PHP: Wykres temperatury Polski z regresją liniową

Wykres można obejrzeć tutaj: https://wiki.ostrowski.net.pl/php_mysql/pol_temp.php

Ten artykuł przedstawia przykład programu w PHP, który pobiera dane z bazy danych MySQL, a następnie za pomocą biblioteki JavaScript **Chart.js** wyświetla wykres liniowy średnich temperatur w Polsce wraz z prostą linią trendu (regresją liniową).

Dane wejściowe

Program korzysta z bazy danych MySQL o nazwie `polandtemperature`, która zawiera tabelę `temp` z następującymi kolumnami:

- `ID` – identyfikator wiersza,
- `Date` – data pomiaru,
- `Temp` – wartość temperatury.

Połączenie z bazą danych

Połączenie z bazą danych realizowane jest za pomocą PDO:

```
$conn = new
PDO("mysql:host=localhost;dbname=polandtemperature;charset=utf8mb4",
"viewer", "viewer");
```

Jeśli połączenie się nie powiedzie, program zakończy działanie z komunikatem błędu.

Pobieranie i przetwarzanie danych

Program wykonuje zapytanie SQL:

```
SELECT * FROM temp ORDER BY ID;
```

Wyniki zapisywane są do tablicy PHP. Z kolumn `Date` i `Temp` wyodrębniane są osobne tablice:

```
$labels = array_column($Data, 'Date');
$temps = array_column($Data, 'Temp');
```

Obliczanie regresji liniowej

W celu dodania trendu liniowego, wykonywane są obliczenia regresji liniowej metodą najmniejszych kwadratów:

```
$slope = ($n * $sum_xy - $sum_x * $sum_y) / ($n * $sum_x2 - $sum_x ** 2);
$intercept = ($sum_y - $slope * $sum_x) / $n;
```

Następnie generowana jest druga tablica zawierająca dane dla linii trendu:

```
$trendLine = array_map(fn($x) => round($slope * $x + $intercept, 2),
$x_vals);
```

Wyświetlanie wykresu za pomocą Chart.js

W HTML wyświetlany jest wykres z dwiema seriami danych:

- rzeczywiste dane temperatur (`Poland Average Temperature`) – czerwona linia,
- linia trendu (`Linear Trend Line`) – przerywana niebieska linia.

```
datasets: [
    {
        label: 'Poland Average Temperature',
        data: [...],
        borderColor: 'rgba(255, 99, 132, 1)'
    },
    {
        label: 'Linear Trend Line',
        data: [...],
        borderColor: 'rgba(54, 162, 235, 1)',
        borderDash: [5, 5]
    }
]
```

Biblioteka **Chart.js** generuje responsywny wykres, który można osadzić na stronie WWW.

Efekt końcowy

Użytkownik widzi liniowy wykres temperatur wraz z prostą, która pokazuje ogólny trend (np. ocieplanie się klimatu lub spadki temperatur). Trend ułatwia interpretację danych historycznych.

Podsumowanie

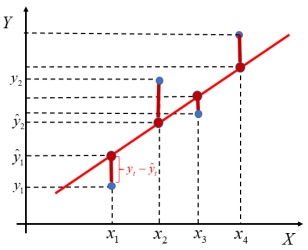
Ten program demonstruje:

- Jak pobrać dane z MySQL w PHP,
- Jak obliczyć regresję liniową,
- Jak wykorzystać Chart.js do wizualizacji danych i trendów.

Dzięki temu rozwiązaniu możemy łatwo tworzyć dynamiczne, interaktywne wykresy statystyczne w aplikacjach webowych.

Matematyka: regresja liniowa

Poniższy fragment opisuje metodę najmniejszych kwadratów (ang. *least squares*) stosowaną do wyznaczenia parametrów prostej regresji liniowej, czyli współczynników nachylenia i wyrazu wolnego.



Źródło: blog.etrapez.pl

Metoda polega na minimalizacji sumy kwadratów odchyleń (residuals) pomiędzy rzeczywistymi wartościami y_i a wartościami przewidywanymi $\hat{y}_i$ przez model liniowy $$y = \beta_0 + \beta_1 x$$, co wyraża funkcja kryterium: $$ S(\beta_0, \beta_1) \;=\;\sum_{i=1}^n \bigl(y_i - (\beta_0 + \beta_1 x_i)\bigr)^2. $$

Aby znaleźć optymalne β_0 i β_1, rozwiązujemy układ tzw. równań normalnych: $$ \begin{cases} \displaystyle \frac{\partial S}{\partial \beta_1} \;=\; -2 \sum_{i=1}^n x_i\,(y_i - \beta_0 - \beta_1 x_i)\;=\;0,\\[1em] \displaystyle \frac{\partial S}{\partial \beta_0} \;=\; -2 \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)\;=\;0. \end{cases} $$

Rozwiązując ten układ, otrzymujemy wzory na estymatory:

- $$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2},$$
- $$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1\,\bar{x},$$

gdzie $\bar{x} = \frac{1}{n}\sum x_i$ i $\bar{y} = \frac{1}{n}\sum y_i$.

Inna, równoważna postać wzoru na nachylenie prostej korzysta z sum iloczynów i sum kwadratów: multiline $$ \hat{\beta}_1 = \frac{n\sum_{i=1}^n x_i y_i \;-\; \sum_{i=1}^n x_i \sum_{i=1}^n y_i} {n\sum_{i=1}^n x_i^2 \;-\; \bigl(\sum_{i=1}^n x_i\bigr)^2}\,, $$ a wyraz wolny: multiline $$ \hat{\beta}_0 = \frac{\sum_{i=1}^n y_i - \hat{\beta}_1\sum_{i=1}^n x_i}{n} =\bar{y}- \hat{\beta}_1\bar{x}\,. $$

W praktycznej implementacji, gdy x_i to kolejne indeksy czasowe (0, 1, …, n-1), obliczenia skracają się do wersji:
$$x_i = i,\quad y_i = \text{Temp}[i],$$ pozwalając łatwo wygenerować tablicę wartości trendu:
$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1\,i.$$

Interpretacja parametrów:

- $\hat{\beta}_1$ – średnia zmiana y przy wzroście x o jednostkę, czyli nachylenie trendu liniowego
- $\hat{\beta}_0$ – przewidywana wartość y dla $x=0$, czyli punkt przecięcia z osią OY

Dzięki tym wzorom możemy obliczyć linię trendu, która najlepiej przybliża dane w sensie najmniejszych kwadratów, ułatwiając analizę długoterminowych tendencji.

Kod

pol_temp.php

```php
<?php
// Database connection settings
$serverName = "localhost";
$database = "polandtemperature";
$username = "";
$password = "";

// Connect using PDO for MySQL
try {
    $conn = new
PDO("mysql:host=$serverName;dbname=$database;charset=utf8mb4",
$username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}

// Fetch data
$Data = [];
$stmt = $conn->query("SELECT * FROM temp ORDER BY ID;");
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    $Data[] = $row;
}

// Close DB connection
$conn = null;

// Extract columns
$labels = array_column($Data, 'Date');
$temps = array_column($Data, 'Temp');

// Convert dates to numeric values (e.g. index) for regression
$x_vals = range(0, count($temps) - 1);
$y_vals = $temps;

// Linear regression calculation (y = a * x + b)
$n = count($x_vals);
$sum_x = array_sum($x_vals);
$sum_y = array_sum($y_vals);
$sum_xy = array_sum(array_map(fn($x, $y) => $x * $y, $x_vals,
$y_vals));
$sum_x2 = array_sum(array_map(fn($x) => $x * $x, $x_vals));

$slope = ($n * $sum_xy - $sum_x * $sum_y) / ($n * $sum_x2 - $sum_x **
2);
$intercept = ($sum_y - $slope * $sum_x) / $n;
```

```
// Generate trend line data
$trendLine = array_map(fn($x) => round($slope * $x + $intercept, 2),
$x_vals);
?>
<!DOCTYPE html>
<html>
<head>
    <title>Temperature Chart with Trend Line</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <h2>Temperature Trend: Poland Average</h2>
    <canvas id="tempChart" width="800" height="400"></canvas>

    <script>
        const ctx =
document.getElementById('tempChart').getContext('2d');
        const tempChart = new Chart(ctx, {
            type: 'line',
            data: {
                labels: <?= json_encode($labels) ?>,
                datasets: [
                    {
                        label: 'Poland Average Temperature',
                        data: <?= json_encode($temps) ?>,
                        borderColor: 'rgba(255, 99, 132, 1)',
                        fill: false,
                        tension: 0.1
                    },
                    {
                        label: 'Linear Trend Line',
                        data: <?= json_encode($trendLine) ?>,
                        borderColor: 'rgba(54, 162, 235, 1)',
                        borderDash: [5, 5],
                        fill: false,
                        pointRadius: 0,
                        tension: 0
                    }
                ]
            },
            options: {
                responsive: true,
                scales: {
                    y: {
                        beginAtZero: false,
                        title: {
                            display: true,
                            text: 'Temperature (°C)'
                        }
                    },
```

```
                    x: {
                        title: {
                            display: true,
                            text: 'Date'
                        }
                    }
                }
            }
        });
    </script>
</body>
</html>
```