Package: osirad (via r-universe)

November 8, 2026

Title PET Voxels and Radiomics from 'OsiriX' Regions of Interest

Version 0.1.0

Description Reads regions of interest (ROIs) drawn in the 'OsiriX' medical image viewer and exported as XML files, and recovers the exact voxels of each ROI on the positron emission tomography (PET) series stored as Digital Imaging and Communications in Medicine (DICOM) files. The voxels are found by matching the voxel values stored in the XML file to the standardised uptake values (SUV) computed from the DICOM header. The resulting voxel table (positions in pixels and millimetres, and SUV) can be passed to the 'Python' library 'PyRadiomics' (van Griethuysen et al. (2017) <doi:10.1158/0008-5472.CAN-17-0339>) through 'reticulate' to compute radiomics features.

License GPL (>= 3)

Encoding UTF-8

Language en-GB

Depends R (>= 4.1.0)

Imports grDevices, graphics, oro.dicom, stats, xml2

Suggests knitr, reticulate, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

SystemRequirements Python (>= 3.9) with the modules 'pyradiomics', 'SimpleITK' and 'numpy' (optional, for pyradiomics_features())

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

URL https://github.com/tao-lan/osirad

BugReports https://github.com/tao-lan/osirad/issues

Config/pak/sysreqs libxml2-dev python3

Repository https://tao-lan.r-universe.dev

Date/Publication 2026-10-23 14:36:00 UTC

RemoteUrl https://github.com/tao-lan/osirad

RemoteRef HEAD

RemoteSha d216fe708a19cb573e02bd55d0153faac21fbcdc

Contents

extract_roi_voxels 2
match_roi_voxels 3
plot_roi_slices 4
pyradiomics_features 5
read_osirix_rois 6
read_pet_slice 7
suv_factor_from_header 8

Index **9**

`extract_roi_voxels` *Extract the voxels of 'OsiriX' ROIs from a PET DICOM series*

Description

Finds the position and the SUV of every voxel inside the regions of interest (ROIs) of an 'OsiriX' ROI XML file. For each ROI slice, the PET DICOM file is read and converted to SUV, and the pixels whose SUV equals the voxel values stored in the XML file are found with `match_roi_voxels()`.

Usage

```
extract_roi_voxels(xml_file, pet_dir, roi_name = NULL)
```

Arguments

`xml_file`	Path to the ROI XML file exported from 'OsiriX'.
`pet_dir`	Folder with the DICOM files of the PET series the ROIs were drawn on (one file per slice, no other files). The files are sorted by name; file number `ImageIndex + 1` is the ROI slice.
`roi_name`	Optional character vector of ROI names to keep. `NULL` (default) keeps all ROIs.

Value

An object of class `osirad_roi`, a list with:

voxels Data frame, one row per ROI voxel: `roi_id`, `roi_name`, `image_index`, pixel position (`x_px` = column, `y_px` = row, both starting at 0), position in mm (`x_mm`, `y_mm`, `z_mm`, patient coordinates) and `suv`.

match Data frame, one row per ROI: the counts of `match_roi_voxels()` and the number of voxels found.

suv_factor SUV factor from `suv_factor_from_header()`.

slices List of SUV slices from `read_pet_slice()`, one per ROI slice, named by `image_index`.

Examples

```
xml_file = system.file("extdata", "phantom", "ROI.xml", package = "osirad")
pet_dir = system.file("extdata", "phantom", "pet", package = "osirad")
x = extract_roi_voxels(xml_file, pet_dir)
x
head(x$voxels)
```

---

match_roi_voxels *Find the voxels of one ROI on one PET slice*

Description

Finds the pixels whose SUV equals the voxel values that 'OsiriX' stored for one region of interest (ROI). The search area is the range of the ROI corner points plus 2 pixels on each side.

Usage

```
match_roi_voxels(point_values, suv, points_px, tol = 1e-05)
```

Arguments

`point_values`	Voxel values of the ROI (`point_values` from `read_osirix_rois()`).
`suv`	SUV matrix of the slice (`suv` from `read_pet_slice()`).
`points_px`	ROI corner points in pixels (`points_px` from `read_osirix_rois()`).
`tol`	Relative tolerance for two values to be equal. The default allows for 'OsiriX' storing the values as 32-bit floating point numbers.

Details

For each distinct value that 'OsiriX' lists m times and that is found at c pixels of the search area:

- $c = m$: clear match.
- $c < m$: 'OsiriX' listed the same pixel more than once; the pixels found are kept.
- $c > m$: tie (the same SUV at several pixels). The m pixels with the most clear matches of this ROI among their 8 neighbours are kept (an ROI is one connected region); if this is still equal, the pixels nearest the centre of the corner points are kept.
- $c = 0$: not found.

Each ROI is searched with its own values, so several ROIs on one slice are handled one by one.

Value

A list with:

rc Two-column matrix (`row`, `col`) of the ROI pixels in `suv`.

counts Named vector: number of XML values (`xml_values`) and how many of them were a clear match (`clear`), listed twice (`listed_twice`), a tie decided by neighbours (`tie_by_neighbours`), a tie decided by distance (`tie_by_distance`) or not found (`not_found`).

Examples

```
xml_file = system.file("extdata", "phantom", "ROI.xml", package = "osirad")
pet_dir = system.file("extdata", "phantom", "pet", package = "osirad")
rois = read_osirix_rois(xml_file)
r = rois[[1]]
f = list.files(pet_dir, full.names = TRUE)
hdr = oro.dicom::readDICOMFile(f[1], pixelData = FALSE)$hdr
sl = read_pet_slice(f[r$k], suv_factor_from_header(hdr))
m = match_roi_voxels(r$point_values, sl$suv, r$points_px)
m$counts
```

---

plot_roi_slices *Plot PET slices with the ROI voxels*

Description

Draws each ROI slice in grey scale (SUV) with the ROI voxels as red squares, four slices per page. Row 1 of the image is at the top, as in 'OsiriX'.

Usage

```
plot_roi_slices(x, image_index = NULL, ncol = 2, nrow = 2)
```

Arguments

`x`	Result of `extract_roi_voxels()`.
`image_index`	Slices to draw (`ImageIndex`). `NULL` (default) draws all ROI slices.
`ncol`, `nrow`	Number of panels per page.

Value

`x`, invisibly. Called for the plot.

Examples

```
xml_file = system.file("extdata", "phantom", "ROI.xml", package = "osirad")
pet_dir = system.file("extdata", "phantom", "pet", package = "osirad")
x = extract_roi_voxels(xml_file, pet_dir)
plot_roi_slices(x, image_index = 4)
```

`pyradiomics_features` *Radiomics features of the ROI voxels with 'PyRadiomics'*

Description

Computes radiomics features of the ROI voxels with the 'Python' library 'PyRadiomics' (van Griethuysen et al. 2017), called through 'reticulate'. The PET series is read in 'Python' and converted to SUV with the same SUV factor as in R; the ROI voxels form one mask (label 1). Only the original image is used (no filters).

Usage

```
pyradiomics_features(
  x,
  pet_dir,
  settings = list(binWidth = 0.25, normalize = FALSE),
 feature_classes = c("shape", "firstorder", "glcm", "glrlm", "glszm", "gldm", "ngtdm"),
  python = NULL
)
```

Arguments

`x`	Result of `extract_roi_voxels()`.
`pet_dir`	Folder with the DICOM files of the PET series (the same as in `extract_roi_voxels()`).
`settings`	Named list of 'PyRadiomics' settings, see https://pyradiomics.readthedocs.io/en/latest/customization.html. The default is a fixed bin width of 0.25 SUV and no intensity normalisation; no resampling.
`feature_classes`	'PyRadiomics' feature classes to compute; all features of each class are computed.
`python`	Optional path to the 'Python' executable that has 'PyRadiomics' installed. It must be set before 'Python' is first used in the R session.

Value

A list with:

features Data frame, one row per feature: `feature` (name) and `value`.

diagnostics Data frame, one row per 'PyRadiomics' diagnostics item (versions, settings, image and mask information): `item` and `value` (as text).

check Named vector: ROI voxels in `x`, voxels in the 'Python' mask, largest distance (mm) between a voxel and its slice, and largest difference between the SUV in 'Python' and in R.

References

van Griethuysen JJM, Fedorov A, Parmar C, et al. (2017). Computational radiomics system to decode the radiographic phenotype. Cancer Research, 77(21), e104-e107. doi:10.1158/0008-5472.CAN170339

Examples

```
# Needs 'Python' with 'PyRadiomics'; the path is read from the
# environment variable OSIRAD_PYTHON
xml_file = system.file("extdata", "phantom", "ROI.xml", package = "osirad")
pet_dir = system.file("extdata", "phantom", "pet", package = "osirad")
x = extract_roi_voxels(xml_file, pet_dir, roi_name = "Tumour")
rad = pyradiomics_features(x, pet_dir, python = Sys.getenv("OSIRAD_PYTHON"))
head(rad$features)
rad$check
```

---

read_osirix_rois *Read regions of interest exported from 'OsiriX'*

Description

Reads the XML file written by the 'OsiriX' menu "Plugins > ROI Tools > Export ROIs" (XML format). The file stores, for each region of interest (ROI) on each slice, the polygon corner points, the value of every voxel inside the ROI and summary statistics. It does not store the voxel positions; `extract_roi_voxels()` finds them.

Usage

```
read_osirix_rois(xml_file, roi_name = NULL)
```

Arguments

`xml_file`	Path to the ROI XML file.
`roi_name`	Optional character vector. Only the ROIs with these names are kept. `NULL` (default) keeps all ROIs.

Value

A list with one element per ROI per slice. Each element is a list with:

image_index Slice number in 'OsiriX' (`ImageIndex`, starts at 0).

k `image_index + 1`, the file number in the sorted DICOM series.

image_width, image_height, image_total_num Slice size in pixels and number of slices in the series.

roi_name ROI name.

series_uid `SeriesInstanceUID` of the image series the ROI was drawn on.

points_px, points_mm Polygon corner points, in pixels (columns `x`, `y`) and in mm (columns `x`, `y`, `z`).

point_values Value of every voxel inside the ROI (`Point_value`).

n_voxels_osirix Length of `point_values`.

area_cm2, area_pix2, roi_mean, roi_min, roi_max, roi_total ROI statistics computed by 'OsiriX'.

Examples

```
xml_file = system.file("extdata", "phantom", "ROI.xml", package = "osirad")
rois = read_osirix_rois(xml_file)
length(rois)
str(rois[[1]])
```

`read_pet_slice` *Read one PET slice in SUV*

Description

Reads one DICOM file of a PET series and converts the pixel values to SUV with the slice's own `RescaleSlope` and `RescaleIntercept` and the SUV factor.

Usage

```
read_pet_slice(file, suv_factor)
```

Arguments

`file`	Path to the DICOM file.
`suv_factor`	SUV factor, from `suv_factor_from_header()`.

Value

A list with:

suv Matrix of SUV, `[row, column]` = `[y_px + 1, x_px + 1]`, in the DICOM orientation (row 1 at the top of the image).

ipp `ImagePositionPatient`: position (mm) of the first pixel.

iop `ImageOrientationPatient`: direction cosines of the rows and columns.

spacing `PixelSpacing` (mm): row spacing, column spacing.

Examples

```
pet_dir = system.file("extdata", "phantom", "pet", package = "osirad")
f = list.files(pet_dir, full.names = TRUE)
hdr = oro.dicom::readDICOMFile(f[1], pixelData = FALSE)$hdr
sl = read_pet_slice(f[4], suv_factor_from_header(hdr))
range(sl$suv)
```

`suv_factor_from_header` *SUV factor from a PET DICOM header*

Description

Computes the factor that converts PET activity concentration (Bq/ml) to body-weight standardised uptake value (SUV):

$$SUV = activity \times weight / dose_{decayed},$$

with the patient weight in g and the injected dose (Bq) decayed from the injection time (`RadiopharmaceuticalStartTime`) to the series start (`SeriesTime`).

Usage

```
suv_factor_from_header(hdr)
```

Arguments

`hdr`	DICOM header table of one slice of the PET series, as returned in the `hdr` element of `oro.dicom::readDICOMFile()`.

Details

The image values must be in Bq/ml (`Units` = `BQML`) and decay corrected to the series start (`DecayCorrection` = `START`); otherwise the function stops.

Value

The SUV factor, a single number (g/Bq). Multiply the rescaled pixel values (Bq/ml) by it to get SUV.

Examples

```
pet_dir = system.file("extdata", "phantom", "pet", package = "osirad")
f = list.files(pet_dir, full.names = TRUE)[1]
hdr = oro.dicom::readDICOMFile(f, pixelData = FALSE)$hdr
suv_factor_from_header(hdr)
```

Index

extract_roi_voxels, 2
extract_roi_voxels(), *4–6*

match_roi_voxels, 3
match_roi_voxels(), *2*, *3*

oro.dicom::readDICOMFile(), *8*

plot_roi_slices, 4
pyradiomics_features, 5

read_osirix_rois, 6
read_osirix_rois(), *3*
read_pet_slice, 7
read_pet_slice(), *3*

suv_factor_from_header, 8
suv_factor_from_header(), *3*, *7*