Automatic Adaptive Signature Generalization (AASG) in R*

Matthew P. Dannenberg[1], Christopher Hakkenberg[2] and Conghe Song[1,2]
[1]Department of Geography
[2]Curriculum for the Environment and Ecology
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599
Contact: mdannenb@live.unc.edu; csong@email.unc.edu

August 15, 2016

TABLE OF CONTENTS

1 Introduction

This document provides an overview of the program `aasg.r`, which contains the necessary functions to automatically generate land cover datasets from remotely sensed imagery using the automatic adaptive signature generalization approach. For full details of the methodology, see Gray & Song (2013) and Dannenberg, Hakkenberg & Song (2016). In the AASG approach, image histograms from two image dates (one of which is associated with a high quality reference classification) are used to identify stable sites from which to extract class spectral signatures for a new classification (Figure 1).

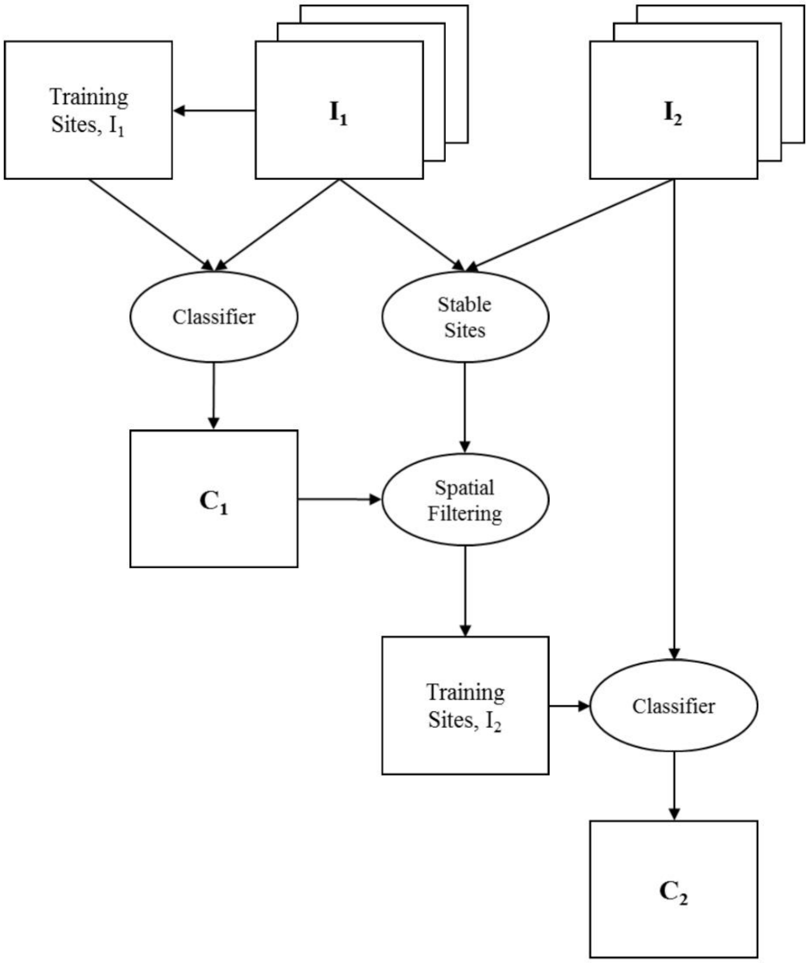

Figure 1. Workflows for the AASG method, in which classified maps C_1 and C_2 are created from input images I_1 and I_2. The AASG approach automatically generates unique training sites for I_2 from stable locations in the scene. (Image from Dannenberg, Hakkenberg & Song, 2016)

2 Before you start

The program `aasg.r` requires several packages that must be installed prior to using the AASG functions. These include:

`raster`: Geographic Data Analysis and Modeling
`rgdal`: Bindings for the Geospatial Data Abstraction Library
`randomForest`: Breiman and Cutler's random forests for classification and regression

In RStudio, these packages can be downloaded by selecting the Tools menu, then "Install packages…" or by installing from the command line:

```
> install.packages("raster")
> install.packages("rgdal")
> install.packages("randomForest")
```

Once these packages are installed, you can then load the functions contained in `aasg.r` by using the `source` command:

```
> source("C:/.../aasg.r")
```

This will load the necessary libraries and functions. Now you just need some data!

3 Data inputs and preparation

At its most basic, the AASG method requires only three data inputs (Figure 1):

- A high quality reference classification (C_1)
- A reference image from which the reference classification was generated (I_1)
- An image from which to generate a new classification (I_2)

The default in `aasg.r` is to assume that I_1 and I_2 are six band Landsat TM/ETM+ images with bands arranged in order from lowest to highest wavelength. However, most functions will accept other formats as well, such as a multi-temporal image stack (see §5 for an example). A digital elevation model (DEM) can also be included as an optional input.

Prior to running the functions in `aasg.r`, all input layers must have the same extent and spatial resolution and be in the same projection and coordinate system. AASG does not do this for you![1] However, you can easily use R's `raster` package to make simple image transformations. See the `raster` package documentation (https://cran.r-project.org/web/packages/raster/raster.pdf) for a comprehensive list and descriptions of the image processing functions, and §5 for an example of data pre-processing using the `raster` package.

---

[1] With one exception: if the resolution, projection or extent of the optional DEM does not match the two images and the reference classification, then AASG will reproject and resample the DEM to match the other input layers.

4 Functions in `aasg.r`

`layer.add`	Add spectral vegetation indices to the image stack

Description

Calculates spectral vegetation indices for a six band Landsat TM/ETM+ image, including:

- Normalized difference vegetation index (NDVI)
- Simple ratio vegetation index (SR)
- Two versions of the normalized difference water index (NDWI) with each of the two Landsat shortwave infrared bands
- Enhanced vegetation index (EVI)
- The first three tasseled cap components (KT1-KT3)

Optionally, NDVI focal statistics (mean and variance within 3- and 7-pixel windows) can be calculated and added to the image stack. This function will not work if the image does not have six bands, and it is the user's responsibility to ensure that the bands are in the proper order (from lowest to highest wavelength).

Usage

```
layer.add(x, focal=FALSE)
```

Arguments

`x`	a six band raster stack, with bands arranged from lowest to highest wavelength
`focal`	logical. If `TRUE`, focal statistics are calculated on the NDVI image. If `FALSE` (default), focal statistics are not calculated.

Value

A raster stack with at least 14 bands. The first six bands are the original six bands of `x`, and the remaining bands are (in order) NDVI, SR, NDWI1 (from TM band 5), NDWI2 (from TM band 7), EVI, KT1, KT2, KT3. If focal=TRUE, four additional bands will follow: NDVI variance within 3-pixel moving windows, NDVI variance within 7-pixel moving windows, NDVI mean within 3-pixel moving windows, and NDVI mean within

7-pixel moving windows. Note that vegetation indices must be calculated from surface reflectance, not from digital numbers. Since the tasseled cap indices (KT1-3) are linear transformations of the original bands, they do not add additional information. Users should therefore use either the original six bands or the three tasseled cap indices, but not both together.

Examples

```
# Read in six band Landsat TM surface reflectance image
nc2001 <- brick("2001.261.tif") * (1/10000)

# Calculate spectral vegetation indices
nc2001.svi <- layer.add(nc2001)
```

`ndvi.add` Add NDVI to the image stack

Description

Same as `layer.add`, but calculates NDVI only.

Usage

```
ndvi.add(x)
```

Arguments

`x`	a six band raster stack, with bands arranged from lowest to highest wavelength

Value

A raster stack with 7 bands. The first six bands are the original six bands of `x`, and the seventh is an NDVI image.

Examples

```
# Read in six band Landsat TM surface reflectance image
nc2001 <- brick("2001.261.tif") * (1/10000)
```

```
# Calculate NDVI
nc2001.ndvi <- ndvi.add(nc2001)
```

`tasseled.cap` Calculate tasseled cap indices

Description

Generates the first three tasseled cap components (brightness, greenness, and “wetness”) using the Kauth-Thomas (KT) transformation.

Usage

```
tasseled.cap(x)
```

Arguments

`x`	a six band raster stack, with bands arranged from lowest to highest wavelength

Value

A raster stack with 3 bands (KT1, KT2, and KT3).

Examples

```
# Read in six band Landsat TM surface reflectance image
nc2001 <- brick("2001.261.tif") * (1/10000)

# Calculate tasseled cap indices
nc2001.kt <- tasseled.cap(nc2001)
```

`topo.add` Calculate topographic indices

Description

Calculates five topographic variables (using the `terrain` function in R's `raster` package) from a digital elevation model:

- Slope (in degrees)
- Aspect
- Topographic position index (TPI)
- Terrain ruggedness index (TRI)
- Roughness

See the `raster` documentation for a description of the variables calculated in the `terrain` function.

Usage

```
topo.add(x)
```

Arguments

`x`	a digital elevation model in raster format

Value

A raster stack with 6 bands. The first band is the original elevation raster of `x`, and the remaining bands are (in order) slope, aspect, TPI, TRI, and roughness.

Examples

```
# Read in DEM from Shuttle Radar Topography Mission
dem <- raster("SRTM_sub")
dem[dem==-32767] = NA

# Calculate topographic indices
topo.data <- topo.add(dem)
```

`make.mask` Generate stable site mask

Description

Identifies stable sites between two image dates (I_1 and I_2) based on normalized image difference histograms and creates a class-specific mask for extraction of class spectral signatures in the function `make.training`.

Usage

```
make.mask(img1, img2, refc, c=NULL, dif.layer1=3,
dif.layer2=3)
```

Arguments

`img1`	raster layer of the reference image
`img2`	raster layer of the image to be classified
`refc`	raster layer of the reference classification (corresponding to `img1`)
`c`	the number of standard deviations from the mean of the image difference histogram to regard as “stable.” If `c=NULL` (the default), then `c` is calculated on a per-class basis. A global `c` can also be specified as a scalar value by the user (e.g., `c=0.25`).
`dif.layer1`	Integer. Indicates which band from `img1` to use for the image difference histogram. By default, `dif.layer1=3` (corresponding to red band in a Landsat TM image).
`dif.layer2`	Integer. Indicates which band from `img2` to use for the image difference histogram. By default, `dif.layer2=3` (corresponding to red band in a Landsat TM image). In most cases, `dif.layer1` and `dif.layer2` will be the same, though there may be some exceptions (e.g., when `img2` is a multiseason image stack).

Value

A raster layer of class-specific stable sites. All non-stable pixels have values of NA.

Examples

```
# Read in six band Landsat TM surface reflectance images
# from two dates
nc2001 <- brick("2001.261.tif") * (1/10000)
nc2010 <- brick("2010.222.tif") * (1/10000)

# Read in reference classification for img1
class2001 <- raster("nlcd2001_sub")

# Generate stable site mask
site.mask <- make.mask(nc2001, nc2010, class2001)
```

`make.training` Extract class signatures using stable site mask

Description

Extracts class signatures from I_2 (the image to be classified) and C_1 (the reference classification) using the stable site mask generated by `make.mask`.

Usage

```
make.training(trmask, img2, refc)
```

Arguments

`trmask`	raster layer of the stable site mask (see `make.mask`)
`img2`	raster layer of the image to be classified
`refc`	raster layer of the reference classification (corresponding to `img1`)

Value

A data frame of training data for classifying I_2. The first column contains the class label (extracted from the reference classification) and the remaining columns are the image covariates (extracted from I_2).

Examples

```
# Read in six band Landsat TM surface reflectance images
# from two dates
nc2001 <- brick("2001.261.tif") * (1/10000)
nc2010 <- brick("2010.222.tif") * (1/10000)

# Read in reference classification for img1
class2001 <- raster("nlcd2001_sub")

# Generate stable site mask
site.mask <- make.mask(nc2001, nc2010, class2001)

# Extract training data
training.data <- make.training(site.mask, nc2010,
     class2001)
```

`aasg` Run the full AASG model

Description

Run all functions of the AASG method and output a new classification.

Usage

```
aasg(img1, refc, img2, c=NULL, dem=NULL, method="RF",
probs=FALSE, dif.layer=c(3,3), diags=FALSE, focal=FALSE,
svi=TRUE, quiet=FALSE)
```

Arguments

`img1`	raster layer of the reference image
`img2`	raster layer of the image to be classified
`refc`	raster layer of the reference classification (corresponding to `img1`)

`c`	the number of standard deviations from the mean of the image difference histogram to regard as “stable.” If `c=NULL` (the default), then `c` is calculated on a per-class basis. In future releases, `c` can also be specified as a scalar value by the user (e.g., `c=0.25`), but this feature is not currently supported.
`dem`	optional raster layer containing a DEM (and additional topographic layers, if desired). By default, if a DEM raster is provided, `topo.add` will be run and will add the topographic variables from the `terrain` function to the topographic raster stack.
`method`	string. Supported methods include `"RF"` for random forest (the default) and `"MLC"` for a maximum likelihood classifier (using the linear discriminant analysis, `LDA`, function from the `MASS` package). Additional methods may be added in future releases.
`probs`	logical. If TRUE, posterior class probabilities for each pixel will be included in the output from the random forest. If FALSE (default), only the hard classification will be provided.
`dif.layer`	vector. Indicates which bands from `img1` and `img2` to use for the image difference histograms. By default, `dif.layer=c(3,3)` (corresponding to red band in a Landsat TM image). If the images are not six band Landsat TM/ETM+ images, then this may need to be changed.
`diags`	logical. Not currently supported, but in future release setting `diags=TRUE` will provide some diagnostics from the random forest model.
`focal`	logical. If `TRUE`, focal statistics are calculated on the NDVI image and added to the image stack. If `FALSE` (default), focal statistics are not calculated.
`svi`	logical. If TRUE, vegetation indices are calculated and added to the image stack (but only if it is a six band image).

`quiet`	logical. If FALSE (default), status messages are periodically displayed in the R terminal. If TRUE, no status messages are displayed unless an error or warning is encountered.

Value

A list containing the classified image, the random forest model object, the stable site mask, and (if probs=TRUE) the poster probabilities of class membership.

Examples

```
# Read in six band Landsat TM surface reflectance images
# from two dates
nc2001 <- brick("2001.261.tif") * (1/10000)
nc2010 <- brick("2010.222.tif") * (1/10000)

# Read in reference classification for img1
class2001 <- raster("nlcd2001_sub")

# Classify 2010 image using AASG
new.class <- aasg(nc2001, class2001, nc2010)

# Display classified image
plot(new.class$Classification)

# Display stable site mask
plot(new.class$StableSites)

# View random forest model
new.class$rf
```

5 An Example Using Landsat and NLCD Data

Below is an example of the `aasg.r` methodology applied to multi-temporal Landsat imagery in the Research Triangle region of North Carolina, with goal being to update a land cover classification from 2001 for the year 2011. Inputs into the model include:

Input	**Date**	**Source**	**Derived Layers**
C_1	2001	National Land Cover Dataset	
I_1	2001 (Day 261)	Landsat TM	
I_2	2011 (Day 97)	Landsat TM	Tasseled Cap
	2010 (Day 222)	Landsat TM	Tasseled Cap
	2011 (Day 305)	Landsat TM	Tasseled Cap
		Shuttle Radar Topography Mission	Slope, UAA, modified UAA, TWI

Let’s begin:

```
## Load AASG functions and set working directory
source("C:/Users/Matthew/Documents/Code/aasg.r")
setwd("C:/Users/Matthew/Documents/AASG_Data")


## Load reference classification
class2001 <- raster("nlcd2001_sub")

# Remove class 21 (Developed, open space) since it is based
# partly on ancillary data and is not spectrally distinct
class2001[class2001==21] = NA

# Get extent of reference classification for clipping
e <- extent(class2001)


## Load topographic data and derived variables
dem <- raster("SRTM_sub"); dem[dem==-32767] = NA
uaa <- raster("UAA.asc")
muaa <- raster("mUAA.asc")
slope <- raster("Slope.asc")
twi <- raster("TWI.asc")
```

```
topo.data <- stack(dem, slope, uaa, muaa, twi)
names(topo.data) <- c("ELEV", "SLOPE", "UAA", "mUAA", "TWI")
topo.data <- crop(topo.data, e)


## Load and process reference image (Summer 2001)
nc2001.sum <- brick("2001.261.tif")

# Re-scale to reflectance units and restrict to [0, 1]
nc2001.sum = nc2001.sum * (1/10000)
nc2001.sum[nc2001.sum>1] = NA
nc2001.sum[nc2001.sum<0] = 0

# Resample to same resolution as NLCD data
nc2001.sum = resample(nc2001.sum, class2001, method="bilinear")


## Load and process 2011 images
nc2011.spr <- brick("2011.097.tif")
nc2011.spr = crop(nc2011.spr, e) * (1/10000)
nc2011.spr[nc2011.spr>1] = NA
nc2011.spr[nc2011.spr<0] = 0

nc2011.sum <- brick("2010.222.tif")
nc2011.sum = crop(nc2011.sum, e) * (1/10000)
nc2011.sum[nc2011.sum>1] = NA
nc2011.sum[nc2011.sum<0] = 0

nc2011.aut <- brick("2011.305.tif")
nc2011.aut = crop(nc2011.aut, e) * (1/10000)
nc2011.aut[nc2011.aut>1] = NA
nc2011.aut[nc2011.aut<0] = 0

nc2011.spr = resample(nc2011.spr, class2011, method="bilinear")
nc2011.sum = resample(nc2011.sum, class2011, method="bilinear")
nc2011.aut = resample(nc2011.aut, class2011, method="bilinear")

# Calculate first 3 tasseled cap components for each image
kt2011.spr <- tasseled.cap(nc2011.spr)
kt2011.sum <- tasseled.cap(nc2011.sum)
kt2011.aut <- tasseled.cap(nc2011.aut)
```

```
# Stack all 2011 inputs
input2011 <- stack(kt2011.spr, kt2011.sum, kt2011.aut,
     topo.data)


## Create stable site mask based on red band of summer images
StableSites = make.mask(nc2001.sum, nc2011.sum, class2001)


## Run AASG to create new 2011 classification
newclass2011 <- aasg(kt2001.sum, class2001, input2011,
     StableSites=StableSites, probs=FALSE)

# Write to TIF
writeRaster(newclass2011$Classification,
     filename="2011.class.aasg.tif", format="GTiff",
     overwrite=TRUE)

# Examine RF model
newclass2011$rf
rf = newclass2011$rf
dotchart(sort(rf$importance[,(dim(rf$importance)[2]-1)]))
```

6 References

Dannenberg, M. P., Hakkenberg, C. R., & Song, C. (2016). Consistent classification of Landsat time series with an improved automatic adaptive signature generalization algorithm. *Remote Sensing*. In Press

Gray, J., & Song, C. (2013). Consistent classification of image time series with automatic adaptive signature generalization. *Remote Sensing of Environment*, *134*, 333–341. http://doi.org/10.1016/j.rse.2013.03.022