PHP: Graph of Poland's temperature with linear regression

The graph can be viewed here: https://wiki.ostrowski.net.pl/php_mysql/pol_temp.php

This article shows an example of a program in PHP that retrieves data from a MySQL database and then, using the JavaScript library **Chart.js** displays a line graph of average temperatures in Poland together with a simple trend line (linear regression).

Input data

The program uses a MySQL database named `polandtemperature`, which contains a `temp` table with the following columns:

- `ID` - row identifier,
- `Date` - the date of the measurement,
- `Temp` - temperature value.

Connection to the database

The connection to the database is implemented using PDO:

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

If the connection fails, the program terminates with an error message.

Data collection and processing

The programme executes a SQL query:

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

The results are written to a PHP array. Separate arrays are extracted from the `Date` and `Temp` columns:

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

Linear regression calculation

In order to add a linear trend, linear regression calculations are performed using the least squares method:

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

A second table containing the data for the trend line is then generated:

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

Displaying the chart using Chart.js

The HTML displays a chart with two series of data:

- actual temperature data (`Poland Average Temperature`) - red line,
- the trend line (`Linear Trend Line`) - dashed blue line.

```
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]
    }
]
```

Library **Chart.js** generates a responsive chart that can be embedded in a web page.

The end result

The user sees a line graph of temperatures together with a straight line that shows the overall trend (e.g. warming or falling temperatures). The trend facilitates the interpretation of historical data.

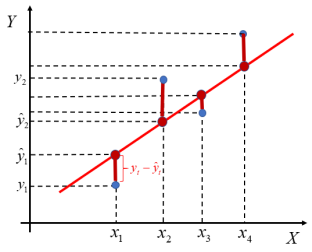
Summary

This programme demonstrates:

- How to retrieve data from MySQL in PHP,
- How to calculate a linear regression,
- How to use Chart.js to visualise data and trends.

With this solution we can easily create dynamic, interactive statistical charts in web applications.

Mathematics: linear regression



Source: blog.etrapez.pl

The excerpt below describes the method of least squares (*least squares*) used to determine the parameters of a simple linear regression, namely the slope coefficients and the free expression.

The method is based on minimising the sum of squares of the deviations (residuals) between the actual values of y and the predicted values of y by a linear model $$y = \beta_0 + \beta_1 x$$, as expressed by the criterion function: $$ S(\beta_0, \beta_1) $$;=\sum_{i=1}^n \bigl(y_i - (\beta_0 + \beta_1 x_i)\bigr)^2. $$ To find the optimal ¯beta_0 and ¯beta_1, we solve a system of so-called normal equations: $$ \$begin{cases} \$ ¢displaystyle ¢frac{partial S}{partial ¢beta_1} \Ù;= Ù; -2 Ùsum_{i=1}^n x_i,(y_i - Ùbeta_0 - Ùbeta_1 x_i)Ù;= Ù;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} $$ Solving this system, we obtain the formulae for the estimators: * $$\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{x}_0 = {barbar{y} - $$hat{x}_1},$$. where $(_bar{x} = _frac{1}{n}sum x_i') and $(_bar{y} = _frac{1}{n}sum y_i').

Another equivalent form of the formula for the slope of a straight line uses sums of products and sums of squares: multiline $$ \Ùhat{sum of squares}_1 ==\{frac{nsum_{i=1}^n x_i y_i }^n x_i {nsum_{i=1}^n y_i} =^{{sum_{i=1}^n x_i^2 }^2}, $$ a free word: multiline $$ \} = \{frac{sum_{i=1}^n y_i - {hat{beta}_1{sum_{i=1}^n x_i}{n} =\bar{y}-\hat{\beta}_1\bar{x}\,. $$

In a practical implementation, when \(x_i} is consecutive time indexes (0, 1, ..., n-1), the computation shortens to a version:
$$x_i = i,\quad y_i = \text{Temp}[i],$$. allowing an array of trend values to be easily generated:
$$hat{y}_i = ‖0 + ‖hat{temp}_1‖,$$.

Interpretation of parameters:

- * ¯(¯hat{beta}_1) - the average change in ¯(y) with an increase in ¯(x) by a unit, i.e. the slope of the linear trend.
- * ¯hat{beta}_0) - the predicted value of ¯(y) for ¯(x=0), i.e. the point of intersection with the OY axis.

With these formulae, we can calculate the trend line that best approximates the data in a least squares sense, facilitating the analysis of long-term trends.

Code

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>
```