HANDS-ON SESSION #2b

Ratio plots capabilities in PyROOT within a Jupyter Notebook (part-b)

Colab file: **`E+_2b_TRatioPlot_errors.ipynb`**

Erasmus+ EU programme

Alexis Pompili (*)

University of Bari Aldo Moro

Hands-on exercises will be carried out within a `Jupyter` Notebook executed in the `Google Colab` framework.

(*) I would like to thank my collaborator **Umit Sozbilir**

Manipulating histograms or arrays of numbers : the `TRatioPlot` class - I

An alternative to the two ways to obtain a ratio of histograms reviewed in exercise 2 part-a (2a) …
… consists in using the `TRatioPlot` class (*).

It can be used **setting two main options**:

- **`div`** (default) — uses **`TGraphAsymmErrors`** class (and its **`Divide`** method) to calculate the ratio which handles **asymmetric uncertanties**
- **`divsym`** — forces the use of the **`Divide`** method of the histogram **TH1** class (see part-2a), resulting in **symmetric uncertainties** on the ratio

For the scope of the course we will show how this approach can be used with the **`divsym`** option providing the same results obtained in the manual calculation!

(*) https://root.cern.ch/doc/master/classTRatioPlot.html

Manipulating histograms or arrays of numbers : the `TRatioPlot` class - II

The `TRatioPlot` class in `ROOT`, by default, uses the `TGraphAsymmErrors::Divide` method to compute the ratio of two histograms. When the **divsym** option is specified, it uses the `TH1::Divide` method, which assumes symmetric errors for the ratio calculation.

In this context we will review the use of `TRatioPlot` with the **divsym** option, and the uncertaintiews will be symmetric.

When plotting with **TRatioPlot** in ROOT, based on the example provided in the official ROOT documentation (https://root.cern.ch/doc/master/classTRatioPlot.html), by using the **divsym** option, the method `sumw2` should be applied to properly account for the errors, even though the official documentation does not explicitly mention it. Using `sumw2` (for one of the histograms) with the **divsym** option ensures accurate error propagation, which would otherwise be incorrect without it, as we have seen in the previous exercise (2a) using directly the **Divide** method. **No cloning** of histograms is necessary in this case.

We will verify that with `sumw2` applied, the uncertainty results are in perfect (!) agreement with the manual calculations; conversely, if `sumw2` is not used, the error values will differ significantly from the expected ones.

INSTALLATION

This is just a reminder about the installations steps (we are now familiar with them):

```
!pip install -q condacolab
import condacolab
condacolab.install()
```

```
!conda create -n SDAL python=3.10 -y
!source /usr/local/etc/profile.d/conda.sh && conda activate SDAL

!conda install -c conda-forge root=6.28.10 -y

import os
os.environ["PATH"] = "/usr/local/envs/SDAL/bin:" + os.environ["PATH"]
os.environ["PYTHONPATH"] = "/usr/local/envs/SDAL/lib/python3.10/site-packages"
```

```
import ROOT
import math
print(ROOT.gROOT.GetVersion())
```

```
Welcome to JupyROOT 6.28/10
6.28/10
```

Manual calculation of the uncertainty on the ratio - I

We use more or less the code already implemented in the part-2a of the hands-on session #2 :

```python
def manual_calculation(numerator, denumerator):
    if numerator == 0 or denumerator == 0:
        return 0, 0
    ratio_manual = numerator / denumerator
    sigma_numerator_manual = math.sqrt(numerator)  # Poisson error approximation: sqrt(A)
    sigma_denumerator_manual = math.sqrt(denumerator)  # Poisson error approximation: sqrt(B)

    # Gaussian error propagation formula
    sigma_R_manual = ratio_manual * math.sqrt((sigma_numerator_manual / numerator) ** 2 + (sigma_denumerator_manual / denumerator) ** 2)

    return ratio_manual, sigma_R_manual, sigma_numerator_manual, sigma_denumerator_manual
```

```python
numerator_manual = [85, 91, 79, 89, 93, 77, 85, 81, 87, 90]
denumerator_manual = [80, 90, 85, 95, 82, 87, 79, 86, 93, 91]
num_bins_manual = len(numerator_manual)

# Initialize empty lists to store manual results
ratios_manual = []
errors_manual = []
sigma_numerator_manual = []
sigma_denumerator_manual = []

# For each bin, calculate ratio and error
for i in range(num_bins_manual):
    A = numerator_manual[i]
    B = denumerator_manual[i]

    # Call the manual_calculation function
    ratio_manual, error_manual, sigma_numerator, sigma_denumerator = manual_calculation(A, B)

    # Store the results
    ratios_manual.append(ratio_manual)
    errors_manual.append(error_manual)
    sigma_numerator_manual.append(sigma_numerator)
    sigma_denumerator_manual.append(sigma_denumerator)
```

our data

$$\sigma_R = R\sqrt{\left(\frac{\sigma_N}{N}\right)^2 + \left(\frac{\sigma_D}{D}\right)^2}$$

where $R = \frac{N}{D}$

Alexis Pompili 26.4

Manual calculation of the uncertainty on the ratio - II

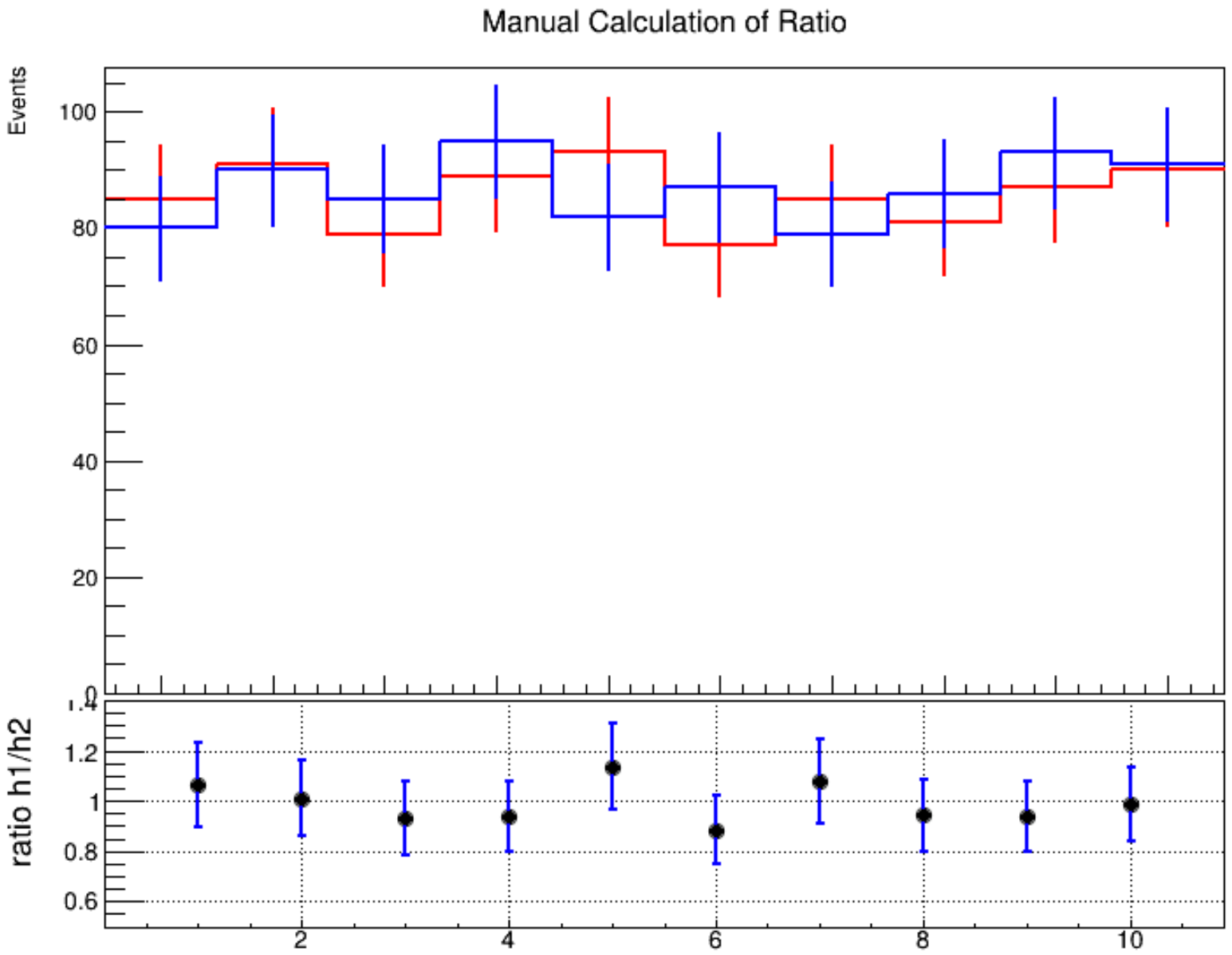
We can also create the visualization approach used in the part-a of the hands-on session #2 :

```
#Just to create similar canvas as in Divide() method as in exercise 2a:

canvas_manual = ROOT.TCanvas("canvas_manual", "Manual Calculation of Ratio", 800, 600)

# Upper pad for the numerator and denominator histograms
pad1_manual = ROOT.TPad("pad1_manual", "pad1_manual", 0, 0.3, 1, 1)
pad1_manual.SetBottomMargin(0.01)
pad1_manual.Draw()
pad1_manual.cd()

hist_numerator_manual = ROOT.TH1F("hist_numerator_manual", "Numerator; Bin Number; Events", num_bins_manual, 0.5, num_bins_manual + 0.5)
hist_denumerator_manual = ROOT.TH1F("hist_denumerator_manual", "Denominator; Bin Number; Events", num_bins_manual, 0.5, num_bins_manual + 0.5)
hist_numerator_manual.SetMinimum(0)
hist_denumerator_manual.SetMinimum(0)
hist_numerator_manual.SetStats(0)
hist_numerator_manual.GetXaxis().SetLabelSize(0.0)

for i in range(num_bins_manual):
    hist_numerator_manual.SetBinContent(i + 1, numerator_manual[i])
    hist_denumerator_manual.SetBinContent(i + 1, denumerator_manual[i])

hist_numerator_manual.SetLineColor(ROOT.kRed)
hist_numerator_manual.SetLineWidth(2)
hist_numerator_manual.SetTitle("Manual Calculation of Ratio")

hist_denumerator_manual.SetLineColor(ROOT.kBlue)
hist_denumerator_manual.SetLineWidth(2)

hist_numerator_manual.Draw("EHIST")
hist_denumerator_manual.Draw("EHIST SAME")

canvas_manual.cd()
pad2_manual = ROOT.TPad("pad2_manual", "pad2_manual", 0, 0.05, 1, 0.3)
pad2_manual.SetTopMargin(0)
pad2_manual.SetBottomMargin(0.1)
pad2_manual.SetGridx()
pad2_manual.SetGridy()
pad2_manual.Draw()
pad2_manual.cd()
```

Manual calculation of the uncertainty on the ratio - III

```
graph_manual = ROOT.TGraphErrors(num_bins_manual)

for i in range(num_bins_manual):
    graph_manual.SetPoint(i, i + 1, ratios_manual[i])  # X: bin number, Y: ratio
    graph_manual.SetPointError(i, 0, errors_manual[i])  # Y error: computed error

graph_manual.SetTitle("")
graph_manual.GetXaxis().SetTitle("Bin Number")
graph_manual.GetYaxis().SetTitle("Ratio (Manual)")

graph_manual.SetLineColor(ROOT.kBlue)
graph_manual.SetLineWidth(2)

y_manual = graph_manual.GetYaxis()
y_manual.SetTitle("ratio h1/h2 ")
y_manual.SetNdivisions(505)
y_manual.SetTitleSize(20)
y_manual.SetTitleFont(43)
y_manual.SetTitleOffset(1.55)
y_manual.SetLabelFont(43)
y_manual.SetLabelSize(15)

# Adjust x-axis settings
x_manual = graph_manual.GetXaxis()

x_manual.SetTitleSize(20)
x_manual.SetTitleFont(43)
x_manual.SetTitleOffset(4.0)
x_manual.SetLabelFont(43)
x_manual.SetLabelSize(15)

graph_manual.SetMarkerStyle(20)
graph_manual.SetMarkerSize(1.1)
graph_manual.SetMaximum(1.4)
graph_manual.SetMinimum(0.5)
graph_manual.Draw("AP")

canvas_manual.Update()
canvas_manual.Draw()

canvas_manual.SaveAs("manual_calculation_ratio_with_histograms.png")
```

Manual calculation of the uncertainty on the ratio - IV



$$\frac{NUM}{DENOM}$$

Manual calculation of the uncertainty on the ratio - V

We can inspect the uncertainties values numerically building a table:

```
print("Manual calculation method\n")
print(f"{'Bin':^5} | {'Ratio':^7} | {'± Error':^7} |")
print("-" * 30)

for i in range(num_bins_manual):
    print(f"{i+1:^5} | {ratios_manual[i]:^7.4} | {errors_manual[i]:^7.4} |")
```

Manual calculation method

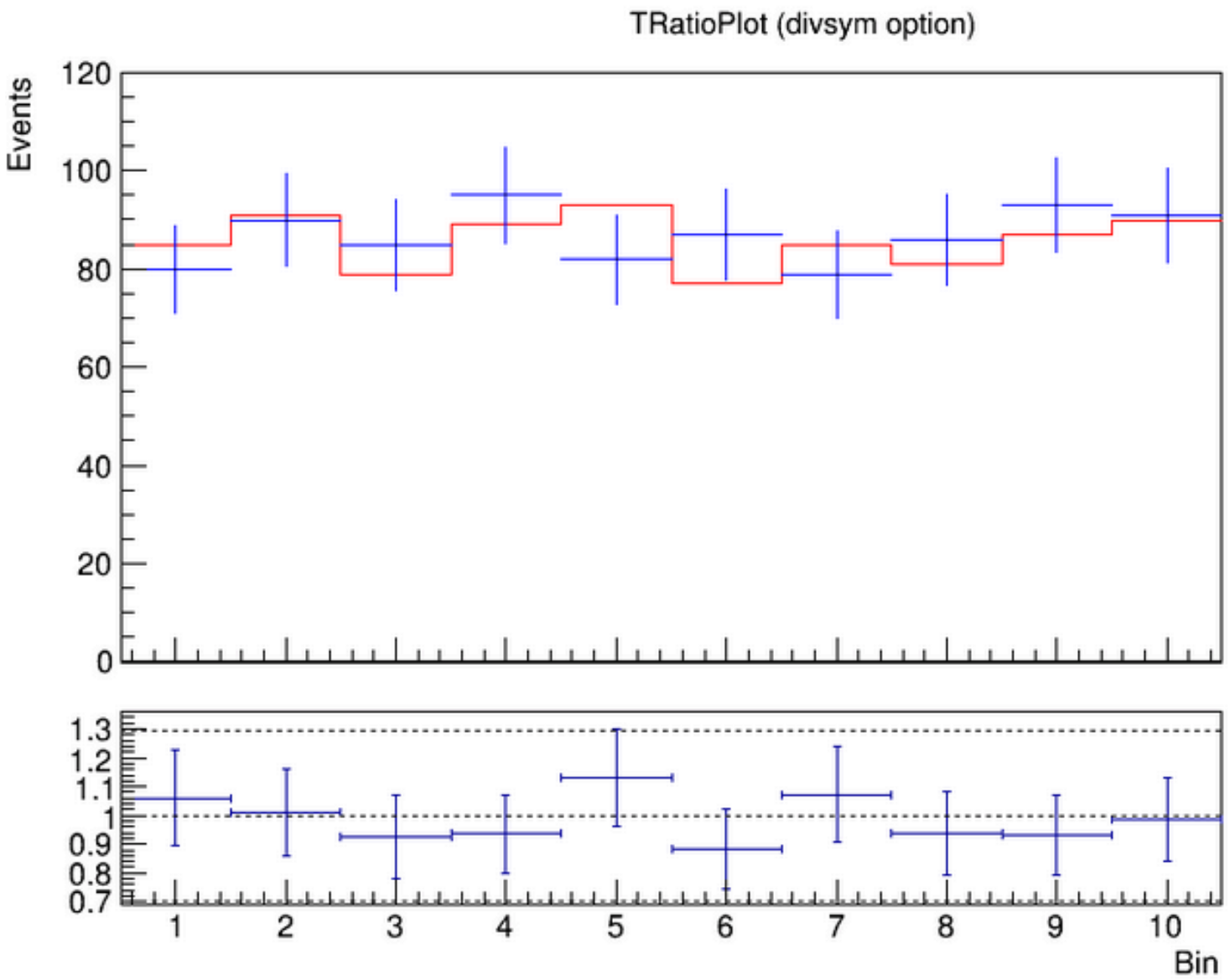
Bin	Ratio	± Error
1	1.062	0.1655
2	1.011	0.1503
3	0.9294	0.1452
4	0.9368	0.1382
5	1.134	0.1718
6	0.8851	0.1385
7	1.076	0.1681
8	0.9419	0.1458
9	0.9355	0.1395
10	0.989	0.147

Ratio with TPlotRatio class and divsym option - I

We now calculate the ratio and its uncertainty (treated as symmetric) using the **TRatioPlot** class and its **dyvsim** option:

```
numerator_tratio_sym = [85, 91, 79, 89, 93, 77, 85, 81, 87, 90]
denumerator_tratio_sym = [80, 90, 85, 95, 82, 87, 79, 86, 93, 91]

num_bins_tratio_sym = len(numerator_tratio_sym)

# Create histograms for numerator and denominator
hist_numerator_tratio_sym = ROOT.TH1F("hist_numerator_tratio_sym", "Numerator; Bin Number; Events", num_bins_tratio_sym, 0.5, num_bins_tratio_sym + 0.5)
hist_denumerator_tratio_sym = ROOT.TH1F("hist_denumerator_tratio_sym", "Denominator; Bin Number; Events", num_bins_tratio_sym, 0.5, num_bins_tratio_sym + 0.5)

hist_numerator_tratio_sym.SetMinimum(0)
hist_numerator_tratio_sym.SetMaximum(120)
hist_numerator_tratio_sym.SetStats(0)
hist_denumerator_tratio_sym.SetMinimum(0)
hist_denumerator_tratio_sym.SetMaximum(120)

for i in range(num_bins_tratio_sym):
    hist_numerator_tratio_sym.SetBinContent(i + 1, numerator_tratio_sym[i])
    hist_denumerator_tratio_sym.SetBinContent(i + 1, denumerator_tratio_sym[i])

hist_numerator_tratio_sym.Sumw2()   # crucial to turn on (at least for one of the two)
# hist_denumerator_tratio_sym.Sumw2()

# Create the TRatioPlot using the Divide method
canvas_tratio_sym = ROOT.TCanvas("canvas_tratio_sym", "TRatioPlot (divsym option)", 800, 600)
ratio_plot_tratio_sym = ROOT.TRatioPlot(hist_numerator_tratio_sym, hist_denumerator_tratio_sym, "divsym")  # div_sym option

ratio_plot_tratio_sym.Draw()
canvas_tratio_sym.Update()

hist_numerator_tratio_sym.SetTitle(f"TRatioPlot (divsym option);Bin;Events")
hist_numerator_tratio_sym.SetLineColor(ROOT.kRed)
hist_denumerator_tratio_sym.SetLineColor(ROOT.kBlue)
canvas_tratio_sym.cd()

ratio_plot_tratio_sym.GetLowerPad().cd()
ratio_plot_tratio_sym.GetLowerPad().Update()
tratio_divsym_low = ratio_plot_tratio_sym.GetLowerRefGraph()

canvas_tratio_sym.Draw()
canvas_tratio_sym.SaveAs("TRatioPlot_DivSym_Method_Manual_vs_Div.png")
```

Ratio with `TPlotRatio` class and `divsym` option - II



Ratio with TPlotRatio class and divsym option - III

Check:

```
if tratio_divsym_low.InheritsFrom("TGraphAsymmErrors"):
    print("The method used is 'div'.")
else:
    print("The method used is 'divsym'.")
```

```
The method used is 'divsym'.
```

Let's accomodate the values of ratio and its uncertainty in a table:

```
ratios_tratio_sym = [tratio_divsym_low.GetY()[i] for i in range(num_bins_tratio_sym)]

errors_tratio_sym = [tratio_divsym_low.GetEYlow()[i] if tratio_divsym_low.InheritsFrom("TGraphAsymmErrors") else tratio_divsym_low.GetEY()[i] for i in range(num_bins_tratio_sym)]

print("TRatio DivSym method\n")

print(f"{'Bin':^5} | {'Ratio':^7} |{'± Error':^7} | ")
print("-" * 27)

for i in range(num_bins_tratio_sym):
    print(f"{i + 1:^5} | {ratios_tratio_sym[i]:^7.4f} |±{errors_tratio_sym[i]:^6.4f} |")
```

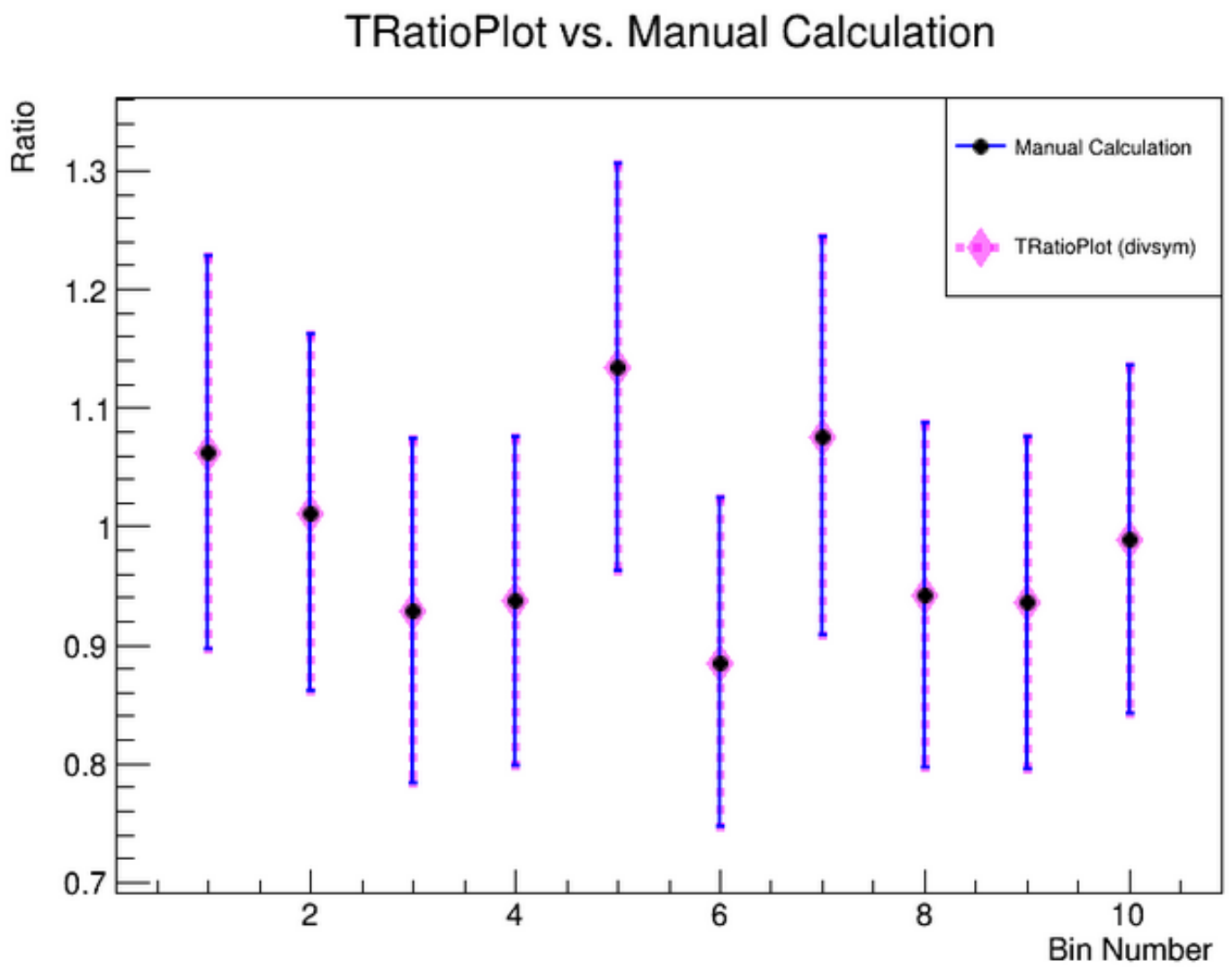
```
TRatio DivSym method

 Bin  |  Ratio  |± Error |
---------------------------
  1   | 1.0625  |±0.1655 |
  2   | 1.0111  |±0.1503 |
  3   | 0.9294  |±0.1452 |
  4   | 0.9368  |±0.1382 |
  5   | 1.1341  |±0.1718 |
  6   | 0.8851  |±0.1385 |
  7   | 1.0759  |±0.1681 |
  8   | 0.9419  |±0.1458 |
  9   | 0.9355  |±0.1395 |
 10   | 0.9890  |±0.1470 |
```

Comparison between TRatioPlot (divsym) and manual calculation - I

Let's verify that the result obtained by the `TRatioPlot` class (with `divsym` option) is exactly what calculated manually:

```
canvas_all_ratios = ROOT.TCanvas("canvas_all_ratios", "Manual vs TRatio (div) vs TRatio (divsym)", 800, 600)

graphic_tratio_sym = ROOT.TGraphErrors(num_bins_tratio_sym)

for i in range(num_bins_tratio_sym):
    graphic_tratio_sym.SetPoint(i, i + 1, ratios_tratio_sym[i])  # X: bin number, Y: ratio
    graphic_tratio_sym.SetPointError(i, 0, errors_tratio_sym[i])  # Same error for low and up

# Set properties for TRatio plot :
graphic_tratio_sym.SetTitle("TRatioPlot vs. Manual Calculation")
graphic_tratio_sym.GetXaxis().SetTitle("Bin Number")
graphic_tratio_sym.GetYaxis().SetTitle("Ratio")
graphic_tratio_sym.SetMarkerColorAlpha(ROOT.kMagenta, 0.5)  # 50% transparency
graphic_tratio_sym.SetLineColorAlpha(ROOT.kMagenta, 0.5)  # 50% transparency
graphic_tratio_sym.SetMarkerStyle(33)
graphic_tratio_sym.SetMarkerSize(3)
graphic_tratio_sym.SetLineWidth(5)
graphic_tratio_sym.SetLineStyle(7)

graphic_tratio_sym.Draw("AP")  # Draw TRatio plot
#
# Superimpose the manual graph on the same canvas
graph_manual.Draw("P SAME")

legend = ROOT.TLegend(0.7, 0.7, 0.9, 0.9)
legend.AddEntry(graph_manual, "Manual Calculation", "pl")
legend.AddEntry(graphic_tratio_sym, "TRatioPlot (divsym)", "pl")
legend.Draw()

canvas_all_ratios.Update()
canvas_all_ratios.Draw()
canvas_all_ratios.SaveAs("manual_vs_tratio-divsym.png")
```

Comparison between `TRatioPlot` (`divsym`) and manual calculation - II



As expected the two approaches provide exactly the same results!

ADDITIONAL MATERIAL

Comparison between manual calculation and the two `TRatioPlot` approaches - I

Find here a full comparison between the two approaches that differ.
Of course the manual calculation provides a result that is the same as that obtained by the divsym option!

TRatio Div (default) method

Bin	Ratio	+ Error	- Error
1	1.0625	+0.1941	-0.1641
2	1.0111	+0.1749	-0.1491
3	0.9294	+0.1703	-0.1440
4	0.9368	+0.1606	-0.1372
5	1.1341	+0.2006	-0.1704
6	0.8851	+0.1624	-0.1373
7	1.0759	+0.1973	-0.1667
8	0.9419	+0.1708	-0.1446
9	0.9355	+0.1624	-0.1385
10	0.9890	+0.1711	-0.1459

Manual calculation method

Bin	Ratio	± Error
1	1.062	0.1655
2	1.011	0.1503
3	0.9294	0.1452
4	0.9368	0.1382
5	1.134	0.1718
6	0.8851	0.1385
7	1.076	0.1681
8	0.9419	0.1458
9	0.9355	0.1395
10	0.989	0.147

TRatio DivSym method

Bin	Ratio	± Error
1	1.0625	±0.1655
2	1.0111	±0.1503
3	0.9294	±0.1452
4	0.9368	±0.1382
5	1.1341	±0.1718
6	0.8851	±0.1385
7	1.0759	±0.1681
8	0.9419	±0.1458
9	0.9355	±0.1395
10	0.9890	±0.1470

Of course the manual calculation provides a result that is the same as that obtained by the divsym option!

Comparison between manual calculation and the two `TRatioPlot` approaches - I

Find here a full comparison between the two approaches that differ.
Of course the manual calculation provides a result that is the same as that obtained by the divsym option!