General R (Reading and Writing Images)

R Basics

Data Classes

.

. `"hey" "I'm a string"`

. `TRUE FALSE` not

Data Types

- `vector`
- `matrix`
- `data.frame`
- `array`

`nifti`

Initializing: vectors

- `c()`

```
v = c(1, 4, 3, 7, 8)
print(v)
```

```
[1] 1 4 3 7 8
```

-

```
w = 1:5
print(w)
```

```
[1] 1 2 3 4 5
```

Assignment

R = <-

```
w = 1:5
w <- 1:5
```

=

.

. $

-

· can . _

Help

. ?

help

. c

```
?c
help(topic = "c")
```

. ?? help.search

```
??c
help.search(pattern = "c")
```

Some Details

. y Y

.

. #

::

```
package::function()
```

```
utils::help("c")
```

Initializing: matrices and arrays

- m

-

```
m = matrix(1:12, nrow = 3)
print(m)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

- a

```
a = array(1:36, dim = c(3, 4, 3))
```

- dim()

```
dim(a)
```

```
[1] 3 4 3
```

Subsetting: vectors

. 1

```
print(v)
```

```
[1] 1 4 3 7 8
```

```
print(v[4])
```

```
[1] 7
```

```
print(v[1:3])
```

```
[1] 1 4 3
```

```
print(v[c(1,3,5)])
```

```
[1] 1 3 8
```

Subsetting: matrices

- `[row,column]`

```
print(m[1,3])
```

```
[1] 7
```

```
print(m[1:2,3:4])
```

```
     [,1] [,2]
[1,]    7   10
[2,]    8   11
```

- `row` `column`

```
print(m[,4])
```

```
[1] 10 11 12
```

```
print(m[2,])
```

```
[1]  2  5  8 11
```

Subsetting: arrays

. [x,y,z]

```
print(a[1,1,1])
```

```
[1] 1
```

```
dim(a[,4,])
```

```
[1] 3 3
```

```
a[,4]
```

Operators in R: return numeric

```
.          + - * / ^
.                    log abs sqrt
```

```
print(v); print(w)
```

```
[1] 1 4 3 7 8
```

```
[1] 1 2 3 4 5
```

```
print(v + 4)
```

```
[1]  5  8  7 11 12
```

```
print(v + w)
```

```
[1]  2  6  6 11 13
```

```
print(sqrt(w^2))
```

```
[1] 1 2 3 4 5
```

Operators in R: return logical

- `>` `>=` `<` `<=` `==` `!=`
- `!` `&` `|`
- `all()` `TRUE` `any()`

```
print(!FALSE)
```

```
[1] TRUE
```

```
print(TRUE | FALSE)
```

```
[1] TRUE
```

```
print(FALSE & FALSE)
```

```
[1] FALSE
```

```
c(all(c(TRUE, FALSE)), any(c(TRUE, FALSE)))
```

```
[1] FALSE  TRUE
```

Subsetting with logicals

`which` `TRUE`

```
which(v > 5)
```

```
[1] 4 5
```

```
v[ which(v > 5) ]
```

```
[1] 7 8
```

```
v[ v > 5 ]
```

```
[1] 7 8
```

Imaging Packages in R

Some packages we will use

- oro.nifti
  - nifti
  -
- neurobase oro.nifti

```
library(oro.nifti)
library(neurobase)
```

Reading in NIfTI images: assignment

readnii neurobase nifti R

t1

```
t1 = readnii("training01_01_mprage.nii.gz")
```

t1

```
class(t1)
```

```
[1] "nifti"
attr(,"package")
[1] "oro.nifti"
```

nifti images

```
print(t1)
```

```
t1

NIfTI-1 format
  Type             : nifti
  Data Type        : 4 (INT16)
  Bits per Pixel   : 16
  Slice Code       : 0 (Unknown)
  Intent Code      : 0 (None)
  Qform Code       : 1 (Scanner_Anat)
  Sform Code       : 1 (Scanner_Anat)
  Dimension        : 256 x 256 x 120
  Pixel Dimension : 0.83 x 0.83 x 1.17
  Voxel Units      : mm
  Time Units       : sec
```

Operations with `nifti` objects

```
                                    img + 2
img1 + img2

.              > >= < <= ==          !=

.         !      &       |

.           + - * / ^

.                    log abs sqrt

t1 + t1 + 2 # still a nifti

NIfTI-1 format
  Type             : nifti
  Data Type        : 4 (INT16)
  Bits per Pixel   : 16
  Slice Code       : 0 (Unknown)
  Intent Code      : 0 (None)
  Qform Code       : 1 (Scanner_Anat)
  Sform Code       : 1 (Scanner_Anat)
  Dimension        : 256 x 256 x 120
  Pixel Dimension : 0.83 x 0.83 x 1.17
  Voxel Units      : mm
  Time Units       : sec
```

Working with `nifti` objects

```
class(t1 > 400) # still a nifti
```

```
[1] "nifti"
attr(,"package")
[1] "oro.nifti"
```

```
head(t1 > 400) # values are now logical vs. numeric
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE
```

t1 > 400

Subsetting with `nifti` objects: like `arrays`

`t1`

```
t1[5, 4, 3]
```

```
[1] 0
```

```
t1[5, 4, ] # returns a vector of numbers (1-d)
t1[, 4, ] # returns a 2-d matrix
t1[1, , ] # returns a 2-d matrix
```

.

. `t1` `head`

```
head(t1[ t1 > 400 ]) # produces a vector of numbers
```

```
[1] 421 617 617 479 456 404
```

`which` with `nifti` objects

`which` `arr.ind = TRUE`

```
head(which(t1 > 400, arr.ind = TRUE))
```

```
     dim1 dim2 dim3
[1,]   98  129    1
[2,]   99  129    1
[3,]  100  129    1
[4,]  163  129    1
[5,]  164  129    1
[6,]  190  129    1
```

```
head(which(t1 > 400, arr.ind = FALSE))
```

```
[1] 32866 32867 32868 32931 32932 32958
```

Working with `nifti` objects: reassignment

```
t1_copy = t1
t1_copy[ t1_copy > 400 ] = 400 # replaced these values!
max(t1_copy) # should be 400
```

```
[1] 400
```

```
max(t1)
```

```
[1] 1505
```

```
           t1_copy                t1
t1_copy         t1
```

Writing Images out

```
t1_copy
```

```
writenii(nim = t1_copy,
         filename = "training01_mprage_under400.nii.gz")
file.exists("training01_mprage_under400.nii.gz")
```

```
[1] TRUE
```

```
file.exists      TRUE
```

```
.                all all(file.exists(VECTOR_OF_FILES))
```

Vectorizing a `nifti`

nifti vector c()

```
vals = c(t1)
class(vals)
```

```
[1] "numeric"
```

array(c(t1), dim = dim(t1))

t1

data.frame

```
df = data.frame(t1 = c(t1), mask = c(t1 > 400)); head(df)
```

```
  t1  mask
1  0 FALSE
2  0 FALSE
3  0 FALSE
4  0 FALSE
5  0 FALSE
6  0 FALSE
```

File helpers - for constructing filenames

`paste` `paste0`

`file.path(directory, filename)` `directory` `filename`

`/`

```
c(paste("img", ".nii.gz"), paste0("img", ".nii.gz"))
```

```
[1] "img .nii.gz" "img.nii.gz"
```

```
x = file.path("output_directory", paste0("img", ".nii.gz")); print(x)
```

```
[1] "output_directory/img.nii.gz"
```

`nii.stub` `bn = TRUE`

```
c(nii.stub(x), nii.stub(x, bn = TRUE))
```

```
[1] "output_directory/img" "img"
```

Main Packages we will use

- `oro.nifti`
- `neurobase` `oro.nifti`
- `fslr`
  -
- `ANTsR`
  -
- `extrantsr` `ANTsR` `oro.nifti`

- `ms.lesion`
  -

Conclusions

-
- nifti
  -
  -
- readnii writenii nifti
-
  -

Website