Emil Gorm Nielsen, Niels Bohr Institute

Flow in O2 with the Generic Framework

O2 hands-on tutorial

April 26th

THE VELUX FOUNDATIONS

VILLUM FONDEN VELUX FONDEN

Overview

- What is the Generic Framework?
- The Generic Framework in O2
    - Initialisation
    - Correlator configurations
    - Filling and calculating correlations
- Walkthrough of the skeleton code
- Exercises

These slides, shell commands and solutions to the exercises can be found in

https://github.com/AliceO2Group/analysis-tutorials/tree/master/o2at-2/PWGCF/GenericFramework

The Generic Framework

- Calculate m-particle correlations with recursive formula fast and exact

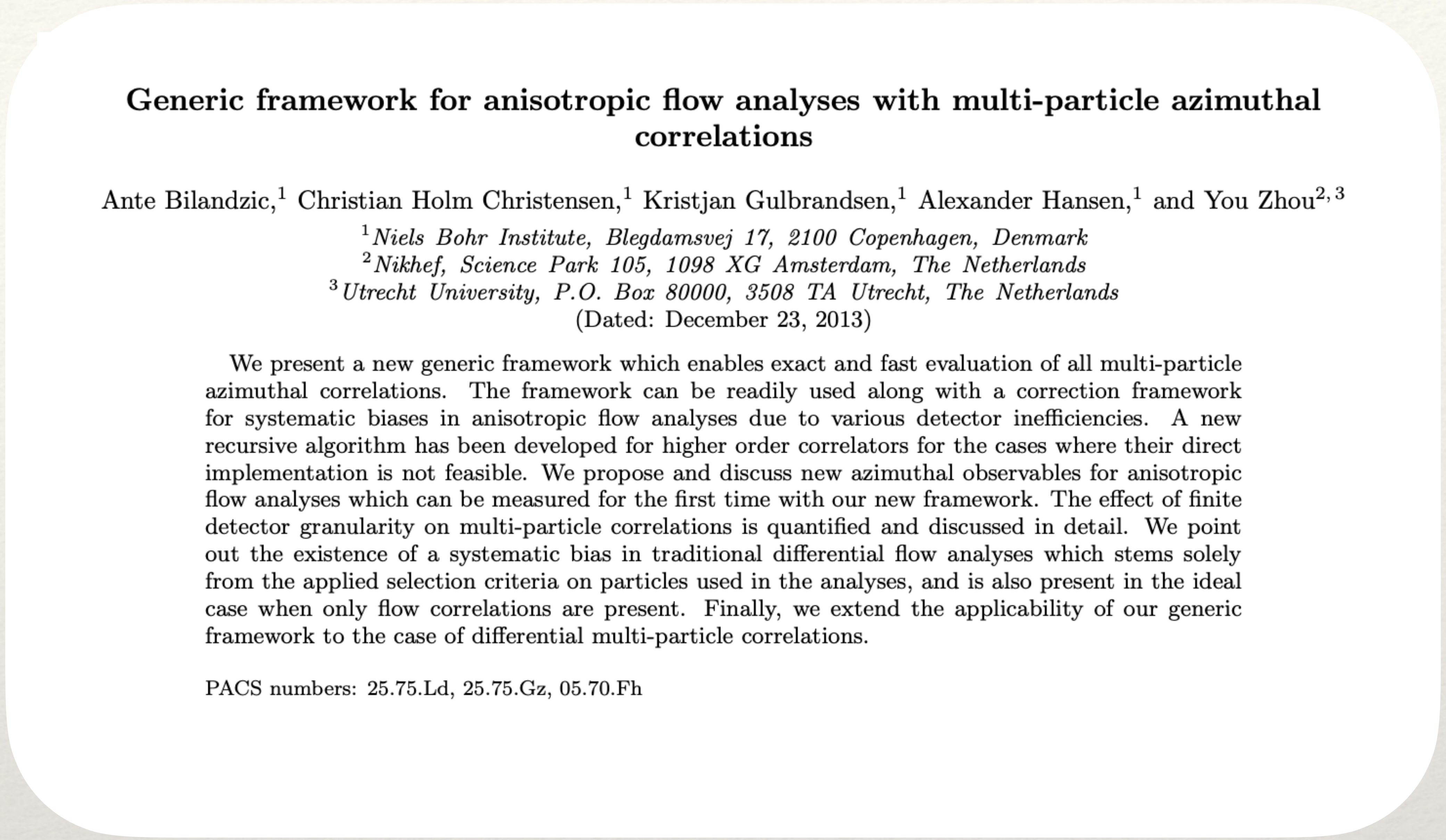
$$\langle m\rangle_{n_1,n_2,\ldots,n_m} \equiv \left\langle e^{i(n_1\varphi_{k_1}+n_2\varphi_{k_2}+\cdots+n_m\varphi_{k_m})}\right\rangle \equiv \frac{\displaystyle\sum_{\substack{k_1,k_2,\ldots,k_m=1\\ k_1\neq k_2\neq\ldots\neq k_m}}^{M} w_{k_1}w_{k_2}\cdots w_{k_m}\, e^{i(n_1\varphi_{k_1}+n_2\varphi_{k_2}+\cdots+n_m\varphi_{k_m})}}{\displaystyle\sum_{\substack{k_1,k_2,\ldots,k_m=1\\ k_1\neq k_2\neq\ldots\neq k_m}}^{M} w_{k_1}w_{k_2}\cdots w_{k_m}}\,. \qquad (7)$$

- Denominator trivially related to numerator

$$\mathrm{N}\langle m\rangle_{n_1,n_2,\ldots,n_m} \equiv \sum_{\substack{k_1,k_2,\ldots,k_m=1\\ k_1\neq k_2\neq\ldots\neq k_m}}^{M} w_{k_1}w_{k_2}\cdots w_{k_m}\, e^{i(n_1\varphi_{k_1}+n_2\varphi_{k_2}+\cdots+n_m\varphi_{k_m})}\,, \qquad (16)$$

$$\mathrm{D}\langle m\rangle_{n_1,n_2,\ldots,n_m} \equiv \sum_{\substack{k_1,k_2,\ldots,k_m=1\\ k_1\neq k_2\neq\ldots\neq k_m}}^{M} w_{k_1}w_{k_2}\cdots w_{k_m} \qquad (17)$$

$$= \mathrm{N}\langle m\rangle_{0,0,\ldots,0}\,. \qquad (18)$$



Generic framework for anisotropic flow analyses with multi-particle azimuthal correlations

Ante Bilandzic,[1] Christian Holm Christensen,[1] Kristjan Gulbrandsen,[1] Alexander Hansen,[1] and You Zhou[2,3]

[1] *Niels Bohr Institute, Blegdamsvej 17, 2100 Copenhagen, Denmark*
[2] *Nikhef, Science Park 105, 1098 XG Amsterdam, The Netherlands*
[3] *Utrecht University, P.O. Box 80000, 3508 TA Utrecht, The Netherlands*

(Dated: December 23, 2013)

We present a new generic framework which enables exact and fast evaluation of all multi-particle azimuthal correlations. The framework can be readily used along with a correction framework for systematic biases in anisotropic flow analyses due to various detector inefficiencies. A new recursive algorithm has been developed for higher order correlators for the cases where their direct implementation is not feasible. We propose and discuss new azimuthal observables for anisotropic flow analyses which can be measured for the first time with our new framework. The effect of finite detector granularity on multi-particle correlations is quantified and discussed in detail. We point out the existence of a systematic bias in traditional differential flow analyses which stems solely from the applied selection criteria on particles used in the analyses, and is also present in the ideal case when only flow correlations are present. Finally, we extend the applicability of our generic framework to the case of differential multi-particle correlations.

PACS numbers: 25.75.Ld, 25.75.Gz, 05.70.Fh

Recursive algorithm

```
$\mathrm{N}\langle 1\rangle'_{n_1}$: return $Q_{n_1,1}$
$\mathrm{N}\langle m\rangle'_{n_1,\ldots,n_m}$:
    $C \leftarrow 0$
    for $k \leftarrow (m-1), 1$ do
        for each combination $c = \{c_1,\ldots,c_k\}$ of $\{n_1,\ldots,n_{m-1}\}$ do
            $q \leftarrow \sum_{j \text{ not in } c} n_j$
            $C \leftarrow C + (-1)^{m-k}\,(m-k-1)! \times \mathrm{N}\langle k\rangle'_{c_1,\ldots,c_k} \times Q_{q,m-k}$
        end for each $c$
    end for $k$
    return $C$ .
```

Initialisation

- The GFW class is a fast, reliable and easy to use c++ class

```
GFW* fGFW = new GFW();
```

Initialise like any c++ object

- Works with 'regions' of η

Name of region η_{min} η_{max} N_{p_T} Bitmask

```
fGFW->AddRegion("full", -0.8, 0.8, 1, 1);
```

- Specify the desired correlations in the correlator configurations

Name of region Harmonics Correlator name p_T flag

```
corrconfigs.push_back(fGFW->GetCorrelatorConfig("full {2 -2}", "ChFull22", kFALSE));
```

- Finalize the initialisation with `fGFW->CreateRegions();`

More documentation at
https://github.com/vvislavi/GFW_Core

Correlator configurations

- The general syntax for a single subevent is

  “[POI] Ref [| Overlap] {harm1[,harm2,…,harmN]”

- Example: calculating p_T-differential flow

  Define the required regions

```
fGFW->AddRegion("refN", -0.8, -0.4, 1, 1);
fGFW->AddRegion("refP", 0.4, 0.8, 1, 1);
fGFW->AddRegion("poiN", -0.8, -0.4, 1+fPtAxis->GetNbins(), 2);
fGFW->AddRegion("olN", -0.8, -0.4, 1, 4);
```

Note: bitmasks for POI (2) and overlap (4) are different from ref (1), so that we can fill them with the correct particles

Fill the correlator configs, while specifying the POI and overlap

```
corrconfigs.push_back(fGFW->GetCorrelatorConfig("refN {2} refP {-2}", "ChGap22", kFALSE));
corrconfigs.push_back(fGFW->GetCorrelatorConfig("poiN refN | olN {2} refP {-2}", "ChGap22", kTRUE));
```

More documentation at https://github.com/vvislavi/GFW_Core

Filling and calculating with the GFW

- Fill the GFW in the track loop

η p_T bin φ NUA/NUE Bitmask

```
fGFW->Fill(track.eta(), 1, track.phi(), wacc * weff, 1);
```

- If we have POI then we can fill according to the bitmasks with some additional selection

```
bool WithinPtPOI = (cfgCutPtPOIMin<pt) && (pt<cfgCutPtPOIMax); //within POI pT range
bool WithinPtRef  = (cfgCutPtMin<pt) && (pt<cfgCutPtMax);  //within RF pT range
if(WithinPtRef) fGFW->Fill(track.eta(), fPtAxis->FindBin(pt)-1, track.phi(), wacc * weff, 1);
if(WithinPtPOI) fGFW->Fill(track.eta(), fPtAxis->FindBin(pt)-1, track.phi(), wacc * weff, 2);
if(WithinPtPOI && WithinPtRef) fGFW->Fill(track.eta(), fPtAxis->FindBin(pt)-1, track.phi(), wacc * weff, 4);
```

- Calculating the correlation values

Correlator config p_T bin Harms to zero flag

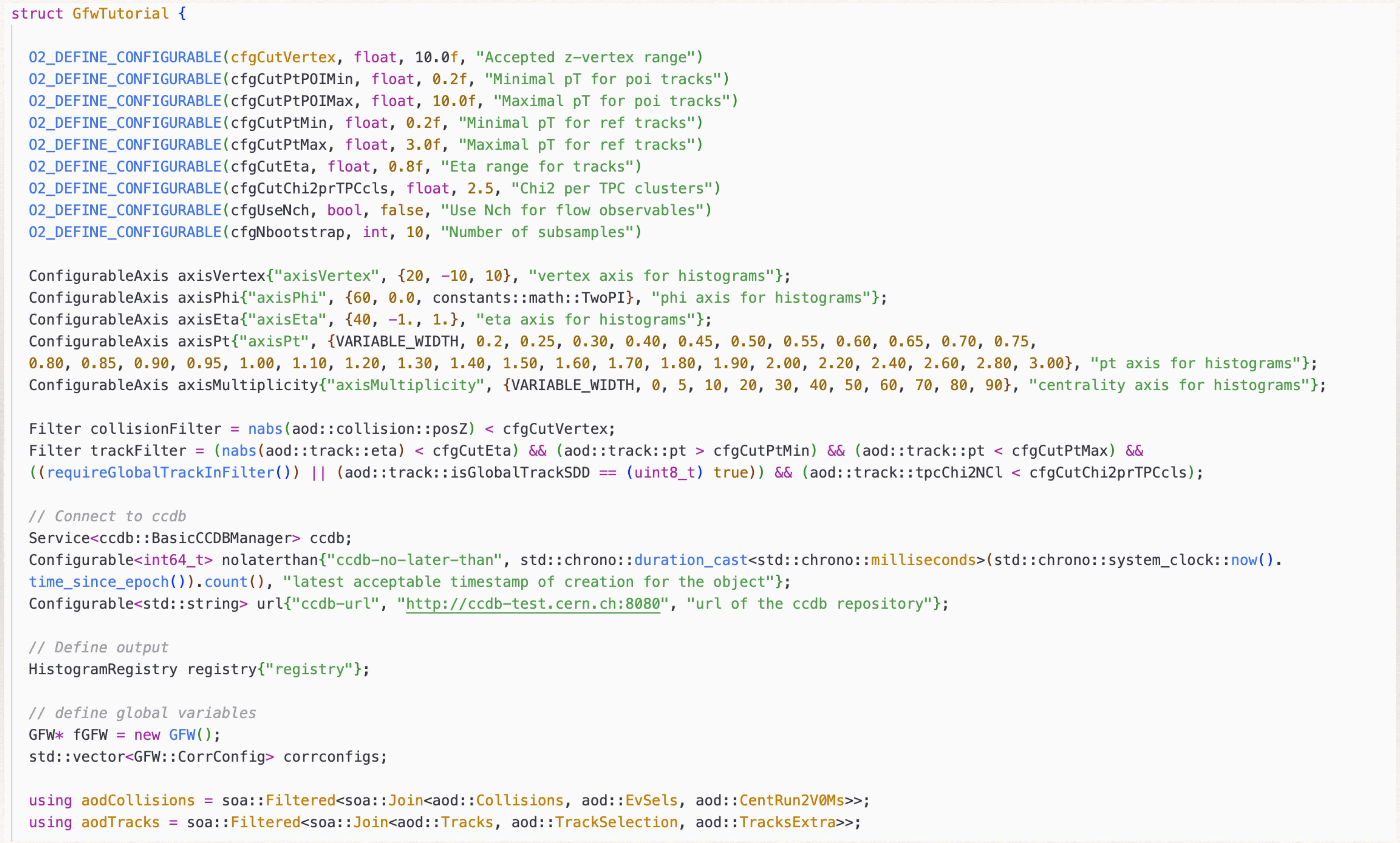
```
dnx = fGFW->Calculate(corrconf,0,kTRUE).real();
```

```
val = fGFW->Calculate(corrconf,0,kFALSE).real()/dnx;
```

- This can then be filled in e.g. TProfile

More documentation at https://github.com/vvislavi/GFW_Core

Skeleton code



```
struct GfwTutorial {

  O2_DEFINE_CONFIGURABLE(cfgCutVertex, float, 10.0f, "Accepted z-vertex range")
  O2_DEFINE_CONFIGURABLE(cfgCutPtPOIMin, float, 0.2f, "Minimal pT for poi tracks")
  O2_DEFINE_CONFIGURABLE(cfgCutPtPOIMax, float, 10.0f, "Maximal pT for poi tracks")
  O2_DEFINE_CONFIGURABLE(cfgCutPtMin, float, 0.2f, "Minimal pT for ref tracks")
  O2_DEFINE_CONFIGURABLE(cfgCutPtMax, float, 3.0f, "Maximal pT for ref tracks")
  O2_DEFINE_CONFIGURABLE(cfgCutEta, float, 0.8f, "Eta range for tracks")
  O2_DEFINE_CONFIGURABLE(cfgCutChi2prTPCcls, float, 2.5, "Chi2 per TPC clusters")
  O2_DEFINE_CONFIGURABLE(cfgUseNch, bool, false, "Use Nch for flow observables")
  O2_DEFINE_CONFIGURABLE(cfgNbootstrap, int, 10, "Number of subsamples")

  ConfigurableAxis axisVertex{"axisVertex", {20, -10, 10}, "vertex axis for histograms"};
  ConfigurableAxis axisPhi{"axisPhi", {60, 0.0, constants::math::TwoPI}, "phi axis for histograms"};
  ConfigurableAxis axisEta{"axisEta", {40, -1., 1.}, "eta axis for histograms"};
  ConfigurableAxis axisPt{"axisPt", {VARIABLE_WIDTH, 0.2, 0.25, 0.30, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.70, 0.75,
  0.80, 0.85, 0.90, 0.95, 1.00, 1.10, 1.20, 1.30, 1.40, 1.50, 1.60, 1.70, 1.80, 1.90, 2.00, 2.20, 2.40, 2.60, 2.80, 3.00}, "pt axis for histograms"};
  ConfigurableAxis axisMultiplicity{"axisMultiplicity", {VARIABLE_WIDTH, 0, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90}, "centrality axis for histograms"};

  Filter collisionFilter = nabs(aod::collision::posZ) < cfgCutVertex;
  Filter trackFilter = (nabs(aod::track::eta) < cfgCutEta) && (aod::track::pt > cfgCutPtMin) && (aod::track::pt < cfgCutPtMax) &&
  ((requireGlobalTrackInFilter()) || (aod::track::isGlobalTrackSDD == (uint8_t) true)) && (aod::track::tpcChi2NCl < cfgCutChi2prTPCcls);

  // Connect to ccdb
  Service<ccdb::BasicCCDBManager> ccdb;
  Configurable<int64_t> nolaterthan{"ccdb-no-later-than", std::chrono::duration_cast<std::chrono::milliseconds>(std::chrono::system_clock::now().
  time_since_epoch()).count(), "latest acceptable timestamp of creation for the object"};
  Configurable<std::string> url{"ccdb-url", "http://ccdb-test.cern.ch:8080", "url of the ccdb repository"};

  // Define output
  HistogramRegistry registry{"registry"};

  // define global variables
  GFW* fGFW = new GFW();
  std::vector<GFW::CorrConfig> corrconfigs;

  using aodCollisions = soa::Filtered<soa::Join<aod::Collisions, aod::EvSels, aod::CentRun2V0Ms>>;
  using aodTracks = soa::Filtered<soa::Join<aod::Tracks, aod::TrackSelection, aod::TracksExtra>>;
```

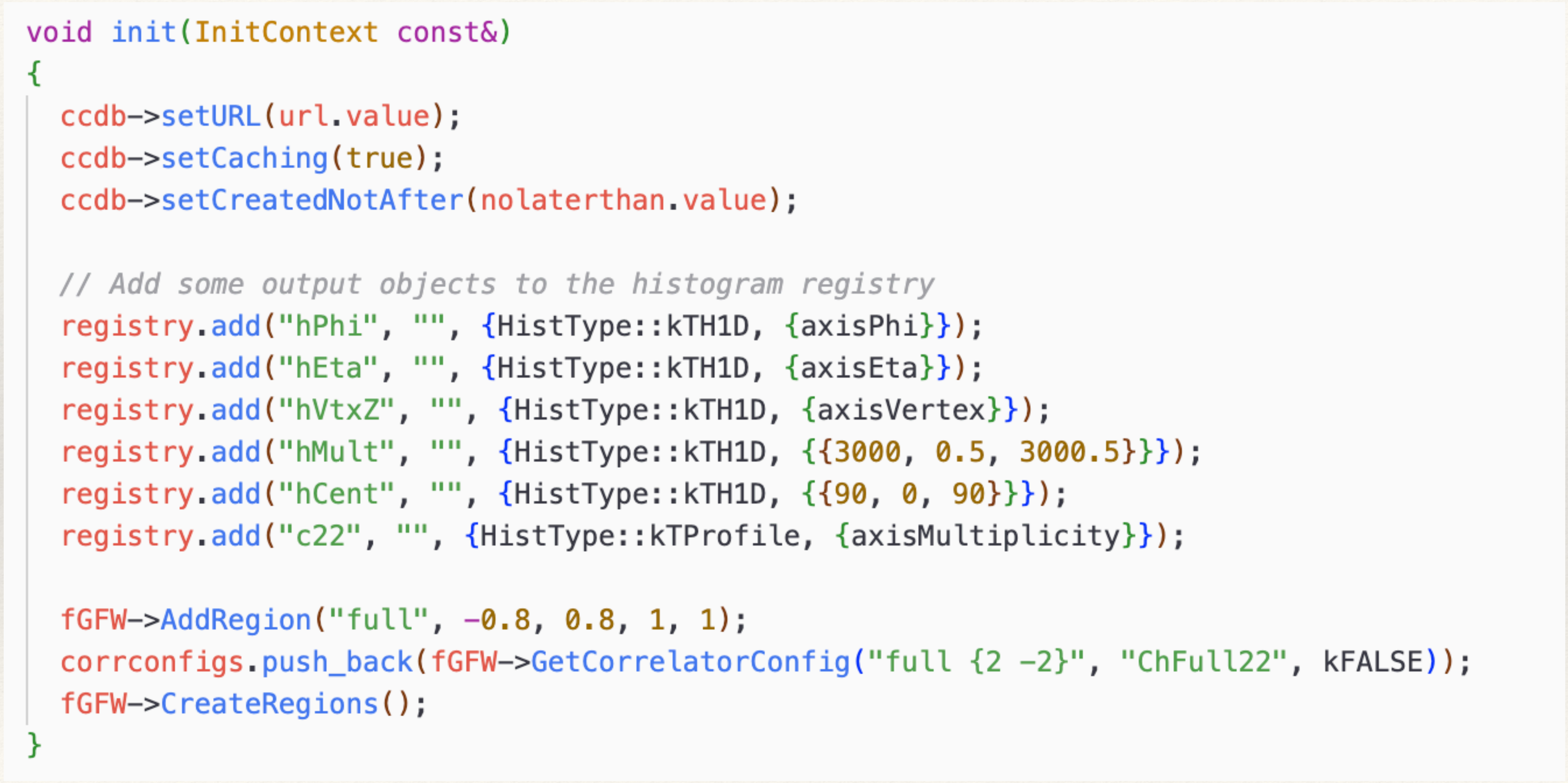
Configurables
(Can be overwritten in the configuration.json)

Track and collision filters

Output objects and gobal variables

aliases for tracks and collision tables

Skeleton code

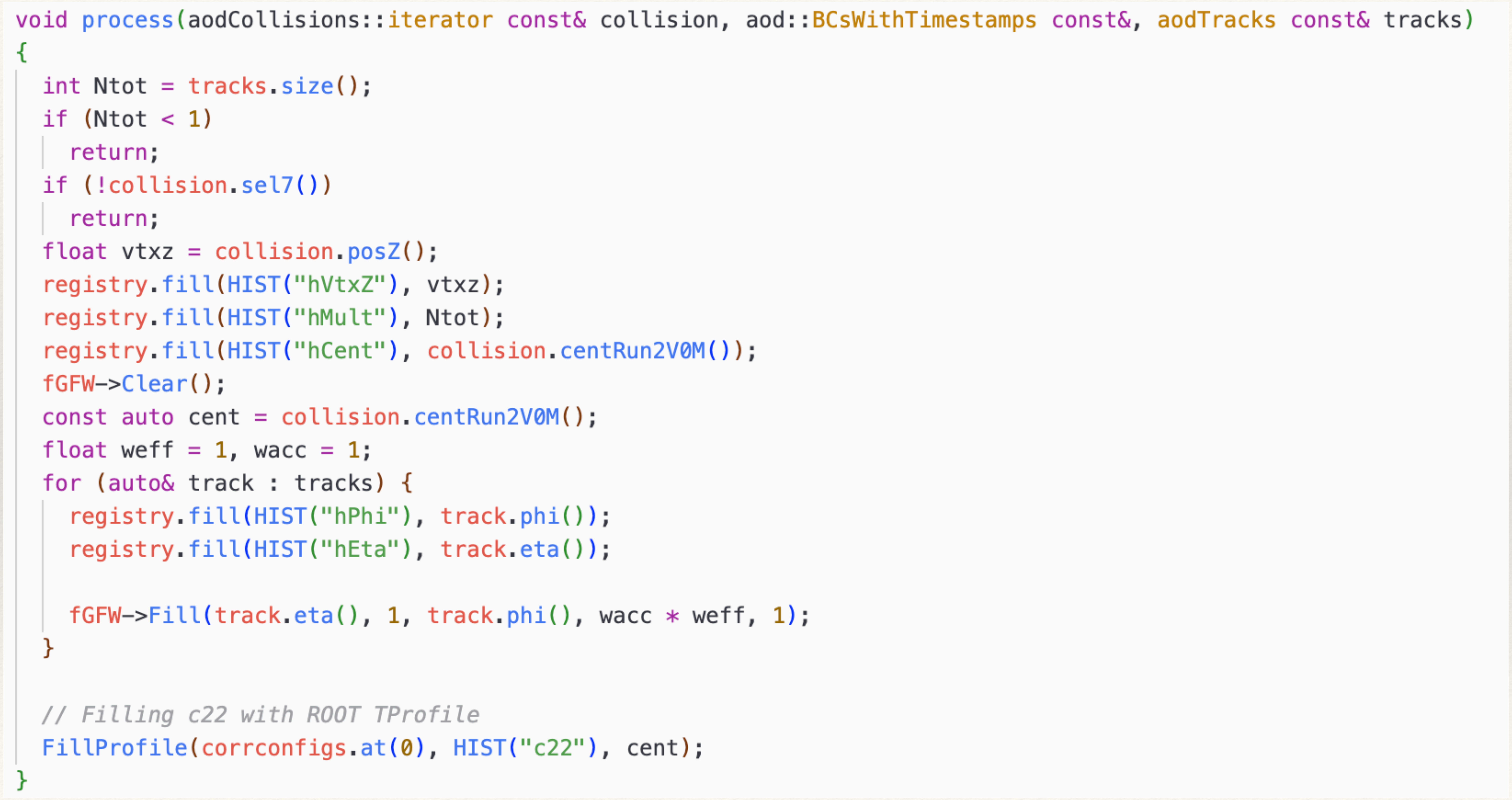


```cpp
void init(InitContext const&)
{
  ccdb->setURL(url.value);
  ccdb->setCaching(true);
  ccdb->setCreatedNotAfter(nolaterthan.value);

  // Add some output objects to the histogram registry
  registry.add("hPhi", "", {HistType::kTH1D, {axisPhi}});
  registry.add("hEta", "", {HistType::kTH1D, {axisEta}});
  registry.add("hVtxZ", "", {HistType::kTH1D, {axisVertex}});
  registry.add("hMult", "", {HistType::kTH1D, {{3000, 0.5, 3000.5}}});
  registry.add("hCent", "", {HistType::kTH1D, {{90, 0, 90}}});
  registry.add("c22", "", {HistType::kTProfile, {axisMultiplicity}});

  fGFW->AddRegion("full", -0.8, 0.8, 1, 1);
  corrconfigs.push_back(fGFW->GetCorrelatorConfig("full {2 -2}", "ChFull22", kFALSE));
  fGFW->CreateRegions();
}
```

Initialise objects in the histogram registry

Initialise the GFW

Skeleton code



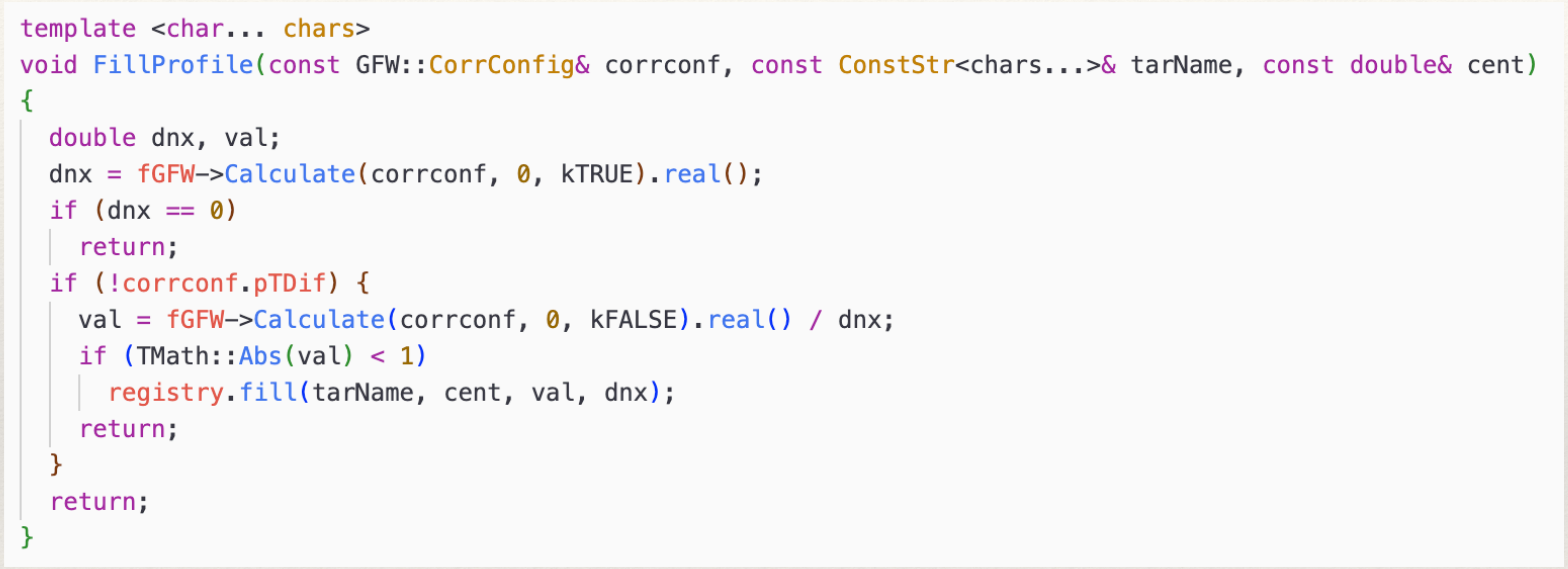
```
void process(aodCollisions::iterator const& collision, aod::BCsWithTimestamps const&, aodTracks const& tracks)
{
  int Ntot = tracks.size();
  if (Ntot < 1)
    return;
  if (!collision.sel7())
    return;
  float vtxz = collision.posZ();
  registry.fill(HIST("hVtxZ"), vtxz);
  registry.fill(HIST("hMult"), Ntot);
  registry.fill(HIST("hCent"), collision.centRun2V0M());
  fGFW->Clear();
  const auto cent = collision.centRun2V0M();
  float weff = 1, wacc = 1;
  for (auto& track : tracks) {
    registry.fill(HIST("hPhi"), track.phi());
    registry.fill(HIST("hEta"), track.eta());

    fGFW->Fill(track.eta(), 1, track.phi(), wacc * weff, 1);
  }

  // Filling c22 with ROOT TProfile
  FillProfile(corrconfigs.at(0), HIST("c22"), cent);
}
```

Track loop

Fill the GFW

Fill correlation values

Skeleton code



```cpp
template <char... chars>
void FillProfile(const GFW::CorrConfig& corrconf, const ConstStr<chars...>& tarName, const double& cent)
{
  double dnx, val;
  dnx = fGFW->Calculate(corrconf, 0, kTRUE).real();
  if (dnx == 0)
    return;
  if (!corrconf.pTDif) {
    val = fGFW->Calculate(corrconf, 0, kFALSE).real() / dnx;
    if (TMath::Abs(val) < 1)
      registry.fill(tarName, cent, val, dnx);
    return;
  }
  return;
}
```

Calculate normalisation

Check for division by zero

Calculate correlation value

Fill TProfile

Running the task

- The provided scripts will take care of downloading and running the tasks with the correct configurations
- Download AO2D data files with the downloadAOD.sh script

```
$> sh downloadAOD.sh
```

- Run the task with the run.sh script

```
$> sh run.sh
```

- To run on Run3 data specify <run3> as argument - will only work if the proper adjustments have been made in the task

```
$> sh downloadAOD.sh run3
$> sh run.sh run3
```

Exercises

1. Add more correlations to calculate v3, v4, v2{4}, etc.
   Add new correlation configurations

2. Add subevents with varying $|\eta|$ gaps
   Add new regions

3. Add p_{T}-differential flow
   Add regions for POI, reference and overlap

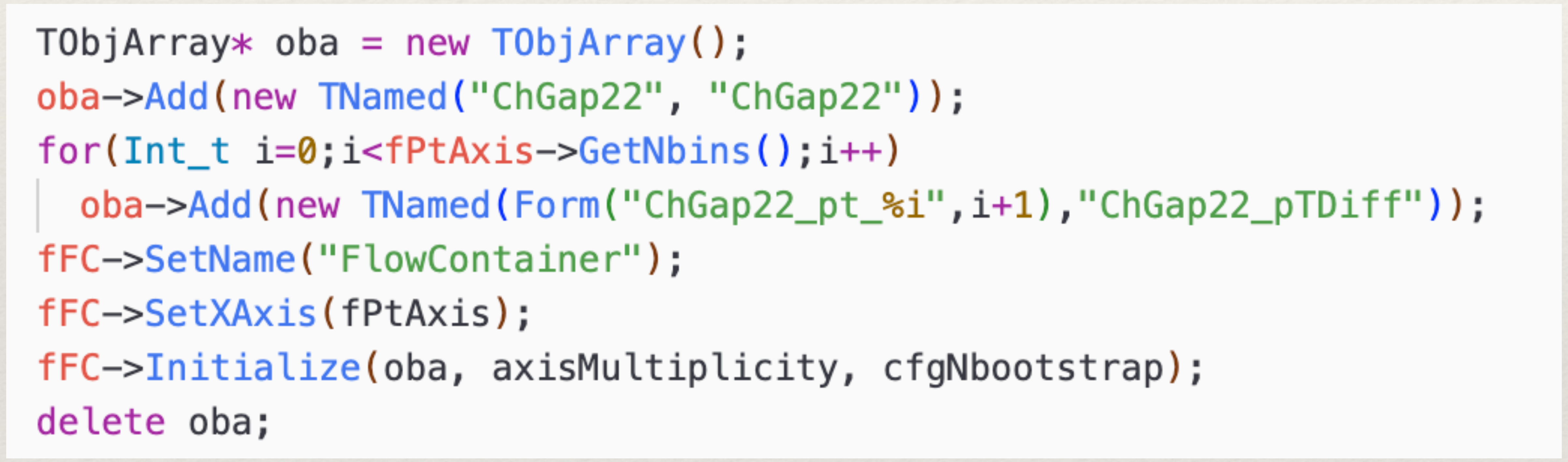
4. Run the previous exercises on run3 data
   Use the appropriate configuration and data file
   Change to a run3 centrality estimator (FT0C) and remove trigger selection

5. Add PID flow
   Add PID tables to the tracks

6. (Optional) Run a flow analysis using the FlowContainer to store the correlations

FlowContainer

- Container that holds the correlations in 2D histogram (correlator config. vs centrality)
  - Handles cumulants calculations, v_n calculations, subsamples for bootstrapping, etc.
- Initialise the FlowContainer



```cpp
TObjArray* oba = new TObjArray();
oba->Add(new TNamed("ChGap22", "ChGap22"));
for(Int_t i=0;i<fPtAxis->GetNbins();i++)
  oba->Add(new TNamed(Form("ChGap22_pt_%i",i+1),"ChGap22_pTDiff"));
fFC->SetName("FlowContainer");
fFC->SetXAxis(fPtAxis);
fFC->Initialize(oba, axisMultiplicity, cfgNbootstrap);
delete oba;
```

ObjArray holds each correlator

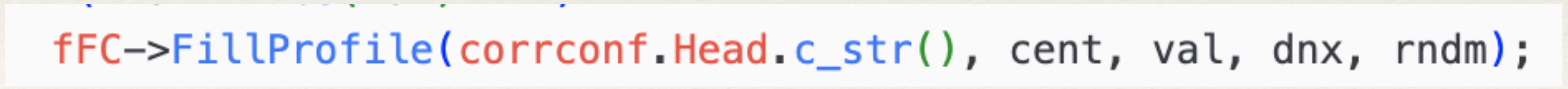
For p_T-differential we add each p_T bin

Setting name and X-axis for p_T-differential

Initialise with the ObjArray, centrality axis, and number of subsamples

- Fill the flow container

Correlator config name — Value — Random number for subsamples



```cpp
fFC->FillProfile(corrconf.Head.c_str(), cent, val, dnx, rndm);
```

Centrality — Normalisation

FlowContainer

- After receiving the analysis output file
  - Fetch the FlowContainer from the file

```
FlowContainer* fc = (FlowContainer*)f->Get("gfw-tutorial/FlowContainer");
```

  - Set the name for the current ID

```
fc->SetIDName("ChGap");
```

  - If not using bootstrap errors, set propagation of errors

```
fc->SetPropagateErrors(kTRUE);
```

- Callers for getting most common observables

```
TH1D* GetVN2VsMulti(int n = 2, int l_pta = 0)
TH1D* GetVN2VsPt(int n = 2, double min = -1, double max = -1)
TH1D* GetCN4VsMulti(int n = 2, int pti = 0)
TH1D* GetCN4VsPt(int n = 2, double min = -1, double max = -1)
TH1D* GetVN4VsMulti(int n = 2, int pti = 0)
TH1D* GetVN4VsPt(int n = 2, double min = -1, double max = -1)
TH1D* GetVN6VsMulti(int n = 2, int pti = 0)
TH1D* GetVN6VsPt(int n = 2, double min = -1, double max = -1)
TH1D* GetVN8VsMulti(int n = 2, int pti = 0)
TH1D* GetVN8VsPt(int n = 2, double min = -1, double max = -1)
```

Example $\longrightarrow$

p_T-integrated v_2 as function of centrality

```
TH1D* hV22 = fc->GetVN2VsMulti(2);
TH1D* hV22pt_gap = fc->GetVN2VsPt(2,0,4.9);
```

p_T-differential v_2 in 0-5% centrality

- Callers for getting just the correlations

```
TH1D* GetHistCorrXXVsMulti(const char* order, int l_pti = 0)
TH1D* GetHistCorrXXVsPt(const char* order, double lminmulti = -1, double lmaxmulti = -1)
```