Introduction to RCSpin

Rob Dunne

July 26, 2014

Contents

1 RC Classes

RCSpin implements the CRC-SPIN model (Rutter, 2008; Rutter and Savarino, 2010) using the Reference Classes object oriented system of R. R how has 3 object systems, S3, S4, and Reference Classes (RC) systems.

RC is good for simulations with complex states. It has mutable objects – i.e changes don't make copies.

the class names are pointers, I think. Why this a good idea? I don know

2 genericSpin

This class implements a natural aging model on a group of people represented as a `study_group` of `Person` class objects. The main aim of this Class is to act as a framework for building more complex models through extension of this class. The simple natural aging model implemented in `updateSubject` acts as an example and place holder for subclasses to override with a more complex implementation of the function.

At the moment it subjects a population to the age specific mortality rates taken from a provided table. See figure **??**.

The function `help(GenericModel)` provides help about the methods. There is also `GenericModel(help)` and `GenericModel$help(updateSubject)` for help on specific methods.

```
> library(devtools)
> install_github("csiro-crc-spin/RCSpin",args=" -l ~/Downloads/temp")
> library(RCSpin,lib.loc="~/Downloads/temp")
> isS4(GenericModel) #TRUE
> GenericModel$methods()
> # [1] "callSuper"        "copy"             "doIteration"      "export"
> # [5] "field"            "getClass"         "getRefClass"      "import"
> # [9] "initFields"       "initialize"       "run"              "show"
> #[13] "show#envRefClass" "trace"            "untrace"          "updateSubject"
> #[17] "usingMethods"
>
> GenericModel$help()
> GenericModel$fields()
```

```
> #          iterations iteration_resolution          study_group
> #           "numeric"            "numeric"               "list"
> #       study_results
> #            "matrix"
```

3 crcSpin

```
> isS4(CrcSpinModel)
> #[1] TRUE
> showClass("CrcSpinModel")
> #Class "CrcSpinModel" [package "RCSpin"]
> #
> #Slots:
> #
> #Name:       .xData
> #Class: environment
> #
> #Extends:
> #Class "GenericModel", directly
> #Class "envRefClass", by class "GenericModel", distance 2
> #Class ".environment", by class "GenericModel", distance 3
> #Class "refClass", by class "GenericModel", distance 3
> #Class "environment", by class "GenericModel", distance 4, with explicit coerce
> #Class "refObject", by class "GenericModel", distance 4
> #
> #Known Subclasses: "DukesCrcSpinModel"
> getAnywhere(CrcSpinModel)
> cc<-CrcSpinModel$new(iterations=99, num_subjects=10,seed=123)
> cc$run()
> cc$study_group[1]
> cc$study_group[[1]]$colon
> #Spin Colon object of class "Colon"
>
> cc$study_group[[1]]$colon$sites[[1]]$initiated_in_year
> #[[1]]
> #Spin Person object of class "PersonWithColon"
> #Age:[1] 88
> #Sex:[1] "F"
> #State:[1] "deceased"
> #In treatment program:[1] "no"
> #Study id:[1] 1
> #Clinical history:
> #Spin ClinicalHistory object of class "ClinicalHistory"
> #Status:character(0)
> #Events:list()
> #Risk level:[1] "standard"
> #Colon clinical characteristic:[1] "clear"
> #Colon:
> #Spin Colon object of class "Colon"
>
>
> CrcSpinModel$methods()
> ##  [1] "adenomaParamsType"               "callSuper"
```

```
> ##  [3] "copy"                                "crcRiskParamsType"
> ##  [5] "doIteration"                         "export"
> ##  [7] "field"                               "getClass"
> ##  [9] "getModelResultSize"                  "getModelResultSize#GenericModel"
> ## [11] "getRefClass"                         "import"
> ## [13] "initFields"                          "initialize"
> ## [15] "initialize#GenericModel"             "modelSubjectDiseaseDevelopment"
> ## [17] "personWithColonType"                 "propegate_model_parameters"
> ## [19] "run"                                 "set_adenoma_modeling_parameters"
> ## [21] "set_crcrisk_modeling_parameters" "show"
> ## [23] "show#envRefClass"                    "subjectHasNotLeftStudy"
> ## [25] "testForAndTreatCRC"                  "trace"
> ## [27] "untrace"                             "updateSubject"
> ## [29] "updateSubject#GenericModel"          "usingMethods"
>
>
> CrcSpinModel$fields()
> #         iterations iteration_resolution          study_group
> #           "numeric"            "numeric"               "list"
> #       study_results     commencement_age crcrisk_model_params
> #            "matrix"            "numeric"      "CrcRiskParams"
> #adenoma_model_params
> #     "AdenomaParams"
>
>
> cc<-GenericModel$new(iterations=99, num_subjects=5)
> cc$trace(run,browser)
> cc$run()
> cc$untrace(run,browser)
> cc$trace(doIteration,browser)
> cc$run()
> cc$untrace(doIteration,browser)
> cc$trace(updateSubject,browser)
> cc$run()
>
```

4 DukesCrcSpinModel

```
> isS4(DukesCrcSpinModel)
> #[1] TRUE
> showClass("DukesCrcSpinModel")
> getAnywhere(DukesCrcSpinModel)
> cc<-DukesCrcSpinModel$new(iterations=99, num_subjects=10,seed=123)
> cc$run()
> cc$study_group[1]
> cc$study_group[[1]]$colon
> #Spin Colon object of class "DukesColon"
>
> cc$study_group[[1]]$colon$sites[[1]]$initiated_in_year
> #[1] 58
>
> DukesCrcSpinModel$methods()
> DukesCrcSpinModel$fields()
```

```
> alex<-Person$new()
> xx$trace(edit, browser)
> ## "Test",
> ## fields = list(
> ##         age="numeric",
> ##         type="character",
> ##         compliance="character",
> ##         result="character",
> ##         state="character"
> ##     )
> ##  methods =   summary ,       show
>
> ## "ClinicalHistory"
> ##     fields = list(
> ##         status="character",
> ##         events="list"
> ##         ),
> ## methods = summary ,show
>
> ## "Person",
> ##     fields = list(
> ##         age="numeric",
> ##         sex="character",
> ##         state="character", #"deceased","living", etc (as needed by subclasses)
> ##         in_treatment_program="character",
> ##         clinical_history="ClinicalHistory",
> ##         study_id="numeric",
> ##         random_seed_state="integer"
> ##     )
> ## methods = initialize
> ##         saveRNGState
> ##         restoreRNGState
> ##         modelDeathFromOtherCauses
> ##         summary
> ##         show
>
>
>
>
> ## "RiskParameters",
> ##             fields = list(
> ##                 baseline_risk="numeric",
> ##                 sex_linked_risk="numeric",
> ##                 age_risk="numeric"
> ##                 )
>
> ## "SymptomaticPresentation",
> ##             fields = list(
> ##                 age="numeric",
> ##                 cancer.stage="character"
> ##                 )
>
>
```

```
>
>
> ##                                updateSubject = function (subject) {
> ##                                      if (subject$state=="living") {
> ##                                          subject$restoreRNGState()
> ##                                          subject$age = subject$age + iteration_resolution
> ##                                          subject$modelDeathFromOtherCauses()
> ##                                          subject$saveRNGState()
> ##                                      }
> ##                                      return((subject$state=="living"))
> ##                                  }
>
> ################
>
> alex<-PersonWithColon$new()
> alex$age #1
> alex$clinical_history
> alex$in_treatment_program
> alex$state
> alex$colon_clinical
> iFOBT.screening(alex)
> alex$clinical_history$events<- lappend(alex$clinical_history$events,
+                                            Test$new(
+                                                age=alex$age,
+                                                type="iFOBT",
+                                                compliance="accept",
+                                                result="positive",
+                                                state= "adenoma")
+                                            )
> #I have put iFOBT.screening() function into the CRCSpinModel class. Apparently I
> #could have put it in the Person class
>
> source("crcSpin.R")
> model<-CrcSpinModel$new(iterations=99, num_subjects=5)
> alex<-PersonWithColon$new()
> model$iFOBTscreening(alex)
> alex
> model$NBCSP(alex)
> #Alternatively, you could just use one of the people already in the model's study_group.
>
>
> model$iFOBTscreening(model$study_group[[1]])
> model$study_group[[1]]
> #Spin Person object of class "PersonWithColon"
> #Age:[1] 20
> #Sex:[1] "F"
> #State:[1] "living"
> #In treatment program:[1] "no"
> #Study id:[1] 1
> #Clinical history:
> #Spin ClinicalHistory object of class "ClinicalHistory"
> #Status:character(0)
> #Events:[[1]]
```

```
> #Spin Test object of class "Test"
> #Age:[1] 20
> #Type of test:[1] "iFOBT"
> #Compliance:[1] "accept"
> #Result:[1] "negative"
> #State:[1] "TN"
> #
> #Risk level:[1] "standard"
> #Colon clinical characteristic:[1] "clear"
> #Colon:
> #Spin Colon object of class "Colon"
>
>
>
>
> ##############################
> source("dukesCrcSpin.R")
> alex<-DukesPersonWithColon$new()
> alex$age #1
> alex$clinical_history
> alex$in_treatment_program
> alex$state
> alex$colon_clinical
> iFOBTscreening(alex) #Error: could not find function "iFOBTscreening"
> model<-DukesCrcSpinModel$new(iterations=99, num_subjects=5)
> model$iFOBTscreening(alex)
> alex
> model$NBCSP(alex)  #0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> alex
> cc$study_group[[1]]$colon
> #Spin Colon object of class "Colon"
>
> cc$study_group[[10]]$colon$sites[[1]]$initiated_in_year
> #cc<-CrcSpinModel$new(iterations=99, num_subjects=50000)
> #cc$run()
> #save(cc,file="script8_cc.Rdata")
> res2<-cc$study_results
> ########################################
>
>
>
>
> y<-res.M[,40:44]  #person1@colon@stage  includes CRC and pre-symptomatic
> x<-c(1:99)
> z<-c(1,2,3,4,5) #z <- val2col(apply(y,2,max), col=COLS)
> #ylim=c(0, 1.2*max(apply(y,1,sum), na.rm=TRUE))
>
> png("t7.png")
> par(mfrow=c(3,1))
> plot.stacked(x,y, xlim=c(0, 100), ylim=c(0, 500),
+               yaxs="i", col=z, border="white", lwd=0.5,order.method="as.is")
> title(main="all CRC, population of 25000 males")
> y<-res.M[,54:57]     #   found from test or symptoms
```

```
> z<-c(1,2,3,4)
> plot.stacked(x,y, xlim=c(0, 100),  ylim=c(0, 500),
+               yaxs="i", col=z, border="white", lwd=0.5,order.method="as.is")
> title(main="entering treatment -- detected by NBCSP")
> y<-res.M[,12:16]     #                        if (object@colon@state=="pre symptomatic CRC")
> z<-c(1,2,3,4,5)
> plot.stacked(x,y, xlim=c(0, 100), ylim=c(0, 500),
+               yaxs="i", col=z, border="white", lwd=0.5,order.method="as.is")
> title(main="undetected  CRC")
> dev.off()
> abline(v=c(55,60,65,70,75))
> cc<-DukesCrcSpinModel$new(iterations=99, num_subjects=5)
> cc$trace(NBCSP,browser)
> cc$run()
> #cc$untrace(run,browser)
>
>
>
>
>
> ##############################################################################
> #on tamar
> library(SpinModels)
> #Loading required package: Rcpp
> #Loading required package: RcppArmadillo
> rr<-CrcSpinModel$new(99,50000,125)
> rr$run()
> res<-rr$study_results
> res.M<-res[(1:99)*2-1,]
> res.F<-res[(1:99)*2,]
> apply(res.M,2,sum)
> # [1] 486534  46117  10907   2432      0 245146  38407   8068   2432    797
> #[11]   2526   2309   1237   2627   1004  21975   2432   2432
> #ylim=c(0, 1.2*max(apply(y,1,sum), na.rm=TRUE))
>
> #1         count(adenoma.state=="adenoma");         A count of adenomas in all patients of t
> #2         count(adenoma.state=="large adenoma");         A count of adenomas in all patient
> #3         count(adenoma.state=="pre symptomatic CRC");         A count of adenomas in all p
> #4         count(person.colon_clinical=="CRC");         A count of patients that have colon_
> #5         count(colon.state=="clear");         A count of how many peoples colons were in s
> #6         count(colon.state=="adenoma");         A count of how many peoples colons were in
> #7         count(colon.state=="large adenoma");         A count of how many peoples colons w
> #8         count(colon.state=="pre symptomatic CRC");         A count of how many peoples co
> #9         count(colon.state=="CRC");         A count of how many peoples colons were in sta
>
>
>
>
> y<-res.M[,7:9]  #person1@colon@stage  includes CRC and pre-symptomatic
> x<-c(1:99)
> z<-c(1,2,3) #z <- val2col(apply(y,2,max), col=COLS)
> #png("t5.png")
> par(mfrow=c(3,1))
```

```
> plot.stacked(x,y, xlim=c(0, 100), ylim=c(0, 1000),
+              yaxs="i", col=z, border="white", lwd=0.5,order.method="as.is")
> title(main="large adenomas, CRC, symptomatic CRC, population of 25000 males")
> #dev.off()
>
>
> y<-res.M[,12:16]     #                        if (object@colon@state=="pre symptomatic CRC")
> z<-c(1,2,3,4,5)
> plot.stacked(x,y, xlim=c(0, 100), ylim=c(0, 1000),
+              yaxs="i", col=z, border="white", lwd=0.5,order.method="as.is")
> title(main="undetected  CRC")
> dev.off()
> abline(v=c(55,60,65,70,75))
> number.F<-25000
> number.M<-25000
> tt <- res.F+res.M
> apply(tt,2,sum)
> ##########################################################################
> #http://seer.cancer.gov
>
> #Based on rates from 2008-2010, 4.82% of men and women born today will
> #be diagnosed with cancer of the colon and rectum at some time during
> #their lifetime.
>
> sum(res.F[,4]+res.M[,4])/(number.F+number.M)
> #[1]   0.1136
>
> #1.91% of men will develop cancer of the colon
> #and rectum between their 50th and 70th birthdays compared to 1.41% for
> #women.
>
> sum(res.F[50:70,4])/(number.F-cumsum(res.F[,16]))[60]
> # 0.05308885
> sum(res.M[50:70,4])/(number.M-cumsum(res.M[,16]))[60]
> #] 0.03492438
>
> # adenoma (adenomas + large adenomsa + pre symptomatic CRC)
>  mean((res.M[40:49,6]+res.F[40:49,6]+res.M[40:49,7]+res.F[40:49,7]+res.M[40:49,8]+res.F[40:4
> #] 0.16715
> mean((res.M[50:75,6]+res.F[50:75,6]+res.M[50:75,7]+res.F[50:75,7]+res.M[50:75,8]+res.F[50:75
>  # 0.21818
> mean((res.M[76:80,6]+res.F[76:80,6]+res.M[76:80,7]+res.F[76:80,7]+res.M[76:80,8]+res.F[76:80
> #  0.216136
>
> # proportion of Adenomas greater the 10 mm
> mean(res.F[-c(1:40),7]/(res.F[-c(1:40),7]+res.F[-c(1:40),6]))
> #0.1589128
>
>
>
>
> load("res_rcpp.Rdata")
> number.M<- 1450000/2
```

```
> number.F<- 1450000/2
> #http://seer.cancer.gov
> #Based on rates from 2008-2010, 4.82% of men and women born today will
> #be diagnosed with cancer of the colon and rectum at some time during
> #their lifetime.
>
> sum(res.F[,4]+res.M[,4])/(number.F+number.M)
> #[1]   0.08714
>
> #1.91% of men will develop cancer of the colon
> #and rectum between their 50th and 70th birthdays compared to 1.41% for
> #women.
>
> sum(res.F[50:70,4])/(number.F-cumsum(res.F[,16]))[60]
> # 0.03027862
> sum(res.M[50:70,4])/(number.M-cumsum(res.M[,16]))[60]
> #]  0.03261103
>
> # adenoma (adenomas + large adenomsa + pre symptomatic CRC)
>  mean((res.M[40:49,6]+res.F[40:49,6]+res.M[40:49,7]+res.F[40:49,7]+res.M[40:49,8]+res.F[40:4
> #] 0.16715
> mean((res.M[50:75,6]+res.F[50:75,6]+res.M[50:75,7]+res.F[50:75,7]+res.M[50:75,8]+res.F[50:75
>  # 0.21818
> mean((res.M[76:80,6]+res.F[76:80,6]+res.M[76:80,7]+res.F[76:80,7]+res.M[76:80,8]+res.F[76:80
> #  0.216136
>
>
>
> # proportion of Adenomas greater the 10 mm
> mean(res.F[-c(1:40),7]/(res.F[-c(1:40),7]+res.F[-c(1:40),6]))
> #0.1589128
>
>
>
>
> ##############################################################################
> ## http://www.r-bloggers.com/data-mountains-and-streams-stacked-area-plots-in-r/
> ##plot.stacked makes a stacked plot where each y series is plotted on top
> ##of the each other using filled polygons
> ##
> ##Arguments include:
> ## 'x' - a vector of values
> ## 'y' - a matrix of data series (columns) corresponding to x
> ## 'order.method' = c("as.is", "max", "first")
> ## "as.is" - plot in order of y column
> ## "max" - plot in order of when each y series reaches maximum value
> ## "first" - plot in order of when each y series first value > 0
> ## 'col' - fill colors for polygons corresponding to y columns (will recycle)
> ## 'border' - border colors for polygons corresponding to y columns (will recycle) (see ?poly
> ## 'lwd' - border line width for polygons corresponding to y columns (will recycle)
> ## '...' - other plot arguments
>
> plot.stacked <- function(x, y,order.method = "as.is",ylab="", xlab="",
```

```
+                           border = NULL, lwd=1,
+                           col=rainbow(length(y[1,])),
+                           ylim=NULL,
+                           ...
+                           ){
+
+     if(sum(y < 0) > 0) error("y cannot contain negative numbers")
+
+     if(is.null(border)) border <- par("fg")
+     border <- as.vector(matrix(border, nrow=ncol(y), ncol=1))
+     col <- as.vector(matrix(col, nrow=ncol(y), ncol=1))
+     lwd <- as.vector(matrix(lwd, nrow=ncol(y), ncol=1))
+
+     if(order.method == "max") {
+         ord <- order(apply(y, 2, which.max))
+         y <- y[, ord]
+         col <- col[ord]
+         border <- border[ord]
+     }
+
+     if(order.method == "first") {
+         ord <- order(apply(y, 2, function(x) min(which(r>0))))
+
+
+         y <- y[, ord]
+         col <- col[ord]
+         border <- border[ord]
+     }
+
+     top.old <- x*0
+     polys <- vector(mode="list", ncol(y))
+     for(i in seq(polys)){
+         top.new <- top.old + y[,i]
+         polys[[i]] <- list(x=c(x, rev(x)), y=c(top.old, rev(top.new)))
+         top.old <- top.new
+     }
+
+     if(is.null(ylim)) ylim <- range(sapply(polys, function(x) range(x$y, na.rm=TRUE)), na.rm
+     plot(x,y[,1], ylab=ylab, xlab=xlab, ylim=ylim, t="n", ...)
+     for(i in seq(polys)){
+         polygon(polys[[i]], border=border[i], col=col[i], lwd=lwd[i])
+     }
+
+ }
> #plot.stream makes a "stream plot" where each y series is plotted
> #as stacked filled polygons on alternating sides of a baseline.
> #
> #Arguments include:
> #'x' - a vector of values
> #'y' - a matrix of data series (columns) corresponding to x
> #'order.method' = c("as.is", "max", "first")
> # "as.is" - plot in order of y column
> # "max" - plot in order of when each y series reaches maximum value
```

```
> # "first" - plot in order of when each y series first value > 0
> #'center' - if TRUE, the stacked polygons will be centered so that the middle,
> #i.e. baseline ("g0"), of the stream is approximately equal to zero.
> #Centering is done before the addition of random wiggle to the baseline.
> #'frac.rand' - fraction of the overall data "stream" range used to define the range of
> #random wiggle (uniform distrubution) to be added to the baseline 'g0'
> #'spar' - setting for smooth.spline function to make a smoothed version of baseline "g0"
> #'col' - fill colors for polygons corresponding to y columns (will recycle)
> #'border' - border colors for polygons corresponding to y columns (will recycle) (see ?polyg
> #'lwd' - border line width for polygons corresponding to y columns (will recycle)
> #'...' - other plot arguments
>
> plot.stream <- function( x, y, order.method = "as.is", frac.rand=0.1,
+                          spar=0.2, center=TRUE, ylab="", xlab="", border = NULL, lwd=1,
+                          col=rainbow(length(y[1,])), ylim=NULL, ...  ){
+
+     if(sum(y < 0) > 0) error("y cannot contain negative numbers")
+
+     if(is.null(border)) border <- par("fg")
+     border <- as.vector(matrix(border, nrow=ncol(y), ncol=1))
+     col <- as.vector(matrix(col, nrow=ncol(y), ncol=1))
+     lwd <- as.vector(matrix(lwd, nrow=ncol(y), ncol=1))
+
+     if(order.method == "max") {
+         ord <- order(apply(y, 2, which.max))
+         y <- y[, ord]
+         col <- col[ord]
+         border <- border[ord]
+     }
+
+     if(order.method == "first") {
+         ord <- order(apply(y, 2, function(x) min(which(r>0))))
+         y <- y[, ord]
+         col <- col[ord]
+         border <- border[ord]
+     }
+
+     bottom.old <- x*0
+     top.old <- x*0
+     polys <- vector(mode="list", ncol(y))
+     for(i in seq(polys)){
+         if(i %% 2 == 1){ #if odd
+             top.new <- top.old + y[,i]
+             polys[[i]] <- list(x=c(x, rev(x)), y=c(top.old, rev(top.new)))
+             top.old <- top.new
+         }
+         if(i %% 2 == 0){ #if even
+             bottom.new <- bottom.old - y[,i]
+             polys[[i]] <- list(x=c(x, rev(x)), y=c(bottom.old, rev(bottom.new)))
+             bottom.old <- bottom.new
+         }
+ }
+
```

```
+     ylim.tmp <- range(sapply(polys, function(x) range(x$y, na.rm=TRUE)), na.rm=TRUE)
+     outer.lims <- sapply(polys, function(r) rev(r$y[(length(r$y)/2+1):length(r$y)]))
+     mid <- apply(outer.lims, 1, function(r) mean(c(max(r, na.rm=TRUE), min(r, na.rm=TRUE)),
+                                          #center and wiggle
+     if(center) {
+         g0 <- -mid + runif(length(x), min=frac.rand*ylim.tmp[1], max=frac.rand*ylim.tmp[2])
+     } else {
+         g0 <- runif(length(x), min=frac.rand*ylim.tmp[1], max=frac.rand*ylim.tmp[2])
+     }
+     fit <- smooth.spline(g0 ~ x, spar=spar)
+
+     for(i in seq(polys)){
+         polys[[i]]$y <- polys[[i]]$y + c(fit$y, rev(fit$y))
+     }
+
+     if(is.null(ylim)) ylim <- range(sapply(polys, function(x) range(x$y, na.rm=TRUE)), na.rm
+     plot(x,y[,1], ylab=ylab, xlab=xlab, ylim=ylim, t="n", ...)
+     for(i in seq(polys)){
+         polygon(polys[[i]], border=border[i], col=col[i], lwd=lwd[i])
+     }
+
+ }
> #this function converts a vector of values("z") to a vector of color
> #levels. One must define the number of colors. The limits of the color
> #scale("zlim") or the break points for the color changes("breaks") can
> #also be defined. when breaks and zlim are defined, breaks overrides zlim.
> val2col<-function(z, zlim, col = heat.colors(12), breaks){
+     if(!missing(breaks)){
+         if(length(breaks) != (length(col)+1)){stop("must have one more break than colour")}
+     }
+     if(missing(breaks) & !missing(zlim)){
+         zlim[2] <- zlim[2]+c(zlim[2]-zlim[1])*(1E-3)#adds a bit to the range in both directi
+         zlim[1] <- zlim[1]-c(zlim[2]-zlim[1])*(1E-3)
+         breaks <- seq(zlim[1], zlim[2], length.out=(length(col)+1))
+     }
+     if(missing(breaks) & missing(zlim)){
+         zlim <- range(z, na.rm=TRUE)
+         zlim[2] <- zlim[2]+c(zlim[2]-zlim[1])*(1E-3)#adds a bit to the range in both directi
+         zlim[1] <- zlim[1]-c(zlim[2]-zlim[1])*(1E-3)
+         breaks <- seq(zlim[1], zlim[2], length.out=(length(col)+1))
+     }
+     CUT <- cut(z, breaks=breaks)
+     colorlevels <- col[match(CUT, levels(CUT))] # assign colors to heights for each point
+     return(colorlevels)
+ }
> ## set.seed(1)
> ## m <- 500
> ## n <- 30
> ## x <- seq(m)
> ## y <- matrix(0, nrow=m, ncol=n)
> ## colnames(y) <- seq(n)
> ## for(i in seq(ncol(y))){
> ## mu <- runif(1, min=0.25*m, max=0.75*m)
```

```
> ## SD <- runif(1, min=5, max=20)
> ## TMP <- rnorm(1000, mean=mu, sd=SD)
> ## HIST <- hist(TMP, breaks=c(0,x), plot=FALSE)
> ## fit <- smooth.spline(HIST$counts ~ HIST$mids)
> ## y[,i] <- fit$y
> ## }
> ## y <- replace(y, y<0.01, 0)
>
>
> ## #Plot Ex. 1 - Color by max value
> ## pal <- colorRampPalette(c(rgb(0.85,0.85,1), rgb(0.2,0.2,0.7)))
> ## BREAKS <- pretty(apply(y,2,max),8)
> ## LEVS <- levels(cut(1, breaks=BREAKS))
> ## COLS <- pal(length(BREAKS )-1)
> ## z <- val2col(apply(y,2,max), col=COLS)
>
> ## #plot.stacked(x,y, xlim=c(100, 400), ylim=c(0, 1.2*max(apply(y,1,sum), na.rm=TRUE)),
> ## #             yaxs="i", col=z, border="white", lwd=0.5)
>
>
>
>
>
>
> source("dukesCrcSpin.R")
> set.seed(123)
> dd <- DukesCrcSpinModel$new(iterations=99, num_subjects=50000)
> dd$run()
> res7<-dd$study_results
> ############################
> t2<-read.table(file="t2.csv",sep=",",header=TRUE)#R5 model,  no intervention, Dukes staging.
>                                                  #DukesCrcSpinModel$new(iterations=99, num_s
> t3<-read.table(file="t3.csv",sep=",",header=TRUE)# S4 model, no intervention, Dukes
> t2<-as.vector(apply(t2,2,sum))
> t3<-as.vector(apply(t3,2,sum))
> source("output_names")
> cbind(t2,t3,nn)
> ##       t2        t3         crc-spin          nn
> ##  [1,] "552896"  "317874"  486534     "1 adenoma                    object@colon@sites -- s
> ##  [2,] "36870"   "32164"    46117     "2 large adenoma              a person can be in more
> ##  [3,] "17719"   "9095"     10907     "3 pre symptomatic CRC      adenoma  large adenoma a
> ##  [4,] "2165"    "108"                "4 deceased"
>
> #the R5 model seems to have an excess number of deaths
> #it also seems to have higher numbers of everythign -- but this might be natural variation
> #
> #          if (length(colon$sites)>0){
> #               tt<-lapply(colon$sites,f<-function(x){(x$state=="deceased")})
> #               tt<-unlist(tt)
> #               aa[4]<-sum(tt)
> #            }
>
>
```

```
>
> plot(log(t2),log(t3))
>
>
> ##############################################################################################
> #http://seer.cancer.gov
>
> #Based on rates from 2008-2010, 4.82% of men and women born today will
> #be diagnosed with cancer of the colon and rectum at some time during
> #their lifetime.
> #crc-spin
> #(2432  +  797)/50000
> #[1] 0.06458
>
>
> # S4 model, no intervention, Dukes
> #(1644+108)/25000
> #[1] 0.07008
>
> # R5 model, no intervention, Dukes
>
>
>
```

References

Rutter, C. (2008). Group health research institute (CRC-SPIN). Version: HI.001.10242008.41523. Document generated: 10/24/2008 `https://cisnet.flexkb.net/mp/pub/cisnet_colorectal_ghc_profile.pdf`.

Rutter, C. M. and Savarino, J. E. (2010). An evidence-based microsimulation model for colorectal cancer: validation and application. *Cancer Epidemiol Biomarkers Prev*, 19(8):1992–2002. Epub 2010 Jul 20.