User interface to extract radio-morphologic features for refined dose-toxicity analysis in radiotherapy

Alaleh Azhir, William Franceshchi, Santiago Appiani
Mentors: Dr. Todd McNutt and Pranav Lakshminarayanan

May 10, 2018

Computer Integrated Surgery II

Dr. Russell Taylor and Ehsan Azizi

Spring 2018

1. Abstract

Radiation therapy for neck and head tumors has been very effective, but usually leads to undesired outcomes such as speech impairment. Previous work has indicated that some regions are more sensitive to radiation exposure, thus analyzing radiotherapy dose distribution data in patients and their associated outcomes could assist in developing better treatment plans. The current tools available to oncologists and radiologists for visualization and analysis of radiotherapy dose-distribution data are limited in functionality and ease of use, thus, we developed a user interface with 3D rendering allows physicians to better analyze dose distributions before treatment and make necessary adjustments to the treatment plan.

2. Introduction

Radiation therapies, sometimes accompanied with surgery or chemotherapy, provides a very effective way to treat many malignant head and neck tumors [1]. However, they are usually accompanied with undesired complications such as decreased saliva production, speech impairment, and soft tissue necrosis, thus reducing patients' quality of life. Research has shown that certain regions of the brain are more sensitive than others to radiation exposure [2], thus to create better outcomes it is best to analyze radiation therapy treatments and reduce radiation to those regions. Although targeted therapy has been quite successful in minimizing radiation to areas outside of tumor, this method still to some degree radiates undesired regions based on the morphology of the tumor. The best treatment plan for patients would be one that maximizes radiation dose to the tumor and minimizes it to the critical regions that are very susceptible to damage by radiation. In order to study various treatment plans and compare them to each other, it is necessary to compare the region that these treatment plans radiated, the dose distribution across this region, and the patient outcome.

Luijik, et al. has shown that exposure to radiation in ductal region of parotid gland is significantly associated with reduced saliva production after the treatment [3]. This result can be confirmed through parotid dose toxicity analysis. Dose toxicity analysis since the 1970s has predominantly been reliant upon dose value histogram (DVH) curves. Applying traditional whole organ-based analyses to head and neck structures is difficult and inconsistent because these structures are small and easily misidentified or overlooked in analysis. Research has shown that a more localized, voxel-based approach would provide clinical insights in radiation therapy of the bladder, prostate, and gastrointestinal tract [4-7]. For example, Monti et al showed that a voxel-based approach to dose toxicity analysis can shed light upon the mechanisms underlying a specific complication following radiation therapy: radiation-induced acute dysphagia (RIAD) [7].

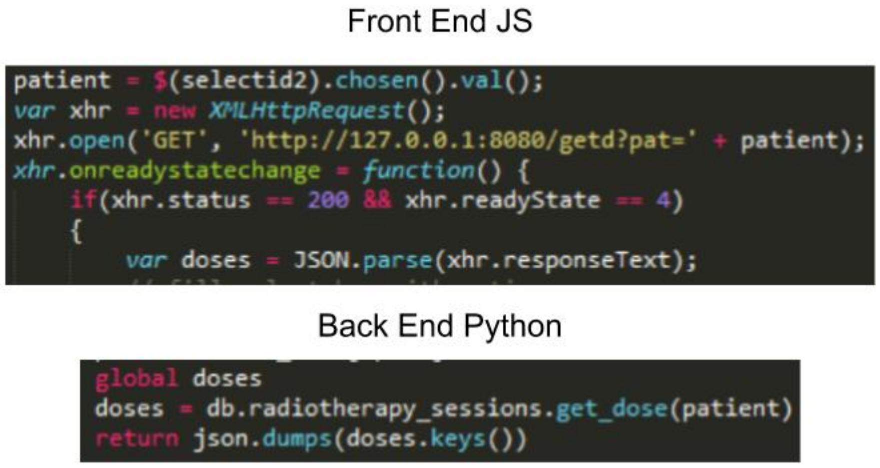
The aim of our tool is thus to provide a way for physicians who lack programming skills a simple method to quantitatively analyze and visualize dose distributions within organs before, during and after treatment. It provides a user interface with 3D rendering would allow physicians to better analyze dose distributions by allowing segmentation into smaller regions to allow for finer detailed analysis. This tool can help simplify the workflow for physicians who need to compare treatments by allowing them visualize and segment various treatment plans organs and their associated DVH curves.

3. Methods

3.1. Back-End Web Development

The server side was written entirely in python. Our mentor, Pranav, provided code for 1) making SQL queries from the Oncotools database, 2) computing DVH curves and point clouds, and 3) segmentation tools. The backend work consisted of constructing a python web framework for passing data via XHR (XMLHttpRequest) requests with the web application, integrating Pranav's code into the python web framework to perform appropriate calculations, and making the data compatible with JSON format and easily integrable with the web application.

Runweb.py is the python file which, when run at the command line, sets up the web framework. The python web framework consists of a list of URLs, with each URL corresponding to a class. The web framework interacts with the web application in a similar manner as calling a function. When the web app makes an XHR request to one of the URLs in the web framework, the python code in the corresponding class runs and returns JSON formatted data, as shown in Figure 1 - XHR Request. Generally, the web-based application sends necessary information to the python framework via XHR request to a specific URL. The python framework will run the code in the class specified by the URL and return a JSON object holding DVH or point cloud data to the web-based application. Specific details for the classes and URLs in the web framework can be found in the Code Documentation section of the Documentation.



```
patient = $(selectid2).chosen().val();
var xhr = new XMLHttpRequest();
xhr.open('GET', 'http://127.0.0.1:8080/getd?pat=' + patient);
xhr.onreadystatechange = function() {
    if(xhr.status == 200 && xhr.readyState == 4)
    {
        var doses = JSON.parse(xhr.responseText);
```

```
global doses
doses = db.radiotherapy_sessions.get_dose(patient)
return json.dumps(doses.keys())
```

Figure 1 - XHR Requests. Example XHR request showing JavaScript front end and python back end. Web app front end makes an XHR request to a URL. This prompts the python web framework to run code in a class corresponding to the URL and return a JSON object. The web app can parse this JSON object.

3.2 Front-End Web Development:

The front end was written using the three languages of HTML, CSS, and JavaScript. We utilized various JavaScript Libraries such as Select.js and D3.js (versions 3 and 4). The front end communicates with the back end in order to a) run analyses such as segmentation, b) send the selection of user in order to fetch relevant data, c) creating DVH curves and d) getting the coordinates for the 3D organ. The communication between the front-end and back-end can be seen in Figure 2.

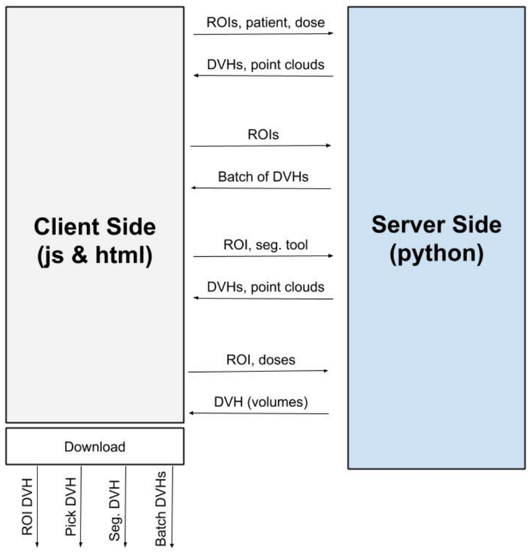


Figure 2 - Code Communication. Client side and server side communication. The data and download options are shown here. Server side, written in python, returns DVH and point cloud data when given necessary patient and dose information.

First step for the user utilizing our tool is to select region(s) of interest, for the select boxes, we used the Select.js library to allow for simultaneous multiple selection and allow for querying choices by typing the name of the region instead of scrolling. After the first selection, the next select boxes are updated based on the previous choices, showing only relevant options (i.e. displaying only patients affected by a certain region of interest.

The second part is the visualization of a 3D organ. For this, we utilize D3.js version 4, as it allows for better dragging and visualization options. An example of the two parotid glands visualized can be found in Figure 3. We also graph the x, y, z axes, whose size are determined by the data points that are visualized, thus, their size automatically adjust as the organ is changed. Furthermore, the visualization of x, y, z axes allows for the user to better track by what angle and by how much, they are rotating the organs when they click and drag their mouse on the interface.

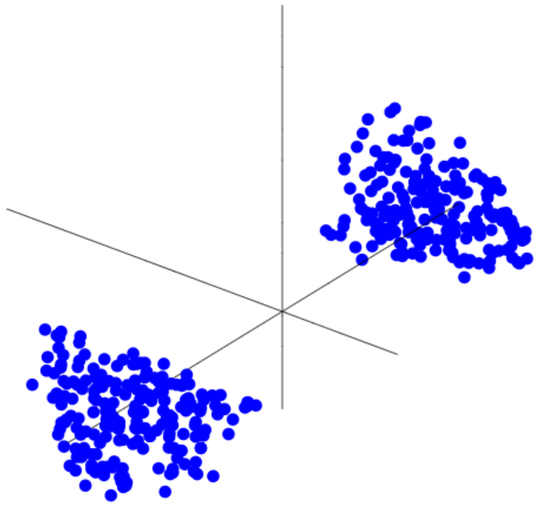


Figure 3 – 3D Region of Interest visualization.

The second part is the visualization of the DVH curve. After the selections are made, this data is sent to the backend, and a DVH curve is calculated using the python back-end layer. The scatter plot x and y coordinates are then sent to the front-end layer for visualization. For this, we utilize D3.js version 3, as the library for a scatter plot was more readily available. The fact that we were utilizing 2 versions of D3.js on one HTML page caused a clashing of two libraries, causing one of the graphs (3D or DVH) to break. To address this issue, we created two different HTML pages, each of which uses its own version of D3.js library and used HTML iframe tag to view both HTML pages from the main HTML page. An example of the two parotid glands DVH curves visualization can be found in Figure 4.

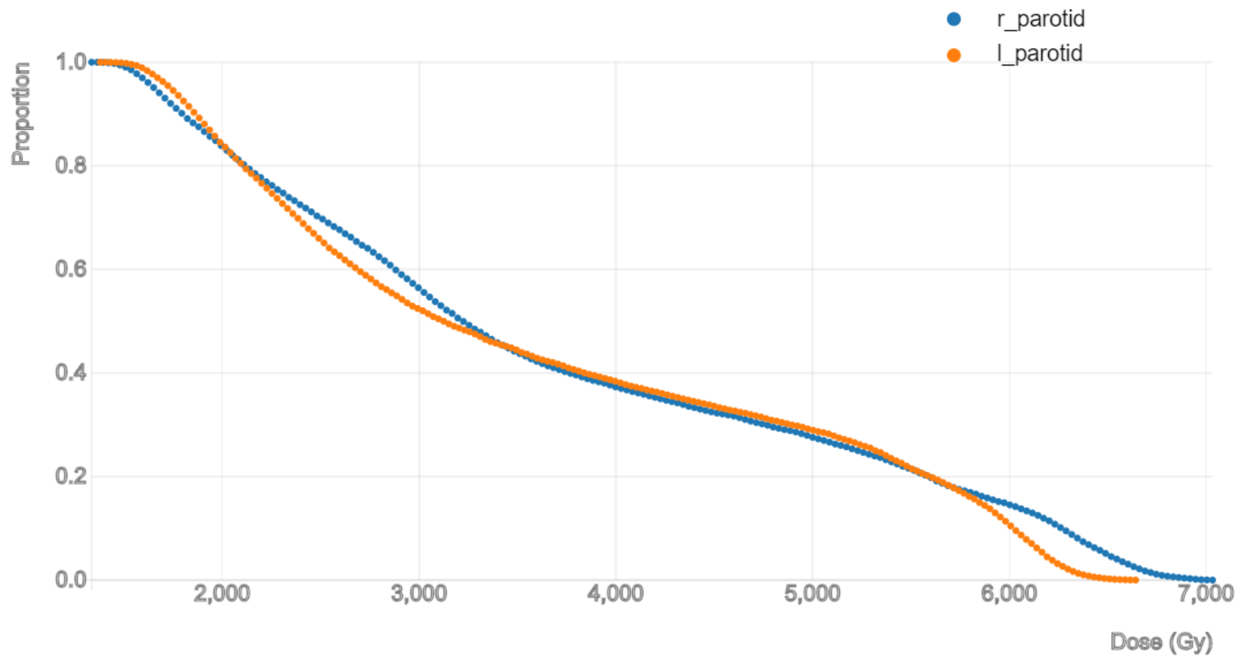


Figure 4 – DVH Curve. This graph is made for left and right parotid.

Front-end also allows the option for the user to download the DVH plot they created in case they need it for further analysis. The last part of the page allows for segmentation of the region of interests and creates new DVH curves for smaller segments. The visualization however is performed in the same way. The result of segmentation of the two parotid can be found in the results section.

3.3 Installation and Usage:

Back-End:

First, it is necessary to install python X.X.X from https://www.python.org/. Once it is installed, one would need to install web.py web framework from http://webpy.org/ to serve incoming HTTP requests. This can be installed using pip using the following command: “pip install web.py”. After installing the web framework, it is necessary to install the oncotools python library in order to interact with a Oncospace database. To install this library, we visited the following link: http://oncospace.github.io/oncotools/index.html# and ran: “python setup.py install” from the source directory. The necessary dependencies for this library are: schema==0.6.7, pyodbc, numpy, scipy, scikit-learn, matplotlib, pandas and pydicom.

Front-End:

To service the web application, we decided to use XAMPP. XAMPP-Virtual Machine can be installed from https://www.apachefriends.org/index.html. At the present date, we used (PHP 7.2.4). Once XAMPP is download, it is necessary to install it by opening the downloaded file. When it is finished installing, open XAMPP and press on “Start”. This will start up the Debian virtual machine that will contain the Apache web server. The status icon should turn green and an IP address should now be visible. This IP address is the IP address that has been assigned to the virtual machine. Inside XAMPP, switch to the "Volumes" tab and press on the "Mount" button. Once it is successfully mounted, press on "Explore". This should open up a new Finder window where you will be located inside the file system of the virtual machine. Open up the folder called "lamp" and then the folder called "htdocs". Inside this "htdocs" folder, remove all of the files and folders and place in all of the HTML and JavaScript code.

Usage:
To start the python web framework from command line, simply run:

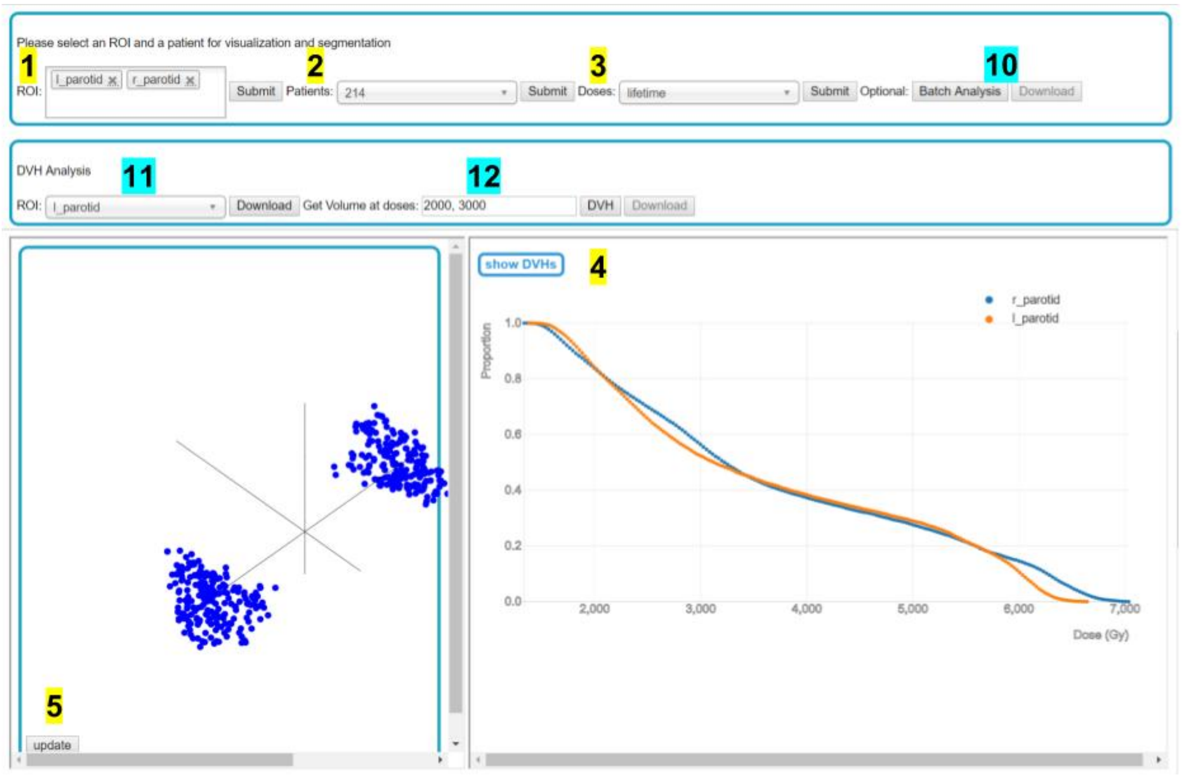
```
    $ python runweb.py
```

To access the web application, use the IP address provided by XAMPP and open up your browser to:

http://<IP_ADDRESS>/

4. Results:

The UI allows users to run through the entire DVH workflow, select segmentation tools, and perform addition DVH analysis on the segmentations. The UI also displays a 3D visualization of the selected ROIs. In addition, the following features were added: 1) generate DVH curves for all patients with selected ROIs 2) generate a "mini" DVH from user input dosages 3) download any DVH generated using the UI. All of these features are shown and explained in figures 3 and 4 - Graphical Workflow.


Please select an ROI and a patient for visualization and segmentation

ROI: l_parotid, r_parotid · Patients: 214 · Doses: lifetime · Optional: Batch Analysis

DVH Analysis

ROI: l_parotid · Get Volume at doses: 2000, 3000

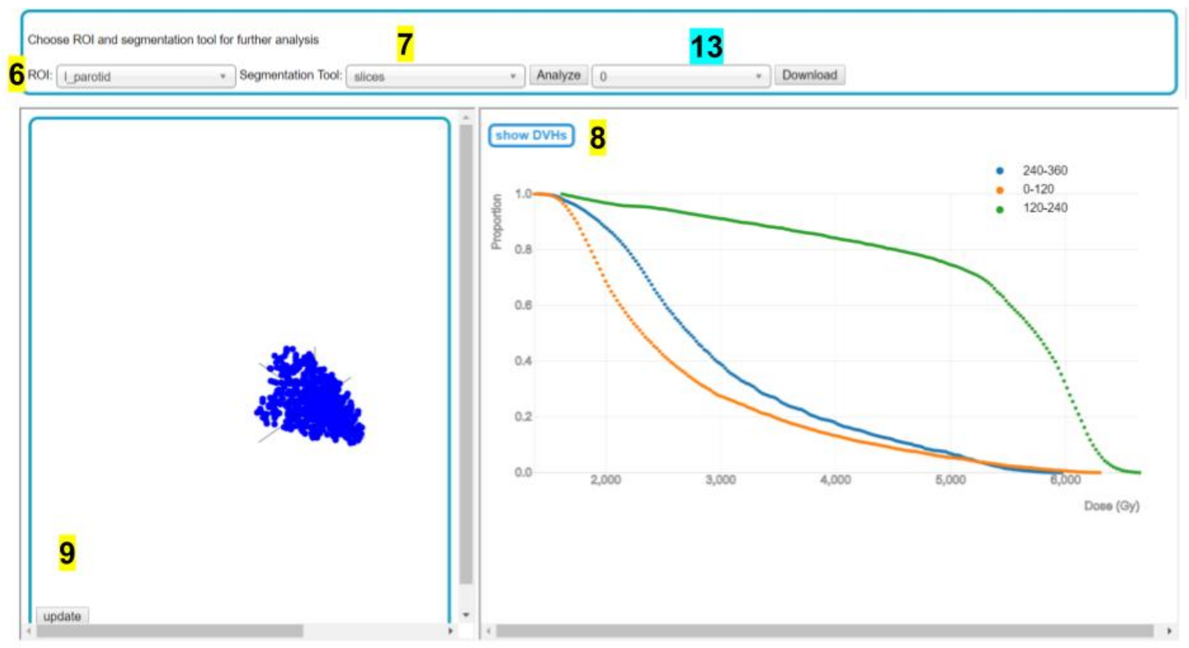


Figure 3 - Graphical Workflow. Full website is shown. Yellow number labels correspond to the core workflow, consisting of generating DVHs for selected ROIs, segmenting the selected ROI, and generating DVHs for segments. Blue labels correspond to auxiliary analysis and DVH download features. Each step is explained in detail in Figure 4.

Workflow ID	Description	Client Side[1]	Server Side[2]
1	- User selects ROIs to filter patient list. - Click 'Submit' to fetch patient list.	- getPatients(...) - populates patient select box - populates all ROI select boxes	- getp class accesses oncotools database to return patient list with specified ROIs
2	- User selects Patient - Click 'Submit' to fetch available doses	- getDoses(...) - populates dose grid select box	- getd class accesses oncotools database to return available dose grids for patient
3	- User selects Dose - Click 'Submit' to fetch DVH and point cloud data for ROIs and patient selected steps 1 & 2	- getDVH(...); getVol(...) - stores DVH data and point cloud data for selected ROIs in session storage	- getdvh class accesses oncotools database and returns JSON object holding DVHs for all ROI - getvol class returns JSON object holding point clouds for all ROI
4	- Click 'Display DVHs' to display DVHs for all ROIs for patient	- scatterplot.html calls display() - written in dvh.js - to display dvh data stored in session storage from step 3	n/a
5	- Click 'Update' to display point clouds for all ROIs for patient	- 3d.html calls init() - written in 3d.js - to display point cloud data stored in session storage from step 3	n/a
6	- User selects ROI	None until step 7	None until step 7
7	- User selects segmentation tool - Click 'Analyze' to fetch dVH and point cloud data for segmentations	- segtools(...) will call appropriate server side segmentation tool - stores DVH data and point cloud data for segmentations in session storage	- one of 4 classes for segmentation tools returns JSON object holding DVHs and point clouds for each segmentation

8	- Click ‘Display DVHs’ to display DVHs for all segments of ROI	- scatterplot_seg.html calls display() - written in dvh_seg.js - to display dvh data stored in session storage from step 7	n/a
9	- Click ‘Update’ to display point clouds for all segments of ROI	- 3d_seg.html calls init() - written in 3d.js - to display point cloud data stored in session storage from step 7	n/a
10	- Click ‘Batch Analysis’ to getch DVHs for all patients with specified ROIs - Click ‘Download’ to download .csv with all of the DVHs	- getBatch() starts the DVH fetching process in the background - downloadBatchCSV(...) downloads csv of all the DVHs	- runbatch class iterates through all patients and specified DVHs and return JSON object holding all DVH data
11	- User selects ROI - Click ‘Download’ to download .csv with DVH data for specified patient and ROI	- downloadCSV(...) downloads .csv of DVH for specified ROI	n/a
12	- User inputs doses; will get corresponding volume proportion at each dose - Click ‘DVH’ to get volumes - Click ‘Download’ to download DVH data	- pickDVH(...) fetches volume proportions corresponding to input doses - downloadPickCSV(...) download .csv of DVH	- pickDVH class iterates through each input dose and calculates the corresponding volume; constructs and returns a DVH
13	- User selects segmentation - Click ‘Download’ to download .csv with DVH data for specified segmentation	- downloadSegCSV(...) downlaods .csv of DVH for specified segmentation	n/a

1. All client side functions are located in startup.js, and called by index.html. Only the 3D and DVH plots differ in that they (workflow ID #s 4, 5, 8, and 9) use functions in 3d.js, dvh.js, and dvh_seg.js which are called by 3d.html, 3d_seg.html, scatterplot.html, and scatterplot_seg.html.
2. All client side functionality is in runweb.py.

Figure 4 - Graphical Workflow Description. Detailed explanation of each step, including client side and server-side actions.

To help guide the user through the DVH workflow, several considerations and small features were added to the UI. First, the labels for steps 1-3 in the Graphical Workflow (Figure 3) are highlighted when the user should move on to the next step. Next, a blank loading screen mask covers the screen when the user attempts to calculate DVH/point clouds. Also, all ‘download’ and some ‘submit’ buttons are inactive until their functionality is ready (necessary variables calculated).

5. Significance:

Despite the fact that radiotherapy has been very successful in treating various forms of head and neck tumors, it can lead to harmful side effects by radiating other nearby regions that are susceptible. To reduce these side effects, further studies need to be done by physicians to examine which treatment options have better outcomes, and what is the does distribution of such treatment options to the organ. Prior to the emergence of computerized planning and precision medicine, these types of decisions were usually made through experience as the doctors learned which general direction the radiation should point to and for how long. However, currently, as the number of patient reports and outcomes increase every day, and their relevant data is stored in the hospital systems, physicians should have the ability to go through this big data and discern various patterns that relate treatments with outcomes. One of the main ways to find patterns is through visual analysis, and for this purpose we have developed our framework.

Our platform enables the physicians, who do not necessarily have any technical background, to run dose toxicity analysis on their regions of interests for various patients. It also allows for segmentation of organs into smaller regions and running the DVH curves on the smaller regions. As the paper by Monti [8] suggested, this finer analysis enabled by segmentation can lead to better analysis and can improve treatment planning.

6. Management Summary:

Who did what? William worked on setting up the server-side, communications between the front-end and back-end using XHR requests, segmentation and the DVH plots on the front-end. Santiago worked on writing segmentation tools and doing data exporting using the csv format. Alaleh worked on the front end, creating the 3D visualization, solving clashes between the two D3.js libraries functioning simultaneously, and enabling querying and multiple choices for the select boxes options using Select library.

Accomplished vs. Planned Deliverables: Our deliverables and their status are as follows:

Minimum: (completed)

- A UI for visualizing organs in 3D, calculating DVH curves, and running python analysis scripts from the JavaScript layer
- Documentation

Expected: (completed)

- A UI for segmenting and analyzing organs in 3D using a list of segmentation options.
- Dose-volume data analysis scripts are integrated and can be performed on segments of organs.
- Results of the analysis can be exported.

Maximum: (ongoing)

- An interactive UI for segmenting and analyzing organs in 3D with a flexible segmentation as indicated by the user in addition to the existing ones. (additional segmentation feature ongoing)
- Create and export DVH data for batch of patients (completed)
- Extract specific points on DVH given list of doses (completed)
- User friendly - help direct user step by step & notify user with loading screen for long processes (completed)

We accomplished all minimum and expected deliverables. Currently, the only maximum deliverable not completed is the additional segmentation feature which is written however not yet tested. We plan for this to be fully implemented by the end of weekend, as it can further help analyze the data by creating a more flexible segmentation methods.

Future Steps: As future work, additional feature and segmentation tools can be added to the user interface. Color coordination between the 3D organ and DVH curves can help further clarify the information. The interface currently is run locally and can later be set up to run in a dedicated server. After usage by oncologists and radiologists, their feedback could help improve the tools by adding other visualization and analysis features.

What we learned: This project allowed us to gain some experience with D3.js, HTML and CSS languages. We also learned about JSON data formatting and various ways to connect python layer to the JavaScript layer. By performing the seminar presentation we learned about the implications of radiotherapy, its side effects, improvements in radiotherapy treatment analysis such as voxel-based dose toxicity analysis, and big data and its implications for precision medicine.

7. Acknowledgements

We would like to thank to our mentors Pranav and Dr. McNutt. They guided us through this process, helped us define our targets and how to approach them. They also provided the segmentation tools and the test data needed to build the interface. We would also like to thank Dr. Taylor for his support, guidance, and feedback.

8. References

[1] Tolentino, E. de S., Centurion, B. S., Ferreria, L. H. C., de Souza, A. P., Damante, J. H., & Rubira-Bullen, I. R. F. (2011) Oral adverse effects of head and neck radiotherapy: literature review and suggestion of a clinical oral care guideline for irradiated patients. *Journal of Applied Oral Science*, *19*(5), 448–454.
[2] 25- Prott FJ, Handschel J, Micke O, Sunderkötter C, Meyer U, Piffko J. Long-term alterations of oral mucosa in radiotherapy patients. Int J Radiat Oncol Biol Phys. 2002;54(1):203-10.
[3] Luijk, P. V., Pringle, S., Deasy, J. O., Moiseenko, V. V., Faber, H., Hovan, A, ... Coppes, R. P. (2015). Sparing the region of the salivary gland containing stem cells preserves saliva production after radiotherapy for head and neck cancer. *Science Translational Medicine, 7*(305).
[4] Huang, B. T. et al. Different definitions of esophagus influence esophageal toxicity prediction for esophageal cancer patients administered simultaneous integrated boost versus standard-dose radiation therapy. Sci Rep 7, 120 (2017).
[5] Acosta, O. et al. Voxel-based population analysis for correlating local dose and rectal toxicity in prostate cancer radiotherapy. Phys Med Biol 58, 2581–95 (2013).
[6] Wortel, R. C. et al. Dose-surface maps identifying local dose-effects for acute gastrointestinal toxicity after radiotherapy for prostate cancer. Radiother Oncol 117, 515–20 (2015).
[7] Palma, G. et al. A Voxel-Based Approach to Explore Local Dose Differences Associated With Radiation-Induced Lung Damage. Int J Radiat Oncol Biol Phys 96, 127–33 (2016).
[8] S. Monti, G. Palma, V. D'Avino, M. Gerardi, G. Marvaso, D. Ciardo, R. Pacelli, B. A. Jereczek-Fossa, D. Alterio and L. Cella, "Voxel-based analysis unveils regional dose differences associated with radiation-induced morbidity in head and neck cancer patients," Scientific Reports, 3 August 2017.