```
import os,sys
import iotbx.mtz

class ReadMtz:
        def __init__(self,mtzfile,merge_option=False):
                self.mtz=mtzfile
                self.isInit=False
                self.isMerge=merge_option
                self.isSymm=False

        def init(self):
                self.m=iotbx.mtz.object(self.mtz)
                if self.isMerge:
                        self.arrays=self.m.as_miller_arrays()
                else:
                        self.arrays=self.m.as_miller_arrays(merge_equivalents=Fa
lse)

        def showSummary(self):
                if self.isInit==False:
                        self.init()
                self.m.show_summary()

        def getIntensityArray(self):
                if self.isInit==False:
                        self.init()

                # yam-god function
                get_I_arrays = lambda x: filter(lambda y: y.is_xray_intensity_ar
ray(), x)

                self.i_related=get_I_arrays(self.arrays)[0]
                return self.i_related

        def getIoverZero(self):
                self.I_ok=self.getIntensityArray()
                self.I_ok=self.I_ok.select(self.I_ok.data()>0.0)

                return self.I_ok

        def getReliableI(self,thresh=2.0):
                self.I_ok=self.getIntensityArray()
                norig=len(self.I_ok.data())
                print "ORIG:%5d"%norig
                self.I_ok=self.I_ok.select(self.I_ok.data()>0.0)
                nsele=len(self.I_ok.data())
                print "I>0.0:%5d"%nsele
                self.I_ok=self.I_ok.select(self.I_ok.data()/self.I_ok.sigmas()>t
hresh)
                nsele=len(self.I_ok.data())
                print "I/sig>%5.2f:%5d"%(thresh,nsele)

                #for d in self.I_ok.data():
                        #print d
                #print norig,nsele

                return self.I_ok

        def getColumn(self,colname):
                if self.isInit==False:
                        self.init()
                obje=filter(lambda a:colname in a.info().labels,self.arrays)[0]
                return obje

        def getSymmOption(self):
                if self.isInit==False:
                        self.init()
                self.ops = [op.inverse().r() for op in self.m.space_group().all_
ops()]
```

```
                self.isSymm=True
                return self.ops

        def getOriginalIndex(self,hkl,isym):
                if self.isSymm!=True:
                        self.getSymmOption()

                # Calculate original index
                sign = -1 if isym%2 == 0 else 1
                ohkl = hkl*self.ops[int((isym-1)/2)]
                ohkl = tuple(map(lambda x:int(x*sign), ohkl))
                return ohkl

        # Take common reflections
        def commonInfo(self,*Is):
                new_Is = []
                Is0 = Is[0]
                for I in Is[1:]:
                        Is0, I = Is0.common_sets(I, assert_is_similar_symmetry=F
alse)
                        new_Is.append(I)
                Is = []
                for I in new_Is:
                        I = I.common_set(Is0, assert_is_similar_symmetry=False)
                        assert len(Is0.data()) == len(I.data())
                        Is.append(I)
                return [Is0,] + Is

if __name__=="__main__":
        filename=sys.argv[1]

        m=ReadMtz(filename)

        #m.showSummary()
        #print m.getSymmOption()
        fracc=m.getColumn("FRACTIONCALC")
        batch=m.getColumn("BATCH")
        m_isym=m.getColumn("M_ISYM")

        fracc,batch,m_isym=m.commonInfo(fracc,batch,m_isym)

        #print m.getMillerOrig()
        #print m.getRealFlex("FRACTIONCALC")
```