Identification et caractérisation d'objets

Brève présentation de l'outil - Épisode 2

Najda Villefranque

CNRM

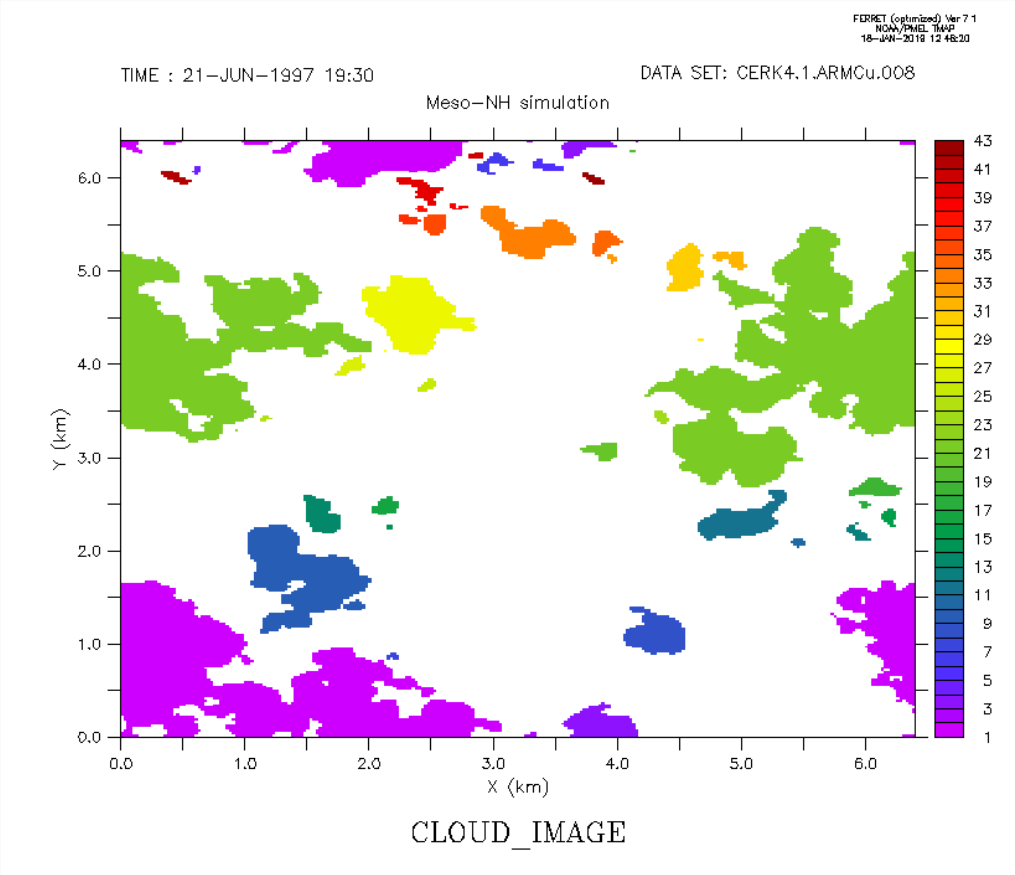
Première étape : identification

- Identifier des structures cohérentes
- Dans des champs 2D, 3D ou 4D
- À partir d'un masque binaire défini par l'utilisateur
- Les cellules du masque = 1 si $\in$ objet, 0 sinon

Structures cohérentes

Amas de cellules "1" contiguës

Outil à compléter, améliorer !



Implémentation

- Un premier module Python : identification_methods.py
  - ncfile : un fichier netCDF qui contient les champs à analyser
  - listVarsNames : la liste des noms des variables, dans le fichier ncfile, dont vous avez besoin pour définir vos objets
  - clouds : une fonction Python qui reçoit en entrée les variables demandées dans listVarsNames, et qui doit retourner un ”masque”

```
import os,sys
sys.path.append('../src/')
from identification_methods import identify
import numpy as np

ncdfFile = "../../data/CERK4.1.ARMCu.008.nc"

listVarNames = ['RCT'] # list of relevant variables.

def cloudMask(dictVars) :
    rc = dictVars['RCT']
    mask=rc*0.   # same shape as RCT
    mask[rc>0]=1 # cell is in cloud if liquid water mixing ratio is positive
    return mask

# To read more info on the function identify, uncomment the following line
#help(identify)
objects, tmp = identify(ncdfFile, listVarNames, cloudMask)
```

Tracking temporel

Toujours un amas de cellules contiguës...

Deux objets distincts à un pas de temps qui se réunissent au pas de temps suivant = le même objet depuis le début
Une partie d'un objet qui se détache reste le même objet

Discussions DEPHY, autres outils de tracking dans la communauté
e.g. Thibaut Dauhut, Nicolas Rochetin

$\Rightarrow$ des développements en commun... à réfléchir !

Seconde étape : caractérisation

- Caractériser les objets identifiés
- Pour chaque ”caractéristique” et pour chaque ”objet” : mean, min, max, std, median, 5th, 95th percentiles
- Ecrits dans un nouveau fichier netCDF
- Un champ par caractéristique, selon les dims (par défaut) : time, nombre d'objets, statistique

```
→  ARMCu ncdump -h CERK4.1.ARMCu.008_charac.nc
netcdf CERK4.1.ARMCu.008_charac {
dimensions:
        stats = 7 ;
        object = 67 ;
        time = 1 ;
variables:
        double objects(time, object, stats) ;
        double RCT(time, object, stats) ;
        double THT(time, object, stats) ;
        double PABST(time, object, stats) ;
        double WT(time, object, stats) ;
```

Implémentation

- Un deuxième module Python : characterize_methods.py
  - ncfile : un fichier netCDF qui contient au minimum le champ ”objets”
  - listVarsNames : la liste des noms des variables, dans le fichier ncfile, dont vous voulez calculer les caractéristiques

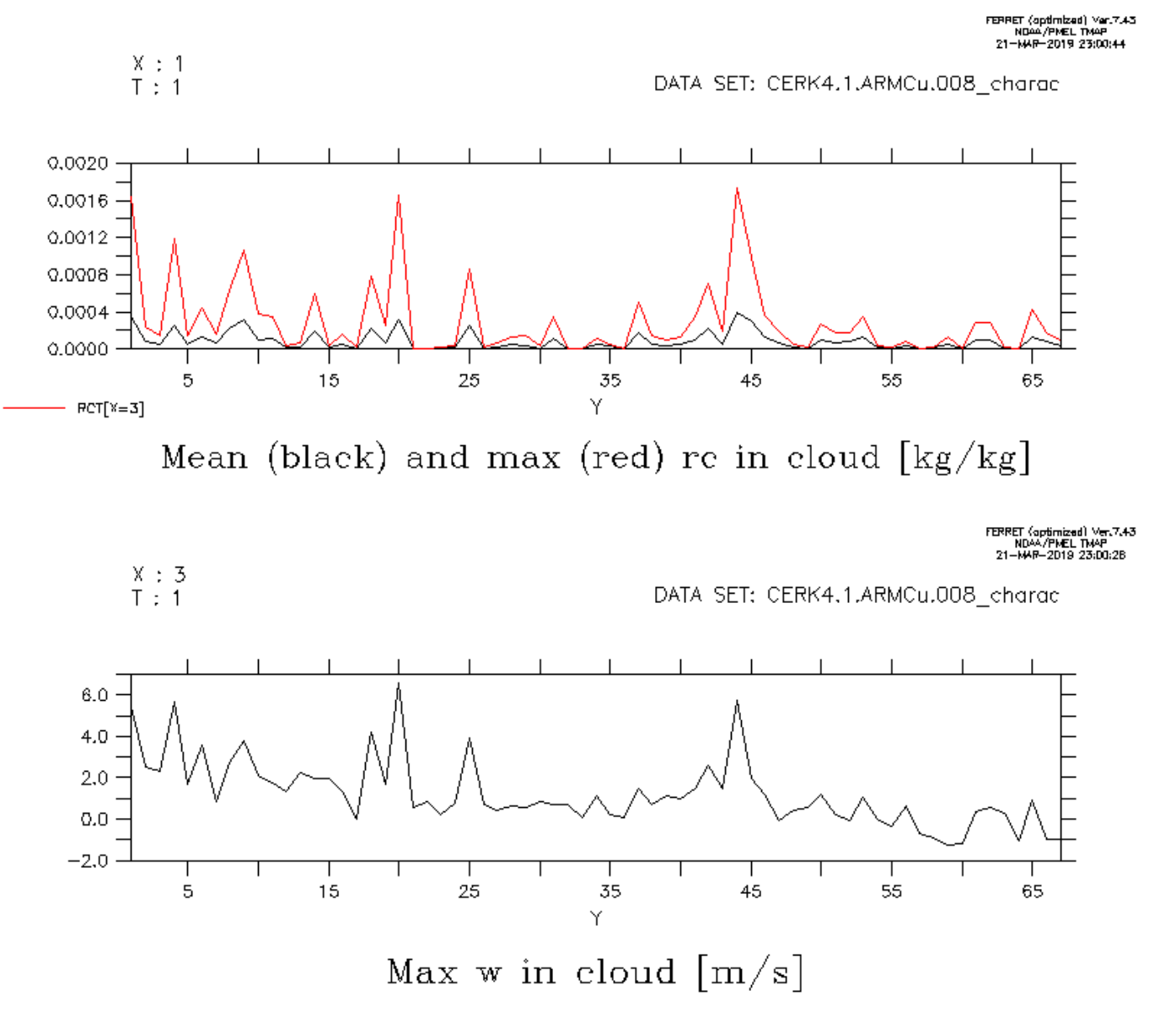
```
import os,sys
sys.path.append('../src/')
from characterization_methods import characterize
import numpy as np

ncfile = "../../data/CERK4.1.ARMCu.008.nc"

listVarsNames = ['RCT','THT','PABST','WT']

characs = characterize(ncfile, listVarsNames)
```

Exemple de résultats



Ajouter des caractéristiques ”utilisateur” : userVars

Un champ calculé une seule fois sur tout le domaine
Typiquement, combinaison des champs dispos dans le netcdf

```
ncfile = "../../data/CERK4.1.ARMCu.008.nc"

listVarsNames = ['RCT','THT','PABST','WT']
ra = 287.
cp = 1004.
ra_over_cp = ra/cp
rho_w = 1000.
r_eff = 10.0e-9
def ke(dictVars) :
  rc = dictVars['RCT']; th = dictVars['THT']; pr = dictVars['PABST']
  p0 = pr[:,0,:,:]
  den = ra*th*p0**(-ra_over_cp)
  rho = pr**(1.-ra_over_cp)/den
  ke = 3./2.*rc*rho/(r_eff*rho_w)
  return ke

characs = characterize(ncfile, listVarsNames,
          dictFuncUserVars={'extinction_coefficient km^-1':ke})
```

Ajouter des caractéristiques ”utilisateur” : calcChar

Une fonction qui sera appelée pour chaque objet avec arg indexes :
liste des coordonnées des mailles qui $\in$ à l'objet courant
Par exemple pour l'objet 3D numéro k de taille N mailles :
indexes = $\left[\left[iz_1^k, iz_2^k, iz_3^k, ..iz_N^k\right], \left[iy_1^k, iy_2^k, iy_3^k, ..iy_N^k\right], \left[ix_1^k, ix_2^k, ix_3^k, ..ix_N^k\right] \right]$

```
ncfile = "../../data/CERK4.1.ARMCu.008.nc"

listVarsNames = ['RCT','THT','PABST','WT',
                 'S_N_direction','W_E_direction','VLEV']

def volume(dictVars,indexes) :
  Xvec = dictVars["W_E_direction"]; dx = Xvec[1]-Xvec[0]
  Yvec = dictVars["S_N_direction"]; dy = Yvec[1]-Yvec[0]
  Zvec = dictVars["VLEV"][:,0,0]  ; dz = Zvec[1]-Zvec[0]
  nbcells = len(indexes[0])
  volume_cloud=nbcells*dx*dy*dz*1e9
  return volume_cloud

characs = characterize(ncfile, listVarsNames,
          dictFuncCalcChar={'volume m^3':volume})
```

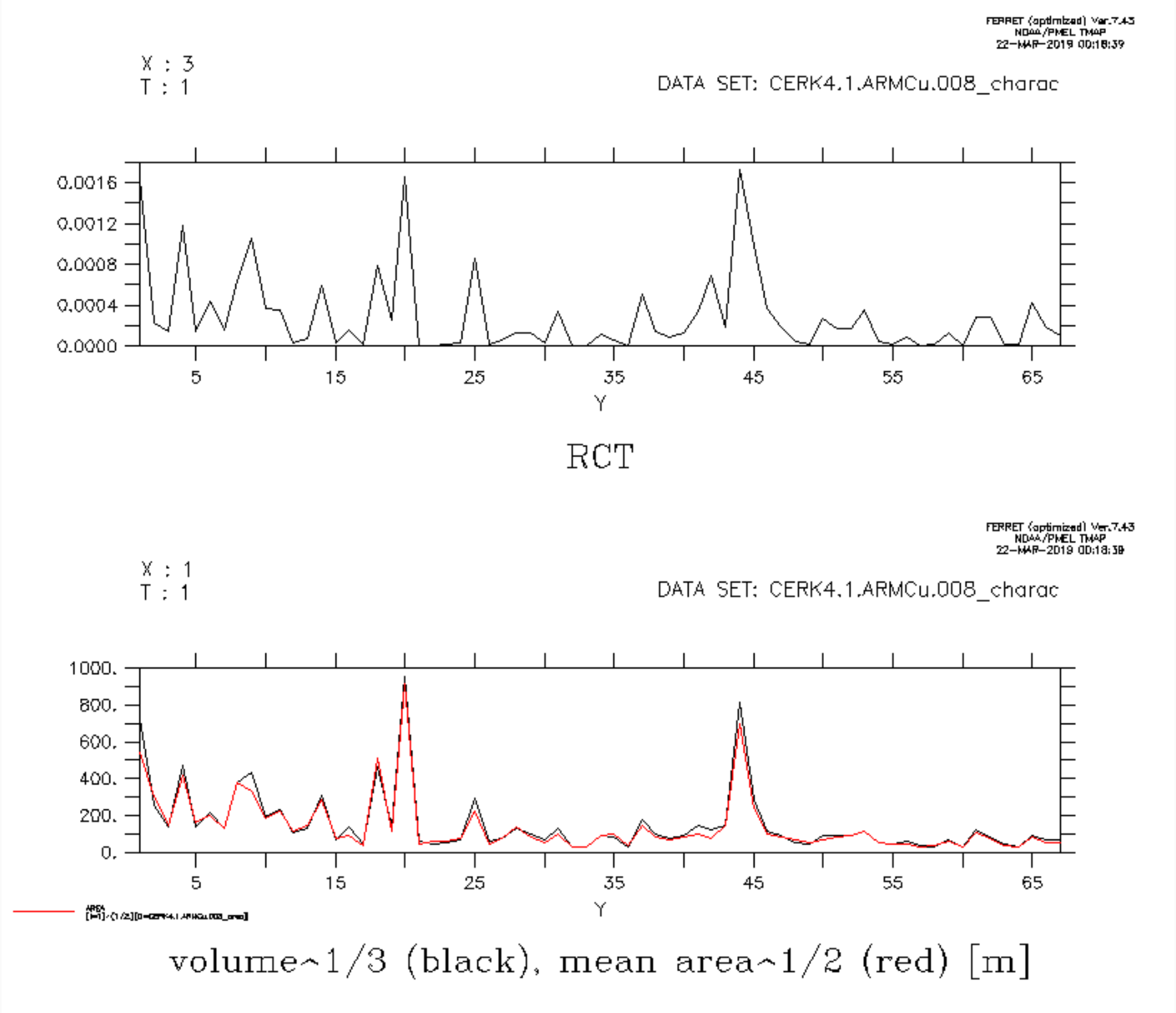
Ajouter des caractéristiques ”utilisateur” : calcChar

Une fonction qui sera appelée pour chaque objet avec arg indexes :
liste des coordonnées des mailles qui $\in$ à l'objet courant
Par exemple pour l'objet 3D numéro k de taille N mailles :
indexes = $\left[\left[iz_1^k, iz_2^k, iz_3^k, ..iz_N^k\right], \left[iy_1^k, iy_2^k, iy_3^k, ..iy_N^k\right], \left[ix_1^k, ix_2^k, ix_3^k, ..ix_N^k\right] \right]$

```
def area(dictVars,indexes) :
  Xvec = dictVars["W_E_direction"]; dx = Xvec[1]-Xvec[0]
  Yvec = dictVars["S_N_direction"]; dy = Yvec[1]-Yvec[0]
  uniqueCloudLevs = np.unique(indexes[0])
  allCloudLevs = indexes[0]
  nbCells_atLevel = [len(allCloudLevs[allCloudLevs==lev])
                     for lev in uniqueCloudLevs]
  return 1.e6*dx*dy*np.array(nbCells_atLevel)

characs = characterize(ncfile, listVarsNames,
          dictFuncCalcChar={'volume m^3':volume,'area m^2':area})
```

Exemple de résultats

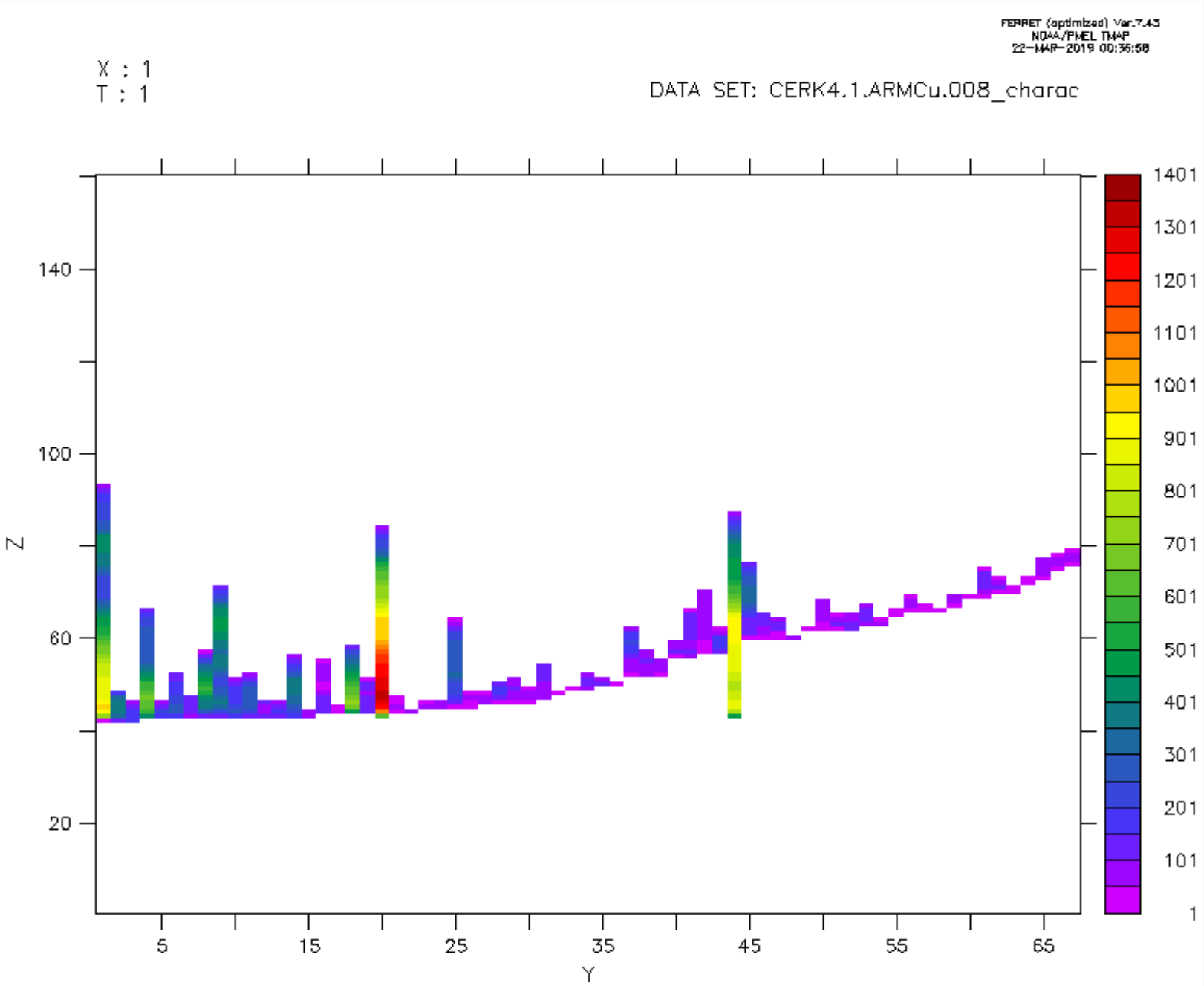


Profils verticaux de caractéristiques : option dimChars

Pour spécifier les dims selon lesquelles seront définies les caracs
Boucle sur les objets et sur les dimensions conservées.
Ici on garde le temps et les niveaux verticaux :
indexes = $\left[\left[iy_1^k, iy_2^k, iy_3^k, ..iy_N^k\right], \left[ix_1^k, ix_2^k, ix_3^k, ..ix_N^k\right] \right]$

```
def area(dictVars,indexes) :
  Xvec = dictVars["W_E_direction"]; dx = Xvec[1]-Xvec[0]
  Yvec = dictVars["S_N_direction"]; dy = Yvec[1]-Yvec[0]
  nbCloudCells = len(indexes[0])
  return dx*dy*nbCloudCells

characs = characterize(ncfile, listVarsNames,
          dictFuncCalcChar={'area km^2':area},
          dimChars={0:"time",1:"vertical_levels"})
```



area~1/2 as a function of cloud and height [m]

Importer les fonctions identify et characterize et le module numpy

```
import os,sys
sys.path.append('path/to/scripts_objects/src/')
from identification_methods import identify
from charaterization_methods import characterize
import numpy as np
```

Codes sources : `https://gitlab.com/tropics/objects/`

```
> git clone https://gitlab.com/tropics/objects/
```

Scripts exemples dans objects/examples/