

---

# Supplementary Material

---

In this supplementary material, we describe the procedure for our analysis of the TotalSegmentator annotations of the NLST dataset. The following are our main contributions:

1. Dashboard: for exploring features extracted from annotations, and analyzing the effect of filters to detect failures in the segmentations: <https://huggingface.co/spaces/ImagingDataCommons/CloudSegmentatorResults>
2. Google Colaboratory notebook: for performing exploratory analysis of the consistency of segmentations and a comparison to measurements from the literature: [https://github.com/ImagingDataCommons/CloudSegmentatorResults/blob/main/part2\\_exploratoryAnalysis.ipynb](https://github.com/ImagingDataCommons/CloudSegmentatorResults/blob/main/part2_exploratoryAnalysis.ipynb)
3. Google Colaboratory notebook: for generating the files needed to power the dashboard (item 1) and the analysis (item 2): [https://github.com/ImagingDataCommons/CloudSegmentatorResults/blob/main/part1\\_derivedDataGenerator.ipynb](https://github.com/ImagingDataCommons/CloudSegmentatorResults/blob/main/part1_derivedDataGenerator.ipynb)

We begin by describing the dashboard, and what sorts of analysis can be performed. We then describe the analysis we can perform in the notebook and include some additional statistics. Lastly, we describe the formation of tables that are used to power the dashboard and the previous analysis.

We made available the artifacts for this analysis under the MIT license in GitHub and Hugging Face.

## A Dashboard

In order to detect failures in the segmentations, and analyze outliers, we developed an interactive Streamlit dashboard (<https://github.com/streamlit/streamlit>), hosted on HuggingFace spaces free tier (<https://huggingface.co/spaces/ImagingDataCommons/CloudSegmentatorResults>). We include two main pages in the dashboard:

1. **Summary page:** This page shows the results of applying the four heuristics to each segmentation. These heuristics are the completeness of the segmentation, the laterality check, the number of connected components=1 check, and that the volume > 5mL. The user can quickly investigate which segmentations are outliers using this page. We display the percentage of series that passed each check.
2. **Plots page:** In this page, we display two types of plots that are dynamically created based on the user input through drop-down menus and sliders. The user can quickly filter for specific regions of interest, and select which heuristics passed/failed. The first plot displayed is a violin plot, where we plot the distributions of the standard deviations of the features before and after applying the filters. These standard deviations are computed for each patient, as each patient is scanned multiple times, and can provide us with information about the consistency of the particular radiomics feature we're interested in. The second set of plots shown are upset plots, which demonstrate the number of segments that passed or failed the combinations of heuristics.

## B Exploration of results using derived tables

The Colab notebook ([https://github.com/ImagingDataCommons/CloudSegmentatorResults/blob/main/part2\\_exploratoryAnalysis.ipynb](https://github.com/ImagingDataCommons/CloudSegmentatorResults/blob/main/part2_exploratoryAnalysis.ipynb)) contains

the use cases that we studied in the manuscript. We include the generation of figures that we presented in the paper, starting with the parquet files generated from the first notebook ([https://github.com/ImagingDataCommons/CloudSegmentatorResults/blob/main/part1\\_derivedDataGenerator.ipynb](https://github.com/ImagingDataCommons/CloudSegmentatorResults/blob/main/part1_derivedDataGenerator.ipynb)). If you want to skip the creation of the parquet files, you can jump straight to this notebook.

The goal of the manuscript was to explore and understand the effect of the heuristics on the ability to filter out problematic segmentations and identify failures. We focused on three main aspects, where our Colab notebook provides the code needed to reproduce figures in our manuscript for the first and third items:

1. Consistency of left vs right structures such as the ribs
2. Within-patient volumes of structures
3. Vertebral volumes compared to a population in the literature

We now provide further details on the first item from above. For the analysis of left vs right structures, our goal was to study the consistency of the left vs right rib volumes after applying each successive heuristic. We computed the normalized difference between the left and right ribs using  $(\text{left} - \text{right}) / (\text{left} + \text{right})$ . We then applied each heuristic successively, where one filter = segmentation completeness, two filters = previous filter + connected components = 1, three filters = previous filters + volume > 5mL, and four filters = previous filters + laterality is correct. We performed five different linear mixed-effects modeling tests, to see the effect of each heuristic. We use the PatientID as a random-effect to account for the fact that each patient had multiple scans.

The table below provides the p values and if each test is significant for each pair of ribs ("is sig"). We can observe that by applying the segmentation completeness check (original data vs one filter), there was a significant difference in the normalized volume difference between almost all of the left vs right ribs. We observe the same when adding the additional heuristic of the connected components = 1 (one filter vs two filters). However, checking for the correctness of the laterality and the volume threshold of 5 mL proved to not have significant effects for most of the ribs. We observe that the 12th rib often did not follow the same trend as the other ribs. There was severe under-segmentation in many of the 12th ribs that were still above the threshold of 5 mL and were therefore not flagged by the heuristics.

Rib	Original data vs one filter		One filter vs two filters		Two filters vs three filters		Three filters vs four filters		Original data vs four filters	
	p value	is sig	p value	is sig	p value	is sig	p value	is sig	p value	is sig
<b>First rib</b>	0.16	no	<0.05	<b>yes</b>	<0.05	<b>yes</b>	1.0	no	<0.05	<b>yes</b>
<b>Second rib</b>	0.04	<b>yes</b>	<0.05	<b>yes</b>	0.78	no	1.0	no	<0.05	<b>yes</b>
<b>Third rib</b>	0.07	no	<0.05	<b>yes</b>	0.96	no	1.0	no	<0.05	<b>yes</b>
<b>Fourth rib</b>	<0.05	<b>yes</b>	<0.05	<b>yes</b>	0.87	no	0.99	no	<0.05	<b>yes</b>
<b>Fifth rib</b>	<0.05	<b>yes</b>	<0.05	<b>yes</b>	1.0	no	1.0	no	<0.05	<b>yes</b>
<b>Sixth rib</b>	<0.05	<b>yes</b>	<0.05	<b>yes</b>	0.99	no	1.0	no	<0.05	<b>yes</b>
<b>Seventh rib</b>	<0.05	<b>yes</b>	<0.05	<b>yes</b>	0.74	no	1.0	no	<0.05	<b>yes</b>
<b>Eighth rib</b>	<0.05	<b>yes</b>	<0.05	<b>yes</b>	0.70	no	1.0	no	<0.05	<b>yes</b>
<b>Ninth rib</b>	<0.05	<b>yes</b>	<0.05	<b>yes</b>	0.35	no	1.0	no	<0.05	<b>yes</b>
<b>Tenth rib</b>	<0.05	<b>yes</b>	<0.05	<b>yes</b>	0.28	no	1.0	no	<0.05	<b>yes</b>
<b>Eleventh rib</b>	<0.05	<b>yes</b>	<0.05	<b>yes</b>	<0.05	<b>yes</b>	1.0	no	<0.05	<b>yes</b>
<b>Twelfth rib</b>	0.92	no	0.07	no	<0.05	<b>yes</b>	1.0	no	<0.05	<b>yes</b>

## C Generation of base and derived tables

In this section, we describe the motivation for the generation of the base tables and then the derived tables for evaluating the heuristics. The four heuristics we developed are in part based upon the metadata extracted from the DICOM Segmentation objects, which are publicly available as part of the Google Public Datasets program at <https://console.cloud.google.com/marketplace/product/bigquery-public-data/nci-ids-data>. However, for the analysis of segmentations, we require the DICOM attribute PerFrameFunctionalGroupsSequence as it contains crucial

segmentation info such as the slices on which the segmentation is located, segment number, and segment label. This attribute was missing in the majority of the DICOM Segmentation Objects because of the limitations of Bigquery. Bigquery has a limitation of 1 MB per DICOM tag <https://cloud.google.com/healthcare-api/docs/how-tos/dicom-bigquery-streaming>. Because we encoded all segmentations (up to 104) into a single DICOM Segmentation object, the `PerframeFunctionalGroupsSequence` was often run over the 1 MB limit. To alleviate this limitation, we extracted the attribute using `pydicom` and created a workflow on Terra. Please see <https://dockstore.org/myworkflows/github.com/ImagingDataCommons/CloudSegmentator/perFrameFunctionalGroupSequenceExtractionOnTerra>. We unnested this otherwise nested attribute to create a flat table and then exported it as a parquet file and made it available as one of the base tables on GitHub as [https://github.com/ImagingDataCommons/CloudSegmentatorResults/releases/download/0.0.1/nlst\\_totalseg\\_perframe.parquet](https://github.com/ImagingDataCommons/CloudSegmentatorResults/releases/download/0.0.1/nlst_totalseg_perframe.parquet)

Secondly, while we encoded only shape and first-order radiomics features into the DICOM Structure Reports, we did not encode the general module <https://pyradiomics.readthedocs.io/en/latest/radiomics.html#module-radiomics.generalinfo> pyradiomics features as to our knowledge, the DICOM standard did not have the necessary means to encode Center of Mass. So we saved them into a JSON file. When we realized, we could make use of the general features, we extracted them into a Bigquery table first. Subsequently, we exported them as parquet files, and consolidated into a single parquet file. We make available this parquet file as the second and last base table on GitHub as [https://github.com/ImagingDataCommons/CloudSegmentatorResults/releases/download/0.0.1/json\\_radiomics.parquet.parquet](https://github.com/ImagingDataCommons/CloudSegmentatorResults/releases/download/0.0.1/json_radiomics.parquet.parquet).

We used the notebook [https://github.com/ImagingDataCommons/CloudSegmentatorResults/blob/main/part1\\_derivedDataGenerator.ipynb](https://github.com/ImagingDataCommons/CloudSegmentatorResults/blob/main/part1_derivedDataGenerator.ipynb) to generate the following derived tables:

- `bodyPartAndLaterality`: This is an intermediate table that contains information about the body part segmented by `TotalSegmentator`, segment number, source CT series, and its laterality.
- `Segmentation Completeness`: This table contains info about whether a segment had at least one slice below and above the segmentation.
- `Laterality`: This table contains if laterality (left vs right) is correctly assigned by `TotalSegmentator`.
- `Qualitative Checks`: This table contains the three heuristics: segmentation completeness, laterality, and connected components. The fourth heuristic is added when merged with the quantitative measurements below.
- `Flat Quantitative Measurements`: This table contains the pivoted quantitative measurements for all `TotalSegmentator` segmentations. Effectively each row represents a segment and all 28 radiomics features are present in their columns.
- `qualitative_checks_and_quant_measurements`: This is the result of combining all the heuristics along with 28 radiomics features for each segment along with the general module features `VolumeNum` which gives us the number of connected components. This file may be the most useful and is the file that is powering the Hugging Face Dashboard and all our analysis in the part 2 colab notebook.

For all the above tables, we included schema files as well in the same release (<https://github.com/ImagingDataCommons/CloudSegmentatorResults/releases/tag/0.0.1>)

## C.1 Compute Environment

The `part2_exploratoryAnalysis` notebook was tested on a free colab instance (2 vCPUs, 13 GB RAM) and takes about 2 hrs. For `part1_derivedDataGenerator` notebook, we initially tested it on a 32vCPUs 256 GB RAM Jetstream2 instance. However, we made several optimizations since then to bring the RAM consumption low despite leading to a longer run times. We were able to run it successfully even on 2vCPUs, 16 GB RAM free tier hugging face jupyterlab space. Run times get better if one has access to better computing resources. Other runtimes, we tested include the 2vCPU

13 GB free colab instance and the 8vCPU, 51 GB Colab Pro High-RAM instance. The runtimes vary anywhere from 4 hrs to 10 hrs. We note that no cloud credentials are necessary as we queried the metadata that is made available for the public for free in AWS buckets. We used duckdb, an in-memory database as it can handle highly complex data in a tiny footprint.