# The Whole architecture of the code.

## Root to CSV

文件: Root2CSV.py 类: Root2CSV

主要函数是: run()

Check()

```
oRoot2CSV=Root2CSV(homeRoot=iParser.homeRoot,homeCSV=iParser.homeCSV,channels=iParser.channels,numEven
tPerFile=iParser.numEventPerFile,numProcess=iParser.numProcess)
oRoot2CSV.Run()
oRoot2CSV.Check()
```

# API:

## Class Channel(homeCSV,channels=None):

### Parameters:

homeCSV : home of csv folder
channels=None : channels used. If None, all channles are used.

### Purpose

根据csv文件夹中的子文件夹,给出channels.如果在输入的时候给定了channels,那么,直接使用这些channels.
According to the subfoloders inthe csv folder, channels are given by the names of the subfolders. If the chann are given at the time of input, these channels are used directly.

### File

Channel.py.

### Functions

#### GetChannels()

return channels

#### GetChannels()

return channels
Get channels list from folder or from input

### Example

```
homeCSV='/home/i/IGSI/data/csv'
# channels=['T09_T05_55_D0', 'T09_T06_55_Dch', 'T09_T07_55_Ds', 'T09_T08_55_Lam', 'T09_T09_55_Lamc']
channels=None
oChannel=Channel(homeCSV=homeCSV,channels=channels)
oChannel.GetChannel()
```

## Class Root2CSV(homeRoot,homeCSV,channels=None,numEventPerFile=2048,numProcess=10):

### Parameters:

homeRoot
homeCSV: home of csv folder
channels=None
numEventPerFile=2048 : number of events in one csv file
numProcess=10 : numbers of processor to used

### Purpose

将root文件转变成csv文件,并储存在homeCSV文件夹中.
Convert root files to csv files and store them in the homeCSV folder.

### File

Root2CSV.py

### Functions

#### ChangeChannelName()

No return.
Rename the channels from "T00_DPM_24" to "T00_24_DPM" with format "Method_Energy_Data"

#### LogGetNumEventMC(root)

return numEventMC

Return the number of events according to the log based on the name of root file.

#### RootGetTree(root)

Return trees name from a root file.

### RootGetBranch(self,root,tree):

Return branches from the specific root and tree.

### RootGetData(self,root,tree,branches)

Return data from specific root, tree and branches

### RootRead_Process_Write_Channel(self,channel)

No return. This function reads the root files from one channel and convert to csv files and write down the csv files on disk.
It creates a runLog file in every channel in the csv folders.

In the runLog file, there are:
nnumEventMC_PID
numEventMC
numEventMC_PID/numEventMC And also: 'id'
'branch'
'mean'
'std'
'min'
'max'
'std=0'
if 'std=0' is True, this std of this branch is 0.

### RootRead_Process_Write_Channels(self)

No return.
Do RootRead_Process_Write_Channel(channel) in parallel.

### Run()

No return.
Run function RootRead_Process_Write_Channels(self).

### Check()

Return channelsChecked, channelsNotChecked

channelsChecked: channel list in which channels are convert successfully.
channelsNotChecked: channel list in which channels are conver failed.

Check every channel if all root files are convert to csv from root and store on disk.
a log file checkLog is created in homeCSV main folder.

### DelFolderChannelsNotChecked()

No return.
Delete all the channels which are in the list channelsNotChecked.

#### SetChannel(channels):

No return.
Set channels online.

#### ReRun():

No return.
Run DelFolderChannelsNotChecked() to delete all the channels which are in the list channelsNotChecked.
Set channels online with list channelsNotChecked.
Run and Check.

### Example

```
homeRoot='/home/i/iWork/data/root'
homeCSV='/home/i/iWork/data/csv'
# channels=['T09_T05_55_D0', 'T09_T06_55_Dch', 'T09_T07_55_Ds', 'T09_T08_55_Lam', 'T09_T09_55_Lamc']
channels=None
numEventPerFile=2048
numProcess=10
oRoot2CSV=Root2CSV(homeRoot=homeRoot,
                   homeCSV=homeCSV,
                   channels=channels,
                   numEventPerFile=numEventPerFile,
                   numProcess=numProcess)


# oRoot2CSV.Run()
oRoot2CSV.Check()
oRoot2CSV.ReRun()
```

## Class Branch(homeCSV,channels=None,pidUse=True):

### Parameters:

homeCSV
channels=None
pidUse=True

### Purpose

Select branches

## File

Branch.py

## Functions

### BranchSel(branchAllList)

Return branchSel,branchSelNO,branchAllListRemain
branchSel: +
branchSelNO: -
branchAllListRemain: unknown

Normally, the branchAllList is gotten from columns of csv files.

## Example

```
homeCSV='/home/i/iWork/data/csv'
channels=None
pidUse=True

oBranch=Branch(homeCSV,channels,pidUse)

branchAllList=pd.read_csv('/home/i/iWork/data/csv/T00_DPM_24/0000000_0002048.csv',nrows=0).columns.tol
ist()
print(branchAllList)
branchSel,branchSelNO,branchAllListRemain=oBranch.BranchSel(branchAllList)
print(branchSel)
print(branchSelNO)
print(branchAllListRemain)
```

# Class Data(homeCSV,channels=None,ratioSetTrain=0.7,ratioSetTest=0.3,ratioSetValid=0.):

## Parameters:

homeCSV
channels=None
ratioSetTrain=0.7
ratioSetTest=0.3

ratioSetValid=0.

## Purpose

This Class is mainly used in DateSet...

Split csvList to listCSV4Test ,listCSV4Valid and listCSV4Train

'listCSV4Test' 'listCSV4Valid' 'listCSV4Train'

'labels' 'labels' 'numClasses' 'branch4Train'

## File

Data.py

## Functions

### GetListCSV4Test():

```
return self.listCSV4Test
```

### GetListCSV4Valid():

```
return self.listCSV4Valid
```

### GetListCSV4Train():

```
return self.listCSV4Train
```

### GetLabels():

```
return self.labels
```

### GetNumClasses():

```
return self.numClasses
```

### GetBranch4Train():

```
return self.branch4Train
```

### branchTrainLog()

No return.

Write log branchTrainLog.

Content:
branchAll: 401
beame beamm beamp beamphi ...

branch4Train (Train !!!): 232
beamp esapl escir esfw1 ...

branch4TrainNOUSE: 169
beame beamm beamphi beampt ...

branchWhilelist NOT exist...: 7
d d0 ddd ...

branchBlacklist NOT exist...: 0

### _Branch4Train()

No return.

Read log branchLog to get branchAll and branchAll.
branchWhitelistHand and branchBlacklistHand are read here.
branchWhitelistHand is read first, and branchBlacklistHand read later, which means that the priority of branchBlacklistHand is higher than it of branchWhitelistHand.

### __Labels()

No Return.
Arrange all the channels with labels. DPM:0
sig: +=1

### _Labels()

No Return.
Set chanels with labels...

### Label(iChannel)

return label
Return a label given the iChannel.

#### _Data()

No Return.
Get the ratioEvent of all Channels.

#### _DataSplit()

No Return.
Split csv list to lists of Test, Valid and Train.

#### __GetList_While_Black(iListFile)

No return.
Read branchWhitelistHand and branchBlacklistHand to get list.

### Example

```
homeCSV='/home/i/iWork/data/csv'

oData=Data(homeCSV,channels=None,ratioSetTrain=0.7,ratioSetTest=0.3,ratioSetValid=0.)


print('listCSV4Test')
print(oData.GetListCSV4Test())
print('listCSV4Valid')
print(oData.GetListCSV4Valid())
print('listCSV4Train')
print(oData.GetListCSV4Train())

print('labels')
print(oData.GetLabels())

print('numClasses')
print(oData.GetNumClasses())

print('branch4Train')
print(oData.GetBranch4Train())
```

## Class DataSet(homeCSV,channels,setTrain=1,setTest=0,setValid=0,ratioSetTrain=0.7,ratioSetTest=0.3,ratioSetValid=0.,readMethod=0):

## Parameters:

homeCSV
channels
setTrain=1
setTest=0
setValid=0
ratioSetTrain=0.7
ratioSetTest=0.3
ratioSetValid=0.
readMethod=0

## Purpose

DataSet for NN.

## File

DataSet.py

## Functions

### Label2Data()

No Return
label2CSV
Writedown a log named label2CSVLog

### __len__()

return number of iterm

### __getitem__()

return data=(iData,iLabel)

### ReadCSV()

No Return. read in data and label from csv parallelly
Create one processor based on every class.
Call function ReadCSV_OneFile(iClass)

### ReadCSV_OneFile(iClass)

return (iData,iLabel) for iClass.

1. Given iClass as label :→ iLabel
2. Get csv list accoring to iClass

3. Choose one csv randomly form csv list
4. read in one csv :-> iData
5. return (iData,iLabel)

### Example

```
```

## Class Channel(homeRoot,channels=None):

### Parameters:

### Purpose

### File

### Functions

GetChannels()

### Example

```
```

## Class Channel(homeRoot,channels=None):

### Parameters:

### Purpose

### File

### Functions

GetChannels()

### Example

```
```