Modelling Tumour Survival After Radiotherapy

Anika Watson

12/10/2020

Preamble and Loading Data

This is an R Markdown document so you can see the code and the outputs even if you don't have R.

First, let's get all the boring stuff out of the way. I've left it in here in case you want to learn to code in R, but if you don't care feel free to skip ahead.

```r
#load packages
library(matlab)
```

```
##
## Attaching package: 'matlab'

## The following object is masked from 'package:stats':
##
##     reshape

## The following objects are masked from 'package:utils':
##
##     find, fix

## The following object is masked from 'package:base':
##
##     sum
```

```r
#create input+output folder
wd <- getwd()

#check if input folder already exists otherwise create one
#####ATTENTION:data to work with should be stored in this folder before running the code!!
folders <- "data_input"
if (file.exists(folders) == FALSE) {
  dir.create(file.path(wd, folders), showWarnings = FALSE)
} else print("Input Folder Already exists")
```

```
## [1] "Input Folder Already exists"
```

```r
#output folder check
figures <- "figures"
if (file.exists(figures) == FALSE) {
  dir.create(file.path(wd, figures), showWarnings = FALSE)
} else print("Output Folder Already exists")
```

```
## [1] "Output Folder Already exists"
```

```r
#map folders to R structure
outputlocation <- paste(wd, "/figures/" , sep = "")
inputlocation <- paste(wd, "/data_input/" , sep = "")
data.files <- list.files(inputlocation)
```

Introduction

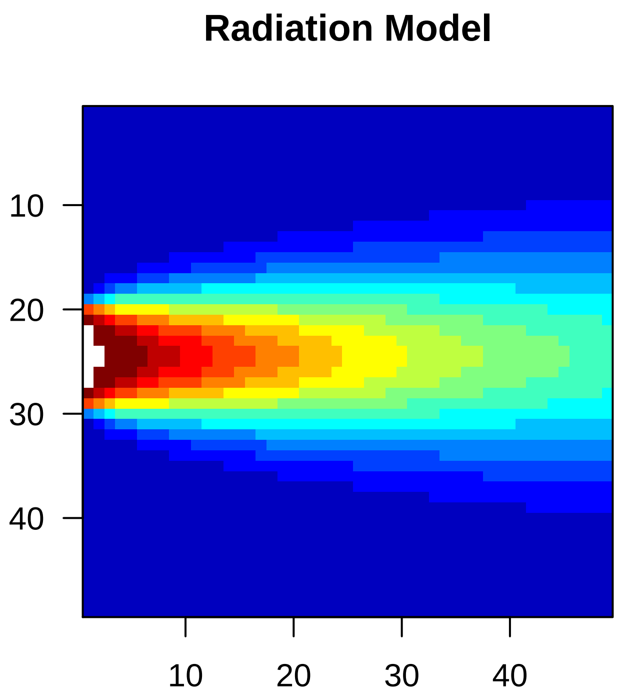
Our bodies are constantly exposed to radiation which can damage DNA, often referred to as "the blueprint of every cell". In some cases, radiation damage may even lead to cancer, a catch-all phrase for the uncontrolled divisionof cells. In order to survive in our environment, we require a certain amount of resistance to this damage. Ourresistance is far from perfect and one particular weakness is that our cells "let down their guard" so to speak, andare prone to radiation damage when dividing into daughter cells. This means cells that divide more rapidly are moresusceptible to radiation. As mentioned above, cancer cells divide "uncontrollably," making them more sensitive toradiation, and allowing radiation to target cancer cells and kill tumours.Of course, nothing is perfect and radiotherapy is no exception. While cancer cells have a a higher radiosensitivitythan healthy ones, radiation still affects healthy tissues, particularly those that grow quickly such as skin, bonemarrow, and intestinal lining.

To quote the American Cancer Society, "Radiation therapy is always a balancebetween destroying the cancer cells and minimizing damage to the normal cells," (American Cancer Society, 2014). To that end we seek to predict how much of a tumor will survive depending on the angles, widths, and intensities of thebeams of radiation to be administered.

Modelling the Beam

First, we need to model how much radiation a beam will deposit in an area of tiddue. Fortunateky, we do not need to derive this model from scratch and can load the data from (Bernard et al., 2012).

```r
#inital load of data
dose <- read.csv(paste(inputlocation, data.files[1], sep=""), stringsAsFactors = FALSE )

# Scale the data so we don't zap the healthy tissues so much
for(i in 1:dim(dose)[1]){
  for(j in 1:dim(dose)[2]){
    dose[i,j] <- (dose[i,j])/1000
  }
}

#And of course let's see what the data looks like
imagesc(as.matrix(dose), xlab="", ylab="", col=jet.colors(16), main = "Radiation Model")
```

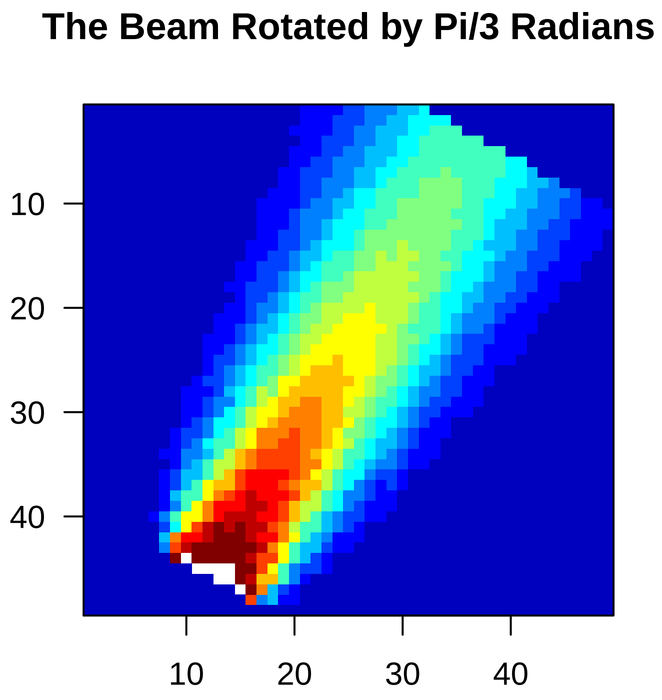


So far so good, we have successfully loaded the data modeling how a beam of radiation penetrates an area of water. Assuming that living tissue has similar properties to water, this same model can apply to radiating tumours.

Rotating the Beam

Now we need to figure out how to rotate the beam an angle θ_1. We are going to achieve this by populating a new data frame with the strength that each point would receive if it were rotated by θ_1 radians. This will achieve the same effect as rotating the beam by -θ_1 radians.

```r
#First let's tell the computer how many iterations we need to fill the grid
iterationsx = dim(dose)[1]
iterationsy = dim(dose)[2]
#now let's set the angle that we want to rotate the beam
theta1 = -pi/3
#and define the rotation matrix that will rotate a vector by that amount
rotation1 = c(cos(theta1), sin(theta1), -sin(theta1), cos(theta1))
A1 = matrix(rotation1, nrow = 2, ncol = 2)
#Now let's make an empty grid (its technically a matrix) to populate with the dosage values
output1 <- matrix(ncol=iterationsy, nrow=iterationsx)
#And now we can populate that grid with the new dosage values from the rotated beam using a for loop
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    vec = matrix(c((i-25),(j-25)), nrow = 2, ncol = 1)
    rot = A1 %*% vec
    if(round(rot[1])<=-25){
      output1[i,j] <- 0
    } else {if(round(rot[1])>=25){
      output1[i,j] <- 0
    } else {if(round(rot[2])>=25){
      output1[i,j] <- 0
    } else {if(round(rot[2])<=-25){
      output1[i,j] <- 0
    } else {output1[i,j] <- dose[(round(rot[1])+25), (round(rot[2])+25)]
    }}
    }
    }
  }
}

#And let's take a peek to ensure we rotated the beam and didn't cause complete chaos
imagesc(output1, xlab="", ylab="", col=jet.colors(16), main = "The Beam Rotated by Pi/3 Radians")
```

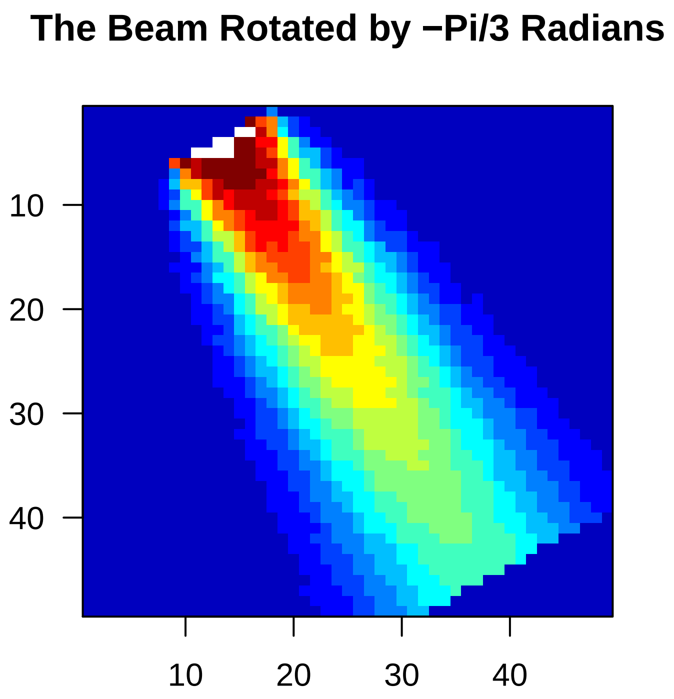


Yay the beam has rotated! Notice however that part of the beam got cut off when we rotated the square grid. This will be okay as long as the tumours we consider reside within a distance of 24.5 square widths of the centre.

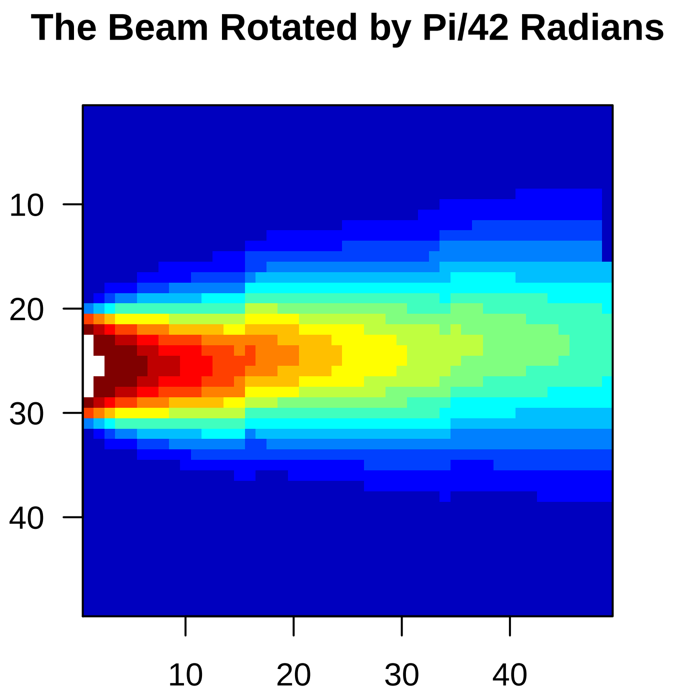
If we want to consider multiple beams penetrating from different angles, say θ_2 and θ_3, then we can simply repeat this process.

```
#set the angle
theta2 = pi/3
#define the rotation matrix
rotation2 = c(cos(theta2), sin(theta2), -sin(theta2), cos(theta2))
A2 = matrix(rotation2, nrow = 2, ncol = 2)
#make an empty grid
output2 <- matrix(ncol=iterationsy, nrow=iterationsx)
#populate that grid with the new dosage values
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    vec = matrix(c((i-25),(j-25)), nrow = 2, ncol = 1)
    rot = A2 %*% vec
    if(round(rot[1])<=-25){
      output2[i,j] <- 0
    } else {if(round(rot[1])>=25){
```

```r
      output2[i,j] <- 0
    } else {if(round(rot[2])>=25){
      output2[i,j] <- 0
    } else {if(round(rot[2])<=-25){
      output2[i,j] <- 0
    } else {output2[i,j] <- dose[(round(rot[1])+25), (round(rot[2])+25)]
    }}
    }
    }
  }
}

#ensure we rotated the beam and didn't cause complete chaos
imagesc(output2, xlab="", ylab="", col=jet.colors(16), main = "The Beam Rotated by -Pi/3 Radians")
```



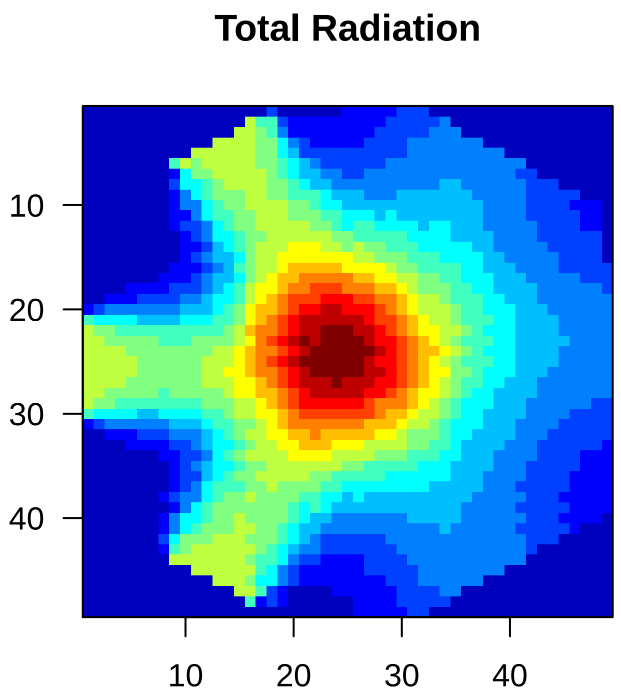
```r
#set the angle
theta3 = -pi/58
#define the rotation matrix
rotation3 = c(cos(theta3), sin(theta3), -sin(theta3), cos(theta3))
A3 = matrix(rotation3, nrow = 2, ncol = 2)
#make an empty grid
output3 <- matrix(ncol=iterationsy, nrow=iterationsx)
#populate that grid
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    vec = matrix(c((i-25),(j-25)), nrow = 2, ncol = 1)
    rot = A3 %*% vec
    if(round(rot[1])<=-25){
      output3[i,j] <- 0
    } else {if(round(rot[1])>=25){
      output3[i,j] <- 0
      } else {if(round(rot[2])>=25){
        output3[i,j] <- 0
      } else {if(round(rot[2])<=-25){
        output3[i,j] <- 0
      } else {output3[i,j] <- dose[(round(rot[1])+25), (round(rot[2])+25)]
      }}
      }
    }
  }
}

#ensure we rotated the beam and didn't cause complete chaos
imagesc(output3, xlab="", ylab="", col=jet.colors(16), main = "The Beam Rotated by Pi/42 Radians")
```



Finding the Total Dosage at Each Point

Now that we have calculated how much radiation each beam deposits at each point, we can take the sum over the beams to calculate the total amount of radiation received at each point.

```
#Let's make a data frame of the total dose received at each  point
TotalDose <- matrix(ncol=iterationsy, nrow=iterationsx)

#And now we can populate that grid with the sum of the doses we found above
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    TotalDose[i,j] <- output1[i,j]+output2[i,j]+output3[i,j]
        }
}
#make sure that the matrix was populated
imagesc(TotalDose, xlab="", ylab="", col=jet.colors(16), main = "Total Radiation")
```

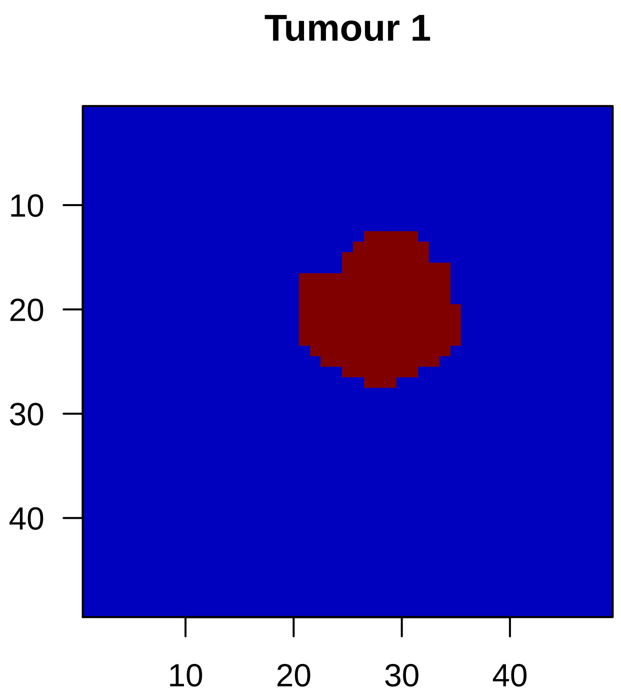


This image shows the total dosage deposited within the region. Recall that this heat map may not be accurate at points outside a circle of radius 24.5 since we canculated it by rotating a square of width 49.

Time to Introduce a Tumour

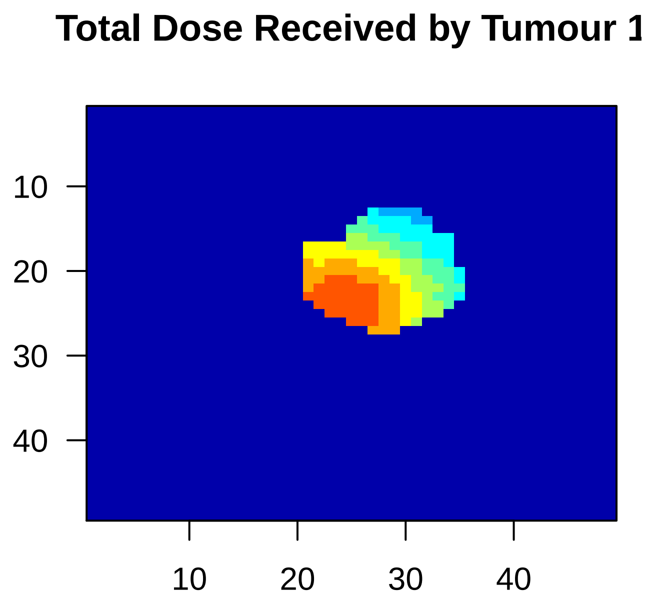
This is all fine and well but we're supposed to be calculating the amount a tumour survives and so far there's no tumour! Let's fix that and load a tumour. I generated this tumour by populating a 49 $\times$ 49 grid of zeros in Excel with the value "1" in every cell that I decided would be part of the tumour.

```
#we need a tumour, so let's load some tumour data
Tumour1 <- read.csv(paste(inputlocation, data.files[3], sep=""), stringsAsFactors = FALSE )
#Let's see what the data looks like
imagesc(as.matrix(Tumour1), xlab="", ylab="", col=jet.colors(16), main = "Tumour 1")
```



Looks like we successfully loaded the tumour, time to calculate the dose received at each point.

```r
#Let's make a data frame of the total dose received at each point in the tumour
#again, we start with a grid
TumourDose1 <- matrix(ncol=iterationsy, nrow=iterationsx)

#And we populate that grid
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    TumourDose1[i,j] <- TotalDose[i,j]*Tumour1[i,j]
  }
}

#take a peek
imagesc(TumourDose1, xlab="", ylab="", col=jet.colors(10), main = "Total Dose Received by Tumour 1")
```



Here we see how much radiation each point in the tumour would hypothetically receive if it underwent treatment with this beam configuration. But we aren't done yet! We don't just want to know how much radiation the tumour gets, we want to know how much of it survives.

Calculating the Survival Rate at Each Point in the Tumour

This is where the Linear Quadratic model defined by McMahon comes in (McMahon, 2018).The model states,

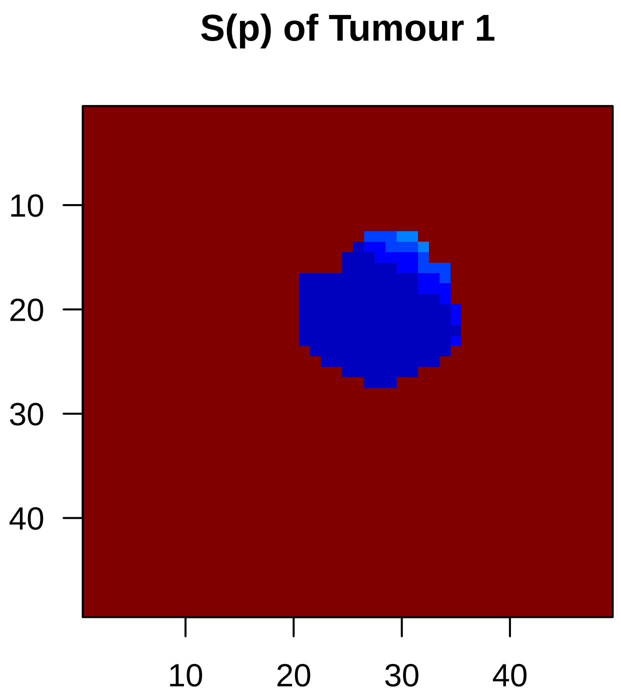
$$S(p) = e^{-\alpha D(p) - \beta D(p)^2},$$

where $D(p)$ is the dose of radiation received at point p. This model has parameters α and β so let's set those based on the findings in the literature review by van Leeuwen (van Leeuwen, 2018).

```
#First we set alpha and beta
a = 0.01
b = 0.0025
```

Now let's calculate the $S(p)$ value for each point p in the tumour.

```
#Next we need a data frame of the cell survival
CellSurvival1 <- matrix(ncol=iterationsy, nrow=iterationsx)

for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    CellSurvival1[i,j] <- exp(-(a*(TumourDose1[i,j])+b*(TumourDose1[i,j])^2))
  }
}
#and let's just see how this looks
imagesc(CellSurvival1, xlab="", ylab="", col=jet.colors(16), main = "S(p) of Tumour 1")
```

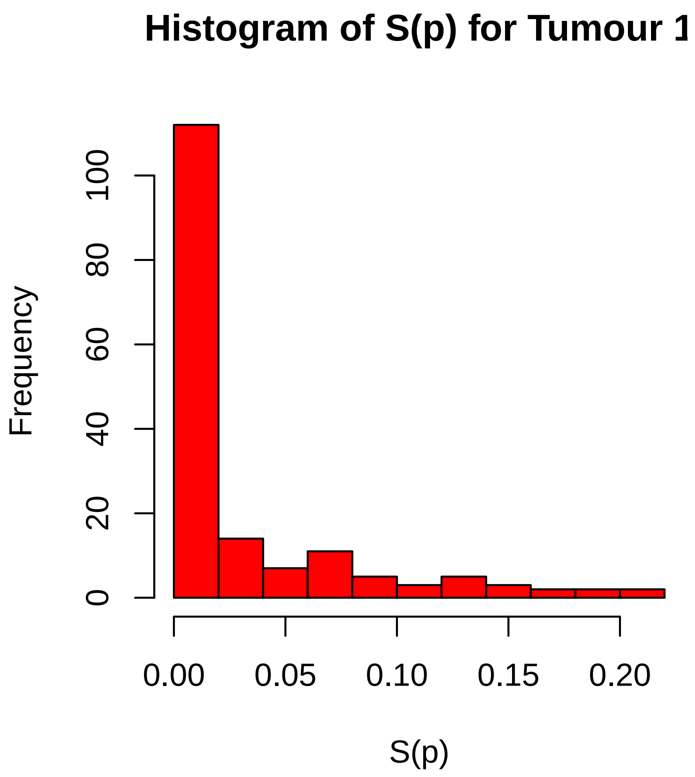


As expected this heat map looks quite similar to the heat map of the radiation of the tumour, but with the colours inverted. This makes sense because the points receiving the highest dosage have the lowest survival rate and vice versa.

Quantifying How Much of the Tumour Survives

At this pont we know the survival rate $S(p)$ for each gridcell, but how do we determine how much of the tumour survives? It's a bit tricky to see with a heat map because there is no scale, so let's make a histogram to get a sense of how many points in the tumour have any given $S(p)$ value.

```r
#First we need to get rid of all the zeros everywhere outside the tumour
#because that will throw the histogram off
HowZapped1 <- matrix(ncol=iterationsy, nrow=iterationsx)
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    if(Tumour1[i,j] == 1){
      HowZapped1[i,j] <- CellSurvival1[i,j]
    } else {
      HowZapped1[i,j] <- NA}
  }
}
HowZapped1 <- data.frame(HowZapped1)

#now we can safely create a histogram
hist(unlist(HowZapped1), main = "Histogram of S(p) for Tumour 1", xlab = "S(p)", col = "red")
```



The spike at the low end of this histogram shows that many of the points in the tumour hve a very low chance of survival, which is good news. At the other extreme end of the histogram we see that only a small number of points have a survival rate of over 20%.

Now that we have calculated $S(p)$, we can calculate SR, the metric we have chosen for the amount of the tumour that survives calculated by,

$$SR = \frac{1}{|\Omega|} \sum_{p \in \Omega} S(p),$$

where Ω is the size of the tumour, in this case, the number of gridcells in the tumour (McMahon, 2018). Let us begin by finding $\sum_{p\in\Omega} S(p)$.

```
#to count the S(p) value at each point, we can use another for loop that adds the S(p) value
#of the ijth cell in the grid for every gridcell in the tumour
sumSp1 = 0
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    if(Tumour1[i,j] == 1){
      sumSp1 <- sumSp1+CellSurvival1[i,j]
    }
    }
}

sumSp1
```

```
## [1] 4.985355
```

Now that we have found $\sum_{p\in\Omega} S(p) \approx 4.9854$, we need to calculate Ω to get SR.

```
#we can quantify the size of the original tumour by counting its cells
totcells1 = 0
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    if(Tumour1[i,j] == 1){
      totcells1 <- totcells1+1
    }
  }
}
#check the number
totcells1
```

```
## [1] 166
```

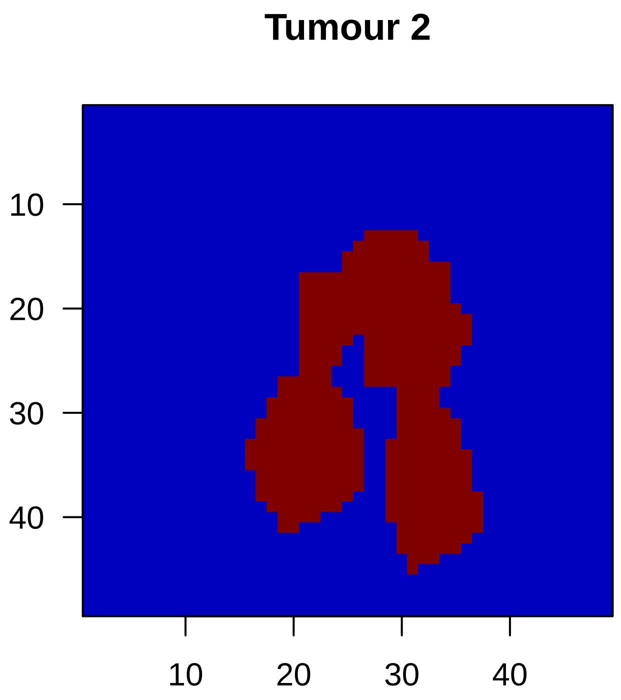
Now we have enough information to calculate SR.

```
SR1 = sumSp1/totcells1
SR1
```

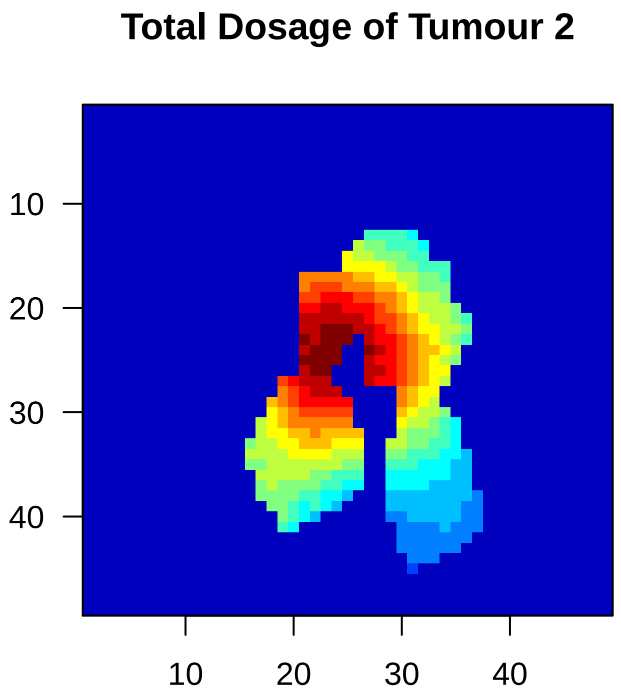
```
## [1] 0.03003226
```

Thus, $SR \approx 0.030032$, meaning that on average any given point in the tumour has about a 3.0032 % chance of surviving the treatment. How does this compare to other tumours? We can replicate this code for another one to find out.

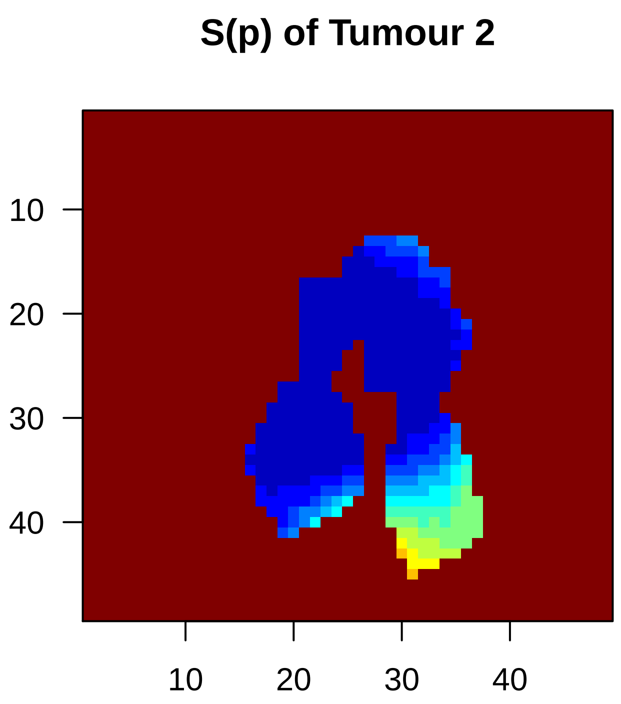
```
#To do so we need a new tumour
Tumour2 <- read.csv(paste(inputlocation, data.files[4], sep=""), stringsAsFactors = FALSE )
#Let's see what the data looks like
imagesc(as.matrix(Tumour2), xlab="", ylab="", col=jet.colors(16), main = "Tumour 2")
```



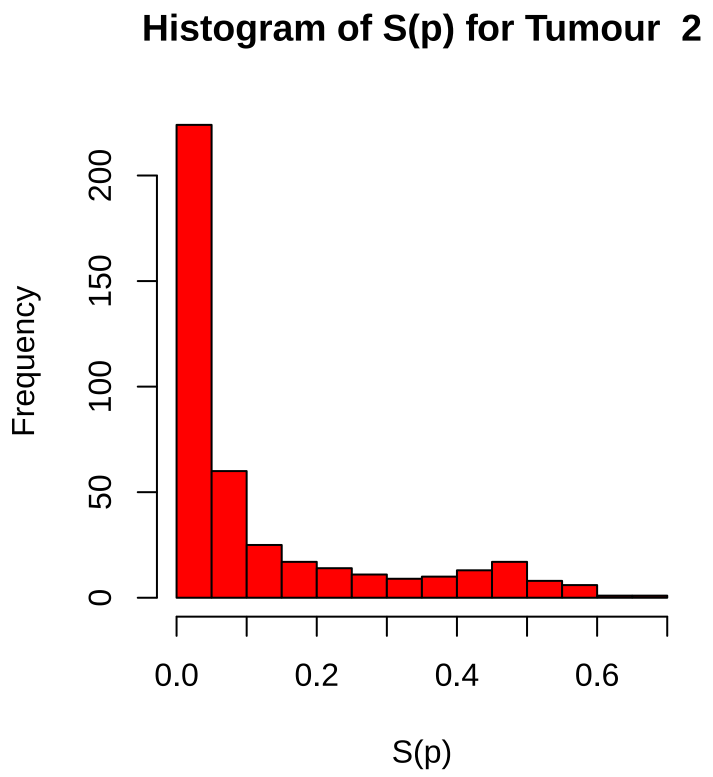
```
#looks good

#Let's make a data frame of the total dose received at each point in the tumour
TumourDose2 <- matrix(ncol=iterationsy, nrow=iterationsx)

#And now we can populate that grid
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    TumourDose2[i,j] <- TotalDose[i,j]*Tumour2[i,j]
  }
}
#check it out
imagesc(TumourDose2, xlab="", ylab="", col=jet.colors(16), main = "Total Dosage of Tumour 2")
```



```
#Let's make a data frame of the cell survival
CellSurvival2 <- matrix(ncol=iterationsy, nrow=iterationsx)

#And now we can populate that grid
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    CellSurvival2[i,j] <- exp(-(a*(TumourDose2[i,j])+b*(TumourDose2[i,j])^2))
  }
}

#visualize
imagesc(CellSurvival2, xlab="", ylab="", col=jet.colors(16), main = "S(p) of Tumour 2")
```



```r
#Now let's make a histogram' to visualize how zapped the tumour was
#Get rid of unecessary the zeros
HowZapped2 <- matrix(ncol=iterationsy, nrow=iterationsx)
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    if(TumourDose2[i,j]>1){
      HowZapped2[i,j] <- CellSurvival2[i,j]
    } else {
      HowZapped2[i,j] <- NA}
  }
}
HowZapped2 <- data.frame(HowZapped2)

hist(unlist(HowZapped2), main = "Histogram of S(p) for Tumour  2", xlab = "S(p)", col = "red")
```



```
#to count the sum of S(p) over tumour 2
sumSp2 = 0
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    if(Tumour1[i,j] == 1){
      sumSp2 <- sumSp2+CellSurvival2[i,j]
    }
    }
}
#let's see how many we got!
sumSp2
```

```
## [1] 11.98496
```

```
#now let's calculate the percentage of the tumour that survived
#to do so we need  to quantify the size  of the original tumour
totcells2 = 0
for(i in 1:iterationsx){
  for(j in 1:iterationsy){
    if(Tumour2[i,j] == 1){
      totcells2 <- totcells2+1
    }
  }
}
totcells2
```

```
## [1] 416
```

```
SR2 = sumSp2/totcells2
SR2
```

```
## [1] 0.02881
```

In this case, the expected survival rate at any given point in the tumour iss about 2.8810 %. Surprisingly, despide haveing some parts of the second tumour receive lower doses than any part of the first tumour received, the SR value for Tumour 2 is lower than that of Tumour 1 indicating that, by this metric, it has a lower survival rate. Thus, we caution the reader that although SR may be a convenient general metric it may be misleading for tumours where some parts are over-radiated and others are under-radiated.

Works Cited

R. Barnard, M. Frank, M. Herty, "Optimal radiotherapy treatment planning using minimum entropy models," Applied Mathematics and Computation, 2012

C.M. van Leeuwen, A.L. Oei, J. Crezee, *et al.* The alfa and beta of tumours: a review of parameters of the linear-quadratic model, derived from clinical radiotherapy studies. *Radiat Oncol* **13,** 96 (2018). https://doi.org/10.1186/s13014-018-1040-z

S. J. McMahon, "The linear quadratic model: usage, interpretation and challenges," Physics in Medicine & Biology, (2018). https://iopscience.iop.org/article/10.1088/1361-6560/aaf26a

"The Science Behind Radiation Therapy," American Cancer Society, (2014). https://www.cancer.org/content/dam/CRC/PDF/P