`HWWAnalysisCode`

Tutorial Sessions

Session 1/2

Basic classes and data management

Andreas Walz
andreas.walz@cern.ch

Albert-Ludwigs-Universität Freiburg

2012-05-08

Where to find...

- The code and `ROOT` files used for this tutorial can be found in the `HWWAnalysisCode` repository (including the solutions to the exercises):

  `HWWAnalysisCode/trunk/tutorial`

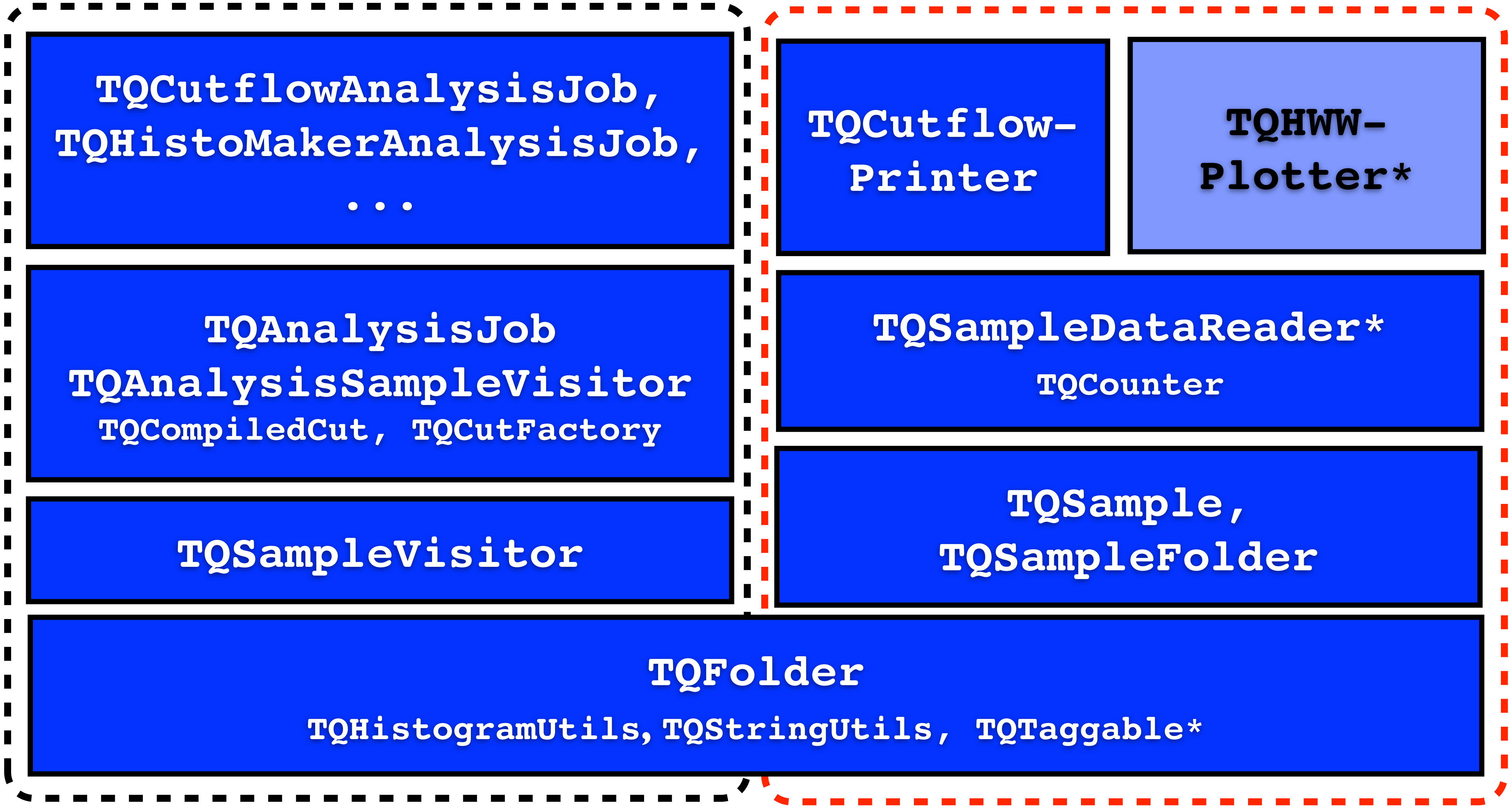
- A set of common ntuples (produced with `HWWNtupleCode-00-00-29`) to be used in some examples and exercises (in session 2) can be found at:

  `/afs/cern.ch/work/a/awalz/public/ntuples`

Structure of the `HWWAnalysisCode`

Running analysis (producing analysis results, histograms, ...)

Management and presentation of analysis results (histograms, ...)



`TQCutflowAnalysisJob`, `TQHistoMakerAnalysisJob`, ...

`TQAnalysisJob`
`TQAnalysisSampleVisitor`
`TQCompiledCut`, `TQCutFactory`

`TQSampleVisitor`

`TQCutflow-Printer`

`TQHWW-Plotter*`

`TQSampleDataReader*`
`TQCounter`

`TQSample`, `TQSampleFolder`

`TQFolder`
`TQHistogramUtils`, `TQStringUtils`, `TQTaggable*`

The TQFolder class

The TQFolder class is the basis for data management in the HWWAnalysisCode library. It is a container (similar to a directory in Linux) for ROOT objects (inheritors of TObject).

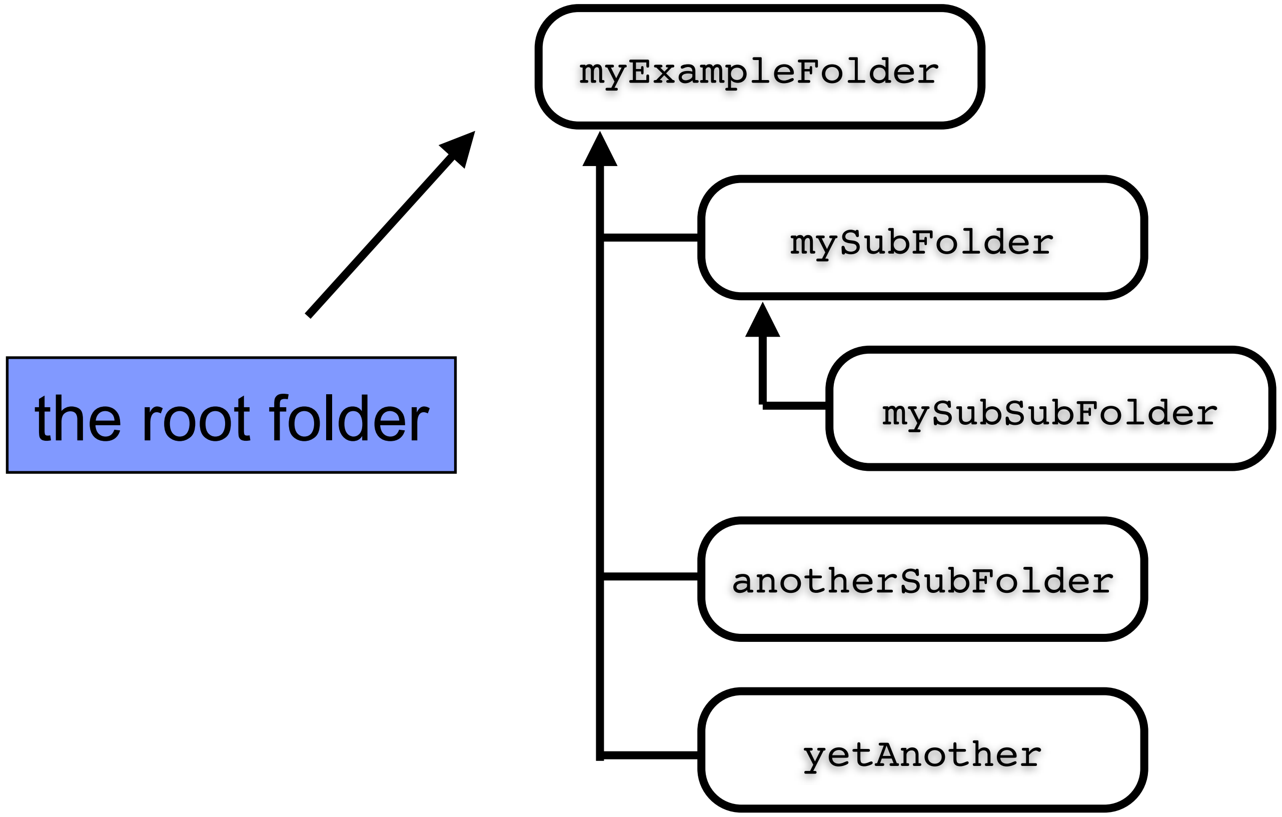
Some very useful/important functions:

- **folder->print();**
  - print the contents of the folder (print-out can be controlled by parameters and flags)
- **folder->getFolder(„myFolder“);**
  - return a pointer to a sub-folder

continued: The `TQFolder` class

The `TQFolder` class keeps a reference to its base folder (if existent) as well as to its sub-folders building a tree-like structure/hierarchy

- Writing the root folder to a `ROOT` file by using `folder->Write()` is sufficient to write the full hierarchy!



the root folder

myExampleFolder

mySubFolder

mySubSubFolder

anotherSubFolder

yetAnother

```
/.

/mySubFolder

/mySubFolder/mySubSubFolder

/anotherSubFolder

/yetAnother
```

Example 1: `TQFolder` class

// always start in the ROOT command line loading the library

```
root[0] .L ../lib/libQFramework.so
```

remember this line

// load the example folder from the external ROOT file

```
TQFolder * folder = TQFolder::loadFolder(
  „example1.root:myExampleFolder“);
```

// print the contents of the folder

```
folder->print();
```

// create a new sub-folder and print the contents again

```
folder->getFolder(„myNewFolder+“);
folder->print();
```

auto-create: creates the folder if it doesn‘t exist

// create nested folders

```
folder->getFolder(„newSub/newSubSub/foo+“);
folder->print(„r“);
```

Example 2: `TQFolder::print()`

The `print()` method of the `TQFolder` class accepts several options to control the print-out:

```
// add a „details“ column to show additional information
folder->print(„d“);
```

use „`rd`“ in this example to see the effect

```
// recursively print sub-folders (max. depth 2)
folder->print(„r2“);
```

can be any integer, no number means unlimitied depth

```
// use a filter (only show elements with matching name)
folder->print(„f[*LM*]“);
```

accepts wildcards „*“ and „?“, usage like in Linux command

```
// combine options
folder->print(„df[*ALL*]r“);
```

continued: `TQFolder::print()`

```
// show the number of sub-elements
folder->print(„c“);
```

```
// show the number of sub-elements (summed recursively)
folder->print(„C“);
```

capital „C“

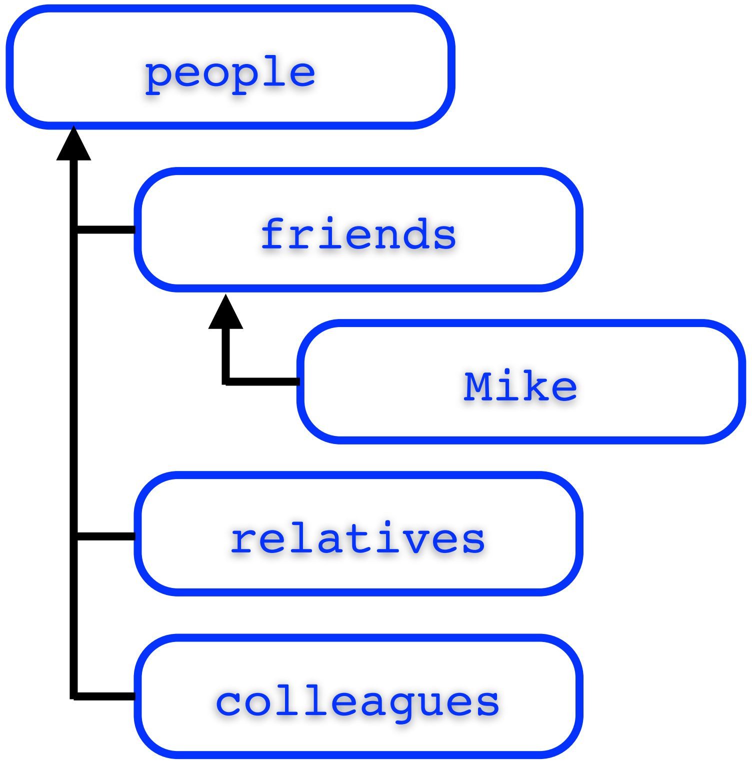
```
// print the contents of a sub-folder (here of „myNewFolder“)
folder->print(„myNewFolder:“);
```

```
// ...combined with some options
folder->print(„myNewFolder:dr2“);
```

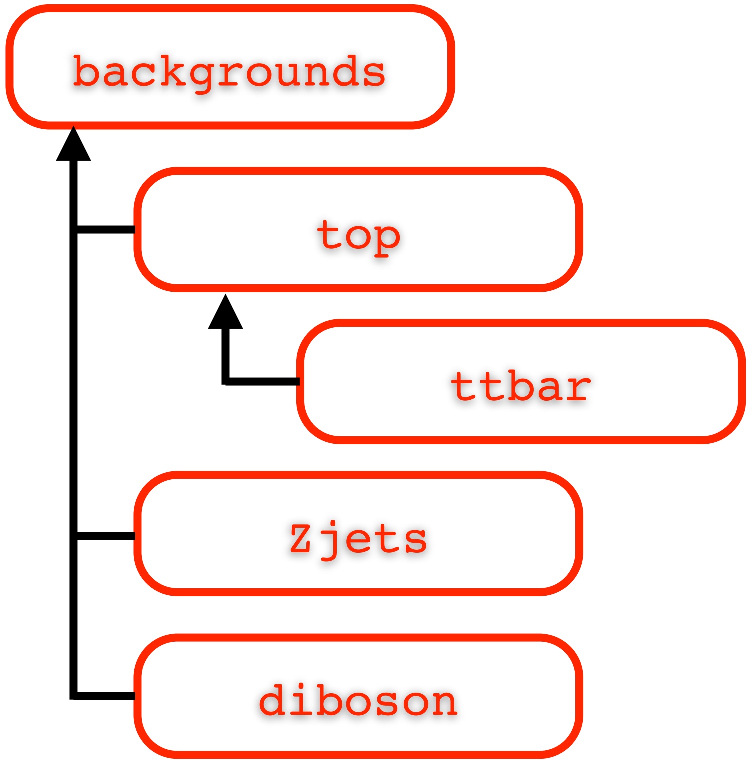
The `print()` method is the first tool to use if something does not work as it is expected !

The TQSampleFolder class

The TQFolder class is „only“ a container for objects derived from TObject and it does not make any assumptions about the deeper meaning of its contents.



The TQSampleFolder class is a specialization of the TQFolder class (inheriting all of its features) and it is designed to represent a certain category of event samples.



Example 3: `TQSampleFolder` class

There is a H➜WW➜lνlν like example sample folder:

```
// load the example sample folder from the external ROOT file
TQSampleFolder * samples =
  TQSampleFolder::loadSampleFolder(
  „example3.root:samples“);

// print the contents of the sample folder
samples->print();

// explore the hierarchy, e.g.:
samples->print(„data:“);
samples->print(„data/ee:“);
samples->print("bkg/ee/diboson/WW/qqWW:d");

// print the full hiearchy
samples->print(„rH“);
```

hide system folders starting with „.“

For ref: `TQ(Sample)Folder` classes

The `TQFolder` as well as the `TQSampleFolder` class provide several methods:

- **`folder->getListOfFolders(„name“)`**
  - return a `TList*` with a list of folders matching `„name“`
- **`folder->list(„name“)`**
  - print a list of folders matching `„name“`
- **`folder->getObject(„name“)`**
  - return a `TObject*` to the object matching `„name“`
- **`folder->deleteObject(„name“)`**
  - delete the object matching `„name“`
- **`sampleFolder->getListOfSampleFolders(„name“)`**
  - return a `TList*` with a list of sample folders matching `„name“`

Exercise 1: Locating a specific sample

Load the sample folder of the previous example and try to locate the sample with name „`105200`“ in the hiearchy. What is its path relative to the root folder?

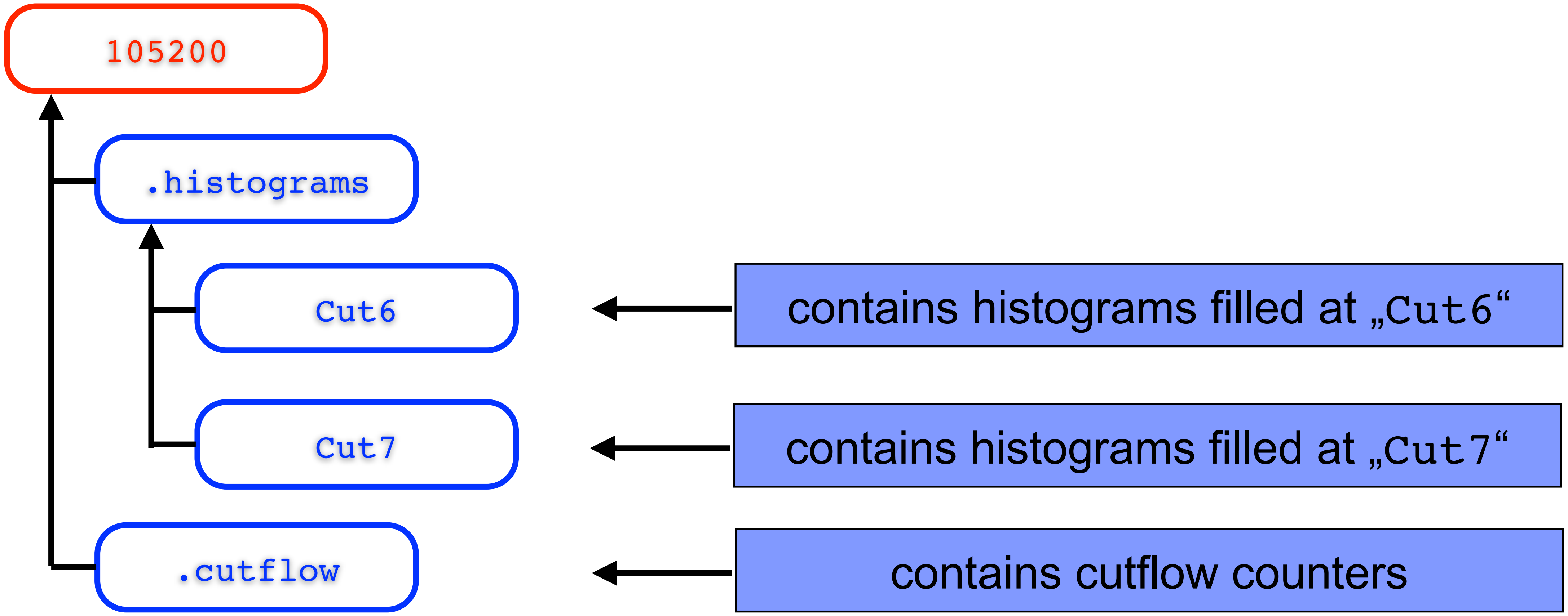
Some hints:

- use the `print()` method to locate the sample (there is more than one sample with this name)
- it is probably much faster doing this in the ROOT command line than writing a macro :-)

Analysis results in a `TQSampleFolder`

Reminder: the `TQSampleFolder` class is designed to represent a certain category of event samples.

Running an analysis means producing histograms, cutflows, event lists, ... („analysis results“) from event samples. These results are stored in the corresponding sample folder.



105200

.histograms

Cut6

Cut7

.cutflow

contains histograms filled at „Cut6“

contains histograms filled at „Cut7“

contains cutflow counters

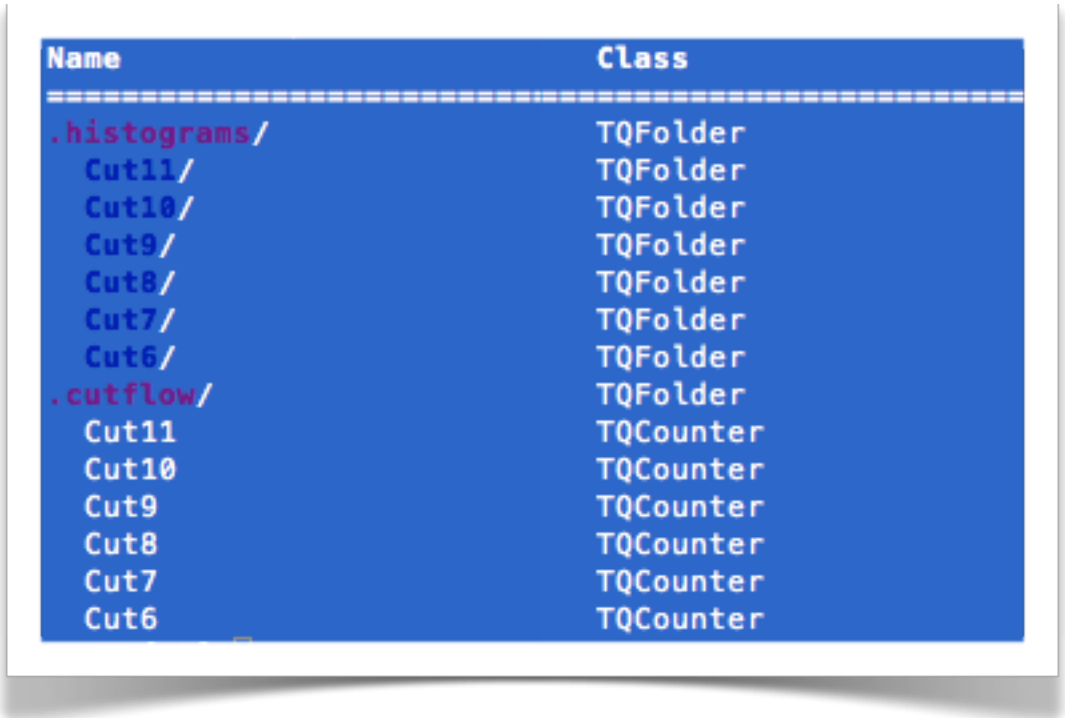
Example 4: Analysis results

// load the example sample folder from the external ROOT file

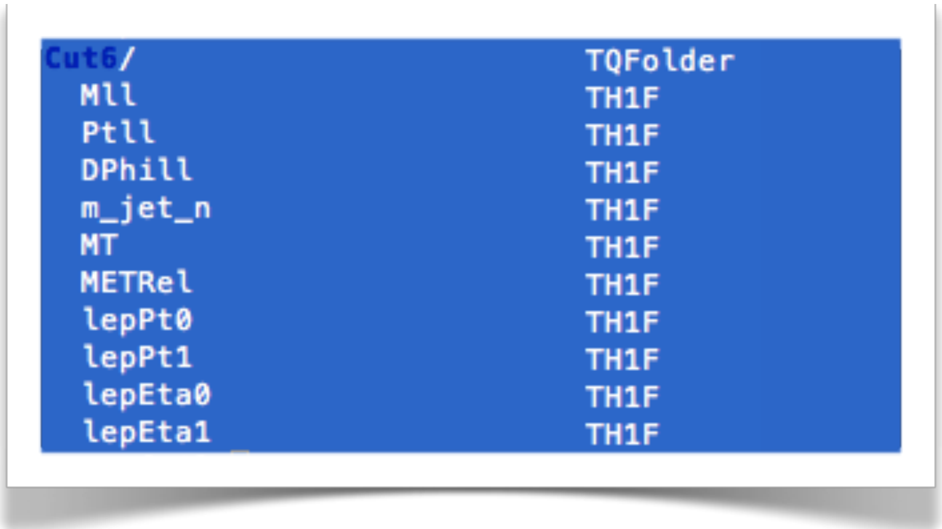
```
TQSampleFolder * samples =
  TQSampleFolder::loadSampleFolder(
  „example3.root:samples“);
```

// print the contents of the sample folder representing the ttbar sample (dataset ID 105200)

```
samples->print(„/bkg/ee/top/ttbar/105200:r1“);
samples->print(„/bkg/ee/top/ttbar/105200:r“);
```



```
Name                          Class
====================================================
.histograms/                  TQFolder
  Cut11/                      TQFolder
  Cut10/                      TQFolder
  Cut9/                       TQFolder
  Cut8/                       TQFolder
  Cut7/                       TQFolder
  Cut6/                       TQFolder
.cutflow/                     TQFolder
  Cut11                       TQCounter
  Cut10                       TQCounter
  Cut9                        TQCounter
  Cut8                        TQCounter
  Cut7                        TQCounter
  Cut6                        TQCounter
```



```
Cut6/                         TQFolder
  Mll                         TH1F
  Ptll                        TH1F
  DPhill                      TH1F
  m_jet_n                     TH1F
  MT                          TH1F
  METRel                      TH1F
  lepPt0                      TH1F
  lepPt1                      TH1F
  lepEta0                     TH1F
  lepEta1                     TH1F
```

The `TQSampleDataReader2` class

The `TQSampleDataReader2` class provides powerful features to retrieve analysis results from a sample folder hierarchy:

- **`reader->getHistogram(„`*`path`*`“, „`*`histogram`*`“);`**
  - return a pointer to a histogram (`TH1*`) which is the sum of all histograms named „*`histogram`*“ in „*`path`*“
- **`reader->getCounter(„`*`path`*`“, „`*`counter`*`“);`**
  - return a poiner to a cutflow counter (`TQCounter*`) which is the sum of all cutflow counters named „*`counter`*“ in „*`path`*“

Example 5: TQSampleDataReader2

```
// load the example sample folder from the external ROOT file
TQSampleFolder * samples =
  TQSampleFolder::loadSampleFolder(
  „example3.root:samples“);

// create an instance of the reader
TQSampleDataReader2 * rd =
  new TQSampleDataReader2(samples);

// make the reader printing error messages (not mandatory but
sometimes helpful)
rd->setVerbose(1);

// retrieve and draw a histogram
TH1 * h = rd->getHistogram(„bkg“, „Cut7/Mll“);
h->Draw(„hist“);
```

continued: `TQSampleDataReader2`

```
// get histogram as sum of „bkg“ and „sig“
rd->getHistogram(„bkg + sig“, „Cut7/Mll“);
```

also supporting subtraction, e.g. `„data - bkg“`

```
// get histogram as sum of „Cut7“ and „Cut8“
rd->getHistogram(„bkg“, „Cut7/Mll + Cut8/Mll“);
```

```
// use wildcards (returns the sum of matching folders)
rd->getHistogram(„bkg/?/top“, „Cut7/Mll“);
```

„?“ matches any sample folder! In this example:

```
bkg/ee/top
bkg/em/top
bkg/mm/top
```

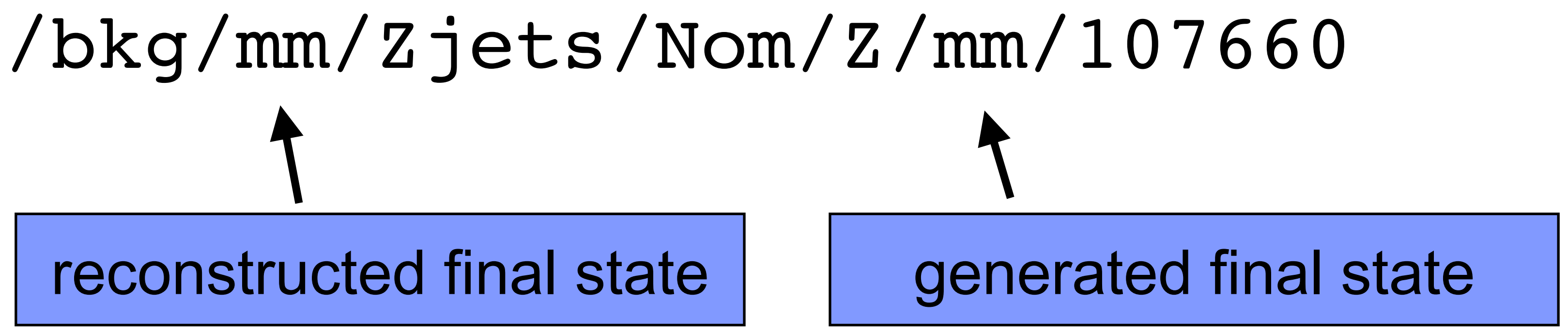
```
// return a normalized histogram (integral = 1.)
rd->getHistogram(„bkg“, „Cut7/Mll“, „norm=true“);
```

Exercise 2: nJets of Z+jets samples

Load the sample folder of exercise 1 and plot the normalized nJets („`m_jet_n`“) distributions of different parton multiplicity (Z➜μμ)+jets samples at „`Cut6`“ in one plot.

Some hints:

- use the `print()` method to locate the Z+jets samples. Recall that option „d“ displays additional information
- concentrate on μμ generated and reconstructed final state, e.g.



```
/bkg/mm/Zjets/Nom/Z/mm/107660
```

The `TQCounter` class

The `TQCounter` class is a representation of an event counter including final event yields, uncertainties and raw event counts (without any normalization or event weights applied):

- **`counter->getCounter();`**
  - return the final event yield
- **`counter->getError();`**
  - return the (statistical) uncertainty
- **`counter->getRawCounter();`**
  - return the raw number of events (integer)

Example 6: Reading cutflows

```
// load the example sample folder from the external ROOT file
TQSampleFolder * samples =
  TQSampleFolder::loadSampleFolder(
  „example3.root:samples“);

// create an instance of the reader
TQSampleDataReader2 * rd =
  new TQSampleDataReader2(samples);

// retrieve the counter for total background at Cut7
TQCounter * c = rd->getCounter(„bkg“, „Cut7“);

// do whatever you like with the numbers
cout << „Total Background = “ << c->getCounter()
  << „ +/- “ << c->getError() << endl;
```

Exercise 3: Cut efficiencies

Load the sample folder of exercise 1 and calculate and print the cut efficiencies of „`Cut11`“ with respect to „`Cut10`“ for the total background and for the signal of a 125 GeV Higgs for ee, eμ and μμ final state, respectively.

Some hints:

- In the default H➔WW➔lνlν structure, the path for different lepton flavour final states is
  - total background: „`bkg/ee`“, „`bkg/em`“, „`bkg/mm`“
  - 125 GeV Higgs signal: „`sig/ee/mh125`“, „`sig/em/mh125`“, „`sig/mm/mh125`“

Tags

You can associate tags (key-value-pairs) to a `TQFolder`, `TQSampleFolder`, ... object to store arbitrary additional information or control the behavior of classes accessing the folder hierarchy

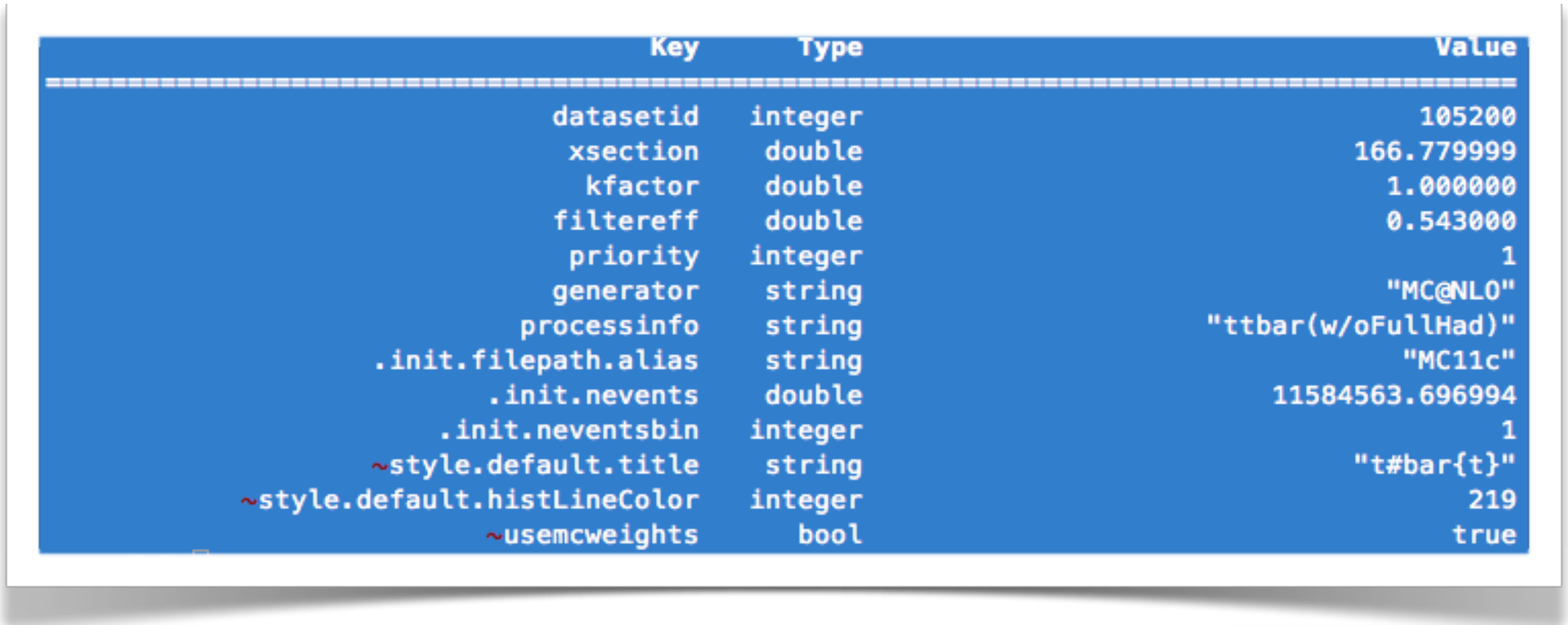
key

- **`folder->setTagInteger(„mass“, 300);`**

value

or print the list of tags associated to the folder

- **`folder->printTags();`**

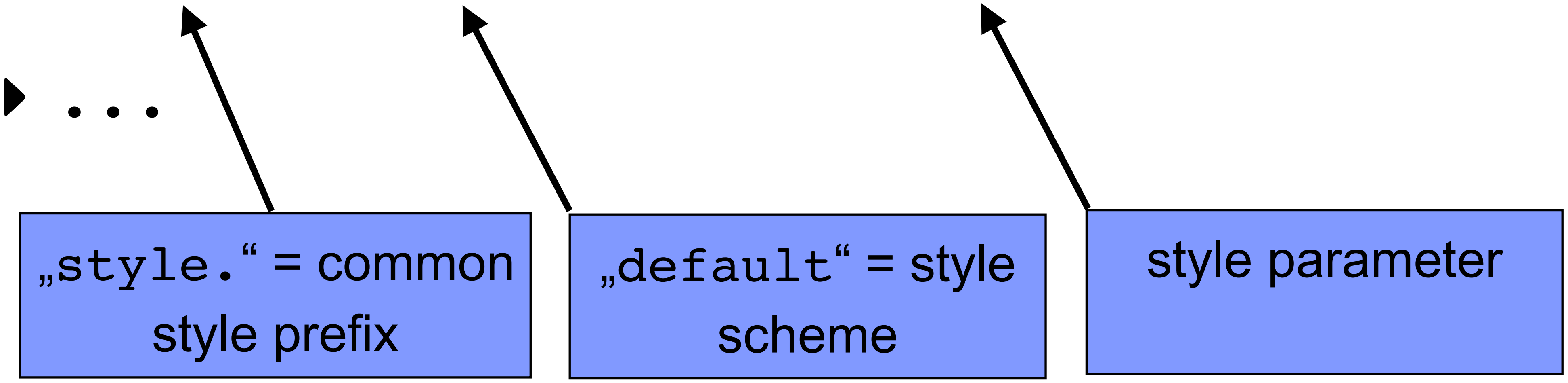


```
                                   Key       Type                                    Value
==========================================================================================
                             datasetid    integer                                   105200
                              xsection     double                               166.779999
                               kfactor     double                                 1.000000
                             filtereff     double                                 0.543000
                              priority    integer                                        1
                             generator     string                                 "MC@NLO"
                           processinfo     string                      "ttbar(w/oFullHad)"
                   .init.filepath.alias     string                                  "MC11c"
                           .init.nevents     double                        11584563.696994
                        .init.neventsbin    integer                                        1
                   ~style.default.title     string                               "t#bar{t}"
           ~style.default.histLineColor    integer                                      219
                           ~usemcweights       bool                                     true
```

continued: `TQSampleDataReader2`

When reading histograms, the `TQSampleDataReader2` class reads specific tags associated to the sample folder (or its base sample folders) and applies corresponding style settings to the histogram to be returned:

- `style.default.histLineColor`
- `style.default.histFillColor`
- `...`



„`style.`“ = common style prefix

„`default`“ = style scheme

style parameter

For reference: style tags

The style tags currently supported by the `TQSampleDataReader2` class:

- `(Integer) style.default.color`
- `(Integer) style.default.histLineColor`
- `(Integer) style.default.histLineStyle`
- `(Integer) style.default.histLineWidth`
- `(Integer) style.default.histFillColor`
- `(Integer) style.default.histFillStyle`
- `(Integer) style.default.histMarkerStyle`
- `(Double)  style.default.histMarkerSize`
- `(Integer) style.default.histMarkerColor`
- `(String)  style.default.title`

Exercise 4: Applying styles

Repeat exercise 2 but now using different line colors for distributions from different samples.

Some hints:

- a third argument of the `setTagInteger()` method can be used to specify the sub folder(s) to apply the tag to

```
folder->setTagInteger(
  „style.default.histLineColor“, kRed,
  „/bkg/mm/Zjets/Nom/Z/mm/107660“);
```

Generalizing histograms

By default histograms are filled and stored for every input sample individually potentially resulting in a very large data structure (you may remember the code crashing when trying to write such a large folder to a file).

In most of the cases it is sufficient to have histograms filled for a set of major sample categories.

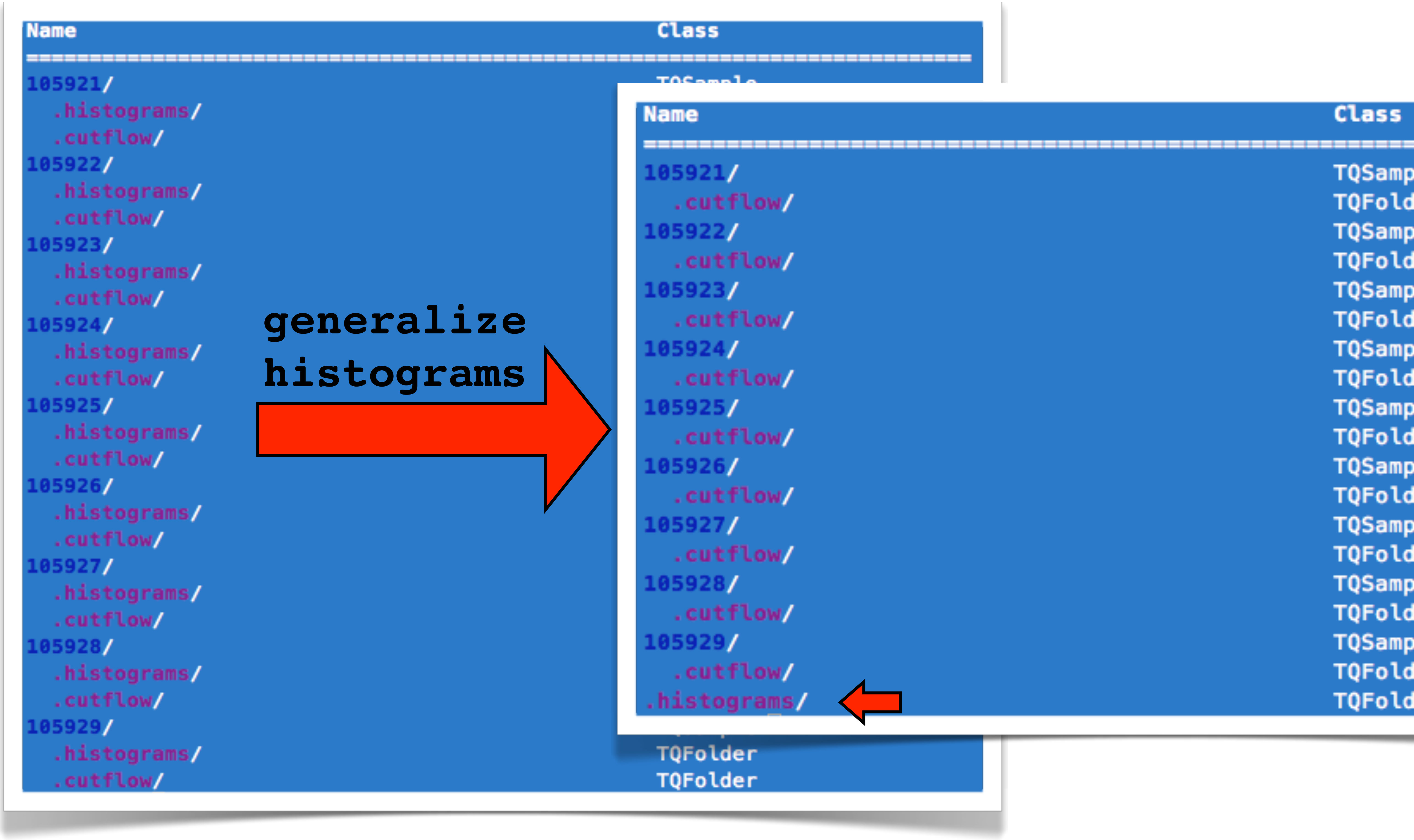
The `TQSampleFolder` class provides a method which automatically „generalizes“ histograms of its sub sample folders to the level of the sample folder you call this method at:

- **`sampleFolder->generalizeHistograms();`**

Example 5: Generalizing histograms

```
// load the example sample folder from the external ROOT file
TQSampleFolder * samples =
  TQSampleFolder::loadSampleFolder(
  „example3.root:samples“);

// show individual histogram folders of qq->WW->evev samples
samples->print(„bkg/ee/diboson/WW/qqWW:rh“);

// generalize the histograms of qq->WW->evev samples
samples->getSampleFolder(
  „bkg/ee/diboson/WW/qqWW“)
  ->generalizeHistograms();

// see what happend to the individual histogram folders
samples->print(„bkg/ee/diboson/WW/qqWW:rh“);
```

Example 5: Generalizing histograms



```
Name                                                        Class
============================================================================
105921/                                                     TQSample
  .histograms/
  .cutflow/
105922/
  .histograms/
  .cutflow/
105923/
  .histograms/
  .cutflow/
105924/
  .histograms/
  .cutflow/
105925/
  .histograms/
  .cutflow/
105926/
  .histograms/
  .cutflow/
105927/
  .histograms/
  .cutflow/
105928/
  .histograms/
  .cutflow/
105929/
  .histograms/                                              TQFolder
  .cutflow/                                                 TQFolder
```

```
Name                                                        Class
============================================================================
105921/                                                     TQSamp
  .cutflow/                                                 TQFold
105922/                                                     TQSamp
  .cutflow/                                                 TQFold
105923/                                                     TQSamp
  .cutflow/                                                 TQFold
105924/                                                     TQSamp
  .cutflow/                                                 TQFold
105925/                                                     TQSamp
  .cutflow/                                                 TQFold
105926/                                                     TQSamp
  .cutflow/                                                 TQFold
105927/                                                     TQSamp
  .cutflow/                                                 TQFold
105928/                                                     TQSamp
  .cutflow/                                                 TQFold
105929/                                                     TQSamp
  .cutflow/                                                 TQFold
.histograms/                                                TQFold
```

Exercise 5: Generalizing histograms

Load the sample folder of a default H→WW→lνlν analysis file and write it to a new file with reduced size by generalizing histograms to the level of the main background categories (W+jets, Z+jets, WW, WZ/ZZ/Zɣ, ttbar, single top).

Some hints:

- use `hww_dataMC.root` as analysis input file
- each background category is present three times (for ee, eμ and μμ final state respectively)

For ref: H→WW→lνlν sample structure

The default H→WW→lνlν sample structure is given by

- bkg/
    - ee/ (em/, mm/)
        - diboson/
            - WW/
                - ...
            - NonWW/
                - ...
        - Zjets/
            - ...
        - Wjets/
            - ...
        - top/
            - ttbar/
            - singletop/
- sig/
    - ee/ (em/, mm/)
        - mh125/ (mh110/, ..., mh600/)
            - ggf/
            - vbf/
            - WH/
            - ZH/
- data/
    - ee/ (em/, mm/)

The `TQHWWPlotter2` class

Producing plots in the H→WW→lνlν style from a corresponding sample folder hierarchy can be as easy as two lines of code (the `TQHWWPlotter2` class is specific to H→WW→lνlν)

```
▶ TQHWWPlotter2 * pl =
    new TQHWWPlotter2(samples);

  pl->plot(„Cut10_ALL/METRel“);

  pl->plot(„mm:Cut10_ALL/METrel“);
```

only μμ final state

Set additional parameter to control the plotter (e.g. show a ratio plot of data/MC):

```
▶ pl->plot(„Cut10_ALL/METRel“,
    „style.showRatio=true“);
```

Example 6: Plots in H→WW→$l\nu l\nu$ style

```
// load the sample folder of exercise 5
TQSampleFolder * samples =
  TQSampleFolder::loadSampleFolder(
  „hww_dataMC_genHisto.root:samples“);

// create an instance of the HWW plotter:
TQHWWPlotter2 * pl = new TQHWWPlotter2(samples);

// create and save a plot with data/MC ratio
TCanvas * c = pl->plot(„Cut10_ALL/METRel“,
  „style.showRatio = true“);
c->SaveAs(„MyHWWPlot.eps“);
```

For reference: TQHWWPlotter2

The paramter currently supported by the `TQHWWPlotter2` class:

- `input.mh`
- `style.logScale`
- `style.showRatio`
- `style.ratioMin`
- `style.ratioMax`
- `style.forceRatioLimits`
- `style.showUnderflow`
- `style.showOverflow`
- `labels.info`
- `labels.lumi`
- `labels.process`

The TQCutflowPrinter2 class

The TQCutflowPrinter2 class is an analysis independent class producing cutflow tables from corresponding sample folder hierarchies.

A cutflow table is matrix-like structure with different processes and cut stages in its rows and columns.

Construct a cutflow table using:

- **printer->addCut(„cut“, „title“);**
- **printer->addProcess(„path“, „title“);**
- **printer->createTable();**
- **printer->writeTable(...);**

Example 7: Cutflow tables

```
// load the sample folder of exercise 5
TQSampleFolder * samples =
  TQSampleFolder::loadSampleFolder(
  „hww_dataMC_genHisto.root:samples“);

// create an instance of the cutflow printer:
TQCutflowPrinter2 * pr = new
  TQCutflowPrinter2(samples);

// define the table
pr->addCut(„Cut10_ALL“, „Z veto“);
pr->addCut(„Cut11_ALL“, „METRel cut“);
pr->addProcess(„sig/ee/mh125“, „Signal“);
pr->addProcess(„bkg/ee/Zjets“, „Z+jets“);
pr->addProcess(„bkg/ee“,       „Total Background“);
pr->addProcess(„data/ee“,      „Observed“);
```

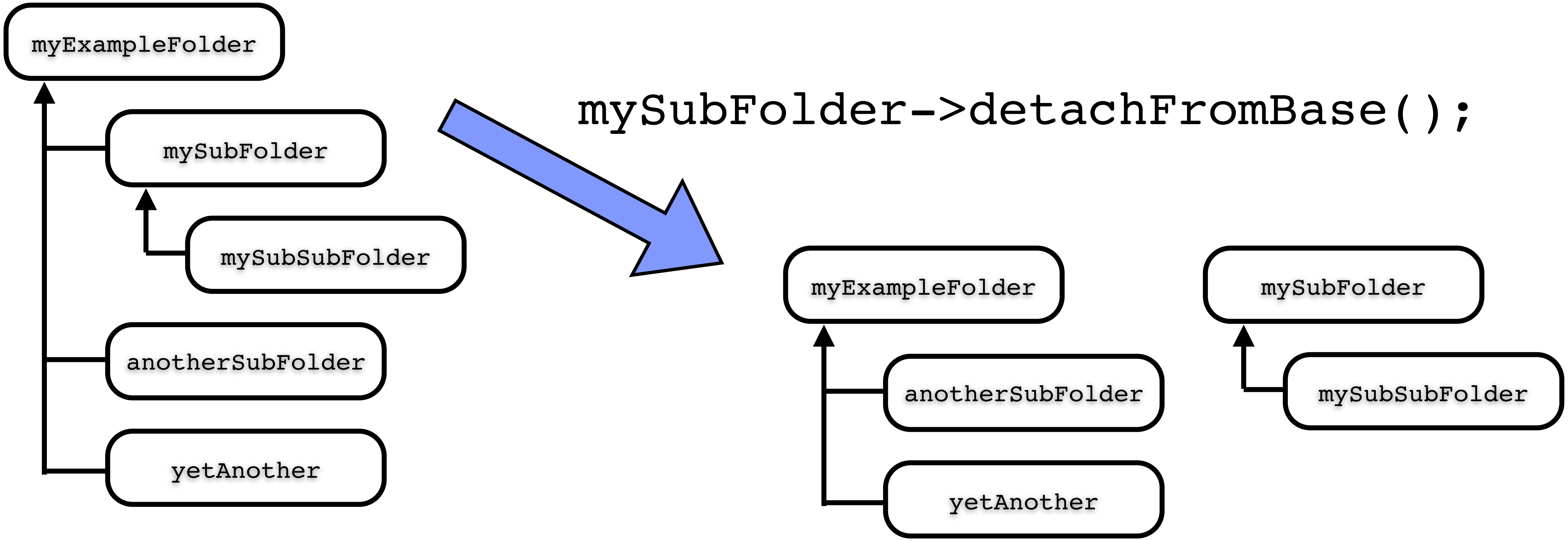
continued: Example 7: Cutflow tables

```
...

// create the table
TList * table = pr->createTable(
  „style.cellWidth = 30“);

// print the table
pr->writeTable(table);

// write table to a file
pr->writeTable(table, „file.txt“);
```

Moving folders

Folders in a hiearchy may be deleted, moved, copied

- **`detachFromBase()`**
- **`addFolder(...), addSampleFolder(...)`**
- **`copy()`**



```
mySubFolder->detachFromBase();
```

Example 8: Deleting folders

```
// load the example sample folder from the external ROOT file
TQSampleFolder * samples =
  TQSampleFolder::loadSampleFolder(
  „example3.root:samples“);

// print the contents of the sample folder
samples->print();

// delete the signal sample folder
delete samples->getSampleFolder(„sig“)
  ->detachFromBase();

// print the contents of the sample folder again
samples->print();
```

Example 9: Moving folders

```
// load the example sample folder from the external ROOT file
TQSampleFolder * samples =
  TQSampleFolder::loadSampleFolder(
  „example3.root:samples“);

// create a new folder to put the Wjets folder into
TQSampleFolder * dest = samples
  ->getSampleFolder(„newWjetsSampleFolder+“);

// detach the Wjets folder from its base...
TQSampleFolder * Wjets = (TQSampleFolder*)samples
  ->getSampleFolder(„bkg/ee/Wjets“)
  ->detachFromBase();

// ...and put it into its new destination folder
dest->addSampleFolder(Wjets);
```

Exercise 6: Moving folders

Load the sample folder written in exercise 5 and the sample folder containing the data-driven W+jets estimate and replace the Monte Carlo W+jets expectation by the data-driven one (for ee final state only).

Some hints:

- use `hww_ddWjets_genHist.root` as data-driven W +jets input file (the root sample folder in this file is named „`Wjets`“)
- the data-driven W+jets estimate for the ee final state is the sum of „`data/ee`“ and „`bkg/ee`“ in the data-driven W +jets sample folder, so you need to copy/move two folders
- please note: these two folders have the same name („`ee`“)

Additional exercises

Load the `hww_dataMC_genHist.root` file and produce a plot (using the `TQHWWPlotter2` class) of the transverse mass („`MT`“ or „`MT_wide`“) distribution in the 0 jet signal region („`Cutt2_0jet_lowmass_SF_ALL`“), but blinding the data.

Some hints:

- `samples->getListOfFolders(„data/*/.histograms/Cutt2_0jet_lowmass_SF_ALL“)` returns a `TList*` of every folder containing data histograms after cut Cutt2_0jet_lowmass_SF_ALL