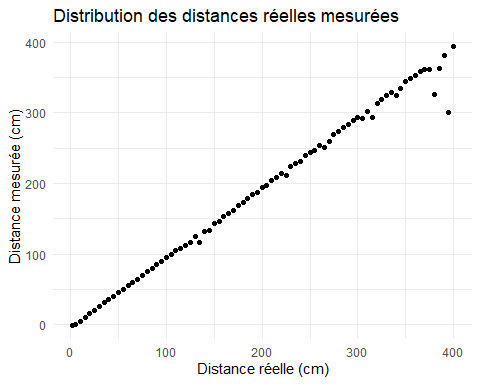
Distruibution des données bruitées.

## Distribution des données  
ggplot(donnees\_detecteur\_obstacle, aes(x = factor(real\_distance\_cm), y = measured\_distance\_cm)) +  
 geom\_boxplot(outlier.shape = NA) +  
 labs(title = "Distribution des distances réelles mesurées",  
 x = "Distance réelle (cm)",  
 y = "Distance mesurée (cm)") +  
 scale\_x\_discrete(breaks = seq(0, 400, by = 8)) +  
 theme\_minimal()



Données agrégées par la moyenne des distances mesurées.

## Distribution des données agrégées  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = real\_distance\_cm, y = measured\_distance\_cm)) +  
 geom\_point() +  
 labs(title = "Distribution des distances réelles mesurées",  
 x = "Distance réelle (cm)",  
 y = "Distance mesurée (cm)") +  
 theme\_minimal()



# Conception d’un filtre Butterworth

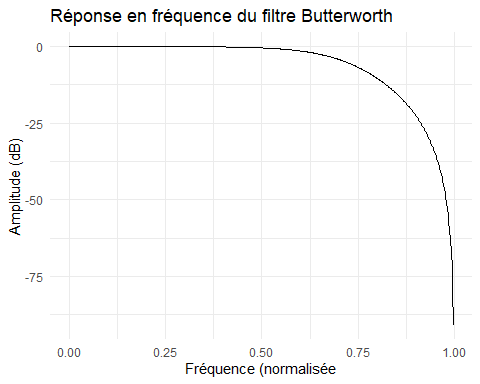
fe <- 1/dt\_moyen # Fréquence d'échantillonnage  
fc <- fe/3 # Fréquence de coupure  
order\_butter <- 2 # Ordre du filtre  
cutoff <- fc/(fe/2) # Fréquence de coupure normalisée  
  
# Conception du filtre Butterworth  
butter\_filter <- butter(order\_butter, cutoff, type = "low")  
  
# Calcul de la réponse en fréquence  
freq\_response <- freqz(butter\_filter, Fs = 2) # Fs = 2 pour des fréquences normalisées (entre 0 et 1)  
  
## Affichage des coefficients du filtre Butterworth  
print(paste("Le numérateur du filtre Butterworth est : ", butter\_filter$b))

## [1] "Le numérateur du filtre Butterworth est : 0.465153077165047"  
## [2] "Le numérateur du filtre Butterworth est : 0.930306154330093"  
## [3] "Le numérateur du filtre Butterworth est : 0.465153077165047"

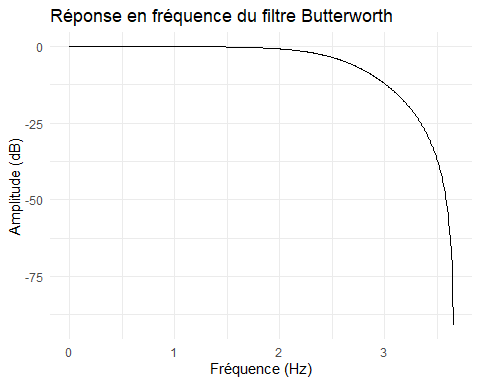
print(paste("Le dénominateur du filtre Butterworth est : ", butter\_filter$a))

## [1] "Le dénominateur du filtre Butterworth est : 1"   
## [2] "Le dénominateur du filtre Butterworth est : 0.620204102886728"  
## [3] "Le dénominateur du filtre Butterworth est : 0.240408205773457"

## Création d'un data frame pour la réponse en fréquence  
# Fréquence normalisée  
freq\_response\_butter <- data.frame(f = freq\_response$f,  
 h = abs(freq\_response$h))  
# Fréquence en Hz  
freq\_response\_butter\_hz <- data.frame(f = freq\_response$f \* (fe/2),  
 h = abs(freq\_response$h))  
  
## Tracer la réponse en fréquence (normalisée)  
ggplot(freq\_response\_butter, aes(x = f, y = 20 \* log10(h))) +  
 geom\_line() +  
 xlab("Fréquence (normalisée") +  
 ylab("Amplitude (dB)") +  
 ggtitle("Réponse en fréquence du filtre Butterworth") +  
 theme\_minimal()



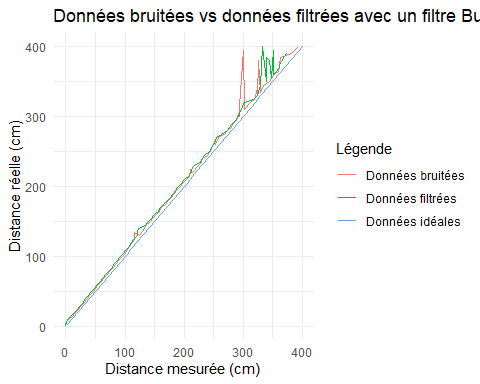
## Tracer la réponse en fréquence (en Hz)  
ggplot(freq\_response\_butter\_hz, aes(x = f, y = 20 \* log10(h))) +  
 geom\_line() +  
 xlab("Fréquence (Hz)") +  
 ylab("Amplitude (dB)") +  
 ggtitle("Réponse en fréquence du filtre Butterworth") +  
 theme\_minimal()



L’équation du **filtre de Butterworth** est alors donnée par :

Appliquons le filtrage sur les données bruitées.

# Filtrage des données bruitées  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(measured\_distance\_cm\_butter = signal::filter(butter\_filter, measured\_distance\_cm))  
  
## Tracer les données bruitées vs les données filtrées  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = measured\_distance\_cm\_butter, y = real\_distance\_cm, col = "Données filtrées")) +  
 labs(title = "Données bruitées vs données filtrées avec un filtre Butterworth",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 theme\_minimal()

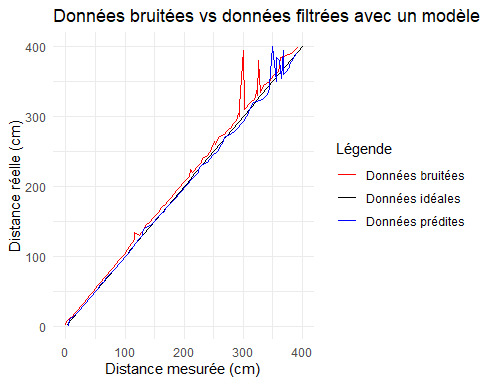


## Création du modèle de régression linéaire  
modele\_butter <- lm(real\_distance\_cm ~ measured\_distance\_cm\_butter, data = donnees\_detecteur\_obstacle\_aggregated)  
  
## Prédiction des valeur réelles avec le filtre de Butterworth  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(modele\_butter = predict(modele\_butter))

L’équation du **modèle de régression linéaire** est alors donnée par :

Voici le modèle de prévision pour la régression linéaire avec le filtre Butterworth.

## Tracer les données bruitées vs les données prédites par le modèle de régression linéaire  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = modele\_butter, y = real\_distance\_cm, col = "Données prédites")) +  
 labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 scale\_color\_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédites" = "blue")) +  
 theme\_minimal()



# Conception d’un filtre Chebyshev de type 1

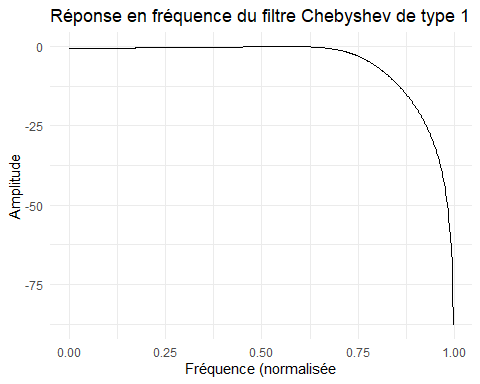
order\_chebyshev1 <- 2 # Ordre du filtre Chebyshev de type 1  
ripple <- 0.5 # Ondulation maximale en dB  
  
# Conception du filtre Chebyshev de type 1  
chebyshev1\_filter <- cheby1(order\_chebyshev1, ripple, cutoff, type = "low")  
  
# Affichage des coefficients du filtre Chebyshev de type 1  
print(paste("Le numérateur du filtre Chebyshev de type 1 est : ", chebyshev1\_filter$b))

## [1] "Le numérateur du filtre Chebyshev de type 1 est : 0.535574540620439"  
## [2] "Le numérateur du filtre Chebyshev de type 1 est : 1.07114908124088"   
## [3] "Le numérateur du filtre Chebyshev de type 1 est : 0.535574540620439"

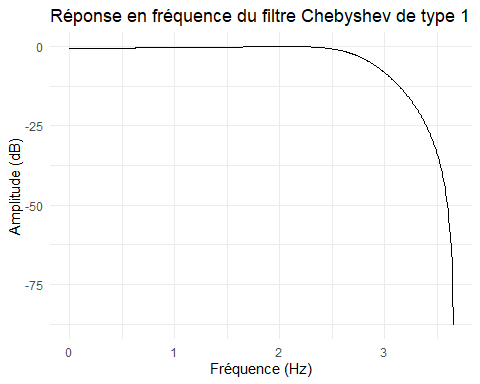
print(paste("Le dénominateur du filtre Chebyshev de type 1 est : ", chebyshev1\_filter$a))

## [1] "Le dénominateur du filtre Chebyshev de type 1 est : 1"   
## [2] "Le dénominateur du filtre Chebyshev de type 1 est : 0.885175597605059"  
## [3] "Le dénominateur du filtre Chebyshev de type 1 est : 0.3840617114442"

# Réponse en fréquence  
freq\_response\_chebyshev1 <- freqz(chebyshev1\_filter, Fs = 2)  
freq\_response\_chebyshev1 <- data.frame(f = freq\_response\_chebyshev1$f,  
 h = abs(freq\_response\_chebyshev1$h))  
freq\_response\_chebyshev1\_hz <- data.frame(f = freq\_response\_chebyshev1$f \* (fe/2),  
 h = abs(freq\_response\_chebyshev1$h))  
  
## Tracer la réponse en fréquence (normalisée)  
ggplot(freq\_response\_chebyshev1, aes(x = f, y = 20 \* log10(h))) +  
 geom\_line() +  
 xlab("Fréquence (normalisée") +  
 ylab("Amplitude") +  
 ggtitle("Réponse en fréquence du filtre Chebyshev de type 1") +  
 theme\_minimal()



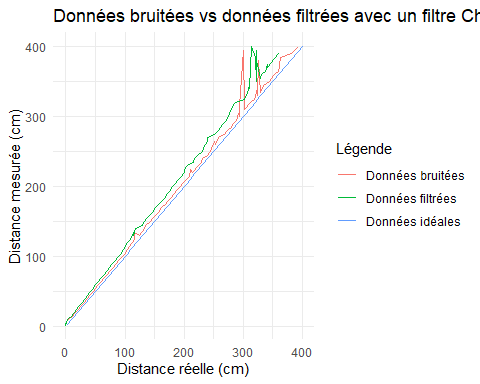
## Tracer la réponse en fréquence (en Hz)  
ggplot(freq\_response\_chebyshev1\_hz, aes(x = f, y = 20 \* log10(h))) +  
 geom\_line() +  
 xlab("Fréquence (Hz)") +  
 ylab("Amplitude (dB)") +  
 ggtitle("Réponse en fréquence du filtre Chebyshev de type 1") +  
 theme\_minimal()



L’équation du **filtre de Chebyshev de type 1** est alors donnée par :

Appliquons maintenant le filtre de Chebyshev de type 1 sur les données bruitées.

# Filtrage des données bruitées  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(measured\_distance\_cm\_chebyshev1 = signal::filter(chebyshev1\_filter, measured\_distance\_cm))  
  
## Tracer les données bruitées vs les données filtrées  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = measured\_distance\_cm\_chebyshev1, y = real\_distance\_cm, col = "Données filtrées")) +  
 labs(title = "Données bruitées vs données filtrées avec un filtre Chebyshev de type 1",  
 x = "Distance réelle (cm)",  
 y = "Distance mesurée (cm)",  
 color = "Légende") +  
 theme\_minimal()

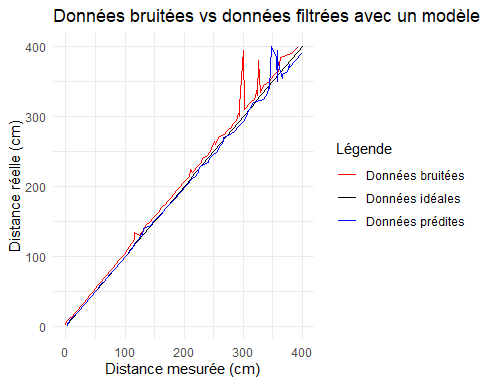


## Création du modèle de régression linéaire avec le filtre de Chebyshev de type 1  
modele\_chebyshev1 <- lm(real\_distance\_cm ~ measured\_distance\_cm\_chebyshev1, data = donnees\_detecteur\_obstacle\_aggregated)  
  
## Prédiction des valeur réelles avec le filtre de Chebyshev de type 1  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(modele\_chebyshev1 = predict(modele\_chebyshev1))

L’équation du **modèle de régression linéaire** est alors donnée par :

Voici le modèle de prévision pour la régression linéaire avec le filtre Chebyshev de type 1.

## Tracer les données bruitées vs les données prédites par le modèle de régression linéaire  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = modele\_chebyshev1, y = real\_distance\_cm, col = "Données prédites")) +  
 labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 scale\_color\_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédites" = "blue")) +  
 theme\_minimal()



# Conception d’un filtre Chebyshev de type 2

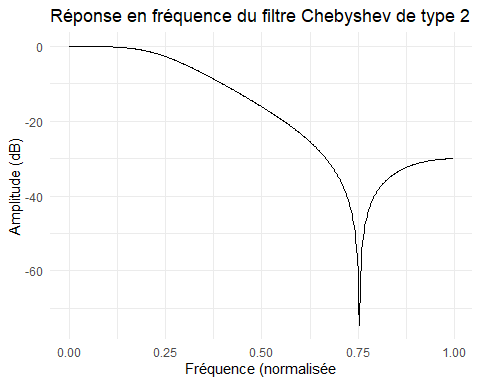
order\_chebyshev2 <- 2 # Ordre du filtre Chebyshev de type 2  
stopband\_attenuation <- 30 # Atténuation du bande d'arrêt en dB  
  
# Conception du filtre Chebyshev de type 2  
chebyshev2\_filter <- cheby2(order\_chebyshev2, stopband\_attenuation, cutoff, type = "low")  
  
# Affichage des coefficients du filtre Chebyshev de type 2  
print(paste("Le numérateur du filtre Chebyshev de type 2 est : ", chebyshev2\_filter$b))

## [1] "Le numérateur du filtre Chebyshev de type 2 est : 0.123255993256761"  
## [2] "Le numérateur du filtre Chebyshev de type 2 est : 0.176079990366802"  
## [3] "Le numérateur du filtre Chebyshev de type 2 est : 0.123255993256761"

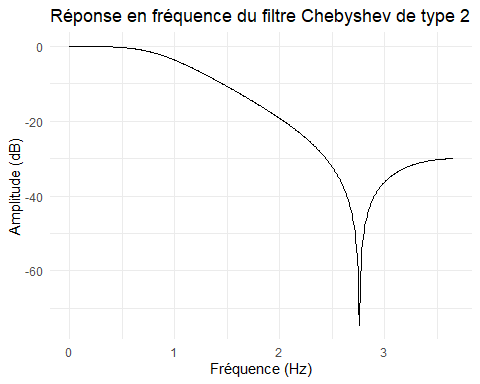
print(paste("Le dénominateur du filtre Chebyshev de type 2 est : ", chebyshev2\_filter$a))

## [1] "Le dénominateur du filtre Chebyshev de type 2 est : 1"   
## [2] "Le dénominateur du filtre Chebyshev de type 2 est : -0.902331651439041"  
## [3] "Le dénominateur du filtre Chebyshev de type 2 est : 0.324923628319367"

# Réponse en fréquence  
freq\_response\_chebyshev2 <- freqz(chebyshev2\_filter, Fs = 2)  
freq\_response\_chebyshev2 <- data.frame(f = freq\_response\_chebyshev2$f,  
 h = abs(freq\_response\_chebyshev2$h))  
freq\_response\_chebyshev2\_hz <- data.frame(f = freq\_response\_chebyshev2$f \* (fe/2),  
 h = abs(freq\_response\_chebyshev2$h))  
  
## Tracer la réponse en fréquence (normalisée)  
ggplot(freq\_response\_chebyshev2, aes(x = f, y = 20 \* log10(h))) +  
 geom\_line() +  
 xlab("Fréquence (normalisée") +  
 ylab("Amplitude (dB)") +  
 ggtitle("Réponse en fréquence du filtre Chebyshev de type 2") +  
 theme\_minimal()



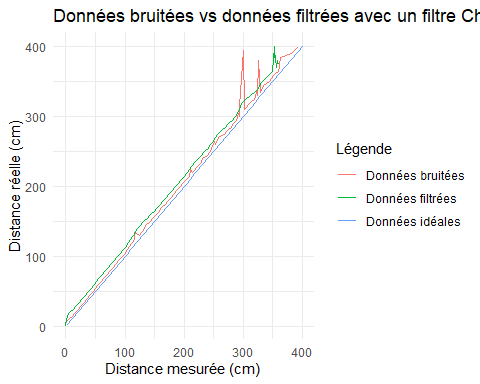
## Tracer la réponse en fréquence (en Hz)  
ggplot(freq\_response\_chebyshev2\_hz, aes(x = f, y = 20 \* log10(h))) +  
 geom\_line() +  
 xlab("Fréquence (Hz)") +  
 ylab("Amplitude (dB)") +  
 ggtitle("Réponse en fréquence du filtre Chebyshev de type 2") +  
 theme\_minimal()



L’équation du **filtre de Chebyshev de type 2** est alors donnée par :

Appliquons maintenant le filtre de Chebyshev de type 2 sur les données bruitées.

# Filtrage des données bruitées  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(measured\_distance\_cm\_chebyshev2 = signal::filter(chebyshev2\_filter, measured\_distance\_cm))  
  
## Tracer les données bruitées vs les données filtrées  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = measured\_distance\_cm\_chebyshev2, y = real\_distance\_cm, col = "Données filtrées")) +  
 labs(title = "Données bruitées vs données filtrées avec un filtre Chebyshev de type 2",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 theme\_minimal()

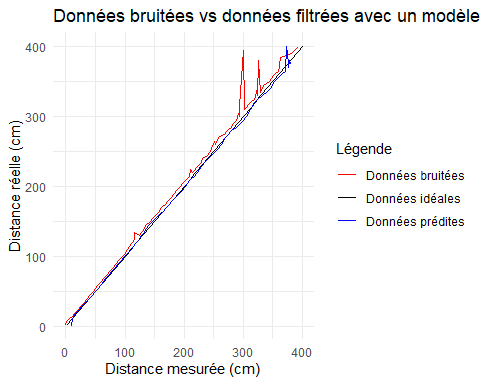


## Création du modèle de régression linéaire avec le filtre de Chebyshev de type 2  
modele\_chebyshev2 <- lm(real\_distance\_cm ~ measured\_distance\_cm\_chebyshev2, data = donnees\_detecteur\_obstacle\_aggregated)  
  
## Prédiction des valeur réelles avec le filtre de Chebyshev de type 2  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(modele\_chebyshev2 = predict(modele\_chebyshev2))

L’équation du **modèle de régression linéaire** est alors donnée par :

Voici le modèle de prévision pour la régression linéaire avec le filtre Chebyshev de type 2.

## Tracer les données bruitées vs les données prédites par le modèle de régression linéaire  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = modele\_chebyshev2, y = real\_distance\_cm, col = "Données prédites")) +  
 labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 scale\_color\_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédites" = "blue")) +  
 theme\_minimal()



# Conception d’un filtre elliptique

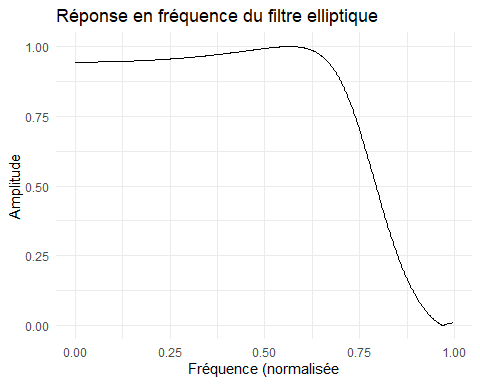
order\_elliptique <- 2 # Ordre du filtre elliptique  
ripple\_passband <- 0.5 # Ondulation maximale en dB dans la bande de passage  
stopband\_attenuation\_elliptique <- 40 # Atténuation du bande d'arrêt en dB  
  
# Conception du filtre elliptique  
elliptic\_filter <- ellip(order\_elliptique, ripple\_passband, stopband\_attenuation\_elliptique, cutoff, type = "low")  
  
# Affichage des coefficients du filtre elliptique  
print(paste("Le numérateur du filtre elliptique est : ", elliptic\_filter$b))

## [1] "Le numérateur du filtre elliptique est : 0.538503349959469"  
## [2] "Le numérateur du filtre elliptique est : 1.07201963425761"   
## [3] "Le numérateur du filtre elliptique est : 0.53850334995947"

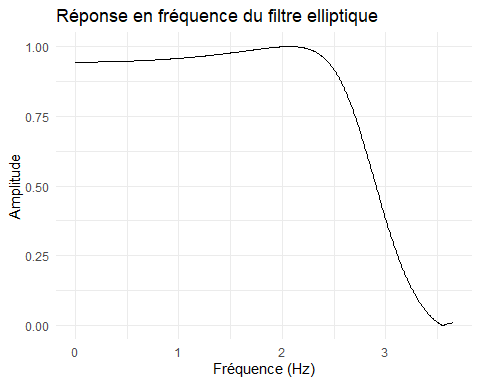
print(paste("Le dénominateur du filtre elliptique est : ", elliptic\_filter$a))

## [1] "Le dénominateur du filtre elliptique est : 1"   
## [2] "Le dénominateur du filtre elliptique est : 0.888820394688326"  
## [3] "Le dénominateur du filtre elliptique est : 0.387543755292276"

# Réponse en fréquence  
freq\_response\_elliptic <- freqz(elliptic\_filter, Fs = 2)  
freq\_response\_elliptic <- data.frame(f = freq\_response\_elliptic$f,  
 h = abs(freq\_response\_elliptic$h))  
freq\_response\_elliptic\_hz <- data.frame(f = freq\_response\_elliptic$f \* (fe/2),  
 h = abs(freq\_response\_elliptic$h))  
  
## Tracer la réponse en fréquence (normalisée)  
ggplot(freq\_response\_elliptic, aes(x = f, y = h)) +  
 geom\_line() +  
 xlab("Fréquence (normalisée") +  
 ylab("Amplitude") +  
 ggtitle("Réponse en fréquence du filtre elliptique") +  
 theme\_minimal()



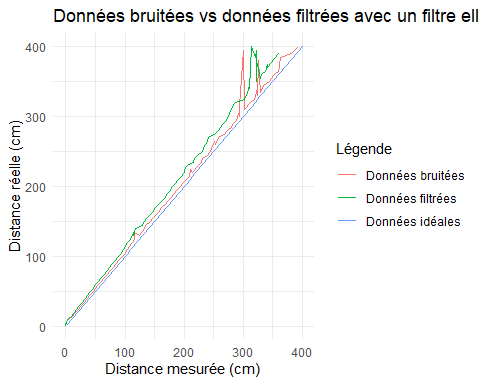
## Tracer la réponse en fréquence (en Hz)  
ggplot(freq\_response\_elliptic\_hz, aes(x = f, y = h)) +  
 geom\_line() +  
 xlab("Fréquence (Hz)") +  
 ylab("Amplitude") +  
 ggtitle("Réponse en fréquence du filtre elliptique") +  
 theme\_minimal()



L’équation du **filtre elliptique** est alors donnée par :

Appliquons maintenant le filtre elliptique sur les données bruitées.

# Filtrage des données bruitées  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(measured\_distance\_cm\_elliptic = signal::filter(elliptic\_filter, measured\_distance\_cm))  
  
## Tracer les données bruitées vs les données filtrées  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = measured\_distance\_cm\_elliptic, y = real\_distance\_cm, col = "Données filtrées")) +  
 labs(title = "Données bruitées vs données filtrées avec un filtre elliptique",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 theme\_minimal()

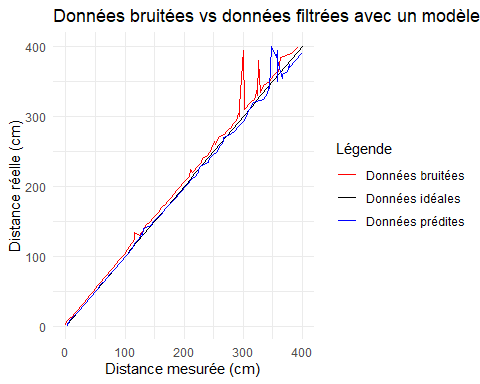


## Création du modèle de régression linéaire avec le filtre elliptique  
modele\_elliptic <- lm(real\_distance\_cm ~ measured\_distance\_cm\_elliptic, data = donnees\_detecteur\_obstacle\_aggregated)  
  
## Prédiction des valeur réelles avec le filtre elliptique  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(modele\_elliptic = predict(modele\_elliptic))

L’équation du **modèle de régression linéaire** est alors donnée par :

Voici le modèle de prévision pour la régression linéaire avec le filtre elliptique.

## Tracer les données bruitées vs les données prédites par le modèle de régression linéaire  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = modele\_elliptic, y = real\_distance\_cm, col = "Données prédites")) +  
 labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 scale\_color\_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédites" = "blue")) +  
 theme\_minimal()



# Création d’un modèle de régression linéaire sans filtre

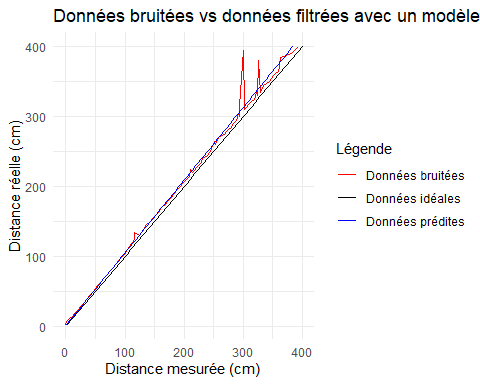
modele\_regression\_lineaire <- lm(measured\_distance\_cm ~ real\_distance\_cm, data = donnees\_detecteur\_obstacle\_aggregated)  
  
modele\_regression\_lineaire$coefficients

## (Intercept) real\_distance\_cm   
## -1.6671026 0.9638523

L’équation du **modèle de régression linéaire** est alors donnée par :

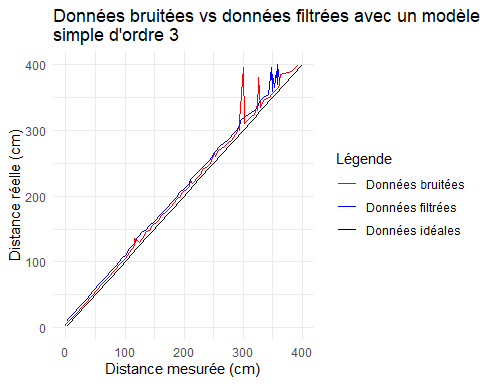
Appliquons maintenant le modèle de régression linéaire sur les données bruitées.

donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(modele\_regression\_lineaire = predict(modele\_regression\_lineaire))  
  
## Tracer les données bruitées vs les données filtrées  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = modele\_regression\_lineaire, y = real\_distance\_cm, col = "Données prédites")) +  
 labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 scale\_color\_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédites" = "blue")) +  
 theme\_minimal()



# Création d’un modèle de moyenne mobile simple d’order 3

modele\_moyenne\_mobile\_simple\_3 <- SMA(donnees\_detecteur\_obstacle\_aggregated$measured\_distance\_cm, n = 3)  
  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(modele\_moyenne\_mobile\_simple\_3 = modele\_moyenne\_mobile\_simple\_3)  
  
## Tracer les données bruitées vs les données filtrées  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = modele\_moyenne\_mobile\_simple\_3, y = real\_distance\_cm, col = "Données filtrées")) +  
 labs(title = "Données bruitées vs données filtrées avec un modèle de moyenne mobile\nsimple d'ordre 3",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 scale\_color\_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données filtrées" = "blue")) +  
 theme\_minimal()



Appliquons maintenant un modèle de régression linéaire sur la moyenne mobile simple d’ordre 3.

## Création du modèle de régression linéaire avec la moyenne mobile simple d'ordre 3  
modele\_moyenne\_mobile\_simple\_3\_predict <- lm(real\_distance\_cm ~ modele\_moyenne\_mobile\_simple\_3, data = donnees\_detecteur\_obstacle\_aggregated, na.action = na.omit)  
coef(modele\_moyenne\_mobile\_simple\_3\_predict)

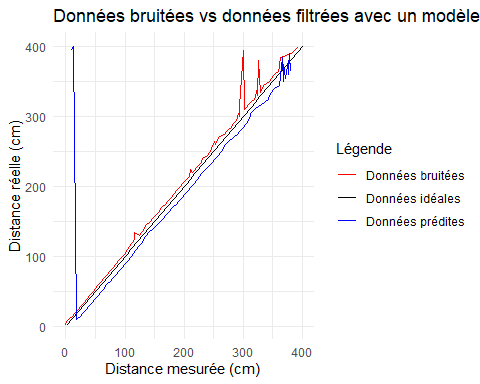
## (Intercept) modele\_moyenne\_mobile\_simple\_3   
## 7.601984 1.031440

## Prédiction des valeur réelles avec la moyenne mobile simple d'ordre 3  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(modele\_moyenne\_mobile\_simple\_3\_predict = ifelse(is.na(modele\_moyenne\_mobile\_simple\_3), NA, predict(modele\_moyenne\_mobile\_simple\_3\_predict)))

L’équation du **modèle de régression linéaire** est alors donnée par :

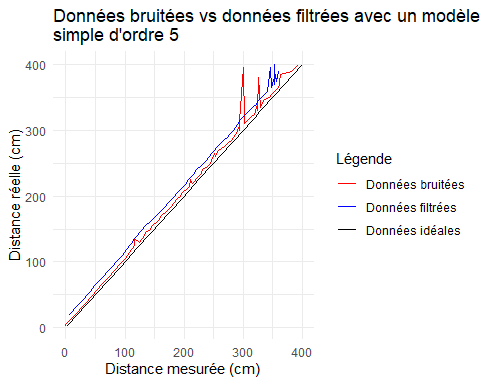
Voici le modèle de prévision pour la régression linéaire avec la moyenne mobile simple d’ordre 3.

## Tracer les données bruitées vs les données prédites par le modèle de régression linéaire  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = modele\_moyenne\_mobile\_simple\_3\_predict, y = real\_distance\_cm, col = "Données prédites")) +  
 labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 scale\_color\_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédites" = "blue")) +  
 theme\_minimal()



# Création d’un modèle de moyenne mobile simple d’ordre 5

modele\_moyenne\_mobile\_simple\_5 <- SMA(donnees\_detecteur\_obstacle\_aggregated$measured\_distance\_cm, n = 5)  
  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(modele\_moyenne\_mobile\_simple\_5 = modele\_moyenne\_mobile\_simple\_5)  
  
## Tracer les données bruitées vs les données filtrées  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = modele\_moyenne\_mobile\_simple\_5, y = real\_distance\_cm, col = "Données filtrées")) +  
 labs(title = "Données bruitées vs données filtrées avec un modèle de moyenne mobile\nsimple d'ordre 5",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 scale\_color\_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données filtrées" = "blue")) +  
 theme\_minimal()



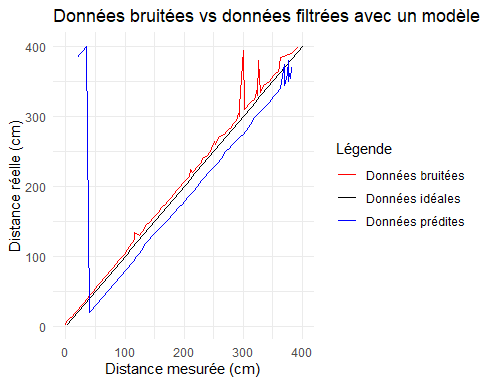
Appliquons maintenant un modèle de régression linéaire sur la moyenne mobile simple d’ordre 5.

## Création du modèle de régression linéaire avec la moyenne mobile simple d'ordre 5  
modele\_moyenne\_mobile\_simple\_5\_predict <- lm(real\_distance\_cm ~ modele\_moyenne\_mobile\_simple\_5, data = donnees\_detecteur\_obstacle\_aggregated, na.action = na.omit)  
  
## Prédiction des valeur réelles avec la moyenne mobile simple d'ordre 5  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(modele\_moyenne\_mobile\_simple\_5\_predict = ifelse(is.na(modele\_moyenne\_mobile\_simple\_5), NA, predict(modele\_moyenne\_mobile\_simple\_5\_predict)))

L’équation du **modèle de régression linéaire** est alors donnée par :

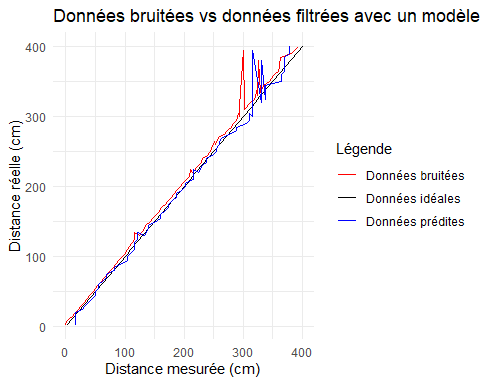
Voici le modèle de prévision pour la régression linéaire avec la moyenne mobile simple d’ordre 5.

## Tracer les données bruitées vs les données prédites par le modèle de régression linéaire  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = modele\_moyenne\_mobile\_simple\_5\_predict, y = real\_distance\_cm, col = "Données prédites")) +  
 labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 scale\_color\_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédites" = "blue")) +  
 theme\_minimal()



# Création d’un modèle Knn

## Pratitionnement des données en train et test  
set.seed(123)  
train\_indices <- createDataPartition(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, p = 0.8, list = FALSE)  
  
data\_knn\_test <- donnees\_detecteur\_obstacle\_aggregated[-train\_indices,] %>%  
 select(real\_distance\_cm, measured\_distance\_cm)  
data\_knn\_train <- donnees\_detecteur\_obstacle\_aggregated[train\_indices,] %>%  
 select(real\_distance\_cm, measured\_distance\_cm)  
  
control <- trainControl(method = "cv", number = 10)  
  
file\_model\_knn = "../data/model\_knn.rds"  
  
if (file.exists(file\_model\_knn)) {  
 model\_knn <- readRDS(file\_model\_knn)  
} else {  
 tune\_grid\_knn <- expand.grid(k = 1:10)  
   
 ## Entrainement du modèle Knn  
 model\_knn <- train(real\_distance\_cm ~ measured\_distance\_cm, data = data\_knn\_train, method = "knn", trControl = control)  
 saveRDS(model\_knn, file\_model\_knn)  
}  
  
## Prédiction avec le modèle Knn  
predictions\_knn <- predict(model\_knn, data\_knn\_test)  
  
donnees\_detecteur\_obstacle\_aggregated <- donnees\_detecteur\_obstacle\_aggregated %>%  
 mutate(modele\_knn = ifelse(is.na(measured\_distance\_cm), NA, predict(model\_knn, .)))  
  
## Tracer les données bruitées vs les données filtrées  
ggplot(donnees\_detecteur\_obstacle\_aggregated, aes(x = measured\_distance\_cm)) +  
 geom\_line(aes(x = real\_distance\_cm, y = real\_distance\_cm, col = "Données idéales")) +  
 geom\_line(aes(x = measured\_distance\_cm, y = real\_distance\_cm, col = "Données bruitées")) +  
 geom\_line(aes(x = modele\_knn, y = real\_distance\_cm, col = "Données prédites")) +  
 labs(title = "Données bruitées vs données filtrées avec un modèle Knn",  
 x = "Distance mesurée (cm)",  
 y = "Distance réelle (cm)",  
 color = "Légende") +  
 scale\_color\_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédites" = "blue")) +  
 theme\_minimal()



# Comparaison des modèles

## Erreur sur le filtre Butterworth  
RMSE\_butter <- rmse(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, donnees\_detecteur\_obstacle\_aggregated$measured\_distance\_cm\_butter)  
MAE\_butter <- mae(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, donnees\_detecteur\_obstacle\_aggregated$measured\_distance\_cm\_butter)  
  
## Erreur sur le filtre Chebyshev de type 1  
RMSE\_chebyshev1 <- rmse(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, donnees\_detecteur\_obstacle\_aggregated$measured\_distance\_cm\_chebyshev1)  
MAE\_chebyshev1 <- mae(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, donnees\_detecteur\_obstacle\_aggregated$measured\_distance\_cm\_chebyshev1)  
  
## Erreur sur le filtre Chebyshev de type 2  
RMSE\_chebyshev2 <- rmse(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, donnees\_detecteur\_obstacle\_aggregated$measured\_distance\_cm\_chebyshev2)  
MAE\_chebyshev2 <- mae(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, donnees\_detecteur\_obstacle\_aggregated$measured\_distance\_cm\_chebyshev2)  
  
## Erreur sur le filtre elliptique  
RMSE\_elliptic <- rmse(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, donnees\_detecteur\_obstacle\_aggregated$measured\_distance\_cm\_elliptic)  
MAE\_elliptic <- mae(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, donnees\_detecteur\_obstacle\_aggregated$measured\_distance\_cm\_elliptic)  
  
## Erreur sur le modèle de régression linéaire  
RMSE\_regression\_lineaire <- rmse(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, donnees\_detecteur\_obstacle\_aggregated$modele\_regression\_lineaire)  
MAE\_regression\_lineaire <- mae(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, donnees\_detecteur\_obstacle\_aggregated$modele\_regression\_lineaire)  
  
## Erreur sur le modèle de moyenne mobile simple d'ordre 3  
valid <- moyenne\_mobile\_simple\_3 <- !is.na(donnees\_detecteur\_obstacle\_aggregated$modele\_moyenne\_mobile\_simple\_3)  
RMSE\_moyenne\_mobile\_simple\_3 <- rmse(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm[valid], donnees\_detecteur\_obstacle\_aggregated$modele\_moyenne\_mobile\_simple\_3[valid])  
MAE\_moyenne\_mobile\_simple\_3 <- mae(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm[valid], donnees\_detecteur\_obstacle\_aggregated$modele\_moyenne\_mobile\_simple\_3[valid])  
  
## Erreur sur le modèle de moyenne mobile simple d'ordre 5  
valid <- moyenne\_mobile\_simple\_5 <- !is.na(donnees\_detecteur\_obstacle\_aggregated$modele\_moyenne\_mobile\_simple\_5)  
RMSE\_moyenne\_mobile\_simple\_5 <- rmse(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm[valid], donnees\_detecteur\_obstacle\_aggregated$modele\_moyenne\_mobile\_simple\_5[valid])  
MAE\_moyenne\_mobile\_simple\_5 <- mae(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm[valid], donnees\_detecteur\_obstacle\_aggregated$modele\_moyenne\_mobile\_simple\_5[valid])  
  
## Error on the Knn model  
RMSE\_knn <- rmse(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, donnees\_detecteur\_obstacle\_aggregated$modele\_knn)  
MAE\_knn <- mae(donnees\_detecteur\_obstacle\_aggregated$real\_distance\_cm, donnees\_detecteur\_obstacle\_aggregated$modele\_knn)  
  
## Création d'un data frame pour les erreurs  
erreurs <- data.frame(modele = c("Butterworth", "Chebyshev de type 1", "Chebyshev de type 2", "Elliptique", "Régression linéaire", "Moyenne mobile simple d'ordre 3", "Moyenne mobile simple d'ordre 5", "Knn"),  
 RMSE = c(RMSE\_butter, RMSE\_chebyshev1, RMSE\_chebyshev2, RMSE\_elliptic, RMSE\_regression\_lineaire, RMSE\_moyenne\_mobile\_simple\_3, RMSE\_moyenne\_mobile\_simple\_5, RMSE\_knn),  
 MAE = c(MAE\_butter, MAE\_chebyshev1, MAE\_chebyshev2, MAE\_elliptic, MAE\_regression\_lineaire, MAE\_moyenne\_mobile\_simple\_3, MAE\_moyenne\_mobile\_simple\_5, MAE\_knn))  
  
print(erreurs)

## modele RMSE MAE  
## 1 Butterworth 14.350809 11.008969  
## 2 Chebyshev de type 1 25.162007 21.065897  
## 3 Chebyshev de type 2 17.186707 16.019051  
## 4 Elliptique 25.148349 21.054370  
## 5 Régression linéaire 9.849407 8.897531  
## 6 Moyenne mobile simple d'ordre 3 15.153396 13.618987  
## 7 Moyenne mobile simple d'ordre 5 19.208796 18.350649  
## 8 Knn 12.351971 6.661376