dicompyler-core Documentation

Release 0.5.5

Aditya Panchal

Jun 01, 2019

CONTENTS

Contents:

CHAPTER

ONE

DICOMPYLER-CORE

A library of core radiation therapy modules for DICOM / DICOM RT used by dicompyler. This package includes:

- `dicomparser`: parse DICOM objects in an easy-to-use manner
- `dvh`: Pythonic access to dose volume histogram (DVH) data
- `dvhcalc`: Independent DVH calculation using DICOM RT Dose & RT Structure Set

1.1 Other information

- Free software: BSD license
- Documentation: Read the docs
- Tested on Python 2.7, 3.5, 3.6

1.2 Dependencies

- numpy 1.2 or higher
- pydicom 0.9.9 or higher (pydicom 1.0 compatible)
- matplotlib 1.3.0 or higher (for DVH calculation)
- six 1.5 or higher
- Optional:
  - Pillow (for image display)
  - Shapely (for structure volume calculation)
  - scikit-image (for DVH interpolation)

1.3 Basic Usage

```
from dicompylercore import dicomparser, dvh, dvhcalc
dp = dicomparser.DicomParser("rtss.dcm")

# i.e. Get a dict of structure information
```

(continues on next page)

(continued from previous page)

```
structures = dp.GetStructures()

>>> structures[5]
{'color': array([255, 128, 0]), 'type': 'ORGAN', 'id': 5, 'empty': False, 'name':
↪'Heart'}

# Access DVH data
rtdose = dicomparser.DicomParser("rtdose.dcm")
heartdvh = dvh.DVH.from_dicom_dvh(rtdose.ds, 5)

>>> heartdvh.describe()
Structure: Heart
DVH Type:  cumulative, abs dose: Gy, abs volume: cm3
Volume:    437.46 cm3
Max Dose:  3.10 Gy
Min Dose:  0.02 Gy
Mean Dose: 0.64 Gy
D100:      0.00 Gy
D98:       0.03 Gy
D95:       0.03 Gy
D2cc:      2.93 Gy

# Calculate a DVH from DICOM RT data
calcdvh = dvhcalc.get_dvh("rtss.dcm", "rtdose.dcm", 5)

>>> calcdvh.max, calcdvh.min, calcdvh.D2cc
(3.0899999999999999, 0.029999999999999999, dvh.DVHValue(2.96, 'Gy'))
```

Advanced Usage and Examples can be found in Binder:

1.4 Citing dicompyler-core

A DOI for dicompyler-core with various citation styles can be found at Zenodo:

1.5 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

At the command line:

```
$ pip install dicompyler-core
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv dicompylercore
$ pip install dicompyler-core
```

CHAPTER

THREE

USAGE

To use dicompyler-core in a project:

DICOM data can be easily accessed using convenience functions using the *dicompylercore.dicomparser.DicomParser* class:

```
from dicompylercore import dicomparser, dvh, dvhcalc
dp = dicomparser.DicomParser("rtss.dcm")

# i.e. Get a dict of structure information
structures = dp.GetStructures()

>>> structures[5]
{'color': array([255, 128, 0]), 'type': 'ORGAN', 'id': 5, 'empty': False, 'name':
↪'Heart'}
```

Dose volume histogram (DVH) data can be accessed in a Pythonic manner using the *dicompylercore.dvh.DVH* class:

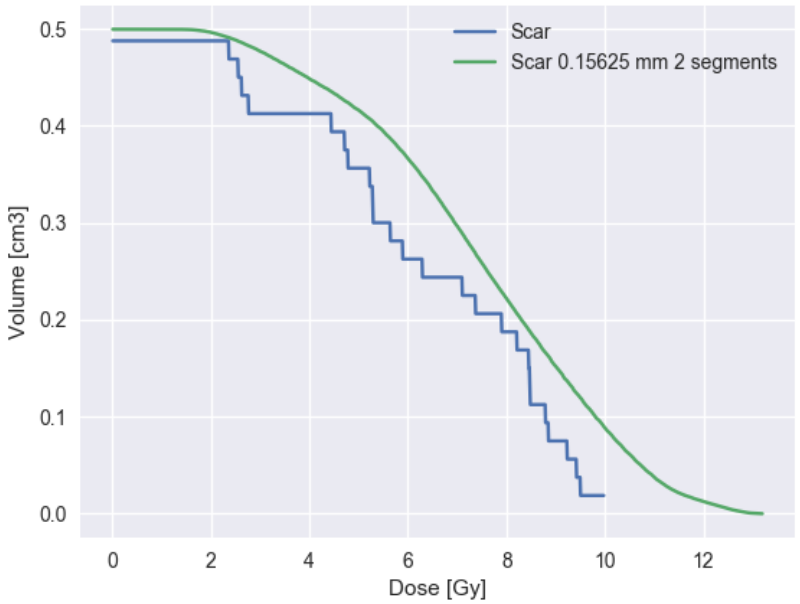
```
rtdose = dicomparser.DicomParser("rtdose.dcm")
heartdvh = dvh.DVH.from_dicom_dvh(rtdose.ds, 5)

>>> heartdvh.describe()
Structure: Heart
-----
DVH Type:  cumulative, abs dose: Gy, abs volume: cm3
Volume:    437.46 cm3
Max Dose:  3.10 Gy
Min Dose:  0.02 Gy
Mean Dose: 0.64 Gy
D100:      0.00 Gy
D98:       0.03 Gy
D95:       0.03 Gy
D2cc:      2.93 Gy

>>> heartdvh.max, heartdvh.min, heartdvh.D2cc
(3.0999999999999779, 0.02, dvh.DVHValue(2.9299999999999815, 'Gy'))
```

Dose volume histograms (DVHs) can be independently calculated using the *dicompylercore.dvhcalc* module:

```
# Calculate a DVH from DICOM RT data
calcdvh = dvhcalc.get_dvh("rtss.dcm", "rtdose.dcm", 5)

>>> calcdvh.max, calcdvh.min, calcdvh.D2cc
(3.0899999999999999, 0.029999999999999999, dvh.DVHValue(2.96, 'Gy'))
```

Structure and dose data for independently calculated DVHs can also be interpolated in-plane and between planes:

```
# Calculate a DVH using interpolation to super-sample the dose grid in plane,
# interpolate dose between planes and restrict calculation to the structure
# extents
interpscar = dvhcalc.get_dvh("rtss.dcm", "rtdose.dcm", 8,
    interpolation_resolution=(2.5/16),
    interpolation_segments_between_planes=2,
    use_structure_extents=True)
interpscar.name += ' interp'

# Compare against un-interpolated DVH
origscar = dvhcalc.get_dvh("rtss.dcm", "rtdose.dcm", 8)

>>> origscar.compare(interpscar)
Structure:          Scar            Scar interp       Rel Diff       Abs diff
-----
DVH Type:  cumulative, abs dose: Gy, abs volume: cm3
Volume:             0.47 cm3          0.50 cm3         +6.55 %          +0.03
Max:                9.50 Gy          13.18 Gy         +38.74 %          +3.68
Min:                2.36 Gy           1.23 Gy         -47.88 %          -1.13
Mean:               6.38 Gy           7.53 Gy         +18.02 %          +1.15
D100:               0.00 Gy           0.00 Gy          +0.00 %          +0.00
D98:                2.36 Gy           2.44 Gy          +3.39 %          +0.08
D95:                2.36 Gy           3.09 Gy         +30.93 %          +0.73
D2cc:               0.00 Gy           0.00 Gy          +0.00 %          +0.00
```



CHAPTER

FOUR

DICOMPYLERCORE

4.1 dicomparser module

Class that parses and returns formatted DICOM RT data.

class dicompylercore.dicomparser.**DicomParser**(*dataset*)
Bases: object

Parses DICOM / DICOM RT files.

CalculatePlaneThickness(*planesDict*)
Calculates the plane thickness for each structure.

CalculateStructureVolume(*coords*, *thickness*)
Calculates the volume of the given structure.

Parameters

- **coords** (*dict*) – Coordinates of each plane of the structure
- **thickness** (*float*) – Thickness of the structure

GetContourPoints(*array*)
Parses an array of xyz points & returns a array of point dicts.

GetDVHs()
Returns cumulative dose-volume histograms (DVHs).

GetDefaultImageWindowLevel()
Determine the default window/level for the DICOM image.

GetDemographics()
Return the patient demographics from a DICOM file.

GetDoseData()
Return the dose data from a DICOM RT Dose file.

GetDoseGrid(*z=0*, *threshold=0.5*)
Return the 2d dose grid for the given slice position (mm).

Parameters

- **z** – Slice position in mm.
- **threshold** – Threshold in mm to determine the max difference from z to the closest dose slice without using interpolation.

Returns An numpy 2d array of dose points.

GetFrameOfReferenceUID()
Determine the Frame of Reference UID of the current file.

GetImage(*window=0*, *level=0*, *size=None*, *background=False*, *frames=0*)
Return the image from a DICOM image storage file.

GetImageData()
Return the image data from a DICOM file.

GetImageLocation()
Calculate the location of the current image slice.

GetImageOrientationType()
Get the orientation of the current image slice.

GetIsodosePoints(*z=0*, *level=100*, *threshold=0.5*)

Return points for the given isodose level and slice position from the dose grid.

Parameters

- **z** – Slice position in mm.
- **threshold** – Threshold in mm to determine the max difference from z to the closest dose slice without using interpolation.
- **level** – Isodose level in scaled form (multiplied by self.ds.DoseGridScaling)

Returns An array of tuples representing isodose points.

GetLUTValue(*data*, *window*, *level*)
Apply the RGB Look-Up Table for the data and window/level value.

GetNumberOfFrames()
Return the number of frames in a DICOM image file.

GetPatientToPixelLUT()
Get the image transformation matrix from the DICOM standard Part 3 Section C.7.6.2.1.1

GetPlan()
Returns the plan information.

GetReferencedBeamNumber()
Return the referenced beam number (if it exists) from RT Dose.

GetReferencedBeamsInFraction(*fx=0*)
Return the referenced beams from the specified fraction.

GetReferencedRTPlan()
Return the SOP Class UID of the referenced RT plan.

GetReferencedSeries()
Return the SOP Class UID of the referenced series.

GetReferencedStructureSet()
Return the SOP Class UID of the referenced structure set.

GetRescaleInterceptSlope()
Return the rescale intercept and slope if present.

GetSOPClassUID()
Determine the SOP Class UID of the current file.

GetSOPInstanceUID()
Determine the SOP Class UID of the current file.

GetSeriesDateTime()
Return the series date/time information.

GetSeriesInfo()
Return the series information of the current file.

GetStructureCoordinates(*roi_number*)
Get the list of coordinates for each plane of the structure.

GetStructureInfo()
Return the patient demographics from a DICOM file.

GetStructures()
Returns a dictionary of structures (ROIs).

GetStudyInfo()
Return the study information of the current file.

HasDVHs()
Returns whether dose-volume histograms (DVHs) exist.

InterpolateDosePlanes(*uplane*, *lplane*, *fz*)

Interpolates a dose plane between two bounding planes at the given relative location.

Parameters

- **uplane** – Upper dose plane boundary.
- **uplane** – Lower dose plane boundary.
- **fz** – Fractional distance from the bottom to the top, where the new plane is located. E.g. if fz = 1, the plane is at the upper plane, fz = 0, it is at the lower plane.

Returns An numpy 2d array of the interpolated dose plane.

4.2 dvh module

Class that stores dose volume histogram (DVH) data.

class dicompylercore.dvh.**DVH**(*counts*, *bins*, *dvh_type='cumulative'*, *dose_units='Gy'*, *volume_units='cm3'*, *rx_dose=None*, *name=None*, *color=None*, *notes=None*)

Bases: object

Class that stores dose volume histogram (DVH) data.

absolute_dose(*rx_dose=None*, *dose_units='Gy'*)
Return an absolute dose DVH.

Parameters

- **rx_dose** (*number, optional*) – Prescription dose value used to normalize dose bins
- **dose_units** (*str, optional*) – Units for the absolute dose

Raises **AttributeError** – Description

absolute_volume(*volume*, *volume_units='cm3'*)
Return an absolute volume DVH.

Parameters

- **volume** (*number*) – Absolute volume of the structure
- **volume_units** (*str, optional*) – Units for the absolute volume

bincenters
Return a numpy array containing the bin centers.

compare(*dvh*)
Compare the DVH properties with another DVH.

Parameters **dvh** (DVH) – DVH instance to compare against

Raises **AttributeError** – If DVHs do not have equivalent dose & volume units

cumulative
Return a cumulative DVH from a differential DVH.

describe()
Describe a summary of DVH statistics in a text based format.

differential
Return a differential DVH from a cumulative DVH.

dose_constraint(*volume*, *volume_units=None*)
Calculate the maximum dose that a specific volume receives.

i.e. D90, D100 or D2cc

Parameters **volume** (*number*) – Volume used to determine the maximum dose that the volume receives. Can either be in relative or absolute volume units.

Returns Dose in self.dose_units units.

Return type number

classmethod from_data(*data*, *binsize=1*)
Initialization for a DVH from raw data.

Parameters

- **data** (*iterable or numpy array*) – An iterable of dose data that is used to create the histogram
- **binsize** (*int, optional*) – Bin width size (in cGy used to create the histogram)

classmethod from_dicom_dvh(*dataset*, *roi_num*, *rx_dose=None*, *name=None*, *color=None*)
Initialization for a DVH from a pydicom RT Dose DVH sequence.

max
Return the maximum dose.

mean
Return the mean dose.

min
Return the minimum dose.

plot()
Plot the DVH using Matplotlib if present.

relative_dose(*rx_dose=None*)
Return a relative dose DVH based on a prescription dose.

Parameters **rx_dose** (*number, optional*) – Prescription dose value used to normalize dose bins

Raises **AttributeError** – Raised if prescription dose was not present either during class initialization or passed via argument.

relative_volume
Return a relative volume DVH.

statistic(*name*)
Return a DVH dose or volume statistic.

Parameters **name** (*str*) – DVH statistic in the form of D90, D100, D2cc, V100 or V20Gy, etc.

Returns Value from the dose or volume statistic calculation.

Return type number

volume
Return the volume of the structure.

volume_constraint(*dose*, *dose_units=None*)
Calculate the volume that receives at least a specific dose.

i.e. V100, V150 or V20Gy

Parameters **dose** (*number*) – Dose value used to determine minimum volume that receives this dose. Can either be in relative or absolute dose units.

Returns Volume in self.volume_units units.

Return type number

class dicompylercore.dvh.**DVHValue**(*value*, *units=''*)
Bases: object

Class that stores DVH values with the appropriate units.

4.3 dvhcalc module

Calculate dose volume histogram (DVH) from DICOM RT Structure/Dose data.

dicompylercore.dvhcalc.**calculate_contour_dvh**(*mask*, *doseplane*, *maxdose*, *dd*, *id*, *structure*)
Calculate the differential DVH for the given contour and dose plane.

dicompylercore.dvhcalc.**calculate_dvh**(*structure*, *dose*, *limit=None*, *calculate_full_volume=True*, *use_structure_extents=False*, *interpolation_resolution=None*, *interpolation_segments_between_planes=0*, *callback=None*)
Calculate the differential DVH for the given structure and dose grid.

Parameters

- **structure** (*dict*) – A structure (ROI) from an RT Structure Set parsed using DicomParser
- **dose** (DicomParser) – A DicomParser instance of an RT Dose
- **limit** (*int, optional*) – Dose limit in cGy as a maximum bin for the histogram.
- **calculate_full_volume** (*bool, optional*) – Calculate the full structure volume including contours outside of the dose grid.

- **use_structure_extents** (*bool, optional*) – Limit the DVH calculation to the in-plane structure boundaries.
- **interpolation_resolution** (*float, optional*) – Resolution in mm to interpolate the structure and dose data to.
- **interpolation_segments_between_planes** (*integer, optional*) – Number of segments to interpolate between structure slices.
- **callback** (*function, optional*) – A function that will be called at every iteration of the calculation.

dicompylercore.dvhcalc.**calculate_plane_histogram**(*plane*, *doseplane*, *dosegridpoints*, *maxdose*, *dd*, *id*, *structure*, *hist*)

Calculate the DVH for the given plane in the structure.

dicompylercore.dvhcalc.**dosegrid_extents_indices**(*extents*, *dd*, *padding=1*)

Determine dose grid extents from structure extents as array indices.

Parameters

- **extents** (*list*) – Structure extents in patient coordintes: [xmin, ymin, xmax, ymax]. If an empty list, no structure extents will be used in the calculation.
- **dd** (*dict*) – Dose data from dicomparser.GetDoseData.
- **padding** (*int, optional*) – Pixel padding around the structure extents.

Returns Dose grid extents in pixel coordintes as array indices: [xmin, ymin, xmax, ymax].

Return type list

dicompylercore.dvhcalc.**dosegrid_extents_positions**(*extents*, *dd*)

Determine dose grid extents in patient coordinate indices.

Parameters

- **extents** (*list*) – Dose grid extents in pixel coordintes: [xmin, ymin, xmax, ymax].
- **dd** (*dict*) – Dose data from dicomparser.GetDoseData.

Returns Dose grid extents in patient coordintes: [xmin, ymin, xmax, ymax].

Return type list

dicompylercore.dvhcalc.**get_contour_mask**(*dd*, *id*, *dosegridpoints*, *contour*)

Get the mask for the contour with respect to the dose plane.

dicompylercore.dvhcalc.**get_dvh**(*structure*, *dose*, *roi*, *limit=None*, *calculate_full_volume=True*, *use_structure_extents=False*, *interpolation_resolution=None*, *interpolation_segments_between_planes=0*, *thickness=None*, *callback=None*)

Calculate a cumulative DVH in Gy from a DICOM RT Structure Set & Dose.

Parameters

- **structure** (*pydicom Dataset*) – DICOM RT Structure Set used to determine the structure data.
- **dose** (*pydicom Dataset*) – DICOM RT Dose used to determine the dose grid.
- **roi** (*int*) – The ROI number used to uniquely identify the structure in the structure set.
- **limit** (*int, optional*) – Dose limit in cGy as a maximum bin for the histogram.
- **calculate_full_volume** (*bool, optional*) – Calculate the full structure volume including contours outside of the dose grid.

- **use_structure_extents** (*bool, optional*) – Limit the DVH calculation to the in-plane structure boundaries.
- **interpolation_resolution** (*float, optional*) – Resolution in mm to interpolate the structure and dose data to.
- **interpolation_segments_between_planes** (*integer, optional*) – Number of segments to interpolate between structure slices.
- **thickness** (*float, optional*) – Structure thickness used to calculate volume of a voxel.
- **callback** (*function, optional*) – A function that will be called at every iteration of the calculation.

dicompylercore.dvhcalc.**get_interpolated_dose**(*dose*, *z*, *resolution*, *extents*)

Get interpolated dose for the given z, resolution & array extents.

Parameters

- **dose** (DicomParser) – A DicomParser instance of an RT Dose.
- **z** (*float*) – Index in mm of z plane of dose grid.dose
- **resolution** (*float*) – Interpolation resolution less than or equal to dose grid pixel spacing.
- **extents** (*list*) – Dose grid index extents.

Returns Interpolated dose grid with a shape larger than the input dose grid.

Return type ndarray

dicompylercore.dvhcalc.**get_resampled_lut**(*index_extents*, *extents*, *new_pixel_spacing*, *min_pixel_spacing*)

Determine the patient to pixel LUT based on new pixel spacing.

Parameters

- **index_extents** (*list*) – Dose grid extents as array indices.
- **extents** (*list*) – Dose grid extents in patient coordinates.
- **new_pixel_spacing** (*float*) – New pixel spacing in mm
- **min_pixel_spacing** (*float*) – Minimum pixel spacing used to determine the new pixel spacing

Returns A tuple of lists (x, y) of patient to pixel coordinate mappings.

Return type tuple

Raises **AttributeError** – Raised if the new pixel_spacing is not a factor of the minimum pixel spacing.

Notes

The new pixel spacing must be a factor of the original (minimum) pixel spacing. For example if the original pixel spacing was `3` mm, the new pixel spacing should be: `3 / (2^n)` mm, where `n` is an integer.

Examples

Original pixel spacing: `3` mm, new pixel spacing: `0.375` mm Derived via: `(3 / 2^16) == 0.375`

dicompylercore.dvhcalc.**interpolate_between_planes**(*planes*, *n=2*)

Interpolate n additional structure planes (segments) in between planes.

Parameters

- **planes** (*dict*) – RT Structure plane data from dicomparser.GetStructureCoordinates.
- **n** (*int, optional*) – Number of planes to interpolate in between the existing planes.

Returns Plane data with additional keys representing interpolated planes.

Return type dict

dicompylercore.dvhcalc.**structure_extents**(*coords*)

Determine structure extents in patient coordinates.

Parameters **coords** (*dict*) – Structure coordinates from dicomparser.GetStructureCoordinates.

Returns Structure extents in patient coordintes: [xmin, ymin, xmax, ymax].

Return type list

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/dicompyler/dicompyler-core/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

5.1.4 Write Documentation

dicompyler-core could always use more documentation, whether as part of the official dicompyler-core docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/dicompyler/dicompyler-core/issues.

If you are proposing a feature:

- Explain in detail how it would work.

- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *dicompyler-core* for local development.

1. Fork the *dicompyler-core* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/dicompyler-core.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv dicompyler-core
$ cd dicompyler-core/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 dicompyler-core tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.5, and 3.6. Check https://travis-ci.org/dicompyler/dicompyler-core/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_dicompyler-core
```

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Aditya Panchal

6.2 Contributors

- Roy Keyes

CHAPTER

SEVEN

HISTORY

7.1 0.5.5 (2019-05-31)

7.1.1 dvhcalc

- Refactored bounding & resampling set up code to only execute if conditions are met.
- Fix a bug where the resampled LUT was not calculated correctly for DVH interpolation.

7.1.2 dvh

- Differential DVH calculation modified. (#60) [Hideki Nakamoto]
- Fix an issue with D100 not returning 0 Gy. (#74) [Gabriel Couture]
- Preserve global maximum dose. (#106) [Akihisa Wakita]

7.1.3 dicomparser

- Remove the test for existence of *ContourImageSequence* in *GetStructureCoordinates*. (#81) [Gabriel Couture]
- Utilize integer division when generating a background for an image.
- Return a string for the patient's name as *PersonName3* cannot be serialized.
- Fix a bug to return a referenced FoR if the FrameOfReference is blank.
- Fix a bug in *GetPlan* where the wrong object names were used. (#43) [gertsikkema]
- Ensure that Rx Dose from RT Plan is rounded instead of truncated.
- Account for holes and bifurcated structures for structure volume calculation.
- Implement structure volume calculation using Shapely.
- Fix window calculation if not present in header.
- Add checks in max, mean, min and dose_constraint for case where counts array is empty or all 0's. (#96) [Nicolas Galler]

7.2 0.5.4 (2018-04-02)

7.2.1 dvhcalc

- Implemented DVH interpolation. (#39)
- Implemented optional user-specified structure thickness for DVH calculation.

7.2.2 dvh

- Fix a bug in absolute_volume if a DVH instance's volume units don't use default of Gy.
- Fix a bug in absolute_dose if a DVH instance's dose units don't use default of Gy. (#19)
- Support decimal values for volume constraints (i.e.V71.6).
- Support decimal values for dose constraints (i.e. D0.03cc).

7.2.3 dicomparser

- Ensure that Rx Dose from RT Plan is rounded instead of truncated.
- Account for holes and bifurcated structures for structure volume calculation.
- Implement structure volume calculation using Shapely. (#28)

7.3 0.5.3 (2017-08-03)

- Added support for plotting structure colors.
- Support Python 2 unicode filenames in dicomparser.
- Support DVH calculation of structures partially covered by the dose grid.

7.4 0.5.2 (2016-07-25)

- Added `DVH` class for Pythonic access to dose volume histogram data.
- Refactored and added unit tests for dvhcalc.
- Added examples and usage for `dvh` and `dvhcalc` modules.
- Jupyter notebook of examples can be found in Binder:

7.5 0.5.1 (2016-02-17)

- Added support for pydicom 0.9.9 so releases from PyPI can be built.

7.6 0.5.0 (2016-02-11)

- First release on PyPI.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

INDEX

H

I

M

P

R

S

V