Exploring_drought_Kenya

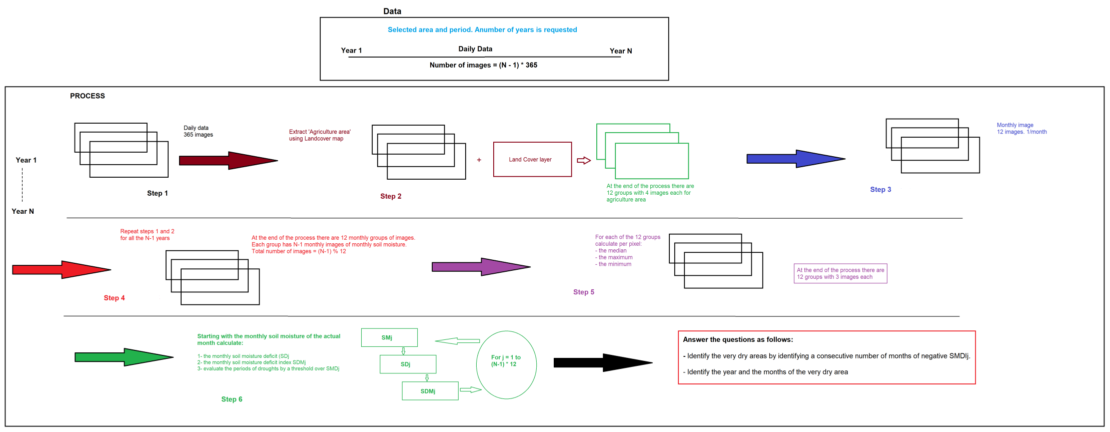
September 8, 2021

1 Exploring the Drought in Kenya using Soil Moisture

License:

2 1. Introduction to the exercise

Kenya is a drought-prone country, mainly due to its peculiar environmental conditions. While agriculture supports up to 75% of Kenya's population and generates almost all food requirements, drought is a major constraint on rain-fed agricultural production, particularly in arid and semi-arid Kenya. In the last century, Kenya has faced almost 29 droughts, some of them in the last decade. The frequency and severity of droughts in Kenya seems to be increasing over time. As a result, crop failures and livestock deaths are leading to severe food shortages in Kenya and food insecurity. The Climate Change Initiative hosts a soil moisture dataset that can be used to generate an index of draught severity called Drought Index. In this exercise, we apply the soil moisture data to generate this Drought Indicator for Kenya.


Data
Selected area and period. Anumber of years is requested
Year 1
Daily Data
Year N
Number of images = (N - 1) * 365
PROCESS
Year 1
Year N
Daily data
365 images
Step 1
Extract 'Agriculture area'
using Landcover map
Step 2
+
Land Cover layer
At the end of the process there are 12 groups with 4 images each for agriculture area
Step 3
Monthly image
12 images. 1/month
Repeat steps 1 and 2
for all the N-1 years
Step 4
At the end of the process there are 12 monthly groups of images.
Each group has N-1 monthly images of monthly soil moisture.
Total number of images = (N-1) % 12
For each of the 12 groups
calculate per pixel:
- the median
- the maximum
- the minimum
Step 5
At the end of the process there are 12 groups with 3 images each
Starting with the monthly soil moisture of the actual month calculate:
1- the monthly soil moisture deficit (SDj
2- the monthly soil moisture deficit index SDMj
3- evaluate the periods of droughts by a threshold over SMDj
Step 6
SMj
SDj
SDMj
For j = 1 to
(N-1) * 12
Answer the questions as follows:
- Identify the very dry areas by identifying a consecutive number of months of negative SMDIj.
- Identify the year and the months of the very dry area

The input data is daily images of soil moisture (SM, m3/m3) for Kenya and for a number of years (2000.01.01-2020.01.01). There are several approaches to obtaining periods of droughts following information on soil moisture. In this xcercise, we follow the Soil Moisture Deficit Index to address drought in Kenya.

1. Download the daily SM images for the area of interest and for the period considered
2. Extract the agricultural area using landcover map
3. Calculate the average SM per month
4. Group the monthly SM by month (all monthly SM of January, February, etc.) for the time span concerned

5. Calculate the median, maximum, and minimum values per pixel of the long term monthly images. Three images per month
6. With the actual image of the SM of a given month (SWj), calculate the monthly soil deficit of the month (SDj)
7. With (SDj) and the Soil Moisture Deficit Index of the month (j-1) (SMDIj-1), calculate the SMDIj. Note: SMDI1= SD1/50 for initial month
8. -4 <= SMDIj <= 4 indicating from very dry to very wet conditions
9. Identify the very dry areas by identifying a consecutive number of months of negative SMDIj.
10. Identify the year and the months of the very dry area in Kenya

2.1 1.1. Exercise objectives

In this exercise you will learn to use Cate in order to: 1. Search for soil moisture data from the ESA CCI Open data portal 2. Apply the soil moisture data to generate information required for the drought model 3. Create a drought index based on soil moisture dataset for Kenya 4. Identify periods and regions of drought over Kenya

2.2 Getting started with jupyter notebooks and the cate webui

Todo: add hyperlink to general instructions on setting up Cate

2.3 1.2. Are you ready to go?

Please follow the steps below, and run all the code cells step by step Here are a few technical reminders: * `tabs` play an important role in python and remove the need for brackets. If you incorrectly align you code, it will not run and you'll get alignment errors. * you can execute **code cells**, by selecting the cell and pressing either the *play* button or pressing `Ctrl+Enter` * Indexing in python is 0-based (i.e. the first entry is denoted by 0) * You're encouraged to try out different settings

2.4 1.3. Initializing some modules

[1]:
```
# Load some parts of the cate module so we can communicate with its datastore

from cate.core.ds import DATA_STORE_POOL
import cate.ops as ops
from cate.ops.io import open_dataset

from datetime import datetime

from shapely import wkt # needed later when we apply regional constraints
import xarray as xr # Xarray allows flexible indexing of multi-dimensional␣
↪arrays. Xarray Datasets and DataArrays underly many routines in cate
import numpy as np

import logging
# There are some irrelevant warnings later which we want to suppressed
```

```
logging.captureWarnings(True) #although when developing code it's usually␣
↪better to keep this disabled)

import warnings
warnings.filterwarnings('ignore')

# plotting tools
import matplotlib.pyplot as plt
import cartopy.crs as ccrs #to work with geographical projections

# When we plot, we would like the figures to appear in the notebook itself
%matplotlib inline

# We need this temporary hack to work around (https://stackoverflow.com/
↪questions/56154176/
↪runtimeerror-asyncio-run-cannot-be-called-from-a-running-event-loop)
#import nest_asyncio
#nest_asyncio.apply()

# the following is needed to run Cate in a Jupyter Notebook
from xcube.util.ipython import enable_asyncio
enable_asyncio()

import cartopy.feature as cfeature
```

To begin, let us see which data stores are available in the Data Store Pool.

[2]:
```
DATA_STORE_POOL.store_instance_ids
```

[2]:
```
['cci-store', 'cci-zarr-store', 'local']
```

[3]:
```
# uncomment/comment the cci_store variable below to use the
cci_store='cci-zarr-store'
# cci_store='cci-store'

data_store = DATA_STORE_POOL.get_store(cci_store)
display(data_store.get_search_params_schema())
```

```
<xcube.util.jsonschema.JsonObjectSchema at 0x7fc4e8768d60>
```

3 2. Find the required data in ESA's CCI Open Data Portal

3.1 2.1. List all data in the CCI ZARR store

Let's start with finding 'Soil Moisture', and 'Land Cover' data by querying for the Essential Climate Variables name `SOILMOISTURE` and `LAND_COVER`

```
[4]: if cci_store == 'cci-zarr-store':
         #you can also list all the data ids of the smaller zarr catalogue
        print(*list(data_store.get_data_ids()),sep="\n")
     else:
         soilmoistureQuery=data_store.search_data(ecv="SOILMOISTURE")
         for i,soilItem in enumerate(soilmoistureQuery):
             print("\nDataset item %d:\nds_id: %s"%(i,soilItem.data_id))
             display(soilItem)
```

```
ESACCI-BIOMASS-L4-AGB-MERGED-100m-2010-2018-fv2.0.zarr
ESACCI-GHG-L2-CH4-SCIAMACHY-WFMD-2002-2011-fv1.zarr
ESACCI-GHG-L2-CO2-SCIAMACHY-WFMD-2002-2012-fv1.zarr
ESACCI-ICESHEETS_Antarctica_GMB-2002-2016-v1.1.zarr
ESACCI-ICESHEETS_Greenland_GMB-2003-2016-v1.1.zarr
ESACCI-L3C_CLOUD-CLD_PRODUCTS-AVHRR_NOAA-1982-2016-fv3.0.zarr
ESACCI-L3C_SNOW-SWE-1979-2018-fv1.0.zarr
ESACCI-L4_GHRSST-SST-GMPE-GLOB_CDR2.0-1981-2016-v02.0-fv01.0.zarr
ESACCI-LC-L4-LCCS-Map-300m-P1Y-1992-2015-v2.0.7b.zarr
ESACCI-OC-L3S-IOP-MERGED-1M_MONTHLY_4km_GEO_PML_OCx_QAA-1997-2020-fv5.0.zarr
ESACCI-PERMAFROST-L4-ALT-MODISLST-AREA4_PP-1997-2018-fv02.0.zarr
ESACCI-SEAICE-L3C-SITHICK-SIRAL_CRYOSAT2-NH25KMEASE2-2010-2017-fv2.0.zarr
ESACCI-SEAICE-L4-SICONC-AMSR_50.0kmEASE2-NH-2002-2017-fv2.1.zarr
ESACCI-SEALEVEL-IND-MSLTR-MERGED-1993-2016-fv02.zarr
ESACCI-SEALEVEL-L4-MSLA-MERGED-1993-2015-fv02.zarr
ESACCI-SOILMOISTURE-L3S-SSMV-COMBINED-1978-2020-fv05.3.zarr
```

3.2 2.2. Select the correct data name

```
[5]: #note the appropriate dataset is called slightly different depending on the␣
      ↪cci-store which is being used
     if cci_store == "cci-zarr-store":
         SMdataset_id="ESACCI-SOILMOISTURE-L3S-SSMV-COMBINED-1978-2020-fv05.3.zarr"
         LCdataset_id="ESACCI-LC-L4-LCCS-Map-300m-P1Y-1992-2015-v2.0.7b.zarr"
     else:
         SMdataset_id="esacci.SOILMOISTURE.day.L3S.SSMV.multi-sensor.multi-platform.
      ↪COMBINED.v05-3.r1"
         LCdataset_id="esacci.LC.L4.LCCS.Map.300m.p1y.v20.7b.r1"# check this name

     #describe the dataset
     soil_descriptor=data_store.describe_data(SMdataset_id)
     display(soil_descriptor)

     LC_descriptor=data_store.describe_data(LCdataset_id)
     display(LC_descriptor)
```

```
<xcube.core.store.descriptor.DatasetDescriptor at 0x7fc499b88f70>

<xcube.core.store.descriptor.DatasetDescriptor at 0x7fc47e72cd60>
```

3.3 2.3. Questions

A number of checks need to be done over the avilable datasets: 1. Note down the units of the soil Moisture maps for information. 2. What kind of classes do you need to find agricultural areas? 3. Who should you cite when using this datasets?

4 3. Download (a subset of) the data and store it 'locally'

Up to now, we just queried the open data portal but did not actually download or process any data. In the following steps you will be extracting the required data by setting constraints in time and space to reduce the data download. The soil moisture data will aslo be sampled on monthly intervals before being stored locally

4.1 3.1. Apply a regional restriction based on a hand-drawn polygon

Polygons, and other geometr directoryical objects in GIS systems can be conveniently expressed in the Well Known Text representation (WKT). The polygon below `Polygon (( ...))`, was drawn using QGIS and exported as WKT. We furthermore need to load this in a Shapely Polygon object so cate can work with.

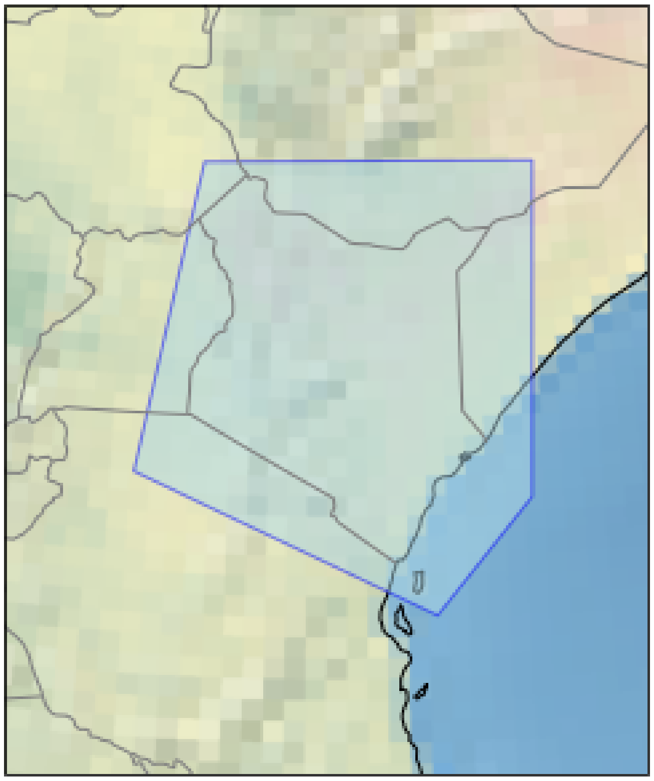
[6]:
```
# The WKT representation representing the rough outline of the Kenya as a␣
↪region of interest

wktkenyaDrought="Polygon ((34.0218774057365323 5.83180890691609211, 43.
↪11430983816897111 5.80626836637555233, 42.62903956789870108 -3.
↪10738028227309826, 40.17714767600680403 -5.96792082281363712, 32.
↪5660665949257222 -2.494407309300124, 32.5660665949257222 -2.494407309300124,␣
↪32.5660665949257222 -2.494407309300124, 34.0218774057365323 5.
↪83180890691609211))"
KenyaPoly=wkt.loads(wktkenyaDrought)

mapproj=ccrs.LambertConformal(central_longitude= 37.9,central_latitude=-0.02)
# Curious as you might be, let's make a quick and dirty plot to see how the␣
↪polygon looks like on a map
ax = plt.axes([0, 0, 1, 1],projection=mapproj)
ax.set_extent([30, 45, 10, -10])
ax.coastlines()
ax.stock_img()
ax.add_geometries([KenyaPoly],crs=ccrs.
↪PlateCarree(),facecolor="lightblue",edgecolor="blue",alpha=0.5)

#spice up your plot with some Natural Earth country boundaries
ax.add_feature(cfeature.BORDERS, edgecolor='gray')
```

[6]:
```
<cartopy.mpl.feature_artist.FeatureArtist at 0x7fc4783033a0>
```



4.2 3.2 Open a subset based upon region and time constraints

```
[26]: withinT=[datetime(2005,1,1),datetime(2020,12,31)]

      # load the dataset
      soilDset=open_dataset(ds_id=SMdataset_id, data_store_id=cci_store,␣
       ↪time_range=withinT,normalize=True,region=wktkenyaDrought)
```

```
[27]: withinT1=[datetime(2015,6,3),datetime(2015,7,3)]
      # load the dataset
      LCDset=open_dataset(ds_id=LCdataset_id,␣
       ↪data_store_id=cci_store,time_range=withinT1,␣
       ↪normalize=True,region=wktkenyaDrought)
```

4.3 3.3. Show that the correct dataset is opened

```
[9]: LCDset
```

```
[9]: <xarray.Dataset>
     Dimensions:              (lat: 4247, lon: 3797, time: 1)
     Coordinates:
```

```
  * lat                (lat) float32 -5.965 -5.963 -5.96 … 5.824 5.826 5.829
  * lon                (lon) float32 32.57 32.57 32.57 … 43.11 43.11 43.11
  * time               (time) datetime64[ns] 2015-07-03
Data variables:
    change_count       (time, lat, lon) float32 dask.array<chunksize=(1, 852,
1236), meta=np.ndarray>
    current_pixel_state  (time, lat, lon) float32 dask.array<chunksize=(1, 852,
1236), meta=np.ndarray>
    lccs_class         (time, lat, lon) float32 dask.array<chunksize=(1, 852,
1236), meta=np.ndarray>
    observation_count  (time, lat, lon) float32 dask.array<chunksize=(1, 852,
1236), meta=np.ndarray>
    processed_flag     (time, lat, lon) float32 dask.array<chunksize=(1, 852,
1236), meta=np.ndarray>
Attributes: (12/38)
    Conventions:               CF-1.6
    TileSize:                  2048:2048
    catalogue_url:             https://catalogue.ceda.ac.uk/uuid/b382ebe6679…
    cdm_data_type:             grid
    comment:
    contact:                   landcover-cci@uclouvain.be
    …                         …
    summary:                   This dataset contains the global ESA CCI land…
    time_coverage_end:         2015-07-03T00:00:00
    time_coverage_start:       2015-07-03T00:00:00
    title:                     ESA CCI Land Cover Map
    tracking_id:               12590bad-9014-4a91-9048-d06b67965490
    type:                      ESACCI-LC-L4-LCCS-Map-300m-P1Y
```

```
[10]: soilDset
```

```
[10]: <xarray.Dataset>
Dimensions:          (lat: 47, lon: 42, time: 1827)
Coordinates:
  * lat              (lat) float64 -5.875 -5.625 -5.375 … 5.125 5.375 5.625
  * lon              (lon) float64 32.62 32.88 33.12 33.38 … 42.38 42.62 42.88
  * time             (time) datetime64[ns] 2016-01-01 2016-01-02 … 2020-12-31
Data variables:
    dnflag           (time, lat, lon) float32 dask.array<chunksize=(9, 47, 42),
meta=np.ndarray>
    flag             (time, lat, lon) float32 dask.array<chunksize=(9, 47, 42),
meta=np.ndarray>
    freqbandID       (time, lat, lon) float32 dask.array<chunksize=(9, 47, 42),
meta=np.ndarray>
    mode             (time, lat, lon) float32 dask.array<chunksize=(9, 47, 42),
meta=np.ndarray>
    sensor           (time, lat, lon) float32 dask.array<chunksize=(9, 47, 42),
meta=np.ndarray>
    sm               (time, lat, lon) float32 dask.array<chunksize=(9, 47, 42),
meta=np.ndarray>
    sm_uncertainty   (time, lat, lon) float32 dask.array<chunksize=(9, 47, 42),
meta=np.ndarray>
    t0               (time, lat, lon) datetime64[ns] dask.array<chunksize=(9, 47,
42), meta=np.ndarray>
Attributes: (12/40)
    Conventions:                 CF-1.7
    cdm_data_type:               Grid
    comment:                     This dataset was produced with funding of the…
    contact:                     cci_sm_contact@eodc.eu
    creator_email:               cci_sm_contact@eodc.eu
    creator_name:                Department of Geodesy and Geoinformation, Tec…
    …                           …
    time_coverage_duration:      P1827D
    time_coverage_end:           2020-12-31T00:00:00
    time_coverage_resolution:    P1D
    time_coverage_start:         2016-01-01T00:00:00
    title:                       ESA CCI Surface Soil Moisture COMBINED active…
    tracking_id:                 174f51a4-6a06-4352-a712-4d3f77bd8166
```

5 4. Extract 'Agricultural Area' class from land cover map

5.1 4.1. Extract soil moisture for the agricultural area.

Use the agricultural area class to extract soil moisture for the agricultural area. First, we need to create a mask based on the land cover class

[28]:
```
# print the land cover attributes (note: we need to split nameing at␣
↪whitespace) and search for
for val,flag in zip(LCDset.lccs_class.attrs["flag_values"],LCDset.lccs_class.
↪attrs["flag_meanings"].split()):
     print(val,flag)

#Let's make a list of the classes we're interested
import re
# in this case let's gather all classes which have 'crop' in it's name
agriclasses=[val for val,flag in zip(LCDset.lccs_class.
↪attrs["flag_values"],LCDset.lccs_class.attrs["flag_meanings"].split()) if re.
↪search('cropland',flag)]

# create  a mask and remove the time axis (it has only one coordinate)
mask=LCDset.lccs_class.isin(agriclasses).squeeze('time')

#interpolate mask onto the soil moisture grid so it can be more easily applied
```

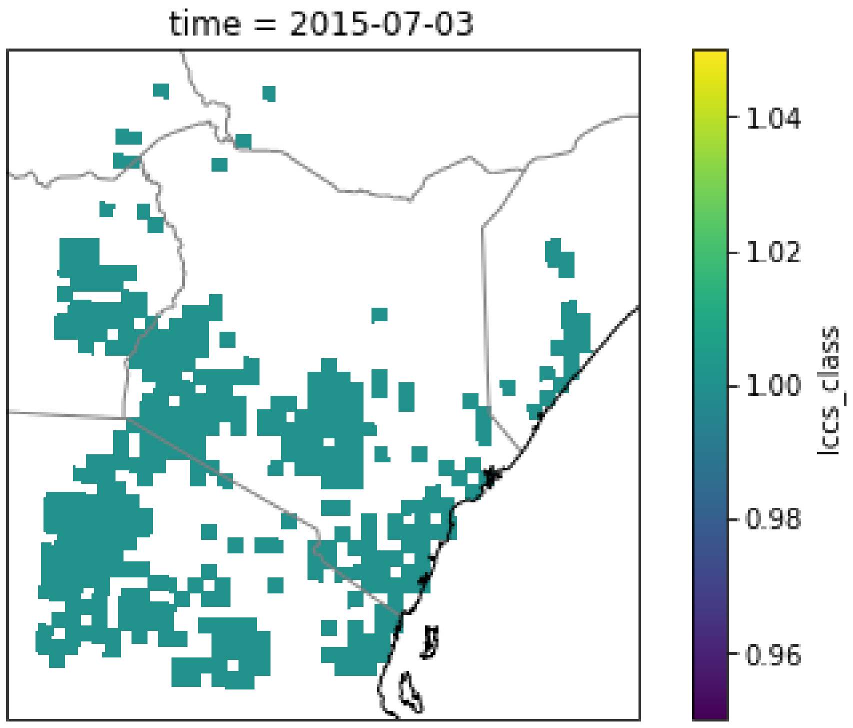
```
masksm=mask.astype(int).interp(lat=soilDset.lat,lon=soilDset.
 ↪lon,method='nearest')
masksm=masksm.where(masksm == 1) # set the 0 values to Nan (not a numbers)


# you can make a plot of the mask if you're interested
p = masksm.plot(transform=ccrs.PlateCarree(),subplot_kws={"projection":␣
 ↪mapproj})
p.axes.coastlines()
p.axes.add_feature(cfeature.BORDERS, edgecolor='gray')
```

```
0 no_data
10 cropland_rainfed
11 cropland_rainfed_herbaceous_cover
12 cropland_rainfed_tree_or_shrub_cover
20 cropland_irrigated
30 mosaic_cropland
40 mosaic_natural_vegetation
50 tree_broadleaved_evergreen_closed_to_open
60 tree_broadleaved_deciduous_closed_to_open
61 tree_broadleaved_deciduous_closed
62 tree_broadleaved_deciduous_open
70 tree_needleleaved_evergreen_closed_to_open
71 tree_needleleaved_evergreen_closed
72 tree_needleleaved_evergreen_open
80 tree_needleleaved_deciduous_closed_to_open
81 tree_needleleaved_deciduous_closed
82 tree_needleleaved_deciduous_open
90 tree_mixed
100 mosaic_tree_and_shrub
110 mosaic_herbaceous
120 shrubland
121 shrubland_evergreen
122 shrubland_deciduous
-126 grassland
-116 lichens_and_mosses
-106 sparse_vegetation
-105 sparse_tree
-104 sparse_shrub
-103 sparse_herbaceous
-96 tree_cover_flooded_fresh_or_brakish_water
-86 tree_cover_flooded_saline_water
-76 shrub_or_herbaceous_cover_flooded
-66 urban
-56 bare_areas
-55 bare_areas_consolidated
-54 bare_areas_unconsolidated
```

```
-46 water
-36 snow_and_ice
```

[28]:
```
<cartopy.mpl.feature_artist.FeatureArtist at 0x7fc478304e50>
```



5.2 4.2. Monthly aggregate

[29]:
```
smMonDset=ops.temporal_aggregation(ds=soilDset,method="mean",period="MS")
```

6 5. Soil Moisture Deficit Index (SMDI)

6.1 5.1. How to compute SMDI?

For soil moisture the method we will use is called the Soil moisture deficit index (SMDI). It is a solution where the soil moisture of the previous month is used to calculate the SMDI of the actual month. A loop is used t cover the whole series.

Variables 1. $SMDI_j$= soil moisture deficit index of the month j 2. $SMDI_{j-1}$= SMDI at month j-1 (previous month) 3. SD_j = monthly soil deficit at month j
4. SW_j = actual monthly soil moisture at month j.The is the soil moiture map corresponding to a certain map 5. MSW_j= long-term monthly **median** available soil water at month j. There will be 12 maps, one per month. 6. $maxSW_j$= long-term monthly **maximum** available soil water at month j. There will be 12 maps, one per month. 7. $minSW_j$ = long-term monthly **minimum**

available soil water at month j. There will be 12 maps, one per month. 8. NOTE: $_j - 1$ stands for the month previous to month j.

The sequence of the equations are: 1. Calculate: MSW_j, $maxSW_j$, $minSW_j$ 2. Calculate the monthly soil deficit for the month j

$$SD_j = (SW_j – MSW_j)/(maxSW_j – SW_j) * 100, \quad if SWj > MSWj$$

$$SD_j = (SW_j – MSW_j)/(MSW_j – minSW_j) * 100, \quad if SW_j =< MSW_j$$

3. Calculate the soil moisture deficit index dor the month 1 (estimated and initial value) $SMDI_1 = SD_1/50$ for initial month 4. Calculate the soil moistiure deficit index for all the other months

$$SMDI_j = 0.5 * SMDI_{j-1} + SD_j/50$$

The $SMDI i a value going from -4 to 4 being the negatives indicators of drought. Persistant drought can be detected by monthly accumulation of negatives SMDI giving room to a number of analysis.

6.2 5.2. Compute the monthly climatology

[30]:
```
# get the median, min and max for all Jan's, Feb's etc.
smClim=soilDset.sm.groupby("time.month")
smMed=smClim.median()
smMin=smClim.min()
smMax=smClim.max()
```

6.3 5.3. Compute SMDI

[31]:
```
# allocate enough data to hold the result (create an xarray.DataArray, and copy␣
↪the coordinate axes lon,lat,time from the orginal dataset)

#Allocate the Soil Moisture deficit index

smdIndex = xr.DataArray(np.zeros(smMonDset.sm.
↪shape),name="smd_indx",dims=["time","lat","lon"],coords={"lon":smMonDset.
↪lon,"lat":smMonDset.lat,"time":smMonDset.time})

for i,dt in enumerate(smMonDset.time.to_pandas()):
    #Monthly anomaly w.r.t. long term climatological median
    print("Compute Soil moisture deficit for %d, %s"%(i,str(dt)),end='\r')
    dSm=soilDset.sm[i,:,:]-smMed[dt.month-1,:,:]
    smMed_m_min=smMed[dt.month-1,:,:]-smMin[dt.month-1,:,:]
    smMax_m_med=smMax[dt.month-1,:,:]-smMed[dt.month-1,:,:]
    smd=100*xr.where(dSm < 0,dSm/smMed_m_min,dSm/smMax_m_med)
    #also mask results in the pixels of interest only
    smd=masksm*smd
    if i == 0:
        smdIndex[i,:,:]=smd/50
    else:
        smdIndex[i,:,:]=0.5*smdIndex[i-1,:,:]+smd/50
```

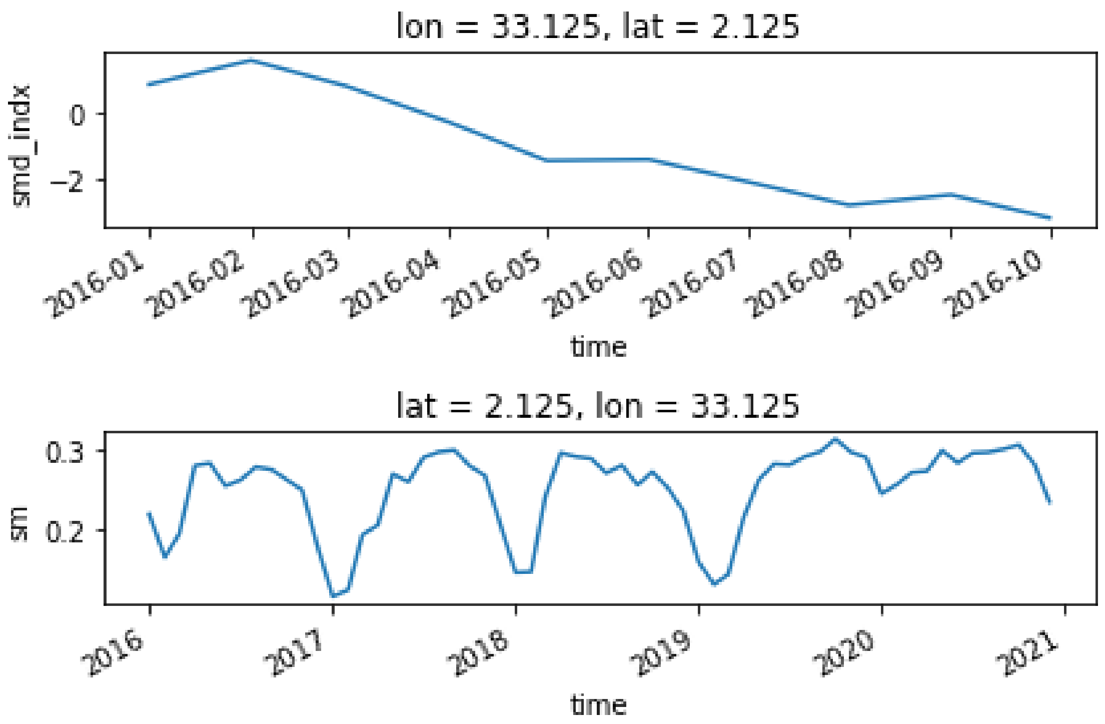
```
Compute Soil moisture deficit for 191, 2020-12-01 00:00:00
```

7 6. Results analysis

Create a profile plot (for a location) and a map at a specific time

8 6.1. Create a simple time series plot of a location

[24]:
```
#plt.figure()
fig, axes = plt.subplots(nrows=2, ncols=1)
ilon=33
ilat=2.0
ploc1=smdIndex.sel(lat=ilat,lon=ilon,method="nearest").plot(ax=axes[0])
ploc2=smMonDset.sm.sel(lat=ilat,lon=ilon,method="nearest").plot(ax=axes[1])
```

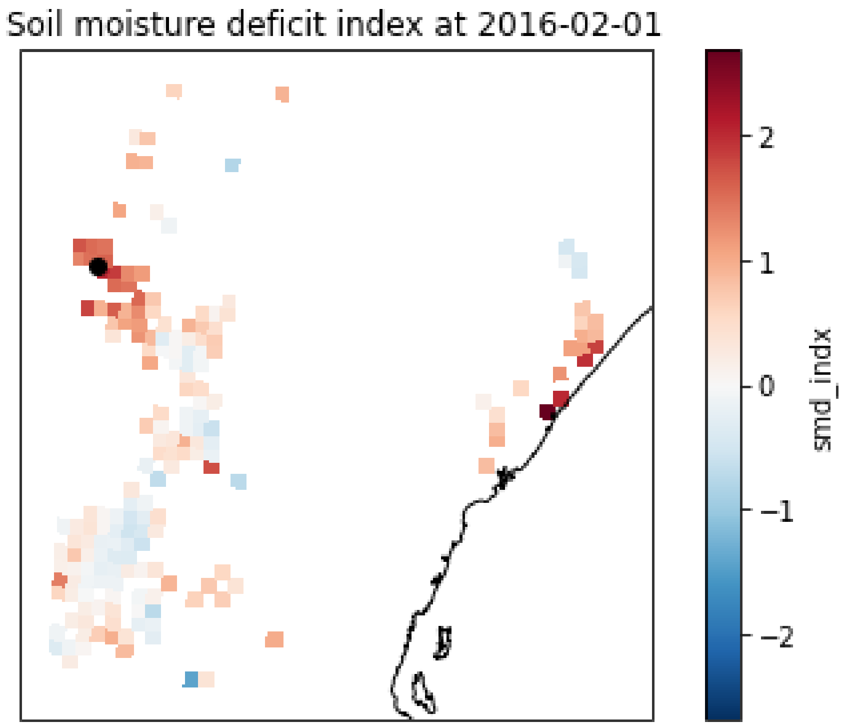


8.1 Create a geographical plot at a certain month

[25]:
```python
# Create a geographical plot at a certain month
it=1
epoch=smdIndex.time[it].data
print("Plotting epoch %s"%str(epoch))
ax = plt.subplot(projection=mapproj)
p = smdIndex.sel(time=epoch).plot(ax=ax,transform=ccrs.PlateCarree())
#also plot the location relevant for the plot above
ax.scatter([ilon],[ilat],transform=ccrs.PlateCarree(),color="black")
p.axes.coastlines()
p.axes.set_title("Soil moisture deficit index at %s"%(str(epoch)[:10]))
```

```
Plotting epoch 2016-02-01T00:00:00.000000000
```

[25]:
```
Text(0.5, 1.0, 'Soil moisture deficit index at 2016-02-01')
```



9 7. Questions

1. Try creating a time plot of the monthly mean of the SMDI
2. Identify the very dry areas by identifying a consecutive number of months of negative SMDIj.
3. Create a plot durng the drought period at certain key locations (e.g., Turkana, Mandera, Marsabit, Garissa, Wajir, Isiolo, Tana River, Machakos, Makueni, and Kitu)
4. Identify the year, and the months of the very dry area in Kenya.

5. Check out the ‘https://foodsecurityindex.eiu.com/Country’ and determoine the Global food security index for Kenya.
6. Check out the ‘http://www.fao.org/publications/sofi/2020/en/’ for Kenya and the interpret the number of undernourished people in that country.

[ ]:
```

```