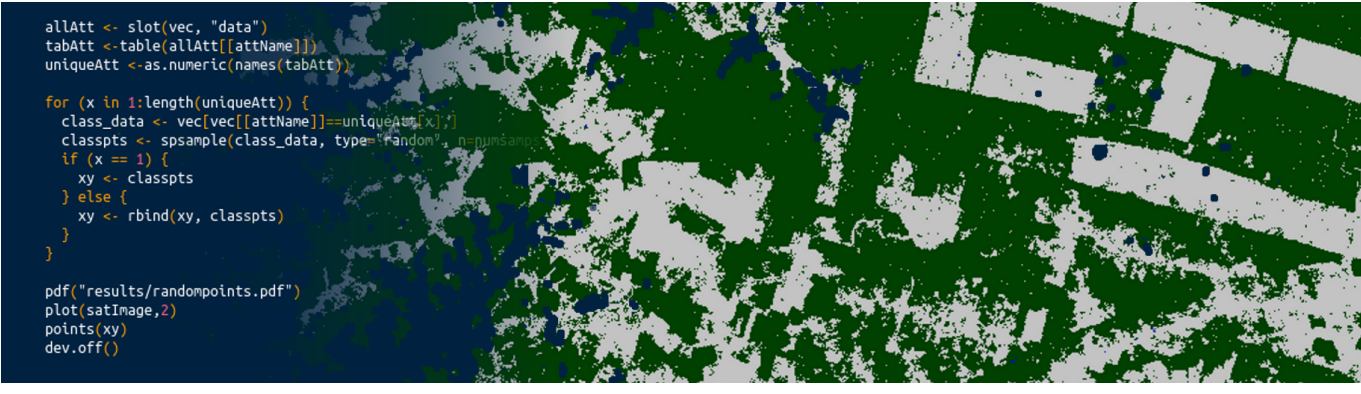

venue year

R and Remote Sensing

Introduction to Remote Sensing with R

author cover | institute cover

R and spatial data handling

- Import of raster data
- query data and plot spatial data
- compute vegetation indices
- we will use Landsat and MODIS data
- further functions will be taught soon after

R and Remote Sensing

the number of R packages specifically designed for spatial data analysis is increasing steadily:

- raster, rgdal: raster data analysis
- maptools, sp: vector analysis
- spatstat, GeoR: point statistics
- spgrass6: interface to GRASS
- RSAGA: interface to SAGA GIS
- read the respective manual pages for further information
- don't hesitate to look at the command syntax!

more information at:
http://geodacenter.asu.edu/r-spatial-projects/
http://r-spatial.sourceforge.net/
http://www.geog.uu.nl/~pebesma/wun/
http://r-spatial.sourceforge.net/gallery/

R Libraries

load libraries after installation

```
1 library(raster)
2 library(rgdal)
```

only if packages are already installed: *install.packages("name")*

Raster import

```
b3 <- raster("/path_to_your_raster/raster.tif", band=3)
b4 <- raster("/path_to_your_raster/raster.tif", band=4)

summary(b3) #check the data
image(b3)   #display raster
plot(b3)
spplot(b3)
```

Raster import

```
b3 <- raster("/path_to_your_raster/raster.tif", band=3)
b4 <- raster("/path_to_your_raster/raster.tif", band=4)

summary(b3) #check the data
image(b3)   #display raster
plot(b3)
spplot(b3)
```

image, plot und spplot commands are similar but have different (dis)advantages

Some useful functions

str(obj)	prompts the structure of an object (esp. useful for S4 Objects (spatial objects))
names(obj)	gives you the names within an object
head(obj)	shows the first rows of an object - good for first data integrity check
summary(obj)	provides summary statistics of your object

NDVI calculation

```
1 ndvi<- (band_4 - band_3)/(band_4+band_3)
```

NDVI calculation

```
1 ndvi<- (band_4 - band_3)/(band_4+band_3)
```

however if the data set is bigger, R provides more efficient ways to do it:

NDVI calculation

```
ndvi<- (band_4 - band_3)/(band_4+band_3)
```

however if the data set is bigger, R provides more efficient ways to do it:

```
# we define a NDVI functions, so we don't have to write it several times
fun_ndvi<- function(nir, red) {(nir-red) /(nir+red)}
```

NDVI calculation

```
ndvi<- (band_4 - band_3)/(band_4+band_3)
```

however if the data set is bigger, R provides more efficient ways to do it:

```
# we define a NDVI functions, so we don't have to write it several times
fun_ndvi<- function(nir, red) {(nir-red) /(nir+red)}
```

```
# NDVI
ndvi<-overlay(band_4, band_3, fun=function(nir, red){(nir-red)/(nir+red)})
#or
ndvi<-overlay(band_4, band_3, fun=fun_ndvi)
```

NDVI calculation

```
ndvi<- (band_4 - band_3)/(band_4+band_3)
```

however if the data set is bigger, R provides more efficient ways to do it:

```
# we define a NDVI functions, so we don't have to write it several times
fun_ndvi<- function(nir, red) {(nir-red) /(nir+red)}
```

```
# NDVI
ndvi<-overlay(band_4, band_3, fun=function(nir, red){(nir-red)/(nir+red)})
#or
ndvi<-overlay(band_4, band_3, fun=fun_ndvi)
```

and you could combine it with an automatic export command (here: SAVI calculation):

```
# SAVI computation with automatic data export
savi<-overlay(band_4, band_3, fun=function(nir, red){(nir-red)/(nir+red + 0.5)*(1+0.5)}, filename="savi.tif",
    format="GTiff")
```

first data manipulation

If the raster data has negativ values and the NDVI does not display properly, do:

```
# copy band 3
b3.a <- b3

#set all values below 0 to NA
b3.a[b3.a < 0] <- NA
```

and redo the NDVI analysis.

first data manipulation

If the raster data has negativ values and the NDVI does not display properly, do:

```
# copy band 3
b3.a <- b3

#set all values below 0 to NA
b3.a[b3.a < 0] <- NA
```

and redo the NDVI analysis. You might also want to modify the range of the NDVI values:

```
# copy the ndvi data
ndvi.a <- ndvi

#set all values below 0 to NA
ndvi.a[ndvi.a < 0] <- NA
```


Compute and compare indices
try also other indices e.g. MSAVI

More details on spatial R functions

General recommendations:

1. use all the time the scripting functions (R script including comments!)

General recommendations:

1. use all the time the scripting functions (R script including comments!)
2. choose meaningful names: bad: my_script.R better: ndvi_landsat.R

General recommendations:

1. use all the time the scripting functions (R script including comments!)
2. choose meaningful names: bad: my_script.R better: ndvi_landsat.R
3. add meta information on top of your script: who did it, when, which R version, package versions

General recommendations:

1. use all the time the scripting functions (R script including comments!)
2. choose meaningful names: bad: my_script.R better: ndvi_landsat.R
3. add meta information on top of your script: who did it, when, which R version, package versions
4. what is the script doing, what is the result/aim

General recommendations:

1. use all the time the scripting functions (R script including comments!)
2. choose meaningful names: bad: my_script.R better: ndvi_landsat.R
3. add meta information on top of your script: who did it, when, which R version, package versions
4. what is the script doing, what is the result/aim
5. and again: do execessive commenting in your script!

some more useful pages:

- http://rseek.org/ for general R search
- http://www.inside-r.org/
- http://www.r-bloggers.com
- http://cran.r-project.org/web/packages/raster/vignettes/Raster.pdf
- http://www.csiro.au/resources/Spatial-Point-Patterns-in-R
- http://www.mo-seph.com/node/296
- http://jeffreybreen.wordpress.com/2010/10/22/incremental-improvements-to-nightlights-mapping-thanks-to-r-bloggers/
- http://r-nold.blogspot.de/2012/06/talking-about-elevation-one-can-also.html
- http://procomun.wordpress./2012/02/20/maps_with_r_2/
- http://procomun.wordpress.com/2011/06/17/raster-cmsaf-and-solar/

Rastertypes in R

in R 3 different raster types exist:

Raster	one-layer raster
Brick	multi-layer raster from one file
Stack	Stack of 'Bricks' und 'Raster' objects from different sources

Rastertypes in R

in R 3 different raster types exist:

Raster one-layer raster

Brick multi-layer raster from one file

Stack Stack of 'Bricks' und 'Raster' objects from different sources

```
# alternative to raster()
b3 <- raster("/path_to_your_raster/raster.tif", band=3)
allband<- brick("/path_to_your_raster/raster.tif")

#stack images or drop one
b34<-stack(b3,allband)
b34<-addLayer(b34, b4)
b5<-dropLayer(b34, 2)

summary(b5) #check the data
```

Rastertypes in R

in R 3 different raster types exist:

- **Raster** one-layer raster
- **Brick** multi-layer raster from one file
- **Stack** Stack of 'Bricks' und 'Raster' objects from different sources

```
# alternative to raster()
b3 <- raster("/path_to_your_raster/raster.tif", band=3)
allband<- brick("/path_to_your_raster/raster.tif")

#stack images or drop one
b34<-stack(b3,allband)
b34<-addLayer(b34, b4)
b5<-dropLayer(b34, 2)

summary(b5) #check the data
```

to crop your data set use crop(), but only a rectangle will be returned, with mask() you also get an irregular shape

Raster Statistics

cellStats	Statistics for each layer (mean, min, max ect)
summary	Statistics overview based on sample of pixels (can be changed using: maxsamp)
count	extract number of defined values
crosstab	crosstable of 2 raster
unique	list of all existing values
zonal	zonal statistics for part of a raster
quantile	compute the quantile

```
summary(ndvi)
Cells:  35819702
NAs  :  0
Min.    0.0000
1st Qu. 0.3832
Median  0.4222
Mean    0.4107
3rd Qu. 0.4528
Max.    0.5748
summary based on a sample of 5000 cells, which is 0.0139587984288647 % of all cells
```

Raster query

```
b3_mod <- b3 #just generate a copy of band 3

b3_mod[b3_mod > 50] <- NA #replace all values larger than 50 with NA (not available)
```

Raster query

```
b3_mod <- b3 #just generate a copy of band 3

b3_mod[b3_mod > 50] <- NA #replace all values larger than 50 with NA (not available)
```

or:

```
function1 <- function(x){x[x > 50] <- NA; return(x)} #define a function that does the same

calc(b3_mod,fun=function1) #execute the function within calc()
```

```
overlay(b3_mod,fun=function1) #execute the command within overlay(); overlay() can deal with more than 1
    variable
```

Raster query

```
b3_mod <- b3 #just generate a copy of band 3

b3_mod[b3_mod > 50] <- NA #replace all values larger than 50 with NA (not available)
```

or:

```
function1 <- function(x){x[x > 50] <- NA; return(x)} #define a function that does the same

calc(b3_mod,fun=function1) #execute the function within calc()
```

```
overlay(b3_mod,fun=function1) #execute the command within overlay(); overlay() can deal with more than 1
    variable
```

Advantage of a function?

Raster query

```
b3_mod <- b3 #just generate a copy of band 3

b3_mod[b3_mod > 50] <- NA #replace all values larger than 50 with NA (not available)
```

or:

```
function1 <- function(x){x[x > 50] <- NA; return(x)} #define a function that does the same

calc(b3_mod,fun=function1) #execute the function within calc()
```

```
overlay(b3_mod,fun=function1) #execute the command within overlay(); overlay() can deal with more than 1
    variable
```

Advantage of a function? you just have to define once a complex function and can easily add it everywhere in your analysis

Raster Logic

```
#replace value - attention: data set will be overwritten (do a copy beforehand)
img[img>15]<-15

#List with Pixel values
list<-img[band1==5 & band2==15]

# Raster values converted to: TRUE | FALSE
# be aware of the capital 'W'!
Which(band1==5 & band2!==15)

#List with all cell-IDs which have a certain logic:
Which(band1>122, cells=TRUE)
```

Raster Algebra

You can perform various raster manipulation tasks:

- calculates values for a raster using a formula
- either using calc() or overlay()
- computing average of all surrounding pixels
- either using focal() or focalFilter()
- compute Moran or Geary Indices (moran())

using:

```
1  formula1 <- function(x) { x * 10 }
2  rc1 <- calc(r, formula1)
3
4  overlay(x, fun, unstack=TRUE)
```

Look into the raster manual for further information and examples

save/export Raster

```
1  # export raster - overwrite if already existent
2  writeRaster(ndvi,datatype='FLT4S', filename='NDVI.tif', format="GTiff", overwrite=TRUE)
3
4  # assign new values to layer
5  setValues(img, values, layer)
6
7  #set all values of first layer to 100
8  img <- setValues(img,values=100,layer=1)
9
10 #equivalent to
11 img[[1]] <- 100
12
13 #export image to GoogleEarth
14 KML(img, filename, col=rainbow(255), maxpixels=100000)
```

save/export Raster - data depth

```
1 dataType(img)<-value
```

Datatype definition	minimum possible value	maximum possible value
LOG1S	FALSE (0)	TRUE (1)
INT1S	-127	127
INT1U	0	255
INT2S	-32,767 32,767	
INT2U	0	65,534
INT4S	-2,147,483,647	2,147,483,647
INT4U	0	4,294,967,296
FLT4S	-3.4e+38	3.4e+38
FLT8S	-1.7e+308	1.7e+308

save/export Raster file

name	long name
raster	R-raster
SAGA	SAGA GIS
IDRISI	IDRISI
BIL	Band by Line
BSQ	Band Sequential
BIP	Band by Pixel
ascii	Arc ASCII
ADRG	ARC Digitized Raster Graphics
BMP	MS Windows Device Independent Bitmap
BT	VTP .bt (Binary Terrain) 1.3 Format
EHdr	ESRI .hdr Labelled
ELAS	ELAS
ENVI	ENVI .hdr Labelled
ERS	ERMapper .ers Labelled
GSBG	Golden Software Binary Grid (.grd)
GTiff	GeoTIFF
GTX	NOAA Vertical Datum .GTX
HDF4Image	HDF4 Dataset
HFA	Erdas Imagine Images (.img)
ILWIS	ILWIS Raster Map
INGR	Intergraph Raster
NITF	National Imagery Transmission Format
NTv2	NTv2 Datum Grid Shift
PCIDSK	PCIDSK Database File
PNM	Portable Pixmap Format (netpbm)
RMF	Raster Matrix Format
RST	Idrisi Raster A.1
SAGA	SAGA GIS Binary Grid (.sdat)
SGI	SGI Image File Format 1.0