ROOT Project - Part 1: Maximum Likelihood

1: Generate 50 Gaussian Data Points

- **Objective:** Create a starting sample of 50 measurements from a Gaussian distribution with true parameters $\mu_{true} = 0.2$ and $\sigma_{true} = 0.1$.
- **ROOT Code ('ML_Project_Part1.C' snippet):**

```
const double true_mean = 0.2;
const double true_sigma = 0.1;
const int N_measurements = 50;

TRandom3 rng(12345); // Seed RNG for reproducibility
std::vector<double> measurements;

for (int i = 0; i < N_measurements; ++i) {
    measurements.push_back(rng.Gaus(true_mean, true_sigma));
}
```

- **Code Breakdown:**
  - `true_mean`, `true_sigma`, `N_measurements`: Define the true mean, true standard deviation, and sample size.
  - `TRandom3 rng(12345)`: Sets up ROOT's random number generator with a fixed seed to ensure the results are reproducible.
  - `measurements`: A `std::vector` to store the generated numbers.
  - The `for` loop iterates 50 times, with `rng.Gaus(...)` pulling a random number from the specified Gaussian distribution and adding it to the `measurements` vector.
- **Outcome:** The `measurements` vector now holds 50 floating-point numbers, representing our "observed" data for the project.

2: ML Estimation of Mean and Variance

- **Objective:** Using the `measurements` sample, calculate the Maximum Likelihood estimates for the mean ($\hat{\mu}_{ML}$) and standard deviation ($\hat{\sigma}_{ML}$).
- **Key Formulas:**
  - The ML estimate for the mean is the sample mean:

$$\hat{\mu}_{ML} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

  - The ML estimate for the variance is the biased sample variance (dividing by N):

$$\hat{\sigma}^2_{ML} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu}_{ML})^2$$

- **ROOT Code ('ML_Project_Part1.C' snippet):**

```
// Calculate ML estimate for mean (ml_mean)
double sum = std::accumulate(measurements.begin(), measurements.end(), 0.0);
double ml_mean = sum / N_measurements;

// Calculate ML estimate for variance and sigma (ml_sigma)
double sum_squared_diff = 0.0;
for (double x : measurements) {
    sum_squared_diff += std::pow(x - ml_mean, 2);
}
double ml_variance = sum_squared_diff / N_measurements; // Note: division by N
double ml_sigma = std::sqrt(ml_variance);
```

- **Outcome from Execution:** The script calculated the following data-driven best guesses for the parameters:
  - **`ml_mean`: 0.188094**
  - **`ml_sigma`: 0.0829625**

  These values differ slightly from the true values of 0.2 and 0.1, as expected from statistical fluctuation in a finite sample.

3: Plot Data, True PDF, and Estimated PDF

- **Objective:** Visualize the generated data in a histogram, and overlay two Gaussian PDFs: one using the true parameters and one using our ML-estimated parameters.
- **ROOT Code ('ML_Project_Part1.C' snippet):**

```
// Create and fill data histogram
TH1F* h_data = new TH1F("h_data", "Sample of 50 Measurements", 20, min, max);
for (double x : measurements) { h_data->Fill(x); }

// Function with ML parameters
TF1* f_ml = new TF1("f_ml", "[0]*TMath::Gaus(x, [1], [2], 1)", min, max);
f_ml->SetParameter(1, ml_mean);
f_ml->SetParameter(2, ml_sigma);

// Function with true parameters
TF1* f_true = new TF1("f_true", "[0]*TMath::Gaus(x, [1], [2], 1)", min, max);
f_true->SetParameter(1, true_mean);
f_true->SetParameter(2, true_sigma);

// Scale functions to match the histogram area
double bin_width = h_data->GetBinWidth(1);
double scale_factor = h_data->Integral() * bin_width;
f_ml->SetParameter(0, scale_factor);
f_true->SetParameter(0, scale_factor);

// Drawing
TCanvas* c_pdf = new TCanvas("c_pdf", "Gaussian PDF Comparison", 800, 600);
h_data->Draw("E1 HIST");
f_ml->Draw("SAME");
f_true->Draw("SAME");
// ... (legend is added here)
```

Project Output and Explanation (PDFs)

The blue curve, representing the PDF derived from the Maximum Likelihood estimates ($\hat{\mu}_{ML} = 0.188094$ and $\hat{\sigma}_{ML} = 0.0829625$), is reasonably close to the red dashed curve, which is the true PDF ($\mu = 0.2, \sigma = 0.1$). The blue curve is, by definition, the best Gaussian fit to the binned data. Slight differences in peak position and width are observable, which is expected from a limited sample of 50 points.

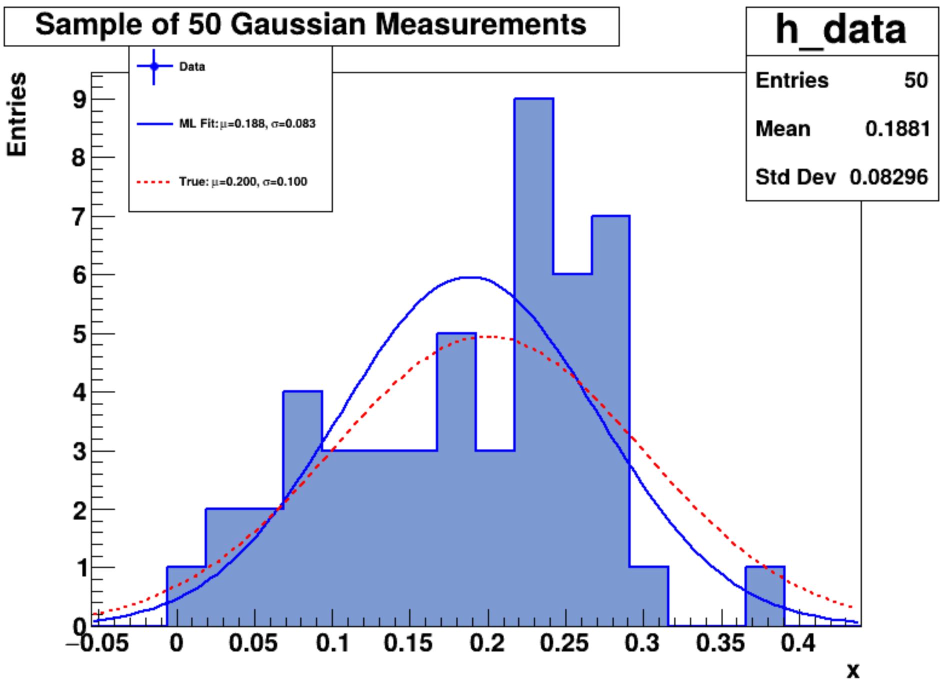

Figure 1: Comparison of the true Gaussian PDF (red, dashed) and the ML-estimated PDF (blue, solid) overlaid on the generated data histogram.

4: Estimate the Variance of the Mean via Monte Carlo

- **Objective:** To understand the variability of our $\hat{\mu}_{ML}$ estimator, we simulate 1000 new "experiments." We will then analyze the distribution of the resulting 1000 mean estimates to find its standard deviation. This standard deviation is our Monte Carlo estimate of the standard error of the mean.
- **ROOT Code (‘ML_Project_Part1.C‘ snippet):**

```
const int N_mc_trials = 1000;
std::vector<double> mean_estimates;

// Use the ML estimates from Step 2 as "true" parameters for the MC
const double mc_true_mean = ml_mean;
const double mc_true_sigma = ml_sigma;

// Main MC loop
for (int trial = 0; trial < N_mc_trials; ++trial) {
    std::vector<double> mc_measurements;
    // Generate 50 measurements for this trial
    for (int i = 0; i < N_measurements; ++i) {
        mc_measurements.push_back(rng.Gaus(mc_true_mean, mc_true_sigma));
    }

    // Calculate ML estimate of mean for this trial and store it
    double mc_sum = std::accumulate(mc_measurements.begin(), mc_measurements.end(), 0.0);
    double mc_mean = mc_sum / N_measurements;
    mean_estimates.push_back(mc_mean);
}

// Histogram the results
TH1F* h_mc = new TH1F("h_mc", "Distribution of ML Mean Estimates", 50, min, max);
for (double mean_est : mean_estimates) {
    h_mc->Fill(mean_est);
}

// Calculate the standard deviation of the distribution of means (unbiased estimator)
```

```
double sum_squared_diff_mc = 0.0;
double mean_of_means = h_mc->GetMean();
for (double mean_est : mean_estimates) {
    sum_squared_diff_mc += std::pow(mean_est - mean_of_means, 2);
}
double s = std::sqrt(sum_squared_diff_mc / (N_mc_trials - 1));
```

- **Outcome from Execution:** A histogram showing the distribution of the 1000 $\hat{\mu}_{ML}$ values was generated. From this distribution, the unbiased standard deviation (s), which quantifies the spread, was calculated. This is the MC estimate of the error.
  - **Unbiased standard deviation (s): 0.0116249**

Project Output and Explanation (MC Mean Estimates)

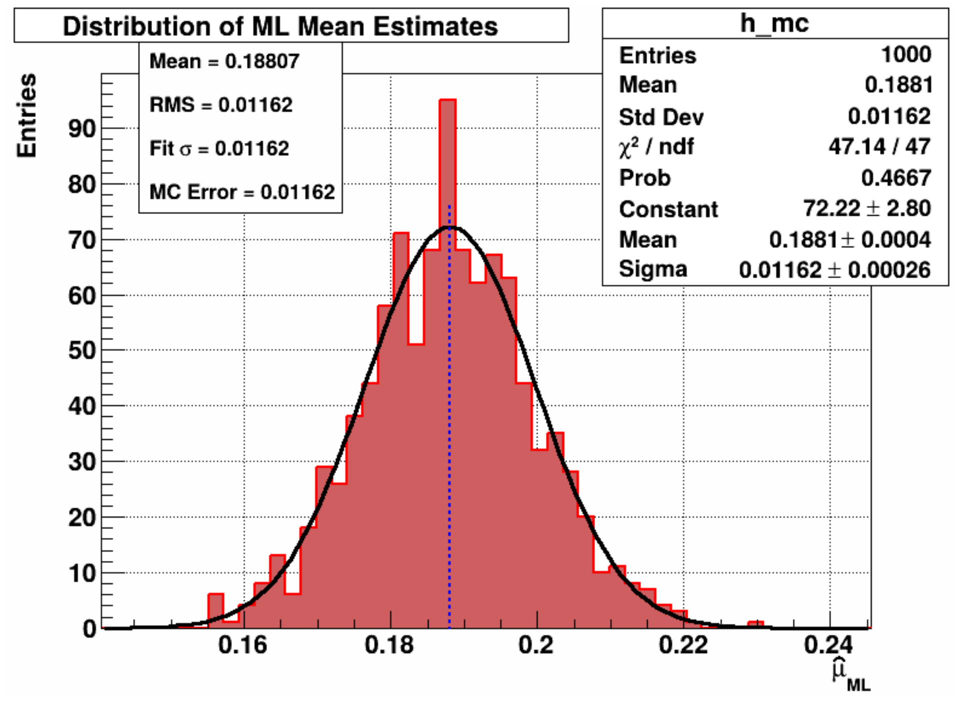

Figure 2: Distribution of 1000 Monte Carlo estimates of the mean ($\hat{\mu}_{ML,j}$). Each $\hat{\mu}_{ML,j}$ was calculated from an independent sample of N=50 data points.

*The histogram displays the distribution of the 1000 $\hat{\mu}_{ML,j}$ values. It approximates a Gaussian distribution, centered near the μ_{gen} = 0.188094 value used for generation. The spread of this distribution visually represents the uncertainty in estimating the mean from a sample of 50 points. The standard deviation of this distribution (s) is our Monte Carlo estimate of the standard error on the mean, calculated to be **0.0116249**.*

5: Compare MC Standard Error with Analytical Solution

- **Objective:** Report the statistical error on the mean found from the Monte Carlo method (s) and compare it with the analytical formula for the standard error of the mean: $\sigma_{\hat{\mu}} = \sigma/\sqrt{N}$.
- **ROOT Code (‘ML_Project_Part1.C‘ snippet):**

```
// The Monte Carlo error is 's', calculated in the previous step.
// Calculate analytical error: sigma / sqrt(n)
// We use mc_true_sigma, which is the sigma used for MC generation (ml_sigma).
```

```cpp
double analytical_error = mc_true_sigma / std::sqrt(N_measurements);

std::cout << "--- Comparing Monte Carlo error with analytical solution ---" << std::endl;
std::cout << "  Monte Carlo error (s): " << s << std::endl;
std::cout << "  Analytical error (sigma/sqrt(n)): " << analytical_error << std::endl;
std::cout << "  Ratio (MC/Analytical): " << s / analytical_error << std::endl;
```

Project Output and Explanation (SE Comparison)

The Monte Carlo estimated standard error of the mean, found by taking the standard deviation of the 1000 mean estimates, was $s_{\hat{\mu}} =$ ***0.0116249****. The analytical standard error,* $\sigma_{\hat{\mu}} = \sigma_{gen}/\sqrt{N} = 0.0829625/\sqrt{50}$*, was calculated as* ***0.0117327****. The ratio of these two values is* $0.0116249/0.0117327 \approx 0.9908$*, indicating a difference of less than 1%. This excellent agreement validates that the Monte Carlo simulation correctly estimates the statistical uncertainty of the ML mean estimator.*