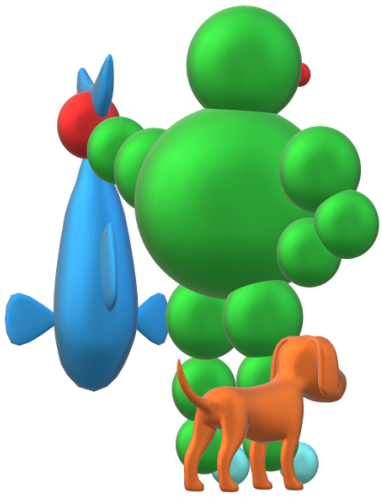
Dose Hunter
User Guide

Abstract

This document describes how to install and how to use DoseHunter, a tool to collect a large number of data in your Varian Eclipse database.



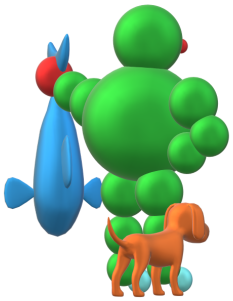
Corresponding authors:

- Luc SIMON: simon.luc@iuct-oncopole.fr

Contributors:

- Luc SIMON
- Bradley BEEKSMA
- François-Xavier ARNAUD
- Killian LACAZE
- Farzam SAYAH

Contents



1. What is DoseHunter?

DoseHunter is a stand-alone executable for Varian Eclipse V15.x. DoseHunter automatically collects dose data (dose max, dose min, D95%, etc.) for a large number of patients in your database. DoseHunter creates a .csv file with your data that can be easily treated with Excel, Python...

2. Installation

If you are lucky the executable application can be directly run on your Eclipse station (**not** a Citrix station): see part 2.1. However it may be necessary to compile the source code (part 2.2).

2.1. Executable

In an empty directory of your Varian Eclipse v15.x Station (tested with v15.6), download the 3 following files from github (*https://github.com/uhqd/DoseHunter*):

- DoseHunter.exe
- VMS.TPS.Common.Model.API.dll
- VMS.TPS.Common.Model.Types.dll

The 2 last files can be replaced by your own .dll corresponding to your version of Eclipse (available on any Eclipse station) at the following path (can be slightly different on your station):

```
C:\Program Files(x86)\Varian\RTM\15.6\esapi\API\VMS.TPS.Common.Model.API
C:\Program Files(x86)\Varian\RTM\15.6\esapi\API\VMS.TPS.Common.Model.Types
```

Copy them in your working directory.

If DoseHunter doesn't work, you probably have to compile the source (see 2.2)

2.2. Compilation

Download the repository from github *https://github.com/uhqd/DoseHunter*. Open the solution (.sln) in *Microsoft Visual Studio Community* for example.

You must add in the reference list of the project, the 2 .dll files (see part 2.1 where to find these files). Compile and build the project in a directory. Copy the two .dll files in the same directory than the .exe file. This working directory must also contains the three user text files: **id.txt**, **planfilter.txt** and **index.txt** (see sections 3.1, 3.2 and 3.3). An example of these files can be found on the github repository (https://github.com/uhqd/DoseHunter).

3. How to use DoseHunter?

A simple double click on the DoseHunter.exe file executes the program. The three following text files must be placed in the same working directory to detail which data you want: **id.txt**, **planfilter.txt** and **index.txt**.

3.1. id.txt

id.txt is a simple text file (can be done with the Microsoft Notepad) that contains only the list of your patient IDs (one by line). It can be done by copy/paste a column from an Excel file in the Notepad. Do not add any space character or any additional line breaks. An exemple can be found here: id.txt.

3.2. index.txt

index.txt is another text file to indicate which data you want to collect for your list of patient (see previous section 3.1). Each line must be in the following format:

```
<struct. name 1>;<struct name 2 (opt)>;<struct. name 3 (opt)>,<index>,<index>,<index> (etc.)
```

This is an example of the file content:

```
Heart;HEART;heart,max,min,median,D95%
PTV,max,D95cc,median
Liver;liver,max,V50cc,vol
```

Please, notice the different use of the comma (,) and the semicolon (;):
Each line is separated in different elements using a comma (,). **The first element** is the structure name. It can be intersting (but not mandatory) to give different spelling for the same stucture name. Then these different

names must be separated by a semicolon (;)
DoseHunter will search the first spelling of the structure name in the patient. If (and only if) DoseHunter doesn't find this structure name, it will try the second one, the third one... In our example, the first stucture can be named **Heart**, **HEART** or **heart**. If, for a patient, the structure **Heart** exists (prefered spelling), other spelling will be ignored.

The others elements on the line are the indexes that you want to collect for a given structure. You can add any number of inex. Candidates are:

- **vol** : volume of the structure (**Vol** and **VOL** are tolerated)
- **min** : minimum dose of the structure (Gy) (**Min** and **MIN** are tolerated)
- **max** : max dose of the structure (Gy) (**Max** and **MAX** are tolerated)
- **mean** : mean dose of the structure (Gy) (**Mean** and **MEAN** are tolerated)
- **median** : median dose of the structure (Gy) (**Median** and **MEDIAN** are tolerated)
- **DXX%** or DXXcc : e.g. D95% or D2.5cc : Dose (Gy) recieved by 95% or 2.55 cc of the structure
- **VXX%** or VXXcc : e.g V49.6% or V49.6cc : Volume in % or cc that recieved 49.6 Gy

- **VXX%** or VXXcc : e.g V49.6% or V49.6cc : Volume in % or cc that recieved 49.6 Gy

- **HI** or hi : Homogeneity Index in the structure:
  $HI = \frac{D2-D98}{D50}$ where D2, D50 and D98 are the doses received by 2%, 50% and 98% of the chosen structure, respectively.

- **CIxx**: e.g. CI95 : Conformity Index for the isodose 95%
  $CI95 = \frac{PIV}{TV}$ where PIV is the volume of the isodose 95% and TV is the structure volume, generally a target.

- **PIxx**: e.g. PI95 : Paddick Conformity Index for the isodose 95%
  $PI95 = \frac{TV_{PIV}^2}{TV \times PIV}$ where TV_{PIV} is the volume of the structure that recieved 95% of the dose, PIV is the volume of the isodose 95% and TV is the structure volume, generally a target.

No relative doses are allowed. All doses are expressed in Gy. A last example: if you write in your file

```
Heart;heart;HEART,vol,V20cc
PTV,vol,PI100,HI
```

DoseHunter will collect for every accepted plan of every patient the volume and the V20Gy (cc) for the structures called **Heart** or **heart** or **HEART**. It will also collect the volume, the Paddick Index and the Homogeneity index of the structure **PTV**.

3.3. planfilter.txt

It may be convenient to choose which plans of your patient you want to explore. For example, you may need to exclude QA plans. It is possible

to filter the plans by using the **planfilter.txt** file. If the file is not in the working directory, default values will be used. The file must not be modified, except the values after the "**:**". This is an example of the file content:

```
# HOW TO FILTER YOUR PLAN
# DO NO MODIFY THIS FILE EXCEPT THE PART AFTER THE ":"

# keep only the plans with a total dose > to a value
Min Total Dose (Gy):60.0

# keep only the plans with a total dose < to a value
Max Total Dose (Gy):200.0

# Treat approved? If "no" treat approved plans will be excluded.
TreatApproved plan:no

# Planning approved? If "no" planning approved plans will be excluded.
PlanningApproved plan:no

# Unapproved? If "no" unapproved plans will be excluded.
Unapproved plan:yes

# Named plans? If "no", plans with a name will be excluded.
Named plan:yes

# Unnamed plans? If "no", plans with no name will be excluded.
Unnamed plan:yes

# keep it if it contains a string? If "yes", plans will be kept only
if they contain the string
Plan name must contain a string:no:toto

# Exclude it if it contains a string? If "yes", plans will refused
if they contain the string
Exclude if plan name contains:no:toto

# Explore Sum plans ?  if no, sum plans are ignored, yes to explore them
Explore Sumplans:yes
```

```
# Explore Uncertainty plans ?  if no, uncertainty plans are ignored, yes to explore them. Warning: it slows the executi
Explore uncertainty:no

# keep the COURSE it if it contains a string? If "yes", course will be kept only if they contain the string
Course name must contain a string:yes:tutu
```

In this example, the user wants to keep only the following plans:

- Total dose between 60 and 200 Gy
- exclusion if plan status is "Treat Approve"
- exclusion if plan status is "Planning Approve"
- keep plans if plan status is "Unapproved"
- keep plans with a name (All plans have an ID but not necessarily a name)
- keep plans without a name
- no test on the ID here. The user could choose to keep or exclude plans with an ID that contains a string, e.g. if you want to exclude all plans for which the ID contains the string '**breast**' or '**BREAST**' the last line of the file must be:

```
Exclude if plan name contains:yes:breast:BREAST
```

- Sum plans are not ignored. Please notice that if Sum plans are explored, all filters will be ignored for these plans except "test if the plan has a name", "keep it if the plan name contains a string", "exclude it if the plan name contains a string". Moreover, the plan sum has no creation date (dose hunter will give the date of the last plan) and creation user

will be set to plan_sum. The number of fractions is the sum of all fractions of all plans. The Total dose is the sum of the total doses of all plans.

- uncertainties plans will not be analyzed. Warning: analysis of plan uncertainties slows a lot the execution. Resolution can be changed by developpers in the code in the class DVH by changing the value 0.001 in the following lines:

```
...
return uncert_plan.GetDVHCumulativeData(structure,
            DoseValuePresentation.Absolute,
            VolumePresentation.Relative, 0.001);
...
 return uncert_plan.GetDVHCumulativeData(structure,
            DoseValuePresentation.Absolute,
            VolumePresentation.AbsoluteCm3, 0.001);
```

- Only course with a couseID containing the string 'tutu' will be analyzed

A lot of other filters could be implemented by advanced users, but you will need to modify the part TEST THE PLAN in the DoseFinder.cs file and to recompile DoseHunter.

3.4. Execution of DoseHunter

When double-clicking on DoseHunter.exe, the patient list is read (see the description of id.txt in section 3.1). Every *course* is opened and for each

course every plan is opened (except is the plan is refused, due to the plan filters, see section 3.3).

Then, for each patient of the list, DoseHunter will try to collect the chosen data (see part 3.2) and to write them in an output .csv file (see part 3.5). No need to open Eclipse on the Station, DoseHunter is a stand alone executable!
When a plan is opened, its name and its total dose is displayed on the console. Then DoseHunter get the DVH data and collect the chosen ones. If the structure is not found the data is not reported but the program continues to run.

3.5. Output

An **output/** directory is crerated for the output files. **Be careful**, this directory is deleted and recreated at each execution of DoseHunter. At the end of the execution the console displays a synthetic report of the execution and wait for the user to type ENTER.

The **output/** directory contains 2 files:

- A **log.txt** file that contains more or less the same messages that have been diplayed on the console during the execution of DoseHunter.
- A **data.csv** file that can be opened with Excel or a Python script and that contains the collected data. Each data is separated by a semicolon (;) and each line has the following format (one line per accepted plan, a patient can have several accepted plans):

```
<patient ID>;<course ID>;<plan ID>;<plan date>;<author of the plan>;
```

```
<total dose>;<dose per fraction>;<number of fractions>;<MU>;<MI (MU/fraction)>;
<plan normalisation value>;<index>;<index>;<index>...
```

The first line of the .csv file gives the titles of the columns. The other lines give the data separated by a semicolon. If the value of an index is not found (e.g. DoseHunter did not find a structure) the cell is left empty (it will appear in the csv file as a double semicolon, e.g. 3;2;4;;1).

.