# LABORATORIUM SENSORY OBSERWACJI SATELITARNEJ

## LABORATORIUM 5

## *Przetwarzanie zobrazowań satelitarnych w R*

---

W środowisku R znajdziemy kilka możliwych do wykorzystania pakietów pozwalających na przetwarzanie danych rastrowych, w tym zobrazowań satelitarnych. Na laboratorium wykorzystamy pakiet `raster`, który zawiera wszystkie podstawowe funkcjonalności GIS i jest łatwy w użyciu.

Włącz program R x64, zainstaluj wymagane pakiety, a następnie wczytaj pakiet `raster` i `rgdal`:

```
install.packages("rgdal")
install.packages("raster")
library(rgdal)
library(raster)
```

Ustaw swój folder roboczy (ten w którym znajdują się materiały do laboratorium): File->Change dir…

Teraz możesz wczytać plik gdansk_2015.tif (zobrazowanie Landsat 8 OLI) za pomocą funcji `stack`:

```
gdansk = stack("sciezka_do_pliku.tif")
```

jeżeli nie wiesz jak używać danej funkcji, użyj pomocy wpisując znak zapytania przed nazwą funkcji, np. `?stack`

Wyświetl obiekt `gdansk` w konsoli wpisując jego nazwę. Sprawdź jakie są rzeczywiste zakresy wartości pikseli w poszczególnych kanałach obiektu `gdansk` za pomocą funkcji `summary()`. Możesz także wyświetlić wszystkie kanały za pomocą funkcji `plot()`. Aby odwołać się do konkretnego kanału obiektu `gdansk` użyj operatora `$` i nazw konkretnych kanałów, przykładowo w postaci `obiekt$nazwaKanal.1`; nazwy kanałów znajdziesz:

(1) wyświetlając obiekt `gdansk` w konsoli (pierwszy krok tego akapitu),

(2) za pomocą funkcji `names()`,

(3) po wpisaniu nazwy obiektu z operatorem `$` możesz użyć klawisza `TAB` aby zobaczyć podpowiedzi,

(4) alternatywnie całą strukturę obiektu `gdansk` możesz przejrzeć funkcją `str()`.

Spróbuj teraz wyświetlić piąty kanał obiektu `gdansk` za pomocą funkcji `plot()` (0,5 pkt)

Pakiet raster zawiera funkcję `plotRGB` do tworzenia kompozycji barwnych, zapoznaj się z jej stroną pomocy `?plotRGB`

Przykładowo `plotRGB(gdansk, r=4, g=3, b=2, scale=2^16)` wyświetli obraz w kolorach rzeczywistych (definicja kanałów w tabeli na następnej stronie), ale obraz nie ma rozciągniętego histogramu – jest za ciemny do prawidłowej interpretacji. Wykorzystaj `plotRGB` aby wyświetlić kompozycję color infrared (CIR) z rozciągniętym histogramem – skorzystaj ze strony pomocy i tabeli poniżej (0,5 pkt)

| Kanał | Nazwa | Długość fali [nm] |
|---|---|---|
| 1 | Costal/Aerosol | 443 |
| 2 | Blue | 482 |
| 3 | Green | 561 |
| 4 | Red | 655 |
| 5 | NIR | 865 |
| 6 | SWIR-1 | 1609 |
| 7 | SWIR-2 | 2201 |

Proste działania na obiektach rastrowych wykorzystujących pakiet `raster` możesz wykonać przy użyciu podstawowych operatorów matematycznych, przykładowo `gdansk/2^16` wykona dzielenie wszystkich kanałów na $2^{16}$,czyli 65536, `log(gdansk$gdansk_2015.1, 10)` obliczy logarytm o podstawie 10 z kanału pierwszego, zaś `gdansk$gdansk_2015.1 > 10000` wykona mapę zgodnie z wyrażeniem logicznym. Przetestuj używając funkcji plot() powyższe przykłady obliczeń.

Aby przeliczyć zobrazowanie landsat 8 na reflektancję należy, zgodnie z metadanymi produktu, przemnożyć każdy kanał przez `0.00002` i dodać do wyniku `−0.1`. Wykonaj te obliczenia i zapisz wynik jako nowy obiekt `reflGdansk`.

Wykorzystując informacje poznane powyżej oblicz wskaźnik NDVI dla obiektu `reflGdansk` i wyświetl go funkcją `plot()`. (1 pkt)

Obliczenia w pakiecie raster można także wykonywać za pomocą bardziej złożonych, wcześniej zdefiniowanych funkcji. Przykładowo, jeżeli chcemy obliczyć współczynnik zmienności (CV), czyli stosunek odchylenia standardowego do średniej dla każdego pixela zdefiniujemy nową funkcję o jednym parametrze x, który będzie przekazywał siedmioelementowy wektor reflektancji z pikseli zobrazowania do dalszych obliczeń:

```
getCV = function(x)
{
sdX = sd(x)
meanX = mean(x)
CV = sdX/meanX
return(CV)
}
```

Funkcja ta jest zawarta w pliku `funkcje.r`. Otwórz ten plik w programie Notepad++ i edytuj w taki sposób, aby wynik był zwracany jako wartość procentowa, a nie bezwymiarowa – jak jest obecnie. Następnie wczytaj plik `funkcje.r` do programu R za pomocą `source("funkcje.r")`.

Teraz możesz już przetestować działanie funkcji na przykładowym dziesięcioelementowym wektorze `test=c(1,2,3,4,5,6,7,8,9,10)` , czyli:

```
getCV(test)
```

Prawidłowy wynik to `55.04819`.

Następnie wykorzystując funkcję `calc` wykonaj obliczenia funkcją `getCV` dla wszystkich pikseli `reflGdansk`, zapisz wynik jako nowy obiekt i wyświetl go funkcją `plot()`. (1 pkt)

Aby poznać jakie są wartości rastra w konkretnych punktach powinno się znać współrzędne punktów zgodnie z odwzorowaniem rastra. W obiekcie `punktyTest` (wczytałeś go już razem z plikiem funkcje.r) znajdują się współrzędne trzech punktów dla wody, lasu i miasta, wyświetl ten obiekt w konsoli. Jak widzisz punkty posiadają współrzędne geograficzne (dokładnie jest to układ WGS 84), jest to inne odwzorowanie, niż odwzorowanie rastra reflGdansk, które możesz sprawdzić za pomocą funkcji proj4string().

Aby nadać prawidłowe odwzorowanie punktom musisz najpierw zmienić je w obiekt przestrzenny. Wykonać to możesz definiując, które kolumny odpowiadają współrzędnym na osi x i y, przykładowo:

```
coordinates(obiekt)=~nazwaX+nazwaY
```

zastanów się co podstawić pod `obiekt`, `nazwaX` i `nazwaY` aby nadać współrzędne obiektowi `punktyTest` i wykonaj to.

Gdy obiekt ma już współrzędne musisz nadać mu układ odwzorowania. Sprawdź jaki numer EPSG ma WGS 84 i podstaw go pod XXXX do poniższego polecenia:

```
proj4string(punktyTest)=CRS("+init=epsg:XXXX")
```

Porównaj teraz proj4string dla `reflGdansk` i dla punktów. Aby wykonać transformację współrzędnych geograficznych z WGS84 do UTM (jak w obiekcie `reflGdansk`), użyj:

```
punktyUTM = spTransform(punktyTest,proj4string(XXXX))
```

jako XXXX podaj obiekt, który posiada docelowy układ współrzędnych UTM.

Możesz teraz wyświetlić punkty jako nową warstwę funkcją `points()` o ile jakieś okno zobrazowania Gdańska jest już otwarte.

Teraz możesz pobrać wartości pikseli w punktach użyj funkcji `extract` (sprawdź jakie ma argumenty za pomocą `?`, nie używaj argumentów opcjonalnych) i przypisz wynik do nowego obiektu o dowolnej nazwie. (0,5 pkt)

Współrzędne punktu dla rastra wyświetlonego w oknie wykresu (funkcją `plot()` ) możesz poznać używając:

```
locator(1)
```

a następnie kliknąć w punkt na zdjęciu którego współrzędne chcesz sprawdzić. Przykładowo współrzędne ujścia Motławy do Bałtyku to około x= 348314 i y= 6031570 metrów UTM34N.

Wykorzystując informacje poznane w instrukcji powyżej uzupełnij funkcje `getMask` w pliku funkcje.r. Funkcja ta powinna pobierać wartości reflektancij ze wskazanego miejsca na zobrazowaniu a następnie za pomocą algorytmu spectral angular mapper (sam) wykonywać maskę podobnych powierzchni. **Po każdej zmianie w pliku funkcje.r musisz go wczytać za pomocą `source()`.** Funkcja powinna wykonywać kolejno:

1. wyświetlanie rastra `reflRaster` funkcją `plotRGB()`

2. (już zaimplementowane) pobieranie współrzędnych punktu za pomocą `locator` i zapis jako obiekt `data.frame` – nazwa obiektu wynikowego to `coord`

3. nadanie odpowiedniego układu współrzędnych obiektowi `coord`

4. pobranie wartości reflektancji w miejscach wskazanych przez `coord` i zapisanie wyniku jako nowy obiekt o nazwie `wzor`

5. Uzupełnij funkcję `sam0` (podfunkcja funkcji `sam`) aby policzyć kąt spektralny między dwoma wektorami wejściowymi, gdzie pierwszy wektor to `a`, a drugi wektor to `b`:

$$sam = \cos^{-1}\left(\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}\right)$$

6. Wykorzystanie wcześniej zdefiniowanej funkcji `sam()` aby obliczyć kąt spektralny między obiektem `wzor`, a każdym pikselem zobrazowania `reflRaster` i zapisanie wyniku jako nowy obiekt `wynikSAM`

7. wykonanie wyrażenia logicznego które wskaże obiekty o kącie mniejszym niż zdefiniowana przez użytkownika wartość `prog` .

8. wynik wyrażenia z punktu 6 powinien być zapisany jako obiekt `wynikMask`, wyświetlony za pomocą `plot()` i zwrócony za pomocą `return()` .
Za wykonanie całego zadania jest 1,5 pkt, za fragment odpowiednia część