Biais/Variance

Cette section est basée sur James et al. (2021), chapitre 2.

Que souhaite-t-on faire ?

Supposons que l'on observe une variable réponse, notée Y, pouvant être quantitative, qualitative ou autre et p variables explicatives, notées $X_1, \dots, X_p$, pouvant aussi être quantitatives, qualitatives ou autres. On suppose aussi qu'il existe une certaine relation entre Y et $X = (X_1, \dots, X_p)$. Cette relation peut s'écrire de façon gérérale comme

$$Y = f(X) + \epsilon. \tag{1}$$

Ici, f est une fonction de $X_1, \dots, X_p$ que l'on souhaite estimer à l'aide des données et ϵ est un terme d'erreur. Dans le cadre de ce cours, on supposera que ϵ est indépendent de X et sa moyenne est nulle. Le modèle Équation 1 est un modèle général dans le sens où tout ce que l'on va faire peut s'écrire sous cette forme, bien que l'on ne soit pas toujours capable de donner une équation pour f. La fonction f représente l'information systématique que X donne à propos de Y.

```
# Load required package
library(ggplot2)

# Set seed for reproducibility
set.seed(42)

# 1. Generate data
n <- 100
x <- sort(runif(n, 0, 2 * pi))
f_x <- sin(x)
epsilon <- rnorm(n, mean = 0, sd = 0.3)
y <- f_x + epsilon

# 2. Create data frame
```

```r
data <- data.frame(x = x, y = y, f_x = f_x)

# 3. Plot
ggplot(data, aes(x, y)) +
  # Vertical lines: error = Y - f(X)
  geom_segment(
    aes(x = x, xend = x, y = f_x, yend = y),
    color = "gray60", linetype = "dashed"
  ) +
  # Observed points
  geom_point(color = "black", alpha = 0.7, size = 2) +
  # True function
  geom_line(aes(x = x, y = f_x), color = "blue", size = 1.2) +
  labs(
    x = "X", y = "Y"
  ) +
  theme_minimal()
```

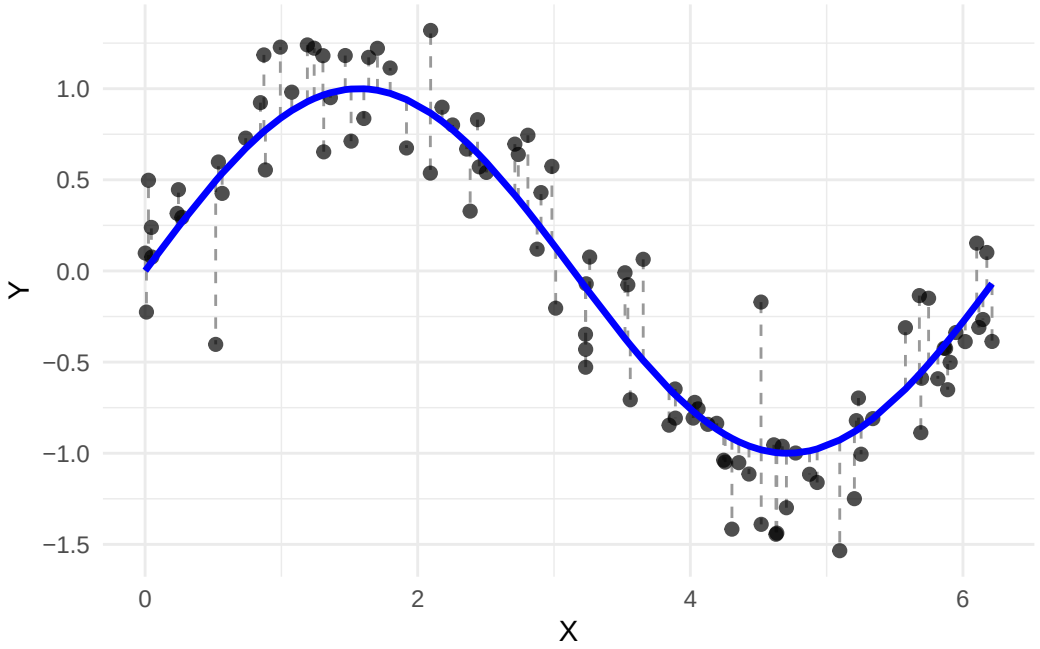

FIGURE 1 – Différent éléments du modèle. Les points représentent les données observées (X, Y). La courbe bleue représente la fonction f et les lignes pointillées représentent l'erreur associée à chaque observation.

Dans la suite du cours, on verra quelques méthodes permettant d'estimer la fonction f. Avant de voir comment estimer f, on va supposer que l'on a déjà un estimateur, noté $\hat{f}$, estimé à partir de n observations de (X, Y) et on va s'intéresser à la qualité de cet estimateur.

Exemple : la régression linéaire simple

Dans le cadre de la régression linéaire simple, on fait l'hypothèse que la fonction f est de la forme : $f(x) = ax+b$. Dans ce cas, l'estimation de la fonction f se résume à l'estimation des coefficients a et b.

Remarque : Compromis exactitude / interprétabilité

Dépendent de l'objectif de l'étude, on peut devoir faire un choix entre l'exactitude de nos prédictions et l'interprétabilité de notre modèle. En effet, si on restreint notre modèle à être linéaire, nos paramètres seront interprétables et même visualisable, mais ce sera peut-être au détriment du pouvoir prédictif du modèle, e.g. si la vraie relation entre X et Y n'est pas linéaire. Inversement, un modèle pouvant estimer des relations plus compliqué (et donc plus flexible) aura plus de paramètres et donc sera plus difficile à interpréter.

Mesurer la qualité de l'ajustement

Pour évaluer la qualité de notre estimateur $\hat{f}$, on a besoin d'une mesure nous indiquant à si nos prédictions, $\hat{y}$, sont proches des données observées. Dit autrement, on cherche à quantifier si la réponse prédite pour chaque observation est proche de la vraie réponse pour cette observation.

Définition : Erreur quadratique moyenne

Dans le cas où Y est une variable quantitative, une mesure de la qualité de $\hat{f}$ est l'**erreur quadratique moyenne** (*mean square error*, MSE). Celle-ci est défini comme

$$MSE(Y, \widehat{Y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2,$$

où $\hat{y}_i = \hat{f}(x_i)$ est la prédiction que $\hat{f}$ donne pour l'observation i.

L'idée est que la MSE sera petite si les réponses prédites sont proches des vraies réponses et grande si elles ne le sont pas. On peut aussi voir la MSE comme une mesure de la distance moyenne entre les vraies valeurs et les valeurs prédites. On cherche donc à avoir une distance moyenne petite.

```r
# Load packages
library(ggplot2)
library(dplyr)
library(tidyr)

set.seed(42)

# 1. Simulate a single dataset
n <- 100
sigma <- 0.3
x <- sort(runif(n, 0, 1))
y <- 4 * x * (1 - x) * log(x) + 2 + rnorm(n, 0, sigma)
df <- data.frame(x = x, y = y)

# 2. Define grid and spans to compare
x_grid <- seq(0, 1, length.out = 300)
spans_to_plot <- c(0.15, 0.3, 0.5, 0.75, 1.0)

# 3. Compute loess fits for each span
fits <- lapply(spans_to_plot, function(s) {
  loess_model <- loess(y ~ x, data = df, span = s)
  y_hat <- predict(loess_model, newdata = data.frame(x = x_grid))
  data.frame(x = x_grid, y_hat = y_hat, span = paste0("  = ", s))
})

fit_df <- bind_rows(fits)

# 4. Plot
ggplot() +
  geom_point(data = df, aes(x, y), color = "black", alpha = 0.5, size = 2) +
  geom_line(data = fit_df, aes(x, y_hat, color = span), size = 1.1) +
  scale_color_viridis_d(option = "C") +
  labs(
    x = "Y", y = "Y",
    color = "Paramètre"
  ) +
  theme_minimal(base_size = 14)
```

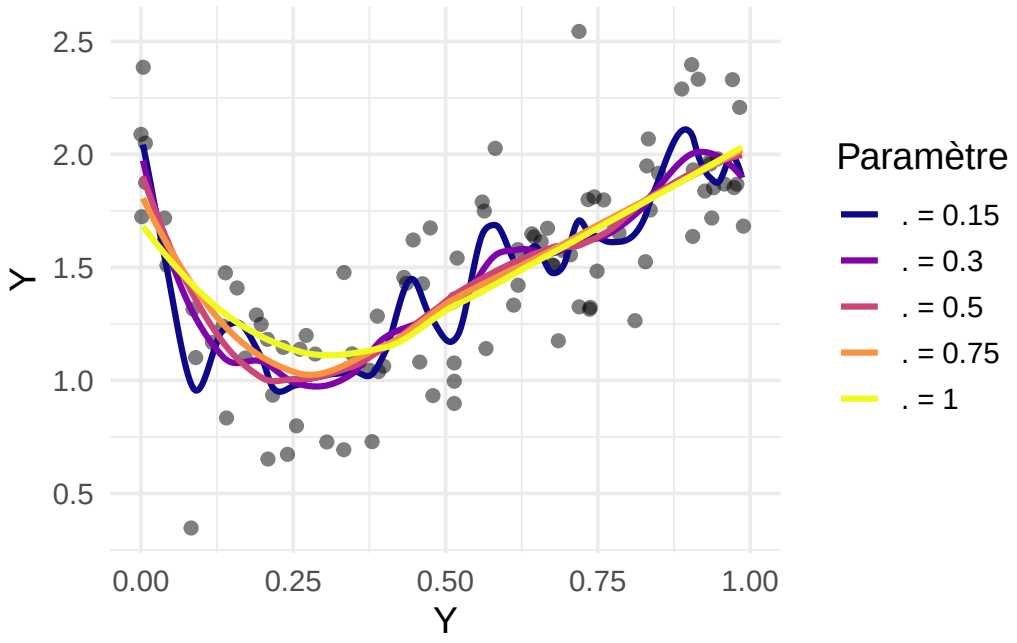

FIGURE 2 – Différents estimateurs de la fonction f.

```r
# Load packages
library(ggplot2)
library(dplyr)
library(tidyr)

set.seed(123)

# Parameters
n <- 100               # number of observations per dataset
n_sim <- 100           # number of simulated datasets
spans <- seq(0.1, 1, length.out = 15)  # LOESS smoothing parameters
sigma <- 0.1           # noise standard deviation
x_grid <- seq(0.1, 1, length.out = 200)
f_true <- 4 * x_grid * (1 - x_grid) * log(x_grid) + 2

# Storage for predictions
results <- list()

for (s in spans) {
  pred_matrix <- matrix(NA, nrow = length(x_grid), ncol = n_sim)
```

```r
  for (sim in 1:n_sim) {
    x <- sort(runif(n, 0.01, 1.1))
    y <- 4 * x * (1 - x) * log(x) + 2 + rnorm(n, 0, sigma)
    df <- data.frame(x = x, y = y)

    # Fit loess model with span = s
    model <- loess(y ~ x, data = df, span = s, degree = 2)
    pred <- predict(model, newdata = data.frame(x = x_grid), )

    pred_matrix[, sim] <- pred
  }

  # For each point in x_grid, compute bias², variance, MSE
  mean_pred <- rowMeans(pred_matrix, na.rm = TRUE)
  bias2 <- (mean_pred - f_true)^2
  var_pred <- apply(pred_matrix, 1, var, na.rm = TRUE)
  mse <- bias2 + var_pred

  results[[as.character(s)]] <- data.frame(
    span = s,
    Biais2 = mean(bias2),
    Variance = mean(var_pred),
    MSE = mean(mse)
  )
}

# Combine and reshape results
results_df <- bind_rows(results)
results_long <- pivot_longer(
  results_df,
  cols = c("Biais2", "Variance", "MSE"),
  names_to = "component", values_to = "value"
)

# Plot
ggplot(results_long, aes(x = span, y = value, color = component)) +
  geom_line(size = 1.2) +
  geom_point(size = 2) +
  scale_color_manual(
    values = c("Biais2" = "#0D0887FF", "Variance" = "#9C179EFF", "MSE" = "#ED7953FF")
  ) +
  labs(
```

```r
    x = "Paramètre de lissage",
    y = "",
    color = ""
  ) +
  theme_minimal(base_size = 14)
```

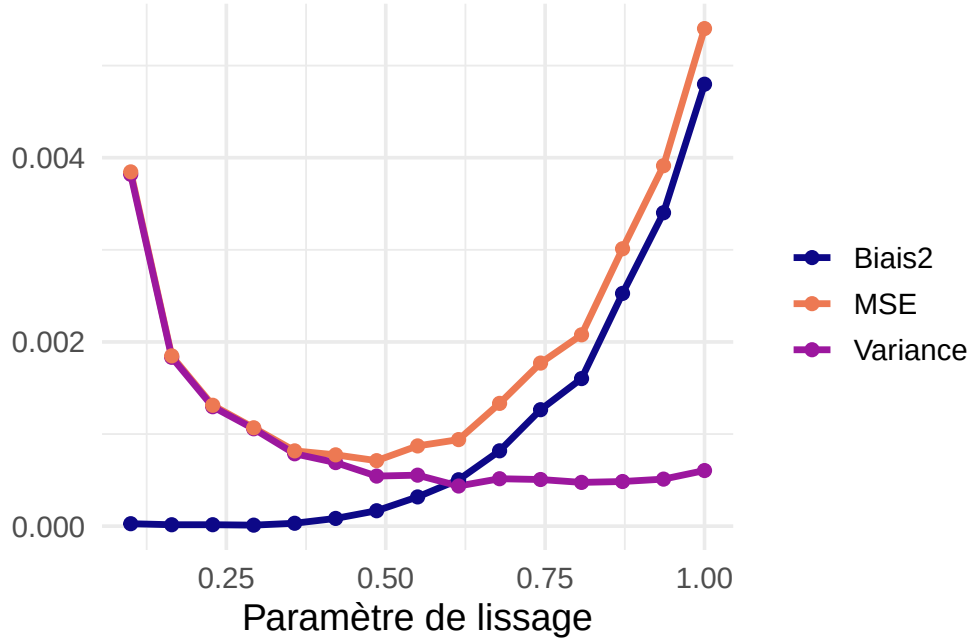

FIGURE 3 – Compromis biais/variance

Compromis biais/variance

Compromis biais/variance

$$\mathbb{E}\left[(Y-\widehat{Y})^2\right] = \mathrm{Var}(\widehat{Y}) + \left(Y - \mathbb{E}(\widehat{Y})\right)^2 + \sigma^2.$$

Preuve

$$\begin{aligned}
\mathbb{E}\left[\left(Y-\widehat{Y}\right)^2\right] &= \mathbb{E}\left[\left(Y-\hat{f}(X)\right)^2\right] \\
&= \mathbb{E}\left[\left(f(X)+\varepsilon-\hat{f}(X)\right)^2\right] \\
&= \mathbb{E}\left[\left(\underbrace{f(X)-\hat{f}(X)}_{\text{erreur estimation}}+\varepsilon\right)^2\right] \\
&= \mathbb{E}\left[\left(f(X)-\hat{f}(X)\right)^2\right] \\
&\quad + 2\mathbb{E}\left[\left(f(X)-\hat{f}(X)\right)\varepsilon\right] \\
&\quad + \mathbb{E}[\varepsilon^2].
\end{aligned}$$

On peut mesurer deux fonctions qui vont nous aider :

— Fonction de perte (L) : c'est la mesure de l'écart par rapport à ce qu'on souhaite mesurer, par example :

$$L(Y, f(X)) = (Y - f(X))^2.$$

— Fonction de risque : c'est la quantité que l'on cherche à minimiser. Il s'agit de l'espérance de la fonction de perte.

Comment trouver f :

Objectif : Trouver une fonction $\hat{f}$ qui minimise le risque.

Comment : Supposer une certaine forme pour $f(X)$ et minimiser la fonction de perte de façon analytique ou numérique.

— Paramétrique : On donne une forme explication à $f(X)$ qui dépend de paramètres. On cherche une méthode d'estimation des paramètres.
— Non-paramétrique : aucune forme forme particulière de f, on estime une courbe ou fonction.

Un mot sur le cas de variables qualitatives

On cherche à prédire G (un groupe ou facteur) à partir de X. Supposons que nous avons $\widehat{G}$ qui prédise le classe des observations sachant X, alors on peut définir la fonction de perte (0-1) comme le nombre d'erreur que l'on a effectué :

$$L(G, \widehat{G}) = \mathbb{1}_{G \neq \widehat{G}}.$$

Avec cette fonction de perte 0-1, $\widehat{G}(x)$ est la classe g qui maximise $\mathbb{P}(g|X = x)$.

On peut décomposer l'erreur quadratique moyenne (EQM) :

$$\mathbb{E}\left((Y - \hat{f}(x_0))^2\right) = \text{Biais}(\hat{f}(x_0))^2 + \text{Var}(\hat{f}(x_0) + \sigma_\epsilon^2.$$

Démontrer la décomposition de l'EQM en un compromis biais variance.

James, Gareth, Daniela Witten, Trevor Hastie, et Robert Tibshirani. 2021. *An Introduction to Statistical Learning : With Applications in R*. Springer Texts in Statistics. New York, NY : Springer US. https://doi.org/10.1007/978-1-0716-1418-1.