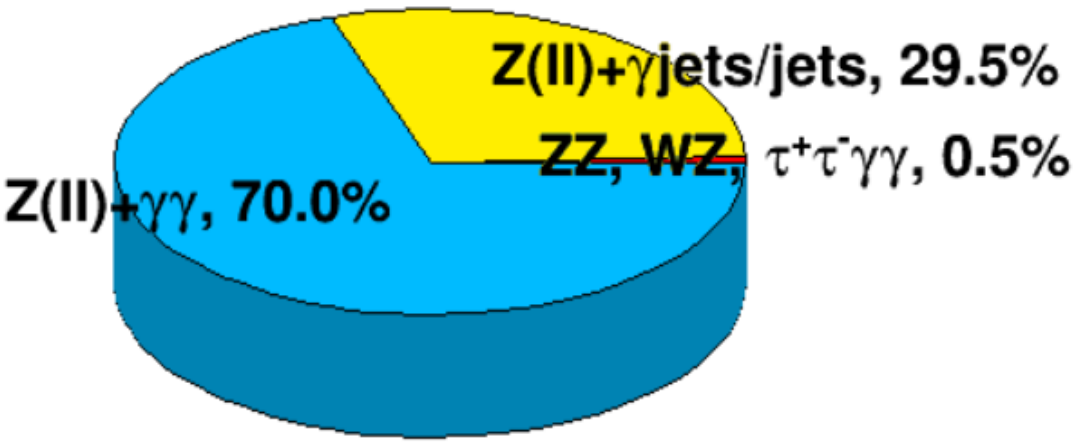
Круговая диаграмма

```
{
    char tmp[128];
  Float_t vals[] = {0.5,29.5,70};
  Int_t colors[] = {2,90,65};
  Int_t nvals = sizeof(vals)/sizeof(vals[0]);
  TCanvas *cpie = new TCanvas("cpie","TPie test",700,700);

  TPie *pie3 = new TPie("pie3",
    "Pie with tangential labels",nvals,vals,colors);

  pie3->GetSlice(0)->SetLineColor(1);
  pie3->GetSlice(0)->SetTitle("");
  pie3->GetSlice(1)->SetTitle("");
  pie3->GetSlice(2)->SetTitle("");
  pie3->SetY(.32);
  pie3->SetLabelsOffset(0.01);
      pie3->SetCircle(.5,.5,.2);
  pie3->Draw("3d rsc");

     sprintf(tmp,"Z(ll)+#gamma#gamma, 70.0%%");
   TLatex *t1y =new TLatex(0.23,0.46,tmp);
   t1y->SetNDC();t1y->SetTextSize(0.04);
   t1y->Draw();

     sprintf(tmp,"ZZ, WZ, #tau^{+}#tau^{-}#gamma#gamma, 0.5%%");
   TLatex *t1y =new TLatex(0.57,0.49,tmp);
   t1y->SetNDC();t1y->SetTextSize(0.04);
   t1y->Draw();

     sprintf(tmp,"Z(ll)+#gammajets/jets, 29.5%%");
   TLatex *t1y =new TLatex(0.54,0.55,tmp);
   t1y->SetNDC();t1y->SetTextSize(0.04);
   t1y->Draw();
}
```



Считывание из текстового файла в root гистограмму

```
{

TH1D *hist=new TH1D("hist","det1",211,0.5,210.5);
char ptr[20];
int ch_num;
double ch_value;

  FILE *f = fopen("Kamera_2.txt","r");

   if (f != NULL)
   {
       while (fgets(ptr, 20, f)!= NULL){
        sscanf(ptr, "%i%lf\n", &ch_num,&ch_value);

            cout <<ch_num<<" "<<ch_value<<endl;
            hist->SetBinContent(ch_num,ch_value);}
   }
   else {
      cout<<"Error file is not found";
   }

  TCanvas *c1 = new TCanvas ("c1", "hist", 1);
  hist->Draw("");

}
```

Запись из гистограмм в текстовый файл

```
{

        TFile *file_CO2=new TFile("qsv_CO2.root");
        TFile *file_nPent=new TFile("qsv_nPent.root");

  TH2D *current_map_CO2_1;// =  new TH1F("reco10_phot_et5","reco10 phot
et5",15,0.,75.);
      file_CO2->GetObject("IvsXY1",current_map_CO2_1);
  TH2D *current_map_nPent_1;// =  new TH1F("reco10_phot_et5","reco10 phot
et5",15,0.,75.);
      file_nPent->GetObject("IvsXY1",current_map_nPent_1);
  TH2D *current_map_CO2_2;// =  new TH1F("reco10_phot_et5","reco10 phot
et5",15,0.,75.);
      file_CO2->GetObject("IvsXY2",current_map_CO2_2);
  TH2D *current_map_nPent_2;// =  new TH1F("reco10_phot_et5","reco10 phot
et5",15,0.,75.);
      file_nPent->GetObject("IvsXY2",current_map_nPent_2);

for(int i=1;i<=39;i++)
{
        for (int j=1; j<=27;j++)
        {
                current_map_1->SetBinContent(i,j,current_map_CO2_1-
>GetBinContent(i,j)/current_map_nPent_1->GetBinContent(i,j));
                current_map_2->SetBinContent(i,j,current_map_CO2_2-
>GetBinContent(i,j)/current_map_nPent_2->GetBinContent(i,j));
        }
}
…
```

```
…

  FILE* file_log;

file_log = fopen("avarage_nPent.dat","w+");

for(int j=1; j<=27; j++)
{
for(int i=1; i<=39; i++)
{
fprintf(file_log,
                              "%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n",
                              current_map_CO2_1->GetXaxis()-
>GetBinLowEdge(i)+(current_map_CO2_1->GetXaxis()->GetBinLowEdge(2)-
current_map_CO2_1->GetXaxis()->GetBinLowEdge(1))/2,
                              current_map_CO2_1->GetYaxis()-
>GetBinLowEdge(j)+(current_map_CO2_1->GetYaxis()->GetBinLowEdge(2)-
current_map_CO2_1->GetYaxis()->GetBinLowEdge(1))/2,
                              0,
                              current_map_nPent_1->GetBinContent(i,j),
                              current_map_nPent_2->GetBinContent(i,j),
                              0,
                              0,
                              0,
                              0);
}
}
 fclose(file_log);
```

Перерисовка оси

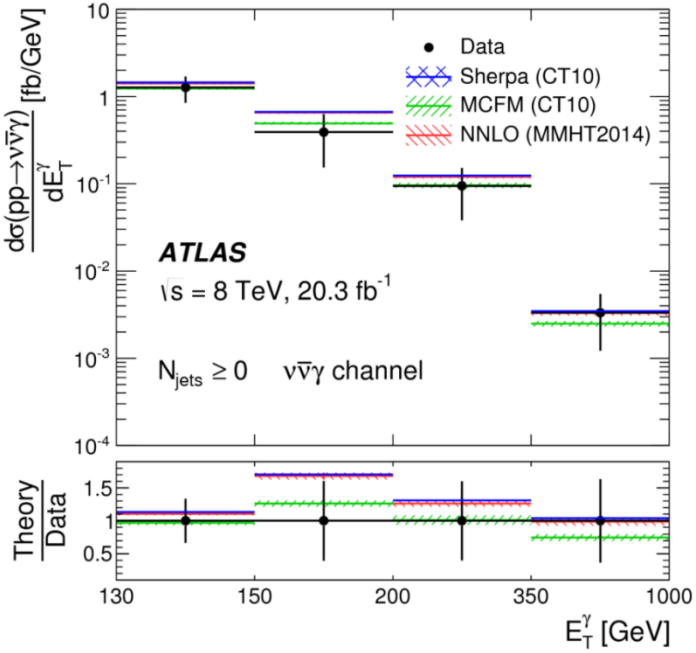
```
{

  // Redraw the new axis

  TCanvas *c1 = new TCanvas ("c1", "hist", 1);

  hist->Draw("colz");

  gPad->Update();
  TGaxis *newaxis = new TGaxis(gPad->GetUxmin(),
                               gPad->GetUymax(),
                               gPad->GetUxmin()-0.001,
                               gPad->GetUymin(),
                               hist->GetYaxis()->GetXmin(),
                               hist->GetYaxis()->GetXmax(),
                               510,"+");
  newaxis->SetLabelOffset(-0.03);
  newaxis->Draw();

}
```

Перерисовка числовых подписей к оси

```
string name_axis_pt_p[7] = {"150","200","250","350","450","600","1100"};

 string* name_axis = new string[7];
 name_axis = name_axis_pt_p;
 double x, y;
 y=0.72;

 TText t;
 t.SetTextSize(0.12);
 t.SetTextAlign(23);

 for (i=0;i<=6;i++) {
   if(i<6) x = hist_ph_et_data_valid->GetXaxis()->GetBinLowEdge(i+1);
   else if(i==6) x = hist_ph_et_data_valid->GetXaxis()->GetBinUpEdge(i);
   t.DrawText(x,y-0.5,name_axis[i].c_str());

 }
```

Рисование гистограммы с «подвалом»

```
TCanvas *c3 = new TCanvas ("c3", "ph_et in", 0.,0.,600,600);

pad_ph_et_21 = new TPad("pad_ph_et_21","This is pad_ph_et_1",0.01,0.31,1,1.0);
pad_ph_et_22 = new TPad("pad_ph_et_22","This is pad_ph_et_2",0.01,0.01,1,0.3);
      pad_ph_et_21->SetBorderSize(0);
      pad_ph_et_21->SetBottomMargin(0.02);
pad_ph_et_21->Draw();
       pad_ph_et_22->SetBorderSize(0);
       pad_ph_et_22->SetBottomMargin(0.40);
pad_ph_et_22->Draw();
pad_ph_et_21->cd(0);
pad_ph_et_21->SetLogy();

hist1->Draw("E");
hist1->Draw("E2same");
hist2->Draw("E2same");
hist2->Draw("same");

   gPad->RedrawAxis();
   c3->Update();

pad_ph_et_22->cd(0);

hist1_valid->Draw("E");      //divided hist1 to hist1 histogram
hist2_valid->Draw("E2same"); //divided hist2 to hist1 histogram
hist2_valid->Draw("Esame");  //divided hist2 to hist1 histogram
hist1_valid->Draw("Esame");  //divided hist1 to hist1 histogram
gPad->RedrawAxis();
```



Работа с массивом гистограмм

```
int file_handler(int, double, char*, TH1F*); // Constructor


void main_anal(){
        TH1F* hist_ph_et[100]; // Definition

        for (int i=0; i<70; i++)
        {
                sprintf(title_hist,"hist_ph_et_%i", i);
                hist_ph_et[i] = new
TH1F(title_hist,"ph_et",n_bin_ph_et,ph_et_matrix);
        }

        file_handler(5,1,“file.root", hist_ph_et[5]);

}


int file_handler(int num, double koef_sig, char *file_name, TH1F *hist_ph_et_MCsig)
{
...
}
```

Автоматическое создание скелета класса для анализа

• Предположим у вас есть файл с деревом данных (tree_data), которые необходимо анализировать.
Там может быть много ветвей.
• В ROOT есть возможность не писать самому скелет программы для анализа (подгружающий всё дерево).
Есть методы *MakeClass* и *MakeSelector*

• Открыв файл можно выполнить:

```
tree_data->MakeClass(“my_analysis");
```

или

```
tree_data->MakeSelector(“my_analysis");
```

• Создадутся заголовочный файл my_analysis.h и основной my_analysis.cxx
Чтение нужного вам дерева уже будет там прописано.
• Скелеты методов также уже будут созданы (инициализация, окончание, основное выполнение).

• Вам остаётся лишь написать код для анализа данных в соответствующем методе. Всё!