Python Scripts

adaboost.py

Supervised classification of multispectral images with ADABOOST.M1.

```
run scripts/adaboost [OPTIONS] filename trainShapefile
Options:
   -h         this help
   -p <list>  band positions e.g. -p [1,2,3,4]
   -L <int>   number of hidden neurons (default 10)
   -n <int>   number of nnet instances (default 50)
   -e <int>   epochs for ekf training (default 3)
```

If the input file is named

```
path/filenbasename.ext
```

then the output classification file is named

```
path/filebasename_class.ext
```

Example: Classify the first 4 principal components of an ASTER PCA image.

```
run scripts/adaboost -p [1,2,3,4] \
   imagery/AST_20070501_pca.tif imagery/train.shp
```

atwt.py

Perform panchromatic sharpening with the *à trous* wavelet transform.

```
run scripts/atwt [OPTIONS] msfilename panfilename
Options:
  -h           this help
  -p  <list>   RGB band positions to be sharpened  (default all)
                              e.g. -p [1,2,3]
  -d  <list>   spatial subset [x,y,width,height] of ms image
                              e.g. -d [0,0,200,200]
  -r  <int>    resolution ratio ms:pan (default 4)
  -b  <int>    ms band for co-registration
```

Example: Pan-sharpen the 6 NIR bands (30m) in an ASTER image with band 3 of the 3 VNIR bands (15m).

```
run scripts/atwt -p [1,2,3,4,5,6] -r 2 -b 3 \
                  imagery/msimage.tif imagery/panimage.tif
```

c_corr.py

Run the C-correction algorithm for solar illumination in rough terrain. Correction is applied only if the correlation between band intensities and the cos(γ) image is > 0.2. If a classification file is provided, the correction will be calculated on a class-specific basis.

```
run scripts/c_corr [OPTIONS] solarAzimuth solarElevation \
                                     msfilename demfilename


Options:
  -h            this help
  -p  <list>    RGB band positions to be sharpened
                  (default all)  e.g. -p [1,2,3]
  -d  <list>    spatial subset [x,y,width,height] of ms image
                               e.g. -d [0,0,200,200]
  -c  <string>  classfilename (default None)
```

The bash shell script `scripts/c-correction.sh` can be used to perform the following sequence:

1. Run a PCA on the multispectral input image.
2. Perform EM clustering on the first three PCs.
3. Run `c_corr.py` using the classified image.

```
!scripts/c-correction.sh spatialDims bandPos numEMClasses \
              solarAzimuth solarElevation msImage demImage
```

classify.py

Supervised classification of multispectral images.

```
run scripts/classify [OPTIONS] filename shapefile
Options:
  -h            this help
  -p  <list>    RGB band positions to be included
                (default all) e.g. -p [1,2,3]
  -a  <int>     algorithm  1=MaxLike
                           2=Gausskernel
                           3=NNet(backprop)
                           4=NNet(congrad)
                           5=NNet(Kalman)
                           6=Dnn(tensorflow)
                           7=SVM
  -e  <int>     number of epochs (default 100)
  -t  <float>   fraction for training (default 0.67)
  -v            use validation (reserve half of training
                   data for validation)
  -P            generate class probability image (not
                         available for MaxLike)
  -n            suppress graphical output
  -L  <list>    list of hidden neurons in each
                   hidden layer (default [10])
```

If the input file is named

```
path/filenbasename.ext
```

then the output classification file is named

```
path/filebasename_class.ext
```

the class probabilities output file is named

```
path/filebasename_classprobs.ext
```

and the test results file is named

```
path/filebasename_<classifier>.tst
```

Example: Classify the first four principal components of an ASTER image using a deep learning network with three hidden layers, 4000 epochs, and generate a class probabilities image as well as a thematic map and test results.

```
run scripts/classify -p [1,2,3,4] -P -a 6 \
           -L [10,10,10] -e 4000 \
           imagery/AST_20070501_pca.tif train.shp
```

crossvalidate.py

Parallelized cross-validation.

```
run scripts/crossvalidate [OPTIONS]  infile trainshapefile
Options:
   -h         this help
   -a  <int>  algorithm  1=MaxLike(default)
                         2=Gausskernel
                         3=NNet(backprop)
                         4=NNet(congrad)
                         5=NNet(Kalman)
                         6=Dnn(tensorflow)
                         7=SVM
  -p  <list>  band positions (default all)
                            e.g. -p [1,2,3]
  -L  <list>  hidden neurons (default [10])
                            e.g. [10,10]
  -e  <int>   epochs (default 100)
```

Prints the misclassification rate and its standard deviation.
Example: Determine the accuracy for SVM classification of the first 4 principal components of an ASTER image.

```
run scripts/ccrossvalidate -p [1,2,3,4] -a 7 \
            imagery/AST_20070501_pca.tif train.shp
```

ct.py

Determine classification accuracy and contingency table from the test results file.

```
run scripts/ct testfile
```

Example: Show results for a neural network classification of an ASTER image.

```
run scripts/ct AST_20070501_pca_NNet(Congrad).tst
```

dispms.py

Displays an RGB composite image, or two images side-by-side.

```
run scripts/dispms [OPTIONS]
Options:
  -h            this help
  -f  <string>  image filename or left-hand image filename
                (if not specified, it will be queried)
  -F  <string>  right-hand image filename, if present
  -e  <int>     left enhancement (1=linear255 2=linear
                3=linear2% saturation 4=histogram equalization
                5=logarithmic (default)
  -E  <int>     right ditto
  -p  <list>    left RGB band positions e.g. -p [1,2,3]
  -P  <list>    right ditto
  -d  <list>    left spatial subset [x,y,width,height]
                                  e.g. -d [0,0,200,200]
  -D  <list>    right ditto
  -c            right display as classification image
  -C            left ditto
  -o  <float>   overlay left image onto right with
                desired opacity 0 to 1
  -r  <list>    class labels (list of strings)
  -s  <string>  save to a file in EPS format
```

Example: Display band 4 of a LANDSAT 7 ETM+ image in a histogram equalization stretch.

```
run scripts/dispms -f imagery/L7_20010525 -e 4 -p [4,4,4]
```

Example: Display RGB composites of bands 1, 2 and 3 of two ASTER images in a linear 2% histogram stretch.

```
run scripts/dispms -f imagery/AST_20010409 -e 3 \
       -p [1,2,3] -F imagery/AST_20010730 -E 3 -P [1,2,3]
```

dwt.py

Perform panchromatic sharpening with the discrete wavelet transform.

```
run scripts/dwt [OPTIONS] msfilename panfilename
Options:
  -h            this help
  -p  <list>    RGB band positions to be sharpened
                          (default all) e.g. -p [1,2,3]
  -d  <list>    spatial subset [x,y,width,height] of ms image
                               e.g. -d [0,0,200,200]
  -r  <int>     resolution ratio ms:pan (default 4)
  -b  <int>     ms band for co-registration
```

Example: Pan-sharpen a 200×200 pixel spatial subset of an IKONOS ms image (4m, 4 bands) with the corresponding panchromatic image (1m) using band 4 of the ms image.

```
run scripts/dwt -r 4 -b 4 -d [50,100,200,200] \
                      imagery/IKON_ms imagery/IKON_pan
```

eeMad.py

A module containing utilities for running the iMad algorithm on the Google Earth Engine.

```
    from auxil.eeMad import imad, radcal, radcalbatch
```

The function `imad` implements the iteratively re-weighted MAD transformation and is called in an iterator as follows:

```
result = ee.Dictionary(inputlist.iterate(imad, first))
```

where `inputlist` is an `ee.List` of arbitrary integers with length equal to the maximum number of iterations. The variable `first` is an `ee.Dictionary`, e.g.,

```
first = ee.Dictionary({'done':ee.Number(0),
          'image':image1.addBands(image2).clip(poly),
          'allrhos': [ee.List.sequence(1,len(bands))],
          'chi2':ee.Image.constant(0),
          'MAD':ee.Image.constant(0)})
```

After iteration, the MAD variates, the chi square image and the canonical correlations can be extracted from the returned dictionary, e.g.,

```
MADs = ee.Image(result.get('MAD'))
```

Similarly, `radcal` and `radcalbatch` are iterator functions for performing radiometric normalization on two resp. several multispectral images.

eeSar_seq.py

A module for running a Jupyter notebook widget interface to the sequential SAR omnibus change detection algorithm on the Google Earth Engine. In a Jupyter notebook input cell, enter

```
from auxil.eeSar_seq import run
run()
```

to start the interface. Use the polygon map tool and the text widgets to select a region of interest, desired time period, orbit properties, etc. Leaving the relative orbit number at 0 will ignore the orbit number in the search. Press `Run` to launch the calculation. An info window will show the results of the search. Here it may be necessary to specify a unique relative orbit number to ensure equal incident angles across the sequence. In that case, re-run with the desired number. Press `Preview` to force calculation at the current scale (defined by the current Zoom level). Note that larger scales will falsify the preview image because re-sampling will change the ENL value. (The exported change maps will have the correct ENL.) Choose a destination file name for your GEE asset repository and press `Export` to save the results to GEE assets.

eeWishart.py

A module containing utilities for running the sequential SAR omnibus change detection algorithm on the Google Earth Engine.

```
from auxil.eeWishart import omnibus
```

The function `omnibus` is called, e.g., as follows:

```
result = ee.Dictionary(
  omnibus(imList, significance=0.0001,median=False))
```

where `imList` is an ee.List of dual pol, diagonal-only Sentinel-1 SAR ee.Image objects. The returned dictionary contains the change maps with keys `cmap, smap, fmap` and `bmap`. For example,

```
cmap = ee.Image(result.get('cmap')).byte()
```

ekmeans.py

Perform extended K-means clustering on a single image band.

```
run scripts/ekmeans [OPTIONS] filename
Options:
  -h           this help
  -b  <int>    band position (default 1)
  -d  <list>   spatial subset [x,y,width,height]
                             e.g. -d [0,0,200,200]
  -k  <int>    number of metaclusters (default 8)
```

Example: Cluster the first principal component of an ASTER image.

```
run scripts/ekmeans -b 1 imagery/AST_20070501_pca.tif
```

em.py

Perform Gaussian mixture clustering on multispectral imagery with the expectation maximization algorithm.

```
run scripts/em [OPTIONS] filename
Options:
  -h           this help
  -p  <list>   band positions e.g. -p [1,2,3,4,5,7]
  -d  <list>   spatial subset [x,y,width,height]
                             e.g. -d [0,0,200,200]
  -K  <int>    number of clusters (default 6)
  -M  <int>    maximum scale (default 2)
  -m  <int>    minimum scale (default 0)
  -t  <float>  initial annealing temperature (default 0.5)
  -s  <float>  spatial mixing factor (default 0.5)
  -P           generate class probabilities image
```

If the input file is named

```
          path/filenbasename.ext then
```

the output classification file is named

```
          path/filebasename_em.ext
```

and the class probabilities output file is named

```
          path/filebasename_emprobs.ext
```

Example: Cluster the first four principal components of an ASTER image with 8 clusters, generating a class probabilities file.

```
run scripts/em -p [1,2,3,4] -K 8 -P \
            imagery/AST_20070501_pca.tif
```

enlml.py

Estimation of ENL for polSAR covariance format images using a maximum likelihood method which uses the full covariance matrix (quad, dual or single).

```
run scripts/enlml [OPTIONS] filename
Options:
   -h        this help
   -n        suppress graphics output
   -d <list> spatial subset list e.g. -d [0,0,400,400]
```

Example: Estimate ENL values in a spatial subset of a quad pol RADARSAT-2 image.

```
run scripts/enlml -d {200,200,200,200] \
                myimagery/RS2_20090525.tif
```

An ENL image will be written to the same directory as the input file with `_enl` appended. A histogram of the ENL values for the chosen spatial subset is displayed, from which the mode can be determined.

gamma_filter.py

Run a gamma MAP filter over the diagonal elements of a polarimetric matrix image.

```
run scripts/gamma_filter [OPTIONS] filename enl
Options:
   -h     this help
   -d     spatial subset list e.g. -d [0,0,300,300]
```

If parallel processing is enabled by running the command

```
ipcluster start -n <number of engines>
```

in a terminal window (available in the Jupyter notebook home menu), then the script will make use of the available engines to perform the calculations. The output file has the same name as the input file with `_gamma` appended.

Example: Filter the three diagonal elements of a RADAESAT-2 quad pol image with ENL of 12.5.

```
run scripts/gamma_filter myimagery/RS2_20090829.tif 12.5
```

hcl.py

Perform agglomerative hierarchical clustering of a multispectral image.

```
run scripts/hcl [OPTIONS] filename
Options:
  -h           this help
  -p  <list>    band positions e.g. -p [1,2,3,4,5,7]
  -d  <list>    spatial subset [x,y,width,height]
                              e.g. -d [0,0,200,200]
  -k  <int>     number of clusters (default 8)
  -s  <int>     number of samples (default 1000)
```

The clustering is performed on the samples only. The resulting clusters are then used to train a maximum likelihood classifier with which all of the pixels are then clustered.

Example: Cluster the first 4 principle components of an ASTER image with 8 clusters and a sample of 2000 pixel vectors.

```
run scripts/hcl -p [1,2,3,4] -k 8 -s 2000 \
                  imagery/AST_20070501_pca.tif
```

iMad.py

Run the iteratively re-weighted MAD transformation on two co-registered multispectral images.

```
run scripts/iMad [OPTIONS] filename1 filename2
Options:
   -h           this help
   -i  <int>    maximum iterations (default 50)
   -d  <list>   spatial subset list e.g. -d [0,0,500,500]
   -p  <list>   band positions list e.g. -p [1,2,3]
   -l  <float>  regularization (default 0)
   -n           suppress graphics
   -c           append canonical variates to output
```

The images must have the same spatial and spectral dimensions. The output MAD variate file has the same format as `filename1` and is named

```
path/MAD(filebasename1-filebasename2).ext1
```

where

```
filename1 = path/filebasename1.ext1
filename2 = path/filebasename2.ext2
```

For ENVI files, `ext1` or `ext2` is the empty string. The output file band structure is as follows:

```
   MAD variate 1
   ...
   MAD variate N
   Chi square
   Image1 canonical variate 1    (optional)
   ...
   Image1 canonical variate N
   Image2 canonical variate 1
   ...
   Image1 canonical variate N
```

Example: Run iterated MAD on two LANDSAT 5 TM images.

```
run scripts/iMad -i 30 imagery/LT5_19980329_sub.tif \
                            imagery/LT5_19980516_sub.tif
```

iMadmap.py

Make a change map from iMAD variates at a given significance level.

```
run scripts/iMadmap [OPTIONS] madfile significance
Options:
   -h           this help
   -m           run a 3x3 median filter over the P-values
   -d  <list>   spatial subset list e.g. -d [0,0,500,500]
```

The `madfile` should not include the canonical variates.
Example: Create a change map from LANDSAT 5 TM MAD variates at significance level 0.0001 and with a median filter on the P-values.

```
run scripts/iMadmap -m \
imagery/MAD(LT5_19980329_sub-LT5_19980516_sub).tif 0.0001
```

kkmeans.py

Perform kernel K-means clustering on multispectral imagery.

```
run scripts/kkmeans [OPTIONS] filename
Options:
  -h            this help
  -p  <list>    band positions e.g. -p [1,2,3,4,5,7]
  -d  <list>    spatial subset [x,y,width,height]
                              e.g. -d [0,0,200,200]
  -k  <int>     number of clusters (default 6)
  -m  <int>     number of samples (default 1000)
  -n  <int>     nscale for Gauss kernel (default 1)
```

Example: Cluster the first 4 principal components of an ASTER image with 8 clusters.

```
run scripts/kkmeans -p [1,2,3,4] -k 8 \
                imagery/AST_20070501_pca.tif
```

kmeans.py

Perform K-means clustering on multispectral imagery.

```
run scripts/kmeans [OPTIONS] filename
Options:
  -h           this help
  -p  <list>   band positions e.g. -p [1,2,3,4,5,7]
  -d  <list>   spatial subset [x,y,width,height]
                              e.g. -d [0,0,200,200]
  -k  <int>    number of clusters (default 6)
```

Example: Cluster the first 4 principal components of an ASTER image with 8 clusters.

```
run scripts/kmeans -p [1,2,3,4] -k 8 \
                imagery/AST_20070501_pca.tif
```

kpca.py

Perform kernel PCA on multispectral imagery.

```
run scripts/kpca [OPTIONS] filename
Options:
  -h           this help
  -p  <list>   band positions e.g. -p [1,2,3,4,5,7]
  -d  <list>   spatial subset [x,y,width,height]
                              e.g. -d [0,0,200,200]
  -k  <int>    kernel: 0=linear, 1=Gaussian (default)
  -s  <int>    sample size for estimation of kernel
               matrix, zero for kmeans to determine
               100 cluster centers (default)
  -e  <int>    number of eigenvectors to keep (default 10)
  -n           disable graphics
```

The output file is named as the input filename with `_kpca` appended.

Example: Perform Kernel PCA with Gaussian kernel on the 6 non-thermal bands of a LANDSAT 5 TM image using 1000 samples and retaining 8 eigenvectors.

```
run scripts/kpca -p [1,2,3,4,5,7] -s 1000 -e 8 \
                        imagery/LT5_19980329.tif
```

krx.py

Kernel RX anomaly detection for multi- and hyperspectral images.

```
run scripts/krx [OPTIONS]  filename
Options:
  -h         this help
  -s  <int>   sample size for kernel matrix (default 1000)
  -n  <int>   nscale parameter for Gauss kernel (default 10)
```

Example: Kernel anomaly detection for an ASTER PCA image.

```
run scripts/krx imagey/AST_20070501_pca.tif
```

mcnemar.py

Compare two classifiers with the McNemar statistic.

```
run scripts/mcnemar testfile1 testfile2
```

Example: Compare neural network and svm classification accuracies for an ASTER image.

```
run scripts/ct AST_20070501_pca_NNet(Congrad).tst \
                                AST_20070501_pca_SVM.tst
```

meanshift.py

Segment a multispectral image with the mean shift algorithm.

```
run scripts/meanshift [OPTIONS] filename
Options:
  -h            this help
  -p  <list>    band positions e.g. -p [1,2,3,4,5,7]
  -d  <list>    spatial subset [x,y,width,height]
                              e.g. -d [0,0,200,200]
  -r  <int>     spectral bandwidth (default 15)
  -s  <int>     spatial bandwidth (default 15)
  -m  <int>     minimum segment size (default 30)
```

Example: Segment a spatial subset of the first 4 principal components of an ASTER image with spatial bandwidth 15, spectral bandwidth 30,and minimum segment size 10.

```
run scripts/meanshift -p [1,2,3,4] -d [500,450,200,200] \
   -s 15 -r 30 -m 10 imagery/AST_20070501_pca.tif
```

mmse_filter.py

Run an MMSE filter over all elements of a polarimetric matrix image.

```
run scripts/mmse_filter [OPTIONS] filename enl
Options:
   -h     this help
   -d     spatial subset list e.g. -d [0,0,300,300]
```

The output file has the same name as the input file with _mmse appended.

Example: Filter the elements of a RADARSAT-2 quad pol image with ENL of 12.5.

```
run scripts/mmse_filter myimagery/RS2_20090829.tif 12.5
```

mnf.py

Calculate minimum noise fraction image.

```
run scripts/mnf  [OPTIONS] filename
Options:
  -h            this help
  -p  <list>    band positions e.g. -p [1,2,3,4,5,7]
  -d  <list>    spatial subset [x,y,width,height]
                              e.g. -d [0,0,200,200]
  -n            disable graphics
```

The output file is named as the input filename with _mnf appended.
Example: Perform MNF transformation on the 6 non-thermal bands of a LANDSAT 5 TM image.

```
run scripts/mnf -p [1,2,3,4,5,7] imagery/LT5_19980329.tif
```

pca.py

Perform principal components analysis on an image.

```
run scripts/pca  [OPTIONS] filename
Options:
  -h          this help
  -p  <list> band positions e.g. -p [1,2,3,4,5,7]
  -d  <list> spatial subset [x,y,width,height]
                          e.g. -d [0,0,200,200]
  -r  <int>  number of components for reconstruction (default 0)
  -n          disable graphics
```

The output files are named as the input filename with _pca or _recon appended.
Example: Perform PCA on the 6 non-thermal bands of a LANDSAT 5 TM image and reconstruct from the first three principal components.

```
run scripts/pca -p [1,2,3,4,5,7] -r 3 \
                         imagery/LT5_19980329.tif
```

plr.py

Probabilistic label relaxation postprocessing of supervised classification images.

```
run scripts/plr [OPTIONS]  classProbFileName
Options:
  -h         this help
  -i  <int>  number of iterations (default 3)
```

Example: Perform PLR on the class probability file generated from a supervised classification of principal components of a LANDSAT 5 TM image.

```
run scripts/plr imagery/LT5_19980329_pca_classprobs.tif
```

The result (classified image) is written to

```
imagery/LT5_19980329_pca_classprobs_plr.tif
```

radcal.py

Automatic radiometric normalization of two multispectral images.

```
run scripts/radcal [OPTIONS] iMadFile [fullSceneFile]
Options:
   -h           this help
   -t  <float>  P-value threshold (default 0.95)
   -d  <list>   spatial subset e.g. -d [0,0,500,500]
   -p  <list>   band positions  e.g. -p [1,2,3]
```

Spatial subset MUST match that of `iMadFile`, spectral dimension of `fullSceneFile`, if present, MUST match those of the target and reference images. The `iMadFile` is assumed to be of the form

```
    path/MAD(filename1-filename2).ext
```

and the output file is named

```
    path/filename2_norm.ext.
```

That is, it is assumed that `filename1` is the reference and `filename2` is the target and the output retains the format of the `imMadFile`. A similar convention is used to name the normalized full scene, if present:

```
          fullSceneFile_norm.ext
```

Note that, for ENVI format, `ext` is the empty string.

readshp.py

Read shapefiles and return training/test data and class labels.

```
    from auxil import readshp
    Xs, Ls, numclasses, classlabels  = readshp.readshp(
        <train shapefile>, <imagefilename>,
        <list of band positions>)
```

This is a helper module for reading shapefiles generated from ENVI ROIs, together with the image file used to define the ROIs, and returning labeled train/test data, the number of classes and their labels.

registerms.py

Perform image-image registration of two optical/infrared images.

```
from auxil import registerms
registerms.register(reffilename,warpfilename,dims,outfile)
    or
run auxil/registermy  [OPTIONS] reffilename warpfilename
Options:
   -h        this help
   -d <list> spatial subset list e.g. -d [0,0,500,500]
   -b <int>  band to use for warping (default 1)
```

Choose a reference image, the image to be warped and, optionally, the band to be used for warping (default band 1) and the spatial subset of the reference image. The reference image should be smaller than the warp image (i.e., the warp image should overlap the reference image completely) and its upper left corner should be near that of the warp image:

```
----------------------
|   warp image
|
|  --------------------
|  |
|  |  reference image
|  |
```

The reference image (or spatial subset) should not contain zero data. The warped image `warpfilename_warp` will be trimmed to the spatial dimensions of the reference image.
Example: Register two LANDSAT 7 ETM+ ENVI format images using VNIR band 4:

```
run auxil/registerms -d [100,100,600,600] -b 4 \
                 imagery/LE7_20010626 imagery/LE7_20010829
```

The warped file will be named `LE7_20010829_warp` and clipped to a 600 × 600 spatial subset.

registersar.py

Perform image-image registration of two polarimetric SAR images in covariance matrix form.

```
from auxil import registersar
registersar.register(reffilename,warpfilename,dims,outfile)
    or
run auxil/registersar  [OPTIONS] reffilename warpfilename
Options:
   -h        this help
   -d <list> spatial subset list e.g. -d [0,0,500,500]
```

Choose a reference image, the image to be warped and the spatial subset of the reference image. The span images (trace of the covariance matrix) will be used for registration. The reference image should be smaller than the warp image (i.e., the warp image should overlap the reference image completely) and its upper left corner should be near that of the warp image:

```
----------------------
|   warp image
|
|  --------------------
|  |
|  |  reference image
|  |
```

The reference image (or spatial subset) should not contain zero data. The warped image `warpfilename_warp` will be trimmed to the spatial dimensions of the reference image.

Example: Register two RADARSAT-2 quad pol images:

```
run auxil/registersar -d [100,100,600,600] \
    myimagery/RS2_20090525.tif myimagery/RS2_20090618.tif
```

The warped file will be named `RS2_20090618_warp.tif` and clipped to a 600×600 spatial subset.

rx.py

RX anomaly detection for multi- and hyperspectral images.

```
run/scripts rx [OPTIONS] filename
Options:
  -h        this help
```

Example: Anomaly detection for an ASTER PCA image.

```
run scripts/rx imagery/AST_20070501_pca.tif
```

sar_seq.py

Perform sequential change detection on multi-temporal, polarimetric SAR imagery with the sequential omnibus algorithm.

```
run scripts/sar_seq [OPTIONS]  infiles* outfile enl
Options:
  -h          this help
  -m          run 3x3 median filter on p-values prior to
              thresholding
  -d  <list>  spatial subset of first image to which all files
              are to be co-registered (default no co-
              registration)
  -s  <float> significance level (default 0.0001)
```

If the `-d` option is not chosen, it is assumed that all images are co-registered and have the same spatial/spectral dimensions. The `infiles*` inputs are the full paths to the input files:

```
/path/to/infile_1 /path/to/infile_1 ... /path/to/infile_k
```

The `outfile` should be without path. The change maps will be written to same directory as `infile_1` with filenames

```
    outfile_cmap: interval of most recent change, 1 band
    outfile_smap: interval of first change, 1 band
    outfile_fmap: number of changes, 1 band
    outfile_bmap: changes in each interval, (k-1)-band
```

`enl` is the equivalent number of looks. If `IPython` engines have been enabled, the co-registration and P-value calculations will be distributed among them.

If no spatial subsetting (and hence no co-registration) is required, the bash shell script `scripts/run_sar_seq.sh` can be used to gather all of the SAR images in a directory and run the algorithm:

```
run_sar_seq.sh pattern imdir enl significance
```

Example: Run the algorithm on all image file names containing the string `S1A` in the directory `imagery` for an ENL of 12 and significance 0.0001.

```
!scripts/run_sar_seq.sh S1A imagery/ 12 0.0001
```

scatterplot.py

Display a scatterplot.

```
run scripts/scatterplot [OPTIONS] filename1 [filename2] \
                                          band1 band2
Options:
   -h          this help
   -d <list>   spatial subset
   -n <int>    samples (default 10000)
   -s <string> save in eps format
```

Example: Show a scatterplot of bands 1 vs 2 of an ASTER image in ENVI format.

```
run scripts/scatterplot imagery/AST_20070501 1 2
```

som.py

A 3D Kohonen self-organizing map for multispectral image visualization in an RGB cube.

```
run scripts/som [OPTIONS] filename
Options:
  -h            this help
  -p  <list>    band positions e.g. -p [1,2,3,4,5,7]
  -d  <list>    spatial subset [x,y,width,height]
                              e.g. -d [0,0,200,200]
  -s  <int>     sample size (default 10000)
  -c  <int>     cube side length (default 5)
```

Example: Determine the SOM for all 9 bands of an ASTER image in ENVI format with cube size of $6 \times 6 \times 6$.

```
run scripts/som -c 6  imagery/AST_20070501
```

subset.py

Perform spatial and/or spectral subsetting of a multispectral image.

```
from auxil import subset
subset.subset(filename,dims,pos,outfile)
          or
run auxil/subset [OPTIONS] filename
Options:
   -h           this help
   -d <list>    spatial subset list e.g. -d [0,0,500,500]
   -p <list>    band position list e.g. -p [1,2,3]
```

Example: Spectrally subset a LANDSAT 7 ETM+ image to eliminate thermal band 6.

```
run auxil/subset -p [1,2,3,4,5,7] imagery/LE7_20010525
```

JavaScript on the GEE Code Editor

The two main change detection algorithms discussed in Chapter 9 and coded in Python, namely iMAD and sequential omnibus, are also runnable directly from the GEE code editor using the JavaScript programs described in this section. The code is shared on the GEE and can be cloned from the Google Earth repository with

```
git clone https://earthengine.googlesource.com/users
                                   /mortcanty/changedetection
```

imad_run

A simple front end for running the iMAD and automatic radiometric normalization algorithms on bi-temporal optical/infrared images. Change maps, together with the original and normalized images are exported to assets. The iMAD convergence details and regression coefficients are exported to Google Drive.

omnibus_run

A front end for running the sequential omnibus change detection algorithm on time series of Sentinel-1 images. Change maps are exported to assets and can be displayed with `omnibus_view`. A temporally de-speckled image consisting of the mean of all the images in the sequence is appended to the change maps to serve as background for animation; see below.

omnibus_view

A viewer for exported sequential omnibus change maps. The layered maps are color coded and an animated change image derived from the bmap change map can be exported to Google Drive.

imad

JavaScript modules for running the iMAD and radiometric normalization algorithms. The functions `radcal` and `imad` are exported.

omnibus

JavaScript modules for the sequential omnibus algorithm. The function `omnibus` is exported.

utilities

Various JavaScript utility modules, including a function `makevideo` for generating change animations.