Appendix X – Full code explanation

In this appendix, the code will be introduced in commented code snippets with additional explanation. The code will be presented mostly in order as in the actual ran code. This document provides extra explanation on made choices that aren't in the code comments.

The R code is created in one file with different headers visible in the RStudio outline. Various parts of the code were finetuned in smaller code documents, available on this project's GitHub repository.

Contents

Libraries

```
# Intro ---
## Working Version for testing
## V1.47


# Libraries ----

#workdir
setwd('C:/Users/gielv/OneDrive/UT/IEM/001 AFSTUDEREN/RProjects/ThesisModel')
  # ^home
#setwd('Z:/UT/IEM/001 AFSTUDEREN/RProjects/ThesisModel')
  # ^work

rm(list=ls())
gc()

library(simmer)
library(simmer.plot)
library(simmer.bricks)

library(fitdistrplus)
library(dplyr)
library(pracma)
library(mvtnorm)
library(gridExtra)
set.seed(111)
```

In the 'Libraries' section, I keep track of the version, I set the working directory to that of my home pc or my work pc, and clear the console and memory of my pc. Then, the various libraries are loaded.

'Simmer', 'simmer.plot', and 'simmer.bricks' are the main libraries used for the simulation. They contain the building blocks to create trajectories and objects that move through them. These libraries are incredibly helpful for creating a discrete event simulation in R (Ucar et al., 2018).

'Fitdistrplus' is used to fit distributions to datasets. 'dplyr' is a data manipulation toolbox, mainly used for working with dataframes. 'Pracma' is used for more practical numerical math routines and brings more advanced functions for numerical analysis. 'mvtnorm' computes multivariate normal and t probabilities, quantiles, random deviates, and densities for statistical applications. 'ggplot2' is the main library for creating plots, and 'gridExtra' helps with the compact display of these plots.

An attempt was made to keep the amount of libraries to a minimum, to save on application weight and runtime.

'Set.seed' sets the seed of the simulation, so that randomly drawn variables may be reproduced over multiple simulation runs.

Strategy Selector

```
# STRATEGY SELECTOR ----

# here you can select the screening strategy for this simulation run.
Start_screen_age <- 50
end_screen_age <- 75
screen_interval <- 2

manualScreenInput <- F
```

```
manualScreenAges <- c(50,51,52,53,54,55)
if (manualScreenInput==1){
  # enter ages at which patient should be screened here:
  ScreenAges <- manualScreenAges
}
```

The strategy selector is placed at the start of the simulation to create an intuitive overview for the user to determine which screening strategy is to be evaluated. The default is bi-annnual screening between ages 50 and 75, which is the case in the Netherlands today. Starting and ending ages as well as the screening interval can be changed, and the simulation will then test a different strategy. If another strategy should be tested, the strategy can be inputted manually by changing the manualScreenAges vector to include the ages at which patients should be screened. The manualScreenInput parameter should then also be changed to True.

Functions

```
# Functions ----
```

The header functions is split up in different categories, highlight functions that are related to those attributtes.

Cancer

```
## Cancer ----
```

All functions related to cancer are described under this header.

```
LifeExpAndCancerAge <- function(){
  # Generate one random value from the copula
  random_cancer <- 2
  random_life_exp <- 1
  if (runif(1)<(0.14)){ # Cancer incidence percentage
    while (random_life_exp<random_cancer){
      random_sample <- rmvnorm(1, mean = c(0, 0), sigma = correlation_matrix)

      # Transform copula data to distribution data
      random_life_exp <- qweibull(pnorm(random_sample[1,1]),
                                  shape = LifeExpShape,
                                  scale = LifeExpScale) *year
      random_cancer <- qnorm(pnorm(random_sample[1,2]),
                             mean = CancerExpMean,
                             sd = CancerExpSD)  *year
    }
  } else {
    random_life_exp <- rweibull(1,LifeExpShape,LifeExpScale) *year
    random_cancer <- 1000*year
  }
  set <- c(random_life_exp,random_cancer)
  return(set)
}
```

LifeExpAndCancerAge is a function that determines the patient's healthy life expectancy and age at which she will get cancer. These values are generated through the copula that was derived in chapter 4 of the main thesis. Care is taken that the healthy life expectancy is always higher than the age of cancer onset, if the generated patient is destined to get cancer. Otherwise, the cancer incidence age is set to an unrealistically high number of a 1000 years. The healthy life expectancy is

taken from a Weibull distribution with variable parameters, the cancer incidence is taken from a normal distribution. Variables in these distributions are explained under parameters.

```
GompGrow <- function(age,onsetage,GR,IsCured){
  if(IsCured ==1){return(0)} else{
    t <- (age-onsetage ) / month
    if (t<0){
      size<-0
    } else {
      #Determine size of tumour MANC
      Volume <- Vm/(1+((Vm/Vc)^0.25-1)*exp(-0.25*GR*t))^4
      #tumour volume at time t
      size <- 2*(Volume/(4/3*pi))^(1/3)
    }
    return(size)
  }
}
```

GompGrow is a simpler version of function GompGrow2, which contains only the function to determine the size of a tumor at a given time. This function resembles the growth of a non-recessive, non-stagnating tumor as described in chapter 4.

```
FindTimeAtSize <- function(size, onsetage, GR, max_size) {

  # Solve for the time delta using a binary search algorithm
  low <- 0
  high <- 1000*year  # set an arbitrarily high upper bound for time delta
  while (high - low > 1e-6) {
    mid <- (low + high) / 2
    #size_mid <- GompGrow(onsetage + mid, onsetage, GR, IsCured=0)
    Volume_mid <- Vm/(1+((Vm/Vc)^0.25-1)*exp(-0.25*GR*mid))^4
    #tumour volume at time t
    size_mid <- 2*(Volume_mid/(4/3*pi))^(1/3)
    if (size_mid < size) {
      low <- mid
    } else {
      high <- mid
    }
  }

  # Return the estimated time when the given size was reached
  return(onsetage + high)
}
```

This function uses the GompGrow function to estimate the tumor size at different time points, and performs a binary search to find the exact time at which the given size was used. The algorithm starts with a lower bound of 0 and an arbitrarily high upper bound. It iteratively halves the search range until the found time converges to the time point at which the desired tumor size is found. This function is used in GompGrow2 below to determine the time at which the tumor started regressing, so that the new size of the tumor at this new time point can be calculated.

```
GompGrow2 <- function(age,onsetage,GR,IsCured,
                      max_size,RegressionSize,StagnateSize){
  age = age/month
  onsetage=onsetage/month
  if (IsCured ==1){return(0)}
  Vm = (4/3)*pi*(max_size/2)**3
  size <- 0
```

```
  delta = age-onsetage
  if (delta<0){
    size <-0
    return (size)
  }
  #Determine size of tumour
  Volume <- Vm/(1+((Vm/Vc)^0.25-1)*exp(-0.25*GR*delta))^4
  #tumour volume at time t
  size <- 2*(Volume/(4/3*pi))^(1/3)
  if (size>StagnateSize){
    size <- StagnateSize
  }
  if (size>RegressionSize){
    size <- RegressionSize
    time_reached = FindTimeAtSize(RegressionSize,onsetage,GR,max_size)
    delta2 = delta-time_reached
    size2 = size - (size*pnorm(delta2,50,25))
    return(size2)
  }
  return(size)
}
```

GompGrow2 takes in a patient's age, cancer onset age, tumor growth rate, whether the patient is cured, the maximum size the tumor can reach, the size at which it will regress and the size at which it will stagnate to get an accurate size for the patient's tumor at that age. For this, it first checks to see if the patient is cured. If the patient is cured, the returned size will be zero. Otherwise, the max volume of the tumor will be determined and the time delta between onset and current age is calculated. If the delta is negative, the size of the tumor will be set to zero.

The volume of the tumor is determined according to the Gompertz equation explained in chapter four, and the diameter is then calculated. If the size is larger than the patients predetermined stagnation size, that size will be returned instead. If the size is larger than the patients regressionsize, the size will first be reset to this regression size, then the FindTimeAtSize function is used to search for the time at which this regression started, and the delta between regression onset and now is calculated. The actual size now is calculated by multiplying the original size with the probability from a normal distribution at that time and detracting that from the original size.

This function is now always able to return a tumor size for a patient at a given point in time.

```
RegStagSizes <- function(){
  p <- runif(1)
  RS = 1000*year
  SS=1000*year
  if (p < 0.05){
    RS = rnorm(1,25,3)

  } else if (p<0.10){
    SS = rnorm(1,25,3)

  }

  return(c(RS,SS))
}
```

This function determines the sizes at which tumor growth will stagnate or start spontaneous regression. The probability that one of these sizes is given to a particular patient, is five percent for each. The sizes around which this happens is drawn from a normal distribution with a mean of

25mm and a standard deviation of 3. These values are taken from Cisnet model profiles (National Cancer Institute, 2022; Trentham-Dietz et al., 2021).

```
Staging <- function(size, p) {
  size_bin <- SizeTable[SizeTable$TumBinL <= size & SizeTable$TumBinR > size,]
  if (nrow(size_bin) == 0) {
    return(0)
  }
  cumulative_p <- cumsum(size_bin[,-c(1,2)])
  col_index <- which(cumulative_p >= p)[1]
  if (col_index == 1){
    return(0.5)
  } else if(col_index==2){
    return(1)
  } else if(col_index==3){
    return(2)
  } else if(col_index==4){
    return(3)
  } else if(col_index==5){
    return(4)
  }
  #return(colnames(SizeTable)[col_index + 2])
}
```

This function determines the stage of a tumor given the size and a probability. The probability is taken from a patient attribute, so that multiple diagnoses at the same size from the same patient will deliver the same result. The size of the tumor is compared to the stage-by-size matrix, which is explained in chapter 4. From this matrix, the stage of the tumor can be read and returned so that the patient is diagnosed.

```
CancerSurvival <- function(stage,lifeExp,currAge){
  p = runif(1)
  if (stage == 0.5){
    NLE = lifeExp #NLE= new life exp
  } else if (stage == 4) {
    NLE = tail(which(SurvivalDf$Stage_IV >= p * 100) - 1,1)*year + currAge
  } else if (stage == 3) {
    NLE = tail(which(SurvivalDf$Stage_III >= p * 100) - 1,1)*year+ currAge
  } else if (stage == 2) {
    NLE = tail(which(SurvivalDf$Stage_II >= p * 100) - 1,1)*year+ currAge
  } else if (stage == 1) {
    NLE = tail(which(SurvivalDf$Stage_I >= p * 100) - 1,1)*year+ currAge
  } else{NLE=lifeExp}

  if (NLE  > lifeExp){NLE <- lifeExp}


  return(NLE)
}
```

The CancerSurvival function takes in the stage of a patient's tumor, healthy life expectancy and current age. It then returns the New Life Expectancy based on the survival curve for different stages of breast cancer. This approach was taken instead of modelling complete treatments strategies, and nets the same result without taking up too much simulation resources. Once a tumor is diagnosed, the path for treatment and survival is the same, regardless of diagnosis through screening or clinical discovery.

```
ScreenResult <- function(TumorSize){
  min_dia <- rweibull(1,MeanMinDetSize,SDMinDetSize) # from miscan paper
  # define BIRADS based on how much above/below this size
  if (TumorSize ==0){
    birads = ifelse(runif(1)<FP_Perc,3,1) #False Positive
  } else if (TumorSize<0.95*min_dia){
    birads=1
  } else if (TumorSize>0.95*min_dia & TumorSize<1.25*min_dia){
    birads = 3
  } else if (TumorSize>1.25*min_dia & TumorSize<2*min_dia){
    birads = 4
  } else if (TumorSize>2*min_dia){
    birads = 5
  }
  return(birads)
}
```

The function screenresult determines a BIRADS value given the size of the tumor. This is used for determining the result of the screening procedure. There is a probability for false positives built in, which can be changed under the parameters settings. If the tumor is of a given size, then the screening will detect it based on a minimal diameter determined through a Weibull distribution. The BIRADS scores of 3,4 or 5 will send a patient to diagnostics, where the actual stage will be determined.

```
ClinicalCheckAtHome <- function(TumorSize,ClinSizeCheck,PatientAge){
  if (PatientAge %in% ScreenAges){return(0)}
  if ((PatientAge/month) %% 2 == 0){ #round(runif(1,1,2))
    if (TumorSize>ClinSizeCheck){
      return(1)
    } else {return(0)}
  }else {return(0)}
}
```

A clinical check at home will take place once every one or two months. Women are currently advised to check their breasts at least monthly themselves, but are not expected to fully adhere to this. Therefore, a slightly lower occurrence of clinical checking was chosen. If patients are also invited for screening at this time, then no clinical check will take place. To determine if the clinical check nets a positive result, the size of the tumor is compared to the minimal size needed for clinical diagnosis, which is different for each patient. If a tumor has been detected clinically, the patient will be referred to diagnostics.

```
GrowRater <- function(startAge){
  GRmultiplier <- 0
  GR <- rgamma(1,shape = Grow_Gamma_shape, rate = Grow_Gamma_rate)
  # print(GR)
  startAge <- startAge/year
  if (startAge <50){
    GRmultiplier = -(62-startAge)/80
  }
  if (startAge >75){
    GRmultiplier <- -(startAge-62)/80
  }
  GR <- GR + GRmultiplier
  if (GR<0){
    GR <- GR - GRmultiplier
  }
```

```
  return(GR)
}
```

The GrowRater function will slightly adjust tumor growth rates based on patient's ages. Tumors generally grow slower or faster at different ages, and these values were found through iterative finetuning in order to recreate the graph of tumor stages found at different ages as seen on the IKNL website (Integraal Kankercentrum Nederland, 2022).

Time

```
## Time ----
```

Time functions are used to determine how long a patient will remain in a certain trajectory before going to the next step.

```
WaitAtHome <- function(EntryAge){
  # find next month
  div <- EntryAge %% month
  return(month - div)
}
```

WaitAtHome takes in the patient's age at entry and creates a timeout until the next month when the next steps are determined.

```
ScreenTime <- function(){
  return(max(0.1,rnorm(1,MeanScreenTime,SDScreenTime)))
}
DiagTime <- function(){
  return(max(0.1,rnorm(1,MeanDiagTime,SDDiagTime)))
}
```

ScreenTime and DiagTime return a duration for how long patients should remain in that trajectory. They are drawn from normal distributions with average durations and standard deviations. A minimum is also built in to prevent returning negative durations.

Next Steps

The function AfterHome determines the next step for the patient: what trajectory should they follow after exiting this trajectory.

```
## Next Steps ----
AfterHome <- function(Invite,Screened,Birads,ClinicalCheck,
                      Referral,PatientAge,HealthyLifeExpectancy,TumorSize){
  # Next Steps update
  #All five steps possible:
  #a. Go to screening after invite-> 2
  #b. Go to diagnostics due to tumor size or referral after screening ->3
  #c. Go to hospital due to referral -> 4
  #d. Die due to age or illness-> 5
  #e. Continue in Home trajectory, nothing's needed -> 1
  #PatientAge <- PatientAge / year
  s <- 0
  if (Invite>0){
    if (PatientAge %in% ScreenAges){ # check if go for screen
      if (Invite==1){
        upt = uptakefirstscreen
      }else if (Invite > 1){
        upt = uptakeotherscreen
      } else {upt = uptakenoscreen}
      if (upt > runif(1)){
        s <- 2
```

```
      }
    }
  }

  if (ClinicalCheck ==1 | Birads == 3 | Birads == 4 | Birads ==5){
    # check if diagnsostics
    s <- 3
  }

  if (Referral ==1){
    s <-4
  }

  if (PatientAge > HealthyLifeExpectancy){
    s <- 5 # die
  }
  if (s==0){s<-1}# stay at home
  return(s)
}
```

The function determines if a patient should go to screening, diagnostics, hospital, death or stay at home as the next step. In order to determine this, it takes in the patient parameters Invite, Screened, Birads, ClinicalCheck, Referral, PatientAge, HealthyLifeExpectancy, and TumorSize.

First, the patient is checked to see if she should go to a screening test. If a patient should go to screening, she should have an invite and the patient's age should be in the vector containing the ages at which patients are screened. Then, the probability at which the patient actually adheres to the invitation is determined. Not all patients go for screening, with around 75% going to their first screening, and 90% going to screenings after their first. If no screening has been done but more than one invite has been received, then the probability of going to the screening drops to 25%. These probabilities are checked against a random value from a uniform distribution, and then the patient might get a NextStep value of 2, meaning she will visit the screening trajectory next.

If the patient has found a tumor clinically or through screening, based on the parameters ClinicalCheck and Birads, the patient will go to diagnostics next.

If after diagnostics the patient is referred to the hospital, the patient will go to the hospital as their next step.

If a patient is older than their healthy life expectancy, they should die in the next step.

If none of the above conditions are true, then the patient will remain in the home trajectory for another month.

Utilities

In the Utilities section, two functions for determining a patient's current utility are presented.

```
## Utilities ----
BaseAgeUtil <- function(Age){
  Age <- Age/year
  if (Age < 31){
    return(1)
  }
  Age <- ceiling((Age-30)/5) # steps of 5
  return(utility_ages[Age,2])
}
```

The base utility of a patient is dependent on age. At higher ages, a higher base utility is used. This function looks up a patient's base utility in the utility_ages table and returns this.

```
CancerBasedUtil <- function(currentUtil,stage,age,startAge){
  if (stage==0){
    s=1
  } else if (stage==0.5){
    s=2
  } else if (stage==1){
    s=3
  } else if (stage==2){
    s=4
  } else if (stage ==3){
    s=5
  } else if (stage==4){
    s=6
  }
  delta <- (age-startAge) / year
  utilDec <- utility_decrements[s,2]
  discUtilDec <- utilDec/((1+UtilDiscount)^floor(delta))
  newUtil = currentUtil * (1-discUtilDec)
  return(newUtil)
}
```

If a patient has cancer, the utility or quality of life of that patient is lower. This function takes in the current utility, often the base utility, the stage of the cancer, the patient's age and the age at which the tumor started. First, it determines the index of which utility decrement to look up based on the cancer's stage. Then the time delta between tumor onset and patient age is determined. After that, the new utility decrement is taken from the utility_decrements table.

Based on the time difference between tumor onset and current age of the patient, the actual value of the utility decrement is found through discounting. A 0.10 discount in utility now should only be a decrement of 0.086 in ten years, as utility is discounted at 1.5%. The new utility is found by multiplying the base utility by 1 minus the decrement. This new utility is then returned as patient parameter.

Costs

```
## Costs----
```

Under Costs, various functions for determining costs of pre-medical procedures, societal occurrences, end-of-life costs and cost discounting are given.

```
ScreenDiagInvitesCost <- function(screened,diaged,invite){
  if (is.na(screened)){screened=0}
  if (is.na(diaged)){diaged=0}
  if (is.na(invite)){invite=0}
  cost<- 0
  cost<- cost + screened* AvgScreenCost
  cost<- cost + diaged*AvgDiagCost
  cost<- cost + invite*AvgInvCost
  return(cost)
}
```

This function determines a patient's costs associated with the number of screenings she's received, how often she has been through diagnosis and how often she has received an invite. It is calculated at the end of a patient's life based on the attributes screened, diagnosed, and invites.

```
TreatmentCost <- function(stage,age,lifeExp){
  # cost is determined by:
    # initial cost for first 12 months post treatment
      # ^always counted
    # costs per month after the first 12, up to the last 6 months
      # counted if this interval exists
    # Costs for the last six months of care
      # ^always counted
  delta <- (lifeExp - age) / month
  #delta is how long we have left
  c<-0 # cost variable
  #determine right row from table:
  if (stage==0.5){
    row <- 2
  } else if (stage ==0){
    row <- 1
  } else {
    row<- stage+2
  }
  InitC <- rnorm(1,TreatCostTable$Initial12Cost[row],
                 TreatCostTable$InitialSD[row]) /12 #round to monthly
  TerminalC <- rnorm(1,TreatCostTable$Terminal6Costs[row],
                     TreatCostTable$TerminalSD[row]) /6 # to monthly
  MidC <- rnorm(1,TreatCostTable$ContinuousMonthlyCost[row],
                TreatCostTable$ContinuousSD[row]) /2

  CInit <- min(delta,12) * InitC #no discount, this is year 0
  CTerminal <- min(delta,6) * TerminalC /
    (1 + CostDiscount)^((round((lifeExp - age)/year)) - 1)
  # discount with 4% per year post diagnosis
  CMid <- annuity_cost_monthly(MidC,delta,CostDiscount)

  c <- CInit + CTerminal + CMid
  return(c)

}
```

Treatment costs is the function that determines the costs of treating a patient's cancer. These costs are based on cancer stage, age and the life expectancy of the patient. The costs are determined in three steps: first the initial costs for the first year of treatment is calculated, then the costs for the final six months of a patient are retrieved and discounted, and then the costs for the months in between initial and final treatment are determined.

The initial costs are taken from the TreatCostTable based on the patient's stage. The costs are then divided by twelve to create monthly costs. Based on the time the patient has left to live, found by creating the delta variable by subtracting the patient's current age from her life expectancy, the costs for this first year are determined. These costs are not discounted, as time of diagnosis is t_0 in the simulation.

The final costs are taken by multiplying 6 or less months by the discounted costs of final treatment. If a patient has less than six months to live at time of diagnosis, not all six months are counted. The costs are discounted at 4% yearly to the time the patient will die in the future. If a patient still has 25 years left to live and the final costs of treatment would be 10000 euros today, the discounted value would be 3750 euros.

For the costs of treatment for all the months between the initial 12 and final 6 months of a diagnosed patients life, the function annuity_cost_monthly was created to sum the discounted

monthly treatment costs. At the end of the function, the three costs variables are summed up and returned to the attribute.

```
VisitationCosts<- function(screened,diaged,hosps){
  costs<- 0
  hosps <- hosps/day #time to days

  #travel costs
  costs <- costs +
    (ifelse(runif(1)<perc_car,1,0) * dist_to_screen * cost_per_km) * screened
  # for screening visits
  costs <- costs +
    (ifelse(runif(1)<perc_car,1,0) * dist_to_hosp * cost_per_km) * diaged
  # for diagnosis in hospital
  costs <- costs +
    (ifelse(runif(1)<perc_car,1,0) * dist_to_hosp * cost_per_km) * hosps

  # productivity loss costs:
    # screening & diag takes average time, hospital takes full working day
  costs <- costs + (screened*MeanScreenTime*prod_cost_hour) +
    (diaged*MeanDiagTime*prod_cost_hour) +
    (hosps*8*hour*prod_cost_hour)

  return(costs)
}
```

The function VisitationCosts determines the societal costs now mandated by the Dutch guidelines on health economic analyses. The costs consists of travel costs to screening, hospital, and diagnosis and are based on standard values from the guidelines (Zorginstituut Nederland, 2016). The costs of productivity loss are also calculated based on the total time a patient has spent in diagnostics, screening and hospital. This time in hours is multiplied with the guideline-given standard value for productivity loss costs per hour. The total societal costs is then returned.

```
PaidCosts <- function(lifeExp,StartAge){
  costs<- 0
  # PAID tool gives insight in expected rest of life costs for if someone
    #lives longer after cancer treatmenet.
  if (lifeExp>StartAge){
    costlist <- WomenCostList[(round(StartAge)/year):(round(lifeExp)/year)]
    yearlist <- 0:(length(costlist)-1)
    disclist<- costlist / (1+0.04)^yearlist
    costs = sum(disclist)
  }
  return(costs)
}
```

The Guidelines also give instructions to include the costs for future illnesses in the calculation of societal costs. These costs are calculated through PAID, which provides a csv file of costs related to other diseases than breast cancer at various ages through their online tool (*PAID 3.0*, n.d.). These costs are calculated for the extra years of life a patient has to live after diagnosis and treatment, and are discounted at 4% to their current value.

```
annuity_cost_monthly <- function(payment, months, rate) {
  monthly_rate <- (1 + rate)^(1/12) - 1
  total_cost <- 0
  if (months<13){
    return(0)
  }
```

```
  months_left = min(0,months-6) #don't count the final 6
  for (i in 13:months_left) {
    discounted_payment <- payment / (1 + monthly_rate)^(i - 1)
    total_cost <- total_cost + discounted_payment
  }
  return(total_cost)
}
```

The annuity_cost_monthly function is used for calculating the mid part of the medical costs. It iteratively sums the monthly discounted values of the costs of treatment post first 12 months and pre final six months, and returns the costs for this section.

Sim

```
## Sim ----
```

Under Sim, two functions are placed that are taken from the UT course Advanced Simulation for Health Economic Analysis. These function allow for the tracking of one or multiple attributes during the whole simulation, so that patients might be traced and the simulation can be checked for errors.

```
getSingleAttribute <- function(attribute,
                               output,
                               all=F) sapply(unique(output[,"name"]),
                                             function(entity) {
  list(
    if(all) {
      output[output[,"name"]==entity & output[,"key"]==attribute, "value"]
    } else {
      tail(output[output[,"name"]==entity &
                    output[,"key"]==attribute, "value"], n=1)
    }
  )
})
```

GetSingleAttribute takes one single attribute as an argument and will find all occurrences of it in the final results data frame if the monitoring of the system is set to true. Then, all these occurrences can be checked in a new data frame.

```
getMultipleAttributes <- function(attributes, output)
as.data.frame(sapply(attributes, function(attribute)
as.numeric(getSingleAttribute(attribute, output))))
```

GetMultipleAttributes applies the function getSingleAttribute multiple times and returns a dataframe containing more than one attribute over time.

Parameters

```
# Parameters ----
```

In the Parameters section, data is loaded and values are set for various variables. Most data is taken from literature and some is found through finetuning.

Patients

```
## Patients----
n.patients=1000;
mon.patients <- ifelse(n.patients<100,2,0);

LifeExpShape <- 7.937
```

```
LifeExpScale <- 86.788
```
Under Patients, the number of patients to be simulated is set. If the number is low enough, the advanced monitoring will kick in which allows for individual patient tracing through the model using the getSingle- and getMultipleAttributes functions.

The LifeExpShape and LifeExpScale parameters have been found through analysis of CBS data on dutch life expectancy, and are used in a Weibull distribution in the function LifeExpAndCancerAge.

Data Loading

```
## Data loading ----

# Transform copula data to distribution data
correlation_matrix <- matrix(c(1, 0.5, 0.5, 1), ncol = 2)
#for age & cancer incidence

SizeTable <- read.csv('StagingFromSizes.csv')
```
Under data loading, first the correlation matrix for LifeExpAndCancerAge is created. The correlation between the two is set to 50%, which netted the best results.

The SizeTabel data is loaded, which contains the data used to classify a tumor of a certain size as a certain stage. This data is discussed in chapter 4.

Time

```
## Time-----
hour = 1;
minute = hour/60;
second=minute/60;
day=hour*24;
year = day*365;
month = year/12


MeanScreenTime <- 4*hour
SDScreenTime <- 30*minute
MeanDiagTime <- 6*hour
SDDiagTime <- 30*minute
```
Various timing related variables are initiated under ‘time’. First of all, the base variables are set so to make it easier to work with simulation time. A base of 1 hour is chosen, and the variables minute, second, day, year, and month are derived from that. These variables are used throughout the simulation and in various functions to make it easier to work with time.

The variables for the mean and standard deviation for screening and diagnosis are also set, so that every patient has slightly different times in those stations. This is also an example of how the time variables of hour and minute are used in defining other time related variables.

Utilities

```
## Utilities ----
# these are base utilities per age-range,
# taken from the MANC model
utility_ages<-data.frame(c(30,35,40,45,50,
                           55,60,65,70,75,
                           80,85,90,95,100),
                         c(0.9383,0.9145,0.9069,0.8824,0.8639,
                           0.8344,0.8222,0.8072,0.8041,0.779,
                           0.7533,0.6985,0.6497,0.6497,0.6497))
```

```
utility_decrements <- data.frame(c('Healthy','DCIS','StageI',
                                   'StageII','StageIII','StageIV'),
                                 c(0,0.1,0.15,0.20,0.25,0.40))
UtilDiscount <- 0.015
```

In the utilities section, various parameters and data frames related to the quality of life of patients are initiated. First, the data frame utility_ages is created. This contains the base utility for patients of a certain age. For example, a 50-year-old woman has a base utility of 0.8639. This data is based on the Manchester model (Wright et al., 2022)

Then, the utility decrements for various stages of cancer are given. Healthy will net a decrement of 0, and stageIV a decrement of 40%. This data is based on Arrospide et al. (Arrospide et al., 2016)

Finally, the mandated 1.5% discount rate for utilities is initiated in the variable UtilDiscount (Zorginstituut Nederland, 2016).

Costs

```
## Costs ----

TreatCostTable <- read.csv('TreatmentCostsVanLuijt2023Eur.csv')
CostDiscount <- 0.04

AvgScreenCost <- 100
AvgInvCost <- 5
AvgDiagCost <- 212
```

In the costs section, the treatment costs table is loaded. This contains the data for costs of treatment per stage. It is split up in initial 12 months, mid-months, and final 6 months. The explanation of usage of this data is found under the function TreatmentCosts. The data can be seen in table **x**

Stage	Initial12Cost	InitialSD	Continuous MonthlyCost	ContinuousSD	Terminal6Costs	TerminalSD
0	0	0	0	0	0	0
0.5	9583.123	406.9691	140.3392	13.83695	23672.61	3798.378
1	14497.39	312.0097	222.954	11.39514	18138.93	1899.189
2	29104.01	651.1506	423.9329	18.0423	22905.14	1220.907
3	35756.04	2170.502	589.8203	97.80824	18846.1	2848.784
4	33628.54	3527.066	1236.087	157.0901	24758.81	2848.784

The data comes from the study of VanLuijt from 2016, and the amounts are corrected to represent 2023 euros. Reliable recent Dutch data was hard to find. A 2013 paper of de Bock, Siesling et al does contain information on costs for breast cancer treatment based on size, but the data was from 2000 and therefore deemed unrealistic, even with corrections for inflation. To make sure the data used in this paper is reliable, the factors for treatment for different stages were checked. In the 2013 paper, the costs for treatment of advanced cancer were up to five times higher than that of simple cancers. This is comparable to the data used in this paper.

Furthermore, in this section the discount rate for costs is set to 4%, which is used everywhere costs are calculated.

The average costs for screening, invites and diagnosis are also given. The costs for screening, invites and diagnosis are based on the annual report of BevolkingsonderzoekNederland and are checked with various studies from the systematic review (Arrospide et al., 2016; PriceWaterhouseCoopers Accountants N.V., 2022; Rafia et al., 2016; van Luijt et al., 2017).

Tumor

```
## Tumor ----
```

In the tumor section, all variables related to the tumor are set, including age of onset, growth parameters and survival probabilities.

```
#Set tumour growth rate parameters
CancerExpMean = 61.871
CancerExpSD = 14.14

Grow_Gamma_shape <- 1.567742
Grow_Gamma_rate <- 1.933883


max_size <- 128 #mm diameter
start_size <- 0.25 #starting size of tumours, diameter in mm
Vc = (4/3)*pi*(start_size/2)^3 #Volume at start
Vm = (4/3)*pi*(max_size/2)^3 #Max volume

#DF to show survival probability after X years at X stage
SurvivalDf <- data.frame(Years_After_Diagnosis = c(0:10),
                         Stage_I = c(100, 100, 100, 99, 99,
                                     98, 97, 97, 96, 95, 95),
                         Stage_II = c(100, 99, 97, 95, 93,
                                      91, 89, 88, 86, 85, 83),
                         Stage_III = c(100, 96, 89, 83, 78,
                                       73, 69, 65, 62, 60, 58),
                         Stage_IV = c(100, 70, 53, 38, 29,
                                      22, 17, 13, 11, 9, 7))
```

The variables CancerExpMean and CancerExpSD are the parameters used in the normal distribution to determine the age of cancer onset using the copula in the function LifeExpAndCancerAge. These values are found through fitting a distribution on the synthetic data of the IKNL (Integraal Kankercentrum Nederland (IKNL), 2022).

The grow parameters are used in the function GrowRater to determine the growth rate of the tumors. How these growth rates are determined, is explained in chapter 4.

Furthermore, the max_size, start size, start volume and max volume of the tumors are defined. These parameters are based on cisnet model profiles, and tuning of this is explained in chapter 4. Not that to determine Vc and Vm, the formula to create the volume of a sphere from a diameter is used (National Cancer Institute, 2022; Trentham-Dietz et al., 2021).

The SurvivalDf is also initiated. This contains the probabilities of survival per stage, and is used in the function CancerSurvival to determine how long someone has left to live after diagnosis.

Detection

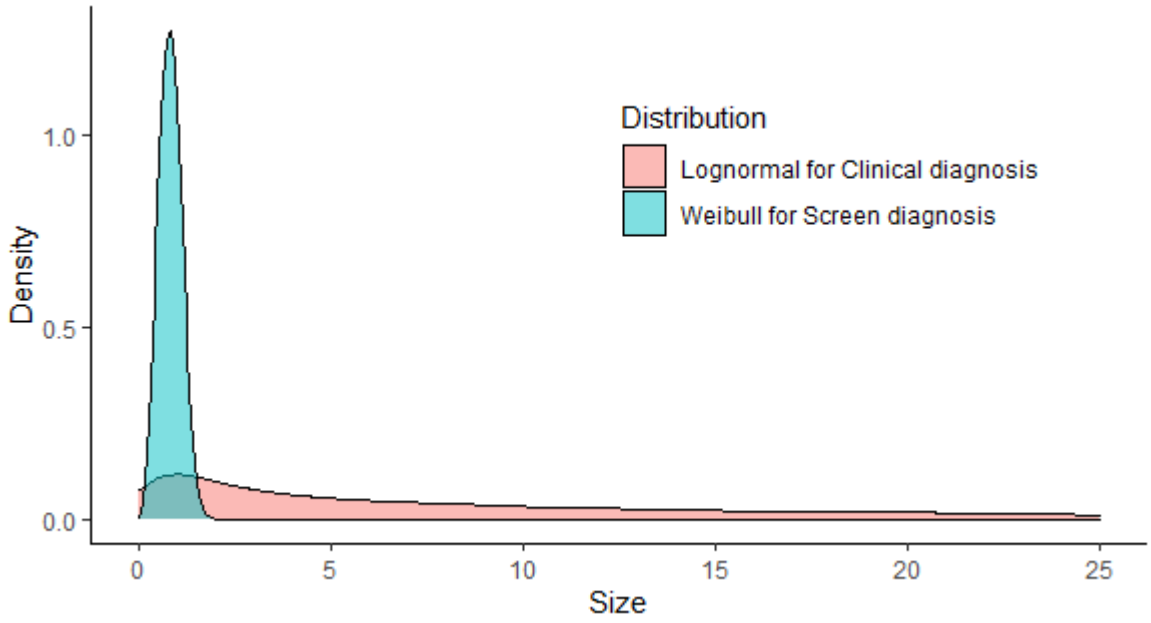
```
## Detection ----
```

Under detection, the various parameters for detection of tumors are set.

```
MeanMinDetSize <- 0.91 # for screening
SDMinDetSize <- 0.34 # for screening

MeanLogClin <-  3.22
SdLogClin <- 2.25
AlwaysFoundSize<- 100 # buld in a max! so min(100,clindiagsize)
```

```
FP_Perc <- 0.005 #FalsePositive
```

First the parameter for screen detection are set. These parameters are used in the Weibull distribution in the function ScreenResult, so the minimum diameter varies every time someone visits screening. The parameters for clinical diagnosis are higher on average, and drawn from a lognormal distribution instead of Weibull. The minimum size for clinical diagnosis is set once at the beginning of a patients life instead of having a new one drawn every check. There is a maximum size of detection of 100mm built in, as women are expected to always notice a tumor themselves if it has reached that size. The overlay of the two distributions is shown in the figure below:



Here, the false positive percentage is also set. The false positive percentage is set at 0.5%, which is lower than the currently found false positives. One thing to consider is that this percentage is taken every screen, so a lot of patients will get checked against this percentage 10 times, meaning that the cumulative probability of getting a false positive once in your lifetime is a lot higher than 0.5%, at 4.9% over 10 screenings. Furthermore, part of the false positives can also come from tumors that do show up during screening, but have spontaneously regressed by the time of diagnosis.

Screen Start Params

```
## Screen Start Params ----
```

In Screen Start Params, various parameters for the screening strategy are set.

```
Start_screen_age <- Start_screen_age *year
end_screen_age <- end_screen_age * year
screen_interval <- screen_interval*year
ScreenAges = seq(Start_screen_age,end_screen_age,by=screen_interval)
ScreenAges <- append(ScreenAges,1000*year)
if (manualScreenInput){
  # enter ages at which patient should be screened here:
  ScreenAges <- manualScreenAges *year
}
# ScreenAges = c(50,52,54,56,58,60,62,64,66,68,70,72,74)*year
```

```
#Screening uptake MANC model to Dutch
uptakefirstscreen<- 0.78#0.605
uptakeotherscreen<-0.9#0.852
uptakenoscreen<-0.25#0.191
```

First, the parameters inputted by the user under STRATEGY SELECTOR are transformed to the right time format to be used in the simulation.

Then, the parameters for the uptake of screening are set. These are used in the AfterHome function to determine if the patients actually visit screening after receiving an invite. The idea is based on the Manchester model, and the numbers are taken from the monitor bevolkingsonderzoek 2020 (van Haperen, 2018; Wright et al., 2022).

Societal Cost Attributes

```
## Societal Cost Attributes ----
```

Here, the various standard values for calculating societal costs are set. These values are based on the **guidelines**.

```
dist_to_hosp <- 7
dist_to_screen <- 1.1
avg_park_cost <- 3
perc_car <- 0.8 # most go by car, definitely for hospital visits

cost_per_km <- 0.19 # for Public transport & car

prod_cost_hour <- 31.6 # for women, Dutch guidelines

AdditionalLifeYearCosts <-
read.csv('PAID_Cost_additional_year_post_BC_Costs_Living_Year_Longer_2023-03-07.csv')
WomenCostList <- AdditionalLifeYearCosts$Unrelated_Women
```

The average distance to hospital is used for diagnosis and treatment, the average distance to screening is used for screening and based on distance to GP. Average parking costs, percentage by car, costs per km and productivity costs per hour are also taken form the guidelines (Zorginstituut Nederland, 2016). The AdditionalLifeYearCosts reads a csv file from the PAID tool discussed earlier. WomenCostList extracts the one useful column from the file. This contains the costs related to other diseases after treatment for breast cancer.

Attributes Recoding

```
# Attributes Recording ----
```

Attributes recording contains one function and one list, which is used to record the final status of all attributes of a patient at the end of her life. The function also returns a 0, to indicate the patient is now dead.

```
allAttributes = c("Index","Name","Alive","PatientAge",”HealthyLifeExpectancy",
                  "TumorStartAge", "LastAge", "TotalCosts",
                  "MedicalCosts", "SocietalCosts","PAIDCosts", "TotalUtility",
                  "CurrentUtility", "TumorSize", "WorstCancer",
                  "WorstStage", "CancerStageT", "IsCured", "StagingProb",
                  "TumorGrowthRate", "max_size",
                  "RegressionSize", "StagnateSize", "ClinSizeCheck",
                  "Invite", "Screened", "BIRADS",
```

```
                  "Referral", "NextStep", "EntryAge", "TimeAtHome",
                  "ClinicalCheck", "TimeAtDiagnostics",
                  "TotalTimeAtDiagnostics", "DiagnosedThrough",
                  "DiagnosticVisits", "TimeAtHospital", "TotalTimeAtHospital",
                  "OriginalLifeExp", "HospitalVisits", "TimeAtScreening",
                  "TotalTimeAtScreening", "FirstStage")
```

AllAttributes is a list containing all the attributes used in the simulation.

```
EndDatDf <- data.frame(matrix(rep('test',length(allAttributes)),
                              ncol=length(allAttributes)))
colnames(EndDatDf) <- allAttributes
```

EndDatDf is a data frame containing a column for each attribute, and by the end of the simulation a row for each patient. The column names are set to the names of the attributes.

```
addAtts <- function(EndDatDf){
  addList <- c(nrow(EndDatDf),
               get_name(basic_sim),
               get_attribute(basic_sim,'Alive'),
               get_attribute(basic_sim,'PatientAge'),
               get_attribute(basic_sim,'HealthyLifeExpectancy'),
               get_attribute(basic_sim,'TumorStartAge'),
               get_attribute(basic_sim,'LastAge'),
               get_attribute(basic_sim,'TotalCosts'),
               get_attribute(basic_sim,'MedicalCosts'),
               get_attribute(basic_sim,'SocietalCosts'),
               get_attribute(basic_sim,'PAIDCosts'),
               get_attribute(basic_sim,'TotalUtility'),
               get_attribute(basic_sim,'CurrentUtility'),
               get_attribute(basic_sim,'TumorSize'),
               get_attribute(basic_sim,'WorstCancer'),
               get_attribute(basic_sim,'WorstStage'),
               get_attribute(basic_sim,'CancerStageT'),
               get_attribute(basic_sim,'IsCured'),
               get_attribute(basic_sim,'StagingProb'),
               get_attribute(basic_sim,'TumorGrowthRate'),
               get_attribute(basic_sim,'max_size'),
               get_attribute(basic_sim,'RegressionSize'),
               get_attribute(basic_sim,'StagnateSize'),
               get_attribute(basic_sim,'ClinSizeCheck'),
               get_attribute(basic_sim,'Invite'),
               get_attribute(basic_sim,'Screened'),
               get_attribute(basic_sim,'BIRADS'),
               get_attribute(basic_sim,'Referral'),
               get_attribute(basic_sim,'NextStep'),
               get_attribute(basic_sim,'EntryAge'),
               get_attribute(basic_sim,'TimeAtHome'),
               get_attribute(basic_sim,'ClinicalCheck'),
               get_attribute(basic_sim,'TimeAtDiagnostics'),
               get_attribute(basic_sim,'TotalTimeAtDiagnostics'),
               get_attribute(basic_sim,'DiagnosedThrough'),
               get_attribute(basic_sim,'DiagnosticVisits'),
               get_attribute(basic_sim,'TimeAtHospital'),
               get_attribute(basic_sim,'TotalTimeAtHospital'),
               get_attribute(basic_sim,'OriginalLifeExp'),
               get_attribute(basic_sim,'HospitalVisits'),
               get_attribute(basic_sim,'TimeAtScreening'),
               get_attribute(basic_sim,'TotalTimeAtScreening'),
               get_attribute(basic_sim,'FirstStage')
```

```
  )
  EndDatDf<<- rbind(EndDatDf,addList)
  return(0)
}
```

The function addAtts takes in the EndDatDf, so that it is always working with the most recent version. Then, it collects all the patients attributes and adds this to a new vector. Using rbind and a double `<<-`, the new list is connected to the global attribute EndDatDf. A zero is returned to indicate the patient is now dead.

Simulation

```
# Simulation ----

## Trajectories ----
```

In the Simulation section, the various trajectories are discussed, the trajectories are visualized and the run settings are given.

Init

```
### Init ----
# trajectory for setting all patient-specific  starting attributes
```

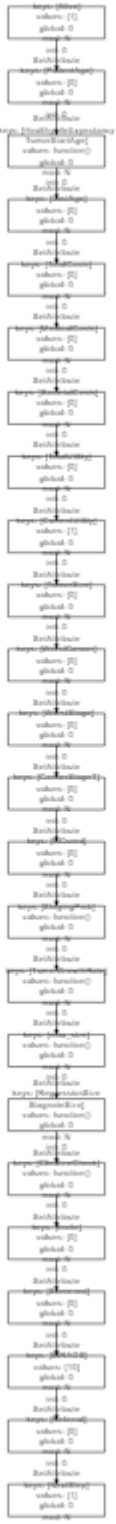
Init is the first trajectory all patients pass through. They pass through it only once, and in this trajectory some attributes are initialized at 0, other are set to their patient specific values.

```
Initialization <- trajectory()%>%
  set_attribute(key='Alive',value=1) %>%
  set_attribute(key='PatientAge',value=0) %>%
  set_attribute(key=c('HealthyLifeExpectancy','TumorStartAge'),
                value= function() LifeExpAndCancerAge())%>%
  set_attribute(key='LastAge',value=0)%>%
  set_attribute(key='TotalCosts',value=0)%>%
  set_attribute(key='MedicalCosts',value=0) %>%
  set_attribute(key='SocietalCosts',value=0) %>%
  set_attribute(key='TotalUtility',value=0)%>%
  set_attribute(key='CurrentUtility',value=1) %>%
  set_attribute(key='TumorSize',value=0) %>%
  set_attribute(key='WorstCancer',value=0) %>%
  set_attribute(key='WorstStage',value=0) %>%
  set_attribute(key='CancerStageT',value=0) %>%
  set_attribute(key='IsCured',value=0) %>%
  set_attribute(key='StagingProb',value=function() runif(1)) %>%
  set_attribute(key = 'TumorGrowthRate',
                value = function() GrowRater(get_attribute(basic_sim,
                                                           'TumorStartAge')))%>%
  set_attribute(key = 'max_size',
                value=function() rnorm(1,max_size,10)) %>%
  set_attribute(key = c('RegressionSize','StagnateSize'),
                value= function() RegStagSizes())%>%
  set_attribute(key='ClinSizeCheck',
                value=function() min(AlwaysFoundSize,
                                     rlnorm(1,MeanLogClin,SdLogClin))) %>%
  # set size needed for clinical find for this patient
  set_attribute(key = 'Invite',value=0)%>%
  set_attribute(key = 'Screened',value=0)%>%
  set_attribute(key = 'BIRADS',value=10) %>%#Invalid BIRADSvaluebefore checkup
  set_attribute(key = 'Referral',value=0) %>%# No referral
```

```
  set_attribute(key='NextStep',value=1)  # go home for first iteration
  #timeout(30*year) %>%
  #log_(paste0('Going Home'))
```

The following happens in the Init trajectory:

1. Attribute `Alive` is initialized at **1**
2. Attribute `PatientAge` is initialized at **0**
3. Random values are drawn from function `LifeExpAndCancerAge` to set the attributes `HealthyLifeExpectancy` and `TumorStartAge`.
4. Attribute `LastAge` is initialized at **0**
5. Attribute `TotalCosts` is initialized at **0**
6. Attribute `MedicalCosts` is initialized at **0**
7. Attribute `SocietalCosts` is initialized at **0**
8. Attribute `TotalUtility` is initialized at **0**
9. Attribute `CurrentUtility` is initialized at **0**
10. Attribute `TumorSize` is initialized at **0**
11. Attribute `WorstCancer` is initialized at **0**
12. Attribute `WorstStage` is initialized at **0**
13. Attribute `CancerStageT` is initialized at **0**
14. Attribute `IsCured` is initialized at **0**
15. Attribute `StagingProb` is set to a random uniform value between 0 and 1
16. Attribute `TumorGrowthRate` is drawn from the function `GrowRater`, and is based on previously set attribute `TumorStartAge.`
17. Attribute `max_size` is set, drawn from a normal distribution around previously set parameter `max_size` (128) with a standard deviation of 10.
18. Attributes `RegressionSize` and `StagnateSize` are set using previously discussed function `RegStagSizes`.
19. Attribute `ClinSizeCheck` is set using the parameters `AlwaysFoundSize` and a lognormal distribution with `MeanLogClin` and `SdLogClin`.
20. Attribute `Invite` is initialized at **0**
21. Attribute `Screened` is initialized at **0**
22. Attribute `BIRADS` is initialized at **10**, to make sure it is invalid at the first check.
23. Attribute `Referral` is initialized at **0**
24. Attribute `NextStep` is initialized at **1**, meaning the patient should go to the `Home` trajectory next.
25. An optional Timeout of 30 years is built in, to skip the first 30 years of patient's lives.
26. An optional log_ is built in for bugfixing and active patient tracking.



The visualisation of the `Init` trajectory can be seen in the figure. The resolution of the `simmer.plot` function is quite poor.

Home

```
### Home ----
```

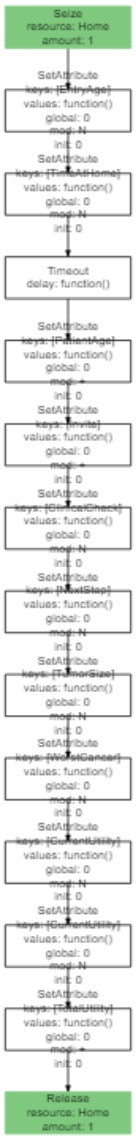
The Home trajectory is where patients spend most of their lives. Every month, patients are checked to see if they should go to another trajectory.

```
VisitHome <- trajectory()%>%
  seize(resource='Home')%>%
  set_attribute(key= 'EntryAge',
                value=function() now(basic_sim)) %>% # record entry age
  set_attribute(key='TimeAtHome',
                value=function() WaitAtHome(
                  get_attribute(basic_sim,'EntryAge'))) %>%
  # determine how long at home (either next step or until next month)
  # time update
  timeout(function() get_attribute(basic_sim,'TimeAtHome')) %>%
  set_attribute(key = 'PatientAge',
                value = function() get_attribute(basic_sim,'TimeAtHome'),
                mod='+',init=0) %>%
  # Next Steps update
  #All five steps possible:
  #a. Go to screening after invite-> 2
  #b. Go to diagnostics due to tumor size or referral after screening ->3
  #c. Go to hospital due to referral -> 4
  #d. Die due to age or illness-> 5
  #e. Continue in Home trajectory, nothing's needed -> 1
  set_attribute(key = 'Invite',
                value= function() ifelse(get_attribute(basic_sim,'PatientAge')
                                         %in% ScreenAges,1,0),
                mod='+')%>% # point a
  set_attribute(key='ClinicalCheck',
                value=function() ClinicalCheckAtHome(
                  get_attribute(basic_sim,'TumorSize'),
                  get_attribute(basic_sim,'ClinSizeCheck'),
                  get_attribute(basic_sim,'PatientAge'))) %>%
  set_attribute('NextStep',
                value=function() AfterHome(
                   get_attribute(basic_sim,'Invite'),
                   get_attribute(basic_sim,'Screened'), # to check point
                   get_attribute(basic_sim,'BIRADS'),
                   get_attribute(basic_sim, 'ClinicalCheck'),# to check pointb
                   get_attribute(basic_sim,'Referral'),  # point c
                   get_attribute(basic_sim,'PatientAge'),
                   get_attribute(basic_sim,'HealthyLifeExpectancy'),
                   get_attribute(basic_sim,'TumorSize') # point D
                    )) %>%
  # Update tumor & cancer parameters:
  set_attribute('TumorSize',
                value=function() GompGrow2(
                  get_attribute(basic_sim,"PatientAge"),
                  get_attribute(basic_sim,"TumorStartAge"),
                  get_attribute(basic_sim,'TumorGrowthRate'),
                  get_attribute(basic_sim,'IsCured'),
                  get_attribute(basic_sim,'max_size'),
                  get_attribute(basic_sim,'RegressionSize'),
                  get_attribute(basic_sim,'StagnateSize'))) %>%
  set_attribute('WorstCancer',
                value = function() ifelse(
```

```
                get_attribute(basic_sim,'TumorSize')>
                  get_attribute(basic_sim,'WorstCancer'),
                get_attribute(basic_sim,'TumorSize'),
                get_attribute(basic_sim,'WorstCancer'))) %>%

# Update utilities
set_attribute(key='CurrentUtility',
              value = function() BaseAgeUtil(
                get_attribute(basic_sim,'PatientAge'))) %>%
set_attribute(key='CurrentUtility',
              value = function() CancerBasedUtil(
                get_attribute(basic_sim,'CurrentUtility'),
                get_attribute(basic_sim,'WorstStage'),
                get_attribute(basic_sim,'PatientAge'),
                get_attribute(basic_sim,'TumorStartAge'))) %>%
set_attribute(key='TotalUtility',
              value=function() get_attribute(
                basic_sim,'TimeAtHome')*
                get_attribute(basic_sim,'CurrentUtility'),
              mod='+')%>%
release(resource='Home')
```

The following steps take place in the Home Trajectory:

1. The Resource `Home` is seized, to track time spent at Home.
2. `EntryAge` of the patient is recorded.
3. `TimeAtHome` is set, using the function `WaitAtHome`. This waits until the next month.
4. A timeout takes place, to update simulation time.
5. The attribute `PatientAge` is updated to reflect this time spent at home.
6. Attribute `Invite` is set to 1 if the patient is eligible for screening.
7. `ClinicalCheck` is set based on the function `ClinicalCheckAtHome`, to see if the patient has a tumour that can be detected by the patient herself.
8. `NextStep` is determined using the `AfterHome` function.
9. The patient's `TumorSize` is updated using the `GompGrow2` function
10. The patient's attribute `WorstCancer` keeps track of the largest cancer size recorded in the patient.
11. The patient's utility is updated. First, the base utility is found based on the patients age
12. Then the `currentuility` is found using the `CancerBasedUtil` function
13. Then the `TotalUtility` is recorded by multiplying the patient's current utility with the time spent at home and adding it to the total.
14. Resource `Home` is released and patients are returned to the main trajectory's branch and rollback.



Screening

```
### Screening ----
```

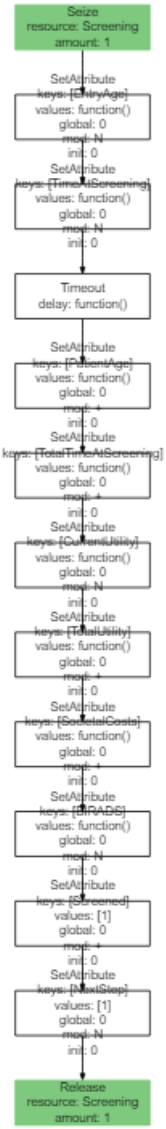
Screening is the trajectory where patients go if they are invited and willing to go to their screening appointment. They are checked to see if they have a tumour.

```
VisitScreening <- trajectory()%>%
  seize('Screening') %>%
  set_attribute(key= 'EntryAge',value=function() now(basic_sim)) %>%
  # record entry age
  set_attribute(key='TimeAtScreening',value= function() ScreenTime()) %>%
  # determine how long at home (either next step or until next month)
  # time update
  timeout(function() get_attribute(basic_sim,'TimeAtScreening')) %>%
  set_attribute(key = 'PatientAge',
                value = function() get_attribute(basic_sim,'TimeAtScreening'),
                mod='+',
                init=0) %>%
  set_attribute(key= 'TotalTimeAtScreening',
                value = function() get_attribute(basic_sim,'TimeAtScreening'),
```

```
                mod='+',
                init=0) %>%

  # add utility
  set_attribute(key = 'CurrentUtility',
                value= function() get_attribute(
                  basic_sim,'CurrentUtility') - 0.15) %>%
  # disutility for during screening
  set_attribute(key='TotalUtility',
                value=function() get_attribute(
                  basic_sim,'TimeAtScreening')*
                  get_attribute(basic_sim,'CurrentUtility'),
                mod='+')%>%
  # perform screening
  set_attribute(key = 'BIRADS',
                value =function() ScreenResult(
                  get_attribute(basic_sim,'TumorSize'))) %>%
  set_attribute(key='Screened',value=1,mod='+',init=0)%>%
  #check next step
  set_attribute(key = 'NextStep',value= 1)%>% #go home after screening

  release('Screening')
```

The following happens in the Screening Trajectory:

1. Resource `Screening` is seized to keep track of how often and for how long it is used.
2. `EntryAge` of the patient is recorded.
3. The time the patient stays at screening is determined with function `ScreenTime`.
4. The simulation time is updated using a timeout.
5. The `PatientAge` attribute is updated to reflect this.
6. The attribute `TotalTimeAtScreening` is updated to keep track of the total time a patient has spent at screening
7. The `CurrentUtility` of a patient, which was set in the Home trajectory, is updated with a minor disutility for the duration of the screening appointment
8. This utility is multiplied with the time spent at screening and added to the `TotalUtility` parameter
9. The screening is performed using the function `ScreenResult` based on the actual size of the tumor.
10. Attribute `Screened` is updated to reflect the number of times the patient has been screened.
11. `NextStep` is set to home, so that the patient will always return home after screening.
12. Resource `Screening` is released.



Diagnostics

```
### Diagnostics ----
```

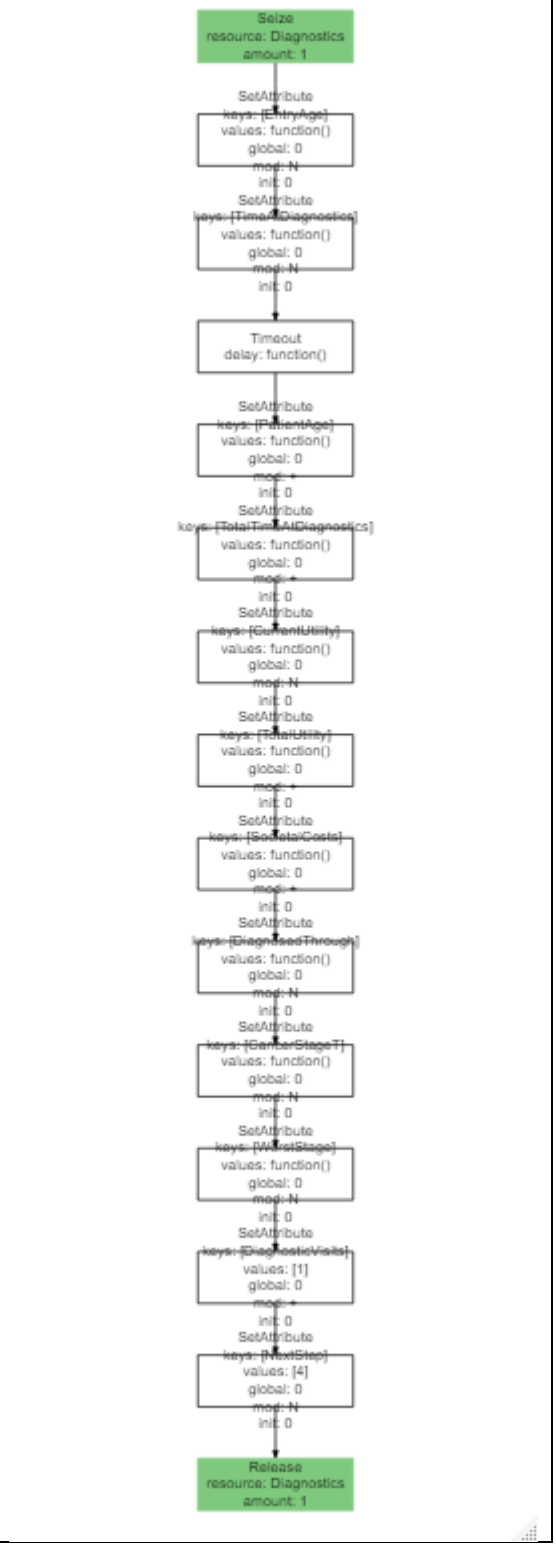
Diagnostics is the trajectory that patients follow if they have either a BIRADS of equal or more than 3, or if they have clinically found a tumour themselves and want this diagnosed.

```
VisitDiagnostics <- trajectory()%>%
  seize('Diagnostics') %>%
  set_attribute(key= 'EntryAge',value=function() now(basic_sim)) %>%
  # record entry age
  set_attribute(key='TimeAtDiagnostics',value= function() DiagTime()) %>%
  # determine how long at home (either next step or until next month)
  # time update
  timeout(function() get_attribute(basic_sim,'TimeAtDiagnostics')) %>%
  set_attribute(key = 'PatientAge',
                value = function()
get_attribute(basic_sim,'TimeAtDiagnostics'),
                mod='+',
                init=0) %>%
```

```
  set_attribute(key= 'TotalTimeAtDiagnostics',
                value = function()
get_attribute(basic_sim,'TimeAtDiagnostics'),
                mod='+',init=0)%>%
  # add utility
  set_attribute(key = 'CurrentUtility',
                value= function() get_attribute(
                  basic_sim,'CurrentUtility') - 0.25) %>% #disutility for diag
  set_attribute(key='TotalUtility',
                value=function() get_attribute(
                  basic_sim,'TimeAtDiagnostics')*
                  get_attribute(basic_sim,'CurrentUtility'),
                mod='+')%>%
  # add costs
  #set_attribute(key='MedicalCosts',value= 600,mod='+',init=0) %>%
  set_attribute(key='SocietalCosts',
                value=function() get_attribute(
                  basic_sim,'TimeAtDiagnostics')/hour*35,
                mod='+') %>%
  set_attribute(key = 'DiagnosedThrough',
                value = function() ifelse(
                  get_attribute(basic_sim,'ClinicalCheck'),1,0)) %>%
  #through screening or clinical?
  set_attribute(key = "CancerStageT",
                value= function() Staging(
                  get_attribute(basic_sim,'TumorSize'),
                  get_attribute(basic_sim,'StagingProb'))) %>%
  set_attribute(key = "WorstStage",
                value= function() ifelse(
                  get_attribute(basic_sim,'CancerStageT')>
                     get_attribute(basic_sim,'WorstStage'),
                  get_attribute(basic_sim,'CancerStageT'),
                  get_attribute(basic_sim,'WorstStage'))) %>%
  set_attribute('DiagnosticVisits',value = 1, mod= '+',init=0) %>%
  set_attribute(key = 'NextStep',
                value= function() ifelse(
                  get_attribute(basic_sim,'CancerStageT')== 0,
                        1,
                        4))%>%
  release('Diagnostics')
```

The following happens in the VisitDiagnostics Trajectory:

1. Resource `Diagnostics` is seized to keep track of how often and for how long it is used.
2. `EntryAge` of the patient is recorded.
3. The time the patient stays at diagnostics is determined with function `DiagTime`.
4. The simulation time is updated using a timeout.
5. The `PatientAge` attribute is updated to reflect this.
6. The attribute `TotalTimeAtDiagnostics` is updated to keep track of the total time a patient has spent at diagnostics
7. The `CurrentUtility` of a patient, which was set in the Home trajectory, is updated with a minor disutility for the duration of the Diagnostics appointment
8. This utility is multiplied with the time spent at diagnostics and added to the `TotalUtility` parameter
9. The attribute `DiagnosedThrough` is set to keep track of if a patient was diagnosed through screening or clinically.
10. The stage of the cancer, recorded in attribute `CancerStageT`, is determined using function `Staging` based on the `TumorSize` and the patients initialized probability variable.
11. The attribute `WorstStage` is updated to keep track of the worst type of cancer a patient has experienced.
12. The attribute `DiagnosticVisits` is updated to reflect the number of times a patient visited Diagnostics
13. The attribute `NextStep` is set to 4 or 1, to send patients to the hospital if they need treatment and send them home if they have nothing.
14. Resource `Diagnostics` is released.



Cured

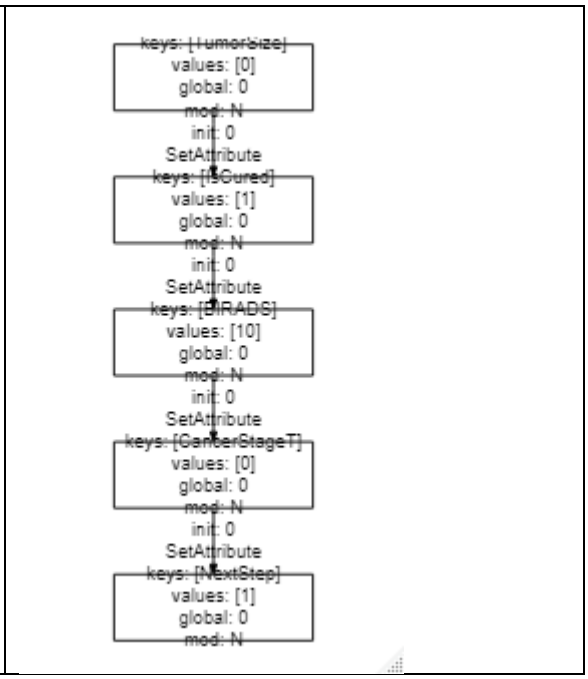
```
### Cured ----
```

Cured is the trajectory that patients visit if they are cured of their cancer, with no chance of recurrence. They visit this trajectory at the end of the hospital trajectory, but it has to be initiated before the hospital trajectory.

```
Cured <- trajectory() %>%
  #log_('cured') %>%
  set_attribute(key='TumorSize', value = 0) %>%
  set_attribute(key='IsCured',value=1) %>%
  set_attribute(key='BIRADS',value=10) %>%
  set_attribute(key='CancerStageT',value=0) %>%
  set_attribute(key = 'NextStep',value=1)
```

In Cured, the following takes place:

1. The patient's `TumorSize` is reset to **0**
2. The attribute `IsCured` is updated to **1**
3. The attribute `BIRADS` is set to 10, a number where no steps take place.
4. The attribute `CancerStageT` is reset to **0**
5. The `nextStep` is set to 1, returning the patient home



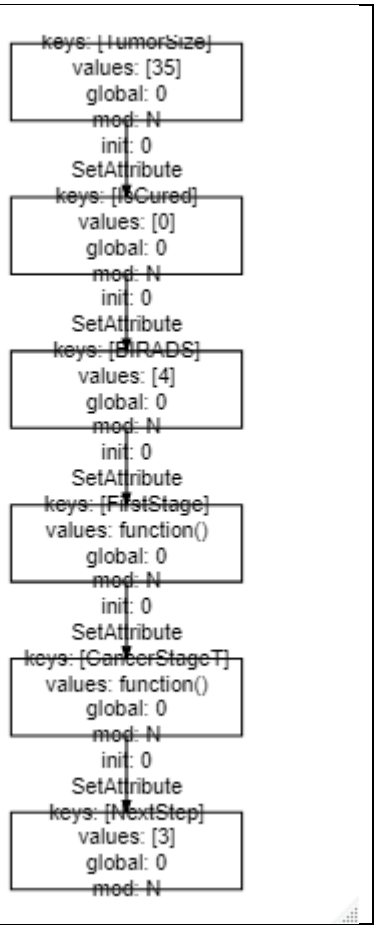
Residuals

```
### Residuals ----
```

Residuals is where patients are sent if they have a tumour that will recur at a later point in time. It is one of the two options at the end of the hospital trajectory.

```
Residual <- trajectory() %>%
  set_attribute('TumorSize', value= 35) %>%
  set_attribute(key='IsCured',value=0) %>%
  set_attribute(key='BIRADS',value=4) %>%
  set_attribute(key='FirstStage',
                value=function() get_attribute(basic_sim,'WorstStage')) %>%
  set_attribute(key='CancerStageT',
                value=function() get_attribute(basic_sim,'WorstStage')) %>%
  # go back home
  set_attribute(key = 'NextStep',value=1)
```

The following happens in this trajectory:

1. The TumorSize is reset to a value of 35. This value is chosen because at that size in diagnostics, there are multiple options for at which stage the tumor will return, in line with the probabilities the IKNL provides. 2. The attribute IsCured is reset to 0. 3. The BIRADS attribute is set to 4, making the person eligible for diagnosis. 4. The FirstStage attribute is created, to save the stage of the initial tumor. 5. The attribute CancerStageT is reset to the WorstStage of the previous tumor. 6. The patient is sent home, where she will be sent back to diagnostics and the hospital.	

Hospital

```
### Hospital ----
```

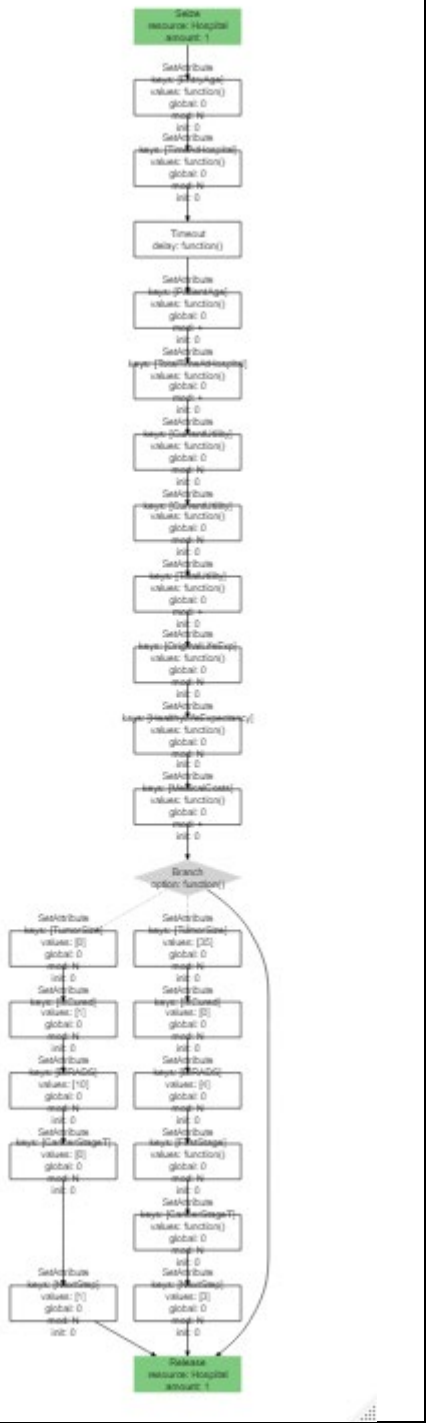
In the Hospital trajectory, the treatment of the patients is simulated in a basic way. It uses averages with variation of treatments per stage instead of simulating the entire treatments. Treatment is similar for patients diagnosed clinically or through screening, so there is no difference there. The choice was made to keep this as simple as possible without compromising the results.

```
VisitHospital <- trajectory()%>%
  seize('Hospital') %>%
  set_attribute(key= 'EntryAge',
                value=function() now(basic_sim)) %>% # record entry age
  set_attribute(key='TimeAtHospital',
                value= function()
                  3^get_attribute(basic_sim,'CancerStageT')*day)%>%
  # determine how long at home (either next step or until next month)
  # time update
  timeout(function() get_attribute(basic_sim,'TimeAtHospital')) %>%
  set_attribute(key = 'PatientAge',
                value = function() get_attribute(basic_sim,'TimeAtHospital'),
                mod='+',
                init=0) %>%
  set_attribute(key= 'TotalTimeAtHospital',
                value = function() get_attribute(basic_sim,'TimeAtHospital'),
                mod='+',
                init=0) %>%
```

```
  # update utilities
  set_attribute(key='CurrentUtility',
                value = function() BaseAgeUtil(
                  get_attribute(basic_sim,'PatientAge'))) %>%
  set_attribute(key='CurrentUtility',
                value = function() CancerBasedUtil(
                  get_attribute(basic_sim,'CurrentUtility'),
                  get_attribute(basic_sim,'CancerStageT'),
                  get_attribute(basic_sim,'PatientAge'),
                  get_attribute(basic_sim,'TumorStartAge'))) %>%
  set_attribute(key='TotalUtility',
                value=function() get_attribute(
                  basic_sim,'TimeAtHospital')*
                  get_attribute(basic_sim,'CurrentUtility'),
                mod='+')%>%
  #instead of cure, go to adjust life exp
  set_attribute(key='OriginalLifeExp',
                value=function() get_attribute(
                  basic_sim,'HealthyLifeExpectancy')) %>%
  set_attribute(key = 'HealthyLifeExpectancy',
                value=function() CancerSurvival(
                  get_attribute(basic_sim,'WorstStage'),
                  get_attribute(basic_sim,'HealthyLifeExpectancy'),
                  get_attribute(basic_sim,'PatientAge'))) %>%
  set_attribute(key='MedicalCosts',
                value=function() TreatmentCost(
                  get_attribute(basic_sim,'WorstStage'),
                  get_attribute(basic_sim,'PatientAge'),
                  get_attribute(basic_sim,'HealthyLifeExpectancy')),
                mod='+') %>%
  branch(option= function() ifelse(runif(1)<0.88,1,2),continue=c(T,T),
         Cured,
         Residual) %>%
  release('Hospital')
```

The following steps are taken in the Hospital trajectory. Note that the plot of the trajectory also includes the trajectories for Cured and Residuals.

1. Resource `Hospital` is seized to keep track of how often and for how long it is used.
2. `EntryAge` of the patient is recorded.
3. The time the patient stays at Hospital is dependant on the stage of the tumor, with later stage tumor taking exponentially longer to treat than low stage tumors.
4. The simulation time is updated using a timeout.
5. The `PatientAge` attribute is updated to reflect this.
6. The attribute `TotalTimeAtHospital` is updated to keep track of the total time a patient has spent in treatment
7. The `CurrentUtility` of a patient is updated to reflect the patient's base utility
8. The `CurrentUtility` is updated with the function CancerBasedUtil
9. This utility is multiplied with the time spent in the Hospital trajectory and added to the `TotalUtility` parameter
10. The attribute `HealthyLifeExpectancy` is saved under new attribute `OriginalLifeExpectancy`
11. The `HealthyLifeExpectancy` is adjusted to reflect the time this patient has left to live based on their stage of cancer, using function `CancerSurvival`.
12. The medical costs for treatment are calculated using the function `TreatmentCost`
13. The trajectory now branches with a 12% chance to residual or cured and the patient ends up in one of the trajectories discussed above.
14. After the cured or residual trajectory, they are returned to this trajectory, and the patients are returned home after the Hospital resource is released



Death

```
### Death ----
```

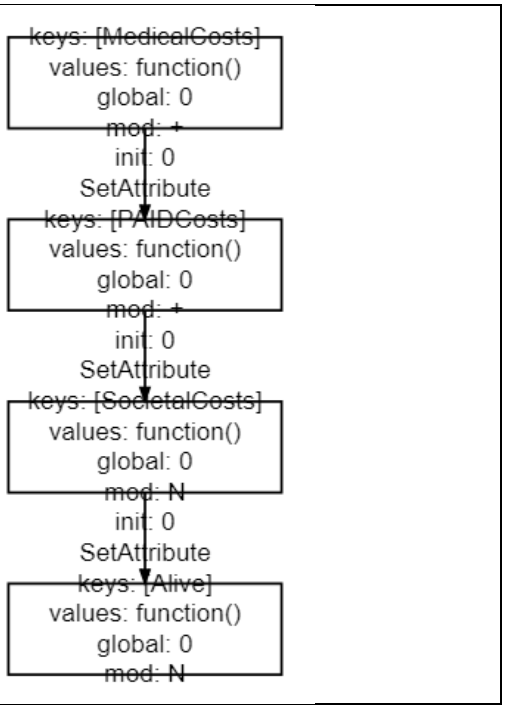
In the Death trajectory, some calculations are made on the patients final attributes, which are also recorded. The patient's trajectories end here and the patients now die.

```
Death <- trajectory()%>%
  #log_('die') %>%
  #compute total costs
  set_attribute('MedicalCosts',
```

```
                value = function() ScreenDiagInvitesCost(
                  get_attribute(basic_sim,'Screened'),
                  get_attribute(basic_sim,'DiagnosticVisits'),
                  get_attribute(basic_sim,'Invite')),
                mod = '+') %>%
  set_attribute(key = 'PAIDCosts',
                value = function() PaidCosts(
                  get_attribute(basic_sim,'HealthyLifeExpectancy'),
                  get_attribute(basic_sim,'TumorStartAge')),
                mod = '+') %>%
  set_attribute(key='SocietalCosts',
                value=function() VisitationCosts(
                  get_attribute(basic_sim,'Screened'),
                  get_attribute(basic_sim,'DiagnosticVisits'),
                  get_attribute(basic_sim,'TotalTimeAtHospital'))) %>%



  # extract all atrributes
  set_attribute(key='Alive',value = function() addAtts(EndDatDf))
```

The following happens in the trajectory when the patients die.

1. The Medical costs are calculated using the function `ScreenDiagInvitesCosts`, which adds the costs associated with the number of visits to screening and diagnosis as well as the invites received. These are added to the already existing attribute `MedicalCosts`.
2. The costs associated with other diseases after treatment for breast cancer are calculated with the PAID tool in the function `PaidCosts`. They are added to the separate attribute `PAIDCosts`.
3. The societal costs associated with visits to screening, diagnosis and hospital are calculated with the function `VisitationCosts`.
4. All the patients final attributes are recorded in the `EndDatDf` using the function `addAtts`, which is also used to set `Alive` to **0**, declaring the patients as dead.



Base Model

```
### Base Model ----
```

The Base Model stitches all the other trajectories together and uses one branch and rollback to select where patients should be sent next.

```
#stitching the blocks together
basic_model <- trajectory()%>%
  # first initialize
  join(Initialization) %>%

    # 'store' patients at home until it's:
```

```
  # time to go to screening
  # time to go to diagnostics
  # time to go to hospital
  # time to die
  branch(option=function() get_attribute(basic_sim,'NextStep'),
         continue=c(T,T,T,T,F),
         VisitHome,
         VisitScreening,
         VisitDiagnostics,
         VisitHospital,
         Death) %>%

  rollback(amount=1)
```

This trajectory creates a new trajectory object, which is first joined with the Initialization trajectory. After this, a branch is created that retrieves the patient's NextStep attribute, and sends the patient to either Home, Screening, Diagnostics, Hospital or Death. Patients are returned to this trajectory after finishing one of the first four, and are removed from the simulation after finishing the death trajectory. After finishing the branch, the patients are rolled back to enter another branching event.

The full model then looks like this:

Plot

```
## Plot ----
```

In plot, there are various lines of code to plot the trajectories, of which the visualisations have been shown in the separate sections above.

```
plot(basic_model,verbose=TRUE)

# plot(Initialization,verbose=T)
# plot(VisitHome,verbose=T)
# plot(VisitScreening,verbose=T)
# plot(VisitDiagnostics,verbose=T)
# plot(Cured,verbose=T)
# plot(Residual,verbose=T)
# plot(VisitHospital,verbose=T)
```

Run

```
## Run----
```

In Run, the simulation is initiated and ran. Various monitoring objects are also retrieved from the simulation.

```
basic_sim <- simmer()%>%
  add_resource('Home',capacity=Inf) %>%
  add_resource('Screening',capacity=Inf) %>%
  add_resource('Diagnostics',capacity=Inf) %>%
  add_resource('Hospital',capacity=Inf) %>%
  add_generator(name_prefix='patient',
                trajectory=basic_model,
                distribution=at(rep(x=0, times=n.patients)),
                mon=mon.patients)

start_time <- Sys.time()
basic_sim %>%
  reset() %>%
  run(progress=progress::progress_bar$new()$update,until=110*year,steps=1000)
end_time <- Sys.time()
RunTime <- end_time - start_time
print(RunTime)

patient_monitor <-
  get_mon_arrivals(basic_sim) %>%
  transform(wait = end_time - start_time - activity_time) %>%
  transform(totalTime = end_time - start_time)
patient_attributes <- get_mon_attributes(basic_sim)

testdf <- get_mon_attributes(basic_sim)

EndDatDf2 = EndDatDf[-1,]
```

First, the Simmer simulation is initialized. The resources Home, Screening, Diagnostics and Hospital are added, and the generator patient is added as movable object. It is send on the basic_model trajectory, from where it will reach different other trajectories. There is no distribution, instead all patients are spawned at the beginning of the simulation. They are monitored for either 0 or 2, based on the number of patients in the model. A ‘2’ allows for tracking of every attribute change of every patient, a 0 will return no attributes at all.

The start time is recorded and the simulation is reset and started. A progress bar is shown, showing the time from 0 to 110 years, giving plenty of time for all patients to live their lives. After the simulation is finished, the end time is recorded and the run time is shown.

The patient_monitor, patient_attributes and testdf are only used if the monitor is set to ‘2’. If it’s set to 0, no attributes will be recorded. The EndDatDf2 is a copy of EndDatDf created in the Death trajectory, to remove the first row containing only the word ‘test’.

Reporting

```
# Reporting ----
```

In the reporting section, various KPIs are collected and plots are created.

```
incpercentages <- c()
PercThroughScreens <- c()

df <- EndDatDf

numsdf = df[-1,]
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}
meanAge = round(mean(round(as.numeric(numsdf$PatientAge)/year)),2)
medianAge = median(round(as.numeric(numsdf$PatientAge)/year))
modeAge = getmode(round(as.numeric(numsdf$PatientAge)/year))
patients_total <- nrow(df)
cancers_found <- table(as.numeric(df$WorstStage))
inc_perc <-   round((cancers_found[2]+
                       cancers_found[3] +
                       cancers_found[4]+
                      cancers_found[5])/ patients_total * 100,2)
```

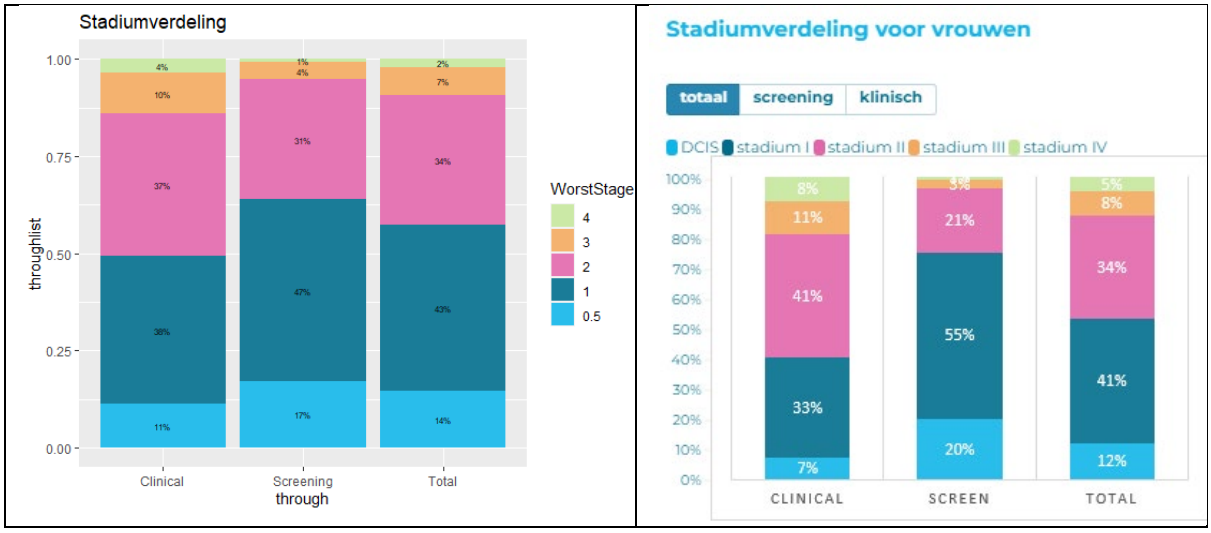
First some lists are initialized and the EndDatDf is copied to df. Then the KPIs meanAge, medianAge, modeAge, patients_total, cancers_found and inc_perc are collected to be displayed at the end.

Stage distribution Plot 2

```
## Stadiumverdeling plot 2 ----
subdf <- df %>%
  dplyr::select(WorstStage,DiagnosedThrough) %>%
  na.omit()


subdf2 <- table(subdf)
subdf3 <- as.data.frame(subdf2)
WorstStage = rep(c(0.5,1,2,3,4),3)
ThroughClin = subdf2[,2][2:6]
ThroughScreen = subdf2[,1][2:6]
ThroughTotal = ThroughClin + ThroughScreen
throughlist = append(ThroughTotal,ThroughClin)
throughlist = append(throughlist,ThroughScreen)
percThroughScreen = round(sum(ThroughScreen) /
                            (sum(ThroughScreen)+ sum(ThroughClin)) *100,2)

incpercentages<-append(incpercentages,inc_perc)
PercThroughScreens<-append(PercThroughScreens,percThroughScreen)


subdf <- df %>%
  dplyr::select('WorstStage','DiagnosedThrough','FirstStage') %>%
  mutate(WorstStage = ifelse(is.na(FirstStage),
                             WorstStage,
                             ifelse(WorstStage>FirstStage,
                                    WorstStage,
                                    FirstStage))) %>%
  select('WorstStage','DiagnosedThrough') %>%
  na.omit()
```

```
subdf2 <- table(subdf)
subdf3 <- as.data.frame(subdf2)
WorstStage = rep(c(0.5,1,2,3,4),3)
ThroughClin = subdf2[,2][1:5]
ThroughScreen = subdf2[,1][1:5]
ThroughTotal = ThroughClin + ThroughScreen
throughlist = append(ThroughTotal,ThroughClin)
throughlist = append(throughlist,ThroughScreen)
through = c('Total','Total','Total','Total','Total',
            'Clinical','Clinical','Clinical','Clinical','Clinical',
            'Screening','Screening','Screening','Screening','Screening')

data3 = data.frame(WorstStage,throughlist,through)

data3$WorstStage <- factor(data3$WorstStage,levels = c("4", "3",
                                                       "2", "1", "0.5"))

# Create a vector of colors for each level of WorstStage
colors <- c("#cbe9a6", "#f4b26f", "#e576b4", "#1a7c98", "#29bdeb")

# Calculate the total count for each level of through
total_count <- aggregate(data3$throughlist, by = list(data3$through), sum)

# Calculate the percentage for each combination of WorstStage and through
data3$percent <- data3$throughlist /
  total_count[match(data3$through, total_count$Group.1), "x"] * 100


plot.new()
p2<-ggplot(data3,aes(fill=WorstStage,y=throughlist,x=through))+
  geom_bar(position='fill',stat='identity')+
  labs(title='Stadiumverdeling')+
  scale_fill_manual(values=colors) +
  geom_text(aes(label = paste0(round(percent), "%")),
            position = position_fill(vjust = 0.5),size=2)


plot2 <- recordPlot()
plot.new()
```

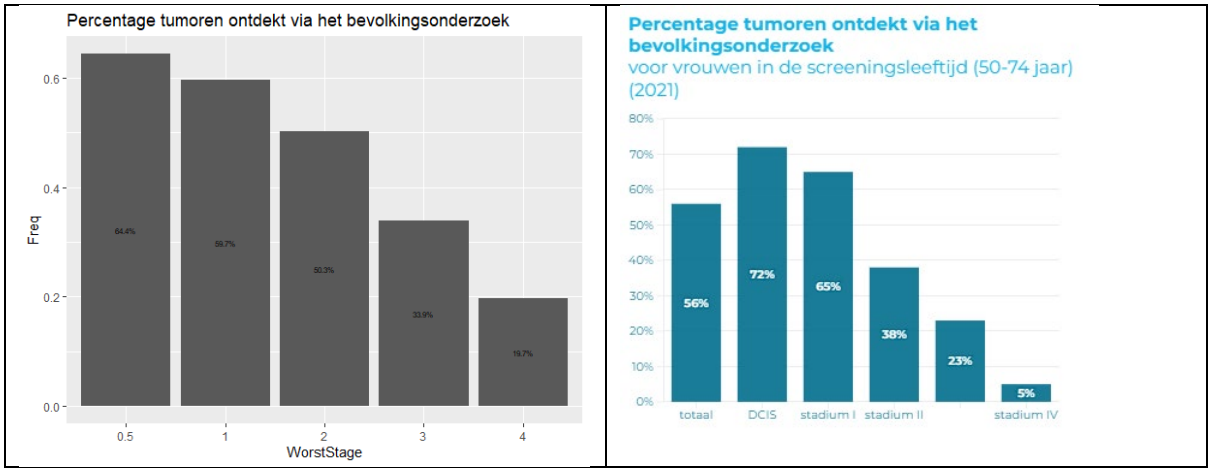
With this code, the plot on the left is created. It aims to resemble the plot on the right, which is from IKNL data.



Percentage Plot 1

```
## Percentage plot 1 ----

subdf4 <- subdf3 %>%
  filter(DiagnosedThrough==0)

# filter subdf3 by DiagnosedThrough == 0 and drop the DiagnosedThrough column
subdf_filtered <- subset(subdf3, DiagnosedThrough == 0,
                         select = -c(DiagnosedThrough))

# calculate the total  count of Freq values for each WorstStage
freq_total <- aggregate(Freq ~ WorstStage, subdf3, sum)$Freq

# divide the Freq column by the total count of Freq values for each WorstStage
subdf_filtered$Freq <- subdf_filtered$Freq / freq_total
# subdf_filtered <- subset(subdf_filtered, row_number() != 6)

p1<-ggplot(subdf_filtered,aes(x=WorstStage,y=Freq))+
  geom_bar(stat='identity')+
  labs(title='Percentage tumoren ontdekt via het bevolkingsonderzoek')+
  scale_fill_brewer(palette="Spectral")+
  geom_text(aes(label = paste0(round(Freq*100, 1), "%")),
            position = position_stack(vjust = 0.5),size=2)
plot1 <- recordPlot()
plot.new()
```

With this code, the plot on the left is created. It aims to resemble the plot on the right, which is from IKNL data. Note that this version did not yet include a total tally, the leftmost column of the IKNL graph.

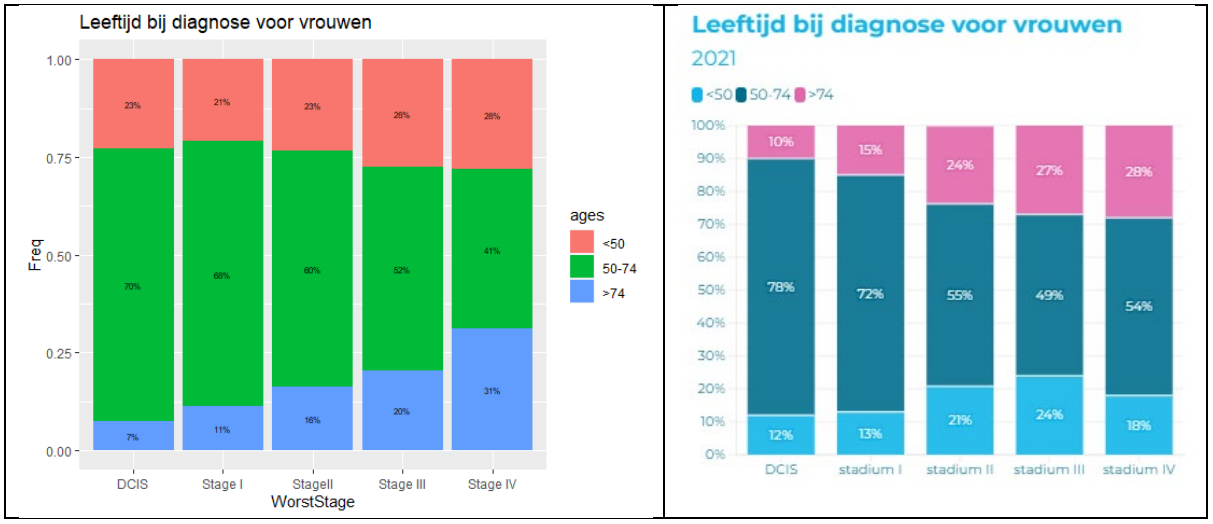


Age of Diagnosis Plot 3

```
## leeftijd diagnose plot 3 ----

subdf5 <- df %>%
  dplyr::select('WorstStage','TumorStartAge') %>%
  na.omit() %>%
  filter(TumorStartAge<100*year) %>%
  filter(WorstStage>0) %>%
  mutate(TumorStartAge = as.numeric(TumorStartAge)) %>%
  mutate(ages = cut(TumorStartAge,breaks=c(0,50*year,74*year,Inf))) %>%
  dplyr::select(ages,WorstStage)

subdf6 = rbind(as.data.frame(table(subdf5)))
```

```
# Calculate the total count for each level of through
total_count2 <- aggregate(subdf6$Freq, by = list(subdf6$WorstStage), sum)

# Calculate the percentage for each combination of WorstStage and through
subdf6$percent <- subdf6$Freq /
  total_count2[match(subdf6$WorstStage, total_count2$Group.1), "x"] * 100

p3<- ggplot(subdf6,aes(fill=ages,y=WorstStage,x=Freq))+
  geom_bar(position='fill',stat='identity')+
  coord_flip()+
  scale_fill_discrete(labels=c('<50', '50-74', '>74'))+
  scale_y_discrete(labels=c("0.5" = "DCIS",
                            "1" = "Stage I",
                            "2" = "StageII",
                            "3" = "Stage III",
                            "4" = "Stage IV"))+
  labs(title='Leeftijd bij diagnose voor vrouwen')+
  geom_text(aes(label = paste0(round(percent),'%')),
            position = position_fill(vjust = 0.5),size=2)
plot3 <- recordPlot()
plot.new()
```

With this code, the plot on the left is created. It aims to resemble the plot on the right, which is from IKNL data.

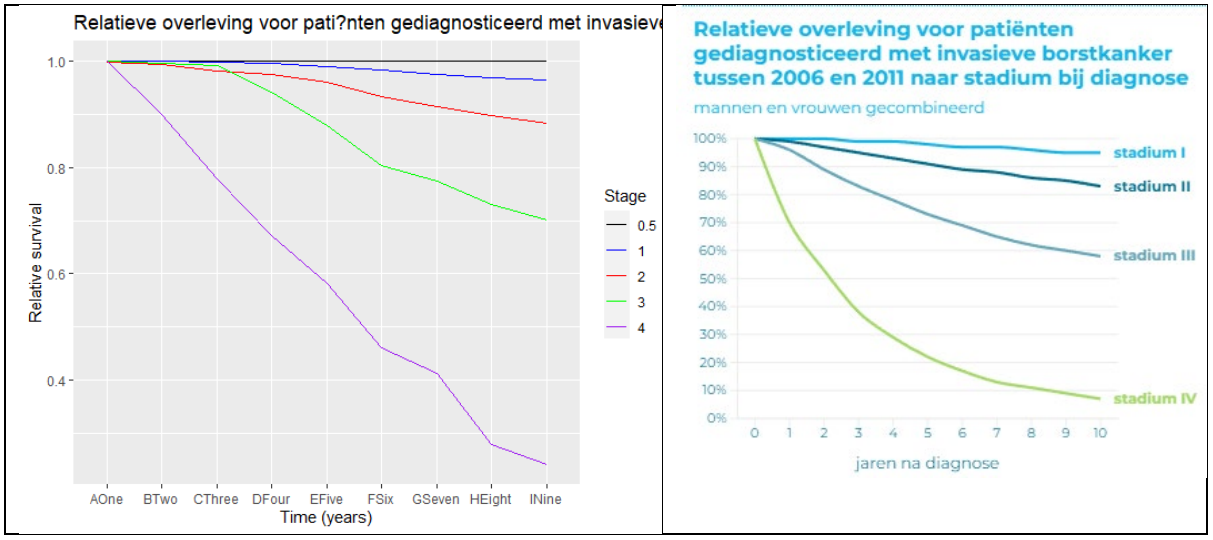


Survival Curve Plot 4

```
## Survival curve plot 4 ----

subdf7 <- df%>%
  dplyr::select(WorstStage,PatientAge, TumorStartAge,OriginalLifeExp)%>%
  filter(TumorStartAge<100*year) %>%
  filter(WorstStage>0) %>%
  mutate(WorstStage = as.numeric(WorstStage)) %>%
  mutate(TumorStartAge = as.numeric(TumorStartAge)/year) %>%
  mutate(PatientAge = as.numeric(PatientAge)/year) %>%
  mutate(OriginalLifeExp = as.numeric(OriginalLifeExp)/year) %>%
  mutate(Survival = PatientAge - TumorStartAge) %>%
  mutate(ShouldveSurvived = OriginalLifeExp - TumorStartAge)%>%
  mutate(One = ifelse(round(Survival)>=1,1,
                      ifelse(round(ShouldveSurvived)>=1,0,NA))) %>%
  mutate(Two = ifelse(round(Survival)>=2,1,
                      ifelse(round(ShouldveSurvived)>=2,0,NA))) %>%
```

```
  mutate(Three = ifelse(round(Survival)>=3,1,
                        ifelse(round(ShouldveSurvived)>=3,0,NA))) %>%
  mutate(Four = ifelse(round(Survival)>=4,1,
                       ifelse(round(ShouldveSurvived)>=4,0,NA))) %>%
  mutate(Five = ifelse(round(Survival)>=5,1,
                       ifelse(round(ShouldveSurvived)>=5,0,NA))) %>%
  mutate(Six = ifelse(round(Survival)>=6,1,
                      ifelse(round(ShouldveSurvived)>=6,0,NA))) %>%
  mutate(Seven = ifelse(round(Survival)>=7,1,
                        ifelse(round(ShouldveSurvived)>=7,0,NA))) %>%
  mutate(Eight = ifelse(round(Survival)>=8,1,
                        ifelse(round(ShouldveSurvived)>=8,0,NA))) %>%
  mutate(Nine = ifelse(round(Survival)>=9,1,
                       ifelse(round(ShouldveSurvived)>=9,0,NA))) %>%
  mutate(Ten = ifelse(round(Survival)>=10,1,
                      ifelse(round(ShouldveSurvived)>=10,0,NA)))

subdf7 <- subdf7 %>%
  group_by(WorstStage) %>%
  summarise(
    AAZero = c(1,1,1,1,1),
    AOne = mean(One, na.rm = TRUE),
    BTwo = mean(Two, na.rm = TRUE),
    CThree = mean(Three, na.rm = TRUE),
    DFour = mean(Four, na.rm = TRUE),
    EFive = mean(Five, na.rm = TRUE),
    FSix = mean(Six, na.rm = TRUE),
    GSeven = mean(Seven, na.rm = TRUE),
    HEight = mean(Eight, na.rm = TRUE),
    INine = mean(Nine, na.rm = TRUE),
    JTen = mean(Ten, na.rm = TRUE)
  )

subdf7
# convert from wide to long format
df_long <- tidyr::pivot_longer(subdf7, cols = c(2:11),
                               names_to = "Column", values_to = "Value")

# create the line plot
p4<-ggplot(data = df_long, aes(x = Column, y = Value,
                               group = WorstStage, color =
factor(WorstStage))) +
  geom_line() +
  scale_color_manual(values = c("black", "blue", "red", "green", "purple")) +
  labs(x = "Time (years)", y = "Relative survival", color = "Stage",
       title='Relatieve overleving voor patiënten gediagnosticeerd met
invasieve borstkanker naar stadium bij diagnose')
plot4 <- recordPlot()
plot.new()
```

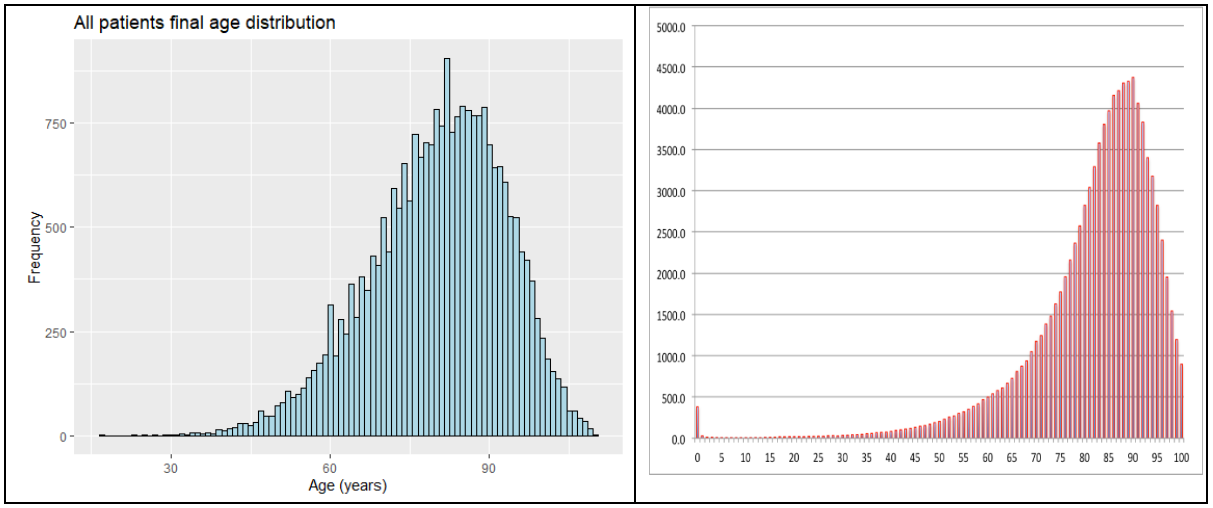
With this code, the plot on the left is created. It aims to resemble the plot on the right, which is from IKNL data.



Final Age Distribution

```
# More plots for finetuning

## Final age distribution ----
p5 <- ggplot(df, aes(x = as.numeric(PatientAge) / year)) +
  geom_histogram(binwidth = 1, color = "black", fill = "lightblue") +
  labs(title = "All patients final age distribution",
       x = "Age (years)",
       y = "Frequency")
```

With this code, the plot on the left is created showing the distribution of the final ages of all patients. This compares to the actual life expectancy in the Netherlands. The graph on the right is included for demonstrative purposes and based on data from the UK: Numbers of women expected to die at each age, out of 100,000 born, assuming mortality rates stay the same as 2010-2012. The expectation is 83, median 86 (*Why "life Expectancy" Is a Misleading Summary of Survival | Understanding Uncertainty*, n.d.).

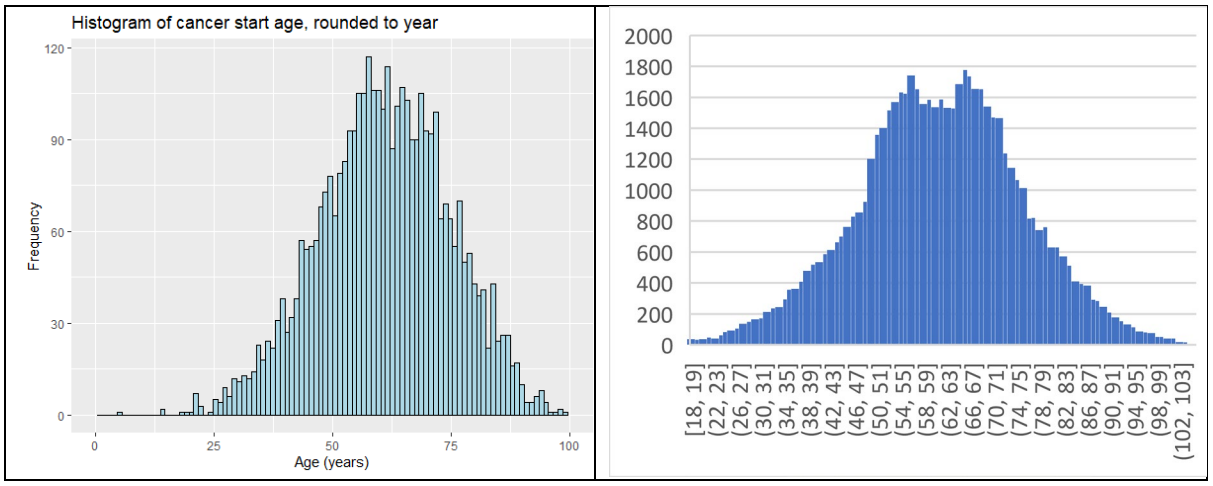


Histogram Cancer Start age

```
## Histogram cancer start age ----
subdf8<- df %>%
  filter(TumorStartAge<1000*year)
```

```
p6 <- ggplot(subdf8, aes(x = as.numeric(TumorStartAge) / year)) +
  geom_histogram(bins = 100, color = "black", fill = "lightblue") +
  xlim(0, 100) +
  labs(title = "Histogram of cancer start age, rounded to year",
       x = "Age (years)",
       y = "Frequency")
```

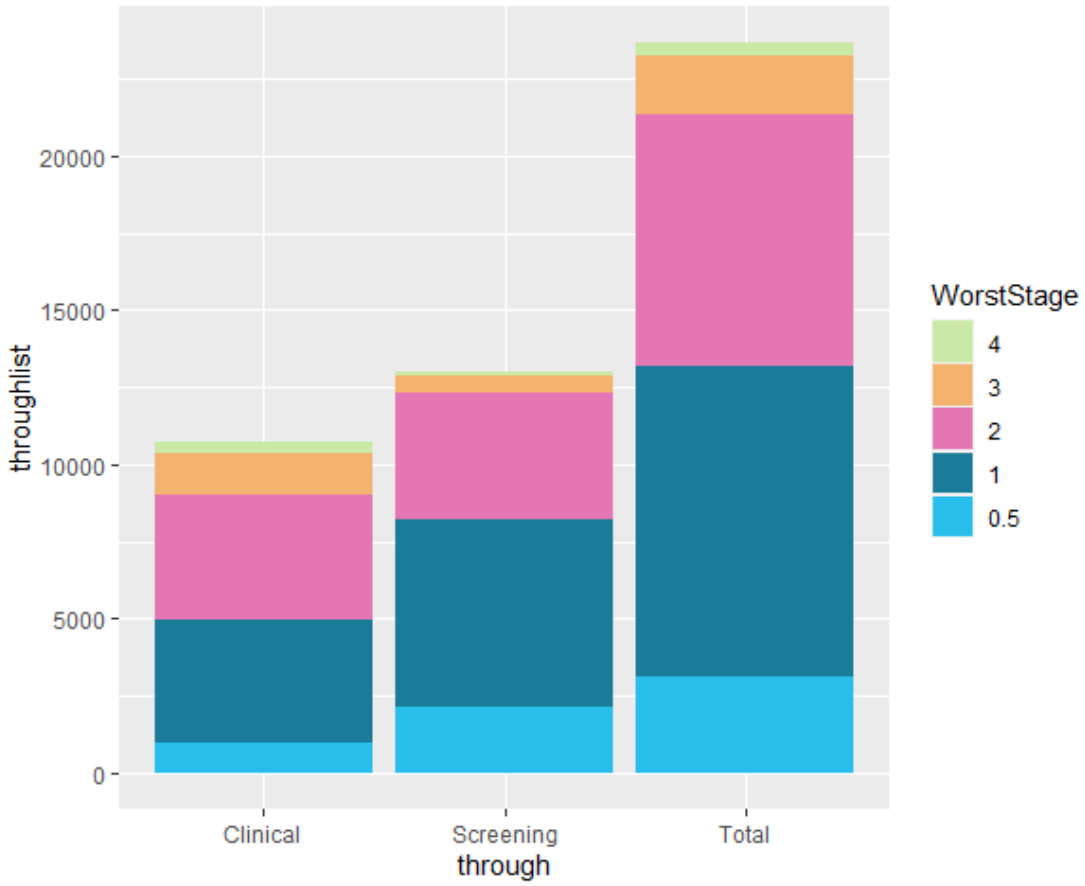
This plot shows the histogram of ages at which patients get tumor. The plot from the simulation is on the left, the histogram from the IKNL data is on the right.



Actual Stage Distribution

```
## actual stage distribtuion ----
p7 <- ggplot(data3,aes(fill=WorstStage,y=throughlist,x=through))+
  geom_bar(position='stack',stat='identity')+
  scale_fill_manual(values=colors)
```

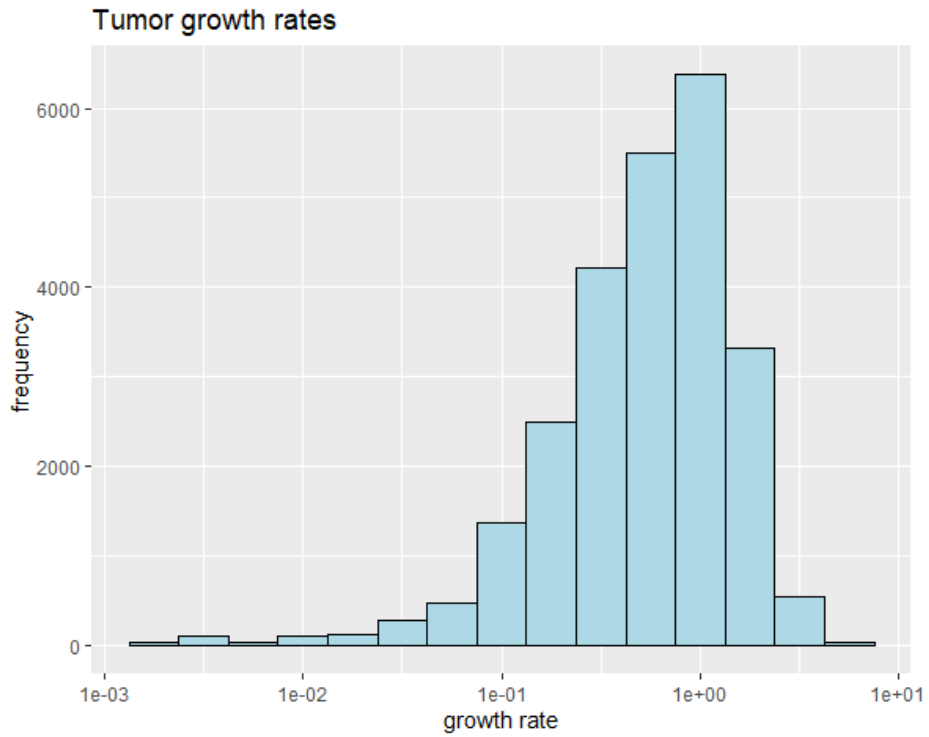
This plot shows the non-normalized stage distributions, to visually inspect which percentage is due to screening and which due to clinical diagnosis.



Tumor Growth Rates Histogram

```
## Tumor growth rates hist ----
p8 <- ggplot(df,aes(x=as.numeric(TumorGrowthRate)))+
  geom_histogram(binwidth = 0.25 , color='black', fill='lightblue')+
  labs(title='Tumor growth rates',
       x='growth rate',
       y= 'frequency')+
  scale_x_log10()
```

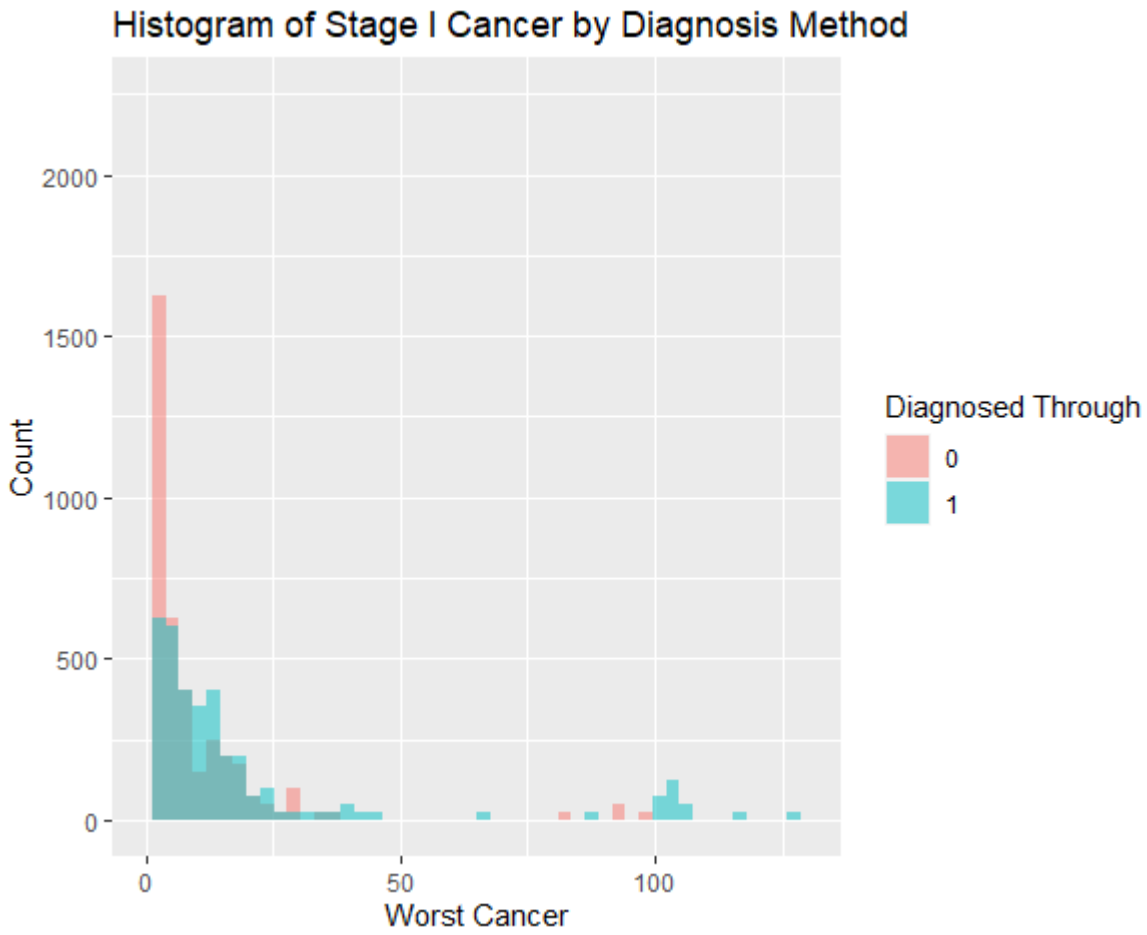
This plot shows a histogram of tumor growth rates on a logistic scale.



Plot Stage Sizes Histogram

```
#finetune stagefromsize diagram plot
## plot stage size hist ----

SizeTestdf<- df %>%
  dplyr::select(DiagnosedThrough,WorstCancer,WorstStage) %>%
  filter(WorstStage == 1) %>%
  mutate(WorstCancer = as.numeric(WorstCancer))

ggplot(SizeTestdf, aes(x = WorstCancer, fill = factor(DiagnosedThrough))) +
  geom_histogram(alpha = 0.5, position = "identity", bins = 50) +
  scale_fill_discrete(name = "Diagnosed Through") +
  labs(title = "Histogram of Worst Cancer by Diagnosis Method",
       x = "Worst Cancer", y = "Count")+
  xlim(c(0,130))
```

This plot gives an overview of at which sizes certain stage tumors have been diagnosed most often, for finetuning purposes and gaining insights in the workings of the model.



KPIs

The following code renders the KPIs and displays them.

```
patients_total <- nrow(df)
cancers_found <- table(as.numeric(df$WorstStage))
inc_perc <-   round((cancers_found[2]+
                       cancers_found[3] +
                       cancers_found[4]+
                       cancers_found[5])/ patients_total * 100,2)
percThroughScreen = round(sum(ThroughScreen) /
                            (sum(ThroughScreen)+ sum(ThroughClin)) *100,2)

numsdf = df[-1,]
meanAge = round(mean(round(as.numeric(numsdf$PatientAge)/year)),2)
medianAge = median(round(as.numeric(numsdf$PatientAge)/year))
modeAge = getmode(round(as.numeric(numsdf$PatientAge)/year))


CostUtilDf <- df %>%
  select(TotalUtility,TotalCosts,MedicalCosts,SocietalCosts,PAIDCosts) %>%
  mutate_all(function(x) as.numeric(x)) %>%
  mutate(TotalCosts = SocietalCosts + MedicalCosts+PAIDCosts)

meanQALY <- mean(CostUtilDf$TotalUtility,na.rm = T) /year
meanTotalCosts <- mean(CostUtilDf$TotalCosts,na.rm = T)
meanMedicalCosts <- mean(CostUtilDf$MedicalCosts,na.rm = T)
meanSocietalCosts <- mean(CostUtilDf$SocietalCosts,na.rm = T)
meanPaidCosts <- mean(CostUtilDf$PAIDCosts,na.rm=T)
```

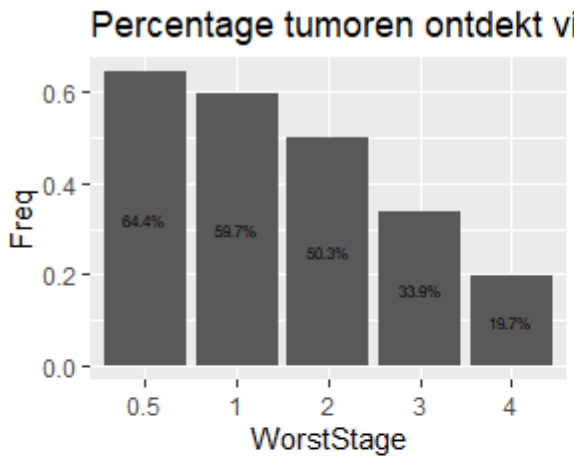
```
## KPIs ----

cat(paste0(strrep("_", 70),'\n',
           'OUTPUT KPIs', '\n',
           strrep("_", 70),'\n',
           'Patients generated: ',patients_total,'\n',
           'incidence percentage: ',inc_perc,'%','\n',
           'Percentage found through screening: ',percThroughScreen,'%','\n',
           'Patient Ages: mean:', meanAge,' mode: ',modeAge, ' median: ',
           medianAge, '\n',
           'Mean Qulity Adjusted Life Expectancy: ', meanQALY,'\n',
           'Mean Total Costs: ', meanTotalCosts,'\n',
           'Mean Medical Costs: ', meanMedicalCosts,'\n',
           'Mean Societal Costs: ', meanSocietalCosts,'\n',
           'Mean PAID Costs: ', meanPaidCosts,'\n'))
```

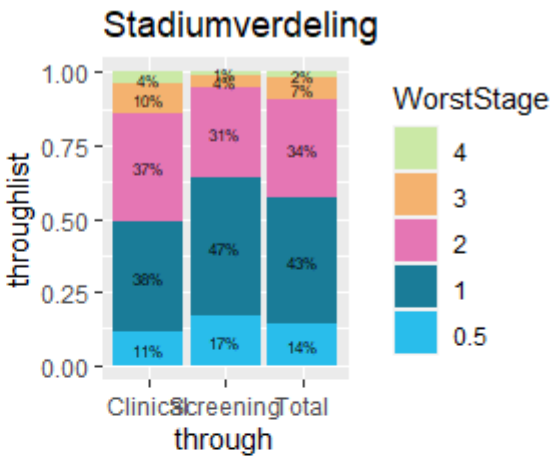
```
______________________________________________________________________
OUTPUT KPIs
______________________________________________________________________
Patients generated: 24969
incidence percentage: 13.64%
Percentage found through screening: 54.54%
Patient Ages: mean:80.27 mode: 82 median: 82
Mean Qulity Adjusted Life Expectancy: 71.743632132996
Mean Total Costs: 18479.1574273357
Mean Medical Costs: 6873.01761111848
Mean Societal Costs: 499.81419110646
Mean PAID Costs: 11106.3256251108
```

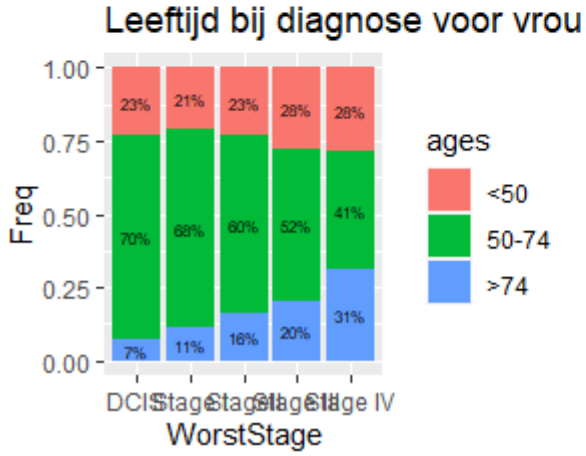
Grid plots

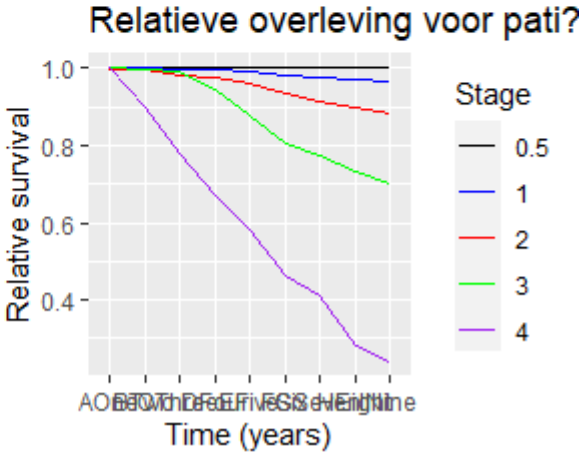
The following code shows two 2x2 grids containing the first eight plots, so that each simulation immediately provides output.

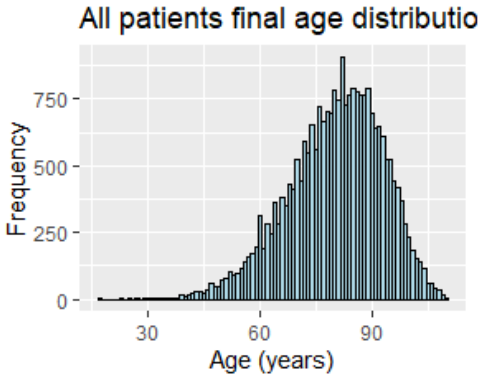
```
grid.arrange(p5,p6,p7,p8,nrow=2)
grid.arrange(p1,p2,p3,p4,nrow=2)
```

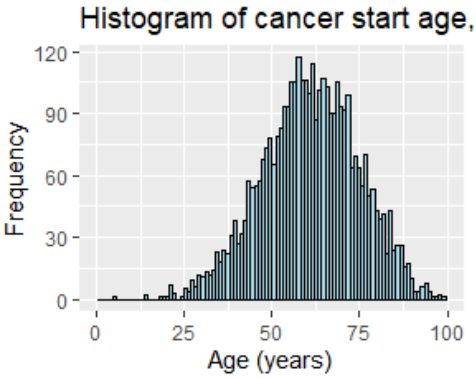

Percentage tumoren ontdekt vi
Freq
0.6
0.4
0.2
0.0
64.4%
59.7%
50.3%
33.9%
19.7%
0.5
1
2
3
4
WorstStage

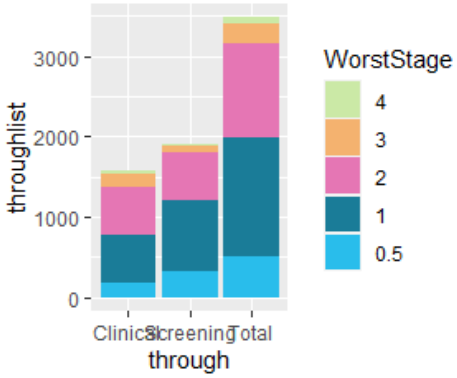

Stadiumverdeling
throughlist
1.00
0.75
0.50
0.25
0.00
4%
10%
37%
38%
11%
1%
4%
31%
47%
17%
2%
7%
34%
43%
14%
Clinical
Screening
Total
through
WorstStage
4
3
2
1
0.5

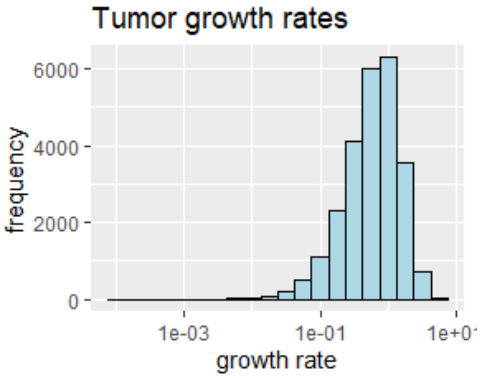

Leeftijd bij diagnose voor vrou
Freq
1.00
0.75
0.50
0.25
0.00
23%
70%
7%
21%
68%
11%
23%
60%
16%
28%
52%
20%
28%
41%
31%
DCIS
Stage I
Stage II
Stage III
Stage IV
WorstStage
ages
<50
50-74
>74


Relatieve overleving voor pati?
Relative survival
1.0
0.8
0.6
0.4
Time (years)
Stage
0.5
1
2
3
4


All patients final age distributio
Frequency
750
500
250
0
30
60
90
Age (years)


Histogram of cancer start age,
Frequency
120
90
60
30
0
0
25
50
75
100
Age (years)


throughlist
3000
2000
1000
0
Clinical
Screening
Total
through
WorstStage
4
3
2
1
0.5


Tumor growth rates
frequency
6000
4000
2000
0
1e-03
1e-01
growth rate

Bibliography (for full code explanation appendix)

Arrospide, A., Rue, M., van Ravesteyn, N. T., Comas, M., Soto-Gordoa, M., Sarriugarte, G., & Mar, J. (2016). Economic evaluation of the breast cancer screening programme in the Basque Country: retrospective cost-effectiveness and budget impact analysis. *BMC Cancer*, *16*(1). https://doi.org/10.1186/S12885-016-2386-Y

Integraal Kankercentrum Nederland. (2022). *Borstkankercijfers*. https://iknl.nl/borstkankercijfers

Integraal Kankercentrum Nederland (IKNL). (2022). *Synthetische dataset Nederlandse Kankerregistratie (NKR)*. Https://Iknl.Nl/NKR/Cijfers-Op-Maat/Synthetische-Data.

National Cancer Institute. (2022). *CISNET: Modeling Approach*. https://cisnet.cancer.gov/modeling/index.html

PAID 3.0. (n.d.). Retrieved March 19, 2023, from https://imta.shinyapps.io/PAID3code/

PriceWaterhouseCoopers Accountants N.V. (2022). *Jaarverslag Stichting Bevolkingsonderzoek Nederland 2021*.

Rafia, R., Brennan, A., Madan, J., Collins, K., Reed, M. W. R., Lawrence, G., Robinson, T., Greenberg, D., & Wyld, L. (2016). Modeling the Cost-Effectiveness of Alternative Upper Age Limits for Breast Cancer Screening in England and Wales. *Value in Health : The Journal of the International Society for Pharmacoeconomics and Outcomes Research*, *19*(4), 404–412. https://doi.org/10.1016/J.JVAL.2015.06.006

Trentham-Dietz, A., Alagoz, O., Chapman, C., Huang, X., Jayasekera, J., van Ravesteyn, N. T., Lee, S. J., Schechter, C. B., Yeh, J. M., Plevritis, S. K., & Mandelblatt, J. S. (2021). Reflecting on 20 years of breast cancer modeling in CISNET: Recommendations for future cancer systems modeling efforts. *PLoS Computational Biology*, *17*(6). https://doi.org/10.1371/JOURNAL.PCBI.1009020

Ucar, I., Hernández, J. A., Serrano, P., & Azcorra, A. (2018). Design and Analysis of 5G Scenarios with simmer: An R Package for Fast DES Prototyping. *IEEE Communications Magazine*, *56*(11), 145–151. https://doi.org/10.1109/MCOM.2018.1700960

van Haperen, V. R. (2018). *Factsheet Bevolkingsonderzoek Borstkanker*. https://www.rivm.nl/sites/default/files/2019-02/Factsheet%20borstkanker%202018%20DEF.pdf

van Luijt, P. A., Heijnsdijk, E. A. M., & de Koning, H. J. (2017). Cost-effectiveness of the Norwegian breast cancer screening program. *International Journal of Cancer*, *140*(4), 833–840. https://doi.org/10.1002/IJC.30513

Why "life expectancy" is a misleading summary of survival | Understanding Uncertainty. (n.d.). Retrieved April 10, 2023, from https://understandinguncertainty.org/why-life-expectancy-misleading-summary-survival

Wright, S. , Gray, E. , Rogers, G. , Donten, A. , Hainsworth, R. , & Payne, K. (2022). *MANC-RISK-SCREEN Version 1.0* (1.0). University of Manchester. https://research.manchester.ac.uk/en/datasets/manc-risk-screen-version-10

Zorginstituut Nederland. (2016). *Guideline for economic evaluations in healthcare*.