

10-multispectral\_sensor

shuxin

2023-02-24



# Contents

<b>1</b>	<b>About</b>	<b>5</b>
1.1	.....	5
<b>2</b>		<b>7</b>
<b>3</b>	<b>(PART*) Base R</b>	<b>9</b>
<b>4</b>	<b>Base R you have to know</b>	<b>11</b>
4.1	.....	11
4.2	R? .....	11
4.3	Vector( ) .....	12
4.4	Lists( ) .....	14
4.5	Matrices( ) .....	14
4.6	Data Frame( ) .....	18
<b>5</b>	<b>(PART*) Tidyverse</b>	<b>21</b>
	<b>All about tidyverse</b>	<b>23</b>
5.1	.....	23
<b>6</b>	<b>(PART*) bookdown</b>	<b>25</b>
	<b>how to write a book</b>	<b>27</b>

<b>7</b>	<b>Raster</b>	<b>29</b>
7.1	Resolution . . . . .	29
7.2	CRS . . . . .	31
7.3	Reference . . . . .	31
<b>8</b>	<b>Multispectral sensors</b>	<b>33</b>

# Chapter 1

# About

shuxin R

R

R

Arcgis Qgis ENVI SNAP

githu

，

emo

## 1.1



## Chapter 2

shuxin R





## Chapter 3

### (PART\*) Base R



# Chapter 4

## Base R you have to know

### 4.1

### 4.2 R?

1992                  Ross Ihaka Robert Gentleman



Figure 4.1: *Ross Ihaka and Robert Gentleman, the creators of R.*

R                          R                          R                          R                          The

R Base Package

B R Python stata R

### 4.3 Vector( )

Vector R

```
1 5 5

x <- c(1,2,3,4,5)
x
#> [1] 1 2 3 4 5
```

c() 1 2 3 4 5 <- x 1 2 3 4 5 x R

R c()

R google :, R

```
x <- c(1:5)
x
#> [1] 1 2 3 4 5
```

```
vector typeof()

typeof(x)
#> [1] "integer"
```

```
length,length()

length(x)
#> [1] 5
```

```
R seq()

seq(1, 9, 0.5)
#> [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5
#> [15] 8.0 8.5 9.0
```

1 9 0.5 3 ?? Console

??seq

Help :: :: package::function cli an

generation

Description Usage,seq(...) seq (), A

vector

```
# Vector of logical values
log_values <- c(TRUE, FALSE, TRUE, FALSE)

log_values
#> [1] TRUE FALSE TRUE FALSE
```

```
# R
```

```
fruits <- c("banana", "apple", "orange", "mango", "lemon", "50")
fruits
#> [1] "banana" "apple" "orange" "mango" "lemon" "50"
```

```
[] brackets, fruits "banana" "mango"
```

```
fruits[c(1,4)]
#> [1] "banana" "mango"
```

```
fruits[1:4]
#> [1] "banana" "apple" "orange" "mango"
```

```
"banana"
```

```
fruits[-1]
#> [1] "apple" "orange" "mango" "lemon" "50"
```

```
sort,
```

```
fruits <- c("banana", "apple", "orange", "mango", "lemon")
numbers <- c(13, 3, 5, 7, 20, 2)

sort(fruits) # Sort a string
#> [1] "apple" "banana" "lemon" "mango" "orange"
sort(numbers) # Sort numbers
#> [1] 2 3 5 7 13 20
```

## 4.4 Lists( )

R list()

```
thislist <- list(
  a = c("apple", "banana", "cherry"),
  b = c(1,2,5,6,7,9),
  c = c(TRUE, FALSE, TRUE)
)
# Print the list
thislist
#> $a
#> [1] "apple" "banana" "cherry"
#>
#> $b
#> [1] 1 2 5 6 7 9
#>
#> $c
#> [1] TRUE FALSE TRUE
```

```
typeof(thislist)
#> [1] "list"
```

```
length(thislist)
#> [1] 3
```

## 4.5 Matrices( )

(column) (row) matrix()

```
# Create a matrix
thismatrix <- matrix(c(1,2,3,4,5,6), nrow = 3, ncol = 2)

# Print the matrix
thismatrix
#>      [,1] [,2]
#> [1,]    1    4
#> [2,]    2    5
#> [3,]    3    6
```

NOTE: `c()`

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)

thismatrix
#>      [,1]      [,2]
#> [1,] "apple"  "cherry"
#> [2,] "banana" "orange"
```

Access Matrix Items You can access the items by using `[ ]` brackets. The first number “1” in the bracket specifies the row-position, while the second number “2” specifies the column-position:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)

thismatrix[1, 2]
#> [1] "cherry"
```

The whole row can be accessed if you specify a comma after the number in the bracket:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)

thismatrix[2,]
#> [1] "banana" "orange"
```

The whole column can be accessed if you specify a comma before the number in the bracket:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)

thismatrix[,2]
#> [1] "cherry" "orange"
```

Access More Than One Row More than one row can be accessed if you use the `c()` function:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange", "grape", "pineapple", "pear", "melon"),
nrow = 2, ncol = 4)

thismatrix[c(1,2),]
#>      [,1]      [,2]      [,3]      [,4]
#> [1,] "apple"  "orange" "pear"   "melon"
#> [2,] "banana" "grape"  "pineapple" "cherry"
```

Access More Than One Column More than one column can be accessed if you use the `c()` function:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange", "grape", "pineapple", "pear"), nrow=3, byrow=TRUE)

thismatrix[, c(1,2)]
#>      [,1]      [,2]
#> [1,] "apple"  "orange"
#> [2,] "banana" "grape"
#> [3,] "cherry" "pineapple"
```

Add Rows and Columns Use the `cbind()` function to add additional columns in a Matrix:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange", "grape", "pineapple", "pear"), nrow=3, byrow=TRUE)

newmatrix <- cbind(thismatrix, c("strawberry", "blueberry", "raspberry"))

# Print the new matrix
newmatrix
#>      [,1]      [,2]      [,3]      [,4]
#> [1,] "apple"  "orange"  "pear"   "strawberry"
#> [2,] "banana" "grape"   "melon" "blueberry"
#> [3,] "cherry" "pineapple" "fig"   "raspberry"
```

Use the `rbind()` function to add additional rows in a Matrix:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange", "grape", "pineapple", "pear"), nrow=3, byrow=TRUE)

newmatrix <- rbind(thismatrix, c("strawberry", "blueberry", "raspberry"))

# Print the new matrix
newmatrix
#>      [,1]      [,2]      [,3]
#> [1,] "apple"  "orange"  "pear"
#> [2,] "banana" "grape"   "melon"
#> [3,] "cherry" "pineapple" "fig"
#> [4,] "strawberry" "blueberry" "raspberry"
```

Remove Rows and Columns Use the `c()` function to remove rows and columns in a Matrix:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange", "mango", "pineapple"), nrow=3, byrow=TRUE)
```



```
#Remove the first row and the first column
thismatrix <- thismatrix[-c(1), -c(1)]

thismatrix
#> [1] "mango"      "pineapple"
```

Check if an Item Exists To find out if a specified item is present in a matrix, use the %in% operator:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)

"apple" %in% thismatrix
#> [1] TRUE
```

Number of Rows and Columns Use the dim() function to find the number of rows and columns in a Matrix:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)

dim(thismatrix)
#> [1] 2 2
```

Matrix Length Use the length() function to find the dimension of a Matrix:

```
thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)

length(thismatrix)
#> [1] 4
```

Combine two Matrices Again, you can use the rbind() or cbind() function to combine two or more matrices together:

```
# Combine matrices
Matrix1 <- matrix(c("apple", "banana", "cherry", "grape"), nrow = 2, ncol = 2)
Matrix2 <- matrix(c("orange", "mango", "pineapple", "watermelon"), nrow = 2, ncol = 2)

# Adding it as a rows
Matrix_Combined <- rbind(Matrix1, Matrix2)
Matrix_Combined
#>      [,1]      [,2]
#> [1,] "apple"  "cherry"
#> [2,] "banana" "grape"
#> [3,] "orange" "pineapple"
```

```
#> [4,] "mango" "watermelon"

# Adding it as a columns
Matrix_Combined <- cbind(Matrix1, Matrix2)
Matrix_Combined
#>      [,1]      [,2]      [,3]      [,4]
#> [1,] "apple" "cherry" "orange" "pineapple"
#> [2,] "banana" "grape" "mango" "watermelon"
```

## 4.6 Data Frame( )

`data.frame()`

```
# Create a data frame
Data_Frame <- data.frame (
  Training = c("Strength", "Stamina", "Other"),
  Pulse = c(100, 150, 120),
  Duration = c(60, 30, 45)
)

# Print the data frame
Data_Frame
#>   Training Pulse Duration
#> 1 Strength   100      60
#> 2  Stamina   150      30
#> 3   Other   120      45
```

Use the `summary()` function to summarize the data from a Data Frame:

```
summary(Data_Frame)
#>   Training      Pulse      Duration
#> Length:3      Min.   :100.0   Min.   :30.0
#> Class :character 1st Qu.:110.0   1st Qu.:37.5
#> Mode  :character Median :120.0   Median :45.0
#>          Mean   :123.3   Mean   :45.0
#>          3rd Qu.:135.0   3rd Qu.:52.5
#>          Max.   :150.0   Max.   :60.0
```

`[]`   `[[ ]]`   `$`

```
Data_Frame[1]
#> Training
#> 1 Strength
#> 2 Stamina
#> 3 Other

Data_Frame[["Training"]]
#> [1] "Strength" "Stamina" "Other"

Data_Frame$Training
#> [1] "Strength" "Stamina" "Other"
```

```
rbind()
```

```
# Add a new row
New_row_DF <- rbind(Data_Frame, c("Strength", 110, 110))

# Print the new row
New_row_DF
#> Training Pulse Duration
#> 1 Strength 100 60
#> 2 Stamina 150 30
#> 3 Other 120 45
#> 4 Strength 110 110
```

```
cbind()
```

```
# Add a new column
New_col_DF <- cbind(New_row_DF, Steps = c(1000, 6000, 2000, 5000))

# Print the new column
New_col_DF
#> Training Pulse Duration Steps
#> 1 Strength 100 60 1000
#> 2 Stamina 150 30 6000
#> 3 Other 120 45 2000
#> 4 Strength 110 110 5000
```

```
rbind() R
```

```
Data_Frame1 <- data.frame (
  Training = c("Strength", "Stamina", "Other"),
  Pulse = c(100, 150, 120),
  Duration = c(60, 30, 45)
```

```

)

Data_Frame2 <- data.frame (
  Training = c("Stamina", "Stamina", "Strength"),
  Pulse = c(140, 150, 160),
  Duration = c(30, 30, 20)
)

New_Data_Frame <- rbind(Data_Frame1, Data_Frame2)
New_Data_Frame
#>   Training Pulse Duration
#> 1 Strength   100      60
#> 2 Stamina   150      30
#> 3   Other   120      45
#> 4 Stamina   140      30
#> 5 Stamina   150      30
#> 6 Strength   160      20

```

`cbind()`      R

```

Data_Frame3 <- data.frame (
  Training = c("Strength", "Stamina", "Other"),
  Pulse = c(100, 150, 120),
  Duration = c(60, 30, 45)
)

Data_Frame4 <- data.frame (
  Steps = c(3000, 6000, 2000),
  Calories = c(300, 400, 300)
)

New_Data_Frame1 <- cbind(Data_Frame3, Data_Frame4)
New_Data_Frame1
#>   Training Pulse Duration Steps Calories
#> 1 Strength   100      60  3000      300
#> 2 Stamina   150      30  6000      400
#> 3   Other   120      45  2000      300

```

## Chapter 5

### (PART\*) Tidyverse



# All about tidyverse

## 5.1





## Chapter 6

### (PART\*) bookdown



# how to write a book

<https://www.youtube.com/watch?v=9i0ElncHGRg&t=905s>



# Chapter 7

## Raster

### 7.1 Resolution

What is resolution of a satellite image? How can we understand it? Generally speaking we can say there are **three** different resolutions for a satellite image in Geoscience.

#### 7.1.1 Spatial resolution

Spatial resolution is the detail in pixels of an image. High spatial resolution means more detail and a smaller grid cell size. Whereas, lower spatial resolution means less detail and larger pixel size. Overall, spatial resolution describes the quality of an image and how detailed objects are in an image. If the grid cells are smaller, this means the spatial resolution has more detail with more pixels.



High Spatial Resolution



Medium Spatial Resolution



Low Spatial Resolution

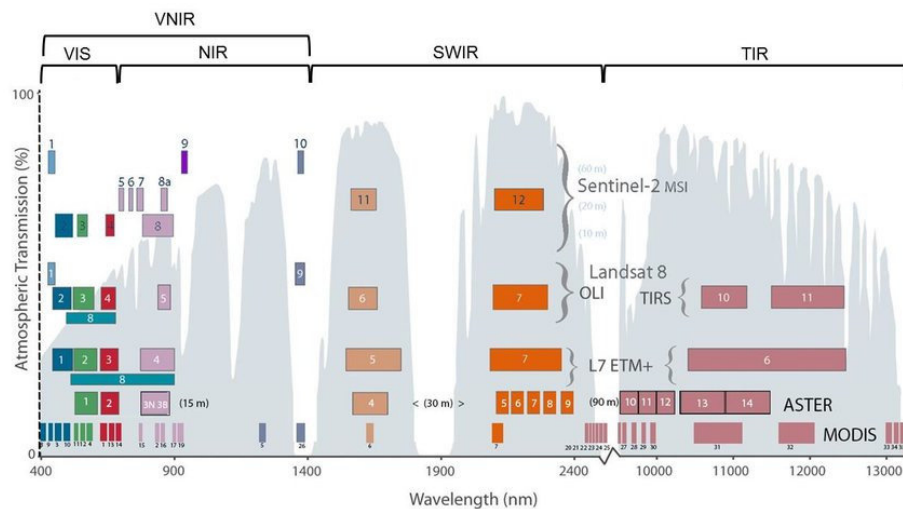
### 7.1.2 Temporal resolution

Same definition of temporal resolution can be applied to polar orbiting satellites. But defining it more precisely, temporal resolution for a polar orbiting satellite is the amount of time that the satellite takes to revisit and recapture a particular site. It is also commonly referred to as a satellite's revisit period.

Mission	Number of satellites	Temporal resolution (single satellite)	Temporal resolution (constellation)
SENTINEL-1	2	12 days	6 days
SENTINEL-2	2	10 days	5 days
LANDSAT 7	1	16 days	16 days
WorldView-3	1	1 day	1 day
Terra	1	16 days	16 days

### 7.1.3 Spectral resolution

Spectral resolution is determined by the width of each band in a wavelength. The more bands in an image, the more complex the color will be.



## 7.2 CRS

## 7.3 Reference

- <http://modern-rstats.eu/index.html#note-to-the-reader>





## Chapter 8

# Multispectral sensors

In this section I'll introduce you to three well-known high-resolution multispectral sensor satellites in this section: MODIS, Landsat, and Sentinel-2.