Documentation of "02-split.py"

A Documentation from Jessica Ahring
j.ahring@fz-juelich.de

Forschungszentrum Jülich
JSC
ESDE
DeepRain
Jülich
February 12, 2020

This is a documentation about the second preprocessing script for the *meteodatacube*, namely "02-split.py". This needs data what was already preprocessed by "01-preproc.py". It splits the data into the timesteps what is needed for importing the data.

name: 02-split.py
location: /mnt/rasdaman/DeepRain/playground
format: python 3
execution: 02-prepro.py *working_directory*

0. Step - Checking

```
# is the source as a parameter given this will be set otherwise the
  script will be stopped
if len(sys.argv) > 1:
  if os.path.isdir(sys.argv[1]):
    sourcepath=os.path.abspath(sys.argv[1])
  else:
    print("Path did not exist\n")
    sys.exit()
else:
  print("No path is given!\n")
  sys.exit()

# change to working directory
os.chdir(sourcepath)

# check if data is splitted in variables
if not glob.glob(sourcepath+"/preproc-cde*"):
  print ("No files found.")
  print ("Data needs to be preprocessed by 01-preproc.py")
  sys.exit()

index=0
summe=len(glob.glob(sourcepath+"/preproc-cde*"))
for data in glob.glob(sourcepath+"/preproc-cde*"):
  index+=1
  print(str(index) +" / " + str(summe) + " ---> " + str(index/float(
  summe)) + "%")

```

As we need data to work with, we first have to check if the given `working_directory` exists. If not the script will return an error message and quit. If it exists, the script will change into this directory since we have to work with the data.
Then the script checks if the data in the directory was already preprocessed by "01-preproc.py" by checking the filename. If not an error message will tell the user what to do and stops the script.
If everything is fine the script can start the actual work by working with every datafile. (Here an output tells the user how much data is left.)

1. Step - Split timesteps

```
1  print(data)
2  args="cdo -s splitsel,1 " + data + " output" + "> /dev/null 2>&1"
3  return_code=subprocess.call(args, shell=True)
4  if(return_code!=0):
5    print("Failed splitting '"+out_file+"'")
6    sys.exit(1)
7  print("Split timpesteps ...ok")
8
```

Printing the data is just a debugging part and will be deleted later. In this part of the script the data will be splitted into the 8 available timesteps in the datafile by using `cdo`. If this fails the script gives an corresponding output and will stop, otherwise it tells the user the current status.

2. Step - Rename data

Since the output of the cdo command are eight files named "output00000X" (with $X \in \{1, ..., 8\}$) this has to be renamed to a meaningful term.

```
1  for var in glob.glob("out*"):
2    time=datetime.strptime(subprocess.check_output(["cdo","-s","-
     showtimestamp",var]).decode(sys.stdout.encoding).strip(), "%Y-%m-%dT%H
     :%M:%S")
3    store=os.path.basename(data)
4    store=store.split("-")[1]                    # cdeYYYYMMDD.FF.mEE.nc
5    store=store.split(".")                       # [cdeYYYYMMDD][FF][mEE][nc]
6    forecast_hour=store[-3]                      # FF
7    ensemble=store[-2]                           # mEE
8    time=time-timedelta(hours=int(forecast_hour))
9    args="ncap2 -s 'time+="+forecast_hour+"-time' -s 'time@units=\"hours
     since "+time.strftime("%Y-%m-%d %H:%M:%S")+"\"' "+var+" time:"+time.
     strftime("%Y%m%d-%H")+"."+forecast_hour+"."+ensemble+".nc"
10   return_code=subprocess.call(args, shell=True)
11   if return_code != 0:
12     print("Time change failed")
13     sys.exit()
14   os.remove(var)
15   print("Time change + mv ...ok")
16
```

So for each "output00000X" file we create a new meaningful name and change the timevalue in the datafile as we need this later importing the datafile.

line 2 extract the stored timevalue in "output00000X"

line 4 get the basename cdeYYYYMMDD.FF.mEE.nc

line 5 extract different parts of the basename

line 6 store the forecasthour (out of the name)

line 7 store the ensemble member (out of the name)

line 8 get the "starttime" by subtracting the forecasthour from the "actual time" (extracted from the timevalue in "output00000X")

line 9 change the metadata in the file such that the time is the forecasthour and the unit is "hours since <starttime>"
Also change the filename from "output00000X" to "<starttime>.<forecast_hour>.<ensemble>.nc"

The "difficult" filename is chosen because we have to use "starttime", "forecast_hour" and "ensemble" in the import script and in this format the values are easy accessible. If this does not work we get an corresponding error message and the script stops. Otherwise the old file "output00000X" will be deleted and a message for the user is printed. We had to change the metadata for importing later. With the unit "hours since..." and the value "forecasthour" the resulting value is the "actual time".

3. Step - Delete netCDF

```
1      os.remove(data)
2      print("Remove            ...ok")
3    print("Execution completed")
4
```

At the end the old netCDF-file is deleted. This is the one which was splitted in the eight "output00000X" files. This is not needed anymore.

Summary

- Checking
  - for all datafiles in working directory
    - split in timesteps
    - for all "output00000X"-files
      - rename file and metadata
    - remove datafile

An overview over the directory-structure: