

# Importing data from CERN ROOT files to *Mathematica*

Ken Hsieh  
Technical Staff Member  
Wolfram Research Inc

February 10, 2010

This notebook demonstrates some examples of importing data from ROOT files.

```
In[1]:= Clear["Global`*"]; {$Version, Directory[], MaxMemoryUsed[]}  
$HistoryLength = 3;  
  
Out[1]= {8.0 for Mac OS X x86 (64-bit) (November 6, 2010), /Users/white, 19255912}  
  
In[3]:= $UserBaseDirectory  
  
Out[3]= /Users/white/Library/Mathematica
```

---

## ROOT Importer Usage

### ■ File MetaInformation (from TKey Objects)

The wrappers to access TKey functionalities have the form:

```
Import[ file, {"ROOT", "Keys"} ]
```

or simply as

```
Import[ file, "ROOT" ]
```

and the output has the form:

```
{ {name1, title1, class1}, {name2, title2, class2}, ... }
```

```
In[4]:= SetDirectory[NotebookDirectory[]];
```

If this is the very first time ROOT converter runs, this call may take some time to complete because the converter is generating some needed libraries. This library-generation happens only once and the libraries will persist in future sessions.

```
In[5]:= Import["cernstaff.root", {"ROOT", "Keys"}]  
  
Out[5]= {{T, CERN 1988 staff data, TTree}}  
  
In[6]:= Import["demo.root", "ROOT"]  
  
Out[6]= {{h0, histo nr:0, TH1F}, {h1, histo nr:1, TH1F}, {h2, histo nr:2, TH1F},  
         {h3, histo nr:3, TH1F}, {h4, histo nr:4, TH1F}, {h5, histo nr:5, TH1F},  
         {h6, histo nr:6, TH1F}, {h7, histo nr:7, TH1F}, {h8, histo nr:8, TH1F},  
         {h9, histo nr:9, TH1F}, {h10, histo nr:10, TH1F}, {h11, histo nr:11, TH1F},  
         {h12, histo nr:12, TH1F}, {h13, histo nr:13, TH1F}, {h14, histo nr:14, TH1F}}  
  
In[7]:= Import["th2f.root", "ROOT"]  
  
Out[7]= {{h2, xygaus + xygaus(5) + xylandau(10), TH2F}}
```

## ■ Interfaces to TTree Objects

### ■ Usage Information

The wrappers to access TTree functionalities have the forms:

retrieve metadata (branch names, data types, etc) with "TTreeMetadata":

```
Import[ file, {"ROOT", "TTreeMetadata", tree} ]
```

retrieve data from a TTree with "TTreeData":

```
Import[ file, {"ROOT", "TTreeData", tree},  
Import[ file, {"ROOT", "TTreeData", tree, branch} ],
```

use the option "Range" to do a partial import:

```
Import[ file, {"ROOT", "TTreeData", tree}, "Range" -> {beg, end} ],  
Import[ file, {"ROOT", "TTreeData", tree, branch}, "Range" -> {beg, end} ],
```

where:

*tree* is the name of the TTree object,

*branch* is the name of a TBranch object,

*beg* is the starting index (1-indexed) when partially importing a branch,

*end* is the ending index (1-indexed) when partially importing a branch.

Note that, since NTuple inherits from TTree, we can also pass  
the name of an NTuple object.

### ■ Importing the TKey information.

---

```
In[8]:= (* This imports the metadata of a given tree *)
Import["cernstaff.root", {"ROOT", "TTreeMetadata", "T"}]

Out[8]= {{Category, Category, Int_t, 3354}, {Flag, Flag, UInt_t, 3354},
{Age, Age, Int_t, 3354}, {Service, Service, Int_t, 3354},
{Children, Children, Int_t, 3354}, {Grade, Grade, Int_t, 3354},
{Step, Step, Int_t, 3354}, {Hrweek, Hrweek, Int_t, 3354}, {Cost, Cost, Int_t, 3354},
{Division, Division, Char_t, 3354}, {Nation, Nation, Char_t, 3354}}
```

```
In[9]:= (* import the branch information of a TTree *)
header = {"Name", "Title", "Data Type", "Entries"};
branchinfo = Import["cernstaff.root", {"ROOT", "TTreeMetadata", "T"}];
Grid[Join[{header}, branchinfo], Frame -> All]
```

Out[11]=

Name	Title	Data Type	Entries
Category	Category	Int_t	3354
Flag	Flag	UInt_t	3354
Age	Age	Int_t	3354
Service	Service	Int_t	3354
Children	Children	Int_t	3354
Grade	Grade	Int_t	3354
Step	Step	Int_t	3354
Hrweek	Hrweek	Int_t	3354
Cost	Cost	Int_t	3354
Division	Division	Char_t	3354
Nation	Nation	Char_t	3354

- Importing one particular branch.

---

```
In[12]:= (* This imports the Age branch and plots it as with Histogram[] *)
dat = Import["cernstaff.root", {"ROOT", "TTreeData", "T", "Age"}];
```

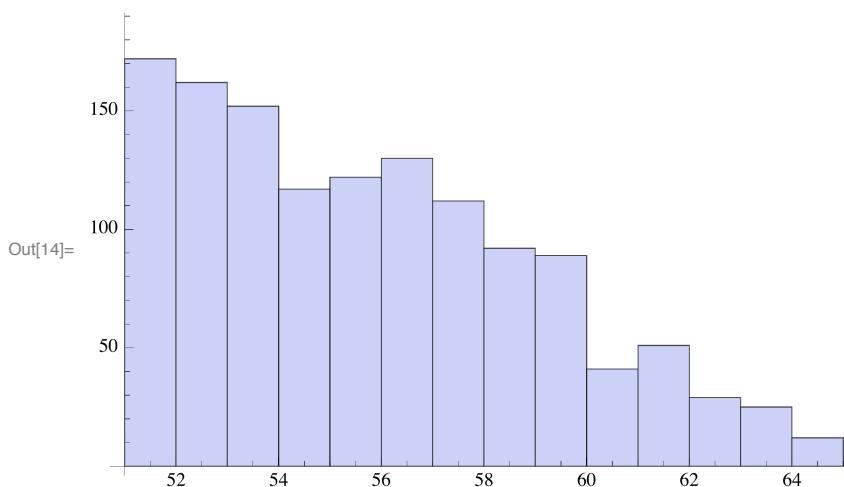
```
In[13]:= Length[dat]
```

```
Out[13]= 3354
```

- I added a couple cells below- SNW.

What may be hard to get used to in *Mathematica* is conditional histogramming. Here is an example using Select:

```
In[14]:= Histogram[Select[dat, # > 50 &]]
```



- Here is an example in free form English.

In[15]:=

**Histogram dat**

```

↳ WolframAlphaClient`Private`podTitle[{title → Histogram}, {1, 1, Input}]
WolframAlphaClient`Private`queryBlobMathematicaForm[
  Histogram[dat, Automatic]]

```

Input interpretation:

Histogram	dat
-----------	-----

```

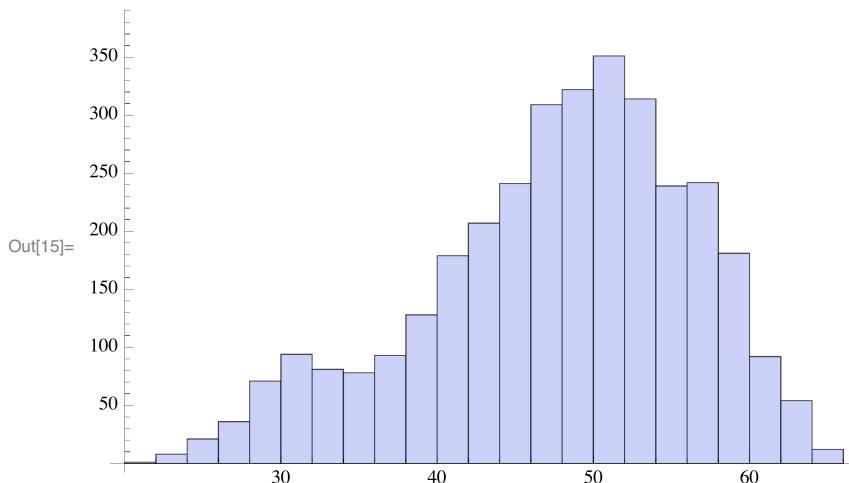
WolframAlphaClient`Private`FormatPod[XMLElement[pod,
  {title → Histogram, scanner → Plot, id → Histogram, position → 200, error → false, numsubpods → 1},
  {XMLElement[subpod, {title → }, {XMLElement[minput, {}, {Histogram[dat, Automatic]}]},
    XMLElement[cell, {compressed → False, string → True}, {Cell[BoxData[FormBox[
      RowBox[{StyleBox[Histogram, FontFamily → Bitstream Vera Sans, FontSize → -1 + Inherited],
        , RowBox[{dat, , StyleBox[Automatic, FontFamily → Bitstream Vera Sans, FontSize →
          -1 + Inherited]}, , RowBox[{"PerformanceGoal", →, "Speed"}], ,,
        RowBox[{StyleBox[ChartBaseStyle, FontFamily → Bitstream Vera Sans,
          FontSize → -1 + Inherited], →, RowBox[{StyleBox[EdgeForm,
            FontFamily → Bitstream Vera Sans, FontSize → -1 + Inherited], ,,
            RowBox[{StyleBox[RGBColor, FontFamily → Bitstream Vera Sans, FontSize →
              -1 + Inherited], [, RowBox[{0.59` , , 0.615` , , 0.71`}], , ]}], , ]]}],
        , RowBox[{StyleBox[ImageSize, FontFamily → Bitstream Vera Sans,
          FontSize → -1 + Inherited], →, 175}], , RowBox[{StyleBox[GridLinesStyle,
            FontFamily → Bitstream Vera Sans, FontSize → -1 + Inherited], →,
            RowBox[{StyleBox[Directive, FontFamily → Bitstream Vera Sans,
              FontSize → -1 + Inherited], [, RowBox[{RowBox[{RowBox[
                {StyleBox[AbsoluteThickness, FontFamily → Bitstream Vera Sans,
                  FontSize → -1 + Inherited], [, 1.01` , ]}], , RowBox[
                    {StyleBox[RGBColor, FontFamily → Bitstream Vera Sans,
                      FontSize → -1 + Inherited], [, RowBox[{1, , 0.7` , , 0.4`}], ,
                    ]}], , ]}], , ]}], , TraditionalForm]], Output]}]],
  XMLElement[dataformats, {}, {plaintext,minput}]]}], XMLElement[pod,
  {title → Histogram, scanner → Plot, id → Histogram,
  {XMLElement[subpod, {title → }, {XMLElement[minp
    XMLElement[cell, {compressed → False, string → True}, {Cell[BoxData[FormBox[
      RowBox[{StyleBox[Histogram, FontFamily → Bitstream Vera Sans, FontSize → -1 + Inherited],
        , RowBox[{dat, , StyleBox[Automatic, FontFamily → Bitstream Vera Sans, FontSize →
          -1 + Inherited]}, , RowBox[{StyleBox[ChartBaseStyle, FontFamily → Bitstream Vera Sans,
            FontSize → -1 + Inherited], , RowBox[{StyleBox[ImageSize, FontFamily → Bitstream Vera Sans,
              FontSize → -1 + Inherited], →, 175}], , RowBox[{StyleBox[GridLinesStyle,
                FontFamily → Bitstream Vera Sans, FontSize → -1 + Inherited], →,
                RowBox[{StyleBox[Directive, FontFamily → Bitstream Vera Sans,
                  FontSize → -1 + Inherited], [, RowBox[{RowBox[{RowBox[
                    {StyleBox[AbsoluteThickness, FontFamily → Bitstream Vera Sans,
                      FontSize → -1 + Inherited], [, 1.01` , ]}], , RowBox[
                        {StyleBox[RGBColor, FontFamily → Bitstream Vera Sans,
                          FontSize → -1 + Inherited], [, RowBox[{1, , 0.7` , , 0.4`}], ,
                        ]}], , ]}], , ]}], , TraditionalForm]], Output]}]],
    XMLElement[dataformats, {}, {plaintext,minput}]]}]]]

```

```

Bitstream Vera Sa
Directive, FontF
[, RowBox[{{
For
[, 1.0
For
[, Row
TraditionalForm]], Output]]], XMLLE
{Histogram, Histogram,
1,
1,
Input,
Histogram[dat,
Automatic]},
True, Histogram dat, Histogram dat,
{AppearanceElements → {Extrusion, Warnings, Assumptions, Pods}, Asynchronous → All,
Method → {ExtrusionChosen → {Histogram, Histogram, 1, 1, Input, Histogram[dat, Automatic]},
Formats → {cell, minput, moutput, msound, dataformats}, ExtrusionOpen → True}},
{TimeZone →
-6.,
Date →
{2011,
2,
24,
11,
58,
57.696006}, Line →
17, SessionID →
22 986 993 089 221 542 634}]

```



■ I thought this query yielded an interesting result:

```
In[16]:= WolframAlpha["CERN ROOT Data Analysis"]
```

Using closest Wolfram|Alpha interpretation: **Data Analysis** [?](#)

---

More interpretations: [Data Analysis](#)

Out[16]=

- Particle Physics: Additional functionality for this topic is under development...

[WolframAlpha](#) +

■ Importing only parts of one particular branch.

```
In[17]:= SetDirectory[NotebookDirectory[]];
dat = Import["cernstaff.root",
 {"ROOT", "TTreeData", "T", "Age"}, "Range" → {11, 20}]
Partial branch import with bounds: {11, 20}
Out[18]= {51, 54, 54, 46, 54, 57, 55, 55, 57, 51}
```

■ Importing all branches of a TTree.

```
In[19]:= (* This imports the metadata and the data a given NTuple *)
SetDirectory[NotebookDirectory[]];
Import["basic.root", {"ROOT", "TTreeMetadata", "ntuple"}]
Import["basic.root", {"ROOT", "TTreeData", "ntuple"}]
Out[20]= {{x, x, Float_t, 4}, {y, y, Float_t, 4}, {z, z, Float_t, 4}}
Out[21]= {{-1., 1., 2., 3.5}, {2., 5., 5., 3.4}, {3., 6., 45., 4.}}
```

■ Importing only certain parts of all branch of a TTree.

```
In[22]:= (* We need to use the "Range" option *)
SetDirectory[NotebookDirectory[]];
Import["basic.root", {"ROOT", "TTreeData", "ntuple"}, "Range" → {2, 4}]
Out[23]= {{1., 2., 3.5}, {5., 5., 3.4}, {6., 45., 4.}}
```

■ One more example

```
In[24]:= Quit
SetDirectory[NotebookDirectory[]];
```

```
(* import only the 11th through 25th entries across all branches *)
alldata =
  Import["cernstaff.root", {"ROOT", "TTreeData", "T"}, "Range" → {11, 25}];

(* present the data visually. *)
header = First /@ Import["cernstaff.root", {"ROOT", "TTreeMetadata", "T"}];
Grid[Join[{header}, Transpose[alldata]], Frame → All]
```

Category	Flag	Age	Service	Children	Grade	Step	Hrweek	Cost	Division	Nation
361	15	51	29	0	7	13	40	7599	PS	FR
303	15	54	31	2	8	13	40	9315	PS	CH
302	15	54	29	0	7	13	40	7599	PS	CH
300	15	46	25	0	8	6	40	7892	PS	CH
361	15	54	26	1	7	13	40	7850	PS	DE
361	15	57	29	0	7	13	40	7599	PS	FR
316	11	55	28	0	8	11	40	8137	PS	CH
303	15	55	26	1	7	13	40	7850	SPS	FR
361	15	57	29	1	7	8	40	7294	PS	FR
361	15	51	28	2	7	13	40	8101	PS	FR
419	13	54	29	0	5	13	40	5720	PS	FR
202	15	57	26	1	12	13	40	15 832	PS	DE
304	15	63	29	1	10	13	40	12 226	PS	NL
204	15	56	27	0	11	9	40	13 135	PS	DE
204	15	49	27	0	9	9	40	9617	LEP	GB

Note that the results of

```
Import[file, {"ROOT", "TreeData", tree}]
```

is of the form

```
{
  {px1, px2, px3, ...}, //branch px
  {py1, py2, py3, ...}, //branch py
  ...
}.
```

To get the first “event”, we would have to Transpose[] the result.

We can easily pick out subportions of the data using the Part[] command of *Mathematica*.

#### ■ Error handling

---

```
(* When the Tree is not present, we exit safely and return $Failed. *)
SetDirectory[NotebookDirectory[]];
Import["cernstaff.root", {"ROOT", "TTreeMetadata", "Invalid_Tree"}]
```

No such TTree exists

```
$Failed
```

---

```
(* When a branch is not present, we exit safely and return $Failed. *)
Import["cernstaff.root", {"ROOT", "TTreeData", "T", "InvalidBranch"}]
```

Allowed TBranches are:

```
{Category, Flag, Age, Service, Children, Grade, Step, Hrweek, Cost, Division, Nation}
and InvalidBranch is invalid.
```

```
$Failed
```

---

```
(* When the bounds are not appropriate, we exit safely and return $Failed. *)
Import["cernstaff.root",
 {"ROOT", "TTreeData", "T", "Age"}, "Range" -> {5, 50 000 000}]
```

Partial branch import with bounds: {5, 50 000 000}

Bound not accessible; the branch Age has only 3354 entires.

```
$Failed
```

## ■ Interface with TH1F Object

The wrappers to access TH1F functionalities have the forms:

```
Import[ file, {"ROOT", "TH1FData", hist} ],
Import[ file, {"ROOT", "TH1FGraphics", hist} ]
```

where:

*hist* is the name of the TH1F object.

```
SetDirectory[NotebookDirectory[]];
```

---

```
(* This imports the histogram data of a given TH1F object. *)
histdata = Import["demo.root", {"ROOT", "TH1FData", "h7"}];
```

The data is of the form:

```
{ {x1, Δx1, count1, error1}, {x2, Δx2, count2, error2}, ... }
```

```
(* show first 10 entries in a grid *)
head = {"x", "Δx", "count", "Δcount"};
Grid[Join[{head}, Take[histdata, 10]], Frame -> All]
```

x	Δx	count	Δcount
- 4.	0.08	0.	0.
- 3.92	0.08	0.	0.
- 3.84	0.08	0.	0.
- 3.76	0.08	0.	0.
- 3.68	0.08	0.	0.
- 3.6	0.08	0.	0.
- 3.52	0.08	0.	0.
- 3.44	0.08	0.	0.
- 3.36	0.08	1.	1.
- 3.28	0.08	0.	0.

---

```
(* Import the histogram directly as a Graphics *)
graphics1 = Import["demo.root", {"ROOT", "TH1FGraphics", "h10"}]
```

---

```
(* Options available to Histogram[] can be passed directly. *)
graphics2 = Import["demo.root", {"ROOT", "TH1FGraphics", "h7"},
  ColorFunction -> Function[{height}, ColorData["Rainbow"] [height]]]
```

## ■ Interface with TH2F Object

The wrappers to access TH1F functionalities have the forms:

```
Import[ file, {"ROOT", "TH2FData", hist} ],
Import[ file, {"ROOT", "TH2FGraphics", hist} ]
```

where:

*hist* is the name of the TH2F object.

```
SetDirectory[NotebookDirectory[]];
```

---

```
(* This imports the histogram data of a given TH2F object. *)
data = Import["th2f.root", {"ROOT", "TH2FData", "h2"}];
```

The data is of the form:

```
{ {x1, Δx1, count1, error1}, {x2, Δx2, count2, error2}, ... }
```

```
(* show first 5 entries in a grid *)
head = {"x", "Δx", "y", "Δy", "count", "Δcount"};
Grid[Join[{head}, Take[data, 5]], Frame -> All]
```

x	Δx	y	Δy	count	Δcount
-4.	0.4	-4.	0.4	1.	1.
-4.	0.4	-3.6	0.4	11.	3.31662
-4.	0.4	-3.2	0.4	16.	4.
-4.	0.4	-2.8	0.4	12.	3.4641
-4.	0.4	-2.4	0.4	9.	3.

---

```
(* Import the histogram directly as a Graphics *)
graphics3D = Import["th2f.root", {"ROOT", "TH2FGraphics", "h2"}]
```

---

```
(* Options available to Histogram3D[] can be passed directly. *)
graphics = Import["th2f.root", {"ROOT", "TH2FGraphics", "h2"},
  ChartElements -> Graphics3D@Cuboid[], ChartStyle -> "Pastel"]
```