# Documentation of "01-prepro.py"

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

Forschungszentrum Jülich
JSC
ESDE
DeepRain
Jülich
February 14, 2020

This is a documentation about one of the preprocessing scripts for the *meteodatacube*, namely "01-prepro.py". It is the first script what get started to import data. It copies the data from the source directory to a working directory and splits it into several datafiles as described below.

name: 01-prepro.py
location: /mnt/rasdaman/DeepRain/playground
format: python 3
execution: 01-prepro.py *source_ directory*

## constants

```
script_dir=os.getcwd()
rot_grid=script_dir+"/gridneu.dat"
destination=script_dir+"/preproc"
ingest_dir=script_dir+"/ingest"
# change for parallel execution
split_dir=script_dir+"/split"
filter_file=script_dir+"/split_filter.txt"

```

`script_dir` in this case would be `/mnt/rasdaman/DeepRain/playground`. This is the operating directory. Because of the format of the data from DWD we have to convert its axis from "rotated pole grid" to an "equidistant grid". Therefore we need a mapping file called `gridneu.dat` which can be found in the actual directory. For importing the data to rasdaman an `ingest-file` is needed. All available ingest-files are stored in the `ingest` directory. Our working directory will be the subdirectory `preproc` which we have to use since we want to change the data and delete some afterwards. This should not be done with the actual data. For splitting the data in the different available variables a `split_dir` is needed. This is also a subdirectory of the actual directory. For this splitting the `filter_file` is needed.

## functions

```
def cleanup():
  shutil.rmtree(split_dir)
  os.remove(filter_file)
  print("Cleaned up")

```

Because of the subdirectory structure created during the splitting process we need to clean up our actual directory if an error occurs or the execution is done. Therefore the `cleanup()`-function is written.

## 0. Step - Checking

Since we have a lot of (constant-)paths that are needed for execution this need to be checked before. If something is missing the execution should stop.

```
if len(sys.argv) > 1:
  if os.path.isdir(sys.argv[1]):
    sourcepath=os.path.abspath(sys.argv[1])
    print(sourcepath+"\n")
  else:
    print("Path does not exist\n")
    sys.exit()
else:
  print("No path is given!\n")
  sys.exit()

#split_dir=sourcepath+"/split" # for parallel execution this needs to
  be in the sourcedirectories, otherwise things will be overwritten
#filter_file=sourcepath+"/split_filter.txt"

# check rot_grid data
if os.path.isfile(rot_grid)==False:
  print("gridneu.dat is not given or not readable\n")
  sys.exit()

# check ingest_dir
if os.path.isdir(ingest_dir):
  ingest_dir=os.path.abspath(ingest_dir)
else:
  print("Ingestion directory does not exist")
  sys.exit()

# change to working directory
  os.makedirs(destination, exist_ok=True)
  os.chdir(destination)

```

So first of all we check if the `source_path` is given. Otherwise the script has no data to work with and the script stops its execution with a corresponding error message.
The same happens if the `gridneu.dat` is not given or the `ingest-directory` does not exist.
After all this things are checked, the destination (working) directory will be created (if it does not exist).

## 1. Step - preparation

```
# extract all possible ingestions (variables)
ingestions=[]
for var in glob.glob(ingest_dir+"/ingest_*.json.template"):
  parts=os.path.basename(var)
  parts=parts.split(".")
  parts=parts[0].split("-")
  ingestions.append(parts[1])
if not ingestions:
  print("No ingestion scripts given.")
  sys.exit(1)
else:
  print("Split data into:")
  for var in ingestions:
    print(var)

# if the split directory exists, it will be deleted and newly created
if os.path.isdir(split_dir):
  shutil.rmtree(split_dir)
os.mkdir(split_dir)

# write filter_file for split-step
if os.path.isfile(filter_file)==True:
  os.remove(filter_file)
f=open(filter_file, "w")
f.write("write \"" + split_dir + "/[shortName].grib[editionNumber]\";")
f.close()

if not glob.glob(sourcepath+"/cde*"):
  print("No files found.")
  print("Files need the form 'cde*'")
  sys.exit()

# every file in the directory will be processed
for input_file in glob.glob(sourcepath+"/cde*"):
  print(os.path.basename(input_file))
  if(int(os.path.basename(input_file).split(".")[-3])>24):
    print("skipped")
    continue

```

In this step the script gets prepared. In every datafile from the DWD up to 74 variables are stored. Since we do not want to store all we extract the ones we need. Therefore the script checks the `ingest`-directory for possible ingredient-files. They have the format "ingest-*.json.template" with * is the identifier for the variable. In the first part of this step we extract all variable-identifier we need to compute later. They are stored in a list. If this is empty no ingest-files where found and we can not import something. So the script stops its execution with an corresponding error message. Otherwise the detected variables are printed out for debugging reasons.
Afterwards the split-directory gets deleted (if it exists) and afterwards newly created.

Also the filter_file gets deleted and newly created.
After this we check if some data is found in the source-directory. If not we can stop the execution.

Now all preparations are done and the script can start the main task of processing all datafiles it founds. Therefore we iterate over all files in the source-directory. To avoid unnecessary computations we check if the datafile has a forecasthour $>$ 24, in this case we skip it, because we would not use it anyway.

## 2. Step - split by filter

As earlier mentioned we have to split each file into its variables but just want to compute the ones given in the ingest-directory. Therefore we use the filter_file for splitting the data with "grib_filter" and store the splitted datafiles in the split-directory. If this does not work the script stops its execution with an error message after it cleaned up, if it worked the user gets a positive return value.

```
return_code = subprocess.call(["grib_filter",filter_file,input_file])
if return_code != 0:
  print("Failed splitting " + input_file)
  cleanup()
  sys.exit(1)
else:
  print("Splitting             ...ok")

```

## 3. Step - remove unneeded split data

After splitting the data in 74 variables we remove the ones we do not need (what is not given in the ingest-directory).

```
for dat in os.listdir(split_dir):
  name=dat.split(".")[0]
  if name not in ingestions:
    os.remove(glob.glob(split_dir+"/"+dat)[0])

```

## 4. Step - Grib2 -> NetCDF

The last step in this preprocessing script is the conversion from Grib2 (or Grib1) to NetCDF data, as we need this format in the next preprocessing step. So the script checks for each file that was filtered and not deleted (so this should be processed in the next step), if the needed directory in the working directory exists and creates it otherwise (all temperature files will be stored in "destination_dir/t/", such that every directory stores all data of one variable). Then the datafile will be converted to netCDF and stored in the working-directory. If an error occurs that gets printed and after a cleanup the script stops.

```
for file_path in os.listdir(split_dir):
    out_file=os.path.basename(input_file)
    extension=os.path.splitext(out_file)
    if extension[1] != ".m*":
        out_file=extension[0]
    out_file_path=destination+"/"+os.path.splitext(file_path)[0]
    out_file=out_file_path+"/preproc-"+out_file+".nc"
    os.makedirs(out_file_path, exist_ok=True)
    in_file=glob.glob(split_dir+"/"+file_path)[0]
    args="cdo -O -s -f nc4 setgrid,'"+rot_grid+"' '"+in_file+"' '"+
   out_file+"'"+"> /dev/null 2>&1"
    return_code=subprocess.call(args, shell=True)
    if return_code!=0:
        print("Failed converting '"+file_path+"' to NetCDF ('"+out_file+"')"
   )
        cleanup()
        sys.exit(1)
    print("Grib2 -> NetCDF      ...ok")

```

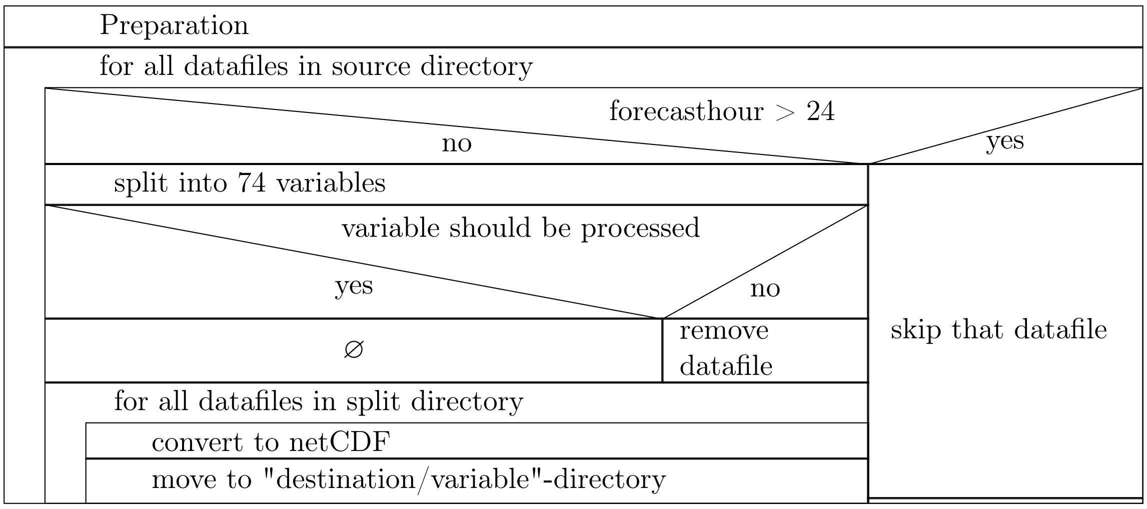
## End

After this is done for every datafile in the `source`-directory, the script starts a cleanup and prints "Execution completed" before it ends.

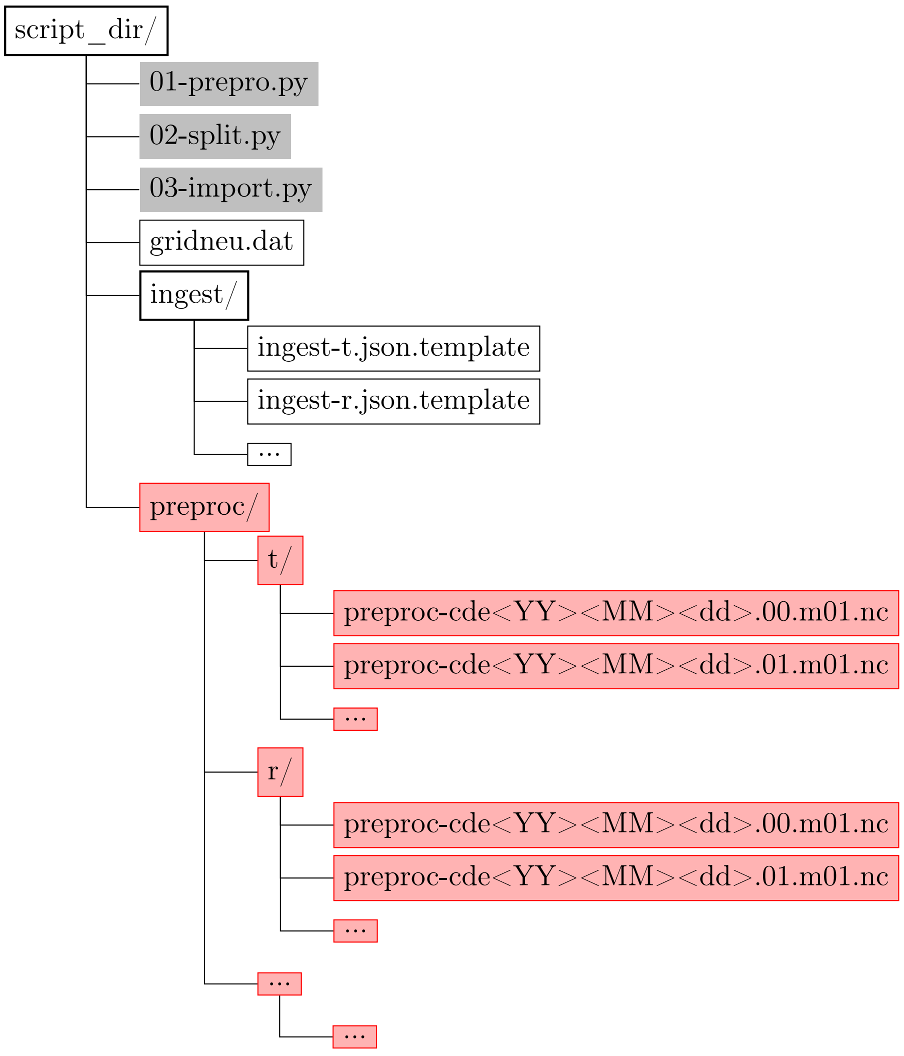
```
cleanup()
print("Execution completed")

```

## Summary


Preparation
for all datafiles in source directory
forecasthour > 24
no
yes
split into 74 variables
variable should be processed
yes
no
∅
remove datafile
for all datafiles in split directory
convert to netCDF
move to "destination/variable"-directory
skip that datafile

An overview over the directory-structure:


script_dir/
01-prepro.py
02-split.py
03-import.py
gridneu.dat
ingest/
ingest-t.json.template
ingest-r.json.template
…
preproc/
t/
preproc-cde<YY><MM><dd>.00.m01.nc
preproc-cde<YY><MM><dd>.01.m01.nc
…
r/
preproc-cde<YY><MM><dd>.00.m01.nc
preproc-cde<YY><MM><dd>.01.m01.nc
…
…
…