```
// vars must not contain the "weight" variable
RooDataSet* RooDKDalitzTools::binnedDataSet(           const  char * name,   const  RooDataSet* dataSet,
                                            const  RooArgSet& vars,      const  char * nameWeight,
                                            const  char * centroidOpt, Bool_t debug)
{
  // iterator over vars
  TIterator* varsIter = vars.createIterator();
  // split vars into real and discrete dimensions, and create iterators
  RooArgSet varsReal,varsCat;
  RooAbsArg* absarg(0);
  varsIter->Reset();
  while (absarg=  dynamic_cast   <RooAbsArg*>(varsIter->Next())) {
    assert(absarg->GetName()!=TString(nameWeight));              // sanity check
    if (dynamic_cast   <RooAbsReal*>(absarg)) varsReal.add(*absarg);              // real dimension
    else   varsCat.add(*absarg);        // discrete dimension
  }
  TIterator* varsRealIter = varsReal.createIterator();
  TIterator* varsCatIter = varsCat.createIterator();
  // Allocate coefficients arrays
  Int_t* idxMultReal =        new  Int_t[varsReal.getSize()];
  Int_t* idxMultCat =       new  Int_t[varsCat.getSize()];
  // Calculate sub-index multipliers for master index and dimension of weight array
  Int_t arrSizeReal(1);
  Int_t n(0),i;
  varsRealIter->Reset();
  RooAbsLValue* absval(0);
  while (absval=  dynamic_cast   <RooAbsLValue*>(varsRealIter->Next())) {
    for  (i=0 ; i<n ; i++) {
     idxMultReal[i] *= absval->numBins() ;
    }
    idxMultReal[n++] = 1 ;
    arrSizeReal *= absval->numBins() ;
  }
  Int_t arrSizeCat(1);
  n=0;
  varsCatIter->Reset();
  while (absval=  dynamic_cast   <RooAbsLValue*>(varsCatIter->Next())) {
    for  (i=0 ; i<n ; i++) {
     idxMultCat[i] *= absval->numBins() ;
    }
    idxMultCat[n++] = 1 ;
    arrSizeCat *= absval->numBins() ;
  }
  cout << "    RooBCPTFitMaster::binnedDataSet: arrSizeReal = " << arrSizeReal << "       arrSizeCat = " << arrSizeCat << endl;
  // Allocate weight and first occurrence arrays
  Double_t* wgt =       new  Double_t[arrSizeReal];
  Double_t* sum =       new  Double_t[arrSizeReal*varsReal.getSize()];
  Bool_t* firstOcc =       new  Bool_t[arrSizeReal];
  // proceed now evaluating weights and building weighted events for each discrete state
  RooDataSet* dataSetBinned(0);
  Int_t iEvt(0);
  if (debug) cout << "     RooBCPTFitMaster::binnedDataSet: operations debug print−out" << endl;
  Int_t entries = dataSet->numEntries();
  Int_t j;
  //   loop over discrete states
  for  (j=0;j<arrSizeCat;j++) {
    // initialize weight and first occurrence arrays
    for  (i=0 ; i<arrSizeReal ; i++) {
     wgt[i] = 0 ;
     firstOcc[i] = kTRUE ;
    }
    // initialize sum
    for  (i=0 ; i<arrSizeReal*varsReal.getSize() ; i++) sum[i] = 0 ;
    //    loop over all events to evaluate weight for current discrete state
   RooArgSet* varsCatLocal =          new  RooArgSet();
   TIterator* varsCatLocalIter = varsCatLocal->createIterator();
   RooArgSet* varsRealLocal =          new  RooArgSet();
   TIterator* varsRealLocalIter = varsRealLocal->createIterator();
    for  (i=0;i<entries;i++) {
       //    get current event
       const  RooArgSet* event = dataSet->get(i);
       //    built vars list and iterator for discrete dimensions for current event
      varsCatLocal->removeAll();
      varsCatIter->Reset();
        while (absarg=  dynamic_cast   <RooAbsArg*>(varsCatIter->Next())) {
        RooAbsArg* item = event->find(absarg->GetName());
        if  (item) varsCatLocal->add(*item);
      }
        //    calculate the index for the weights array corresponding to
        //    to the bin enclosing the current coordinates of the internal argset
      Int_t idxCat(0),k(0);
      varsCatLocalIter->Reset();
        while (absval=   dynamic_cast   <RooAbsLValue*>(varsCatLocalIter->Next())) {
        idxCat += idxMultCat[k++]*absval->getBin();
      }
        //    move to next event if current events does not match discrete state
        if  (idxCat!=j)     continue  ;
        //    built vars list and iterator for real dimensions for current event
      varsRealLocal->removeAll();
      varsRealIter->Reset();
        while (absarg=  dynamic_cast   <RooAbsArg*>(varsRealIter->Next())) {
        RooAbsArg* item = event->find(absarg->GetName());
        if  (item) varsRealLocal->add(*item);
      }
        //    calculate the index for the weights array corresponding to
        //    to the bin enclosing the current coordinates of the internal argset
      Int_t idxReal(0);k=0;
      varsRealLocalIter->Reset();
        while (absval=   dynamic_cast   <RooAbsLValue*>(varsRealLocalIter->Next())) {
        idxReal += idxMultReal[k++]*absval->getBin();
      }
        //    update weight
      wgt[idxReal] += 1.;
        //    update sum
      varsRealLocalIter->Reset();
      k=0;
        while (absval=   dynamic_cast   <RooAbsLValue*>(varsRealLocalIter->Next())) {
        sum[idxReal+k*arrSizeReal] += (         dynamic_cast    <RooAbsReal*>(absval))->getVal();
        k++;
      }
    }
     //    loop over all events to build the weighted event for current discrete state
     for  (i=0;i<entries;i++) {
       //    get current event
       const  RooArgSet* event = dataSet->get(i);
       //    built vars list and iterator for discrete dimensions for current event
      varsCatLocal->removeAll();
      varsCatIter->Reset();
        while (absarg=  dynamic_cast   <RooAbsArg*>(varsCatIter->Next())) {
        RooAbsArg* item = event->find(absarg->GetName());
        if  (item) varsCatLocal->add(*item);
      }
        //    calculate the index for the weights array corresponding to
        //    to the bin enclosing the current coordinates of the internal argset
      Int_t idxCat(0),k(0);
      varsCatLocalIter->Reset();
        while (absval=   dynamic_cast   <RooAbsLValue*>(varsCatLocalIter->Next())) {
        idxCat += idxMultCat[k++]*absval->getBin();
      }
        //    move to next event if current events does not match discrete state
        if  (idxCat!=j)     continue  ;
        //    built vars list and iterator for real dimensions for current event
      varsRealLocal->removeAll();
      varsRealIter->Reset();
        while (absarg=  dynamic_cast   <RooAbsArg*>(varsRealIter->Next())) {
        RooAbsArg* item = event->find(absarg->GetName());
        if  (item) varsRealLocal->addOwned(*(RooAbsArg*)item->Clone());
      }
        //    calculate the index for the weights array corresponding to
        //    to the bin enclosing the current coordinates of the internal argset
      Int_t idxReal(0);k=0;
      varsRealLocalIter->Reset();
        while (absval=   dynamic_cast   <RooAbsLValue*>(varsRealLocalIter->Next())) {
        idxReal += idxMultReal[k++]*absval->getBin();
      }
        //    dump event if weight>0 and if first occurrence
        if  ((wgt[idxReal]>0.)&&(firstOcc[idxReal])) {
        iEvt++;
        varsRealLocalIter->Reset();
        if  (debug) cout << "      iEvt = " << iEvt << endl;
        k=0;
        while (absarg=  dynamic_cast   <RooAbsArg*>(varsRealLocalIter->Next())) {
          if  ( dynamic_cast    <RooAbsReal*>(absarg)) {
            //    set variable to bin center for real dimensions
           RooAbsRealLValue* absrlval =          dynamic_cast    <RooAbsRealLValue*>(absarg);
           Double_t binc=absrlval->getBinning().binCenter(absrlval->getBin());
           Double_t cofg=sum[idxReal+k*arrSizeReal]/wgt[idxReal];
           RooRealVar* rval =       dynamic_cast    <RooRealVar*>(absarg);
            if  (debug) cout << "        " << absarg->GetName() << "         value = " << rval->getVal() << endl;
            if  (TString(centroidOpt)==""      binCenter") rval->setVal(binc);
            else  if  (TString(centroidOpt)==""       centerOfGravity") rval->setVal(cofg);
            else  assert(0);
           k++;
            if  (debug) {
             cout << "      " << absarg->GetName() << "         bin center = " << binc << endl;
             cout << "      " << absarg->GetName() << "         center of gravity = " << cofg << endl;
             cout << "      " << absarg->GetName() << "         centroid = " << rval->getVal() << endl;
           }
          }
        }
        firstOcc[idxReal] = kFALSE;
        if  (debug) cout << "       weight = " << wgt[idxReal] << endl;
        RooRealVar* weight =          new  RooRealVar(nameWeight,nameWeight,1.);
        weight->setVal(wgt[idxReal]*dataSet->weight());                  // update current weight (if any)
        //cout << "i= " << i << " : " << dataSet->weight() << " , " <<  weight->getVal() << endl;
        varsRealLocal->addOwned(*weight);
        RooArgSet varsDump;
        varsDump.add(*varsRealLocal);varsDump.add(*varsCatLocal);
        if  (dataSetBinned==0) dataSetBinned =          new  RooDataSet(name,name,varsDump);
        dataSetBinned->add(varsDump);
      }
    }
     delete  varsCatLocalIter;
    varsCatLocal->removeAll();
     delete  varsCatLocal;
     delete  varsRealLocalIter;
    varsRealLocal->removeAll();
     delete  varsRealLocal;
  }
  //if (dataSetBinned) dataSetBinned->setWeightVar(nameWeight);
  // clean-up
  delete  varsIter;
  delete  varsRealIter;
  delete  varsCatIter;
  delete  idxMultReal;
  delete  idxMultCat;
  delete  wgt;
  delete  sum;
  delete  firstOcc;
  // debug print-out
  if (debug&&dataSetBinned) {
   cout << "    RooBCPTFitMaster::binnedDataSet: binned data sample debug print−out" << endl;
    for  (i=0;i<dataSetBinned->numEntries();i++) {
       const  RooArgSet* event = dataSetBinned->get(i);
       //cout << "iEvt " << i+1 << " with weight " << dataSetBinned->weight() << endl;
     cout << "    iEvt " << i+1 << endl;
     event->Print("      v");
    }
  }
  // return binned data set
  return  dataSetBinned;
}
```