cnr-05-all-subjects

December 2, 2021

1 TUMOR - CNR - Characteristic for `roi` and `dil2` mask for all subjects (CC1, CC2, CC3)

Created: 2021.11.24 / Updated: 2021.12.02

ver: 0.02

- function: `load_subject_image_resolution_and_voxel_size()`
- count voxel volume, volume of roi/mask,
- save tables to latex file format,
- save notebook as pdf.

[1]:
```
%load_ext autoreload
%autoreload 2
```

[2]:
```
%matplotlib inline

import os
import pathlib
import glob

import numpy as np
import pandas as pd
import nibabel as nib
import matplotlib.pyplot as plt
```

[3]:
```
import utils
import functions1 as f1
```

2 GLOBAL VARIABLES

[4]:
```
HOME_DIR = pathlib.Path(os.getcwd()).parent
DATA_DIR = HOME_DIR / 'data'
RESULTS_DIR = DATA_DIR / 'results'
PLOT_DIR = DATA_DIR / 'plots'
TABLE_DIR = DATA_DIR / 'tables'
```

```
CURRENT_NOTEBOOK_NAME = 'cnr-05-all-subjects'
CURRENT_NOTEBOOK_NUMBER = CURRENT_NOTEBOOK_NAME.split('-')[1]
```

---

3 OUR FUNCTIONS

[5]:
```
def load_images_for_subject(sub_name, folder=RESULTS_DIR):
    """
    Loades images for a subject:

    Parameters:
    -------------------------------
    sub_name - subject prefix, e.g., 'CC1'
    folder - path to the folter with images, by default it is RESULTS_DIR (./
 ↪data/results).

    Returns:
    -------------------------------
    img - loaded image,
    roi - loaded roi image,
    bla - loaded bladder image

    C: 2021.11.28 / U: 2021.11.30
    """
    img = nib.load(RESULTS_DIR / f'{sub_name}_t2.nii.gz').get_fdata()
    roi = nib.load(RESULTS_DIR / f'{sub_name}_t2_roi.nii.gz').get_fdata()
    bla = nib.load(RESULTS_DIR / f'{sub_name}_t2_bladder.nii.gz').get_fdata()

    return img, roi, bla
```

[6]:
```
def read_subject_image_resolution_and_voxel_size(sub_name, folder=RESULTS_DIR):
    """
    For selected subject gets images resolution and voxel size.
    Those values for all images should be the same, however we do this for each␣
 ↪image type separately
    just to make sure that they are the same.

    Parameters:
    -------------------------------
    sub_name - subject prefix, e.g., 'CC1'
    folder - path to the folter with images, by default it is RESULTS_DIR (./
 ↪data/results).

    Returns:
    -------------------------------
```

```
    df - data frame with all values


    C: 2021.12.02 / U: 2021.12.02
    """
    img_nii = nib.load(RESULTS_DIR / f'{sub_name}_t2.nii.gz')
    roi_nii = nib.load(RESULTS_DIR / f'{sub_name}_t2_roi.nii.gz')
    bla_nii = nib.load(RESULTS_DIR / f'{sub_name}_t2_bladder.nii.gz')

    # for IGM
    img_hdr = img_nii.header
    dct_img = {}
    dct_img['sub'] = sub_name
    dct_img['mask'] = 'IMG'
    dct_img['res']= [img_hdr['dim'][1:4]]
    dct_img['vox_size'] = [img_hdr['pixdim'][1:4]]
    df_img = pd.DataFrame.from_dict(dct_img)

    # for ROI
    roi_hdr = roi_nii.header
    dct_roi = {}
    dct_roi['sub'] = sub_name
    dct_roi['mask'] = 'ROI'
    dct_roi['res'] = [roi_hdr['dim'][1:4]]
    dct_roi['vox_size'] = [roi_hdr['pixdim'][1:4]]
    df_roi = pd.DataFrame.from_dict(dct_roi)

    # for BLA
    bla_hdr = bla_nii.header
    dct_bla = {}
    dct_bla['sub'] = sub_name
    dct_bla['mask'] = 'BLA'
    dct_bla['res']= [bla_hdr['dim'][1:4]]
    dct_bla['vox_size'] = [bla_hdr['pixdim'][1:4]]
    df_bla = pd.DataFrame.from_dict(dct_bla)

    return pd.concat([df_img, df_roi, df_bla], axis=0)
```

[7]:
```
def im_info(im, name='image'):
    """
    The basic info about image.

    Parameters:
    ---------------
    im - image,
    name - displayed image name (hepful in multiple use in the row)
    skip_zeros - skip voxels with zero values
```

```
    C: 2021.11.06 / U:2021.11.30
    """
    if isinstance(im, np.ndarray):
        im = [im]
    if isinstance(name, str):
        name = [name]
    if len(im) != len(name):
        print('Wrong lists length! Fix it :P ')
        return None

    print('\nFor 3D image(s) printed values take into account background voxels␣
↪as well:')
    for i,n in zip(im,name):
        print(f'\t*** {n.upper()} ***,\tmax={i.max()}, min={i.min()}, mean={i.
↪mean():.2f}, shape={i.shape}, #voxels={i.size}')
```

[8]:
```
# prepare_voxel_image_under_mask
def get_voxles_inside_mask(img, mask):
    """
    Returns an 3D image with voxels values included inside a maks and the same␣
↪voxels stored in a vector.

    Assumption: Voxels inside mask are equalled to 1.

    C: 2021.11.28 / U: 2021.11.30
    """
    image_shape = np.where(mask, img, 0)
    voxel_vector = img[mask==1]

    return image_shape, voxel_vector
```

[9]:
```
# msk_info_as_df
def voxel_vector_stats_as_df(vox_vec, sub, msk_type, res_vox_size_df=None):
    """
    Basic statistics about voxels in a vector.

    Parameters
    ----------------------------------
    voxel_vector - vector with voxels to get info about
    sub - subject name, e.g., CC1
    msk_type - type of mask image, to build the full name with subject and mask␣
↪name e.g., roi, dil2, ball5,..
    res_vox_size_df - df with read image resolution and voxel sizes for; img,␣
↪roi, bla.
```

```
    C: 2021.11.28 / U: 2021.12.02
    """

    if vox_vec.ndim > 1:
        print('We need a vector not a matrix! Fix it now! :P ')
        return None

    d={'sub':[sub],'mask':[msk_type] ,'max':[vox_vec.max()],
       'min':[vox_vec.min()], 'mean':[vox_vec.mean()], 'std':[vox_vec.std()],␣
↪'#vox':[vox_vec.size]}


    if res_vox_size_df is not None:
        tmp_df = res_vox_size_df.loc[(res_vox_size_df['sub'] == sub) &␣
↪(res_vox_size_df['mask'] == msk_type)]
        d['res'] = tmp_df['res']
        d['vox_size'] = tmp_df['vox_size']

        # calculate volume of the single voxel
        vs = tmp_df['vox_size'].values[0]
        vox_vol = vs[0] * vs[1] * vs[2]
        d['vox_vol'] = vox_vol

        # calculate volume of the objece inside  mask
        d['vol'] = vox_vol * d['#vox'][0]

    df = pd.DataFrame.from_dict(d)

    # set display precision
    df.loc[:, "mean"] = df["mean"].map('{:.2f}'.format)
    df.loc[:, "std"] = df["std"].map('{:.3f}'.format)


    if 'vox_vol' in df.columns: df.loc[:, "vox_vol"] = df["vox_vol"].map('{:.
↪3f}'.format)
    if 'vol' in df.columns: df.loc[:, "vol"] = df["vol"].map('{:.1f}'.format)

    return df
```

[10]:
```
def show_not_empty_slices_in_3d_mask_image(msk_img, mask_name, sub, fontsize=18,␣
↪figsize=(24,20), cmap='gray',
                                           notebook_nr=CURRENT_NOTEBOOK_NUMBER):
    """
    Show all slices from a 3D image/mask wihth not empty slices, i.e. only those␣
↪slices that belonge to the mask.

    C: 2021.11.28 / 2021.11.28
```

```
    """

    slices_idx = np.where(msk_img >0)
    r1 = slices_idx[2].min()-1
    r2 = slices_idx[2].max()+1
    slices_rng = r2 - r1

    if slices_rng <=15: cols=3
    else:cols=4

    r,c = divmod(slices_rng,cols)
    #if c == 0: rows = r
    # else:
    rows = r + 1

    f, ax = plt.subplots(rows,cols,sharex=True, sharey=True, figsize=figsize)
    axf=ax.flat[:]

    for sl in range(slices_rng+1):
        sl_im = sl + r1
        axf[sl].imshow(msk_img[:,:,sl_im], cmap=cmap)
        axf[sl].set_title(f'sl={sl+r1}', fontsize=fontsize)

    for k in range(slices_rng+2, rows*cols+1):
        axf[k-1].set_axis_off()

    title = f'Consequtive not-empty slices of a {mask_name.upper()} image for␣
↪subject {sub}'
    _=plt.suptitle(title, fontsize=fontsize+4, fontweight='bold')
    plt.tight_layout()
    plt.subplots_adjust(top=0.94)



    save_name = PLOT_DIR / f'{notebook_nr}-{sub}-{mask_name}.png'
    print(f'Figure saved to:\n\t{save_name}')
    plt.savefig(save_name)

    plt.show()
```

[11]:
```
def calculate_CNRs(roi_vec, rim_vec, bla_vec, sub):
    """

    C: 2021.11.28 / U: 2021.11.30
    """

    CNR_std = (roi_vec.mean() - rim_vec.mean()) / bla_vec.std()
```

```
    #print(f'CNR_std={CNR_std:.3f}')
    CNR_mean = (roi_vec.mean() - rim_vec.mean()) / bla_vec.mean()
    #print(f'CNR_mean={CNR_mean:.3f}')

    d = {'sub':[sub],'CNR_std':[CNR_std], 'SNR_mean':[CNR_mean]}
    df = pd.DataFrame.from_dict(d)
    # df.index = [sub]

    df.loc[:, "CNR_std"] = df["CNR_std"].map('{:.2f}'.format)
    df.loc[:, "SNR_mean"] = df["SNR_mean"].map('{:.2f}'.format)

    return df
```

[12]:
```
def _textWrap(text, spaces=4, hashes=3):
    """
    C: 2019.06.18
    M: 2021.11.30
    """
    print()
    p = len(text) + (hashes*2 + spaces*2)
    sp = spaces*' '
    h = hashes*'#'
    print(p * '#')
    print(f"{h}{sp}{text}{sp}{h}")
    print(p * '#')
```

[13]:
```
def save_df_to_latex(df, save_file_root_name='a-table', folder=TABLE_DIR,␣
 ↪notebook_nr=CURRENT_NOTEBOOK_NUMBER, **kw):
    """
    Save df to latex.

    C: 2021.12.02 / U: 2021.12.02
    """
    if not save_file_root_name.endswith('.tex'): save_file_root_name += '.tex'
    save_path = folder / f'{notebook_nr}-{save_file_root_name}'

    df.to_latex(save_path)
    print(f'Table saved to:\n\t{save_path}\n')
```

[14]:
```
def pipeline(sub, info=False, show_mask_slices=False, cmap='nipy_spectral'):
    """
    Full pipeline for one subject.

    Paramteres:
    ----------------------------------------
    sub: subject prefix (e.g., CC1)
    info: if True prints additional statistisc information
```

```
    show_mask_slices: if True display content of masks slices

    C: 2021.11.30 / U: 2021.11.30
    """

    # load images from disc
    img, roi, bla = load_images_for_subject(sub)

    # read image resolution and voxel size
    res_vox_size_df = read_subject_image_resolution_and_voxel_size(sub)
    res_vox_size_df.index = range(res_vox_size_df.shape[0])

    # prepare df with statistics for img
    img_voxel_vec_df = voxel_vector_stats_as_df(img.ravel(), sub, 'IMG',␣
↪res_vox_size_df)

    # load expanded roi and create rim image
    full_image_name = f'{sub}_t2_roi_dilated2.nii.gz'
    extanded_roi = nib.load(RESULTS_DIR / full_image_name).get_fdata()
    rim = extanded_roi - roi

    # copy voxels from an image img to roi image
    roi_voxel_image, roi_voxel_vec = get_voxles_inside_mask(img, roi)
    roi_voxel_vec_df = voxel_vector_stats_as_df(roi_voxel_vec, sub, 'ROI',␣
↪res_vox_size_df)

    # copy voxesl from img to rim image
    rim_voxel_image, rim_voxel_vec = get_voxles_inside_mask(img, rim)
    rim_voxel_vec_df = voxel_vector_stats_as_df(rim_voxel_vec, sub, 'RIM')

    # manualy set imager resolution and voxel size to RIM image as there is no␣
↪header
    rim_voxel_vec_df['res'] = roi_voxel_vec_df['res'].to_numpy()
    rim_voxel_vec_df['vox_size'] = roi_voxel_vec_df['vox_size'].to_numpy()
    # calculate volume of the single voxel
    vs = roi_voxel_vec_df['vox_size'].values[0]
    vox_vol = vs[0] * vs[1] * vs[2]
    rim_voxel_vec_df['vox_vol'] = vox_vol
    # calculate volume of the objece inside  mask
    rim_voxel_vec_df['vol'] = vox_vol * rim_voxel_vec_df['#vox'][0]
    rim_voxel_vec_df.loc[:, "vox_vol"] = rim_voxel_vec_df["vox_vol"].map('{:.
↪3f}'.format)
    rim_voxel_vec_df.loc[:, "vol"] = rim_voxel_vec_df["vol"].map('{:.1f}'.format)


    # copy voxels from img to bladder (bla image)
    bla_voxel_image, bla_voxel_vec = get_voxles_inside_mask(img, bla)
```

```
    bla_voxel_vec_df = voxel_vector_stats_as_df(bla_voxel_vec, sub, 'BLA',␣
↪res_vox_size_df)


    # concatenate all df with stats into one df
    ccX_stats = pd.concat([img_voxel_vec_df, roi_voxel_vec_df, rim_voxel_vec_df,␣
↪bla_voxel_vec_df], axis=0)

    # calculate cnr for a subject
    ccX_snr = calculate_CNRs(roi_voxel_vec, rim_voxel_vec, bla_voxel_vec, sub)


    if info:

        _textWrap(sub, spaces=6, hashes=4)


        imgs = [img,roi,bla, rim]
        names = ['Loaded img', 'Loaded roi','Loaded bladder', 'Prepared rim']

        imgs.extend([roi_voxel_image, roi_voxel_vec])
        names.extend(['Image with voxels inside ROI mask','Vector with voxels␣
↪inside ROI mask'])

        imgs.extend([rim_voxel_image, rim_voxel_vec])
        names.extend(['Image with voxels inside RIM mask','Vector with voxels␣
↪inside RIM mask'])

        imgs.extend([bla_voxel_image, bla_voxel_vec])
        names.extend(['Image with voxels inside BLADDER mask','Vector with␣
↪voxels inside BLADDER mask'])

        im_info(imgs,names)


    if show_mask_slices:
        # roi
        mask_name = 'roi'
        show_not_empty_slices_in_3d_mask_image(roi_voxel_image, mask_name, sub,␣
↪cmap=cmap)

        # rim
        mask_name = 'rim'
        show_not_empty_slices_in_3d_mask_image(rim_voxel_image, mask_name, sub,␣
↪cmap=cmap)
```

```
        # bladder
        mask_name = 'bladder'
        show_not_empty_slices_in_3d_mask_image(bla_voxel_image, mask_name, sub,␣
↪cmap=cmap)

    return ccX_stats, ccX_snr
```

---

4 STEP BY STEP ALGORITHM FOR A SINGLE SUBJECT

[15]:
```
sub = 'CC1'
```

4.1 LOAD IMAGES FROM DISC

[16]:
```
img, roi, bla = load_images_for_subject(sub)
```

4.1.1 READ IMAGE RESOLUTION AND VOXEL SIZE FOR SUBJECT (SHOULD BE THE SAME FOR ALL IMAGES)

[17]:
```
res_vox_size_df = read_subject_image_resolution_and_voxel_size(sub)
res_vox_size_df.index = range(res_vox_size_df.shape[0])
display(res_vox_size_df)
```

```
   sub mask             res               vox_size
0  CC1  IMG  [320, 320, 25]  [0.5625, 0.5625, 3.0]
1  CC1  ROI  [320, 320, 25]  [0.5625, 0.5625, 3.0]
2  CC1  BLA  [320, 320, 25]  [0.5625, 0.5625, 3.0]
```

4.1.2 PREPARE VOXEL BRIGHTNESS STATISTICS FOR `IMG` AS A DF

[18]:
```
img_voxel_vec_df = voxel_vector_stats_as_df(img.ravel(), sub, 'IMG',␣
 ↪res_vox_size_df)
img_voxel_vec_df
```

[18]:
```
   sub mask    max  min    mean      std     #vox             res  \
0  CC1  IMG  988.0  0.0  252.81  145.556  2560000  [320, 320, 25]

                vox_size vox_vol        vol
0  [0.5625, 0.5625, 3.0]   0.949  2430000.0
```

4.2 LOAD EXPANDED ROI AND PREPARE RIM IMAGE

[19]:
```
full_image_name = f'{sub}_t2_roi_dilated2.nii.gz'
expanded_roi = nib.load(RESULTS_DIR / full_image_name).get_fdata()
rim = expanded_roi - roi
```

[20]:
```
if 1:
    im_info([img,roi,bla, rim], ['Loaded img', 'Loaded roi','Loaded bladder',␣
 ↪'Prepared rim'])
```

```
For 3D image(s) printed values take into account background voxels as well:
        *** LOADED IMG ***,     max=988.0, min=0.0, mean=252.81, shape=(320,
320, 25), #voxels=2560000
        *** LOADED ROI ***,     max=1.0, min=0.0, mean=0.00, shape=(320, 320,
25), #voxels=2560000
        *** LOADED BLADDER ***, max=1.0, min=0.0, mean=0.00, shape=(320, 320,
25), #voxels=2560000
        *** PREPARED RIM ***,   max=1.0, min=0.0, mean=0.00, shape=(320, 320,
25), #voxels=2560000
```

4.3 NEW ROI IMAGE WITH VOXEL VALUES FROM IMG THAT ARE INCLUDED INSIDE ROI MASK

[21]:
```
# get 3D image and vector of voxels included inside roi
roi_voxel_image, roi_voxel_vec = get_voxles_inside_mask(img, roi)
# voxel stat info saved as df
roi_voxel_vec_df = voxel_vector_stats_as_df(roi_voxel_vec, sub, 'ROI',␣
 ↪res_vox_size_df)

if 1:
    im_info([roi_voxel_image, roi_voxel_vec], ['Image with voxels inside ROI␣
 ↪mask','Vector with voxels inside ROI mask'])
```

```
For 3D image(s) printed values take into account background voxels as well:
        *** IMAGE WITH VOXELS INSIDE ROI MASK ***,      max=507.0, min=0.0,
mean=0.61, shape=(320, 320, 25), #voxels=2560000
        *** VECTOR WITH VOXELS INSIDE ROI MASK ***,     max=507.0, min=0.0,
mean=168.94, shape=(9180,), #voxels=9180
```

[22]:
```
roi_voxel_vec_df
```

[22]:

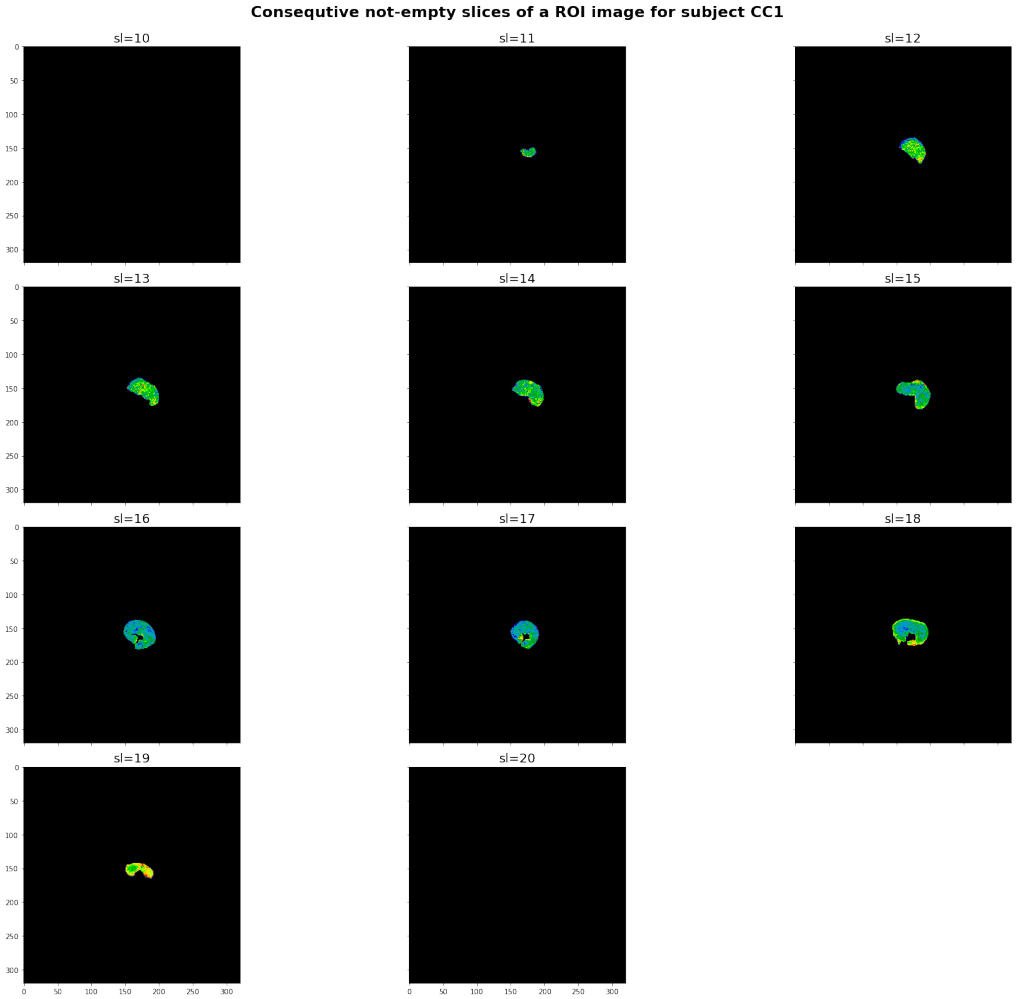
	sub	mask	max	min	mean	std	#vox	res	\
1	CC1	ROI	507.0	0.0	168.94	79.020	9180	[320, 320, 25]	

	vox_size	vox_vol	vol
1	[0.5625, 0.5625, 3.0]	0.949	8713.8

4.3.1 PLOT CONSEQUTIVE SLICES OF ROI IMAGE

[23]:
```
if 1:
    mask_name = 'roi'
    cmap = 'nipy_spectral'
    show_not_empty_slices_in_3d_mask_image(roi_voxel_image, mask_name, sub,␣
 ↪cmap=cmap)
```

```
Figure saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/plots/05-CC1-roi.png
```



4.4 RIM IMAGE FILLED WITH VOXELS FROM IMG

[24]:
```
# 3D image and voxel vector
rim_voxel_image, rim_voxel_vec = get_voxles_inside_mask(img, rim)
# voxel inside rim stats as df
rim_voxel_vec_df = voxel_vector_stats_as_df(rim_voxel_vec, sub, 'RIM')


if 1:
    im_info([rim_voxel_image, rim_voxel_vec], ['Image with voxels inside RIM␣
 ↪mask','Vector with voxels inside RIM mask'])
```

```
For 3D image(s) printed values take into account background voxels as well:
        *** IMAGE WITH VOXELS INSIDE RIM MASK ***,      max=801.0, min=0.0,
mean=1.16, shape=(320, 320, 25), #voxels=2560000
        *** VECTOR WITH VOXELS INSIDE RIM MASK ***,     max=801.0, min=0.0,
mean=301.37, shape=(9872,), #voxels=9872
```

4.4.1 MANUALLY FILL IN VOXEL SIZE INFO AS THERE IS NO HEATHER FOR RIM

[25]:
```
rim_voxel_vec_df['res'] = roi_voxel_vec_df['res'].to_numpy()
rim_voxel_vec_df['vox_size'] = roi_voxel_vec_df['vox_size'].to_numpy()

# calculate volume of the single voxel
vs = roi_voxel_vec_df['vox_size'].values[0]
vox_vol = vs[0] * vs[1] * vs[2]
rim_voxel_vec_df['vox_vol'] = vox_vol

# calculate volume of the objece inside  mask
rim_voxel_vec_df['vol'] = vox_vol * rim_voxel_vec_df['#vox'][0]

rim_voxel_vec_df.loc[:, "vox_vol"] = rim_voxel_vec_df["vox_vol"].map('{:.3f}'.
 ↪format)
rim_voxel_vec_df.loc[:, "vol"] = rim_voxel_vec_df["vol"].map('{:.1f}'.format)
```

[26]:
```
rim_voxel_vec_df
```

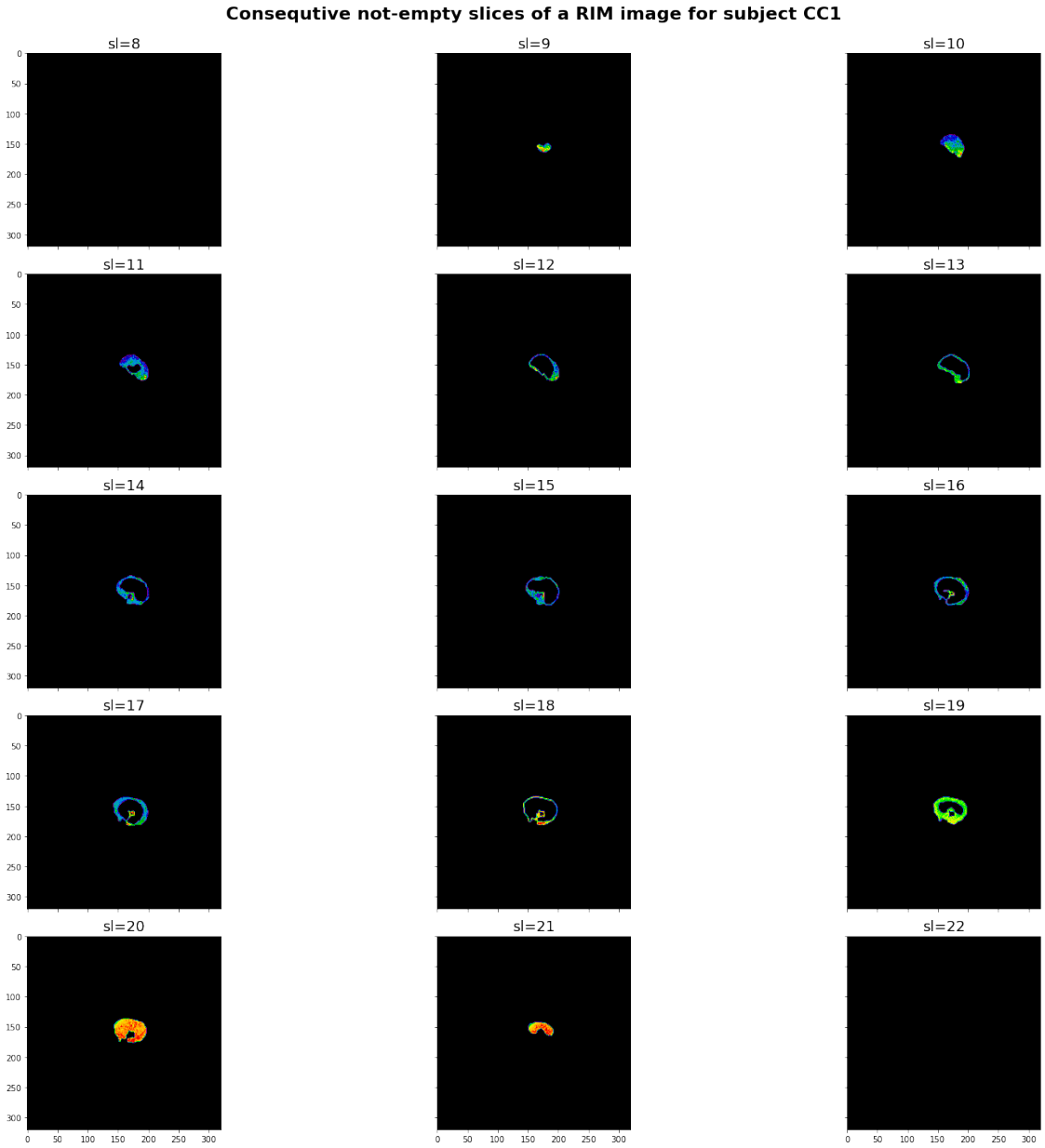
[26]:

	sub	mask	max	min	mean	std	#vox	res	vox_size	vox_vol	vol
0	CC1	RIM	801.0	0.0	301.37	177.010	9872	[320, 320, 25]	[0.5625, 0.5625, 3.0]	0.949	9370.7

4.4.2 PLOT CONSEQUTIVE SLICES FROM RIM IMAGE

[27]:
```
if 1:
    cmap='nipy_spectral'
    mask_name = 'rim'
    show_not_empty_slices_in_3d_mask_image(rim_voxel_image, mask_name, sub,␣
 ↪cmap=cmap)
```

```
Figure saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/plots/05-CC1-rim.png
```



4.5 BLADDER IMAGE FILLED WITH VOXELS FROM IMG

[28]:
```
bla_voxel_image, bla_voxel_vec = get_voxles_inside_mask(img, bla)
bla_voxel_vec_df = voxel_vector_stats_as_df(bla_voxel_vec, sub, 'BLA',␣
 ↪res_vox_size_df)

if 1:
    im_info([bla_voxel_image, bla_voxel_vec], ['Image with voxels inside BLADDER␣
 ↪mask','Vector with voxels inside BLADDER mask'])
```

```
For 3D image(s) printed values take into account background voxels as well:
        *** IMAGE WITH VOXELS INSIDE BLADDER MASK ***,  max=462.0, min=0.0,
mean=0.23, shape=(320, 320, 25), #voxels=2560000
        *** VECTOR WITH VOXELS INSIDE BLADDER MASK ***, max=462.0, min=294.0,
mean=378.78, shape=(1572,), #voxels=1572
```

[29]:
```
bla_voxel_vec_df
```

[29]:

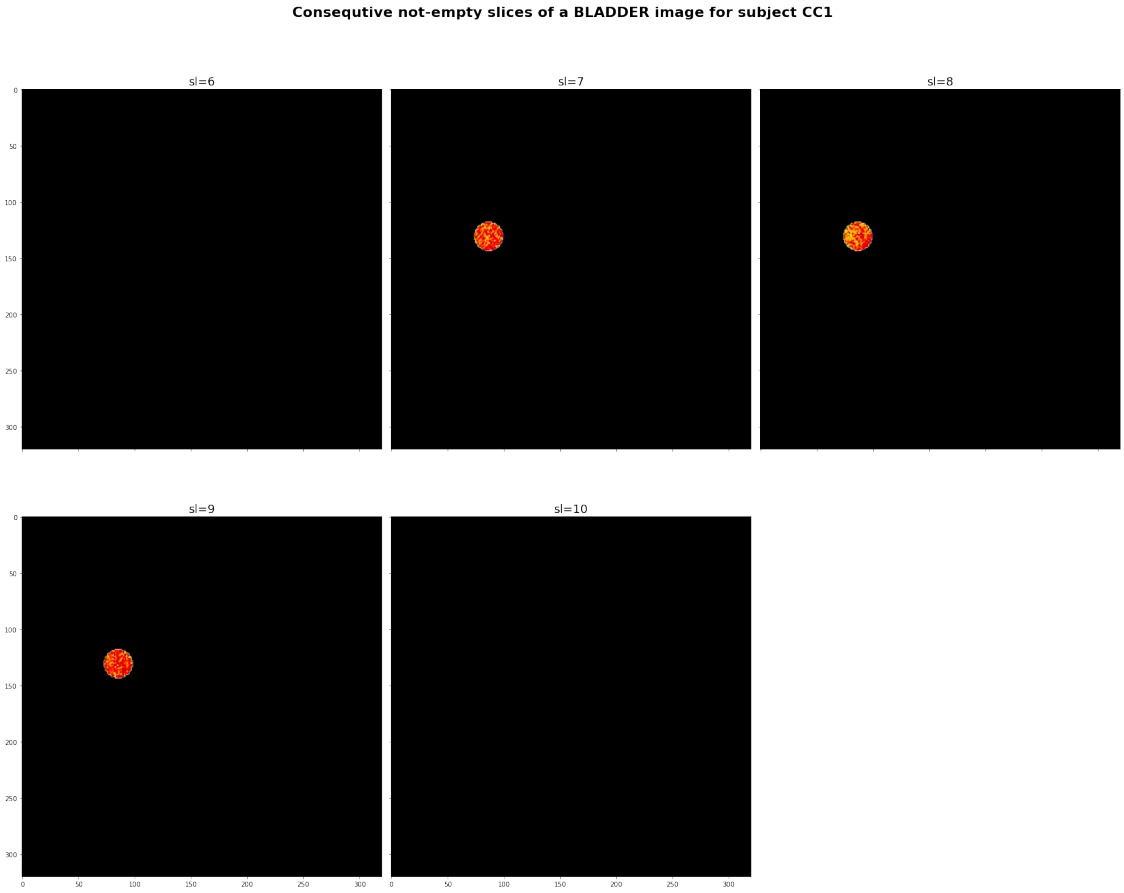
	sub	mask	max	min	mean	std	#vox	res	\
2	CC1	BLA	462.0	294.0	378.78	26.680	1572	[320, 320, 25]	

	vox_size	vox_vol	vol
2	[0.5625, 0.5625, 3.0]	0.949	1492.2

4.5.1 PLOT CONSEQUTIVE SLICES FROM BLADDER IMAGE

[30]:
```
if 1:
    mask_name = 'bladder'
    cmap='nipy_spectral'
    #tit =  f'Consequtive not-empty slices of a {mask_name.upper()} image for␣
 ↪subject {sub}'
    show_not_empty_slices_in_3d_mask_image(bla_voxel_image, mask_name, sub,␣
 ↪cmap=cmap)
```

```
Figure saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/plots/05-CC1-bladder.png
```



4.6 PRINT ALL PARAMETERS IN ONE TABLE

[31]:
```
cc1 = pd.concat([img_voxel_vec_df, roi_voxel_vec_df, rim_voxel_vec_df,␣
 ↪bla_voxel_vec_df], axis=0)
cc1.index = range(cc1.shape[0])
cc1
```

[31]:

	sub	mask	max	min	mean	std	#vox	res	\
0	CC1	IMG	988.0	0.0	252.81	145.556	2560000	[320, 320, 25]	
1	CC1	ROI	507.0	0.0	168.94	79.020	9180	[320, 320, 25]	
2	CC1	RIM	801.0	0.0	301.37	177.010	9872	[320, 320, 25]	
3	CC1	BLA	462.0	294.0	378.78	26.680	1572	[320, 320, 25]	

	vox_size	vox_vol	vol
0	[0.5625, 0.5625, 3.0]	0.949	2430000.0
1	[0.5625, 0.5625, 3.0]	0.949	8713.8
2	[0.5625, 0.5625, 3.0]	0.949	9370.7
3	[0.5625, 0.5625, 3.0]	0.949	1492.2

4.7 CNRs FOR A SINGLE SUBJECT

[32]:
```
cc1 = calculate_CNRs(roi_voxel_vec, rim_voxel_vec, bla_voxel_vec, 'cc1')
cc1
```

[32]:

	sub	CNR_std	SNR_mean
0	cc1	-4.96	-0.35

---

5 STATS FOR ALL SUBJECTS - WITH THE USE OF PIPELINE

[33]:
```
stat1, cnr1 = pipeline('CC1', info=False, show_mask_slices=False)
stat2, cnr2 = pipeline('CC2', info=False, show_mask_slices=False)
stat3, cnr3 = pipeline('CC3', info=False, show_mask_slices=False)
```

5.0.1 CONCATENATE STATS IN ONE DATA FRAME

[34]:
```
stats = pd.concat([stat1, stat2, stat3])
stats.index = range(stats.shape[0])
save_df_to_latex(stats, save_file_root_name='all-subs-full-table')
stats
```

```
Table saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/tables/05-all-subs-full-table.tex
```

[34]:

	sub	mask	max	min	mean	std	#vox	res	\
0	CC1	IMG	988.0	0.0	252.81	145.556	2560000	[320, 320, 25]	
1	CC1	ROI	507.0	0.0	168.94	79.020	9180	[320, 320, 25]	
2	CC1	RIM	801.0	0.0	301.37	177.010	9872	[320, 320, 25]	
3	CC1	BLA	462.0	294.0	378.78	26.680	1572	[320, 320, 25]	
4	CC2	IMG	1771.0	0.0	336.25	254.844	4128768	[384, 384, 28]	
5	CC2	ROI	613.0	184.0	405.74	76.955	1216	[384, 384, 28]	
6	CC2	RIM	1007.0	92.0	257.82	115.222	2462	[384, 384, 28]	
7	CC2	BLA	673.0	589.0	628.98	16.266	2064	[384, 384, 28]	
8	CC3	IMG	1640.0	0.0	443.37	326.626	2867200	[320, 320, 28]	
9	CC3	ROI	939.0	15.0	265.72	75.046	25999	[320, 320, 28]	
10	CC3	RIM	1220.0	0.0	520.81	286.132	16420	[320, 320, 28]	
11	CC3	BLA	660.0	544.0	609.81	19.216	300	[320, 320, 28]	

	vox_size	vox_vol	vol
0	[0.5625, 0.5625, 3.0]	0.949	2430000.0
1	[0.5625, 0.5625, 3.0]	0.949	8713.8
2	[0.5625, 0.5625, 3.0]	0.949	9370.7

```
3          [0.5625, 0.5625, 3.0]   0.949     1492.2
4    [0.5208333, 0.5208333, 3.3]   0.895  3695999.8
5    [0.5208333, 0.5208333, 3.3]   0.895     1088.5
6    [0.5208333, 0.5208333, 3.3]   0.895     2203.9
7    [0.5208333, 0.5208333, 3.3]   0.895     1847.7
8            [0.625, 0.625, 3.3]   1.289  3696000.0
9            [0.625, 0.625, 3.3]   1.289    33514.3
10           [0.625, 0.625, 3.3]   1.289    21166.4
11           [0.625, 0.625, 3.3]   1.289      386.7
```

5.0.2 STATS FOR IMGs

[35]:
```
stats_img = stats.loc[stats['mask'] == 'IMG']
save_df_to_latex(stats_img, save_file_root_name='all-subs-img')
stats_img
```

```
Table saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/tables/05-all-subs-img.tex
```

[35]:
```
   sub mask     max  min    mean      std     #vox             res  \
0  CC1  IMG   988.0  0.0  252.81  145.556  2560000  [320, 320, 25]
4  CC2  IMG  1771.0  0.0  336.25  254.844  4128768  [384, 384, 28]
8  CC3  IMG  1640.0  0.0  443.37  326.626  2867200  [320, 320, 28]

                      vox_size vox_vol        vol
0        [0.5625, 0.5625, 3.0]   0.949  2430000.0
4  [0.5208333, 0.5208333, 3.3]   0.895  3695999.8
8          [0.625, 0.625, 3.3]   1.289  3696000.0
```

STATS FOR ROI

[36]:
```
stats_roi = stats.loc[stats['mask'] == 'ROI']
save_df_to_latex(stats_roi, save_file_root_name='all-subs-roi')
stats_roi
```

```
Table saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/tables/05-all-subs-roi.tex
```

[36]:
```
   sub mask    max    min    mean     std   #vox             res  \
1  CC1  ROI  507.0    0.0  168.94  79.020   9180  [320, 320, 25]
5  CC2  ROI  613.0  184.0  405.74  76.955   1216  [384, 384, 28]
9  CC3  ROI  939.0   15.0  265.72  75.046  25999  [320, 320, 28]
```

```
                      vox_size vox_vol      vol
1         [0.5625, 0.5625, 3.0]   0.949   8713.8
5   [0.5208333, 0.5208333, 3.3]   0.895   1088.5
9           [0.625, 0.625, 3.3]   1.289  33514.3
```

STATS FOR RIM

[37]:
```
stats_rim = stats.loc[stats['mask'] == 'RIM']
save_df_to_latex(stats_rim, save_file_root_name='all-subs-rim')
stats_rim
```

```
Table saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/tables/05-all-subs-rim.tex
```

[37]:
```
    sub mask     max   min    mean      std   #vox             res  \
2   CC1  RIM   801.0   0.0  301.37  177.010   9872  [320, 320, 25]
6   CC2  RIM  1007.0  92.0  257.82  115.222   2462  [384, 384, 28]
10  CC3  RIM  1220.0   0.0  520.81  286.132  16420  [320, 320, 28]

                      vox_size vox_vol      vol
2         [0.5625, 0.5625, 3.0]   0.949   9370.7
6   [0.5208333, 0.5208333, 3.3]   0.895   2203.9
10          [0.625, 0.625, 3.3]   1.289  21166.4
```

5.0.3 STATS FOR BLADDER

[38]:
```
stats_bla = stats.loc[stats['mask'] == 'BLA']
save_df_to_latex(stats_bla, save_file_root_name='all-subs-bla')
stats_bla
```

```
Table saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/tables/05-all-subs-bla.tex
```

[38]:
```
    sub mask    max    min    mean     std  #vox             res  \
3   CC1  BLA  462.0  294.0  378.78  26.680  1572  [320, 320, 25]
7   CC2  BLA  673.0  589.0  628.98  16.266  2064  [384, 384, 28]
11  CC3  BLA  660.0  544.0  609.81  19.216   300  [320, 320, 28]

                      vox_size vox_vol     vol
3         [0.5625, 0.5625, 3.0]   0.949  1492.2
7   [0.5208333, 0.5208333, 3.3]   0.895  1847.7
11          [0.625, 0.625, 3.3]   1.289   386.7
```

5.0.4 CONCATENATE CNRs IN ONE DATA FRAME

[39]:
```
cnr = pd.concat([cnr1, cnr2, cnr3])
cnr.index=range(cnr.shape[0])
save_df_to_latex(cnr, save_file_root_name='all-subs-cnr')
cnr
```

```
Table saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/tables/05-all-subs-cnr.tex
```

[39]:

	sub	CNR_std	SNR_mean
0	CC1	-4.96	-0.35
1	CC2	9.09	0.24
2	CC3	-13.28	-0.42

---

6 INFO ABOUT IMAGES FOR ALL SUBJECTS (IN PIPELINE)

[40]:
```
_, _ = pipeline('CC1', info=True, show_mask_slices=False)
_, _ = pipeline('CC2', info=True, show_mask_slices=False)
_, _ = pipeline('CC3', info=True, show_mask_slices=False)
```

```
########################
####      CC1       ####
########################

For 3D image(s) printed values take into account background voxels as well:
        *** LOADED IMG ***,     max=988.0, min=0.0, mean=252.81, shape=(320,
320, 25), #voxels=2560000
        *** LOADED ROI ***,     max=1.0, min=0.0, mean=0.00, shape=(320, 320,
25), #voxels=2560000
        *** LOADED BLADDER ***, max=1.0, min=0.0, mean=0.00, shape=(320, 320,
25), #voxels=2560000
        *** PREPARED RIM ***,   max=1.0, min=0.0, mean=0.00, shape=(320, 320,
25), #voxels=2560000
        *** IMAGE WITH VOXELS INSIDE ROI MASK ***,      max=507.0, min=0.0,
mean=0.61, shape=(320, 320, 25), #voxels=2560000
        *** VECTOR WITH VOXELS INSIDE ROI MASK ***,     max=507.0, min=0.0,
mean=168.94, shape=(9180,), #voxels=9180
        *** IMAGE WITH VOXELS INSIDE RIM MASK ***,      max=801.0, min=0.0,
mean=1.16, shape=(320, 320, 25), #voxels=2560000
        *** VECTOR WITH VOXELS INSIDE RIM MASK ***,     max=801.0, min=0.0,
mean=301.37, shape=(9872,), #voxels=9872
```

```
        *** IMAGE WITH VOXELS INSIDE BLADDER MASK ***,  max=462.0, min=0.0,
mean=0.23, shape=(320, 320, 25), #voxels=2560000
        *** VECTOR WITH VOXELS INSIDE BLADDER MASK ***, max=462.0, min=294.0,
mean=378.78, shape=(1572,), #voxels=1572

########################
####      CC2      ####
########################

For 3D image(s) printed values take into account background voxels as well:
        *** LOADED IMG ***,     max=1771.0, min=0.0, mean=336.25, shape=(384,
384, 28), #voxels=4128768
        *** LOADED ROI ***,     max=1.0, min=0.0, mean=0.00, shape=(384, 384,
28), #voxels=4128768
        *** LOADED BLADDER ***, max=1.0, min=0.0, mean=0.00, shape=(384, 384,
28), #voxels=4128768
        *** PREPARED RIM ***,   max=1.0, min=0.0, mean=0.00, shape=(384, 384,
28), #voxels=4128768
        *** IMAGE WITH VOXELS INSIDE ROI MASK ***,      max=613.0, min=0.0,
mean=0.12, shape=(384, 384, 28), #voxels=4128768
        *** VECTOR WITH VOXELS INSIDE ROI MASK ***,     max=613.0, min=184.0,
mean=405.74, shape=(1216,), #voxels=1216
        *** IMAGE WITH VOXELS INSIDE RIM MASK ***,      max=1007.0, min=0.0,
mean=0.15, shape=(384, 384, 28), #voxels=4128768
        *** VECTOR WITH VOXELS INSIDE RIM MASK ***,     max=1007.0, min=92.0,
mean=257.82, shape=(2462,), #voxels=2462
        *** IMAGE WITH VOXELS INSIDE BLADDER MASK ***,  max=673.0, min=0.0,
mean=0.31, shape=(384, 384, 28), #voxels=4128768
        *** VECTOR WITH VOXELS INSIDE BLADDER MASK ***, max=673.0, min=589.0,
mean=628.98, shape=(2064,), #voxels=2064

########################
####      CC3      ####
########################

For 3D image(s) printed values take into account background voxels as well:
        *** LOADED IMG ***,     max=1640.0, min=0.0, mean=443.37, shape=(320,
320, 28), #voxels=2867200
        *** LOADED ROI ***,     max=1.0, min=0.0, mean=0.01, shape=(320, 320,
28), #voxels=2867200
        *** LOADED BLADDER ***, max=1.0, min=0.0, mean=0.00, shape=(320, 320,
28), #voxels=2867200
        *** PREPARED RIM ***,   max=1.0, min=0.0, mean=0.01, shape=(320, 320,
28), #voxels=2867200
        *** IMAGE WITH VOXELS INSIDE ROI MASK ***,      max=939.0, min=0.0,
mean=2.41, shape=(320, 320, 28), #voxels=2867200
        *** VECTOR WITH VOXELS INSIDE ROI MASK ***,     max=939.0, min=15.0,
mean=265.72, shape=(25999,), #voxels=25999
```

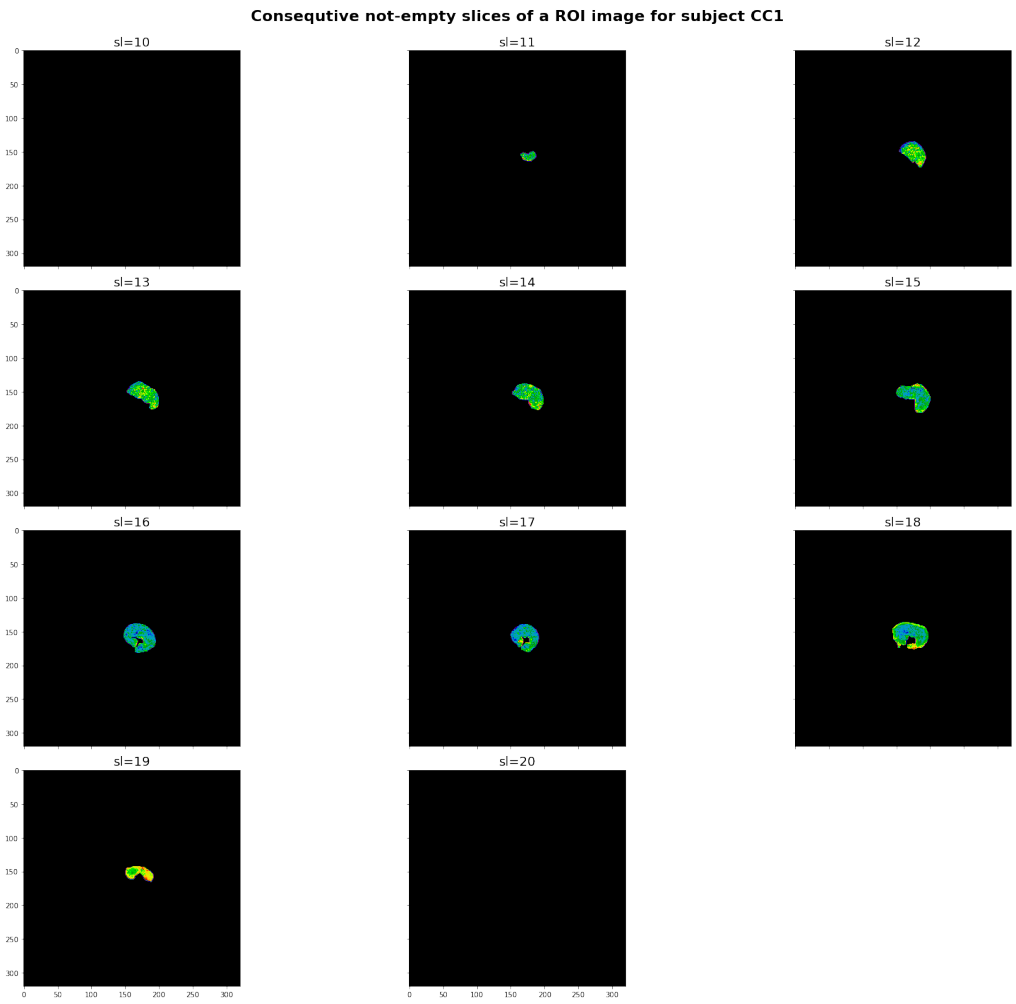
```
        *** IMAGE WITH VOXELS INSIDE RIM MASK ***,      max=1220.0, min=0.0,
mean=2.98, shape=(320, 320, 28), #voxels=2867200
        *** VECTOR WITH VOXELS INSIDE RIM MASK ***,     max=1220.0, min=0.0,
mean=520.81, shape=(16420,), #voxels=16420
        *** IMAGE WITH VOXELS INSIDE BLADDER MASK ***,  max=660.0, min=0.0,
mean=0.06, shape=(320, 320, 28), #voxels=2867200
        *** VECTOR WITH VOXELS INSIDE BLADDER MASK ***, max=660.0, min=544.0,
mean=609.81, shape=(300,), #voxels=300
```

---

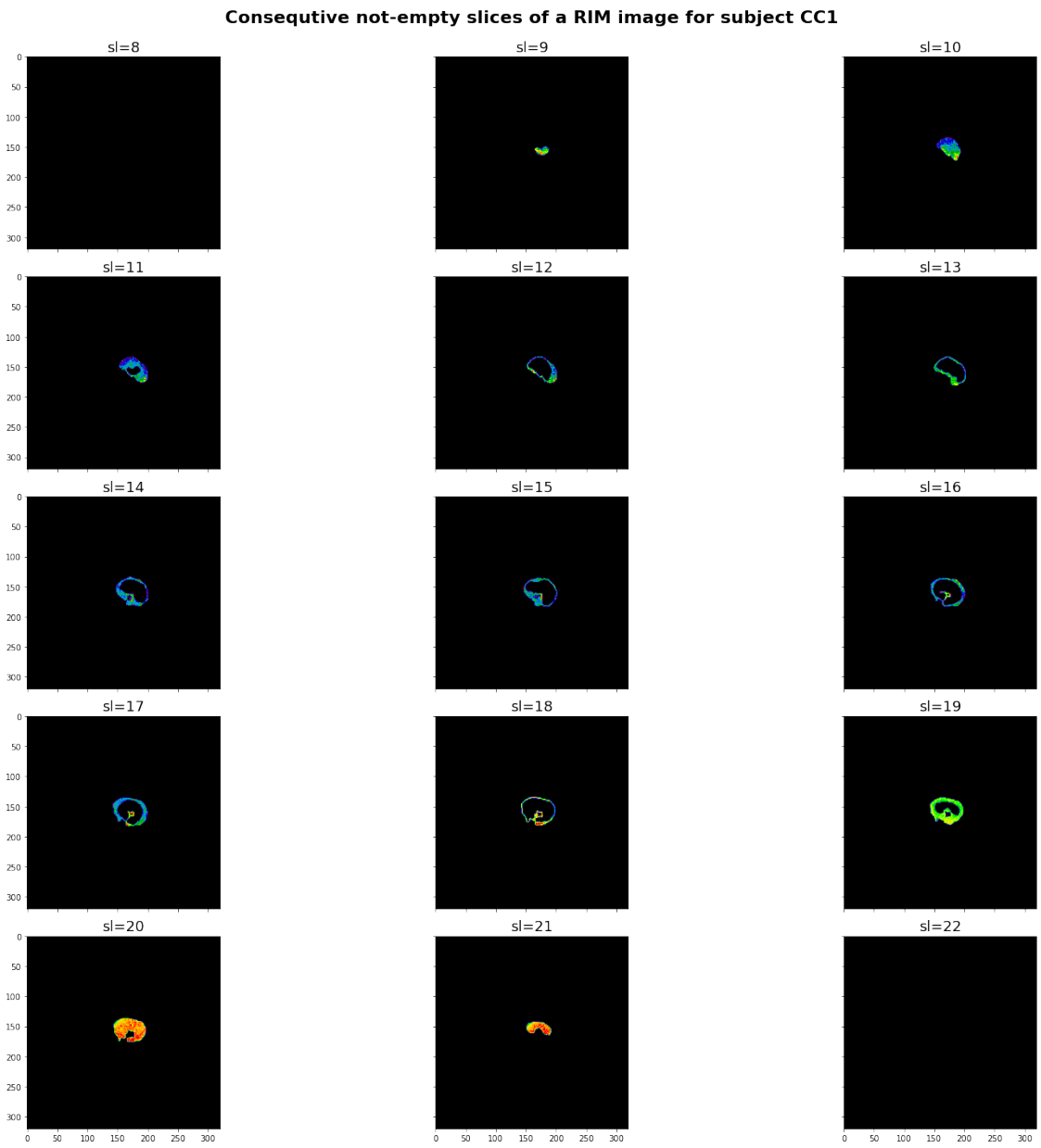
7 SLICE IMAGES FOR ALL SUBJECTS (IN PIPELINE)

[41]:
```
_, _ = pipeline('CC1', info=False, show_mask_slices=True)
_, _ = pipeline('CC2', info=False, show_mask_slices=True)
_, _ = pipeline('CC3', info=False, show_mask_slices=True)
```

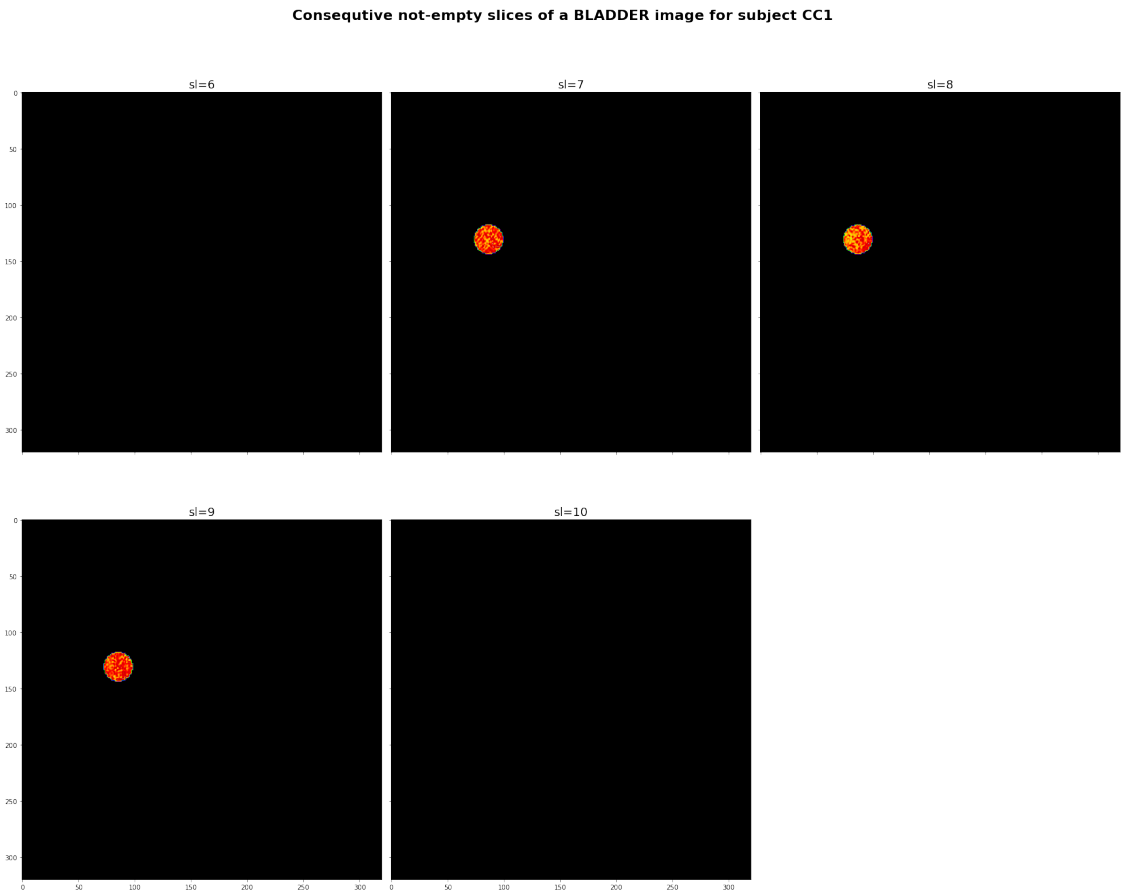
```
Figure saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/plots/05-CC1-roi.png
```



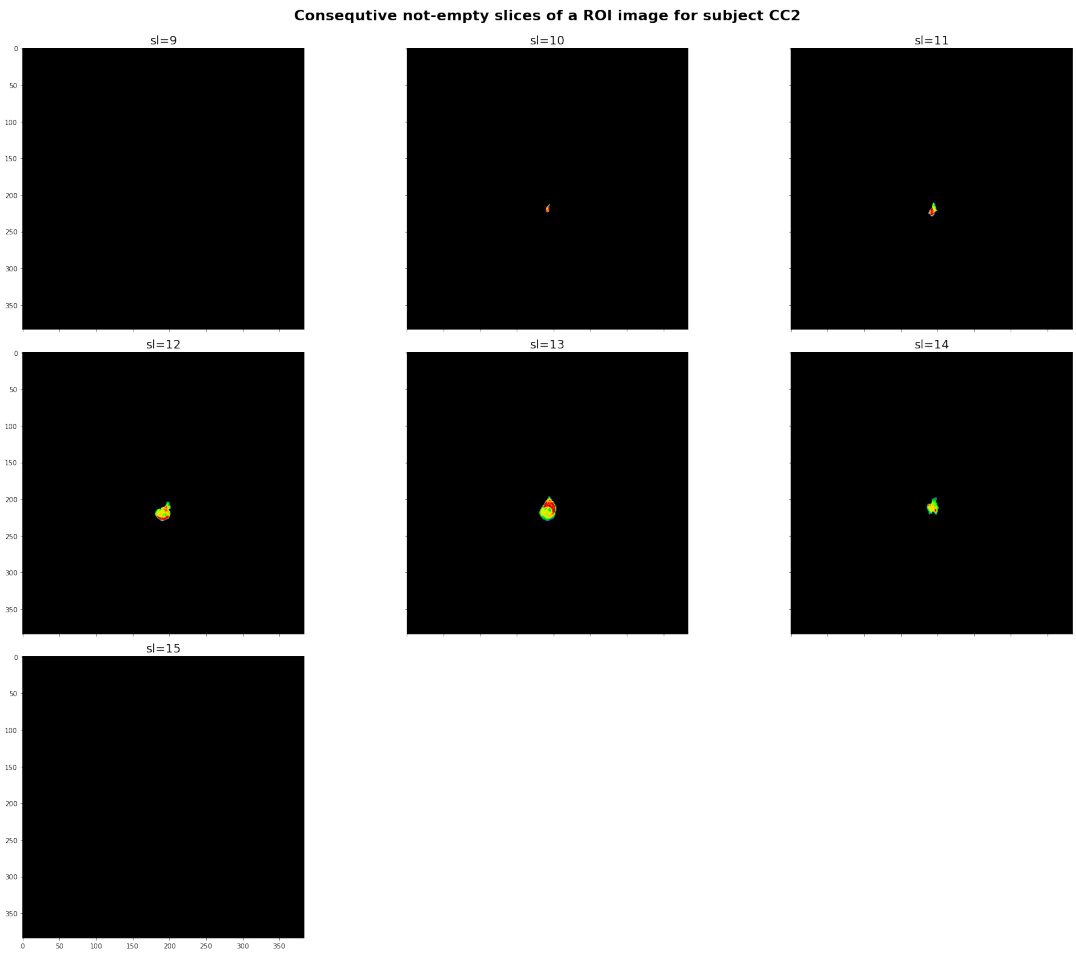

```
Figure saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/plots/05-CC1-rim.png
```



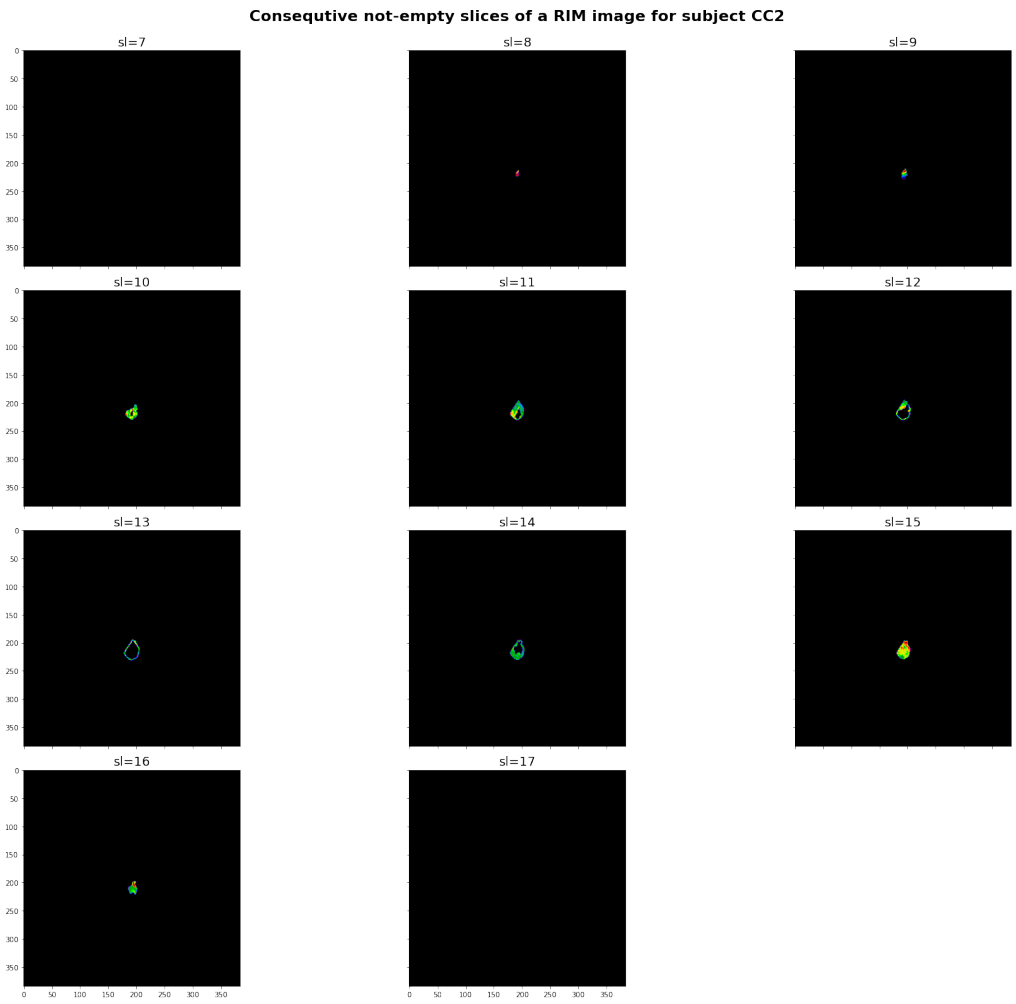

```
Figure saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/plots/05-CC1-bladder.png
```



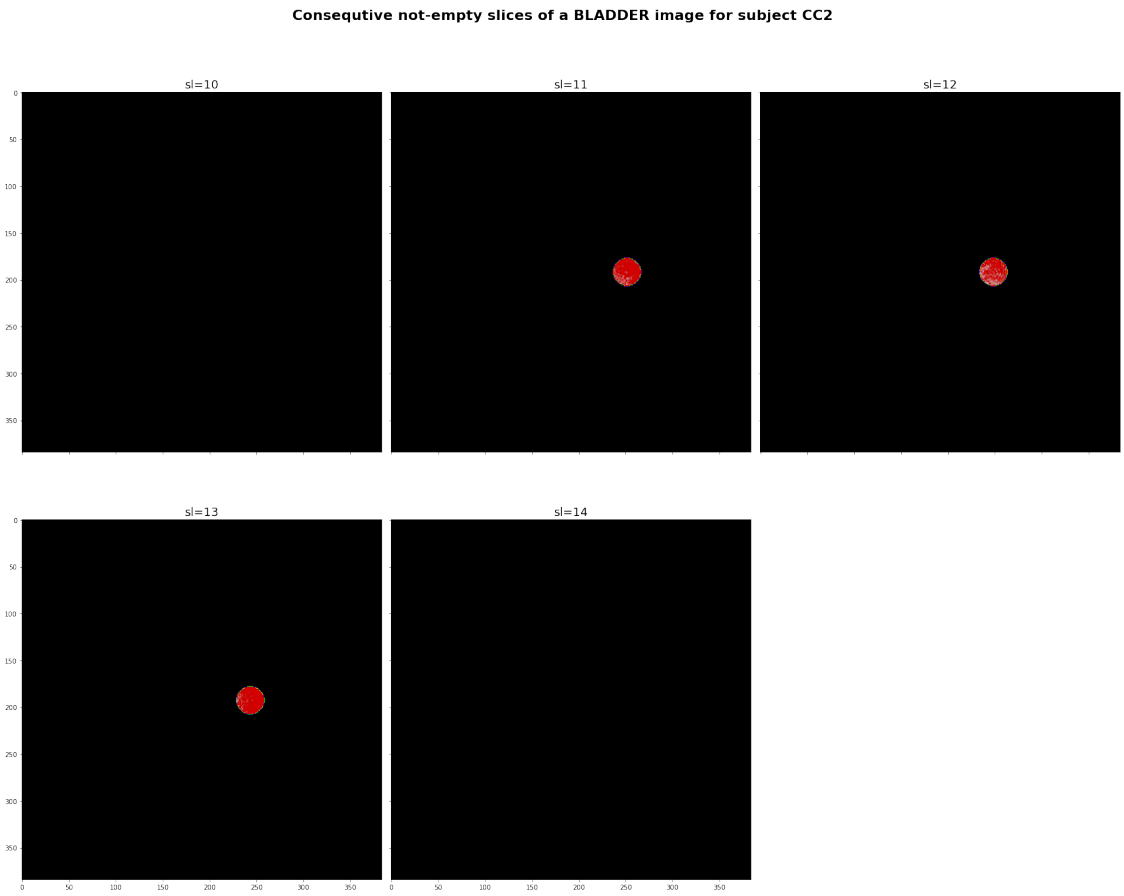

```
Figure saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/plots/05-CC2-roi.png
```



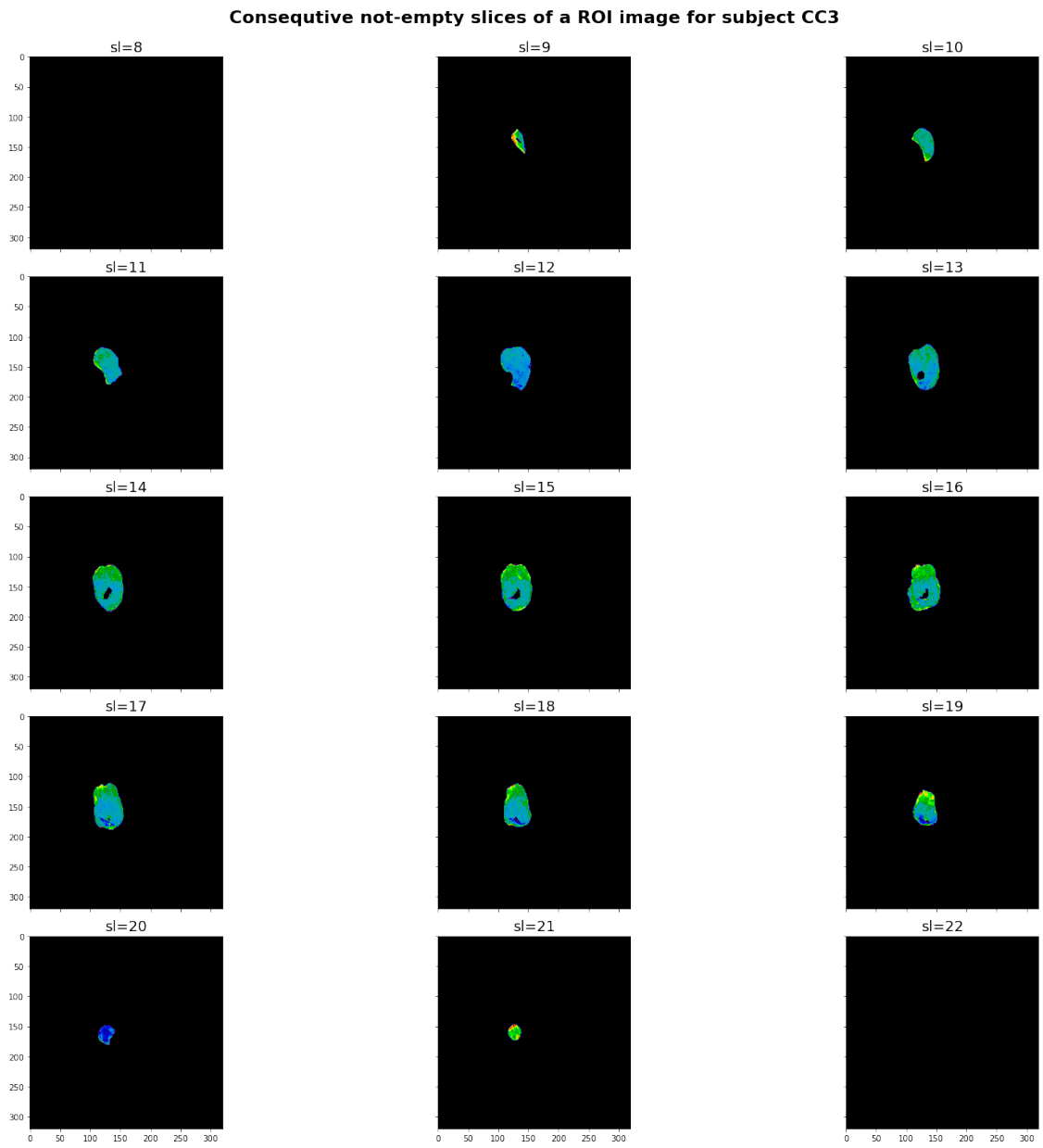

```
Figure saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/plots/05-CC2-rim.png
```



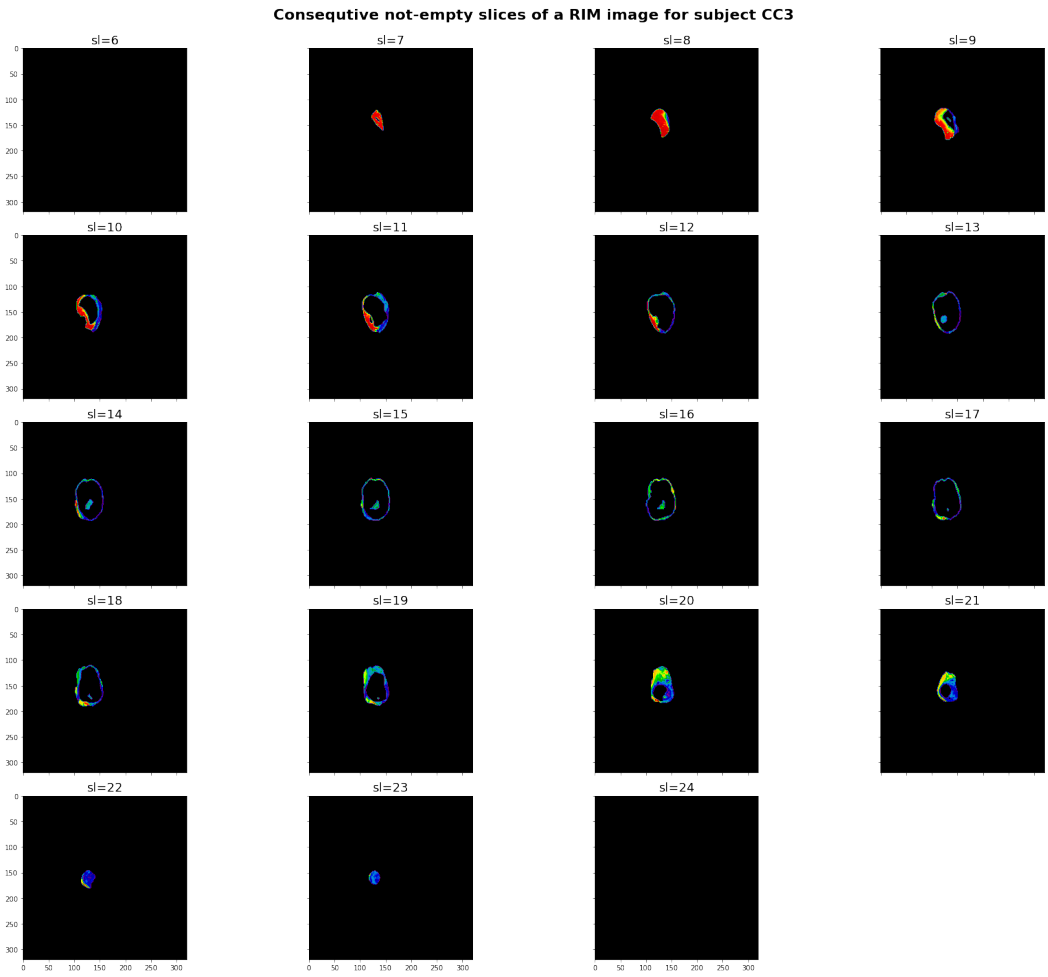

```
Figure saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/plots/05-CC2-bladder.png
```



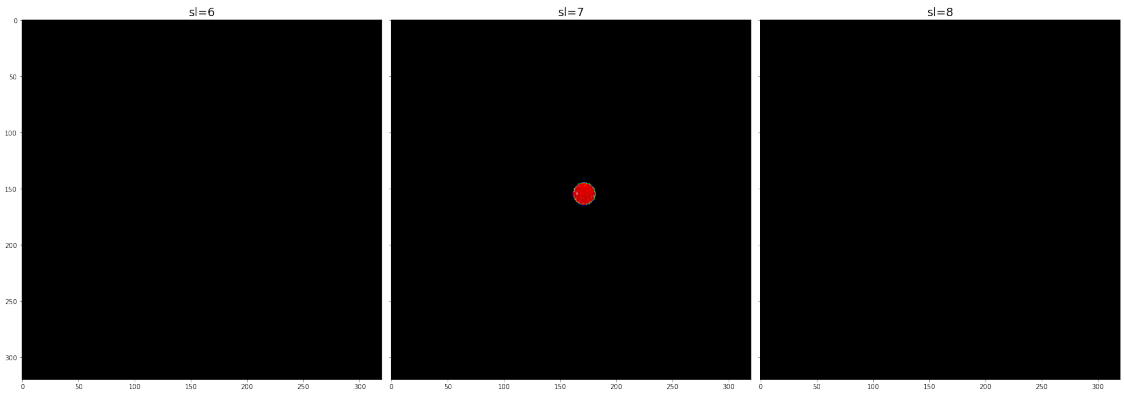

```
Figure saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/plots/05-CC3-roi.png
```




```
Figure saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/plots/05-CC3-rim.png
```




```
Figure saved to:
        /home/marek/Dropbox/p1ext4_P/no_work/TUMOR-CNR/tumor-cnr-
git/data/plots/05-CC3-bladder.png
```



8 CLOSING SETUP

[42]:
```
utils.print_date(5)
```

```
02-Dec-2021 18:34:53
```

[43]:
```
utils.save_notebook_as_html(file_name=CURRENT_NOTEBOOK_NAME, cleaning_delay=2)
```

[ ]:
```
utils.save_notebook_as_pdf(file_name=CURRENT_NOTEBOOK_NAME, cleaning_delay=2)
```