Package: RIA (via r-universe)

September 5, 2024

Type Package

Title Radiomics Image Analysis Toolbox for Medial Images

Version 1.7.2

Date 2024-01-03

Maintainer Marton Kolossvary <`marton.kolossvary@gmail.com`>

Description Radiomics image analysis toolbox for 2D and 3D
radiological images. RIA supports DICOM, NIfTI, nrrd and npy
(numpy array) file formats. RIA calculates first-order, gray
level co-occurrence matrix, gray level run length matrix and
geometry-based statistics. Almost all calculations are done
using vectorized formulas to optimize run speeds. Calculation
of several thousands of parameters only takes minutes on a
single core of a conventional PC. Detailed methodology has been
published: Kolossvary et al. Circ: Cardiovascular Imaging.
2017;10(12):e006843 <doi:10.1161/CIRCIMAGING.117.006843>.

License AGPL-3

Depends R (>= 3.3.0)

Imports oro.dicom (>= 0.5.0), oro.nifti (>= 0.9.1)

LazyData TRUE

RoxygenNote 7.2.3

NeedsCompilation no

Suggests knitr, rmarkdown, nat (>= 1.8.11), reticulate(>= 1.20)

Encoding UTF-8

URL `https://pubmed.ncbi.nlm.nih.gov/29233836/`

Author Marton Kolossvary [aut, cre]

Repository CRAN

Date/Publication 2024-01-08 11:50:21 UTC

Contents

`DICOM_codes` *rda containing DICOM header codes to include in* RIA_image *object*

Description

rda data file containing Name, Group and Element codes of DICOM header info to be included into *RIA_image* object by default when using `load_dicom` function. Can be edited to change defaults.

Usage

```
DICOM_codes
```

Format

Each row is a DICOM header input

Value

3 column data.frame

References

Márton KOLOSSVÁRY et al. Radiomic Features Are Superior to Conventional Quantitative Computed Tomographic Metrics to Identify Coronary Plaques With Napkin-Ring Sign Circulation: Cardiovascular Imaging (2017). DOI: 10.1161/circimaging.117.006843 `https://pubmed.ncbi.nlm.nih.gov/29233836/`

Márton KOLOSSVÁRY et al. Cardiac Computed Tomography Radiomics: A Comprehensive Review on Radiomic Techniques. Journal of Thoracic Imaging (2018). DOI: 10.1097/RTI.0000000000000268 `https://pubmed.ncbi.nlm.nih.gov/28346329/`

`load_dicom` *Loads DICOM images to RIA image format*

Description

Loads DICOM images to a *RIA_image* object. *RIA_image* is a list with three mandatory attributes.

- **RIA_data** is a *RIA_data* object, which has two potential slots. *$orig* contains the original image after loading and is a 3D array of integers created with `create3D`. *$modif* contains the image that has been modified using functions.
- **RIA_header** is a *RIA_header* object, which is list of DICOM header information.
- **RIA_log** is a *RIA_log* object, which is a list updated by RIA functions and acts as a log and possible input for some functions.

Further attributes may also be added by RIA functions.

Usage

```
load_dicom(
  filename,
  mask_filename = NULL,
  keep_mask_values = 1,
  switch_z = FALSE,
  crop_in = TRUE,
  replace_in = TRUE,
  center_in = TRUE,
  zero_value = NULL,
  min_to = -1024,
  header_add = NULL,
  header_exclude = NULL,
  verbose_in = TRUE,
  recursive_in = TRUE,
  exclude_in = "sql",
  mode_in = "integer",
  transpose_in = TRUE,
  pixelData_in = TRUE,
  mosaic_in = FALSE,
  mosaicXY_in = NULL,
  sequence_in = FALSE,
  ...
)
```

Arguments

`filename`	string, file path to directory containing *dcm* files.

`mask_filename`	string vector, file path to optional directory containing *dcm* files of mask image. If multiple are supplied, then those voxels are kept which have one of the values of *keep_mask_values* in any of the supplied masks.
`keep_mask_values`	integer vector or string, indicates which value or values of the mask image to use as indicator to identify voxels wished to be processed. Usually 1-s indicate voxels wished to be processed. However, one mask image might contain several segmentations, in which case supplying several integers is allowed. Furthermore, if the same string is supplied to *filename* and *mask_filename*, then the integers in *keep_mask_values* are used to specify which voxel values to analyze. This way the provided image can be segmented to specific components. For example, if you wish to analyze only the low-density non-calcified component of coronary plaques, then *keep_mask_values* can specify this by setting it to: -100:30. If a single string is provided, then each element of the mask will be examined against the statement in the string. For example, if *'>0.5'* is provided i.e. the mask is probabilities after a DL algorithm, then all voxels with values >0.5 in the mask image will be kept. This can be a complex logical expression. The data on which the expression is executed is called *data* or *data_mask*, depending on whether you wish to filter the original image, that is the original image is supplied as a mask, or if you have unique mask files respectively. Therefore for complex logical expressions you can define for example: *'>-100 & data<30'* to consider data values between -100 and 30, or *'>0.5 & data_mask<0.75'* to select voxels based-on mask values between 0.5 and 0.75 for example if they represent a probability mask.
`switch_z`	logical, indicating whether to change the orientation of the images in the Z axis. Some software reverse the order of the manipulated image in the Z axis, and therefore the images of the mask image need to be reversed.
`crop_in`	logical, indicating whether to crop *RIA_image* to smallest bounding box.
`replace_in`	logical, whether to replace smallest values indicated by *zero_value*, which are considered to indicate no signal, to NA.
`center_in`	logical, whether to shift data so smallest value is equal to *min_to* input parameter.
`zero_value`	integer, indicating voxels values which are considered not to have any information. If left empty, then the smallest HU value in the image will be used, if *replace_in* is TRUE.
`min_to`	integer, value to which data is shifted to if *center_in* is TRUE.
`header_add`	dataframe, with three columns: Name, Group and Element containing the name, the group and the element code of the DICOM fields wished to be added to the*RIA_header*.
`header_exclude`	dataframe, with three columns: Name, Group and Element containing the name, the group and the element code of the DICOM fields wished to be excluded from the default header elements present in *DICOM_codes* rda file.
`verbose_in`	logical, indicating whether to print detailed information. Most prints can also be suppresed using the `suppressMessages` function.
`recursive_in`	*recursive* parameter input of `readDICOM`.

`exclude_in`	*exclude* parameter input of `readDICOM`.
`mode_in`	*mode* parameter input of `create3D`.
`transpose_in`	*transpose* parameter input of `create3D`.
`pixelData_in`	*pixelData* parameter input of `create3D`.
`mosaic_in`	*mosaic* parameter input of `create3D`.
`mosaicXY_in`	*mosaicXY* parameter input of `create3D`.
`sequence_in`	*sequence* parameter input of `create3D`.
`...`	additional arguments to `readDICOM`, `readDICOMFile` and `create3D`.

Details

load_dicom is used to transform DICOM datasets into the RIA environment. *RIA_image* object was developed to facilitate and simplify radiomics calculations by keeping all necessary information in one place.

RIA_data stores the DICOM image that is converted to numerical 3D arrays using `readDICOM` and `create3D`. The function stores the original loaded image in *RIA_data$orig*, while all modified images are stored in *RIA_data$modif*. By default, the original image *RIA_data$orig* is untouched by functions other than those operating in *load_dicom*. While other functions operate on the *RIA_data$modif* image by default.
Due to memory concerns, there can only be one *RIA_data$orig* and *RIA_data$modif* image present at one time in a *RIA_image*. Therefore, if image manipulations are performed, then the *RIA_data$modif* will be overwritten. However, functions can save images into new slots of *RIA_image*, for example discretized images can be saved to the *discretized* slot of *RIA_image*.
load_dicom not only loads the DICOM image based on parameters that can be set for `readDICOM` and `create3D`, but also can perform minimal manipulations on the image itself.
crop_in logical variable is used to indicate, whether to crop the image to the smallest bounding box still containing all the information. If TRUE, then all X, Y and potentially Z slices containing no information will be removed. This allows significant reduction of necessary memory to store image data.
zero_value parameter is used to indicate HU values which contain no information. If left empty, then the smallest value will be considered as indicating voxels without a signal.
replace_in logical can be used to change values that are considered to have no signal to NA. This is necessary to receive proper statistical values later on.
center_in logical is used to indicate whether the values should be shifted. Some vendors save HU values as positive integers to spare memory and minimalize file sizes. Therefore, in some instances shift of the scale is needed. By default, the values are shifted by -1024, but in other cases a different constant might be required, which can be set using the *min_to* input.

RIA_header is a list containing the most basic patient and examination information needed for further analysis. The default DICOM set is present in *DICOM_codes*, which can be edited to anyones needs. But if we wish only to add of remove specific DICOM header rows, then the *header_add* and *header_exclude* can be used.

RIA_log is a list of variables, which give an overview of what has been done with the image. If the whole *RIA_image* is supplied to a function, the information regarding the manipulations are written into the *$events* array in chronological order. Furthermore, some additional information is also saved in the log, which might be needed for further analysis.

Value

Returns a *RIA_image* object. *RIA_image* is a list with three mandatory attributes.

- **RIA_data** is a *RIA_data* object containing the image in *$orig* slot.
- **RIA_header** is a *RIA_header* object, which is s list of DICOM information.
- **RIA_log** is a *RIA_log* object, which is a list updated by RIA functions and acts as a log and possible input for some functions.

References

Márton KOLOSSVÁRY et al. Radiomic Features Are Superior to Conventional Quantitative Computed Tomographic Metrics to Identify Coronary Plaques With Napkin-Ring Sign Circulation: Cardiovascular Imaging (2017). DOI: 10.1161/circimaging.117.006843 https://pubmed.ncbi.nlm.nih.gov/29233836/

Márton KOLOSSVÁRY et al. Cardiac Computed Tomography Radiomics: A Comprehensive Review on Radiomic Techniques. Journal of Thoracic Imaging (2018). DOI: 10.1097/RTI.0000000000000268 https://pubmed.ncbi.nlm.nih.gov/28346329/

Examples

```
## Not run:
 #Image will be croped to smallest bounding box, and smallest values will be changed to NA,
 while 1024 will be substracted from all other data points.
 RIA_image <- load_dicom("/Users/Test/Documents/Radiomics/John_Smith/DICOM_folder/")

## End(Not run)
```

`load_nifti` *Loads NIfTI images to RIA image format*

Description

Loads NIfTI images to a *RIA_image* object. *RIA_image* is a list with three mandatory attributes.

- **RIA_data** is a *RIA_data* object, which has two potential slots. *$orig* contains the original image after loading *$modif* contains the image that has been modified using functions.
- **RIA_header** is a *RIA_header* object, which is list of header information.
- **RIA_log** is a *RIA_log* object, which is a list updated by RIA functions and acts as a log and possible input for some functions.

Further attributes may also be added by RIA functions.

Usage

```
load_nifti(
  filename,
  image_dim = 3,
  mask_filename = NULL,
  keep_mask_values = 1,
  switch_z = FALSE,
  crop_in = TRUE,
  replace_in = TRUE,
  center_in = FALSE,
  zero_value = NULL,
  min_to = -1024,
  verbose_in = TRUE,
  reorient_in = TRUE,
  ...
)
```

Arguments

`filename`	string, file path to directory containing *NIfTI* file.
`image_dim`	integer, dimensions of the image.
`mask_filename`	string vector, file path to optional directory containing *NIfTI* file of mask image. If multiple are supplied, then those voxels are kept which have one of the values of *keep_mask_values* in any of the supplied masks.
`keep_mask_values`	integer vector or string, indicates which value or values of the mask image to use as indicator to identify voxels wished to be processed. Usually 1-s indicate voxels wished to be processed. However, one mask image might contain several segmentations, in which case supplying several integers is allowed. Furthermore, if the same string is supplied to *filename* and *mask_filename*, then the integers in *keep_mask_values* are used to specify which voxel values to analyze. This way the provided image can be segmented to specific components. For example, if you wish to analyze only the low-density non-calcified component of coronary plaques, then *keep_mask_values* can specify this by setting it to: -100:30. If a single string is provided, then each element of the mask will be examined against the statement in the string. For example, if *'>0.5'* is provided i.e. the mask is probabilities after a DL algorithm, then all voxels with values >0.5 in the mask image will be kept. This can be a complex logical expression. The data on which the expression is executed is called *data* or *data_mask*, depending on whether you wish to filter the original image, that is the original image is supplied as a mask, or if you have unique mask files respectively. Therefore for complex logical expressions you can define for example: *'>-100 & data<30'* to consider data values between -100 and 30, or *'>0.5 & data_mask<0.75'* to select voxels based-on mask values between 0.5 and 0.75 for example if they represent a probability mask.
`switch_z`	logical, indicating whether to change the orientation of the images in the Z axis. Some software reverse the order of the manipulated image in the Z axis, and therefore the images of the mask image need to be reversed.

`crop_in`	logical, indicating whether to crop *RIA_image* to smallest bounding box.
`replace_in`	logical, whether to replace smallest values indicated by *zero_value*, which are considered to indicate no signal, to NA.
`center_in`	logical, whether to shift data so smallest value is equal to *min_to* input parameter.
`zero_value`	integer, indicating voxels values which are considered not to have any information. If left empty, then the smallest HU value in the image will be used, if *replace_in* is TRUE.
`min_to`	integer, value to which data is shifted to if *center_in* is TRUE.
`verbose_in`	logical, indicating whether to print detailed information. Most prints can also be suppresed using the `suppressMessages` function.
`reorient_in`	*reorient* parameter input of `readNIfTI`.
`...`	additional arguments to `readNIfTI`, `nifti_header`.

Details

load_nifti is used to transform NIfTI datasets into the RIA environment. *RIA_image* object was developed to facilitate and simplify radiomics calculations by keeping all necessary information in one place.

RIA_data stores the image that is converted to numerical 3D arrays using `readNIfTI`. The function stores the original loaded image in *RIA_data$orig*, while all modified images are stored in *RIA_data$modif*. By default, the original image *RIA_data$orig* is untouched by functions other than those operating in *load_nifti*. While other functions operate on the *RIA_data$modif* image by default.
Due to memory concerns, there can only be one *RIA_data$orig* and *RIA_data$modif* image present at one time in a *RIA_image*. Therefore, if image manipulations are performed, then the *RIA_data$modif* will be overwritten. However, functions can save images into new slots of *RIA_image*, for example discretized images can be saved to the *discretized* slot of *RIA_image*.
load_nifti not only loads the image based on parameters that can be set for `readNIfTI`, but also can perform minimal manipulations on the image itself.
crop_in logical variable is used to indicate, whether to crop the image to the smallest bounding box still containing all the information. If TRUE, then all X, Y and potentially Z slices containing no information will be removed. This allows significant reduction of necessary memory to store image data.
zero_value parameter is used to indicate HU values which contain no information. If left empty, then the smallest value will be considered as indicating voxels without a signal.
replace_in logical can be used to change values that are considered to have no signal to NA. This is necessary to receive proper statistical values later on.
center_in logical is used to indicate whether the values should be shifted. Some vendors save HU values as positive integers to spare memory and minimalize file sizes. Therefore, in some instances shift of the scale is needed. By default, the values are shifted by -1024, but in other cases a different constant might be required, which can be set using the *min_to* input.

RIA_header is a list containing the most basic patient and examination information present in the NIfTI file.

RIA_log is a list of variables, which give an overview of what has been done with the image. If the whole *RIA_image* is supplied to a function, the information regarding the manipulations are written into the *$events* array in chronological order. Furthermore, some additional information is also saved in the log, which might be needed for further analysis.

Value

Returns a *RIA_image* object. *RIA_image* is a list with three mandatory attributes.

- **RIA_data** is a *RIA_data* object containing the image in *$orig* slot.
- **RIA_header** is a *RIA_header* object, which is s list of meta information.
- **RIA_log** is a *RIA_log* object, which is a list updated by RIA functions and acts as a log and possible input for some functions.

References

Márton KOLOSSVÁRY et al. Radiomic Features Are Superior to Conventional Quantitative Computed Tomographic Metrics to Identify Coronary Plaques With Napkin-Ring Sign Circulation: Cardiovascular Imaging (2017). DOI: 10.1161/circimaging.117.006843 https://pubmed.ncbi.nlm.nih.gov/29233836/

Márton KOLOSSVÁRY et al. Cardiac Computed Tomography Radiomics: A Comprehensive Review on Radiomic Techniques. Journal of Thoracic Imaging (2018). DOI: 10.1097/RTI.0000000000000268 https://pubmed.ncbi.nlm.nih.gov/28346329/

Examples

```
## Not run:
 #Image will be croped to smallest bounding box, and smallest values will be changed to NA,
 while 1024 will be substracted from all other data points.
 RIA_image <- load_nifti("/Users/Test/Documents/Radiomics/John_Smith/NIfTI_folder/sample.nii")

## End(Not run)
```

`load_npy` *Loads npy files to RIA image format*

Description

Loads numpy arrays from python to a *RIA_image* object using the *reticulate* package. Requires python and numpy to be installed! *RIA_image* is a list with three mandatory attributes.

- **RIA_data** is a *RIA_data* object, which has two potential slots. *$orig* contains the original image after loading *$modif* contains the image that has been modified using functions.
- **RIA_header** is a *RIA_header* object, which is list of header information.
- **RIA_log** is a *RIA_log* object, which is a list updated by RIA functions and acts as a log and possible input for some functions.

Further attributes may also be added by RIA functions.

Usage

```
load_npy(
  filename,
  mask_filename = NULL,
  keep_mask_values = 1,
  switch_z = FALSE,
  crop_in = TRUE,
  replace_in = TRUE,
  center_in = FALSE,
  zero_value = NULL,
  min_to = -1024,
  PixelSpacing = 1,
  SpacingBetweenSlices = 1,
  verbose_in = TRUE,
  ...
)
```

Arguments

`filename`	string, file path to *npy* file.
`mask_filename`	string vector, file path to *npy* file of mask image. If multiple are supplied, then those voxels are kept which have one of the values of *keep_mask_values* in any of the supplied masks.
`keep_mask_values`	integer vector or string, indicates which value or values of the mask image to use as indicator to identify voxels wished to be processed. Usually 1-s indicate voxels wished to be processed. However, one mask image might contain several segmentations, in which case supplying several integers is allowed. Furthermore, if the same string is supplied to *filename* and *mask_filename*, then the integers in *keep_mask_values* are used to specify which voxel values to analyze. This way the provided image can be segmented to specific components. For example, if you wish to analyze only the low-density non-calcified component of coronary plaques, then *keep_mask_values* can specify this by setting it to: -100:30. If a single string is provided, then each element of the mask will be examined against the statement in the string. For example, if *'>0.5'* is provided i.e. the mask is probabilities after a DL algorithm, then all voxels with values >0.5 in the mask image will be kept. This can be a complex logical expression. The data on which the expression is executed is called *data* or *data_mask*, depending on whether you wish to filter the original image, that is the original image is supplied as a mask, or if you have unique mask files respectively. Therefore for complex logical expressions you can define for example: *'>-100 & data<30'* to consider data values between -100 and 30, or *'>0.5 & data_mask<0.75'* to select voxels based-on mask values between 0.5 and 0.75 for example if they represent a probability mask.
`switch_z`	logical, indicating whether to change the orientation of the images in the Z axis. Some software reverse the order of the manipulated image in the Z axis, and therefore the images of the mask image need to be reversed.

`crop_in`	logical, indicating whether to crop *RIA_image* to smallest bounding box.
`replace_in`	logical, whether to replace smallest values indicated by *zero_value*, which are considered to indicate no signal, to NA.
`center_in`	logical, whether to shift data so smallest value is equal to *min_to* input parameter.
`zero_value`	integer, indicating voxels values which are considered not to have any information. If left empty, then the smallest HU value in the image will be used, if *replace_in* is TRUE.
`min_to`	integer, value to which data is shifted to if *center_in* is TRUE.
`PixelSpacing`	numerical, Pixel spacing value of image.
`SpacingBetweenSlices`	numerical, Spacing between the slices value of the image.
`verbose_in`	logical, indicating whether to print detailed information. Most prints can also be suppressed using the `suppressMessages` function.
`...`	additional arguments to *numpy.load*.

Details

load_npy is used to transform numpy array datasets into the RIA environment. *RIA_image* object was developed to facilitate and simplify radiomics calculations by keeping all necessary information in one place.

RIA_data stores the numpy image that is converted to numerical 3D arrays using the reticulate package. The function stores the original loaded image in *RIA_data$orig*, while all modified images are stored in *RIA_data$modif*. By default, the original image *RIA_data$orig* is untouched by functions other than those operating in *load_npy*. While other functions operate on the *RIA_data$modif* image by default.
Due to memory concerns, there can only be one *RIA_data$orig* and *RIA_data$modif* image present at one time in a *RIA_image*. Therefore, if image manipulations are performed, then the *RIA_data$modif* will be overwritten. However, functions can save images into new slots of *RIA_image*, for example discretized images can be saved to the *discretized* slot of *RIA_image*.
load_npy not only loads the image, but also can perform minimal manipulations on the image itself.
crop_in logical variable is used to indicate, whether to crop the image to the smallest bounding box still containing all the information. If TRUE, then all X, Y and potentially Z slices containing no information will be removed. This allows significant reduction of necessary memory to store image data.
zero_value parameter is used to indicate HU values which contain no information. If left empty, then the smallest value will be considered as indicating voxels without a signal.
replace_in logical can be used to change values that are considered to have no signal to NA. This is necessary to receive proper statistical values later on.
center_in logical is used to indicate whether the values should be shifted. Some vendors save HU values as positive integers to spare memory and minimalize file sizes. Therefore, in some instances shift of the scale is needed. By default, the values are shifted by -1024, but in other cases a different constant might be required, which can be set using the *min_to* input.

RIA_header is a list containing the most basic patient and examination information present in the npy file. Data is limited to the pixel spacing and spacing between the slices information.

RIA_log is a list of variables, which give an overview of what has been done with the image. If the whole *RIA_image* is supplied to a function, the information regarding the manipulations are written into the *$events* array in chronological order. Furthermore, some additional information is also saved in the log, which might be needed for further analysis.

Value

Returns a *RIA_image* object. *RIA_image* is a list with three mandatory attributes.

- **RIA_data** is a *RIA_data* object containing the image in *$orig* slot.
- **RIA_header** is a *RIA_header* object, which is s list of header information.
- **RIA_log** is a *RIA_log* object, which is a list updated by RIA functions and acts as a log and possible input for some functions.

References

Márton KOLOSSVÁRY et al. Radiomic Features Are Superior to Conventional Quantitative Computed Tomographic Metrics to Identify Coronary Plaques With Napkin-Ring Sign Circulation: Cardiovascular Imaging (2017). DOI: 10.1161/circimaging.117.006843 https://pubmed.ncbi.nlm.nih.gov/29233836/

Márton KOLOSSVÁRY et al. Cardiac Computed Tomography Radiomics: A Comprehensive Review on Radiomic Techniques. Journal of Thoracic Imaging (2018). DOI: 10.1097/RTI.0000000000000268 https://pubmed.ncbi.nlm.nih.gov/28346329/

Examples

```
## Not run:
 #Image will be croped to smallest bounding box, and smallest values will be changed to NA
 RIA_image <- load_npy("/Users/Test/Documents/Radiomics/John_Smith/npy_folder/sample.npy")

## End(Not run)
```

`load_nrrd` *Loads nrrd images to RIA image format*

Description

Loads nrrd images to a *RIA_image* object. *RIA_image* is a list with three mandatory attributes.

- **RIA_data** is a *RIA_data* object, which has two potential slots. *$orig* contains the original image after loading *$modif* contains the image that has been modified using functions.
- **RIA_header** is a *RIA_header* object, which is list of header information.
- **RIA_log** is a *RIA_log* object, which is a list updated by RIA functions and acts as a log and possible input for some functions.

Further attributes may also be added by RIA functions.

Usage

```
load_nrrd(
  filename,
  mask_filename = NULL,
  keep_mask_values = 1,
  switch_z = FALSE,
  crop_in = TRUE,
  replace_in = TRUE,
  center_in = FALSE,
  zero_value = NULL,
  min_to = -1024,
  verbose_in = TRUE,
  origin_in = NULL,
  ReadByteAsRaw_in = "unsigned",
  ...
)
```

Arguments

`filename`	string, file path to directory containing *nrrd* file.
`mask_filename`	string vector, file path to optional directory containing *nrrd* file of mask image. If multiple are supplied, then those voxels are kept which have one of the values of *keep_mask_values* in any of the supplied masks.
`keep_mask_values`	integer vector or string, indicates which value or values of the mask image to use as indicator to identify voxels wished to be processed. Usually 1-s indicate voxels wished to be processed. However, one mask image might contain several segmentations, in which case supplying several integers is allowed. Furthermore, if the same string is supplied to *filename* and *mask_filename*, then the integers in *keep_mask_values* are used to specify which voxel values to analyze. This way the provided image can be segmented to specific components. For example, if you wish to analyze only the low-density non-calcified component of coronary plaques, then *keep_mask_values* can specify this by setting it to: -100:30. If a single string is provided, then each element of the mask will be examined against the statement in the string. For example, if *’>0.5’* is provided i.e. the mask is probabilities after a DL algorithm, then all voxels with values >0.5 in the mask image will be kept. This can be a complex logical expression. The data on which the expression is executed is called *data* or *data_mask*, depending on whether you wish to filter the original image, that is the original image is supplied as a mask, or if you have unique mask files respectively. Therefore for complex logical expressions you can define for example: *’>-100 & data<30’* to consider data values between -100 and 30, or *’>0.5 & data_mask<0.75’* to select voxels based-on mask values between 0.5 and 0.75 for example if they represent a probability mask.
`switch_z`	logical, indicating whether to change the orientation of the images in the Z axis. Some software reverse the order of the manipulated image in the Z axis, and therefore the images of the mask image need to be reversed.

`crop_in`	logical, indicating whether to crop *RIA_image* to smallest bounding box.
`replace_in`	logical, whether to replace smallest values indicated by *zero_value*, which are considered to indicate no signal, to NA.
`center_in`	logical, whether to shift data so smallest value is equal to *min_to* input parameter.
`zero_value`	integer, indicating voxels values which are considered not to have any information. If left empty, then the smallest HU value in the image will be used, if *replace_in* is TRUE.
`min_to`	integer, value to which data is shifted to if *center_in* is TRUE.
`verbose_in`	logical, indicating whether to print detailed information. Most prints can also be suppresed using the `suppressMessages` function.
`origin_in`	*origin* parameter input of `read.nrrd`.
`ReadByteAsRaw_in`	*origin* parameter input of `read.nrrd`.
`...`	additional arguments to `read.nrrd`, `read.nrrd.header`.

Details

load_nrrd is used to transform nrrd datasets into the RIA environment. *RIA_image* object was developed to facilitate and simplify radiomics calculations by keeping all necessary information in one place.

RIA_data stores the nrrd image that is converted to numerical 3D arrays using `read.nrrd`. The function stores the original loaded image in *RIA_data$orig*, while all modified images are stored in *RIA_data$modif*. By default, the original image *RIA_data$orig* is untouched by functions other than those operating in *load_nrrd*. While other functions operate on the *RIA_data$modif* image by default.
Due to memory concerns, there can only be one *RIA_data$orig* and *RIA_data$modif* image present at one time in a *RIA_image*. Therefore, if image manipulations are performed, then the *RIA_data$modif* will be overwritten. However, functions can save images into new slots of *RIA_image*, for example discretized images can be saved to the *discretized* slot of *RIA_image*.
load_nrrd not only loads the image based on parameters that can be set for `read.nrrd`, but also can perform minimal manipulations on the image itself.
crop_in logical variable is used to indicate, whether to crop the image to the smallest bounding box still containing all the information. If TRUE, then all X, Y and potentially Z slices containing no information will be removed. This allows significant reduction of necessary memory to store image data.
zero_value parameter is used to indicate HU values which contain no information. If left empty, then the smallest value will be considered as indicating voxels without a signal.
replace_in logical can be used to change values that are considered to have no signal to NA. This is necessary to receive proper statistical values later on.
center_in logical is used to indicate whether the values should be shifted. Some vendors save HU values as positive integers to spare memory and minimalize file sizes. Therefore, in some instances shift of the scale is needed. By default, the values are shifted by -1024, but in other cases a different constant might be required, which can be set using the *min_to* input.

RIA_header is a list containing the most basic patient and examination information present in the nrrd file.

RIA_log is a list of variables, which give an overview of what has been done with the image. If the whole *RIA_image* is supplied to a function, the information regarding the manipulations are written into the *$events* array in chronological order. Furthermore, some additional information is also saved in the log, which might be needed for further analysis.

Value

Returns a *RIA_image* object. *RIA_image* is a list with three mandatory attributes.

- **RIA_data** is a *RIA_data* object containing the image in *$orig* slot.
- **RIA_header** is a *RIA_header* object, which is s list of nrrd information.
- **RIA_log** is a *RIA_log* object, which is a list updated by RIA functions and acts as a log and possible input for some functions.

References

Márton KOLOSSVÁRY et al. Radiomic Features Are Superior to Conventional Quantitative Computed Tomographic Metrics to Identify Coronary Plaques With Napkin-Ring Sign Circulation: Cardiovascular Imaging (2017). DOI: 10.1161/circimaging.117.006843 https://pubmed.ncbi.nlm.nih.gov/29233836/

Márton KOLOSSVÁRY et al. Cardiac Computed Tomography Radiomics: A Comprehensive Review on Radiomic Techniques. Journal of Thoracic Imaging (2018). DOI: 10.1097/RTI.0000000000000268 https://pubmed.ncbi.nlm.nih.gov/28346329/

Examples

```
## Not run:
 #Image will be croped to smallest bounding box, and smallest values will be changed to NA,
 while 1024 will be substracted from all other data points.
 RIA_image <- load_nrrd("/Users/Test/Documents/Radiomics/John_Smith/nrrd_folder/sample.nrrd")

## End(Not run)
```

---

merge_RIA_images *Merges multiple loaded images into one volume*

Description

Merges multiple *RIA_image* class objects loaded using any of the load functions. All images need to have the same dimensions. Further, during loading the images should not be cropped to assure that the orientation and position of the data is maintained. Data of the new combined image is updated sequentially, using data from the *data$orig* slot, that is only parts of the image that do not have data (which are converted to NA during the load process) are updated in the order of provided

RIA_images. If multiple images contain data in for the same element, the first value is used in the new image. Data in the *data$log* slot is updated based on the new combined image, while data in the *data$header* slot is copied from the first provided image.

Usage

```
merge_RIA_images(RIA_data_in, crop_in = TRUE, verbose_in = TRUE)
```

Arguments

`RIA_data_in`	List of Multiple *RIA_images*.
`crop_in`	logical, indicating whether to crop the merged image to smallest bounding box.
`verbose_in`	logical indicating whether to print detailed information. Most prints can also be suppressed using the `suppressMessages` function.

Value

RIA_image containing the merged volume with updated log and header data

References

Márton KOLOSSVÁRY et al. Radiomic Features Are Superior to Conventional Quantitative Computed Tomographic Metrics to Identify Coronary Plaques With Napkin-Ring Sign Circulation: Cardiovascular Imaging (2017). DOI: 10.1161/circimaging.117.006843 `https://pubmed.ncbi.nlm.nih.gov/29233836/`

Márton KOLOSSVÁRY et al. Cardiac Computed Tomography Radiomics: A Comprehensive Review on Radiomic Techniques. Journal of Thoracic Imaging (2018). DOI: 10.1097/RTI.0000000000000268 `https://pubmed.ncbi.nlm.nih.gov/28346329/`

Examples

```
## Not run:
#Load multiple images and combine them
d1 <- load_nifti(ABC_p1.nii.gz, crop_in = FALSE)
d2 <- load_nifti(ABC_p2.nii.gz, crop_in = FALSE)
d  <- merge_RIA(list(d1, d2))

## End(Not run)
```

`Non_NRS` RIA_image *object of a plaque without the napkin-ring sign*

Description

rda containing an example *RIA_image* object of a patients plaque which does not show the napkin-ring sign.

Usage

```
NRS
```

Format

RIA_image object

Value

RIA_image object

References

Márton KOLOSSVÁRY et al. Radiomic Features Are Superior to Conventional Quantitative Computed Tomographic Metrics to Identify Coronary Plaques With Napkin-Ring Sign Circulation: Cardiovascular Imaging (2017). DOI: 10.1161/circimaging.117.006843 https://pubmed.ncbi.nlm.nih.gov/29233836/

Márton KOLOSSVÁRY et al. Cardiac Computed Tomography Radiomics: A Comprehensive Review on Radiomic Techniques. Journal of Thoracic Imaging (2018). DOI: 10.1097/RTI.0000000000000268 https://pubmed.ncbi.nlm.nih.gov/28346329/

`NRS` RIA_image *object of a plaque with the napkin-ring sign*

Description

rda containing an example *RIA_image* object of a patients plaque which shows the napkin-ring sign.

Usage

```
NRS
```

Format

RIA_image object

Value

RIA_image object

References

Márton KOLOSSVÁRY et al. Radiomic Features Are Superior to Conventional Quantitative Computed Tomographic Metrics to Identify Coronary Plaques With Napkin-Ring Sign Circulation: Cardiovascular Imaging (2017). DOI: 10.1161/circimaging.117.006843 `https://pubmed.ncbi.nlm.nih.gov/29233836/`

Márton KOLOSSVÁRY et al. Cardiac Computed Tomography Radiomics: A Comprehensive Review on Radiomic Techniques. Journal of Thoracic Imaging (2018). DOI: 10.1097/RTI.0000000000000268 `https://pubmed.ncbi.nlm.nih.gov/28346329/`

`radiomics_all` *Calculates all radiomic statistics on supplied RIA_image*

Description

Calculates specified radiomic statistics on *RIA_image*. Parameters of radiomic functions may be set. By default the the images are discretized to 8, 16 and 32 bins using equally sized and probable binning. First-order statistics are calculated on the original image and if asked then on all discretizations. Symmetric GLCMs are calculated for all directions at a distance of 1 for all discretizations. GLRLMs are also calculated for all discretizations. Geometry-based statistics are calculated for the original image as well as all discretizations is requested.

Usage

```
radiomics_all(
  RIA_data_in,
  bins_in = c(8, 16, 32),
  equal_prob = "both",
  fo_discretized = FALSE,
  distance = c(1),
  statistic = "mean(X, na.rm = TRUE)",
  geometry_discretized = TRUE,
  verbose_in = TRUE
)
```

Arguments

`RIA_data_in`	*RIA_image*.
`bins_in`	integer vector, number of bins specified.
`equal_prob`	logical or string, indicating to cut data into bins with equal relative frequencies. If FALSE, then equal interval bins will be used. If *"both"* is supplied, the both equally probable and equal interval bins will be created.
`fo_discretized`	logical, indicating whether to calculate first-order statistics on discretized images.
`distance`	integer, distance between the voxels being compared.

`statistic`	string, defining the statistic to be calculated on the array of GLCM statistics. By default, statistic is set to *"mean"*, however any function may be provided. The proper syntax is: function(X, attributes). The supplied string must contain a "X", which will be replaced with the array of the GLCM statistics value. Further attributes of the function may also be given. For example, if you wish to calculate the median of all GLCMs calculated in different directions, then it must be supplied as: *median(X, na.rm = TRUE)*.
`geometry_discretized`	logical, indicating whether to calculate geometry-based statistics on discretized images.
`verbose_in`	logical, indicating whether to print detailed information. Most prints can also be suppressed using the `suppressMessages` function.

Value

RIA_image containing the statistical information.

References

Márton KOLOSSVÁRY et al. Radiomic Features Are Superior to Conventional Quantitative Computed Tomographic Metrics to Identify Coronary Plaques With Napkin-Ring Sign Circulation: Cardiovascular Imaging (2017). DOI: 10.1161/circimaging.117.006843 `https://pubmed.ncbi.nlm.nih.gov/29233836/`

Márton KOLOSSVÁRY et al. Cardiac Computed Tomography Radiomics: A Comprehensive Review on Radiomic Techniques. Journal of Thoracic Imaging (2018). DOI: 10.1097/RTI.0000000000000268 `https://pubmed.ncbi.nlm.nih.gov/28346329/`

Examples

```
## Not run:
#Discretize loaded image and then calculate all radiomic statistics
RIA_image <- radiomics_all(RIA_image, equal_prob = "both", bins_in= c(32,64), distance = c(1:2))

## End(Not run)
```

`save_RIA` *Export radiomics calculations of RIA image to csv*

Description

Exports given slots of statistics from RIA_image. Names of slots have to be defined which the user wishes to export using the *stats* parameter. Using the *group_name* parameter the user can lable the cases with a group ID, for example "Case", which can be used as a grouping variable for further analysis.

Usage

```
save_RIA(
  RIA_image,
  save_to = "C:/",
  save_name = "RIA_stat",
  group_name = "Case",
  stats = c("stat_fo", "stat_glcm_mean", "stat_glrlm_mean", "stat_geometry")
)
```

Arguments

`RIA_image`	*RIA_image* with calculated statistics.
`save_to`	string, path of folder to save results to.
`save_name`	string, path of folder to save results to.
`group_name`	string, a ID defining which group the case belongs to.
`stats`	string vector, identifing which slots to export

References

Márton KOLOSSVÁRY et al. Radiomic Features Are Superior to Conventional Quantitative Computed Tomographic Metrics to Identify Coronary Plaques With Napkin-Ring Sign Circulation: Cardiovascular Imaging (2017). DOI: 10.1161/circimaging.117.006843 `https://pubmed.ncbi.nlm.nih.gov/29233836/`

Márton KOLOSSVÁRY et al. Cardiac Computed Tomography Radiomics: A Comprehensive Review on Radiomic Techniques. Journal of Thoracic Imaging (2018). DOI: 10.1097/RTI.0000000000000268 `https://pubmed.ncbi.nlm.nih.gov/28346329/`

Index