SAE - 3.03 V2

Dorian RELAVE

2023-10-24

Contents

1 1. Présentation des données

1.1 Source et description

Les données utilisées pour cette analyse proviennent du site SteamDB et ont été rendues disponibles sur la plateforme Kaggle.

Le jeu de données représente le nombre quotidien de joueurs connectés simultanément sur la plateforme Steam ("in-game players"). Les observations sont régulières, collectées chaque jour sans interruption depuis le 20 février 2019.

1.2 Période d'étude

Pour éviter les données manquantes et les effets de la pandémie de COVID-19 sur les habitudes de jeu, nous travaillerons sur l'année 2019 uniquement, plus précisément du **25 février 2019** (un lundi) au **22 décembre 2019** (un dimanche). Cette période représente 43 semaines complètes, soit 301 jours.

2 2. Chargement et préparation des données

```
# Chargement des données depuis le fichier CSV
A = read.csv(file = "./data/Steam Players.csv", sep = ",", header = TRUE)

# Définition de la période d'étude (7 jours = 1 semaine)
PERIODE = 7

# Extraction des données pertinentes pour notre période d'étude
# Indices 5523 à 5823 du fichier original
DATA = A$In.Game[5523:5823]

# Vérification des dates de début et fin
cat("Date de début:", A$DateTime[5523], "\n")
```

```
## Date de début: 2019-02-25 00:00:00
```

```
cat("Date de fin:", A$DateTime[5823], "\n")
```

```
## Date de fin: 2019-12-22 00:00:00
```

```
# Calcul du nombre de périodes (semaines) dans notre jeu de données
nbr_de_periode = length(DATA)/PERIODE
cat("Nombre de semaines:", nbr_de_periode, "\n")
```

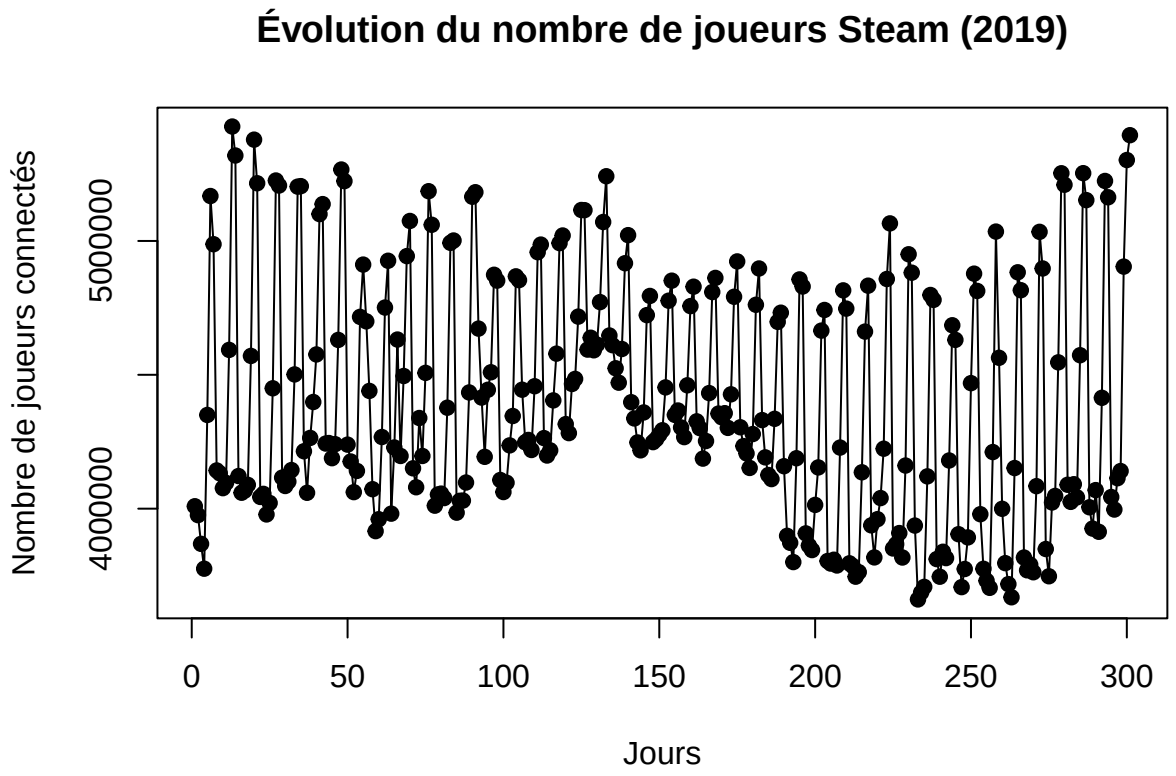
```
## Nombre de semaines: 43
```

3 3. Analyse exploratoire des données

3.1 3.1 Visualisation chronologique complète

Commençons par visualiser l'ensemble de nos données dans l'ordre chronologique pour observer les tendances générales et les patterns saisonniers.

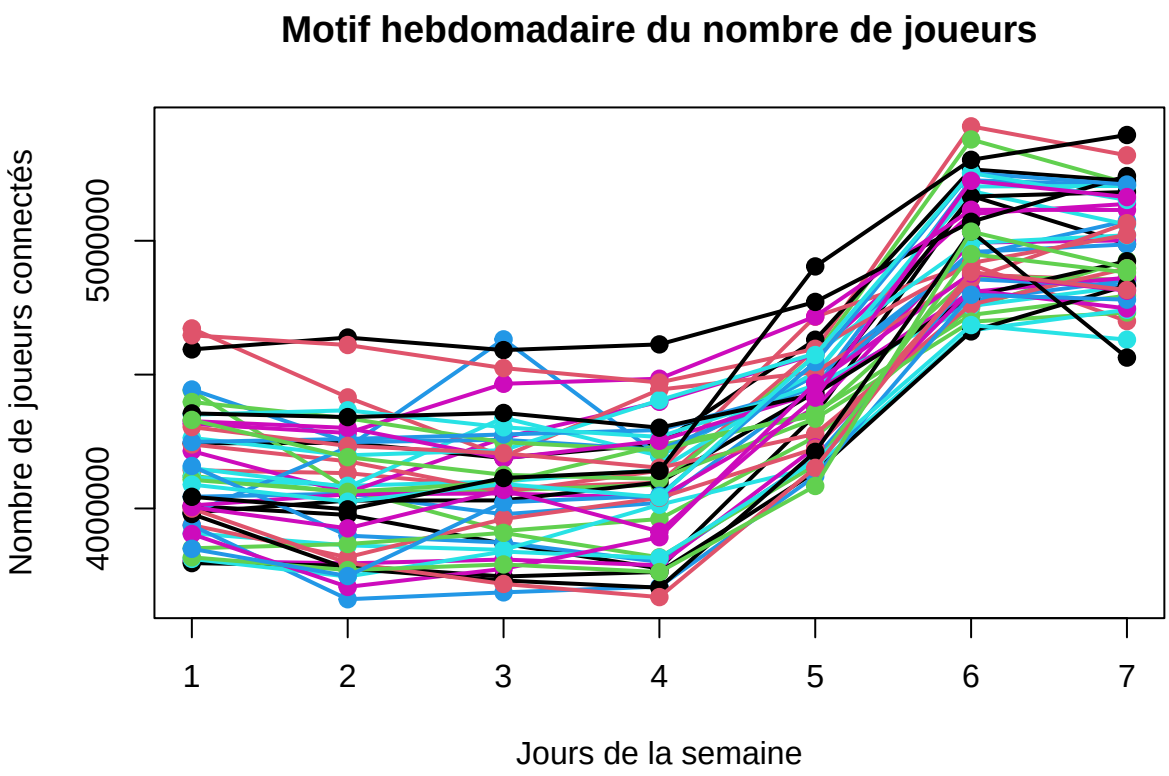
```r
# Représentation chronologique des données
plot(DATA, type = "o", pch = 19, xlab = "Jours", ylab = "Nombre de joueurs connectés",
     main = "Évolution du nombre de joueurs Steam (2019)", lwd = 1)
```



Cette représentation montre une série temporelle avec des motifs périodiques visibles. Pour mieux analyser ces patterns, nous allons également examiner la composante saisonnière (hebdomadaire).

3.2 3.2 Visualisation du motif hebdomadaire

```r
# Réorganisation des données par jour de la semaine
DATA_groupe <- matrix(DATA, nrow = PERIODE, byrow = FALSE)

# Visualisation des données par jour de la semaine
# Chaque ligne colorée représente une semaine différente
matplot(DATA_groupe, type = "o", lty = 1, lwd = 2, pch = 19, xlab = "Jours de la semaine",
        ylab = "Nombre de joueurs connectés", main = "Motif hebdomadaire du nombre de joueurs")
```



Observation : On constate clairement que le nombre de joueurs connectés augmente considérablement en fin de semaine (jours 5, 6 et 7, qui correspondent à vendredi, samedi et dimanche). Ce phénomène est cohérent avec les habitudes de loisirs typiques.

4 4. Décomposition de la série temporelle

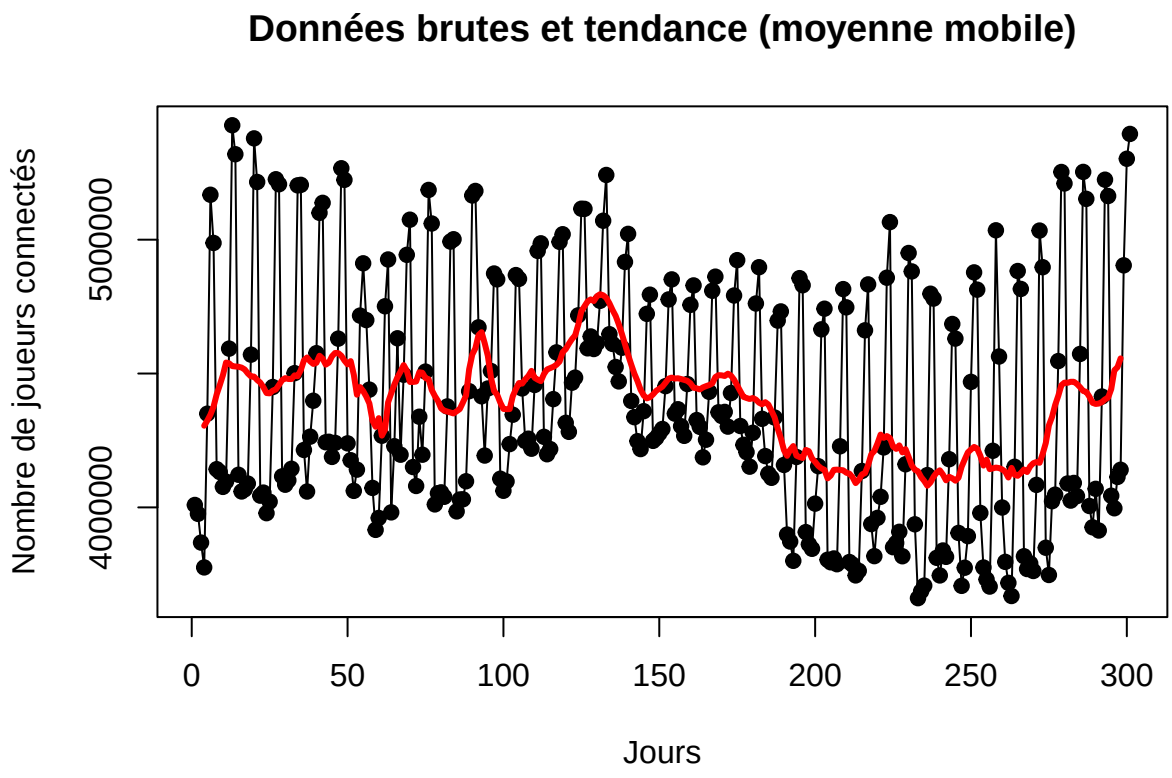
Pour analyser notre série temporelle, nous allons la décomposer en trois composantes principales : - **Tendance** : l'évolution à long terme de la série - **Saisonnalité** : les variations régulières (hebdomadaires dans notre cas) - **Résidus** : les variations irrégulières non expliquées par les deux composantes précédentes

4.1 4.1 Calcul de la tendance par moyenne mobile

La moyenne mobile permet d'estimer la tendance en lissant les variations saisonnières et aléatoires.

```
# Fonction pour calculer la moyenne mobile selon que la période est paire ou impaire
moyenne_mobile <- function(DATA, WINDOW) {
  # Si la période est paire
  if (WINDOW %% 2 == 0) {
    k <- WINDOW / 2
    n <- length(DATA)
    result <- numeric(n - WINDOW)

    for (i in 1:(n - WINDOW)) {
      # Moyenne mobile centrée pour période paire
      result[i] <- (0.5 * DATA[i] + sum(DATA[(i + 1):((i + 2*k)-1)]) + 0.5 * DATA[i + 2*k])/WINDOW
    }
```

```r
    # Ajout des valeurs NA au début et à la fin
    result <- c(rep(NA, k), result, rep(NA, k))
  }
  # Si la période est impaire
  else if (WINDOW %% 2 == 1){
    k <- (WINDOW - 1) / 2
    n <- length(DATA)
    result <- numeric(n - (WINDOW-1))

    for (i in 1:(n - (WINDOW-1))) {
      # Moyenne mobile centrée pour période impaire
      result[i] <- (sum(DATA[(i):(i+WINDOW-1)])) / WINDOW
    }

    # Ajout des valeurs NA au début et à la fin
    result <- c(rep(NA, k), result, rep(NA, k))
  }

  return(result)
}

# Calcul de la moyenne mobile sur une période de 7 jours
DATA_sma <- moyenne_mobile(DATA, PERIODE)

# Visualisation des données brutes avec la tendance (moyenne mobile)
plot(DATA, type = "o", pch = 19, xlab = "Jours", ylab = "Nombre de joueurs connectés",
     main = "Données brutes et tendance (moyenne mobile)", lwd = 1)
lines(DATA_sma, col = "red", lwd = 3)
```



Interprétation : La ligne rouge représente la tendance générale du nombre de joueurs connectés, obtenue par moyenne mobile sur 7 jours. Elle lisse les variations hebdomadaires et permet de mieux visualiser

l'évolution à long terme.

4.2 4.2 Estimation de la composante saisonnière

Nous allons maintenant extraire la composante saisonnière en comparant les valeurs observées à la tendance. Cette analyse sera faite selon deux modèles : - **Modèle additif** : les composantes s'additionnent (Data = Tendance + Saisonnalité + Résidus) - **Modèle multiplicatif** : les composantes se multiplient (Data = Tendance × Saisonnalité × Résidus)

```
# Fonction pour calculer les écarts à la tendance (modèle additif)
delta_additif <- function(DATA, sma, WINDOW){
  # Initialisation selon la parité de la période
  if (WINDOW %% 2 == 0) {
    k <- WINDOW / 2
  } else {
    k <- (WINDOW - 1) / 2
  }

  # Calcul des écarts (valeur observée - tendance)
  delta <- c()
  for (i in 1:length(DATA)-2*k){
    delta[i+k] <- DATA[i+k] - sma[i+k]
  }

  # Formatage du résultat en matrice par semaine
  delta_result <- c(delta, rep(NA, k))
  return(matrix(delta_result, ncol=WINDOW, byrow = TRUE))
}

# Fonction pour calculer les ratios à la tendance (modèle multiplicatif)
delta_multiplicatif <- function(DATA, sma, WINDOW){
  # Initialisation selon la parité de la période
  if (WINDOW %% 2 == 0) {
    k <- WINDOW / 2
  } else {
    k <- (WINDOW - 1) / 2
  }

  # Calcul des ratios (valeur observée / tendance)
  delta <- c()
  for (i in 1:length(DATA)-2*k){
    delta[i+k] <- DATA[i+k] / sma[i+k]
  }

  # Formatage du résultat en matrice par semaine
  delta_result <- c(delta, rep(NA, k))
  return(matrix(delta_result, ncol=WINDOW, byrow = TRUE))
}

# Fonction pour calculer les coefficients saisonniers (médiane de chaque jour de la semaine)
coef_saison <- function(delta_result){
  med_ <- c()
  for (i in 1:ncol(delta_result)){
    col = as.vector(na.omit(delta_result[,i]))
    med_[i] <- median(col)
  }
  return(med_)
}

# Fonction pour calculer la série corrigée des variations saisonnières (CVS)
CVS <- function(Y, S, periode, type){
  if (type == "add") { # Additif
    CVS = Y - rep(S, periode)
  } else if (type == "mult"){ # Multiplicatif
```

```
    CVS = Y / (1+S)
  }
  return(CVS)
}

# --- Modèle multiplicatif ---
DATA_delta_mult = delta_multiplicatif(DATA, DATA_sma, PERIODE)
DATA_S_prim_mult <- coef_saison(DATA_delta_mult)
# Centrage des coefficients saisonniers
DATA_S_chap_mult = DATA_S_prim_mult - mean(DATA_S_prim_mult)

# Calcul de la série corrigée des variations saisonnières (modèle multiplicatif)
DATA_CVS_mult = CVS(DATA, DATA_S_chap_mult, nbr_de_periode, "mult")

# --- Modèle additif ---
DATA_delta_add <- delta_additif(DATA, DATA_sma, PERIODE)
DATA_S_prim_add <- coef_saison(DATA_delta_add)
# Centrage des coefficients saisonniers
DATA_S_chap_add = DATA_S_prim_add - mean(DATA_S_prim_add)

# Calcul de la série corrigée des variations saisonnières (modèle additif)
DATA_CVS_add = CVS(DATA, DATA_S_chap_add, nbr_de_periode, "add")

# Affichage des coefficients saisonniers pour les deux modèles
cat("Coefficients saisonniers (modèle multiplicatif):\n")
```

```
## Coefficients saisonniers (modèle multiplicatif):
```

```
print(round(DATA_S_chap_mult, 4))
```

```
## [1] -0.0616 -0.0765 -0.0724 -0.0705  0.0050  0.1363  0.1397
```

```
cat("\nCoefficients saisonniers (modèle additif):\n")
```

```
##
## Coefficients saisonniers (modèle additif):
```

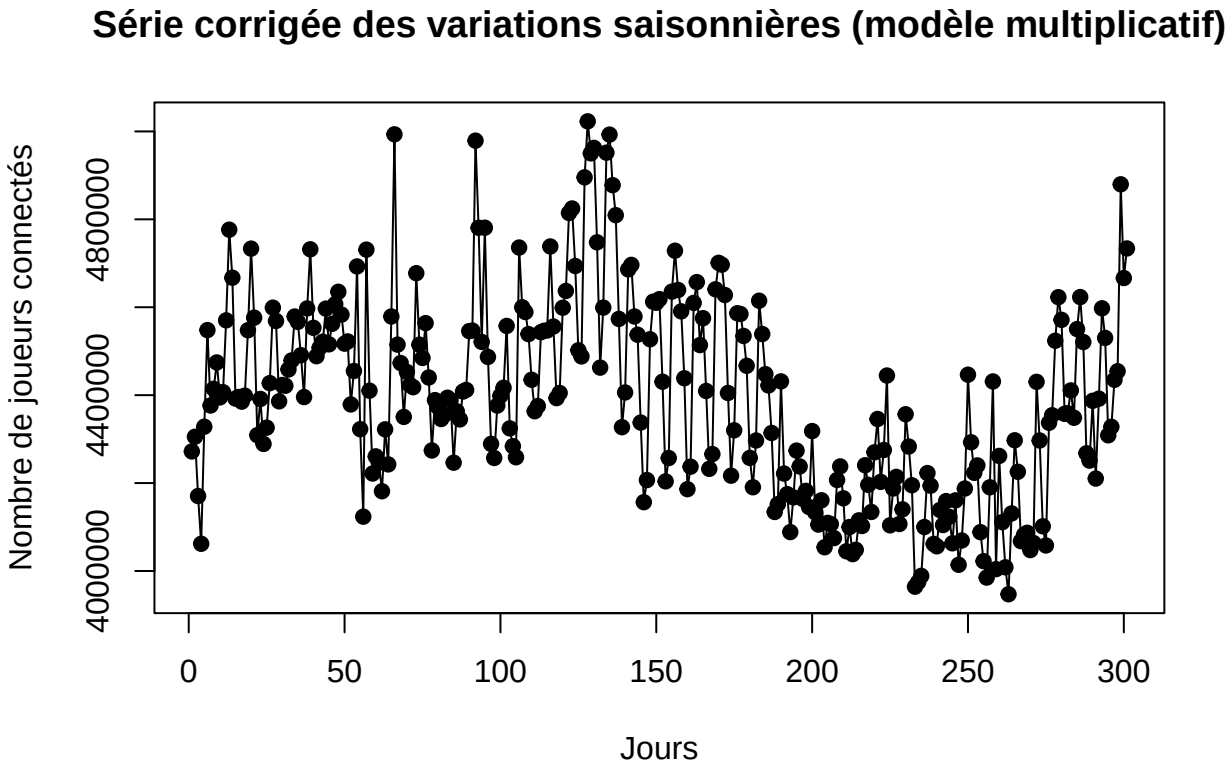
```
print(round(DATA_S_chap_add, 0))
```

```
## [1] -274016 -325373 -312357 -305385   20584  591391  605156
```

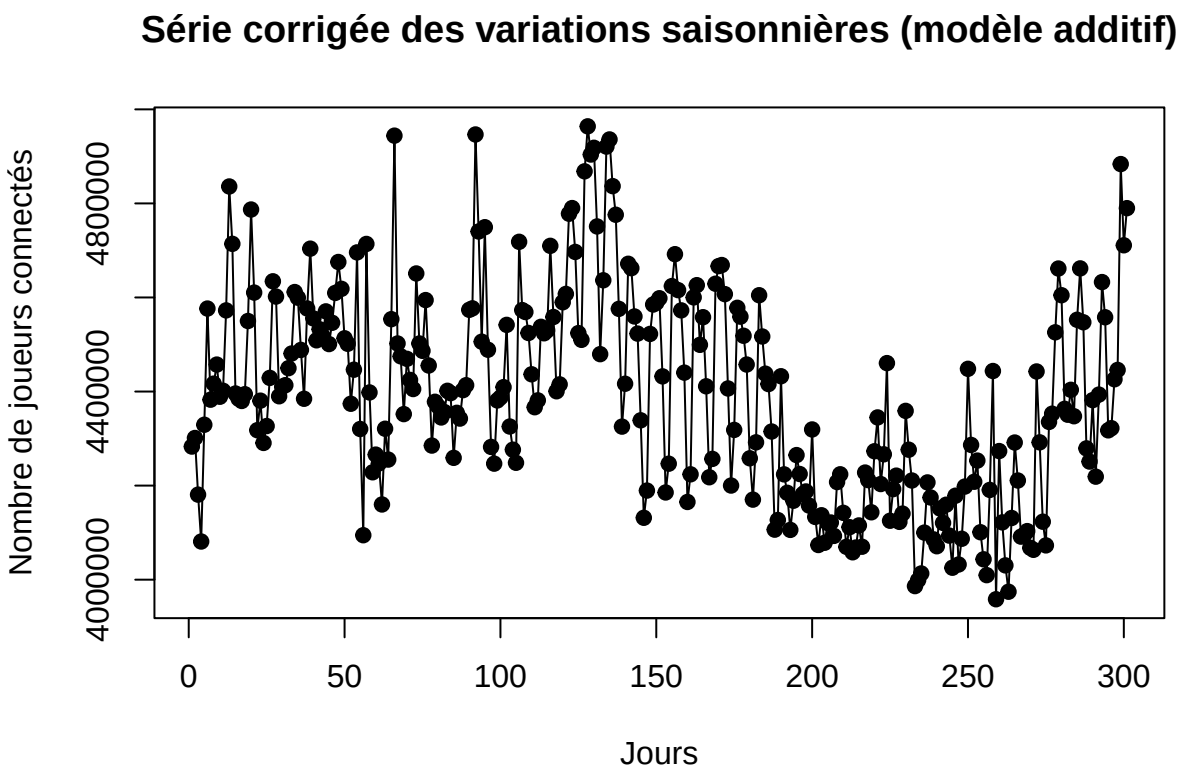
Interprétation des coefficients saisonniers : - Les valeurs positives indiquent les jours où le nombre de joueurs est supérieur à la moyenne hebdomadaire - Les valeurs négatives indiquent les jours où le nombre de joueurs est inférieur à la moyenne hebdomadaire - On observe clairement l'augmentation en fin de semaine (jours 5-7) et la diminution en milieu de semaine

4.3 4.3 Visualisation des séries corrigées des variations saisonnières (CVS)

```
# Visualisation de la série corrigée des variations saisonnières (modèle multiplicatif)
plot(DATA_CVS_mult, type = "o", pch = 19,
     main = "Série corrigée des variations saisonnières (modèle multiplicatif)",
     xlab = "Jours", ylab = "Nombre de joueurs connectés")
```




```r
# Visualisation de la série corrigée des variations saisonnières (modèle additif)
plot(DATA_CVS_add, type = "o", pch = 19,
     main = "Série corrigée des variations saisonnières (modèle additif)",
     xlab = "Jours", ylab = "Nombre de joueurs connectés")
```

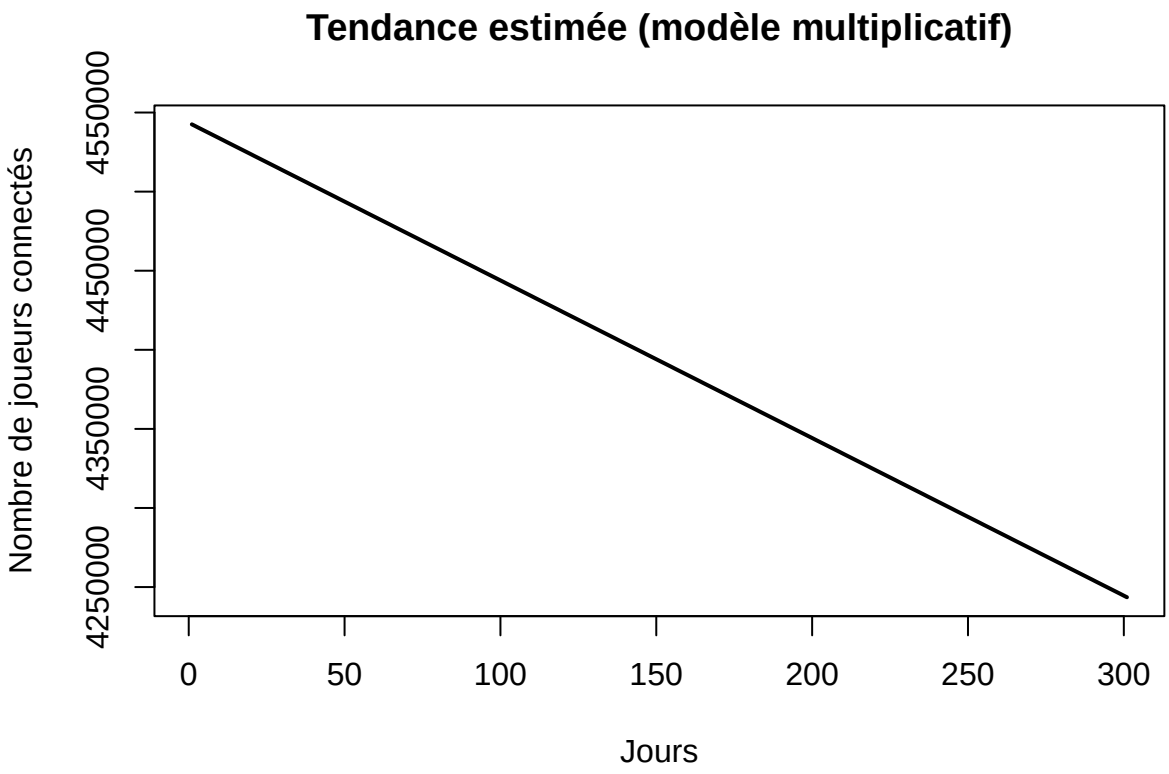


Observation : Ces graphiques montrent les données après suppression de l'effet saisonnier. Ils permettent de mieux visualiser la tendance et les variations non saisonnières.

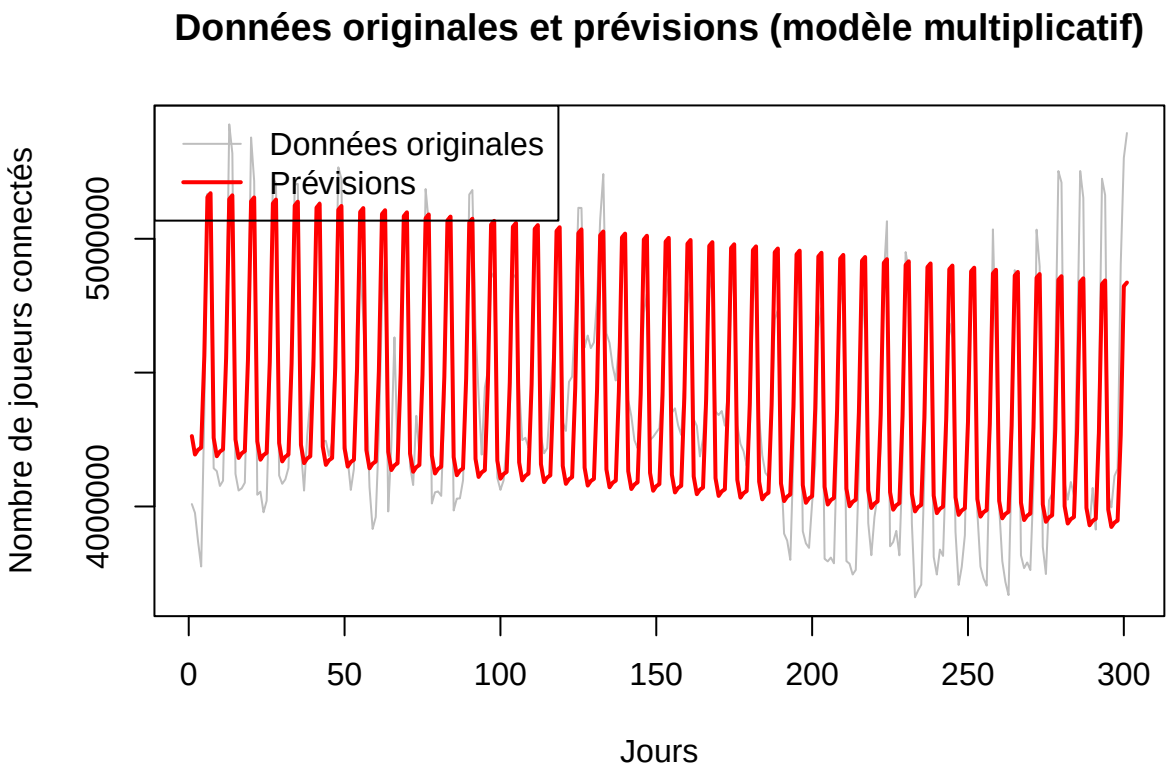
5 5. Modélisation de la tendance

Maintenant que nous avons isolé la composante saisonnière, nous pouvons modéliser la tendance à l'aide d'une régression linéaire.

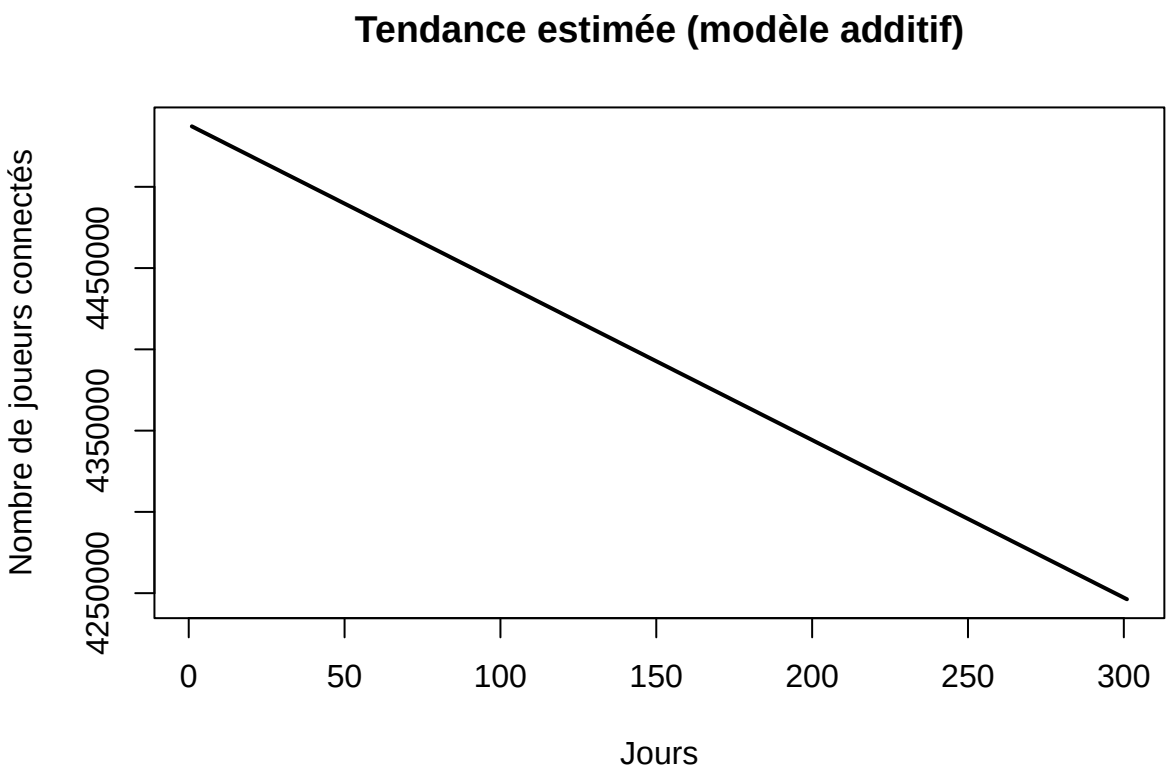
```
# Création d'une variable temporelle
t = seq(length(DATA))

# --- Modèle multiplicatif ---
# Régression linéaire sur la série CVS
modele_mult = lm(DATA_CVS_mult ~ t)
b0_mult = modele_mult$coefficients[1]  # Ordonnée à l'origine
b1_mult = modele_mult$coefficients[2]  # Pente

# Visualisation de la tendance estimée
plot(modele_mult$fitted.values, type = 'l',
     main = "Tendance estimée (modèle multiplicatif)",
     lwd = 2, xlab = "Jours", ylab = "Nombre de joueurs connectés")
```



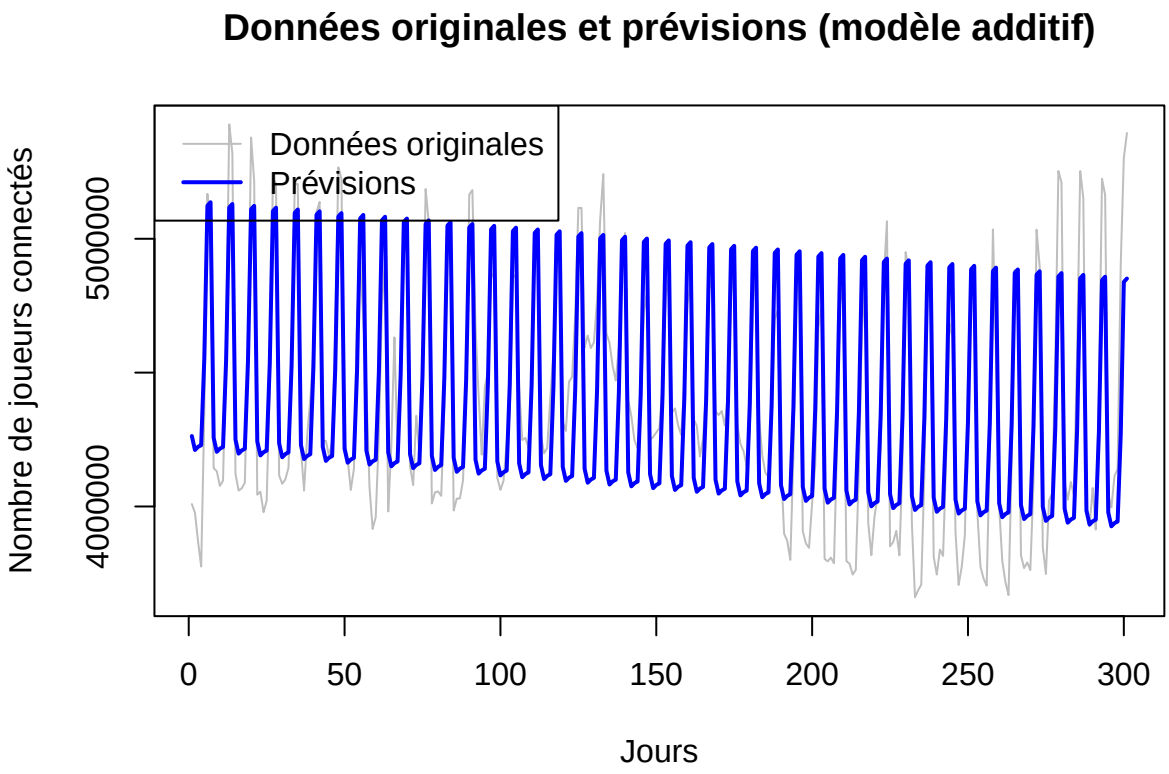

```r
# Calcul de la tendance et des prévisions complètes
f_chap_mult = b0_mult + b1_mult*t  # Tendance
y_chap_mult = f_chap_mult*(1 + DATA_S_chap_mult)  # Prévisions (tendance × saisonnalité)

# Visualisation des prévisions complètes
plot(DATA, type = "l", col = "gray",
     main = "Données originales et prévisions (modèle multiplicatif)",
     xlab = "Jours", ylab = "Nombre de joueurs connectés")
lines(y_chap_mult, col = "red", lwd = 2)
legend("topleft", legend = c("Données originales", "Prévisions"),
       col = c("gray", "red"), lwd = c(1, 2))
```




```r
# --- Modèle additif ---
# Régression linéaire sur la série CVS
modele_add = lm(DATA_CVS_add ~ t)
b0_add = modele_add$coefficients[1]  # Ordonnée à l'origine
b1_add = modele_add$coefficients[2]  # Pente

# Visualisation de la tendance estimée
plot(modele_add$fitted.values, type = 'l',
     main = "Tendance estimée (modèle additif)",
     lwd = 2, xlab = "Jours", ylab = "Nombre de joueurs connectés")
```



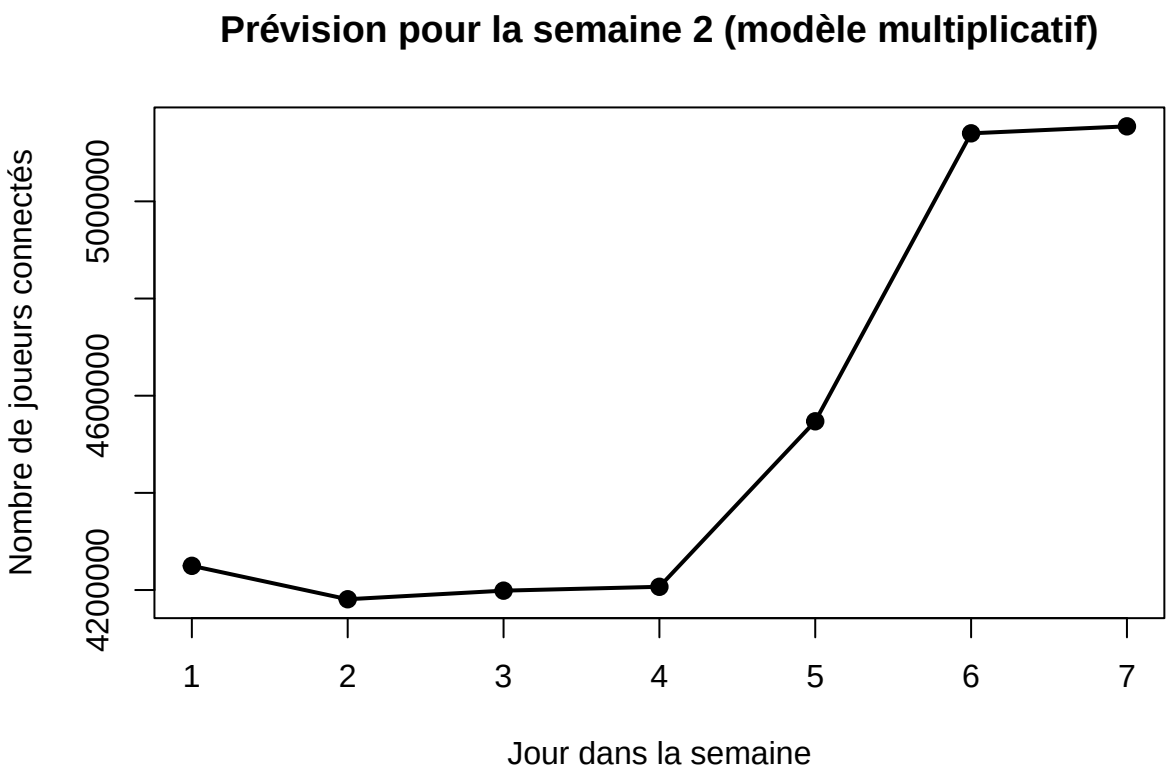

```r
# Calcul de la tendance et des prévisions complètes
f_chap_add = b0_add + b1_add*t  # Tendance
y_chap_add = f_chap_add + rep(DATA_S_chap_add, nbr_de_periode)  # Prévisions (tendance +
↪  saisonnalité)

# Visualisation des prévisions complètes
plot(DATA, type = "l", col = "gray",
     main = "Données originales et prévisions (modèle additif)",
     xlab = "Jours", ylab = "Nombre de joueurs connectés")
lines(y_chap_add, col = "blue", lwd = 2)
legend("topleft", legend = c("Données originales", "Prévisions"),
       col = c("gray", "blue"), lwd = c(1, 2))
```



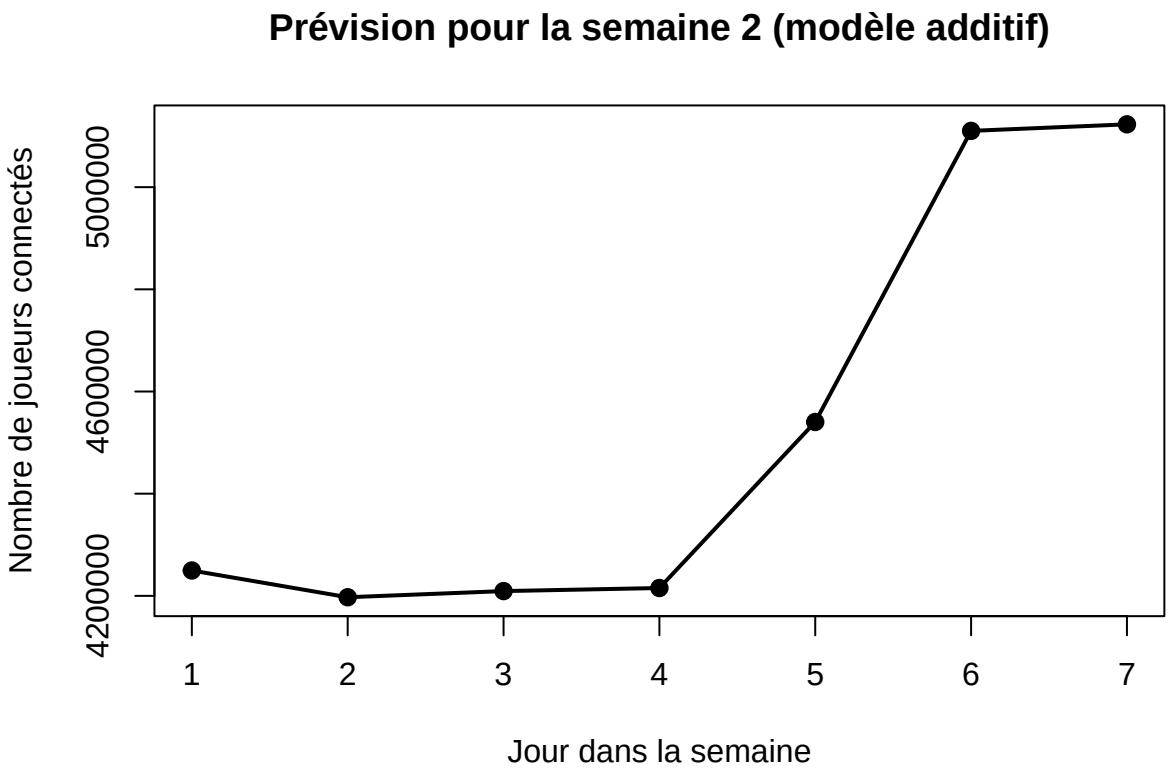
Interprétation : - La tendance linéaire montre une légère diminution du nombre de joueurs sur la période étudiée - Les prévisions intègrent à la fois cette tendance et le motif saisonnier hebdomadaire - Visuellement, le modèle additif semble mieux correspondre à nos données

5.1 5.1 Prévision pour une semaine spécifique

```
# Semaine pour laquelle on souhaite faire une prévision (exemple: 2ème semaine)
x = 2

# --- Modèle multiplicatif ---
# Calcul de la tendance pour la semaine x
f_chap_en_t_mult = b0_mult + b1_mult*(x*PERIODE + (1:PERIODE))
# Calcul des prévisions complètes pour la semaine x
y_chap_en_t_mult = f_chap_en_t_mult*(1 + DATA_S_chap_mult)

# Visualisation des prévisions pour la semaine x
plot(y_chap_en_t_mult, type = "o", pch = 19, lwd = 2,
     main = "Prévision pour la semaine 2 (modèle multiplicatif)",
     xlab = "Jour dans la semaine", ylab = "Nombre de joueurs connectés")
```




```r
# --- Modèle additif ---
# Calcul de la tendance pour la semaine x
f_chap_en_t_add = b0_add + b1_add*(x*PERIODE + (1:PERIODE))
# Calcul des prévisions complètes pour la semaine x
y_chap_en_t_add = f_chap_en_t_add + DATA_S_chap_add

# Visualisation des prévisions pour la semaine x
plot(y_chap_en_t_add, type = "o", pch = 19, lwd = 2,
     main = "Prévision pour la semaine 2 (modèle additif)",
     xlab = "Jour dans la semaine", ylab = "Nombre de joueurs connectés")
```

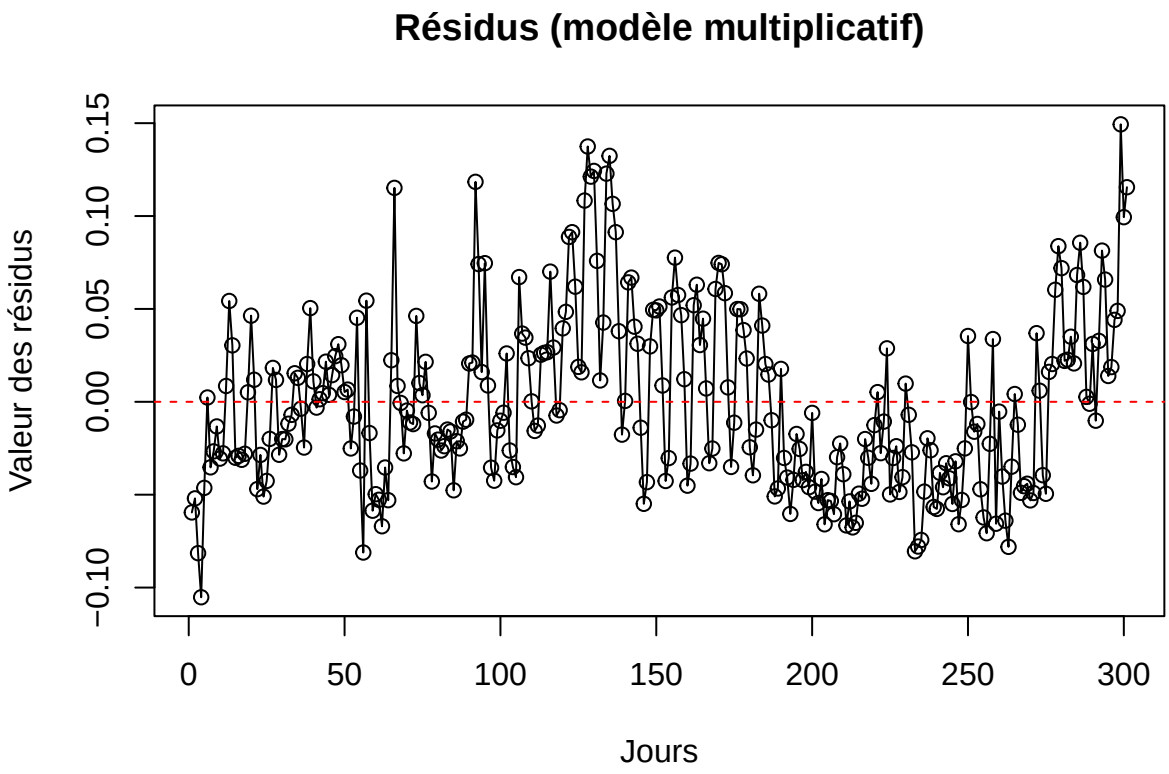


Note : Ces prévisions permettent d'anticiper le nombre de joueurs connectés pour chaque jour d'une semaine spécifique, ce qui peut être utile pour la planification des ressources serveur ou pour des stratégies marketing.

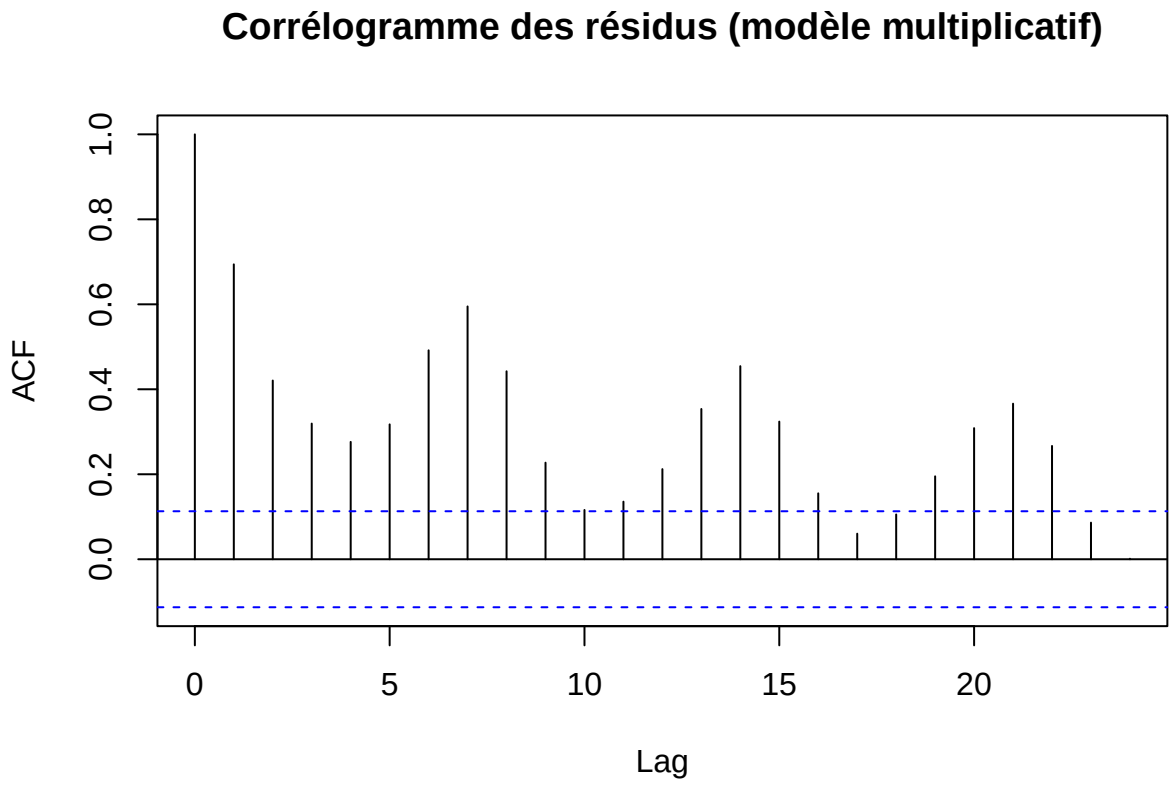
6 6. Analyse des résidus

L'analyse des résidus permet de vérifier la qualité de notre modèle et de détecter d'éventuelles anomalies ou patterns non pris en compte.

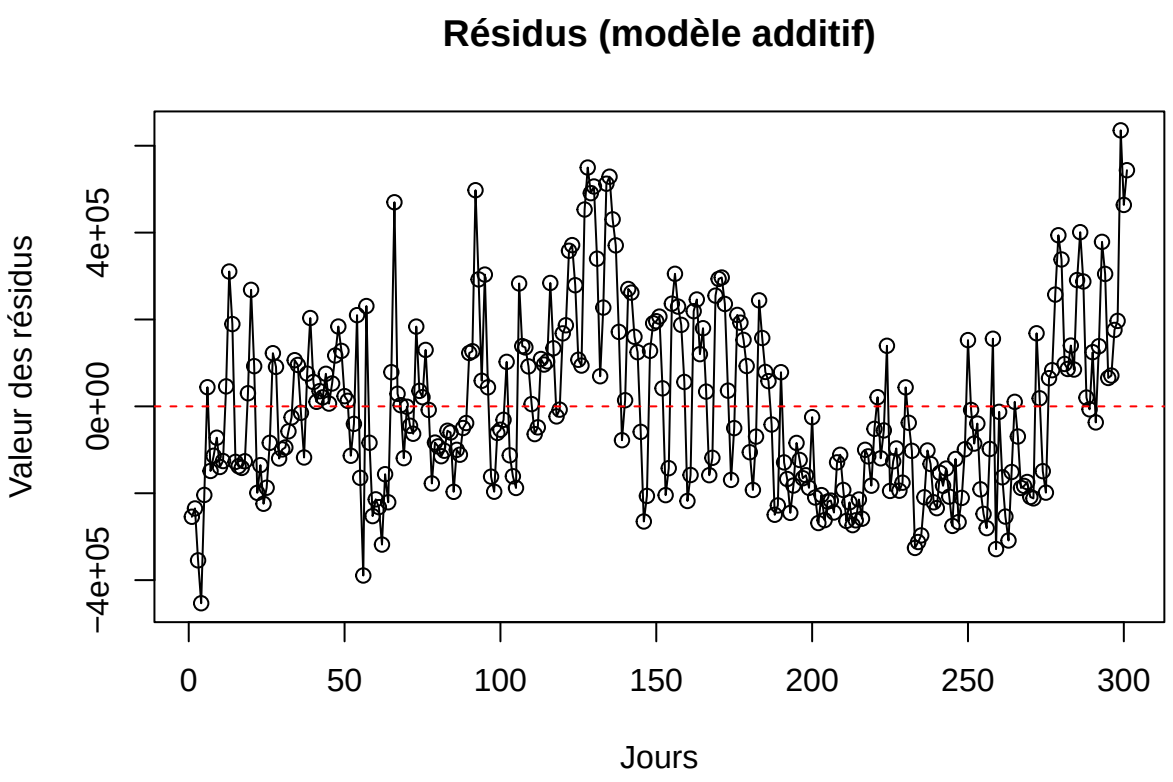
```
# --- Modèle multiplicatif ---
# Calcul des résidus (écart entre valeurs observées et prévues)
res_mult = (DATA/ ((b0_mult+b1_mult*t)*(1+rep(DATA_S_chap_mult, nbr_de_periode)))) - 1
# Alternative: res_mult = (DATA_CVS_mult/(b0_mult+b1_mult*t)) - 1

# Visualisation des résidus
plot(res_mult, type = "o",
     main = "Résidus (modèle multiplicatif)",
     xlab = "Jours", ylab = "Valeur des résidus")
abline(h = 0, col = "red", lty = 2)
```



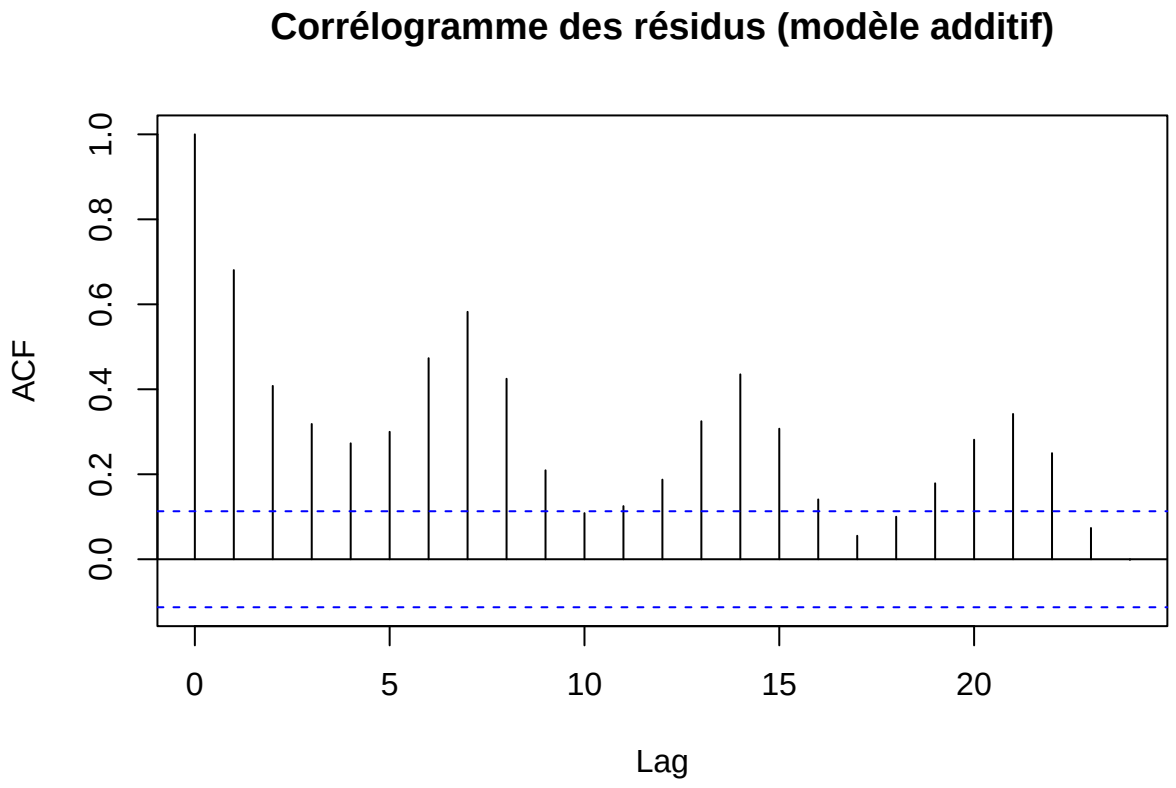

```r
# Autocorrélation des résidus
acf(res_mult, main = "Corrélogramme des résidus (modèle multiplicatif)")
```



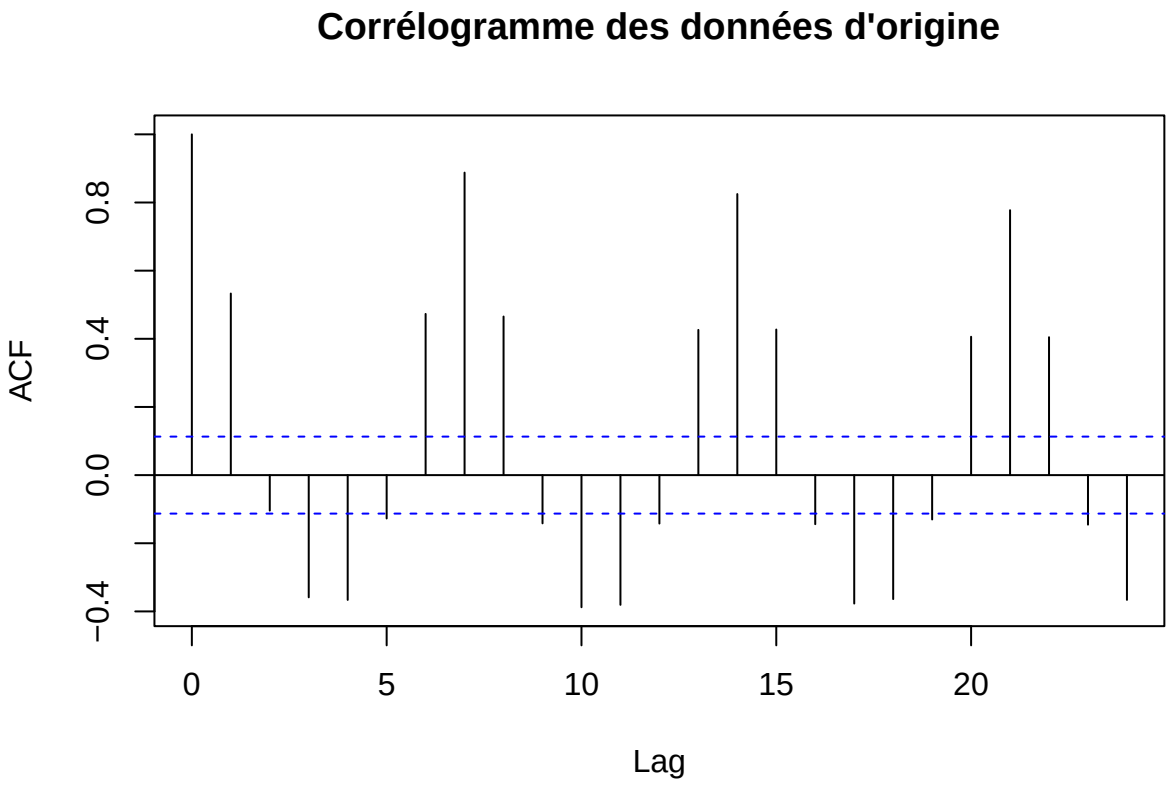

```r
# --- Modèle additif ---
# Calcul des résidus (écart entre valeurs observées et prévues)
res_add = DATA - y_chap_add
# Alternative: res_add = DATA - b0_add - b1_add*t - rep(DATA_S_chap_add, nbr_de_periode)

# Visualisation des résidus
plot(res_add, type = "o",
     main = "Résidus (modèle additif)",
     xlab = "Jours", ylab = "Valeur des résidus")
abline(h = 0, col = "red", lty = 2)
```




```r
# Autocorrélation des résidus
acf(res_add, main = "Corrélogramme des résidus (modèle additif)")
```




```r
# Autocorrélation des données d'origine (pour comparaison)
acf(DATA, main = "Corrélogramme des données d'origine")
```



Interprétation des résidus : - Les résidus oscillent autour de zéro, ce qui est un bon signe - Le corrélogramme des résidus montre encore des pics espacés de 7 jours, ce qui suggère qu'une partie de la saisonnalité n'a pas été complètement capturée par notre modèle - Entre les deux modèles testés, le modèle additif semble présenter des résidus plus stables

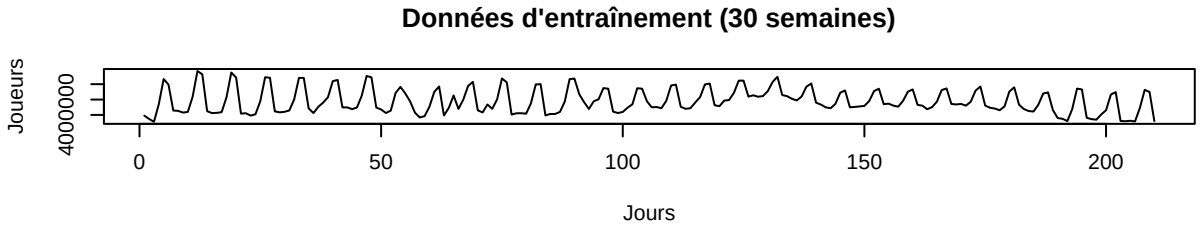
7 7. Prévision à court terme avec méthodes de lissage exponentiel

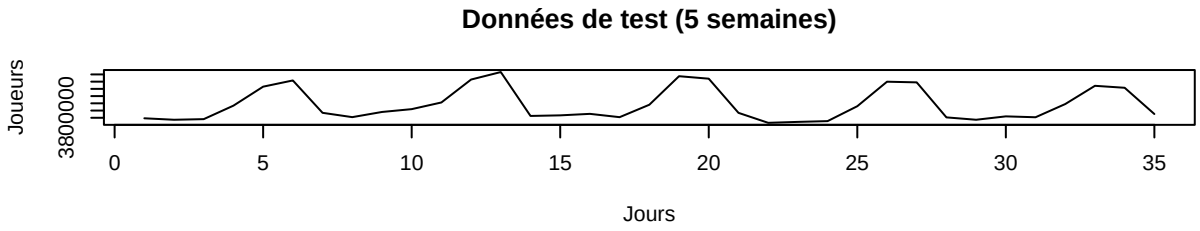
Dans cette section, nous allons comparer trois méthodes de lissage exponentiel pour la prévision à court terme : 1. **Lissage exponentiel simple** : adapté aux séries sans tendance ni saisonnalité 2. **Méthode de Holt** : prend en compte la tendance 3. **Méthode de Holt-Winters** : prend en compte à la fois la tendance et la saisonnalité

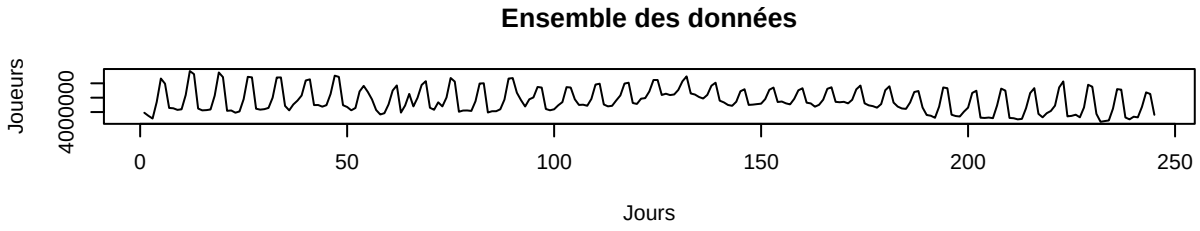
Pour cette analyse, nous diviserons nos données en deux ensembles : - Données d'entraînement : 30 semaines - Données de test : 5 semaines (pour évaluer la qualité des prévisions)

```
# Nombre de semaines pour l'entraînement et le test
nb_smn_train = 30
nb_smn_test = 5

# Récupération du jeu de données complet
DATA <- A$In.Game

# Création des ensembles d'entraînement et de test
DATA_train <- DATA[(5523+1):(5523+(7*nb_smn_train))]
DATA_test <- DATA[(5523+1+(7*nb_smn_train)):(5523+(7*nb_smn_test+7*nb_smn_train))]

# Visualisation des ensembles de données
par(mfrow=c(3,1))
plot(DATA_train, type = "l", main = "Données d'entraînement (30 semaines)",
```

```
     xlab = "Jours", ylab = "Joueurs")
plot(DATA_test, type = "l", main = "Données de test (5 semaines)",
     xlab = "Jours", ylab = "Joueurs")
plot(c(DATA_train, DATA_test), type = "l", main = "Ensemble des données",
     xlab = "Jours", ylab = "Joueurs")
```



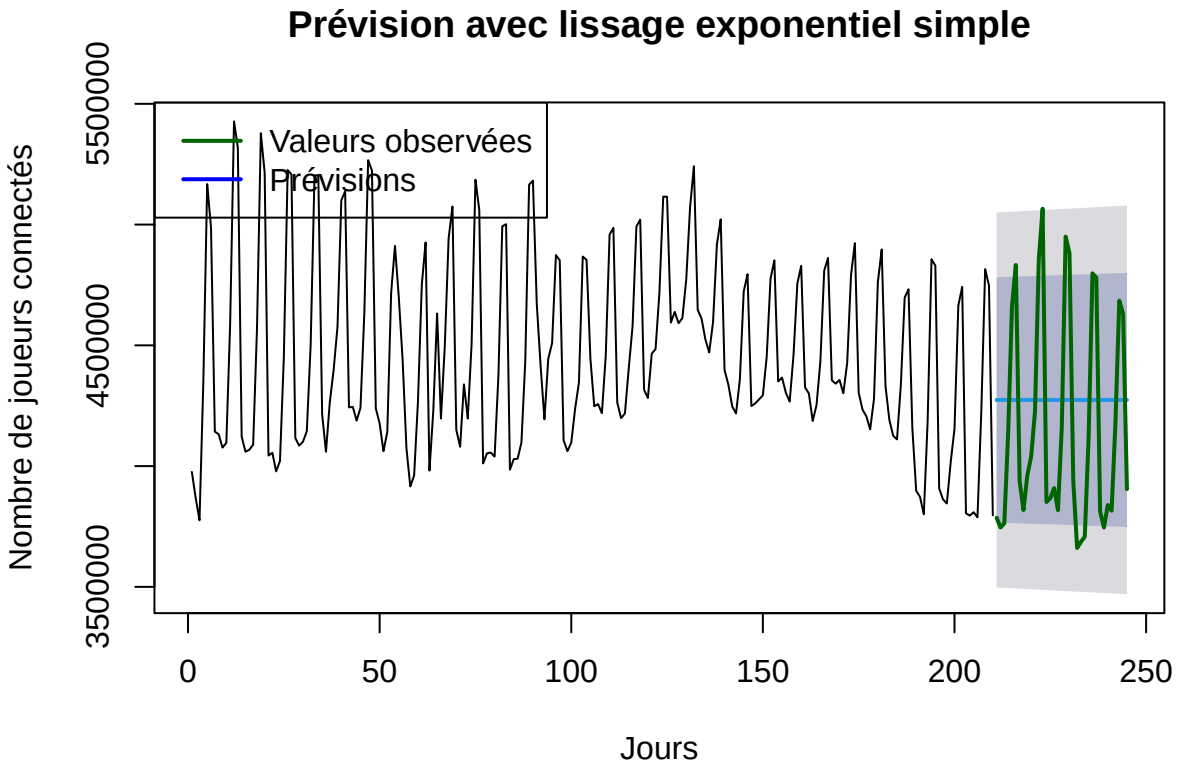





```
par(mfrow=c(1,1))
```

7.1 7.1 Lissage exponentiel simple

```
# Ajustement du modèle de lissage exponentiel simple (sans tendance ni saisonnalité)
fit_LES <- ets(DATA_train, model = "ANN", opt.crit = "mse")

# Prévision pour les 5 prochaines semaines
pred_LES <- forecast(fit_LES, h = (7*nb_smn_test))

# Visualisation des prévisions
plot(pred_LES, main = "Prévision avec lissage exponentiel simple",
     xlab = "Jours", ylab = "Nombre de joueurs connectés")

# Ajout des valeurs réelles pour comparaison
lines(seq(length(DATA_train)+1, length(DATA_train)+length(DATA_test)),
      DATA_test, col = 'darkgreen', lwd = 2)

# Légende
legend('topleft', c("Valeurs observées", "Prévisions"),
       col = c("darkgreen", "blue"), lty = rep(1,2), lwd = rep(2,2))
```

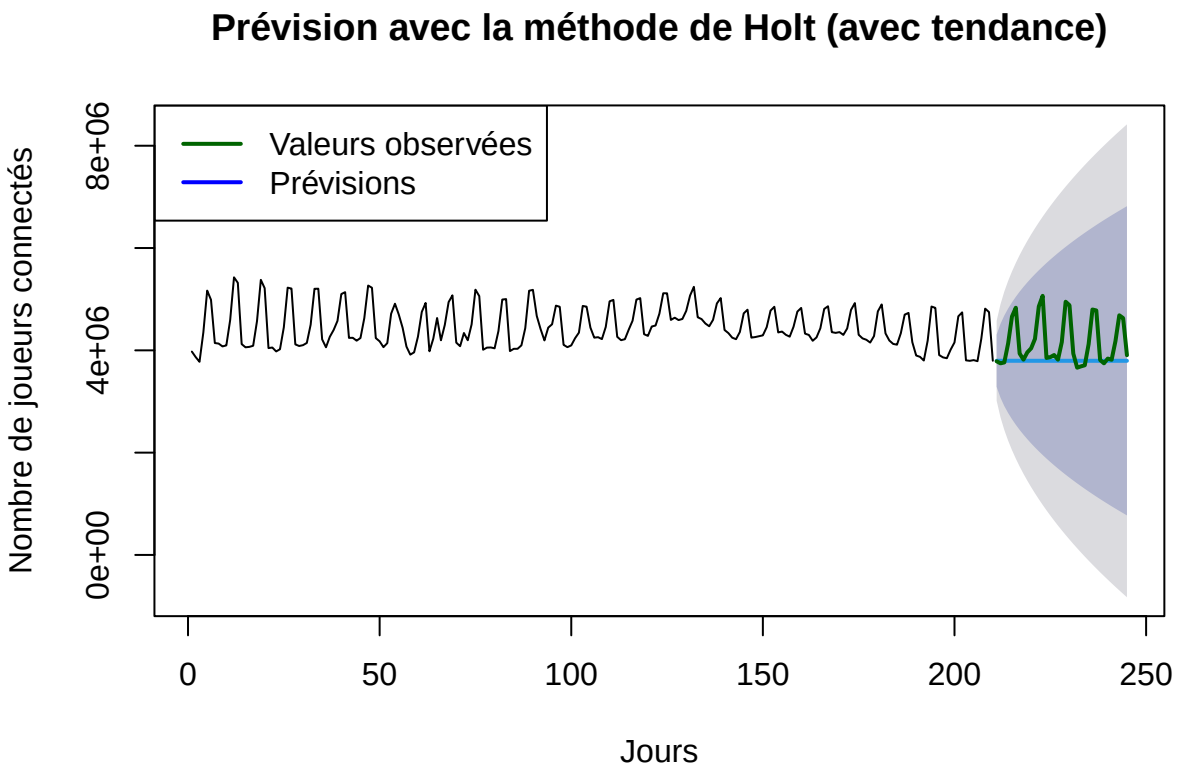



```
# Stockage des prévisions pour comparaison ultérieure
pred_LES_values <- pred_LES$mean
```

Observation : Le lissage exponentiel simple prédit une valeur constante, car il ne prend pas en compte la tendance ni la saisonnalité. Les valeurs réelles fluctuent autour de cette prévision.

7.2 7.2 Méthode de Holt (avec tendance)

```
# Ajustement du modèle de Holt (avec tendance, sans saisonnalité)
fit_H <- ets(DATA_train, model = "AAN")

# Prévision pour les 5 prochaines semaines
pred_H <- forecast(fit_H, h = (7*nb_smn_test))

# Visualisation des prévisions
plot(pred_H, main = "Prévision avec la méthode de Holt (avec tendance)",
     xlab = "Jours", ylab = "Nombre de joueurs connectés")

# Ajout des valeurs réelles pour comparaison
lines(seq(length(DATA_train)+1, length(DATA_train)+length(DATA_test)),
      DATA_test, col = 'darkgreen', lwd = 2)

# Légende
legend('topleft', c("Valeurs observées", "Prévisions"),
       col = c("darkgreen", "blue"), lty = rep(1,2), lwd = rep(2,2))
```

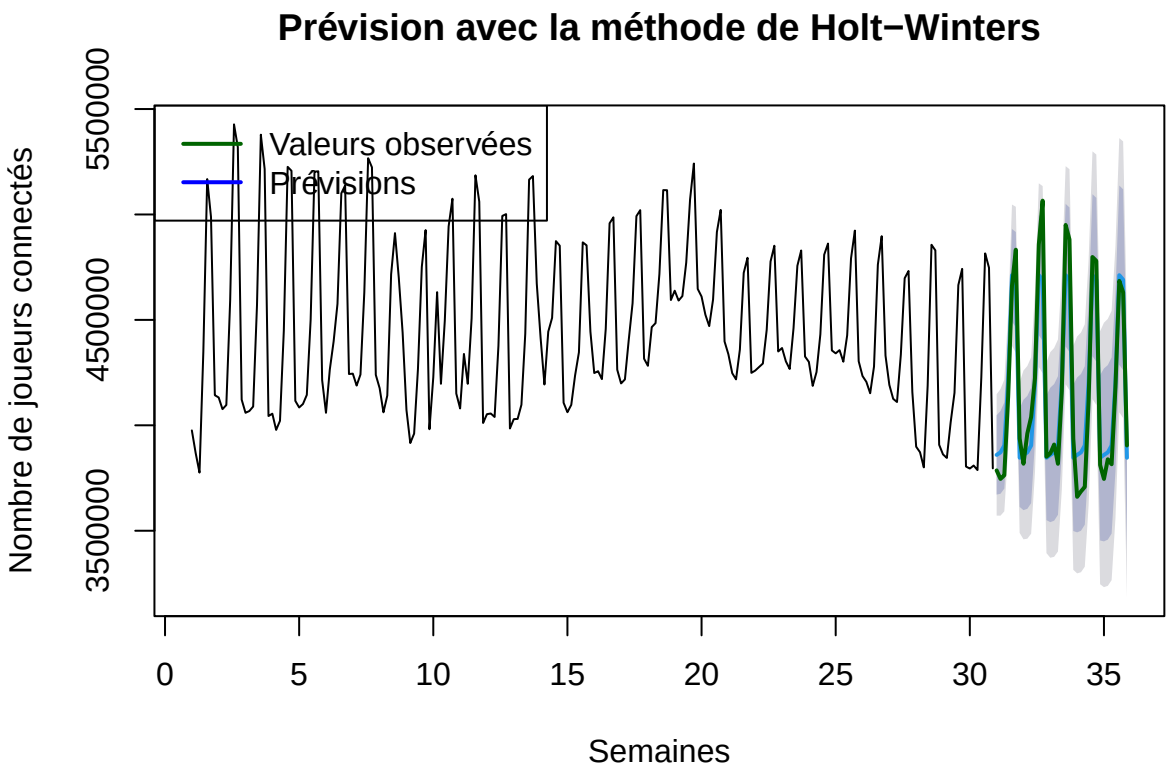



```
# Stockage des prévisions pour comparaison ultérieure
pred_H_values <- pred_H$mean
```

Observation : La méthode de Holt capture la tendance à la baisse, mais ignore toujours les variations hebdomadaires, ce qui explique les écarts importants avec les valeurs réelles.

7.3 7.3 Méthode de Holt-Winters (avec tendance et saisonnalité)

```
# Conversion des données en série temporelle avec une fréquence hebdomadaire
DATA_train_ts <- ts(DATA_train, frequency = 7)
DATA_test_ts <- ts(DATA_test, start = (nb_smn_train+1), frequency = 7)

# Ajustement du modèle de Holt-Winters (avec tendance et saisonnalité)
fit_HW <- ets(DATA_train_ts, model = "AAA")

# Prévision pour les 5 prochaines semaines
pred_HW <- forecast(fit_HW, h = (7*nb_smn_test))

# Visualisation des prévisions
plot(pred_HW, main = "Prévision avec la méthode de Holt-Winters",
     xlab = "Semaines", ylab = "Nombre de joueurs connectés")

# Ajout des valeurs réelles pour comparaison
lines(DATA_test_ts, col = 'darkgreen', lwd = 2)

# Légende
legend('topleft', c("Valeurs observées", "Prévisions"),
       col = c("darkgreen", "blue"), lty = rep(1,2), lwd = rep(2,2))
```




```
# Stockage des prévisions pour comparaison ultérieure
pred_HW_values <- pred_HW$mean
```

Observation : La méthode de Holt-Winters capture à la fois la tendance et le motif saisonnier hebdomadaire, ce qui lui permet de produire des prévisions beaucoup plus proches des valeurs réelles.

7.4 7.4 Comparaison des méthodes de prévision

Pour déterminer quelle méthode fournit les meilleures prévisions, nous allons calculer l'erreur quadratique moyenne (MSE - Mean Squared Error) pour chaque modèle.

```
# Calcul du MSE pour chaque modèle
MSE_LES = mean((DATA_test - pred_LES_values)^2)
MSE_H = mean((DATA_test - pred_H_values)^2)
MSE_HW = mean((DATA_test - pred_HW_values)^2)

# Création d'un tableau comparatif
mse_table <- data.frame(
  Modèle = c("Lissage Exponentiel Simple", "Holt (avec tendance)", "Holt-Winters (avec tendance et
 ↪  saisonnalité)"),
  MSE = c(MSE_LES, MSE_H, MSE_HW)
)

# Affichage du tableau
print(mse_table)
```

```
##                                    Modèle          MSE
```

```
## 1                    Lissage Exponentiel Simple 207771129497
## 2                          Holt (avec tendance) 326727082571
## 3 Holt-Winters (avec tendance et saisonnalité)  15594648439
```

```r
# Identification du meilleur modèle
best_model_index <- which.min(c(MSE_LES, MSE_H, MSE_HW))
best_model_name <- mse_table$Modèle[best_model_index]
best_model_mse <- mse_table$MSE[best_model_index]

cat("\nLe modèle le plus performant est:", best_model_name,
    "avec un MSE de", round(best_model_mse, 2), "\n")
```

```
##
## Le modèle le plus performant est: Holt-Winters (avec tendance et saisonnalité) avec un MSE de 155946
```

8 8. Conclusion

Notre analyse des données de joueurs Steam en 2019 a révélé plusieurs caractéristiques importantes :

1. **Saisonnalité hebdomadaire marquée** : Le nombre de joueurs connectés augmente significativement en fin de semaine (vendredi, samedi, dimanche), ce qui correspond aux habitudes de loisirs typiques.
2. **Légère tendance à la baisse** : Sur la période étudiée, on observe une légère diminution du nombre moyen de joueurs connectés.
3. **Comparaison des modèles** :

- Le modèle additif semble mieux adapté que le modèle multiplicatif pour décrire nos données
- Parmi les méthodes de prévision, la méthode de Holt-Winters est la plus performante car elle prend en compte à la fois la tendance et la saisonnalité

Ces résultats permettent plusieurs applications pratiques : - **Dimensionnement des infrastructures serveur** : Anticiper les pics d'utilisation en fin de semaine pour ajuster les ressources en conséquence - **Planification des maintenances** : Privilégier les périodes de faible affluence (milieu de semaine) - **Tarification publicitaire** : Ajuster les tarifs en fonction de l'affluence prévue - **Planification des promotions et événements** : Optimiser le timing des campagnes marketing

La méthode de Holt-Winters est donc recommandée pour effectuer des prévisions fiables du nombre de joueurs connectés à court et moyen terme.