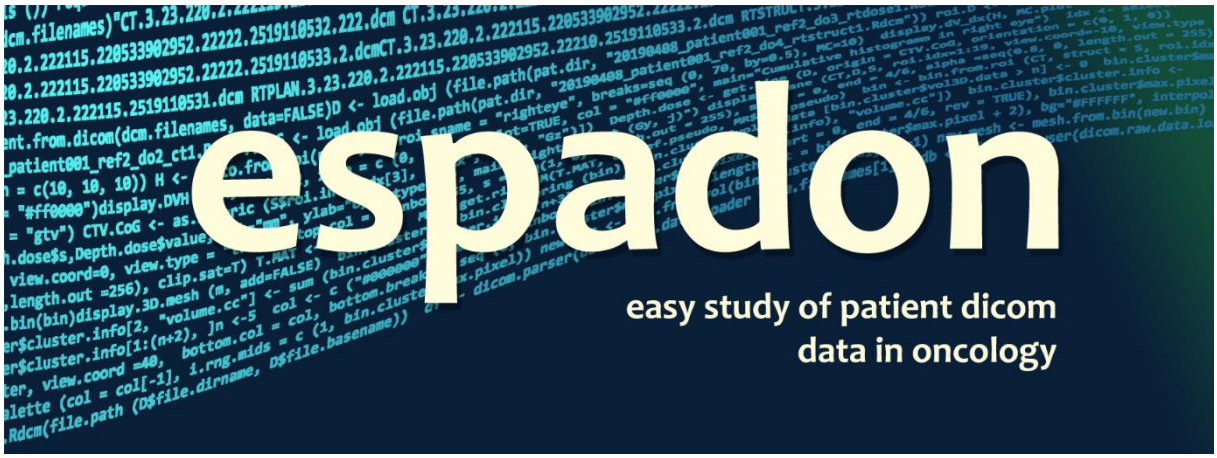


Computing cumulative dose

C. Fontbonne and J.-M. Fontbonne | LPC Caen
Normandie Univ, ENSICAEN, UNICAEN, CNRS/IN2P3, LPC Caen, 14000 Caen, France
Contact.espadon@lpccaen.in2p3.fr
https://espadon.cnrs.fr/

Table of content

1 The case

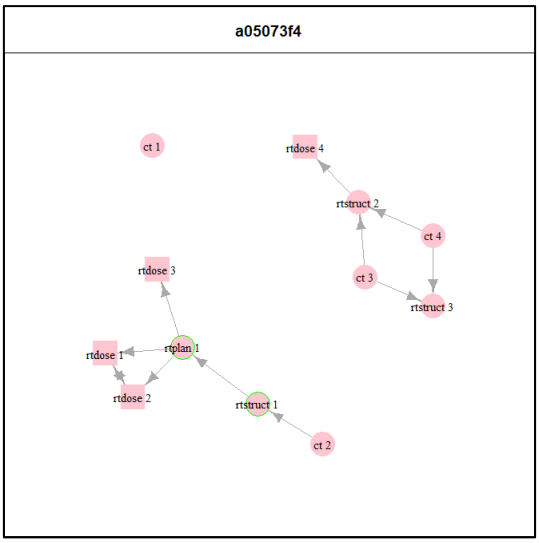
The patient had a proton beam treatment followed by a boost with a CyberKnife (CK) machine (because the tumor volume was located near the optical nerve). We want to cumulate both plans in order to get the total dose received by organs.

First, we load the plans and display patient-oriented view:

```
require (espadon)

pat.dir <- choose.dir ()
pat <- load.patient.from.dicom (pat.dir)

dev.new (width = 4, height = 4, noRStudioGD = T)
display.obj.links (pat)
```



The patient-oriented view

The proton plan consists of two beams (rtdose 1 and rtdose 2) whose sum is stored in rtdose 3. The CK plan is in rtdose 4. Each plan has been built on its own CT, ct 2 for the proton plan, and ct 3 or 4 for the CK.

The two plans are linked by a reg file since their colours are identical. So we have no particular problem. We will also illustrate how to do this when this connection does not exist.

We now have to load the necessary data as suggested by the patient-oriented view:

```
D.p <- load.obj.data (pat$rtdose[[3]])
S.p <- load.obj.data (pat$rtstruct[[1]])
CT.p <- load.obj.data (pat$ct[[2]])

D.CK <- load.obj.data (pat$rtdose[[4]])
S.CK <- load.obj.data (pat$rtstruct[[3]])
```

Note there is far more information in the objects themselves to perform this operation automatically.

Just for information:

```
D.p$max.pixel

[1] 53.03664
```

```
D.CK$max.pixel

[1] 20.25316
```

The proton plan delivered 53 Gy, and the CK plan, 20.5 Gy. The question is “what is the cumulative dose ?”.

2 How to do in this case?

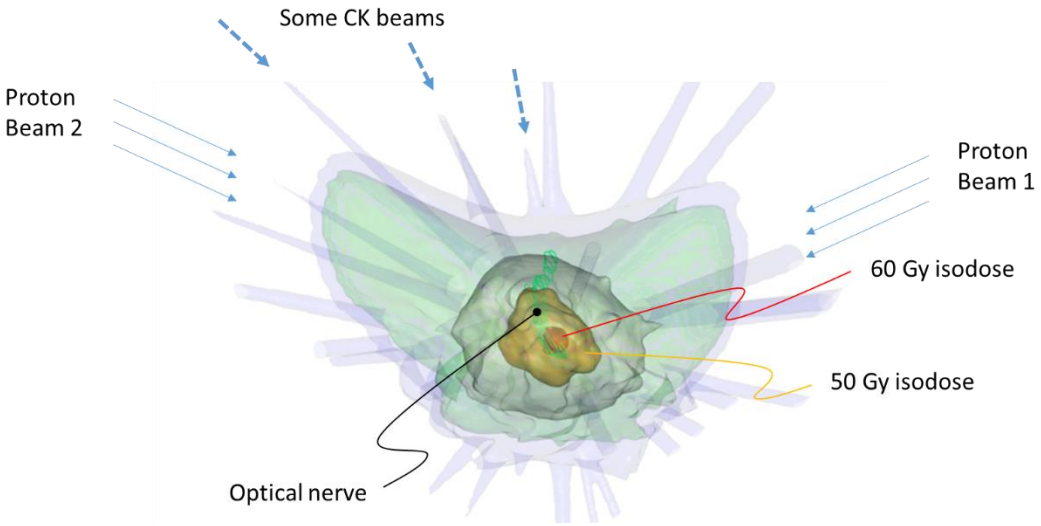
It is not possible to add two volumes if they do not share the same gridding. So, we just have to choose a destination grid (the proton plan), and resample the CK on the proton grid. Note we had to transmit the `T.MAT` as volumes do not share the same reference frame. In doing so, since the CK plan is smaller than the proton plan, espadon fills in the unknown areas with `NA`. These NAs must then be replaced by the desired value (in this case, zero), before summing to obtain the cumulative plan:

```
D.CK.in.p <- vol.regrid (D.CK, D.p, T.MAT=pat$T.MAT)
D.CK.in.p$vol3D.data[is.na (D.CK.in.p$vol3D.data)] <- 0
D.tot <- vol.sum (D.p, D.CK.in.p)
```

Now, we can work on the cumulative plan `D.tot`! That is all we have to do.

The following code is just for checking everything goes well:

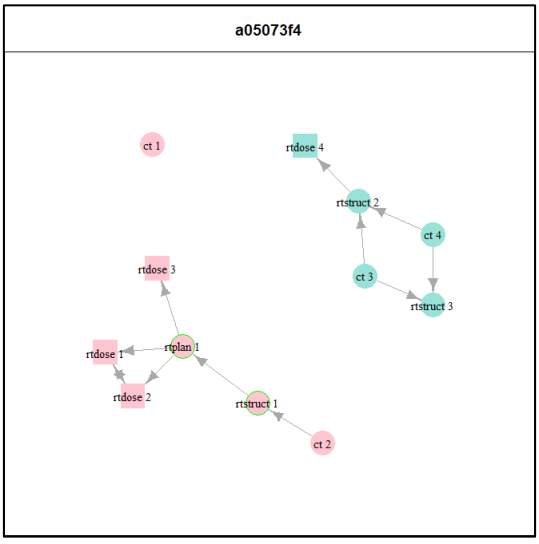
```
D.60 <- bin.from.vol(D.tot, min=60)
m.60 <- mesh.from.bin (D.60)
display.3D.mesh(m.60, col="#FF0000", alpha=1)
D.50 <- bin.from.vol(D.tot, min=50)
m.50 <- mesh.from.bin (D.50)
display.3D.mesh(m.50, col="#FFA500", alpha=0.5)
D.20 <- bin.from.vol(D.tot, min=20)
m.20 <- mesh.from.bin (D.20)
display.3D.mesh(m.20, col="#FFFFE0", alpha=0.3)
D.5 <- bin.from.vol(D.tot, min=5)
m.5 <- mesh.from.bin (D.5)
display.3D.mesh(m.5, col="#00FF00", alpha=0.1)
D.1 <- bin.from.vol(D.tot, min=1)
m.1 <- mesh.from.bin (D.1)
display.3D.mesh(m.1, col="#0000FF", alpha=0.05)
display.3D.contour(S.p, roi.name="nod")
```



3D representation of the cumulative dose produced by the two proton beams and the CK.

3 What if plans are not connected by a reg file?

Once loaded, the plan accumulation by itself uses only three instructions, thanks to the `T.MAT` process. However, it is often the case that the change of reference frame matrices are not available… In these conditions, one must be creative. As each plan has its own rt-struct, it is possible (although debatable) to use the common ROIs to create your own transfer matrix.



The same patient as above where we removed the reg file. Now, the proton and CK plan cannot be simply merged as their colors are not the same.

What we will attempt below is to find the transformation matrix ${}^{p}T_{CK}$ that transforms certain organs contoured in the CK plan into the same organs contoured in the proton plan. If we have N organs in each plan, the problem consists in finding the matrix that satisfies:

$$M_p = {}^{p}T_{CK} \cdot M_{CK}$$

Where M_p and M_{CK} are 4xN matrixes (the last row being filled by ones).

The following lines fill the requested point matrix for six organs using their center of gravity:

```
organs <- c("oeild", "oeilg", "chiasma", "hypophyse", "oreilled", "oreilleg")
p.sel <- select.names (S.p$roi.info$roi.pseudo, roi.sname = organs)
CK.sel <- select.names (S.CK$roi.info$roi.pseudo, roi.sname = organs)

M.p <- t (as.matrix (cbind (S.p$roi.info[p.sel, c("Gx", "Gy", "Gz")], t = 1)))
M.CK <- t (as.matrix (cbind (S.CK$roi.info[CK.sel, c("Gx", "Gy", "Gz")], t = 1)))
```

The result is :

```
M.p

            4         5        10        13         49        50
Gx -25.797562 32.610765  2.724677  2.031338 -25.149906 27.705327
Gy  17.063561 19.085867 66.846765 68.086479  97.052791 98.059785
Gz   8.573492  9.033885 28.186862 18.433472   5.256013  3.927604
t    1.000000  1.000000  1.000000  1.000000   1.000000  1.000000

M.CK

            4        5        10        13         49        50
Gx -30.224554 28.18934 -1.111027 -2.063366 -29.500811 23.673654
Gy  17.449021 19.29326 67.000020 68.335599  97.303935 98.150588
Gz   6.966172  6.32097 26.001325 16.348917   3.635473  1.418336
t    1.000000  1.00000  1.000000  1.000000   1.000000  1.000000
```

Looking at the matrices, it is quite clear that, in the case illustrated here, the transformation essentially involves a translation. However, in the general case, the transformation can also imply rotations about the various axes. We will deal with this general case.

The transformation matrix can be written as a combination of three rotations (one around each axis), followed by a translation:

$$^{p}T_{CK} = T \cdot R_z \cdot R_y \cdot R_x$$

With :

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R_y = \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T = \begin{pmatrix} 0 & 0 & 0 & \Delta_x \\ 0 & 0 & 0 & \Delta_y \\ 0 & 0 & 0 & \Delta_z \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Therefore, we have six parameters to estimate, three of which are non-linear (the rotations).

To do this, we will use a solver. It needs a cost function to minimize and will try to find the six parameters by performing a gradient descent on this cost function. The cost function we will develop is simply the mean square deviation between the points in the space of the proton plan and the transformed points in the space of the CK plan. The transformation matrix is itself defined from the six parameters:

```
build.T <- function (p) {
  Rx <- matrix (c(1, 0, 0, 0,
                  0, cos (p[1]), -sin (p[1]), 0,
                  0, sin (p[1]), cos (p[1]), 0,
                  0, 0, 0, 1), nrow = 4, byrow = TRUE)
  Ry <- matrix (c (cos (p[2]), 0, sin (p[2]), 0,
                   0, 1, 0, 0,
                   -sin (p[2]), 0, cos (p[2]), 0,
                   0, 0, 0, 1), nrow = 4, byrow = TRUE)
  Rz <- matrix (c (cos (p[3]), -sin (p[3]), 0, 0,
                   sin (p[3]), cos (p[3]), 0, 0,
                   0, 0, 1, 0,
                   0, 0, 0, 1), nrow = 4, byrow = TRUE)
  Tr <- matrix (c (1, 0, 0, p[4],
                   0, 1, 0, p[5],
                   0, 0, 1, p[6],
                   0, 0, 0, 1), nrow = 4, byrow = TRUE)
  Trans <- Tr %*% Rz %*% Ry %*% Rx

  return (Trans)
}

cost <- function (p, M1, M2) {
  M2_1 <- build.T (p) %*% M2
  sqrt (sum (apply (M1 - M2_1, 2, function (V) sum (V^2))) / ncol (M1))
}

fit <- nlm (cost, c(0,0,0,0,0,0), M.p, M.CK)
```

the result gives:

```
str (fit)

List of 5
 $ minimum   : num 0.127
 $ estimate  : num [1:6] -0.000776 -0.018049 0.00361 4.596098 -0.222995 ...
 $ gradient  : num [1:6] -1.66e-06 1.08e-06 -1.39e-06 3.11e-08 -4.24e-08 ...
 $ code      : int 2
 $ iterations: int 49

build.T (fit$estimate)

            [,1]         [,2]          [,3]      [,4]
[1,] 0.999830607 -0.003595703 -0.0180506864  4.596098
[2,] 0.003609115  0.999993235  0.0007104633 -0.222995
[3,] 0.018048010 -0.000775490  0.9998368207  2.202375
[4,] 0.000000000  0.000000000  0.0000000000  1.000000

old.pat$T.MAT$matrix.list$`ref1<-ref2` # here, we kept a copy of the registration stored in old.pat
                                       # for illustration purpose. In general, you do not have this matrix

           [,1]          [,2]          [,3]       [,4]
[1,] 0.99981790 -0.0030698590 -0.0188329900  4.6060400
[2,] 0.00308085  0.9999951000  0.0005546421 -0.2967592
[3,] 0.01883120 -0.0006125627  0.9998225000  2.1832490
[4,] 0.00000000  0.0000000000  0.0000000000  1.0000000
```

`fit$minimum` tells us the transformed points are 0.127 mm_{RMS} away what they should be. `fit$code` 2 means the solver hopes he has the right solution which is stored in `fit$estimate`. If we send this solution to the `build.T` function, we get the estimated transform matrix that is compared with the ground truth

pat$T.MAT$matrix.list$`ref1<-ref2` as in this example, we had this matrix. As can be seen, the estimation of translations is relatively correct, as well as for the rotation angles (which here are almost zero).

Last, we have to declare this new transform matrix in order to use is later.

```
Transf <- build.T (fit$estimate)
new.T.MAT <- ref.srctodest.add ("ref2", "ref1", Transf, pat$T.MAT)
```

From then on, it is sufficient to use the T.MAT mechanism, specifying, whenever necessary, to use `new.T.MAT` instead of `pat$T.MAT`.

Of course, before using this type of connection, it is important to ensure that it is legitimate, as there is nothing to prevent the algorithm from returning a result, even if it does not make sense. For this to have a chance of working, you need to :

1. that the selected volumes are not coplanar (unless we are only looking for a translation, in which case we do not need to identify the whole transformation matrix)
2. that the tumour has not changed volume, otherwise it disturbs the center of gravity of the neighbouring volumes
3. that the position of the patient does not have too much impact on the position of the center of gravity of the control volumes, otherwise the problem is not reduced to a simple rotation/translation
4. that the volumes are equally contoured in both planes, otherwise the center of gravity are not correct.
5. we are certainly forgetting many necessary conditions...