Моделі часових рядів

Олексій Веретьонкін

2025-08-28

Вибір та зчитування часового ряду

Яу часовий ряд було обрано статистику використання пошукового рушія Google. Дані було завантаженно з датасету https://www.kaggle.com/datasets/michau96/internet-search-engine-worldwide

Зчитуємо дані для та відображажмо графік використання з 2009 по 2024 (кінець 2023 року):

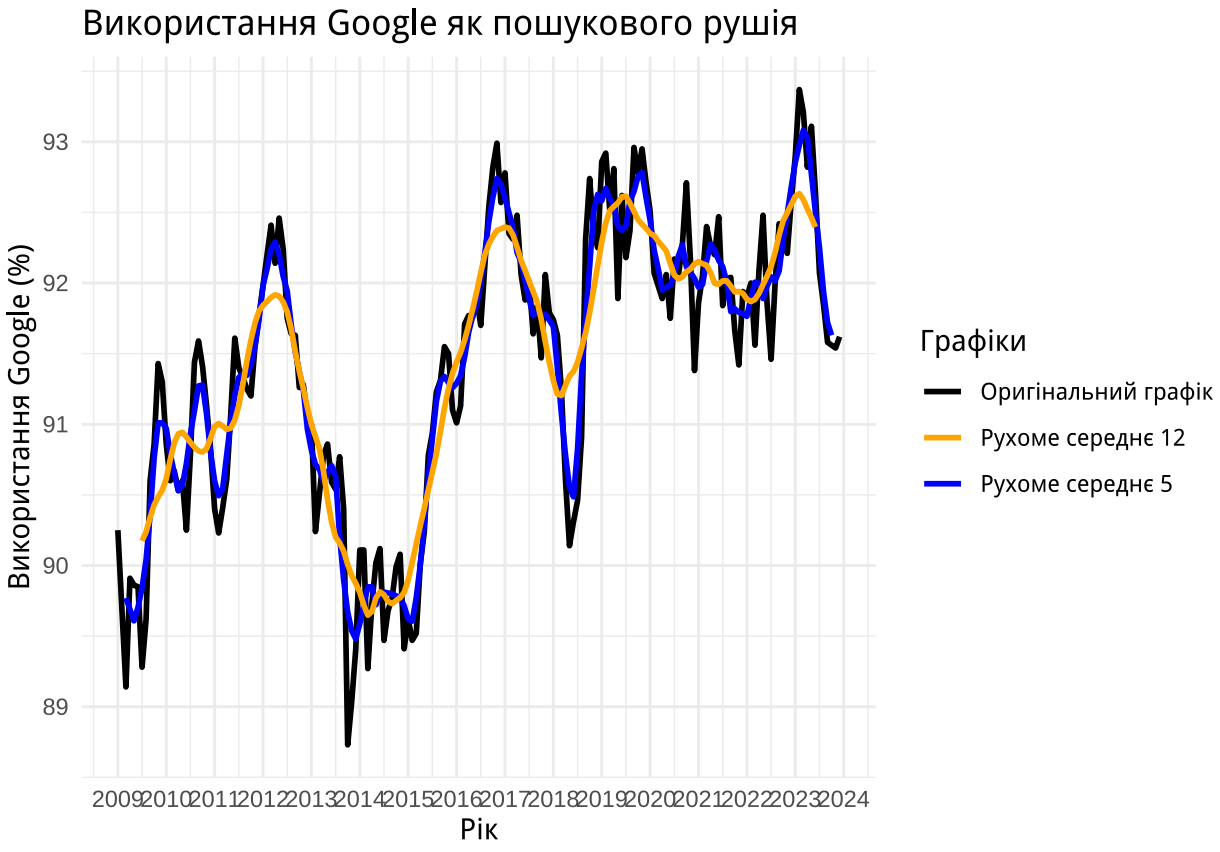
```
# Зчитування файлу
data <- read_csv("search_engine_data.csv")

google_data <- head(data$Google, -10)

google_ts <- ts(google_data, start = c(2009, 1), frequency = 12)

# Конвертація в датафрейм для ggplot2
df <- data.frame(Date = time(google_ts), Google = as.numeric(google_ts))

ggplot(data.frame(Date = time(google_ts), Google = as.numeric(google_ts)),
       aes(x = Date, y = Google)) +
  geom_line() +
  scale_x_continuous(breaks = seq(2009, 2024, by = 1)) +  # Деталізація підписів
  scale_y_continuous(breaks = seq(0, 100, by = 1)) +
  labs(title = "Використання Google як пошукового рушія", x = "Рік", y = "Використання Google (%)") +
  theme_minimal()
```

Згладжування часового ряду

Виконаємо зглажування методом рухомого середнього з порядком 5 та 12:

```
# Згладжування з кроком 5
moving_avarate_with_order_5 <- ma(google_ts, order = 5)

# Згладжування з кроком 12
moving_avarate_with_order_12 <- ma(google_ts, order = 12)

# Відображаємо результати
ggplot() +
  geom_line(
    data = data.frame(Date = time(google_ts), Google = as.numeric(google_ts)),
    aes(x = Date, y = Google, colour = 'Оригінальний графік', ),
    linewidth = 1
  ) +
  geom_line(
    data = data.frame(
      Date = time(moving_avarate_with_order_5),
      Google = as.numeric(moving_avarate_with_order_5)
    ),
    aes(x = Date, y = Google, colour = 'Рухоме середнє 5'),
    linewidth = 1
  ) +
```

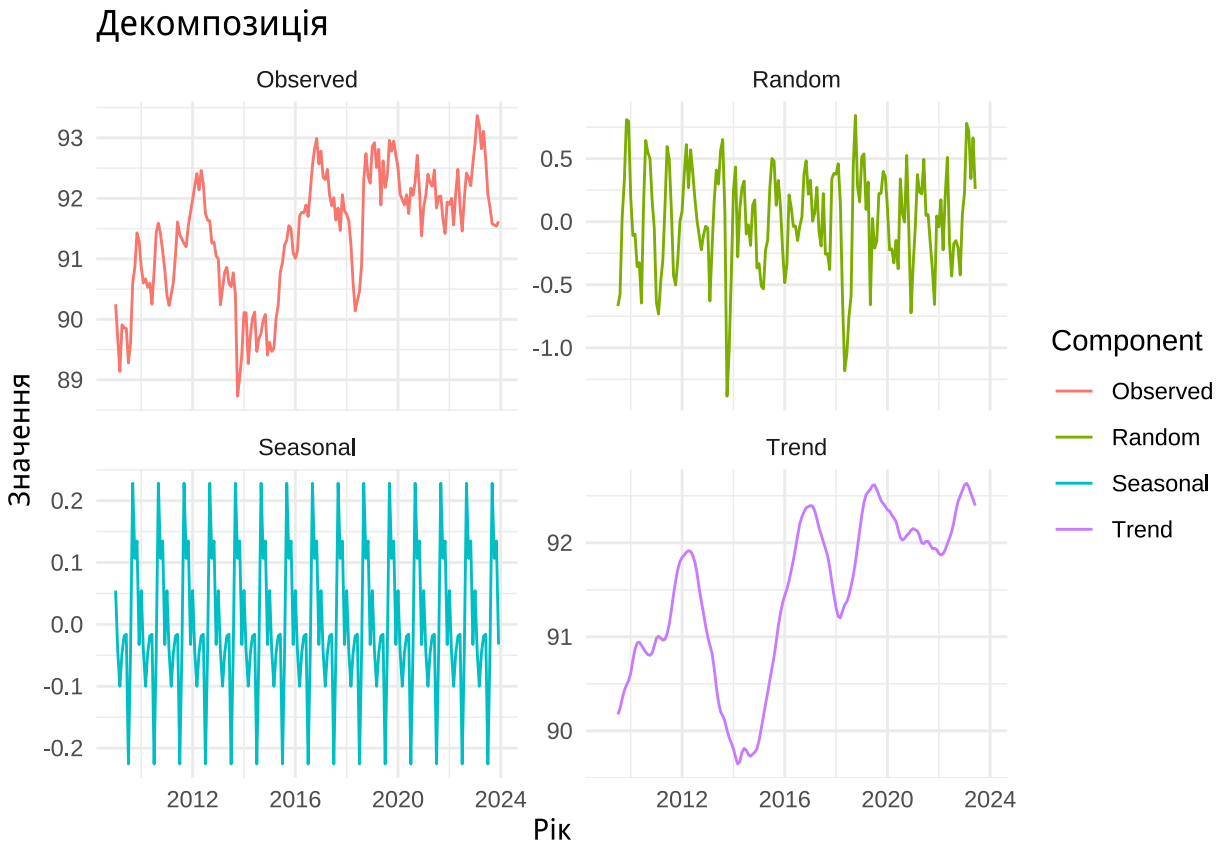
```r
geom_line(
  data = data.frame(
    Date = time(moving_avarate_with_order_12),
    Google = as.numeric(moving_avarate_with_order_12)
  ),
  aes(x = Date, y = Google, colour = 'Рухоме середнє 12'),
  linewidth = 1
) +
scale_color_manual(
  values = c(
    "Оригінальний графік" = 'black',
    "Рухоме середнє 5" = "blue",
    "Рухоме середнє 12" = "orange"
  )
) +
scale_x_continuous(breaks = seq(2009, 2024, by = 1)) +  # Деталізація підписів
labs(
  title = "Використання Google як пошукового рушія",
  x = "Рік",
  y = "Використання Google (%)",
  colour = 'Графіки'
) +
theme_minimal()
```



Декомпозиція часового ряду

Виконуємо декомпозицію ряду та відображаємо графіки компонент:

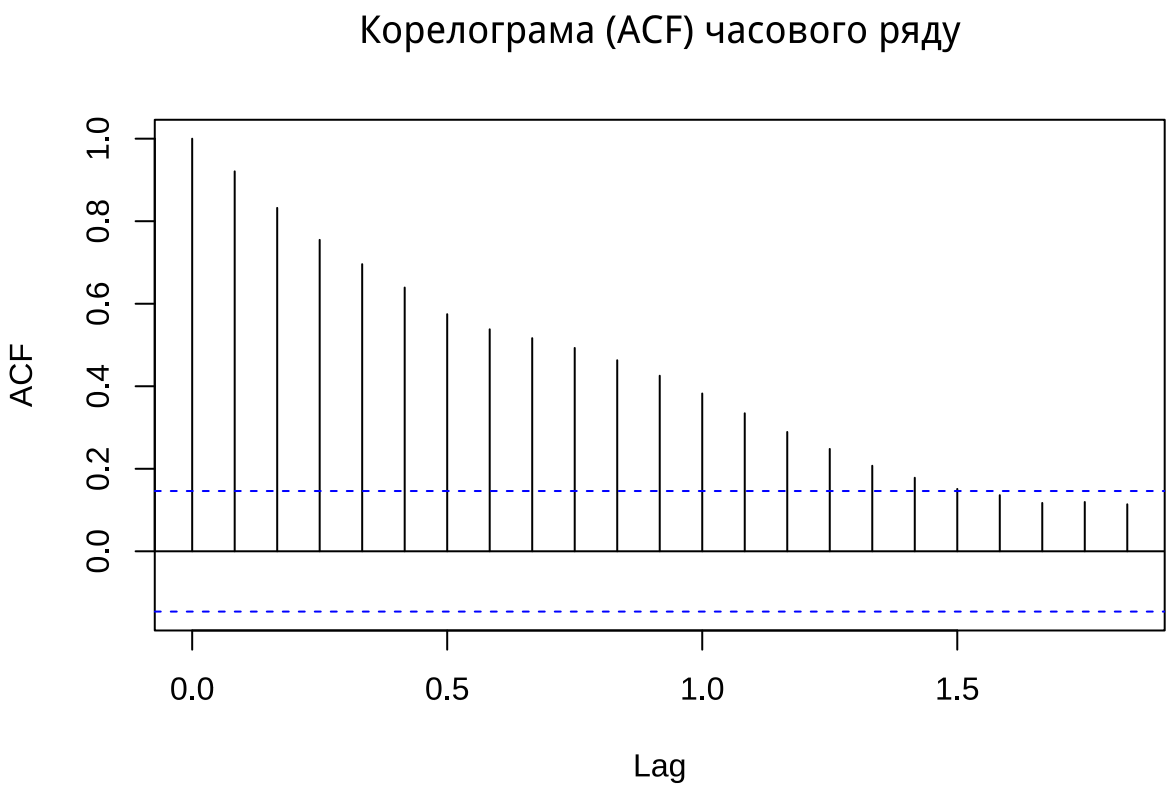
```
decomposed_google_ts <- decompose(google_ts, type = "additive")

# Використання decompose
decomposed <- decomposed_google_ts

# Перетворення результатів у data.frame для ggplot
decomposed_df <- data.frame(
  Time = time(google_ts),
  Observed = as.numeric(decomposed_google_ts$x),
  Trend = as.numeric(decomposed_google_ts$trend),
  Seasonal = as.numeric(decomposed_google_ts$seasonal),
  Random = as.numeric(decomposed_google_ts$random)
) %>%
  pivot_longer(cols = -Time,
               names_to = "Component",
               values_to = "Value")

# Побудова графіка
ggplot(decomposed_df, aes(x = Time, y = Value, color = Component)) +
  geom_line() +
  facet_wrap( ~ Component, scales = "free_y") +
  theme_minimal() +
  labs(
    title = "Декомпозиція",
    x = "Рік",
    y = "Значення",
    color = "Component"
  )
```



Побудова корелограми та часткової корелограми

Будуємо корелограму та часткову корелограму (використано стандартний інструмент побудови графіків замість ggplot)

```
acf(google_ts, main = "Корелограма (ACF) часового ряду")
```




```
pacf(google_ts, main = "Часткова корелограма (PACF) часового ряду")
```

Трансформація часового ряду для досягнення стаціонарності, перевірка на стаціонарність

Побудовані корелограми дають підстави вважати що ряд не є стаціонарним. Зробимо перевірку на стаціонарність тестом Дікі-фуллера. Якщо тест вкаже на те що ряд не стаціонарний - продиференціюємо його для досягнення стаціонарності та знову проведемо тест. Важливим параметром для тесту є p-value - якщо його значення менше за 0.05 то ряд можна вважати стаціонарним.

```
# Тест Дікі-Фуллера для перевірки стаціонарності
# Якщо p-value < 0.05, відхиляємо H₀, і ряд вважається стаціонарним.
ts_frequency <- frequency(google_ts)

adf.test(google_ts, alternative = "stationary", k = ts_frequency)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  google_ts
## Dickey-Fuller = -2.9591, Lag order = 12, p-value = 0.1754
## alternative hypothesis: stationary
```

```
# Якщо ряд не стаціонарний, можна застосувати логарифмування та диференціювання
log_google_ts <- log(google_ts)
diff_google_ts <- diff(google_ts)
diff_log_google_ts <- diff(log_google_ts)

# Повторний тест
adf.test(diff_google_ts, alternative = "stationary", k = ts_frequency)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_google_ts
## Dickey-Fuller = -3.6138, Lag order = 12, p-value = 0.03385
## alternative hypothesis: stationary
```

Аналізуємо результати: для оригіналтного ряду ми отримали p-value = 0.175, тому ряд не стаціонарний; після диференціювання p-value = 0.03385, тобто можна вважати що ряд було зведено до стаціонарного.

Вибір моделі та прогнозування

Для прогнозування використаємо модель Хольта-Вінтерса та модель ARIMA з автоматичним підбором параметрів

```
# Модель Хольта-Вінтерса
hw_model <- HoltWinters(diff_google_ts)

# Прогноз на 24 місяці вперед
hw_forecast <- forecast(hw_model, h = 24)
```

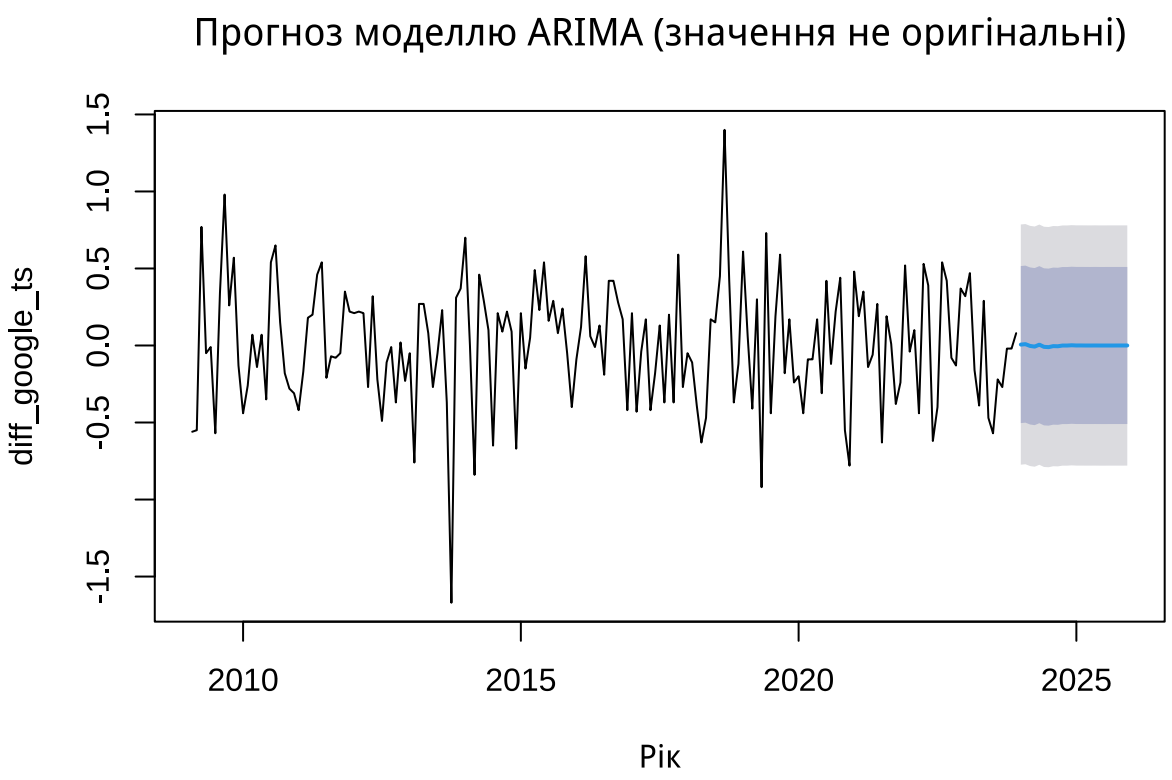
```r
# Відображаємо прогноз
plot(hw_forecast,
    main = "Прогноз методом Хольта-Вінтерса (значення не оригінальні)",
    ylab = "diff_google_ts",
    xlab = "Рік")
```


```r
# Автоматичний підбір моделі ARIMA
auto_arima_model <- auto.arima(diff_google_ts, )

# Параметри моделі
auto_arima_model
```

```
## Series: diff_google_ts
## ARIMA(0,0,0)(1,0,0)[12] with zero mean
##
## Coefficients:
##         sar1
##       0.0187
## s.e.  0.0784
##
## sigma^2 = 0.1582:  log likelihood = -88.49
## AIC=180.98   AICc=181.05   BIC=187.35
```

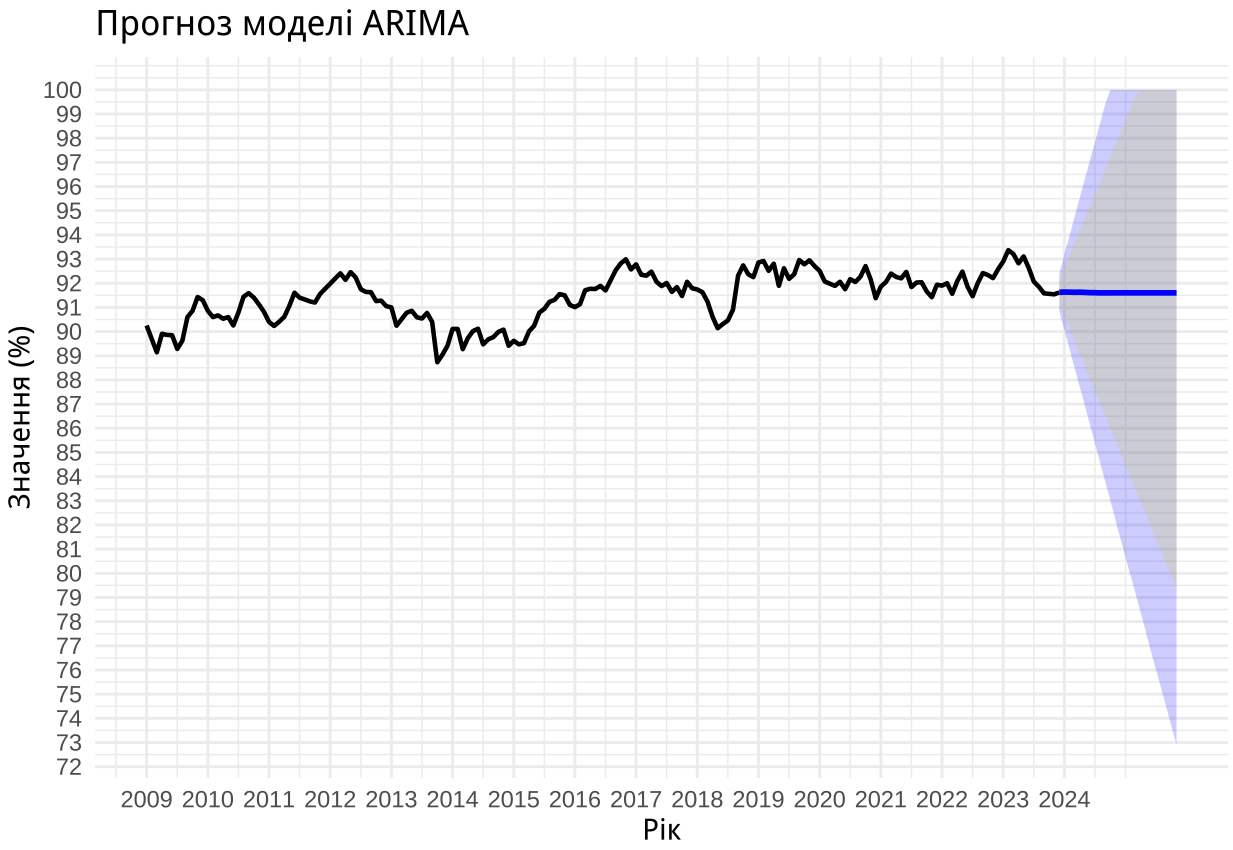
```r
# Прогноз на 24 місяці вперед
arima_forecast <- forecast(auto_arima_model, h = 24)

# Відображаємо прогноз
plot(arima_forecast,
```

```
  main = "Прогноз моделлю ARIMA (значення не оригінальні)",
  ylab = "diff_google_ts",
  xlab = "Рік")
```




```
# Отримуємо останнє значення початкового ряду (перетворюємо в число)
last_value <- as.numeric(tail(google_ts, 1))

build_forcast_plot <- function(forecast, last_value, google_ts, title) {
 # Відновлений часовий ряд з правильною шкалою часу
 arima_forecast_ts <- ts(
  cumsum(forecast$mean) + last_value,
  start = c(2023, 12),
  frequency = 12
 )

 lower_95_original <- pmax(cumsum(forecast$lower[, 2]) + last_value, 0)
 upper_95_original <- pmin(cumsum(forecast$upper[, 2]) + last_value, 100)

 lower_80_original <- pmax(cumsum(forecast$lower[, 1]) + last_value, 0)
 upper_80_original <- pmin(cumsum(forecast$upper[, 1]) + last_value, 100)

 # Підготовка даних для ggplot2
 forecast_df <- data.frame(
  Date = time(arima_forecast_ts),
  Forecast = as.numeric(arima_forecast_ts),
  Lower_80 = lower_80_original,
  Upper_80 = upper_80_original,
  Lower_95 = lower_95_original,
```

```r
    Upper_95 = upper_95_original
  )

  # Фактичні дані
  actual_df <- data.frame(Date = time(google_ts), Actual = as.numeric(google_ts))

  # Побудова графіка
  ggplot() +
    # Фактичні дані
    geom_line(
      data = actual_df,
      aes(x = Date, y = Actual),
      color = "black",
      size = 0.8
    ) +
    # Довірчий інтервал 80%
    geom_ribbon(
      data = forecast_df,
      aes(x = Date, ymin = Lower_80, ymax = Upper_80),
      fill = "yellow",
      alpha = 0.2
    ) +
    # Довірчий інтервал 95%
    geom_ribbon(
      data = forecast_df,
      aes(x = Date, ymin = Lower_95, ymax = Upper_95),
      fill = "blue",
      alpha = 0.2
    ) +
    # Лінія прогнозу
    geom_line(
      data = forecast_df,
      aes(x = Date, y = Forecast),
      color = "blue",
      size = 1
    ) +

    scale_x_continuous(breaks = seq(2009, 2024, by = 1)) +
    scale_y_continuous(breaks = seq(0, 100, by = 1)) +
    labs(title = title, x = "Рік", y = "Значення (%)") +
    theme_minimal()
}

build_forcast_plot(
  forecast = arima_forecast,
  last_value = last_value,
  google_ts = google_ts,
  title = "Прогноз моделі ARIMA"
)
```

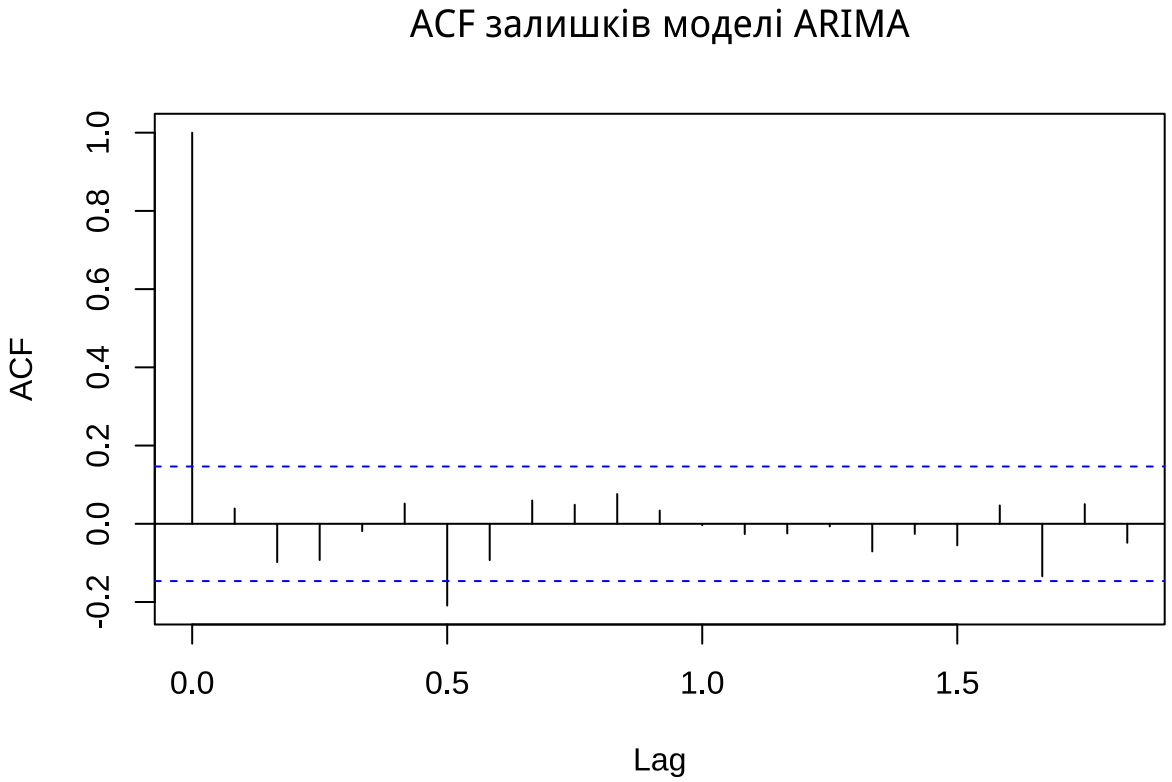



```
build_forcast_plot(
  forecast = hw_forecast,
  last_value = last_value,
  google_ts = google_ts,
  title = "Прогноз методом Хольта-Вінтерса"
)
```

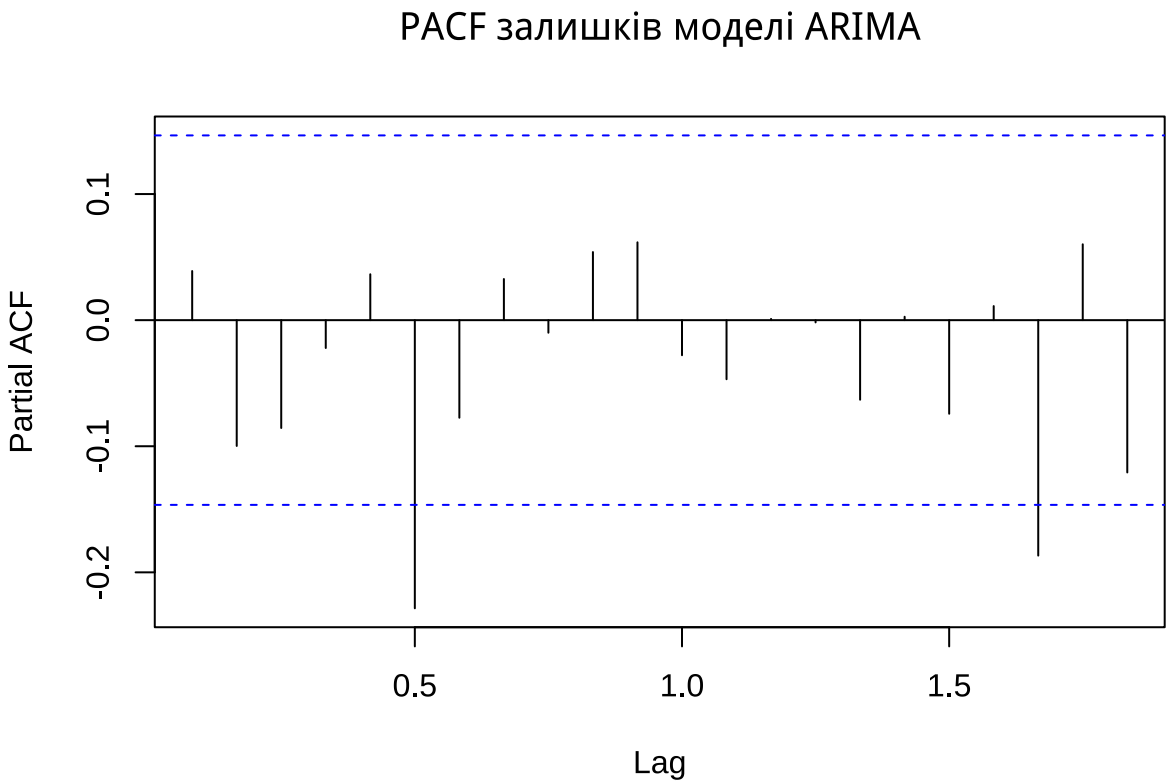
Аналіз залишків та оцінка якості прогнозу

Виконаємо аналіз побудованих прогнозів. Побудуємо корелограму залишків та зробимо тест Льюнга-Бокса щоб перевірити наявність автокореляції. Для тесту Льюнга-Бокса - якщо P-value менше 0.05, це вказує на те що модель недосконала

```
# Залишки моделі ARIMA
arima_residuals <- residuals(auto_arima_model)

# Корелограма залишків
acf(arima_residuals, main = "ACF залишків моделі ARIMA")
```



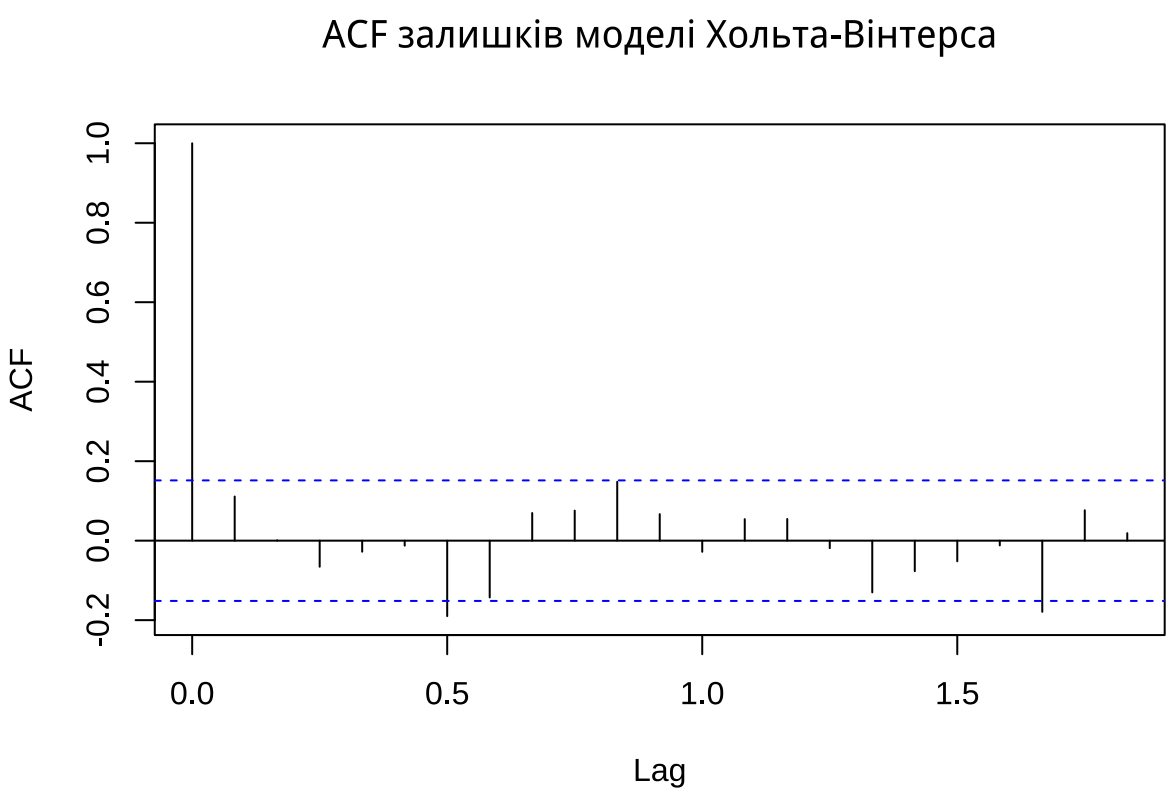

```r
pacf(arima_residuals, main = "PACF залишків моделі ARIMA")
```



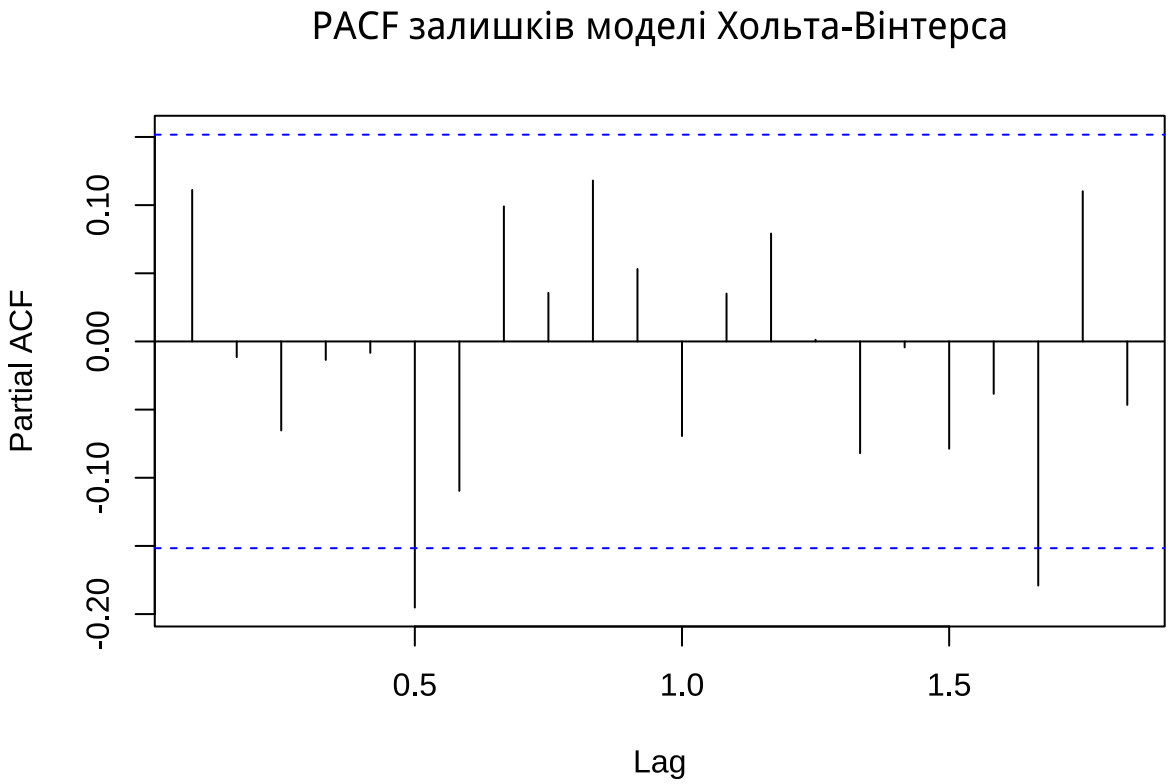
```r
# Тест Льюнга-Бокса:
# P-value менше 0.05 вказує на наявність автокореляції в залишках,
# що може свідчити про недосконалість моделі.
Box.test(arima_residuals, type = "Ljung-Box", lag = 12)
```

```
##
##  Box-Ljung test
##
## data:  arima_residuals
## X-squared = 16.454, df = 12, p-value = 0.1713
```

```r
# Залишки моделі Хольта-Вінтерса
hw_residuals <- residuals(hw_model)

# Корелограма залишків
acf(hw_residuals, main = "ACF залишків моделі Хольта-Вінтерса")
```




```r
pacf(hw_residuals, main = "PACF залишків моделі Хольта-Вінтерса")
```




```
# Тест Льюнга-Бокса
Box.test(hw_residuals, type = "Ljung-Box", lag = 12)
```

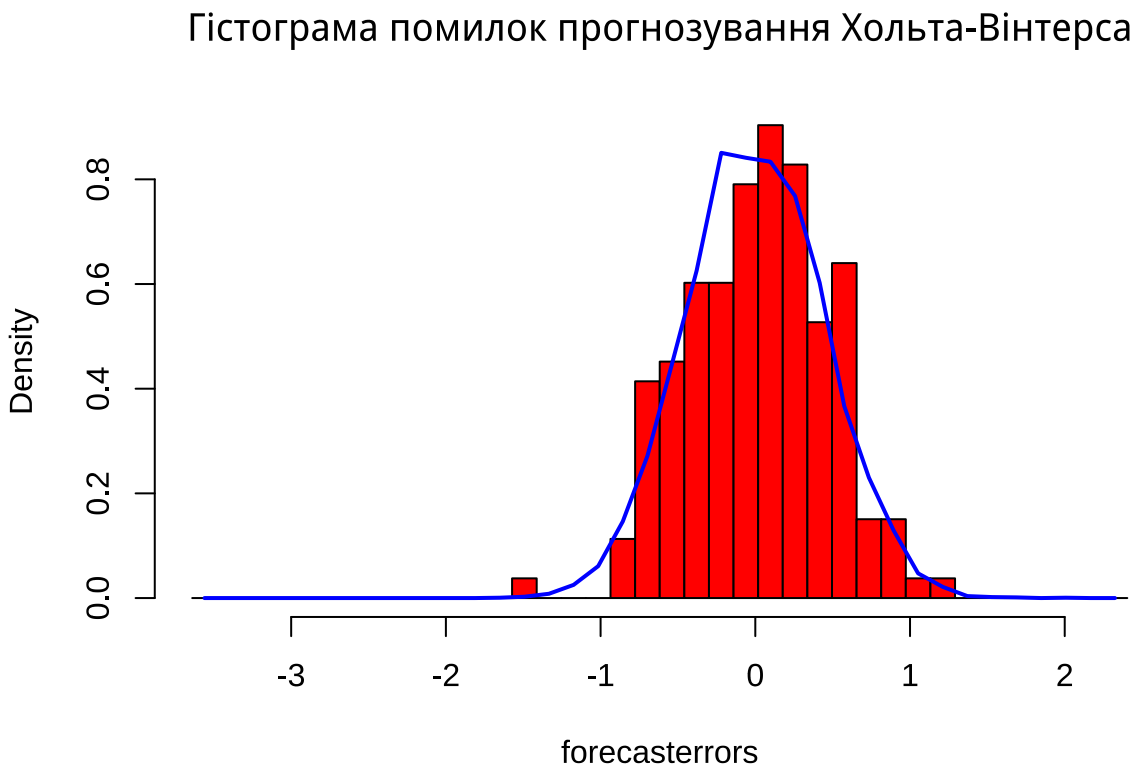
```
##
##  Box-Ljung test
##
## data:  hw_residuals
## X-squared = 19.715, df = 12, p-value = 0.07268
```

В результаті аналізу отримуємо P-Value = 0.07268 для моделі Хольта-Вінтерса та P-Value = 0.1713 для моделі ARIMA. Оскільки ці значення більше за 0.05 то залишки можна вважати випадковими, і моделі непогано описують дані.

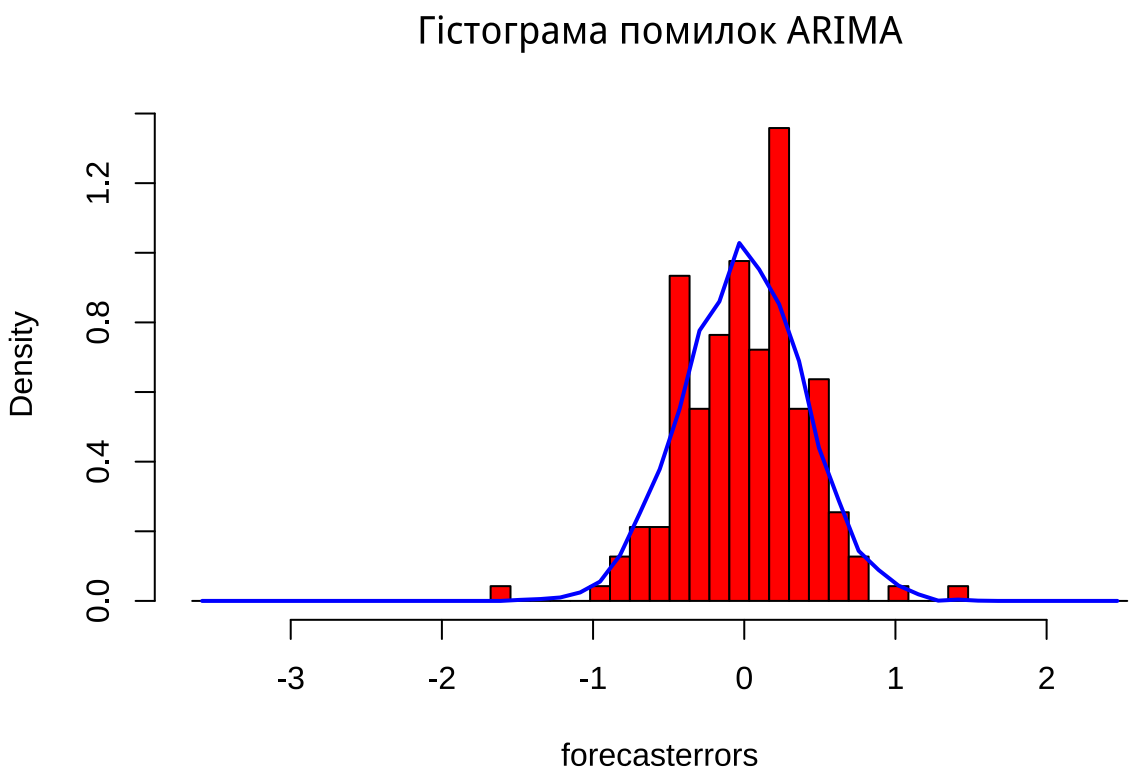
Побудуємо гістограму помилок прогнозу з накладеною нормальною кривою:

```
plotForecastErrors <- function(forecasterrors, title="Гістограма помилок прогнозування") {
  # make a histogram of the forecast errors:
  mybinsize <- IQR(forecasterrors)/4
  mysd   <- sd(forecasterrors)
  mymin  <- min(forecasterrors) - mysd*5
  mymax  <- max(forecasterrors) + mysd*3
  # generate normally distributed data with mean 0 and standard deviation mysd
  mynorm <- rnorm(10000, mean=0, sd=mysd)
  mymin2 <- min(mynorm)
  mymax2 <- max(mynorm)
  if (mymin2 < mymin) { mymin <- mymin2 }
  if (mymax2 > mymax) { mymax <- mymax2 }
  # make a red histogram of the forecast errors, with the normally distributed data overlaid:
```

```
  mybins <- seq(mymin, mymax, mybinsize)
  hist(forecasterrors, col="red", freq=FALSE, breaks=mybins, main=title)
  # freq=FALSE ensures the area under the histogram = 1
  # generate normally distributed data with mean 0 and standard deviation mysd
  myhist <- hist(mynorm, plot=FALSE, breaks=mybins)
  # plot the normal curve as a blue line on top of the histogram of forecast errors:
  points(myhist$mids, myhist$density, type="l", col="blue", lwd=2)
}


plotForecastErrors(hw_residuals, title = "Гістограма помилок прогнозування Хольта-Вінтерса")
```




```
plotForecastErrors(arima_residuals, title = "Гістограма помилок ARIMA")
```


Гістограма помилок ARIMA
Density
0.0
0.4
0.8
1.2
-3
-2
-1
0
1
2
forecasterrors