Simplified Surface Energy Balance (SSEBop) ETa Version5

March 2019

Stefanie Kagone

[Simplified Surface Energy Balance (SSEBop) ETa Version5 1](#_Toc2326403)

[Introduction 3](#_Toc2326404)

[Required Software/Access 3](#_Toc2326405)

[Data 4](#_Toc2326406)

[Input data 4](#_Toc2326407)

[LST 4](#_Toc2326408)

[Emissivity 4](#_Toc2326409)

[NDVI 4](#_Toc2326410)

[Maximum air temperature 4](#_Toc2326411)

[ETo,d 5](#_Toc2326412)

[Albedo 5](#_Toc2326413)

[​Albedo Desert Mask 5](#_Toc2326414)

[Desert and Water mask 5](#_Toc2326415)

[dT calculation 6](#_Toc2326416)

[ArcMap templates 6](#_Toc2326417)

[Colormap and Layerfile 6](#_Toc2326418)

[Logos of funding agencies 7](#_Toc2326419)

[Output Data 7](#_Toc2326420)

[Actual ET products: 7](#_Toc2326421)

[ET Anomaly Products: 7](#_Toc2326422)

[Methods 8](#_Toc2326423)

[dT calculation 8](#_Toc2326424)

[Rn calculation 8](#_Toc2326425)

[Rns calculation 8](#_Toc2326426)

[Google Earth Engine code 8](#_Toc2326427)

[Rnl calculation 10](#_Toc2326428)

[Ra calculation 10](#_Toc2326429)

[Python code for Ra calculation 10](#_Toc2326430)

[Python code for Rn calculation 11](#_Toc2326431)

[dT calculation 13](#_Toc2326432)

[Python code for dT calculation 13](#_Toc2326433)

[Temperature correction (dynamic c factor) 14](#_Toc2326434)

[c factor calculation: 14](#_Toc2326435)

[ETf Calculation 15](#_Toc2326436)

[“Cloud Mask” or PAPA (previous after previous after) algorithm 18](#_Toc2326437)

[QA band creation 25](#_Toc2326438)

[Calculate Actual ET 25](#_Toc2326439)

Introduction

Actual ET (ETa) is produced using the operational Simplified Surface Energy Balance (SSEBop) model (Senay et al., 2013) for the period 2003 to present. The SSEBop setup is based on the Simplified Surface Energy Balance (SSEB) approach (Senay et al., 2007, 2011) with unique parameterization for operational applications. It combines ET fractions generated from remotely sensed MODIS thermal imagery, acquired every 10 days (dekad), with reference ET using a thermal index approach. The unique feature of the SSEBop parameterization is that it uses pre-defined, seasonally dynamic, boundary conditions that are unique to each pixel for the “hot/dry” and “cold/wet” reference points. The original formulation of SSEB is based on the hot and cold pixel principles of SEBAL (Bastiaanssen et al., 1998) and METRIC (Allen et al., 2007) models.

References:

Allen, R.G., Tasumi, M., Trezza, R., 2007. Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) – Model. ASCE J. Irrigation and Drainage Engineering 133, 380-394.

Bastiaanssen, W.G.M., M. Menenti, R.A. Feddes, and A. A. M. Holtslag, 1998. The surface energy balance algorithm for land (SEBAL): Part 1 formulation. Journal of Hydrology 212–213: 198–212.

Senay, G.B., M. Budde, J.P. Verdin, and A.M. Melesse, 2007. A coupled remote sensing and simplified surface energy balance approach to estimate actual evapotranspiration from irrigated fields. Special issue: Remote sensing of natural resources and the environment. SENSORS, 1, 979-1000.

Senay, G.B., M. Budde, J.P. Verdin, 2011. Enhancing the Simplified Surface Energy Balance (SSEB) approach for estimating landscape ET: Validation with the METRIC model. Agricultural Water Management, 98: 606-618.

Senay, G.B., S. Bohms, R. Singh, P.A. Gowda, N.M. Velpuri, H. Alemu and J.P. Verdin, 2013. Operational evapotranspiration mapping using remote sensing and weather datasets: A new parameterization for the SSEB approach. Journal of American Water Resources Research. In Press.

Required Software/Access

ArcGIS software

Python

Stornext access to disk 1) fewspsnfs2 and 2) scienceweb1

Abbreviations:

YYYY – year (e.g. 2013)

YY - year (e.g. 13)

MM – month

DD – day

DDD - dekad

cty – region (e.g. afg, irq, pak, taj)

Data

Input data

LST

Saved at: D:\Stornext\fewspsnfs2\WaterSmart\Data\Temperature\Global\_LST\V006

10-day land surface temperature (Ts) data from MODIS global land surface temperature (LST) and emissivity data (Aqua, 1km) product from website <https://dds.cr.usgs.gov/emodis/Global_LST6/>

Emissivity

Saved at: D:\Stornext\fewspsnfs2\WaterSmart\Data\Emissivity\Global\_EMIS\V006

Emissivity is a band in the LST dataset. Scale factor is emissivity \* 0.002 + 0.49 (offset).

NDVI

Using new Version 6 MODIS data, Saved at: D:\Stornext\fewspsnfs2\WaterSmart\Data\NDVI\Global\_NDVI\V006

NDVI “too little light in winter” issue fix:

Periods before 073 and after 281 don’t have the full global extent of NDVI due to no enough daylight to reflect to the satellite. Therefore, the missing values are filled with the value “-500”.

The NDVI data downloaded is improved using following steps:

* Mosaic “-500” value raster to affected dekads
* Create 8-day time steps by dividing the 2 neighboring periods (001+016/2 = 009)
* Reassign 8-day periods to dekadal time steps by ignoring periods 017, 057, 097, 137, 177, 209, 241, 281, 321, 361

Those steps are all included in a python script named D:\Stornext\fewspsnfs2\WaterSmart\Data\NDVI\Global\_NDVI\NDVI\_processing1.py

For Dynamic C Factor grid creation use: D:\Stornext\fewspsnfs2\WaterSmart\Data\NDVI\Global\_NDVI\med\_dek0317 🡪 median NDVI for each dekad from 2003 – 2017.

For ET fraction calculation use: D:\Stornext\fewspsnfs2\WaterSmart\Data\NDVI\Global\_NDVI\maxNDVI0317.tif file for operational processing because the previous year Annual maximum NDVI is not available yet, which will be used when the year is reprocessed at the beginning of the new year. The annual maximum NDVI has a range to 10000 by setting all values > 10000 to 10000.

Maximum air temperature

Tmax from Daymet (<https://daymet.ornl.gov/>, Version 3) and Tmax from WorldClim (http://www.worldclim.org/, Version 2, smoothed with a “3 running average” to level monthly time steps) where combined to create “Worldmet” data set, what is now the best available data for the globe all in one. D:\Stornext\fewspsnfs2\WaterSmart\Data\Temperature\TA\_worldmet\tmax

ETo,d

D:\Stornext\fewspsnfs2\WaterSmart\Data\ReferenceET\Global\_PET\ETod

ETo,d for dekad is a combination of GDAS PET from the FEWS website (daily, 100km) and IWMI PET. The values from GDAS were applied to the spatial resolution of 1km from the IWMI PET. The average from 2003 – 2012, aggregated from daily to dekadal sum. Resampled GDAS to 0.1, bilinear method. Used “mosaic to new raster” tool in ArcGIS to combine the two datasets, GDAS (resampled) provides the values for IWMI, used minimum mosaic operator.

Albedo

D:\Stornext\fewspsnfs2\WaterSmart\Data\Albedo\Global\_ALBEDO\006\medalb

Global data from LP DAAC MCD43A3.006 MODIS Albedo Daily 500m, band White Sky Albedo (WSA) shortwave (Band 61). Using Google Earth Engine, a 10 day median albedo from 2003 – 2017 was created. Postprocessing of Albedo files was necessary to obtain a full extent dataset for ET processing. Steps for processing included:

* Creating 1 dataset by mosaicking the 2 GEE export files
* Resample and snap the raster to match the Global LST dataset for cell size and alignment
* Build raster attribute table
* Set background value “0” to nodata to be able to fill those with values for processing
* The “too little light issue” is present in dekads 011-031 and 101-123. Thus, all “0” values are assigned a value of 500. (See Albedo correction for more detail). This allows the ETf to have the full extent and not be cut off in the northern latitudes.

​Albedo Desert Mask

Different approaches to artificially increase LST temperature to reduce ET in high reflectance areas were tested and the Koeppen Climate Zones were used to mask out high temperature areas. The value of 500 was chosen during analysis of the Albedo data and the condition of the Albedo correction for filling the missing data extent, which is parameterized to increase LST if the Albedo value is greater than 250 and the NDVI is greater than 0 and it’s a high temperature area. Those areas are defined in the Albedo Desert mask. The Albedo mask is not including lakes or small islands like the Florida Keys. Therefore, the mask was spatially enhanced by the dT grid’s spatial extent to include as much detail as possible.

The climate zone data were taken from <http://koeppen-geiger.vu-wien.ac.at/> and are created by Dr. Markus Kottek and Dr. Franz Rubel ([Kottek et al. (2006)](http://koeppen-geiger.vu-wien.ac.at/present.htm), [Rubel and Kottek (2010)](http://koeppen-geiger.vu-wien.ac.at/shifts.htm) and [Rubel et al. (2017)](http://koeppen-geiger.vu-wien.ac.at/alps.htm)). For the mask we used the classes BWk, BWh, BSk, BSh and created a raster file with value 1 for those 4 classes and value 2 for the other areas. The final raster is saved at D:\FEWS\DataPortal\templates\etaV4\layers\Albedo\_desert\_mask\_all.tif and D:\Stornext\fewspsnfs2\FEWS\Users\Stefanie\operational\GlobalET\masks\ClimateZones1976-2000

Desert and Water mask

Water mask: MODIS Landuse product (MCD12Q1) from 2013, extract water (value 0) from raster file, resample to 1 km (from 500m), and extract just the inland water features.

To avoid covering vegetation around lakes where there is possible no water, a water occurance raster, where water = 1 and the sum of all water pixel from 2003-2013 is 12 (the number of years) -> that mean there is stable water (or min extent). Water\_occurance\_mask.tif is showing water where all 12 years had water.

ET calculation = water bodies identified in the mask are replaced with PET \*0.85 (85%).

Desert mask: after some testing of different extents and methods to delineate the deserts we decided on following: 10x10 median cfactor raster and max MODIS NDVI < 0.2, when both interest then is desert. NDVI is used for smoothing out the edges and make sure no irrigation is adjusted (included) in the mask.

ET calculation = ET desert identified in the mask are replaced with ET \* 0.32.

Correction coefficient of 0.32 was derived from (2003-2015)

Where, if that values >= 3 (3 times ET) then take the average (zonal) for the values from . The average for those pixels = 0.32. Note: Had to take out outliers’ value 20000 to not skew the average.

dT calculation

Data is saved at D:\Stornext\fewspsnfs2\FEWS\Users\Stefanie\operational\Ra\_calculationV5\dT\_RsRsoK016\dT and under D:\Stornext\fewspsnfs2\WaterSmart\Data\Temperature\dT\_world\dT in the main data folder.

The equations are based on the FAO Irrigation and Drainage Paper No. 56 by Richard Allen, et al.

See Methods section for details.

ArcMap templates

ArcMap templates for output graphics, Input shapefiles for water, boundaries, etc.is all under

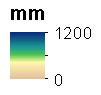
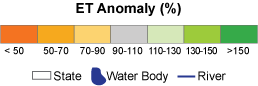
\\FEWS\DataPortal\templates\etaV5

Layout notes:

* No frame around map
* Country boundary thicker then every else and black
* Surrounding country lines 60% grey and thin
* No water color
* Anomaly grey = 30%
* Lakes below the country boundaries

Colormap and Layerfile

The actual ET data unit is millimeters (mm) and the associated color ramp (see Figure 1a.) is saved at \\Stornext\fewspsnfs2\WaterSmart\Data\Symbology\SSEBopEvapotranspiration.style. The anomalies are the ratio of ETa and the corresponding median ETa, expressed as a percent value and have the color ramp (see Figure 1b.) attached. The color ramp, txt file with RGB color codes, and colormap file (.clr) are available in the Symbology folder. The layer file used in the python script is saved under D:\Stornext\fewspsnfs2\FEWS\DataPortal\templates\etaV4\anomaly.lyr



about average

above average

below average

Figure 1a. Figure 1b.

Logos of funding agencies

Logos included in the final graphic are from USGS, USAID, and FEWS NET. The graphic is located under D:\Stornext\fewspsnfs2\FEWS\DataPortal\templates\logos\3logos.gif.

Output Data

Actual ET products:

* Global – Annual, monthly, dekadal (geotiff, no color attached)
* Africa – Annual, monthly, dekadal (geotiff, no color attached)
* Central Asia - Annual, monthly, dekadal (geotiff, no color attached)

ET Anomaly Products:

The anomalies are the ratio of ETa and the corresponding median ETa, expressed as a percent value.

* Global – Annual, monthly, Mar-Oct, Oct-May (png/pdf, geotiff)
* Africa – Annual, monthly, (png/pdf, geotiff)
* East Africa – Jan-Jul, May-Sep, Aug-Dec, Oct-Jan (png/pdf, geotiff)
* West Africa – Jun-Oct (png/pdf, geotiff)
* Southern Africa – Jan-May, Oct-May (png/pdf, geotiff)
* North Africa – not published on website
* Central Asia - Annual, monthly, Apr-Oct (png/pdf, geotiff)
* South Central Asia – monthly, Mar-Oct (png/pdf, geotiff)
* Afghanistan – monthly, Mar-Oct (png/pdf, geotiff)
* Pakistan – monthly, Mar-Oct (png/pdf, geotiff)
* Tajikistan – monthly, Mar-Oct (png/pdf, geotiff)
* Kazakhstan – monthly, Apr-Oct (png/pdf, geotiff)
* Yemen – monthly, May-Oct (png/pdf, geotiff)

Products are available from 2003 – current at the Earlywarning website <https://earlywarning.usgs.gov/fews>

Staged for distribution to website under D:\Stornext\scienceweb1\shared\fews\web.

Metadata:

All geotiff files are accompanied by a .xml metadata file in the .zip folder. The .xml file is created as part of the python script from a template. The template is located at D:\Stornext\fewspsnfs2\FEWS\DataPortal\templates\etaV5\ template\_modisSSEBopETv5.xml.

Methods

dT calculation

Radiation data (Rn) can be derived from the air temperature difference. Once Rn is determined, dT can be calculated.

In the SSEBop model following process is used to determine

1. net radiation (Rn)
2. temperature difference (dT)

The equations are based on the FAO Irrigation and Drainage Paper No. 56 by Richard Allen, et al.

Rn calculation

(Equ. 40)

where  
Rns - incoming net shortwave radiation  
Rnl - net outgoing longwave radiation

Rns calculation

(Equ. 38)  
where  
α - albedo (value 0.23 for the hypothetical grass reference crop)  
Rs - incoming solar radiation

(Equ. 50)

where  
kRs - adjustment coefficient for coastal or interior land  
Tmax - daymet V3 data  
Tmin - daymet V3 data  
Ra - extraterrestrial radiation (calculated)

The Rs equation is an update from the SSEBop ET model Version 4.  
For ‘interior’ locations, where land mass dominates and air masses are not strongly influenced by a large water body, kRs ≅ 0.16, which was used in the Rs calculation. The daymet V3 data is a 30 year median (1985-2014) processed with Google Earth Engine by MacKenzie Friedrichs. The daily data was summarized by dekad for use in global ET model.

Google Earth Engine code

/\* Create 30 year climatology for Daymet V3

Needed for dT calculation: Tmax, Tmin, Tavg

Create daily median daymet for Temp Daymet data and WorldClim data is later combined to create best available data set \*/

var doy\_list = ee.List.sequence(1,365).getInfo();

for (var i=0; i < doy\_list.length; i++){

// Start and End year for climatology

var start\_year = '1985-01-01';

var end\_year = '2014-12-31';

// var doy = 60;

var doy = doy\_list[i];

// Function to add Julian Date

var addJulian = function(image) {

var doy = ee.Date(image.get('system:time\_start')).getRelative('day', 'year');

var year = ee.Date(image.get('system:time\_start')).get('year');

var doy = doy.add(1);

return image.set({'doy': doy, 'year': year});

}

//Create 30 year median for Tmax and Tmin

var daymet\_collection = ee.ImageCollection('NASA/ORNL/DAYMET\_V3').select(['tmax'])

var daymet\_collection = daymet\_collection.map(addJulian)

.filterDate(start\_year, end\_year)

.filter(ee.Filter.dayOfYear(doy, doy));

//remove leap days from collection

var leap\_day\_filter = ee.Filter.and(

ee.Filter.calendarRange(2, 2, 'month'),

ee.Filter.calendarRange(29, 29, 'day\_of\_month')).not();

var labelname = 'Median DOY ' + String(doy);

print(daymet\_collection, labelname);

//Create a daily median image in Kelvin and display

//Adding 0.5 to 273.15 to round up before integerizing.

var MedianTmax = daymet\_collection.reduce(ee.Reducer.median()).add(273.65).int();

Map.setCenter(-100, 39, 3);

Map.addLayer(MedianTmax, {min: 270, max: 320});

//Export the image to match the original

var daymet\_crs = 'EPSG:4326'

var daymet\_dimensions = '9612,8075';

var daymet\_crs\_tx = '1000,0,-5802250,0,-1000,4984000';

var daymet\_region = ee.Geometry.Rectangle([-175, 15, -15, 90]);

Export.image.toDrive({

image: MedianTmax,

description: 'tmax\_median\_'+doy,

folder: 'daymet',

dimensions: daymet\_dimensions,

region: daymet\_region,

scale:1000,

crs: daymet\_crs,

//crsTransform: daymet\_crs\_tx,

maxPixels:150000000

});

}

Rnl calculation

(Equ. 39)  
where  
σ - Stefan-Boltzmann constant [4.903 10-9 MJ K-4 m-2 day-1]  
Tmax,K - maximum absolute temperature during the 24-hour period  
Tmin,K - minimum absolute temperature during the 24-hour period  
ea - actual vapour pressure  
Rs - measured or calculated (Equ. 35) solar radiation  
Rso - calculated (Equ. 37) clear-sky radiation

(Equ. 37)  
where  
z - elevation  
Ra - extraterrestrial radiation (calculated)

Ra calculation

In Version 4, the extraterrestrial radiation Ra was calculated using a 10km latitude grid, which caused smaller Islands and coastal areas to be partially or completely missing. An updated latitude grid at 1km was created using ‘Create Fishnet’ tool in ArcGIS.

(Equ. 21)

where

Ra extraterrestrial radiation [MJ m-2 day-1],

Gsc solar constant = 0.0820 MJ m-2 min-1,

dr inverse relative distance Earth-Sun (Equation 23),

ωs sunset hour angle (Equation 25 or 26) [rad],

ϕlatitude [rad] (Equation 22),

δsolar declination (Equation 24) [rad].

Python code for Ra calculation

#extraterrestrial Radiation Ra

# latitude grid

lat\_grid = 'D:\\sbohms\\lat\_pts\\lat\_grid1km'

#day in the year (from 1- 365,366)

j = 1

year = 2010 #raw\_input ("Enter year (yyyy):")

Gsc = 0.0820

dem = 'D:\\sbohms\\mn30\_grd\\mn30\_grd'

outPathscrap = 'D:\\sbohms\\scrap'

if not os.path.exists(outPathscrap):

os.makedirs(outPathscrap)

arcpy.env.scratchWorkspace = outPathscrap

while j <= 365:

print j

day1 = '00' + str(j)

day = day1[-3:]

print day

# Deg to rad

radv = (math.pi/180) \* Raster(lat\_grid)

print radv

arcpy.env.extent = radv

vara = (2 \* math.pi \* j) / 365

print vara

dr = 1 + 0.033 \* math.cos(vara)

print dr

varb = ((2 \* math.pi \* j) / 365) - 1.39

print varb

d = 0.409 \* math.sin(varb)

print d

varc1 = (- Tan(radv)) \* math.tan(d)

print varc1

varca = Con(varc1 > 1.0, 1.0, varc1) #!!!valid range 1 <-> -1, above 55 degree "no" Ra

varcd = Con(varca < -1.0, -1.0, varca)

ws = ACos(varcd)

print ws

sin\_lat = Sin(radv) \* math.sin(d)

print sin\_lat

cos\_lat = Cos(radv) \* math.cos(d)

print cos\_lat

winkel = ws \* (sin\_lat) + cos\_lat \* Sin(ws)

print winkel

Ra = ((24 \* 60) / math.pi) \* Gsc \* dr \* winkel

Ra.save('D:\\sbohms\\Ragrids\_daily' +os.sep+ 'Ra' + day)

j = j + 1

Python code for Rn calculation

tempminC = Raster(TempMin\_path+os.sep+tempmin) - 273.15

tempmaxC = Raster(TempMax\_path+os.sep+tempmax) - 273.15

tempdiff = tempmaxC - tempminC

OutputPathRS = OutputPath + os.sep + 'RS'

if not os.path.exists(OutputPathRS):

os.makedirs(OutputPathRS)

#equation for Kt correction factor not used instead use 0.16 static value from literature recommended in FAO56

Kt = 0.16

arcpy.env.extent = 'MAXOF'

OutputPathRS = OutputPath + os.sep + 'RS'

if not os.path.exists(OutputPathRS):

os.makedirs(OutputPathRS)

# clear-sky solar radiation Rs

varg = Kt \* (SquareRoot(tempmaxC-tempminC))

Rs = varg \* Raster(Ra\_path+os.sep+x)

Rs.save(OutputPathRS + os.sep+ 'Rs' + x[-3:])

The Rs equation is an update in SSEBop ET model Version 5.

OutputPathRSO = OutputPath +os.sep+ 'RSo'

if not os.path.exists(OutputPathRSO):

os.mkdir(OutputPathRSO)

# clear-sky solar radiation Rso

varg = ((0.00002 \* Raster(dem)) + 0.75)

Rso = varg \* Raster(Ra\_path+os.sep+x)

Rso.save(OutputPathRSO +os.sep+ 'Rso' + x[-3:])

OutputPathRNS = OutputPath + os.sep + 'RNS'

if not os.path.exists(OutputPathRNS):

os.makedirs(OutputPathRNS)

Rns = (1 - float(alb\_value)) \* Rs

Rns.save(OutputPathRNS + os.sep+ 'Rns' + x[-3:])

# net longwave radiation

# T in Celcius30 year daymet

OutputPathEA = OutputPath + os.sep + 'EA'

if not os.path.exists(OutputPathEA):

os.makedirs(OutputPathEA)

vard = (17.27 \* tempminC) / (tempminC + 237.3)

vart = Exp(vard)

ea = 0.6108 \* vart

ea.save(OutputPathEA + os.sep + 'ea' + x[-3:])

# T in K

tempmaxK = Raster(TempMax\_path+os.sep+tempmax) #+ 273.15

constTmax = Power(tempmaxK, 4)

print constTmax

tempminK = Raster(TempMin\_path+os.sep+tempmin) #+ 273.15

constTmin = Power(tempminK, 4)

print constTmin

#Condition Rs > Rso --> Rs = Rso

#add 0.01 to both rasters to avoid division by 0

#and resulting nodata in output

Rsc = Con(Rs > Rso, Rso, Rs)

Rsc.save(OutputPathRS +os.sep+ 'Rsc' + x[-3:])

rr = Float(Rsc+0.01) / Float(Rso+0.01)

Rs predicted by Equ. 50 should be limited to ≤ Rso from Equ. 37.  
Which means that fraction Rs/Rso is never more then 1. To avoid nodata values in the Rn raster because of Rso = 0 in the Northern Hemisphere winter months, 0.01 was added to Rsc and Rso.

OutputPathRNL = OutputPath + os.sep + 'RNL'

if not os.path.exists(OutputPathRNL):

os.makedirs(OutputPathRNL)

Rnl = (4.903 \* 0.000000001 \* ((constTmax + constTmin) / 2)) \*

(0.34 - 0.14 \* SquareRoot(ea)) \* ((1.35 \* rr) - 0.35)

Rnl.save(OutputPathRNL + os.sep + 'Rnl' + x[-3:])

OutputPathRN = OutputPath + os.sep + 'RN'

if not os.path.exists(OutputPathRN):

os.makedirs(OutputPathRN)

Rn = Rns - Rnl

Rn.save(OutputPathRN + os.sep +'Rn' + x[-3:])

dT calculation

where  
ρ - atmospheric density

cp - 1.013/1000 86400

Rn - net radiation (calculated)

cofrs - crop restistence factor of 110

(Equ. 3-5)  
where  
P - Atmospheric pressure at elevation z  
TKv - virtual temperature

(Equ. 3-4)

(Equ. 3-7)

where  
z - elevation T - mean daily temperature for 24-hour

Python code for dT calculation

OutputPathRN = OutputPath +os.sep+ 'RN'

Rn\_path = OutputPathRN

env.workspace = Rn\_path

rns = arcpy.ListRasters()

rns.sort()

print rns

logfile.write('Net Radiation: ' + str(Rn\_path) + '\n')

for rn in rns:

for tempavg in tempavgs:

if tempavg.split('.')[0][-3:] == rn[-3:]:

varf = 0.0065 \* Raster(dem)

vare = (293.0 - varf) / 293.0

P = 101.3 \* Power(vare, 5.26)

#T in K

Tkv = 1.01 \* (Raster(TempAvg\_path +os.sep+ tempavg))

den = 3.486 \* (P/Tkv)

cp = 1.013 / 1000

OutputPathdT = OutputPath +os.sep+ 'dT'

if not os.path.exists(OutputPathdT):

os.makedirs(OutputPathdT)

dT = (Raster(Rn\_path+os.sep+rn) \* float(cofrs)) / (den \* cp \* 86400)

dTc = Int(Con(dT, dT, 1, "VALUE > 1") + 0.5)

dTc.save(OutputPathdT +os.sep+ 'dT' + rn[-3:])

print 'dT' + str(rn[-3:])

Temperature correction (dynamic c factor)

The temperature correction coefficient is used to create the lower temperature boundary – Tcold - for the ETf calculation. The coefficient is based on the ratio of LST and maximum air temperature.

Input data sets for the so-called c factor are:

* LST
* dekadal median NDVI (2003-2017)
* Tmax (Worldmet)
* temp diff (dT)

For tcorr grid process use script: 1\_MODIS\_DynamicCfactor\_medianC.py under D:\Stornext\fewspsnfs2\FEWS\Users\Stefanie\operational\GlobalET

c factor calculation:

The c factor is based on the ratio of LST and Tmax (maximum air temperature) conditioned and summarized by a set of different conditions.

1. condition: median NDVI (2003-2017) must be between 0.7 – 1.0
2. condition: is between -10 and 5K
3. condition: LST > 270K

The Tcorr values that meet all 3 conditions are then summarized by a grid based on MODIS tiles, but divided further from 1 MODIS tile into 25 tiles (5x5 tiles). That number was determined based on that there are 90 rows and 180 columns overall. This new grid is stored as a shapefile under D:\Stornext\fewspsnfs2\FEWS\Users\Stefanie\operational\GlobalET\shapefiles\czones\_global\_5x5s\_wgs84.shp (shapefile is created with the Fishnet tool).

Next steps in calculation are:

* Zonal Statistic as Table with fields: MEAN, STD
* Join the grid shapefile and the analysis table. Field needed for further process: COUNT, AREA, MEAN, STD
* Add field CF (c factor) to shapefile and calculate value with
* set all c factor values to 0 when there are less than 30 pixels after applying the 3 conditions.

This avoids a c factor value based on just a couple pixels.

* That this point save the c factor by:
  + Converting the shapefile to a raster
  + Setting all 0 values to NODATA
  + Smooth the raster with Focal Statistics of a rectangle (200x200)
* Save as: D:\Stornext\fewspsnfs2\FEWS\Users\Stefanie\operational\GlobalET \cfactor\run1\_YYYY with file name ctorrYYYYDDD.tif.

At this point the c factor grid has gaps, to fill those, following median data sets need to be created:

D-Dmean-Amedian-ZAmedian-Smooth

1) Dmean: dekad mean value 2003-2017

2) Amedian: annual median of the mean value, i.e. median of the mean of 36 values

3) 3median: use the annual median and the Focal Stats tool with cell size 3x3 (660x660 km) to create

1 median grid for filling

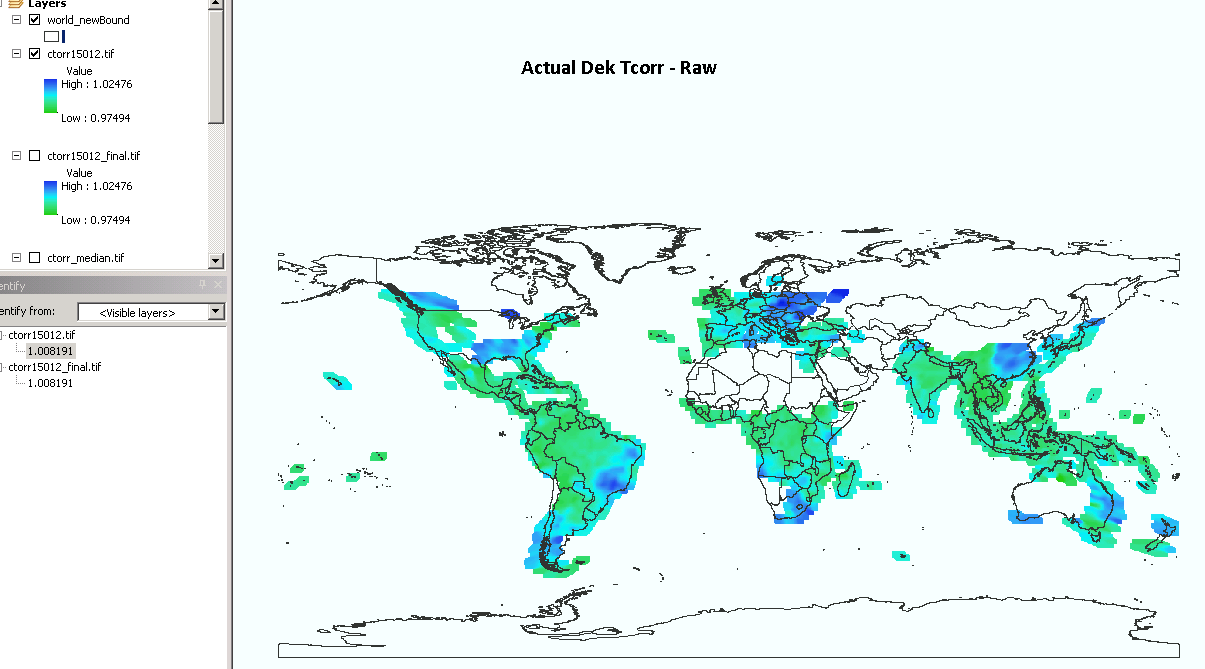
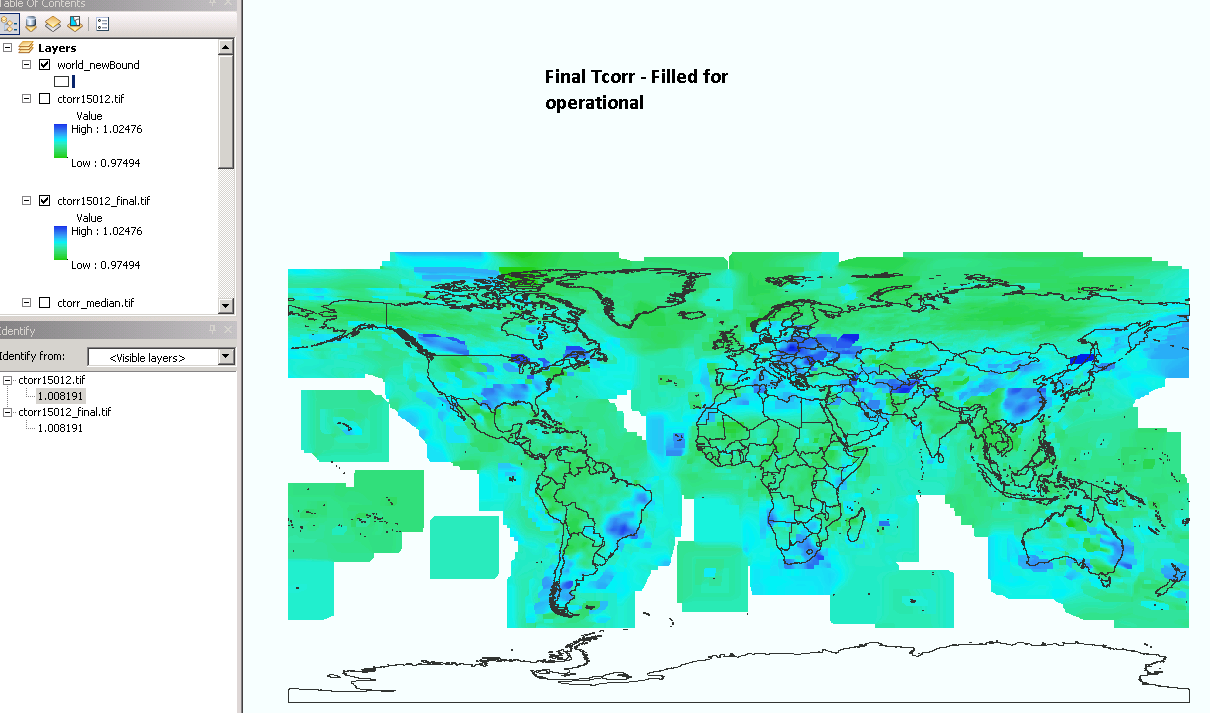
4) 3median: use the annual median and the Focal Stats tool with cell size 10x10 (2200x2200 km) to

create 1 median grid for filling

Once the median data sets are created, all NODATA values in the smoothed ctorr rasters is going to be replaced with the median data sets in the order noted above. This will fill all the gaps and ensures a complete smooth c factor grid.

The final output is saved under: D:\Stornext\fewspsnfs2\WaterSmart\Data\Tcorr\Global\_Tcorr\YYYY\ ctorrYYYYDDD.tif

Graphics show before and after the filling step.

ETf Calculation

After the c factor is processed we can go ahead with the ET fraction calculation. Input datasets are LST, EMIS, and Tcorr grid, NDVI. Median input datasets are albedo, max NDVI (2003-2017), max Temp and temp diff (dT).

# 2\_Global\_ETf.py

# LST and Emis

outPathLST = DirInData +os.sep+ 'Temperature\\Global\_LST\\V006' +os.sep+ str(year)

outPathEmis = DirInData +os.sep+ 'Emissivity\\Global\_EMIS\\V006' +os.sep+ str(year)

outPathTcorr = DirInData +os.sep+ 'Tcorr\\Global\_Tcorr' +os.sep+ str(year)

Albedo\_desert\_mask = r'D:\FEWS\DataPortal\templates\etaV4\layers\Albedo\_desert\_mask.tif'

LSTinw = outPathLST

env.workspace = LSTinw

lsts = arcpy.ListRasters('\*.tif')

lsts.sort()

log\_file.write('LST: ' + str(LSTinw) + '\n')

Emisinw = outPathEmis

env.workspace = Emisinw

emiss = arcpy.ListRasters()

emiss.sort()

Tcorrinw = outPathTcorr

env.workspace = Tcorrinw

tcorrs = arcpy.ListRasters('ctorr\*.tif')

tcorrs.sort()

OutputPathetf = DirOut +os.sep+ 'ETa\\grids\\ETf' +os.sep+ str(year)

if not os.path.exists(OutputPathetf):

os.makedirs(OutputPathetf)

arcpy.env.workspace = outPathscrap

# Loop through input data and calculate ETf one dekad at the time

for lstinp, emisinp, tcorrinp in zip(lsts, emiss, tcorrs):

if emisinp[5:8] == lstinp[5:8] == tcorrinp[7:10]:

dek = lstinp[5:8]

print dek

print '-----Start calculating ETa grid'

log\_file.write('-----Start calculating ETa grid \n')

#latest period available

LSTin = outPathLST +os.sep+ lstinp # LST raster folder

Emisin = outPathEmis +os.sep+ emisinp

Tcorrin = outPathTcorr +os.sep+ tcorrinp

#median datasets

Albedoin=Raster(DirInData+os.sep+'Albedo\\Global\_ALBEDO\\V006\\medalb\\medAlb\_'+str(dek) +'.tif')

TaRasterin = Raster(DirInData +os.sep+ 'Temperature\\TA\_worldmet\\tmax\\tmax' + str(dek) +'.tif')

rad\_dTa = Raster('D:\\Stornext\\fewspsnfs2\\WaterSmart\\Data\\Temperature\\dT\_world\\dT\\dt' + str(dek)) #+'.tif'

NDVIin = Raster(DirInData +os.sep+ 'NDVI\\Global\_NDVI\\V006\\MaxAnnualNDVI'

+os.sep+'maxndvi'+str(year)+'.tif')

NDVIdekin = Raster(DirInData +os.sep+ 'NDVI\\Global\_NDVI\\V006'+os.sep+

str(year)+os.sep+ str(year)+dek+'.1\_km\_16\_days\_NDVI.tif')

print 'Input parameter for ETa calculation:'

print LSTin

print Emisin

print Albedoin

print TaRasterin

print NDVIin

print NDVIdekin

print rad\_dTa

print Tcorrin

log\_file.write('Input parameter for ETa calculation: \n')

log\_file.write(LSTin + '\n')

log\_file.write(str(Albedoin) + '\n')

log\_file.write(str(TaRasterin) + '\n')

log\_file.write(str(NDVIin) + '\n')

log\_file.write(str(NDVIdekin) + '\n')

log\_file.write(str(rad\_dTa) + '\n')

log\_file.write(Tcorrin + '\n')

#--------cap dT at 6 and 25 degree----------------

rad\_dT1 = Con(rad\_dTa < 6, 6, rad\_dTa)

rad\_dT = Con(rad\_dT1 > 25, 25, rad\_dT1)

#-------- raster in kelvin ------------------

lstout = Raster(LSTin)

#NO LAPSE RATE CORRECTION \_ FOR GLOBAL ALREADY DONE BECAUSE TA IS A MEAN DATASET

#-------- create Tc------------------

Tc = TaRasterin \* Raster(Tcorrin)

#-------- create Hot pixel------------------

Th = Tc + rad\_dT

#--------LST modification (albedo correction)----------

#if albedo >250 and Albedo desert mask is 1 and NDVI > 0 than out\_LST\_k + 0.1 x

(albedo - 250) else LST value as is

LSTa = Con((Albedoin>=250)&(NDVIdekin>=0)&(Raster(Albedo\_desert\_mask)==1),

lstout+0.1\*(Albedoin-250),lstout)

#--------LST modification (emissivity correction)----------

#if emis>0.965 treshold and ndvi<0.25 than LSTa\*((emis/0.965))else LSTa as is

emisout = (Raster(Emisin) \* 0.002) + 0.49

LSTe = Con((emisout > 0.965)& (NDVIin >= 1) & (NDVIin <= 2500), LSTa \*

(emisout/0.965), LSTa)

#arcpy.env.extent = 'MAXOF'

#arcpy.env.extent =

#'-60.0 23.0 -130.0 47.0' #'41.0 36.5 2.0 63.0'

#--------ET fraction calculation------

# dTh - dTx(LSTa-Ta) = gridtop / dTh - dTc = gridbottom = ETfgrid

ETfgrid = (Th - LSTe) / rad\_dT

ETf = Con(ETfgrid < 0, 0, ETfgrid)

ETf.save(OutputPathetf +os.sep+ 'etf'+str(year)[-2:] + dek)

The steps in the ETf calculation are:

* Cap dT raster between 6 and 25 K
* Create Tcold with Tmax \* c factor
* Create Thot with Tcold + limited dT raster
* Albedo corrected LSTa
* Emissivity corrected LSTe
* Calculate ETf = (Thot – LSTe) / limited dT raster

Albedo correction on LST

if albedo >250 and Albedo desert mask is 1 and NDVI > 0

than LST + 0.1 x (albedo - 250) else LST value as is

For the case of MODIS images, most snow free vegetated surfaces will have an albedo value less than 0.25, so a correction is applied for areas with albedo > 0.25.

LSTa (albcorr)=LST+100(∝-0.25)

where α is the broadband albedo as reported in the MODIS dataset; LST is the reported LST image value (K); and LST corrected is the corrected LST value (K) which will increase the LST value in a greater albedo area. Again, the value of this correction is to modify results from non-vegetated surfaces that may exhibit lower LST than the expected value. The lower than expected LST values can be caused by either poor parameterization of the emissivity values in MODIS or lack of the SSEBop model in solving for the full energy balance equation at the surface.

Emissivity correction on albedo-corrected LST

if ndvi <0.25 than LSTa \* ((emis / 0.965)) else LSTa value as is

Similarly, areas with high emissivity such as dark lava rocks tend to have low thermal LST values from imagery. Although this temperature index method works well when the emissivity is around 0.97, high emissivity surface types tend to have smaller LST than the surrounding region. The lower LST is not generally accompanied by changes in the Ta, thus tend to create a lower ETf. These areas are generally located in dark rock features such as lava rocks in parts of east Africa and the mountains of the Sahara Desert. These areas need to be treated differently or they will produce a spuriously high ETf and thus a high ET value.

LSTe= LSTa \* emissivity/0.965

where 0.965 is the threshold of emissivity. It increases LST by that fraction to lower the ETf and avoid that spuriously high ETf. The threshold was picked based on tests with different values.

The actual ETf is calculated as

ETf = (Thot - LSTe) / dT, then all values < 0 set to 0

>> Save as \\FEWS\DataPortal\data\Global\Dekadal\ETa\grids\ETf\YYYY\etfYYDDD

“Cloud Mask” or PAPA (previous after previous after) algorithm

Once the ETf is processed, the data values need to be corrected for too high values, above 1.3. High values are present in areas where the difference between Thot – LSTe is significantly higher than the dT. Reasons here fore include cloud contamination and input data error. To address this, we created the PAPA algorithm, which replaces ETf values above 1.3 with the previous, next or median ETf.

#3\_CloudMask.py

year = sys.argv[1]

DirETf = r'D:\Stornext\fewspsnfs2\FEWS\Users\Stefanie\operational\GlobalET\ETa\grids\ETf'

DirOut = r'D:\Stornext\fewspsnfs2\FEWS\Users\Stefanie\operational\GlobalET\ETa\grids\ETfca'

if not os.path.exists(DirOut):

os.makedirs(DirOut)

outPathscrap = DirOut +os.sep+ 'scrapetfqa'

if os.path.exists(outPathscrap):

shutil.rmtree(outPathscrap)

print 'deleted ' + outPathscrap

os.makedirs(outPathscrap)

else:

print 'Directory does not exist!'

os.makedirs(outPathscrap)

time\_stamp = time.strftime('%Y.%m.%d\_%H.%M.%S', (time.localtime(time.time())))

strFile = DirETf +os.sep+ 'Log\_%s.txt' %time\_stamp

log\_file = open(strFile, 'a')

qadir = r'D:\Stornext\fewspsnfs2\FEWS\Users\Stefanie\operational\GlobalET\ETa\grids' +os.sep+

'QAband' +os.sep+ str(year)

if not os.path.exists(qadir):

os.makedirs(qadir)

print '-----cloud masking for ETf'

log\_file.write('-----cloud masking for ETf \n')

arcpy.env.scratchWorkspace = outPathscrap

listETfs = DirETf +os.sep+ str(year)

arcpy.env.workspace = listETfs

ETfs = arcpy.ListRasters()

ETfs.sort()

log\_file.write(str(ETfs) + '\n')

QAraslist = []

QAoutlist = []

#bogus1 = 1.05

#bogus2 = 2.0

run = 2

r = 0

if run == 3: #QA raster only – see script for code

if run == 1: #First run of ETf cloud mask with pre/nex and leave nodata for creating median

ETf for 2. Run – see script for code

if run == 2: #Second run of ETf cloud mask with pre/nex,earlier created median ETf,&fill 0.75

log\_file.write('create corrected ETf&QA raster with fill information for each pixel' + '\n')

l = int(len(ETfs))

medETfs = r'D:\Stornext\fewspsnfs2\FEWS\Users\Stefanie\operational\Global\_operational\

Median\_InputDATA\MedianETfV5'

print medETfs

for r, ETf in enumerate(ETfs):

print ETf

print r

dek = ETf[-3:]

print dek

log\_file.write('apply correction to ' +str(ETf) + '\n')

qaname = qadir +os.sep+ 'qa'+str(dek)+'.tif'

ETfaaDir = DirOut +os.sep+ 'Run1'+str(year)

if not os.path.exists(ETfaaDir):

os.makedirs(ETfaaDir)

ETfafinal = ETfaaDir +os.sep+ 'etfca' + ETf[-5:]

# 1. condition cloud-mask when ETf >= 1.3, pre, nex, pre2, nex2

#1.pre

if ETfs[r-1] == ETfs[l-1]:

condras = 'None'

print 'previous dekad not available - try next dekad'

log\_file.write('previous dekad not available - try next dekad' + '\n')

etfa1 = Raster(ETf)

etq1 = 99

pass

else:

condras = ETfs[r-1]

print condras

log\_file.write('fill pixels with ' +condras+ '\n')

#etfa1 = Con(etfa >= 2.0, condras, etfa)

etfa1 = Con(Raster(ETf) >= 1.3, condras, Raster(ETf))

etq1 = condras

#1.nex

#print (r+1), l

if r == l-1:

p = 0

pp = l-1

else:

p = 1

pp = 0

if ETfs[r+p] == ETfs[pp]:

condras = 'None'

print 'next dekad not available - try 2.previous dekad'

log\_file.write('next dekad not available - try 2.previous dekad' + '\n')

etfa2 = etfa1

etq2 = 99

pass

else:

condras = ETfs[r+p]

print condras

log\_file.write('fill pixels with ' +condras+ '\n')

#etfa2 = Con(etfa1 >= 2.0, condras, etfa1)

etfa2 = Con(etfa1 >= 1.3, condras, etfa1)

etq2 = condras

#2.pre

if r == 0: #1.run

p = 2

else: #other runs

p = 1

if ETfs[r-2] == ETfs[l-p]:

condras = 'None'

print '2.previous dekad not available - try 2.next dekad'

log\_file.write('2.previous dekad not available - try 2.next dekad' + '\n')

etfa3 = etfa2

etq3 = 99

pass

else:

condras = ETfs[r-2]

print condras

log\_file.write('fill pixels with ' +condras+ '\n')

#etfa3 = Con(etfa2 >= 2.0, condras, etfa2)

etfa3 = Con(etfa2 >= 1.3, condras, etfa2)

etq3 = condras

#2.nex

if r == l-2: #vorletzter run

p = 1

pp = l-1

elif r == l-1: #letzter run

p = 0

pp = l-1

else:

p = 2

pp = 1

if ETfs[r+p] == ETfs[pp]:

condras = 'None'

print '2.next dekad not available'

log\_file.write('2.next dekad not available - try 0.75' + '\n')

#etfa4 = Con(etfa3 >= 2.0, 0.75, etfa3)

#etfa4 = Con(etfa3 >= 1.3, 0.75, etfa3)

etfaf = etfa3

etq4 = 99

#log\_file.write('fill pixels with ' + str(0.75) + '\n')

pass

else:

condras = ETfs[r+p]

print condras

log\_file.write('fill pixels with ' +condras+ '\n')

#etfa4 = Con(etfa3 >= 2.0, condras, etfa3)

etfa4 = Con(etfa3 >= 1.3, condras, etfa3)

etq4 = condras

#etfa4 = Con(etfa4a >= 2.0, 0.75, etfa4a)

#etfa4 = Con(etfa4a >= 1.3, 0.75, etfa4a)

#etq6 = 0.75

#log\_file.write('fill pixels with ' + str(0.75) + '\n')

# 2. condition assign 1.05 if 1.05 < ETf < 1.3

etfaf = Con((etfa4 > 1.05) & (etfa4 < 1.3), 1.05, Con(etfa4 > 1.3, 1.3, etfa4))

etfaf.save(ETfafinal)

print '-------'

log\_file.write('-------' + '\n')

log\_file.write('ETf correction done' + '\n')

log\_file.close()

if run == 2:

# Second run of ETf cloud mask with pre/nex, earlier created median ETf, and fill 0.75

log\_file.write('create corrected ETf and QA raster with fill information for each pixel' + '\n')

l = int(len(ETfs))

medETfs = r'D:\Stornext\fewspsnfs2\FEWS\Users\Stefanie\operational\Global\_operational\Median\_InputDATA\MedianETfV5'

print medETfs

for r, ETf in enumerate(ETfs):

print ETf

print r

dek = ETf[-3:]

print dek

log\_file.write('apply correction to ' +str(ETf) + '\n')

#QAraslistaa = []

qaname = qadir +os.sep+ 'qa'+str(dek)+'.tif'

ETfaaDir = DirOut +os.sep+ str(year)

if not os.path.exists(ETfaaDir):

os.makedirs(ETfaaDir)

ETfafinal = ETfaaDir +os.sep+ 'etfca' + ETf[-5:]

# 1. condition assign 1.05 if 1.05 < ETf < 1.3 --> now last condition

#etfa = Con((Raster(ETf) > 1.05) & (Raster(ETf) < 1.3), 1.05, Raster(ETf))

# mask out all 0 values so they are not assigned to ETf raster if value is over 1.3

etfa = Raster(ETf)

#etfa = Con(etfa1 > 0, etfa1)

print etfa

# 2. condition cloud-mask when ETf >= 1.3, pre, nex, pre2, nex2

#1.pre

if ETfs[r-1] == ETfs[l-1]:

print ETfs[r-1] == ETfs[l-1]

condras = 'None'

print 'previous dekad not available - try next dekad'

log\_file.write('previous dekad not available - try next dekad' + '\n')

etfa1 = etfa

etq1 = 99

pass

else:

condras = ETfs[r-1]

print condras

log\_file.write('fill pixels with ' +condras+ '\n')

nozero\_ras1 = Con(condras > 0, condras)

etfa1 = Con((etfa >= 1.3) & (etfa != 0), nozero\_ras1, etfa)

print etfa

etq1 = condras #if last etf condition raster has the value from etq1 make it 2 in QA, else it will be 1 -> orig ETf

print etq1

#del nozero\_ras1

#1.nex

#print (r+1), l

if r == l-1:

p = 0

pp = l-1

else:

p = 1

pp = 0

if ETfs[r+p] == ETfs[pp]:

condras = 'None'

print 'next dekad not available - try 2.previous dekad'

log\_file.write('next dekad not available - try 2.previous dekad' + '\n')

etfa2 = etfa1

etq2 = 99

pass

else:

condras = ETfs[r+p]

print condras

log\_file.write('fill pixels with ' +condras+ '\n')

nozero\_ras2 = Con(condras > 0, condras)

etfa2 = Con((etfa1 >= 1.3) & (etfa1 != 0), nozero\_ras2, etfa1)

etq2 = condras

#del nozero\_ras2

#2.pre

if r == 0: #1.run

p = 2

else: #other runs

p = 1

if ETfs[r-2] == ETfs[l-p]:

condras = 'None'

print '2.previous dekad not available - try 2.next dekad'

log\_file.write('2.previous dekad not available - try 2.next dekad' + '\n')

etfa3 = etfa2

etq3 = 99

pass

else:

condras = ETfs[r-2]

print condras

log\_file.write('fill pixels with ' +condras+ '\n')

nozero\_ras3 = Con(condras > 0, condras)

etfa3 = Con((etfa2 >= 1.3) & (etfa2 != 0), nozero\_ras3, etfa2)

etq3 = condras

#del nozero\_ras3

#2.nex

if r == l-2: #vorletzter run

p = 1

pp = l-1

elif r == l-1: #letzter run

p = 0

pp = l-1

else:

p = 2

pp = 1

if ETfs[r+p] == ETfs[pp]:

condras = 'None'

print '2.next dekad not available'

log\_file.write('2.next dekad not available - try median' + '\n')

med = Raster(medETfs +os.sep+ 'med'+dek+'.tif')

etfa4 = Con((etfa3 >= 1.3) & (etfa3 != 0), med, etfa3)

#log\_file.write('if median not available - fill with 0.75' + '\n')

#etfa5 = Con(etfa4 >= 1.3, 0.75, Con(IsNull(etfa4), 0.75, etfa4))

etfa5 = Con(IsNull(etfa4), 0.75, etfa4)

# last condition assign 1.05 if 1.05 < ETf < 1.3

etfa6 = Con((etfa5 > 1.05) & (etfa5 <= 1.3), 1.05, etfa5)

etfaf = ExtractByMask(etfa6, etfa)

etq4 = 99

pass

else:

condras = ETfs[r+p]

print condras

log\_file.write('fill pixels with ' +condras+ '\n')

nozero\_ras = Con(condras > 0.0, condras)

etfa4a = Con((etfa3 >= 1.3) & (etfa3 != 0), nozero\_ras, etfa3)

etq4 = condras

log\_file.write('2.next dekad still not all filled - try median' + '\n')

med = Raster(medETfs +os.sep+ 'med'+dek+'.tif')

etfa4 = Con((etfa4a >= 1.3) & (etfa4a != 0), med, etfa4a)

#log\_file.write('if median not all filled - fill with 0.75' + '\n')

etq5 = med

etq6 = 0.75

etfa5 = Con(IsNull(etfa4), 0.75, etfa4)

# last condition assign 1.05 if 1.05 < ETf < 1.3

etfa6 = Con((etfa5 > 1.05) & (etfa5 <= 1.3), 1.05, etfa5)

etfaf = ExtractByMask(etfa6, etfa)

etfaf.save(ETfafinal)

# QA raster

##qaname = qadir +os.sep+ 'qa'+str(year) + str(dek)+'.tif'

##QAras = Con(etfaf == etfa, 1, Con(etfaf == etq1, 2, Con(etfaf == etq2, 3, Con(etfaf == etq3, 4, Con(etfaf == etq4, 5, Con(etfaf == etq5, 6, Con(etfaf == 0.75, 7)))))))

#arcpy.CopyRaster\_management(QAras,qaname, "", "", "255", "", "", "8\_BIT\_SIGNED") #CopyRasterQAras.save(qaname)

# etf, pre, nex, prepre, nexnex, med, 1.05

#QAras1 = Con(etfaaa, 1, Con(etfaaa == etq1, 2, Con(etfaaa == etq2, 3, Con(etfaaa == etq3, 4,

# Con(etfaaa == etq4, 5, Con(etfaaa == etq5, 6))))))

#QAras2 = Con(etfaaa == 1.05, 7, QAras1)

#QAras3 = Con(etfaaa == 0.75, 8)

#QAras = Con(IsNull(QAras3), 1, QAras3) # lakes are nodata, so make them 1 in QA raster

##arcpy.CopyRaster\_management(QAras, qaname, "", "", "255", "", "", "8\_BIT\_SIGNED") #CopyRasterQAras.save(qaname)

print '-------'

log\_file.write('-------' + '\n')

del etfaf

del etfa6

del etfa5

del etfa4

del etfa3

del etfa2

del etfa1

##log\_file.write('ETf QA band' + '\n')

## print QAraslist()

## for q in QAraslist():

## qaout1 = Con(q == 1, 1)

## listout = QAoutlist.append(qaout1)

## qaout2 = CellStatistics(listout, "SUM", "DATA")

## qaann = qadir +os.sep+ 'qa'+str(year)+'.tif'

## qaout2.save()

log\_file.write('ETf correction done' + '\n')

log\_file.close()

The steps are:

* If ETf > 1.3 check the previous (pre), next (nex), pre-previous (pre2), next-next (nex2) ETf grid if it has a valid value. NOTE: In the operational process the whole year is not available, therefore only checks for the previous grids are performed.
* If so far, no valid value is available check the median ETf for that dekad (created previously)
* Assign 1.05 if 1.05 < ETf < 1.3

>> Save as \\FEWS\DataPortal\data\Global\Dekadal\ETa\grids\ETfca\YYYY\etfcaYYJJJ

The corrected ETf is used in the actual ET calculation.

Historical processing:

To create the corrected ETf, 2 runs are needed. First the ETf is corrected with the PAPA algorithm only (pre, nex, prepre, nexnex) and a median dataset is created from 2003-2017 (15 years). After the median is created the corrected ETf is processed including the median.

QA band creation

The decision was made to include a layer indicating where the original calculated ETf was replaced with values from the PAPA algorithm. QA raster has following categories:

ETf as is = 1

Pre value = 2

Nex value = 3 (not checked in operational process)

Pre2 value = 4

Nex2 value = 5 (not checked in operational process)

Median ETf value = 6

Change to 1.05 = 7

>> Save as \\Stornext\fewspsnfs2\FEWS\Users\Stefanie\operational\GlobalET\ETa\grids\QAbands \YYYY\qaYYYYDDD.tif

This file needs to be added to the zip folder for data distribution.

​

Calculate Actual ET

Input datasets: corrected ETf, PET and Desertmask/water occurance mask \\FEWS\DataPortal\templates\etaV4\layers\desert\_mask.tif, desert\_coeff = 0.32 and watermask \\FEWS\DataPortal\templates\etaV4\layers\water\_occurance\_mask.tif, water\_coeff = 0.85. The kgrid value is still be to determined at this point based on evaluation efforts.

Steps in calculation are:

* Actual ETa = corrected ETf \* PET

>> Save as

\\FEWS\DataPortal\data\Global\Dekadal\ETa\grids\YYYYuc\ucetYYDDD

* Bias correct ETa with kgrid (alfalfa coefficient) ETa = ETa \*1.2
* Apply desertmask to bias corrected ETa. Extract desert areas identified in the mask and reduce by 70% 🡪 0.32.
* Adjust ETa for waterbodies with PET. Clip PET using the waterbody mask and reduce it by 15% ETa\_w = ETa \*0.85.
* Mosaic bias corrected ETa and desert-corrected ET and water-corrected ET together for final ETa grid

>> Save as \\FEWS\DataPortal\data\Global\Dekadal\ETa\geotiff\YYYY\dYYYYDDD\_modisSSEBopETv4\_actual\_mm.tif

year = sys.argv[1]

DirOut = 'D:\\Stornext\\fewspsnfs2\\FEWS\\Users\\Stefanie\\operational\\GlobalET\\ETa'

DirInData = 'D:\\Stornext\\fewspsnfs2\\WaterSmart\\Data'

try:

outPathscrap = DirOut + os.sep + 'scrapetfca'+ str(year)

if os.path.exists(outPathscrap):

shutil.rmtree(outPathscrap)

print 'deleted ' + outPathscrap

os.makedirs(outPathscrap)

else:

print 'Directory does not exist!'

os.makedirs(outPathscrap)

print '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'

print '-----create ETa --> ETf \* PET = ETa'

#logfile.write('-----create ETa --> ETf \* PET = ETa \n')

DirETfca = DirOut +os.sep+ 'grids' +os.sep+ 'ETfca' +os.sep+ str(year)

arcpy.env.workspace = DirETfca

etfcas = arcpy.ListRasters('etfca\*')

etfcas.sort()

print etfcas

DirPET = DirInData + os.sep + 'ReferenceET\\Global\_PET\\ETod'

arcpy.env.workspace = DirPET

pets = arcpy.ListRasters()

pets.sort()

print pets

OutputPathetaun = DirOut +os.sep+ 'grids' +os.sep+ str(year)+'uc'

if not os.path.exists(OutputPathetaun):

os.makedirs(OutputPathetaun)

OutputPatheta = DirOut + os.sep + 'geotiff' +os.sep+ str(year)

if not os.path.exists(OutputPatheta):

os.makedirs(OutputPatheta)

arcpy.env.workspace = outPathscrap

kgrid = 1.0

watermask = r'D:\Stornext\fewspsnfs2\FEWS\DataPortal\templates\etaV4\layers\water\_occurance\_mask.tif'

desratio = r'D:\Stornext\fewspsnfs2\FEWS\DataPortal\templates\etaV4\layers\desert\_mask.tif'

arcpy.env.cellSize = 'MINOF'

for etfca, pet in zip(etfcas, pets):

if etfca[-3:] <= '123':

if etfca[-3:] == pet.split('.')[0][-3:]:

dek = pet.split('.')[0][-3:]

print pet

ETfin = Raster(DirETfca + os.sep + etfca)

PETin = Raster(DirPET + os.sep + pet)

ucet\_file = OutputPathetaun +os.sep+ 'ucet'+str(year)[-2:]+dek

arcpy.env.extent = ETfin

etax = Int((ETfin \* PETin) +0.5)

etax.save(ucet\_file)

#Bias Correction

etax0 = Int(etax \* kgrid +0.5)

#Desert mask - dont use 1.2 because excluded in the MPI kgrid analysis and

to keep it low!

etax1 = ExtractByMask(etax, desratio)

etax2 = Int(etax1 \* 0.32 + 0.5)

#Water mask

etax3 = ExtractByMask(PETin, watermask)

etax4 = Int(etax3 \* 0.85 + 0.5)

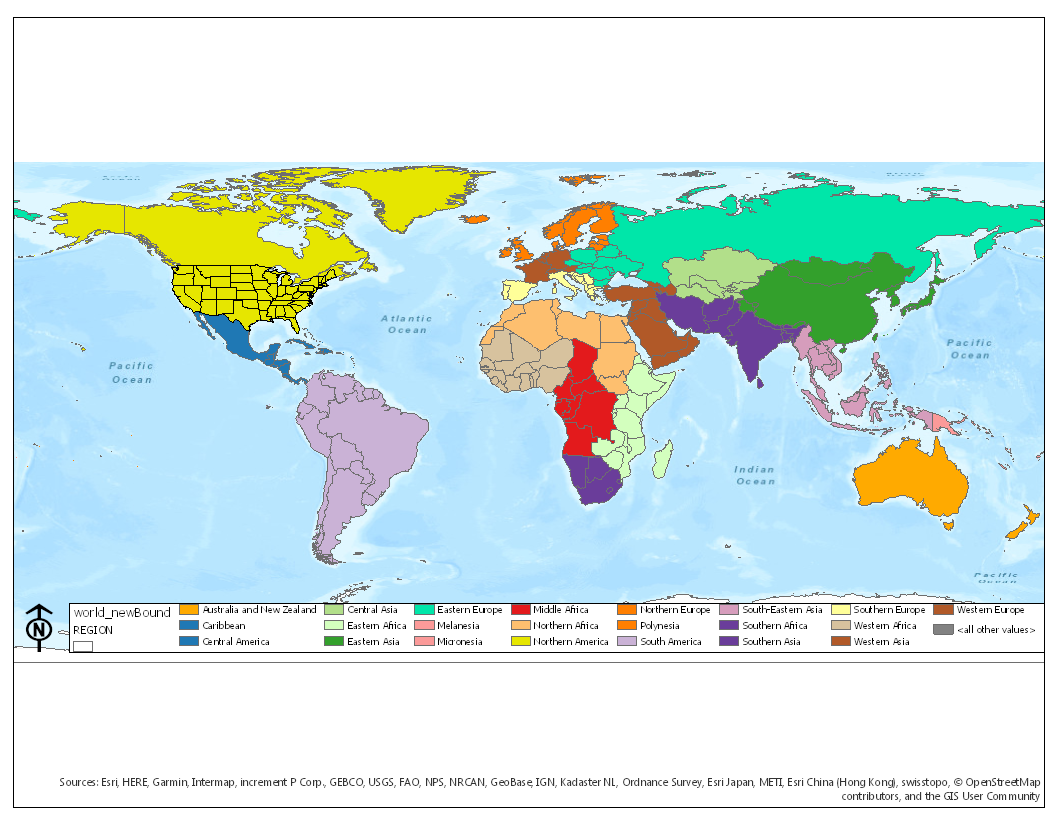
RasList = [etax0, etax2 ,etax4]

arcpy.MosaicToNewRaster\_management(RasList, OutputPatheta, 'd'+str(year)+str(dek)+'\_modisSSEBopET\_actual\_mm.tif', "#", "16\_BIT\_SIGNED", "", "1", "LAST", "FIRST") #Use 16 bit for ETa

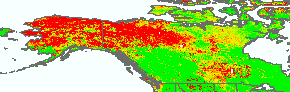
**ETa** **Explained..**

**known matters with SSEBop ET**

There are known issues with the ETa as we try to estimate ET for the whole globe with one single model. Those artifacts are listed and described below to help gain a full understanding of the data and its limitations.



Regions

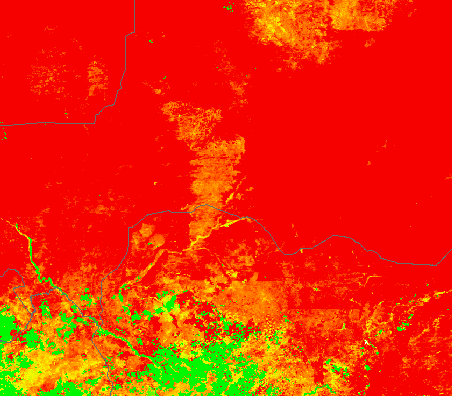
Northern America  Alaska is showing 0 ET in the winter because LST is on average 10 degree K warmer than the air temperature.

Central America

Caribbean

South America

Northern Africa

Western Africa Niger land form with a lot of ET

Eastern Africa

Middle Africa

Southern Africa

Northern Europe

Western Europe

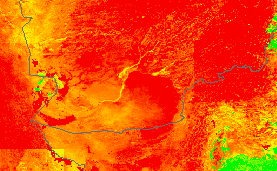
Southern Europe

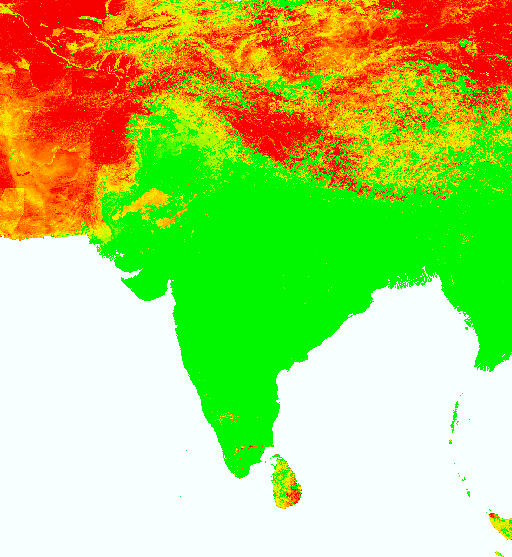
Eastern Europe

Western Asia

Central Asia

Southern Asia

 Afghanistan- too much ET showing in the south

 India has because of the monsoon season in the summer always no valid ET data. It pretty much is “gone” – too cloudy.

Eastern Asia

Southeastern Asia

Australia and New Zealand

Oceania