# Contents

**Tutorial 1: Basic Histogram Creation and Filling..... 2**
**Tutorial 2: Creating Graphs with Arrays..... 3**
**Tutorial 3: Reading Data from File to Histogram..... 4**
**Tutorial 4: Reading Two-Column Data to Graph..... 5**
**Tutorial 5: Random Number Generation..... 6**
**Tutorial 6: Gaussian Distribution and File Writing..... 7**
**Tutorial 7: Fitting Data with Functions..... 8**
**Summary of Key ROOT Classes..... 10**
**Common Drawing Options..... 10**

# Tutorial 1: Basic Histogram Creation and Filling

**Concepts:** Creating histograms, manual data entry, axis labels, canvas drawing

```
void tut1()
{
    // Create a 1D histogram
    // TH1F(name, title, nbins, xmin, xmax)
    TH1F *hist = new TH1F("hist","Title",100,0,100);

    // Manually fill histogram with values
    hist->Fill(10);
    hist->Fill(20);

    // Set axis titles
    hist->GetXaxis()->SetTitle("X Axis");
    hist->GetYaxis()->SetTitle("Y Axis");

    // Create canvas and draw
    TCanvas *c1 = new TCanvas();
    hist->Draw();
}
```

**Key Points:**

- `TH1F` creates a 1D histogram with float precision
- Parameters: internal name, display title, number of bins, min value, max value
- `Fill()` adds entries to the histogram
- Always create a `TCanvas` before drawing

# Tutorial 2: Creating Graphs with Arrays

**Concepts:** TGraph objects, marker styling, drawing options

```
void tut2()
{
    // Define x and y data arrays
    double x[5] = {1,2,3,4,5};
    double y[5] = {1,4,9,16,25};

    // Create graph from arrays
    // TGraph(npoints, x_array, y_array)
    TGraph *gr = new TGraph(5, x, y);

    // Customize marker appearance
    gr->SetMarkerStyle(5);  // Cross marker
    gr->SetMarkerSize(4);

    TCanvas *c1 = new TCanvas();
    gr->Draw("ACP");  // A=Axis, C=Smooth curve, P=Points
}
```

**Key Points:**

- `TGraph` is used for x-y scatter plots
- Drawing options: "A" (draw axes), "C" (smooth curve), "P" (points)
- Marker styles can be customized

# Tutorial 3: Reading Data from File to Histogram

**Concepts:** File I/O, reading single-column data, EOF handling

```
void tut3()
{
    TH1F *hist = new TH1F("hist","Title",6,0,100);

    fstream file;
    file.open("data.txt", ios::in);

    double value;

    // Read values until end of file
    while(1){
        file >> value;
        hist->Fill(value);
        if(file.eof()) break;
    }

    file.close();

    TCanvas *c1 = new TCanvas();
    hist->Draw();
}
```

**Key Points:**

- Use `fstream` for file operations
- `ios::in` opens file for reading
- Always check for `eof()` to avoid reading past file end
- Remember to close files after use

# Tutorial 4: Reading Two-Column Data to Graph

**Concepts:** Reading paired data, dynamic point addition, line styling

```
void tut4()
{
    TGraph *gr = new TGraph();

    fstream file;
    file.open("data2.txt",ios::in);

    while(1){
        double x,y;
        file >> x >> y;  // Read two values per line
        gr->SetPoint(gr->GetN(), x, y);  // GetN() returns current point
count
        if(file.eof()) break;
    }
    file.close();

    // Customize appearance
    gr->SetLineColor(kRed);
    gr->SetLineWidth(2);
    gr->SetMarkerStyle(kFullCircle);

    TCanvas *c1 = new TCanvas();
    gr->Draw("ACP");
}
```

**Key Points:**

- `SetPoint(index, x, y)` adds points dynamically
- `GetN()` returns the current number of points
- `kRed`, `kFullCircle` are predefined ROOT constants

# Tutorial 5: Random Number Generation

**Concepts:** Random distributions, uniform distribution, Y-axis range setting

```
void tut5()
{
    TRandom2 *rand = new TRandom2(0);  // Seed = 0

    TH1F *hist = new TH1F("hist","Histogram",100,0,100);

    // Generate 10,000 random numbers
    for(int i=0; i<10000; i++)
    {
        double r = rand->Rndm()*100;  // Uniform [0,100]
        hist->Fill(r);
    }

    TCanvas *c1 = new TCanvas();
    hist->GetYaxis()->SetRangeUser(0,200);  // Set Y-axis range
    hist->Draw();
}
```

**Key Points:**

- `TRandom2` generates random numbers
- `Rndm()` returns uniform random [0,1]
- `SetRangeUser()` manually sets axis display range

# Tutorial 6: Gaussian Distribution and File Writing

**Concepts:** Gaussian generation, writing to file, reading back data

```
void tut6()
{
    TH1F *hist = new TH1F("hist","Title",100,0,10);

    TRandom2 *rand = new TRandom2(0);

    // Write Gaussian data to file
    fstream file;
    file.open("data3.txt",ios::out);
    for(int i=0; i<10000; i++)
    {
        double r = rand->Gaus(5,1);  // mean=5, sigma=1
        file << r << endl;
    }
    file.close();

    // Read data back from file
    file.open("data3.txt", ios::in);
    double value;
    while(1){
        file >> value;
        hist->Fill(value);
        if(file.eof()) break;
    }
   file.close();

    hist->GetXaxis()->SetTitle("Gaussian");
    hist->GetYaxis()->SetTitle("Entries");

    TCanvas *c1 = new TCanvas();
    hist->Draw();
}
```

**Key Points:**

- `Gaus(mean, sigma)` generates Gaussian-distributed random numbers
- `ios::out` opens file for writing
- Same file can be reopened for reading

# Tutorial 7: Fitting Data with Functions

**Concepts:** Function fitting, parameter extraction, fit range specification

```
void tut7()
{
    // [Same data generation as tut6]
    TH1F *hist = new TH1F("hist","Title",100,0,10);

    TRandom2 *rand = new TRandom2(0);
    fstream file;
    file.open("data3.txt",ios::out);
    for(int i=0; i<10000; i++)
    {
        double r = rand->Gaus(5,1);
        file << r << endl;
    }
    file.close();

    file.open("data3.txt", ios::in);
    double value;
    while(1){
        file >> value;
        hist->Fill(value);
        if(file.eof()) break;
    }
    file.close();

    hist->GetXaxis()->SetTitle("Gaussian");
    hist->GetYaxis()->SetTitle("Entries");

    // Create fit function
    // TF1(name, function, xmin, xmax)
    TF1 *fit = new TF1("fit", "gaus", 4, 6);

    // Fit histogram ("R" = use function range)
    hist->Fit("fit","R");

    // Set initial parameters (optional)
    fit->SetParameters(100,5,1);

    // Extract fit parameters
    double mean = fit->GetParameter(1);   // Parameter 1 = mean
    double sigma = fit->GetParameter(2);  // Parameter 2 = sigma

    cout << mean/sigma << endl;
```

```
    TCanvas *c1 = new TCanvas();
    hist->Draw();
}
```

**Key Points:**

- `TF1` creates a 1D function (built-in: "gaus", "pol1", "expo", etc.)
- `Fit(function_name, options)` performs the fit
- Fit option "R" restricts fit to the function's defined range
- Gaussian parameters: [0]=amplitude, [1]=mean, [2]=sigma
- `GetParameter(n)` extracts fitted parameters

## Summary of Key ROOT Classes

| Class | Purpose | Common Methods |
|---|---|---|
| `TH1F` | 1D histogram | `Fill()`, `Draw()`, `GetXaxis()` |
| `TGraph` | x-y scatter plot | `SetPoint()`, `Draw()`, `SetMarkerStyle()` |
| `TCanvas` | Drawing surface | Constructor creates display |
| `TRandom2` | Random numbers | `Rndm()`, `Gaus()` |
| `TF1` | 1D function | `SetParameters()`, `GetParameter()` |

## Common Drawing Options

- **Histograms**: Default, "E" (errors), "HIST" (no errors)
- **Graphs**: "A" (axes), "P" (points), "L" (line), "C" (curve)