Simulation in Drug Development

Greg Cicconetti, Ph.D.

Saturday, December 06, 2014

Contents

1 Introduction

This document contains R code to accompany a subset of the topics planned for the Deming 2014 Conference Tutorial Session: *Simulation in Drug Development*.

1.1 Loading packages and referencing functions

The following packages are used throughout this document.

```
require(ggplot2)
require(MASS)
require(chron)
require(lubridate)
require(plyr)
require(reshape2)
require(gridExtra)
```

```r
require(scales)
require(mvtnorm)
require(survival)
# # Use this for powerpoint slides
#  theme_set(theme_grey(base_size = 18) %+replace%
#          theme(plot.margin=unit(c(0,0,0,0), units="line"),
#               axis.text=element_text(color="black")))
# Use this for pdf
theme_set(theme_grey())
```

For more information on these packages, please see refer to their help files and documentation. Citations for these packages are included in the Reference section.

2 Monopoloy

2.1 The problem

Characterize the number of rolls of the dice required before landing in jail in a game of Monopoly.

2.2 Problem restatement

The environment: 40 spaces around square board. Play begins on space 0 (= space 40). An iteration rolls 2 dice repeatedly until one of the following conditions occurs:

- Land on Go to Jail: Space: 30
- Roll doubles 3 times in a row: Go to Jail
- Chance cards spaces 7, 22, 36: Go to Jail with probability 1/16
- Community Chest cards spaces: 2, 17, 33: Go to Jail with probability 1/16

Note that this problem restatement is a simplification of the real world play. See e.g., The Shortest Possible Game of Monopoly: 21 Seconds

2.3 Initializing values and setting constants

Some constants and initial values are set. These reflect constants that will not change during game play.

```r
monopoly.spaces <- 1:40
go.to.jail <- 30
chance <- c(7, 22, 36)
com.chest <- c(2, 17, 33)
```

2.4 A function to simulate a single iteration

This function continues to roll the dice until conditions send player to jail. This function is arguably inefficient. E.g., card1 and card2 are superfluously drawn with each roll. In general, code optimization is another step in the simulation development workflow.

```r
rollToJail <- function(){
  # Initializing objects within the function
  # These items will change during game play
  die1 <- die2 <- roll <- c()
  status <- "free"
  post <- 0
  post <- 0
  for.switch <- 1
  reason <- "Rolling Starts"
  # The repetition
  repeat {
    die1 <- c(sample(1:6, size=1), die1)
    die2 <- c(sample(1:6, size=1), die2)
    roll <- c(sum(die1[1], die2[1]), roll)
    post <- c((post[1] + roll[1]) %% 40, post)
    card1 <- (runif(1) < 1/16)*1
    card2 <- (runif(1) < 1/16)*1
    forswitch <- min(
      which(
        c(
          !((length(roll)>3 & (sum(head(die1,3)==head(die2,3)) ==3 )) |
              (post[1] == 30) |
              (post[1] %in% chance & card1)==T|
              (post[1] %in% com.chest & card2==1)==T ),
          (length(roll) > 3 & (sum(head(die1,3)==head(die2,3)) ==3)),
          (post[1] == 30),
          (post[1] %in% chance & card1 == 1) == T,
          (post[1] %in% com.chest & card2 == 1) == T)
      )
    )
    # Identifying the status following the roll and determining whether to stop
    reason <- c(
      switch(forswitch,
             "1" = "Rolling continues",
             "2" = "3 Doubles in a Row",
             "3" = "Board Square",
             "4" = "Chance card",
             "5" = "Community Chest Card"), reason)
    if(reason[1]!="Rolling continues") break
  }

  # Collecting the results
  temp <<- data.frame(Board.Position=post[-length(post)],
                      Die.1=die1,Die.2=die2,
                      Sum=roll, Attempt=rev(1:length(roll)),
                      Reason=reason[1:length(roll)])

  return(temp)
}
```

2.5 Setting a seed and reviewing a single iteration

The function base::set.seed initializes the random number generator ensuring examples are reproducible.

```
set.seed(1)
rollToJail()
```

```
   Board.Position Die.1 Die.2 Sum Attempt           Reason
1              30     5     1   6      10     Board Square
2              24     3     2   5       9 Rolling continues
3              19     6     3   9       8 Rolling continues
4              10     2     3   5       7 Rolling continues
5               5     6     2   8       6 Rolling continues
6              37     5     6  11       5 Rolling continues
7              26     5     3   8       4 Rolling continues
8              18     4     1   5       3 Rolling continues
9              13     2     6   8       2 Rolling continues
10              5     2     3   5       1 Rolling continues
```

2.6 Replicate the iterations

The function plyr::rdply replicates the rollToJail function 10000 times. Non-Windows users can remove the .progress argument. The function base::Sys.time is used to time the simulation.

```
set.seed(8675309)
start.time <- Sys.time()
results <- rdply(.n = 10000, .expr = rollToJail,
                 .progress = progress_win(title="Working..."))
stop.time <- Sys.time()
cat(paste("Simulation took:", round(difftime(stop.time, start.time),2)))
```

```
Simulation took: 50.68
```

2.7 Inspecting the results

The object returned is a data.frame with 264823 rows holding the full details of the 100000 replicates of the rollToJail function. The functions base::head, base::tail and utils::dim are used to inspect.

```
head(results)
```

```
  .n Board.Position Die.1 Die.2 Sum Attempt           Reason
1  1             30     4     2   6      16     Board Square
2  1             24     3     2   5      15 Rolling continues
3  1             19     5     3   8      14 Rolling continues
4  1             11     3     5   8      13 Rolling continues
5  1              3     3     1   4      12 Rolling continues
6  1             39     6     6  12      11 Rolling continues
```

```
tail(results)
```

```
          .n Board.Position Die.1 Die.2 Sum Attempt           Reason
264818 10000              6     4     1   5       6 Rolling continues
264819 10000              1     3     5   8       5 Rolling continues
264820 10000             33     4     2   6       4 Rolling continues
```

```
264821 10000             27     6     3   9       3 Rolling continues
264822 10000             18     6     6  12       2 Rolling continues
264823 10000              6     1     5   6       1 Rolling continues
```

```
dim(results)
```

```
[1] 264823      7
```

2.8 Post-processing results

The base functions table, prop.table, and subset, are used to summarize the 10000 iterations.

```
ptable <- data.frame(prop.table(table(
  subset(results, Reason!="Rolling continues")$Reason)))
names(ptable) <- c("Reason", "Prop")
head(ptable) # We're not interested in 'Rolling continues' row.
```

```
                Reason   Prop
1         Board Square 0.6657
2    Rolling continues 0.0000
3   3 Doubles in a Row 0.0939
4 Community Chest Card 0.1226
5          Chance card 0.1178
```

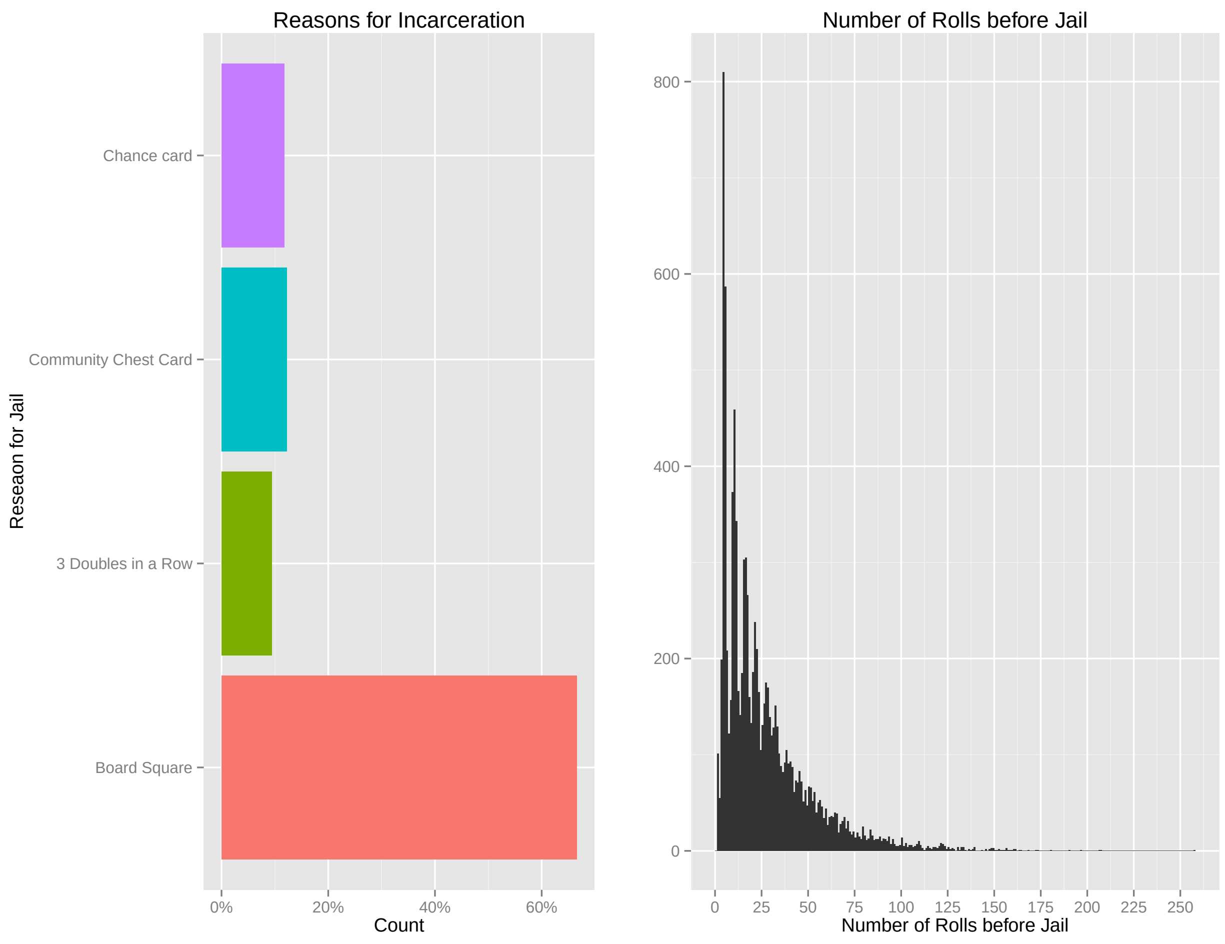
```
ptable <- subset(ptable, Prop>0)
print(ptable)
```

```
                Reason   Prop
1         Board Square 0.6657
3   3 Doubles in a Row 0.0939
4 Community Chest Card 0.1226
5          Chance card 0.1178
```

2.9 Visualizing the results

The ggplot2 package is used to create some figures. The function gridExtra::grid.arrange juxtaposes the figures.

```
p1 <- ggplot(data=ptable, aes(x=Reason, y=Prop, fill=Reason)) +
  geom_bar(stat="identity") +
  labs(x="Reseaon for Jail", y="Count") +
  ggtitle("Reasons for Incarceration") +
  coord_flip() +
  scale_y_continuous(label=percent) +
  guides(fill=FALSE)

p2 <- ggplot(data=subset(results,
                         Reason!="Rolling continues"), aes(x=Attempt)) +
  geom_histogram(binwidth=1) +
  scale_x_continuous(breaks=seq(0,300,25)) +
  labs(x="Number of Rolls before Jail", y="") +
  ggtitle("Number of Rolls before Jail")

grid.arrange(p1,p2, nrow=1)
```



2.10 Comparing simulated results to a known candidate distribution

We're rolling dice until a particular event is realized. Given the nature of the experiment, if presented with data, one may be tempted to appeal to the geometric distribution. All models are wrong, but how wrong is the geometric model? The MASS::fitdistr function is used to find the MLE *presuming the number of rolls to until jail follows a geometric distribution.*

```
geometric.p <- fitdistr(subset(results, Reason!="Rolling continues")$Attempt,
                        densfun="geometric")$estimate
cat("MLE estimate for geometric success probability:", geometric.p)
```

```
MLE estimate for geometric success probability: 0.03639
```

A data.frame is built to hold values of the geometric probability mass function evaluated through 251.

```r
geometric.mass <- data.frame(x=1:251)
geometric.mass$pmf_x <- dgeom(geometric.mass$x, geometric.p)
head(geometric.mass)
```

```
  x   pmf_x
1 1 0.03506
2 2 0.03379
3 3 0.03256
4 4 0.03137
5 5 0.03023
6 6 0.02913
```

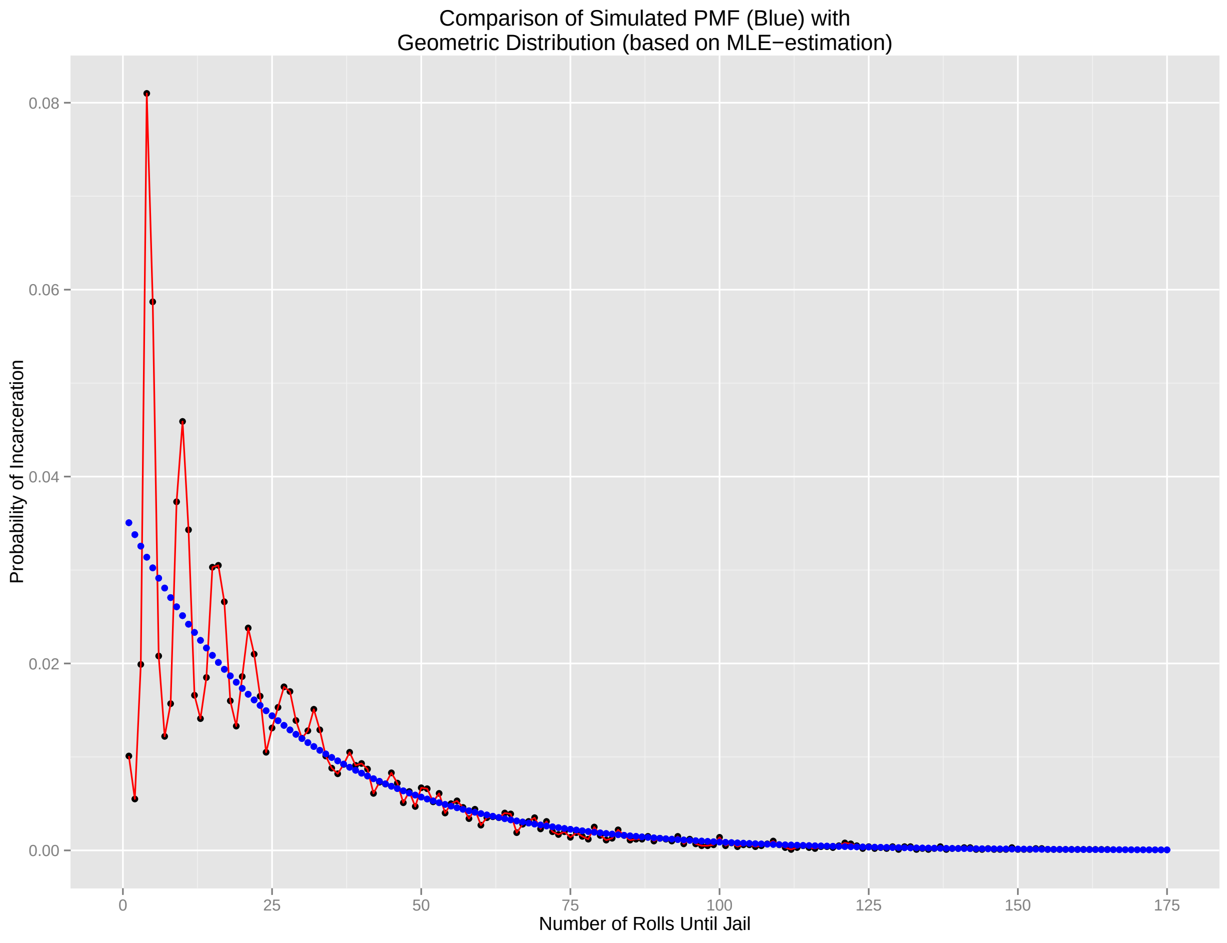
The relative frequencies of the observed data are computed.

```r
Attempt.tally <- data.frame(table(subset(results,
                                         Reason!="Rolling continues")$Attempt))
Attempt.tally$prop <- Attempt.tally$Freq/sum(Attempt.tally$Freq)
head(Attempt.tally)
```

```
  Var1 Freq   prop
1    1  101 0.0101
2    2   55 0.0055
3    3  199 0.0199
4    4  810 0.0810
5    5  587 0.0587
6    6  208 0.0208
```

```r
p <- ggplot(data=Attempt.tally, aes(x=as.numeric(Var1),
                                    y=prop, group=factor(1))) +
  geom_point(data=geometric.mass,  aes(x=x,y=pmf_x),color="blue") +
  geom_point() +
  scale_x_continuous(breaks=seq(0,300,25), limits=c(0,175)) +
  geom_line(color="red") +
  geom_point(data=geometric.mass, aes(x=x,y=pmf_x),color="blue") +
  labs(x="Number of Rolls Until Jail", y="Probability of Incarceration") +
  ggtitle(paste0("Comparison of Simulated PMF (Blue) with\n",
                 "Geometric Distribution (based on MLE-estimation)"))
```

```r
print(p)
```



Comment: The geometric distribution fails as a model because in this experiment the memory-less property is violated.

3 Mixture sampling and data-based densitiy

The following function leverages rbinom and rnorm (stats package) to sample from the mixture of normal distributions.

```
rmix2norm <- function(n=10, mean1=0, sd1=3, mean2=10,sd2=2, prop=1/3){
  bernoulli.sample <- rbinom(n=n, size=1, prob=prop)
  return(rnorm(n=n, mean=mean1*bernoulli.sample + mean2*(1-bernoulli.sample),
               sd=sd1*bernoulli.sample + sd2*(1-bernoulli.sample)
               ))
  }
```

Tip: Do a check to ensure the mean of the sample data corresponds with the mean from the target distribution. E.g., in this case one could derive the expected value and variance of the normal mixture and confirm summary statistics based on large samples align.

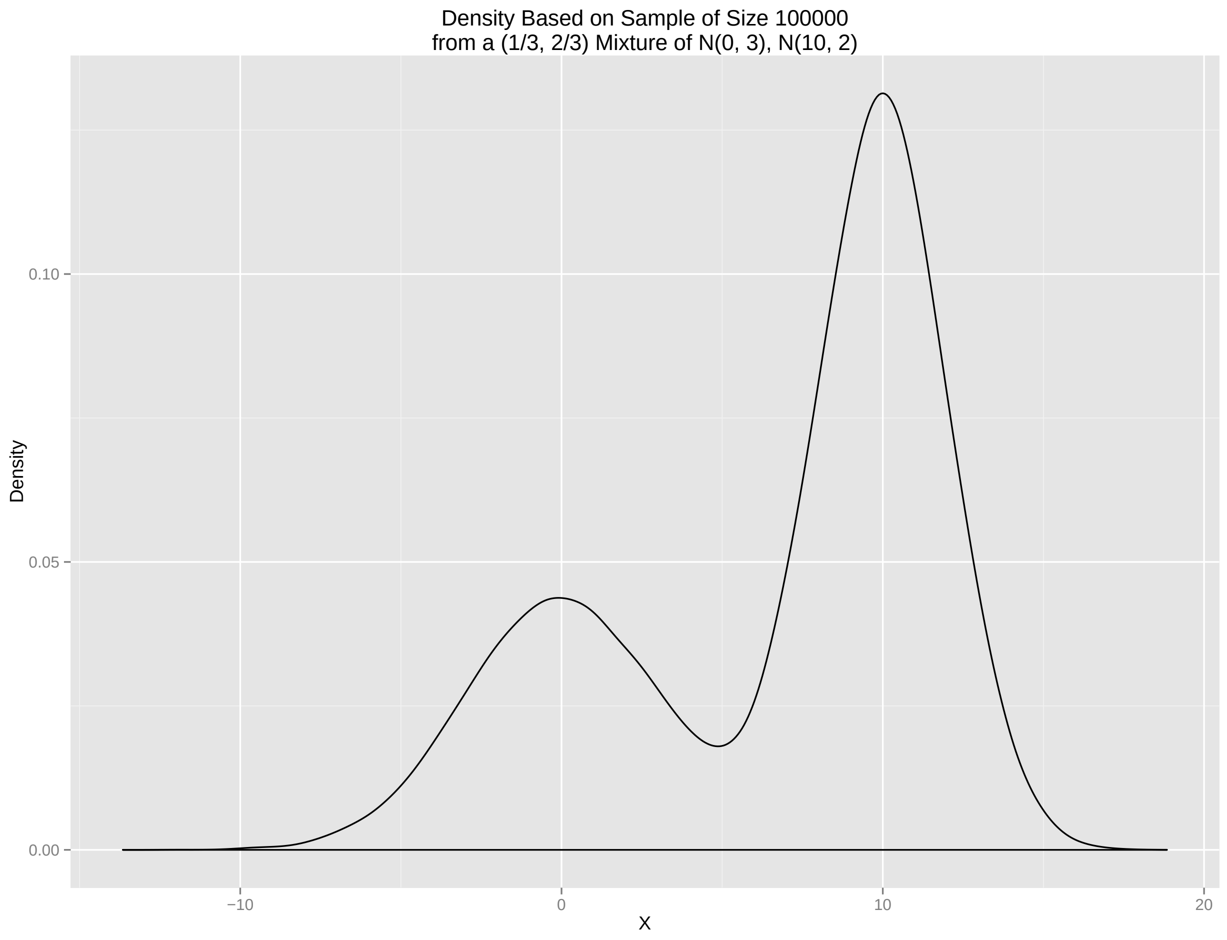
```
mix2norm.sample <- data.frame(x=rmix2norm(n=100000))
list(sd=sd(mix2norm.sample$x), five.number.summary =summary(mix2norm.sample$x))
```

```
$sd
[1] 5.282

$five.number.summary
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 -13.70    2.03    8.65    6.65   10.60   18.80 
```

Here's a density function of the large sample.

```
p1 <- ggplot(data=mix2norm.sample, aes(x=x)) +
  geom_density() +
  labs(x="X", y="Density",
       title=paste0("Density Based on Sample of Size 100000\n",
                    "from a (1/3, 2/3) Mixture of N(0, 3), N(10, 2)"))
```

```
print(p1)
```



4 The inverse CDF theorem applied to kernel density estimate-based empricial CDFs

Note: The the set.seed function is purposely not being set in this section. Running this code will therefore yield figures that differ slightly with course slides and figures in this document.

The eosinophil count data:

```
EOS.data <- data.frame(EOS=c(-4, -2, 0, 4, 5.5, 10.1, 10.2, 11, 11.2, 12, 20))
```

4.1 A function to obtain KDE-based eCDFs

This next chunk of code accomplishes the following:

- Obtains the kernel density estimate (KDE) (stats::density)
- Interpolates the density using the approxfun function (stats::approx)
- Approximates the cumulative distribution function (CDF) (stats::integrate, plyr::ddply)

```
pdf <- density(EOS.data$EOS)
f <- approxfun(pdf$x, pdf$y, yleft = 0, yright = 0)
integral <- function(x) as.numeric(integrate(f, -Inf, upper = x)[[1]])
df <- data.frame(x = seq(-15, 35, 0.1))
cdf <- ddply(df, .(x), mutate, y=integral(x))
```

In this version, a different approximation to the CDF is formed by manipulating the bandwidth used in the density estimation step.

```
pdf2 <- density(EOS.data$EOS, bw=bw.nrd(EOS.data$EOS)/3)
f2 <- approxfun(pdf2$x, pdf2$y, yleft = 0, yright = 0)
integral2 <- function(x) {
  as.numeric(integrate(f2, -Inf, upper = x,stop.on.error = FALSE)[[1]])
  }
df2 <- data.frame(x=seq(-15, 35, .1))
cdf2 <- ddply(df2, .(x), mutate, y = integral2(x))
```

Finally, the CDF obtained by MLE of the mean and standard deviation of the data is obtained.

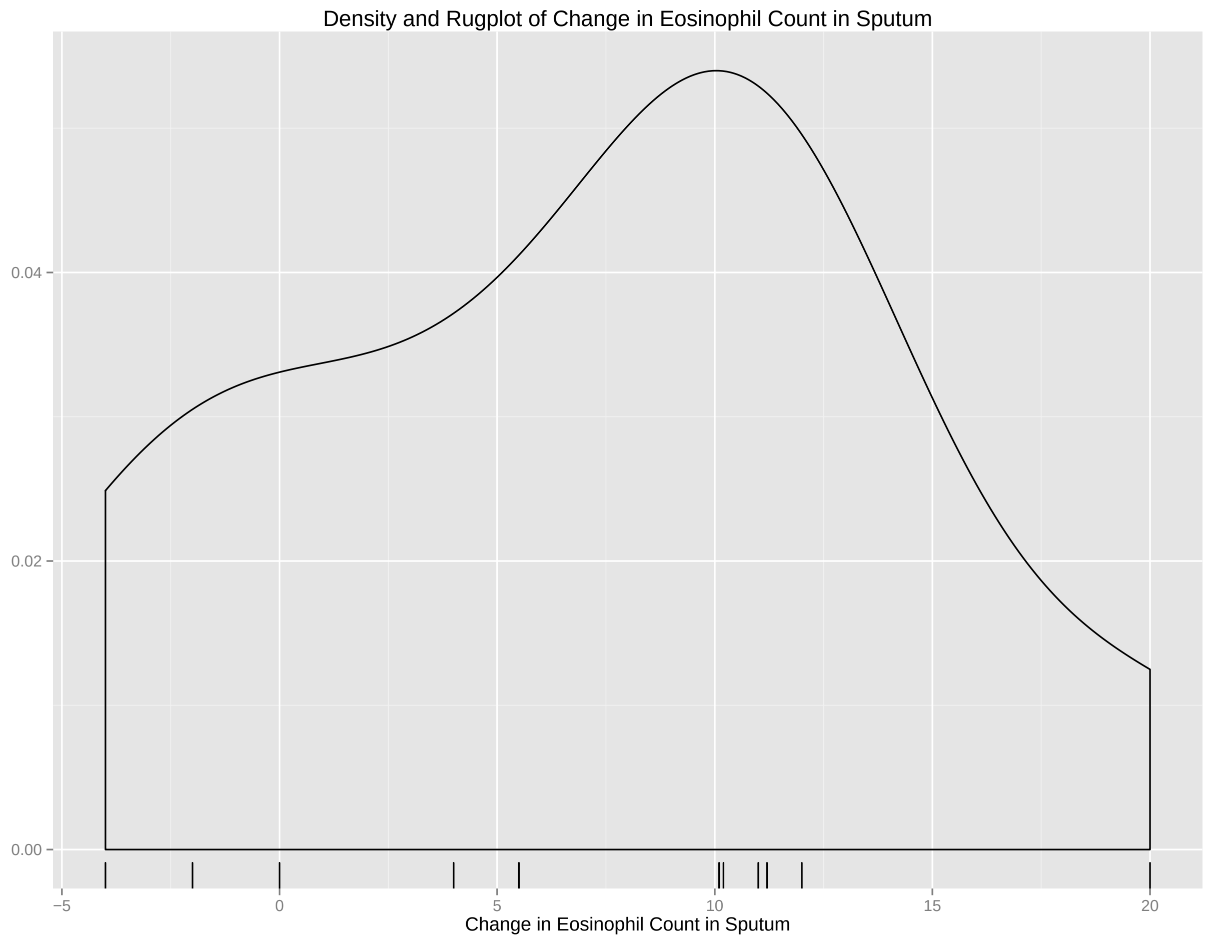
```
cdf.norm <- data.frame(
  x=cdf2$x,
  y=pnorm(cdf2$x,
          mean=fitdistr(EOS.data$EOS, "normal")$estimate[1],
          sd=fitdistr(EOS.data$EOS, "normal")$estimate[2]))
```

4.1.1 Slide 1

A default density function.

```
emp.den <- ggplot(data=EOS.data, aes(x=EOS)) +
  geom_density(fill="transparent") +
  geom_rug() +
  labs(x="Change in Eosinophil Count in Sputum", y="") +
  ggtitle("Density and Rugplot of Change in Eosinophil Count in Sputum")
```

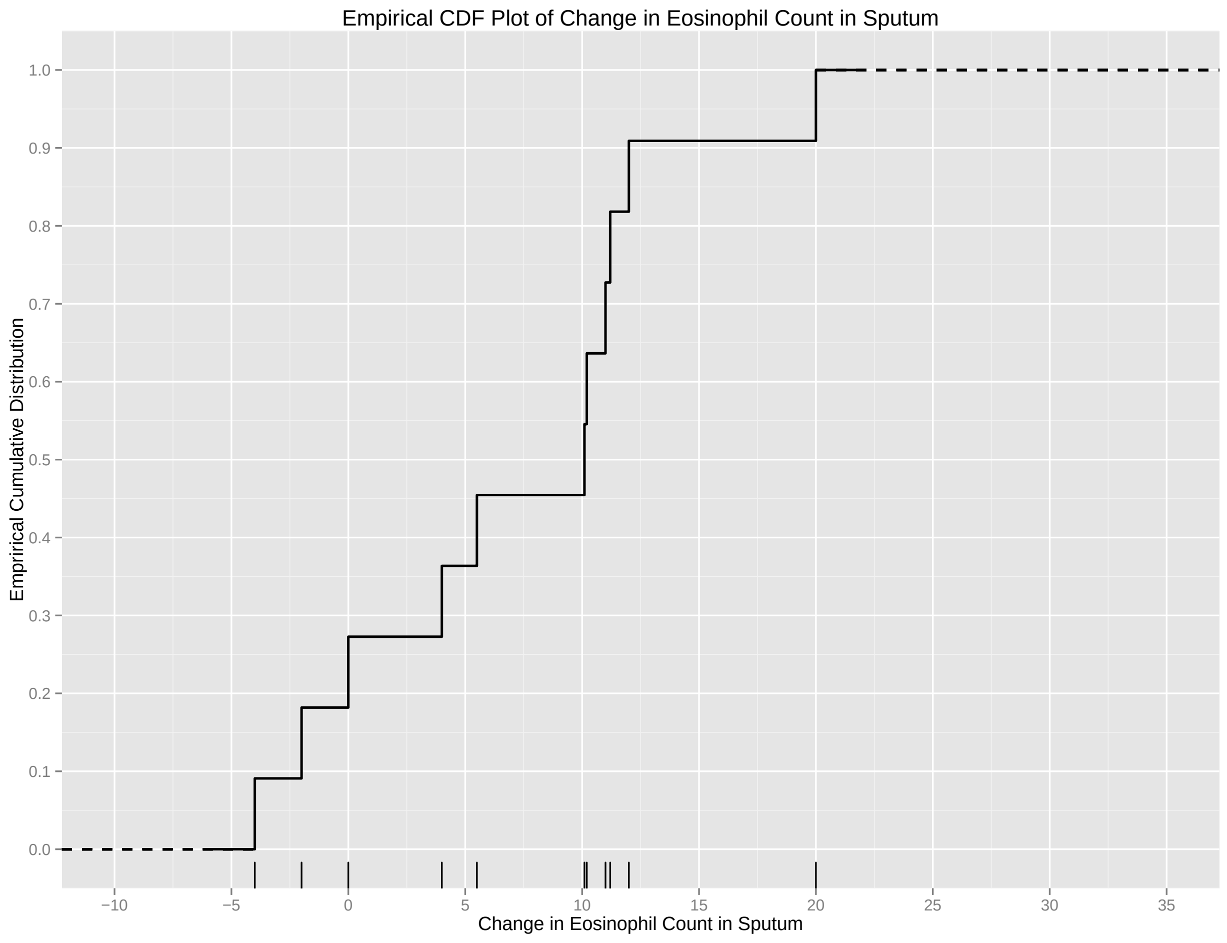
```
print(emp.den)
```



4.1.2 Slide 2

```
emp.cdf <- ggplot(data=EOS.data, aes(x=EOS)) +
  stat_ecdf(size=.75) +geom_rug() +
  labs(x="Change in Eosinophil Count in Sputum",
       y="Emprirical Cumulative Distribution",
       title="Empirical CDF Plot of Change in Eosinophil Count in Sputum") +
  scale_x_continuous(limits=c(-10,35), breaks=seq(-10,35,5)) +
  scale_y_continuous(limits=c(0,1), breaks= seq(0,1,.1)) +
  geom_segment(x=-Inf, xend=-4, y=0,yend=0, linetype="dashed", size=.75) +
  geom_segment(x=20, xend=Inf, y=1,yend=1, linetype="dashed", size=.75)
```

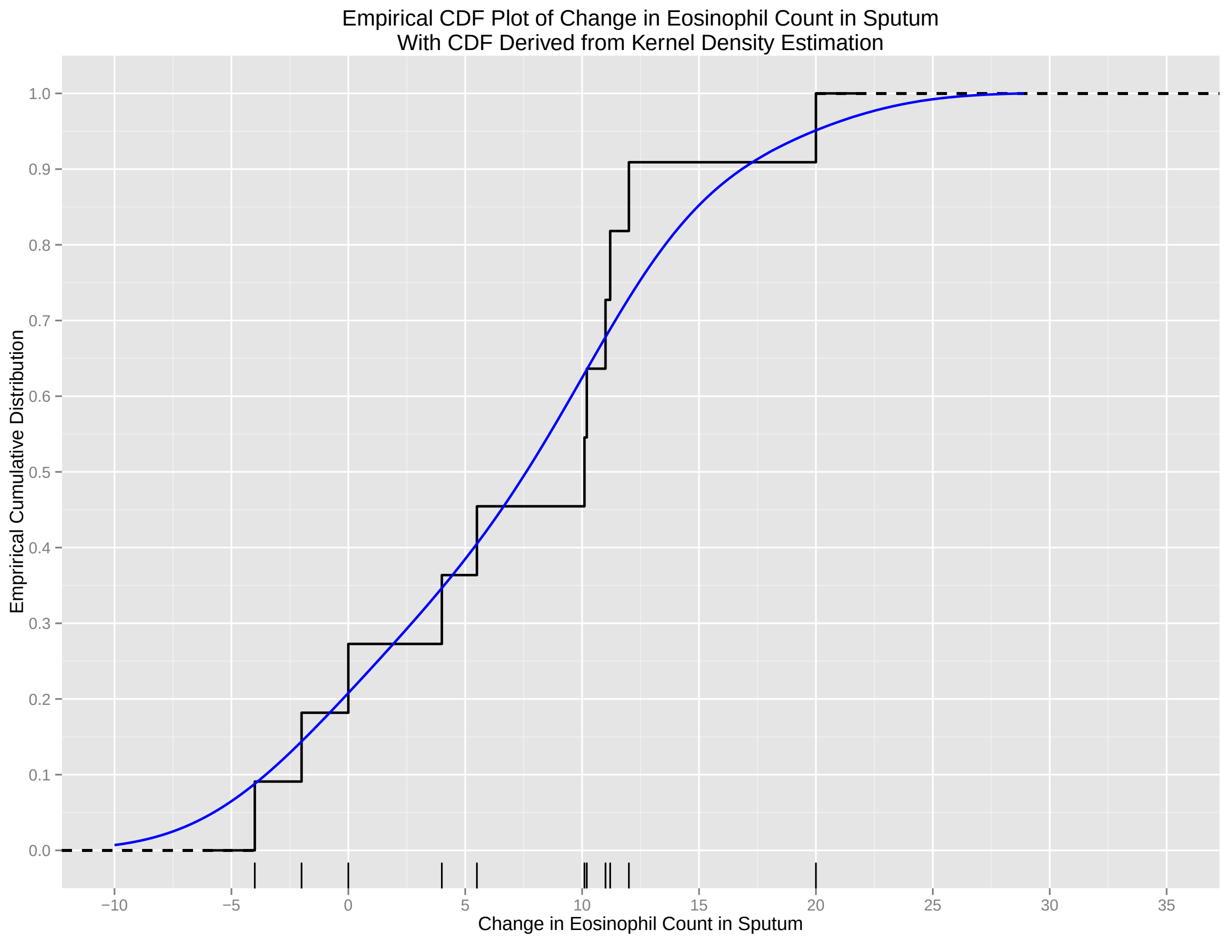
```
print(emp.cdf)
```



4.1.3 Slide 3

```r
emp.cdf.1 <- emp.cdf +
  geom_line(data=cdf, aes(x=x, y=y), color="blue", size=.75) +
  labs(title=paste0(
    "Empirical CDF Plot of Change in Eosinophil Count in Sputum\n",
    "With CDF Derived from Kernel Density Estimation"))
```

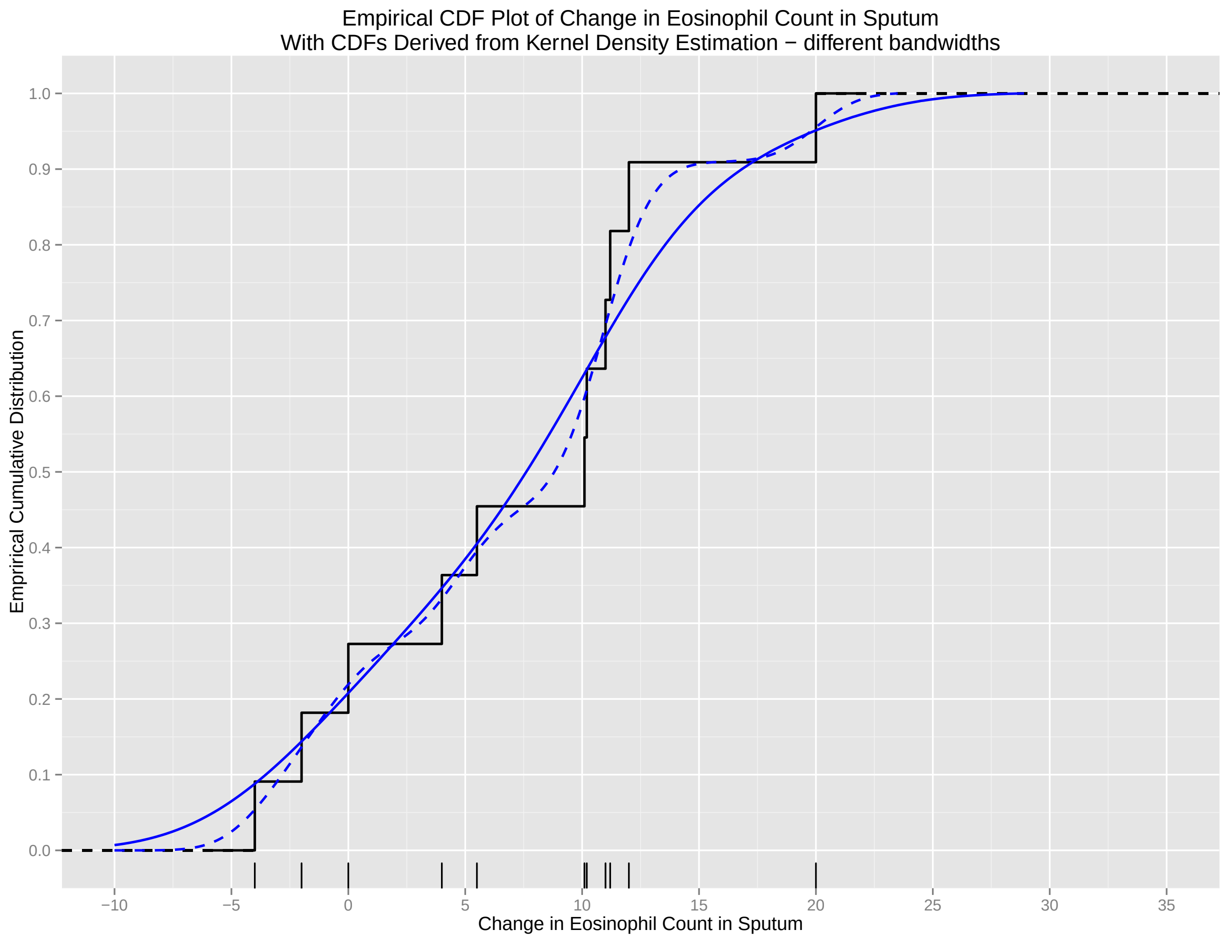
```r
print(emp.cdf.1)
```



4.1.4 Slide 4

```r
emp.cdf.2 <- emp.cdf.1 +
  geom_line(data=cdf2, aes(x=x, y=y),
            color="blue", size=.75, linetype="dashed") +
  labs(title=paste0(
    "Empirical CDF Plot of Change in Eosinophil Count in Sputum\n",
    "With CDFs Derived from Kernel Density Estimation - different bandwidths"))
```

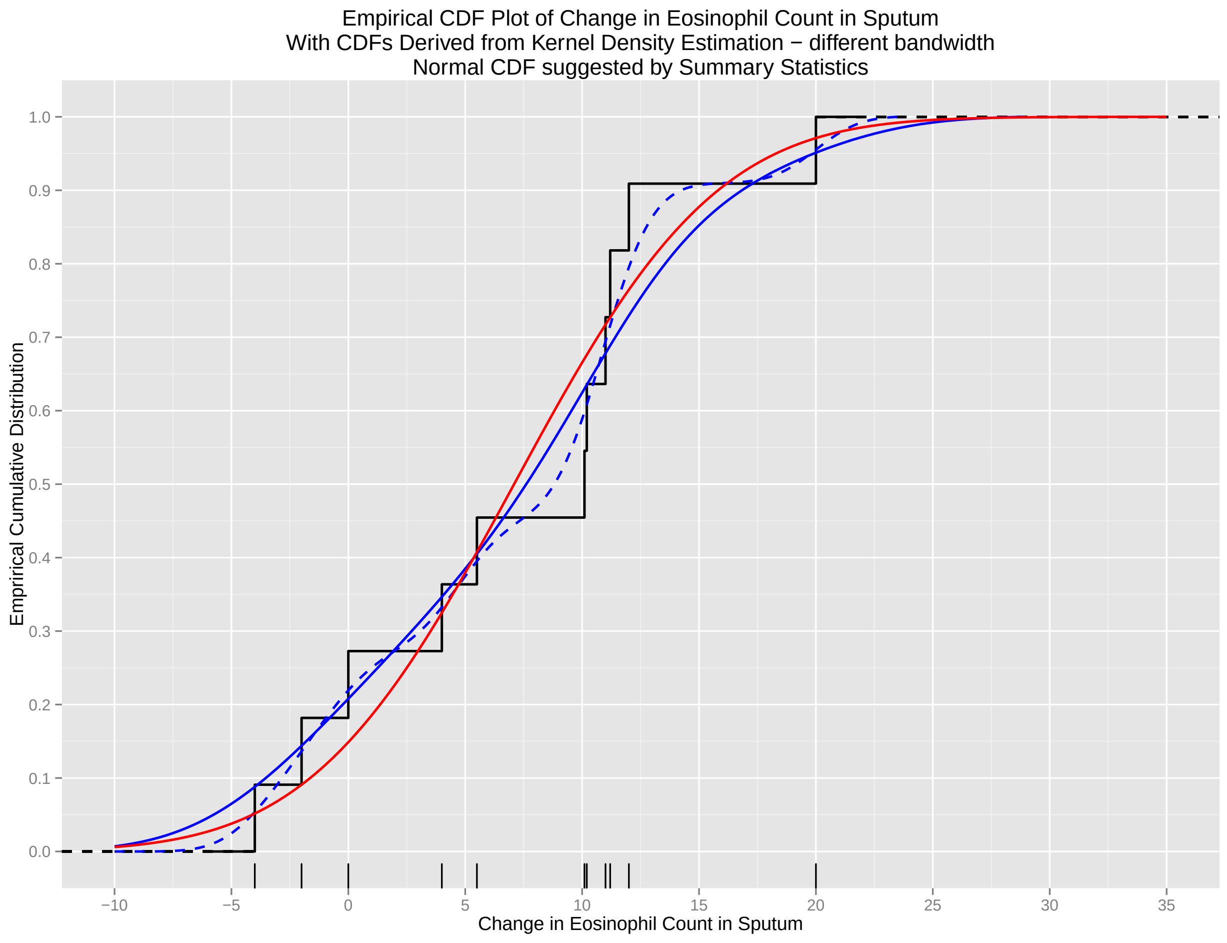
```r
print(emp.cdf.2)
```



4.1.5 Slide 5

```r
emp.cdf.3 <- emp.cdf.2 +
  geom_line(data=cdf.norm, aes(x=x, y=y), color="red", size=.75) +
  labs(title=paste0(
    "Empirical CDF Plot of Change in Eosinophil Count in Sputum\n",
    "With CDFs Derived from Kernel Density Estimation - different bandwidth\n",
    "Normal CDF suggested by Summary Statistics"))
```

```r
print(emp.cdf.3)
```



4.1.6 Slide 6

This function accomplishes the following steps to simulate a sample of size n:

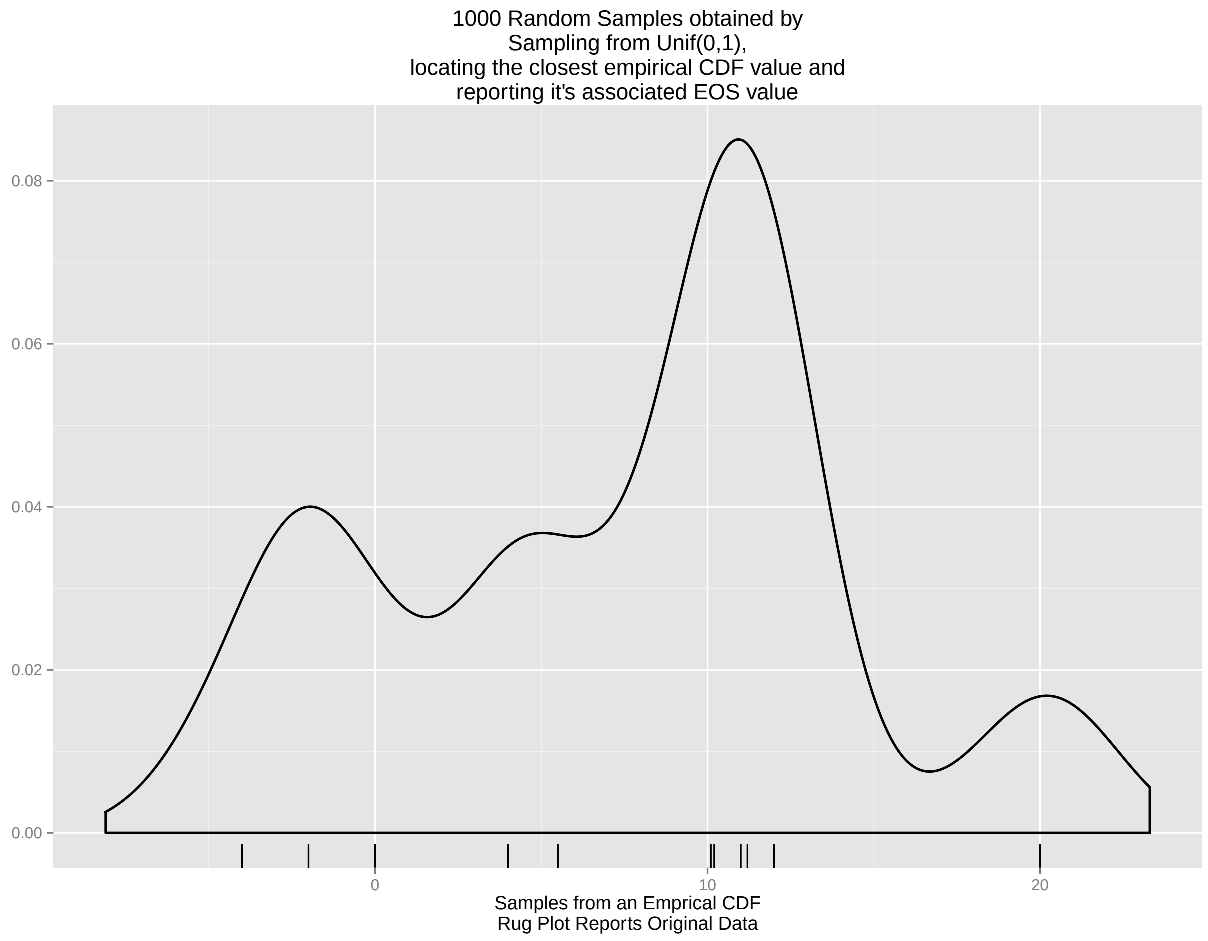
- Obtain a random uniform, U
- Identify the value from the empirical CDF that is closest to U
- Report back that corresponding x value

Consider the role of the grid chosen to estimate the empirical CDF.

```
get.random.sample <- function(n, df.cdf=cdf2){
  random.sample <- c()
  for(i in 1:n){
    random.sample <- c(random.sample,
                       df.cdf$x[which.min(abs(df.cdf$y - runif(1)))])
    }
  return(random.sample)
  }
```

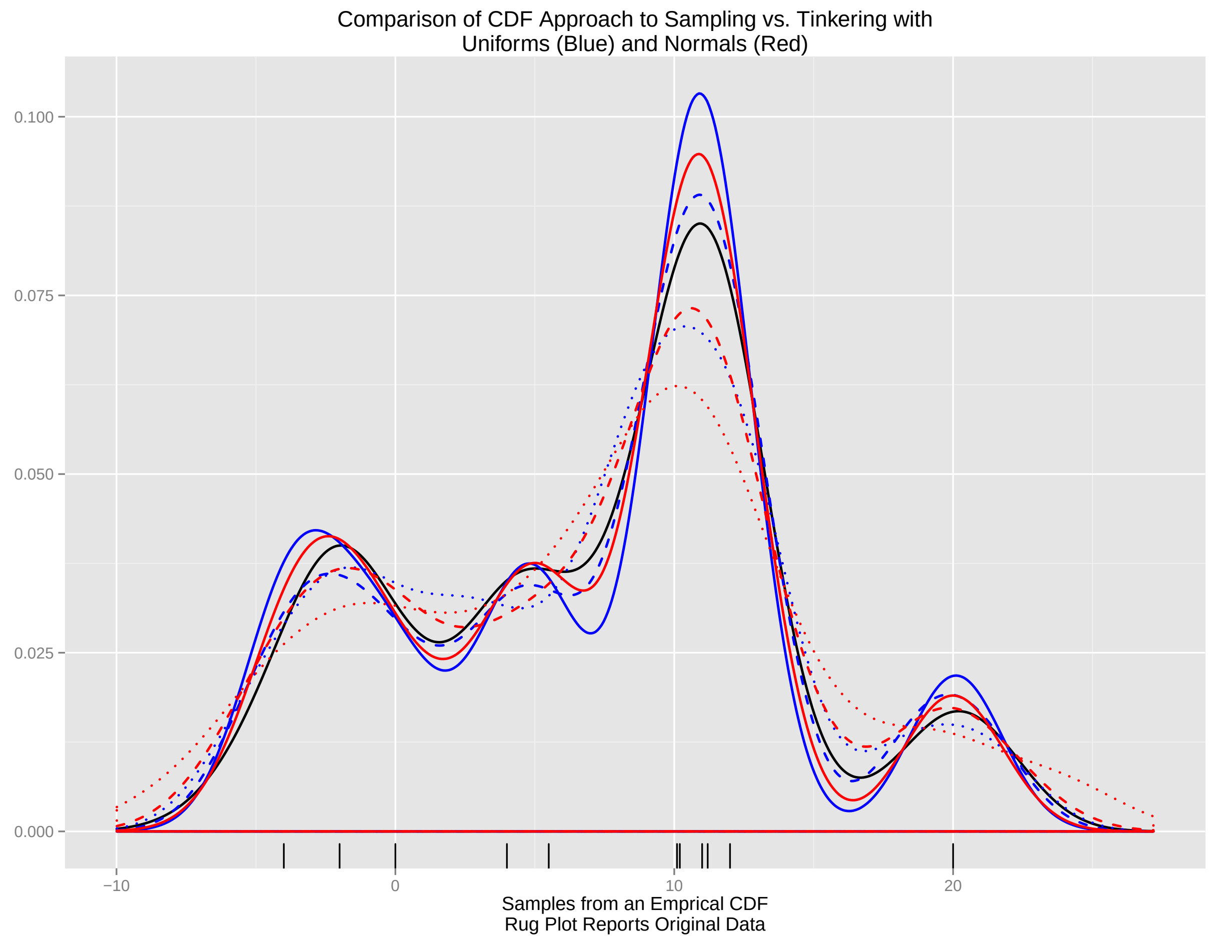
```r
p1 <- ggplot(data=data.frame(x= get.random.sample(1000)), aes(x=x)) +
  geom_density(size=.75) +
  geom_rug(data=EOS.data, aes(x=EOS)) +
  labs(x="Samples from an Emprical CDF\nRug Plot Reports Original Data",
       y="",
       title=paste0("1000 Random Samples obtained by\n",
                    "Sampling from Unif(0,1),\n",
                    "locating the closest empirical CDF value and\n",
                    "reporting it's associated EOS value"))
```

```r
print(p1)
```



4.2 Tinkering

```r
# Tinkering: Sampling for data points and uniform errors
sample.1 <- data.frame(x = sample(size=1000, x=EOS.data$EOS, replace=TRUE) +
                         runif(n=1000,min=-1, max=1))
sample.2 <- data.frame(x = sample(size=1000, x=EOS.data$EOS, replace=TRUE) +
                         runif(n=1000,min=-2, max=2))
sample.3 <- data.frame(x = sample(size=1000, x=EOS.data$EOS, replace=TRUE) +
                             runif(n=1000,min=-3, max=3))
# Tinkering: Sampling for data points and normal errors
sample.4 <- data.frame(x = sample(size=1000, x=EOS.data$EOS, replace=TRUE) +
                             rnorm(n=1000))
sample.5 <- data.frame(x = sample(size=1000, x=EOS.data$EOS, replace=TRUE) +
                             rnorm(n=1000, sd=2))
sample.6 <- data.frame(x = sample(size=1000, x=EOS.data$EOS, replace=TRUE) +
                             rnorm(n=1000, sd=3))

p1 +
  geom_density(data=sample.1, aes(x=x), color="blue", size=.75) +
  geom_density(data=sample.2, aes(x=x), color="blue", linetype="dashed",
               size=.75) +
  geom_density(data=sample.3, aes(x=x), color="blue", linetype="dotted",
               size=.75) +
  geom_density(data=sample.4, aes(x=x), color="red", size=.75) +
  geom_density(data=sample.5, aes(x=x), color="red", linetype="dashed",
               size=.75) +
  geom_density(data=sample.6, aes(x=x), color="red", linetype="dotted",
               size=.75) +
  ggtitle(paste0("Comparison of CDF Approach to Sampling vs. Tinkering with\n",
                 "Uniforms (Blue) and Normals (Red)"))
```



5 Enrollment modeling

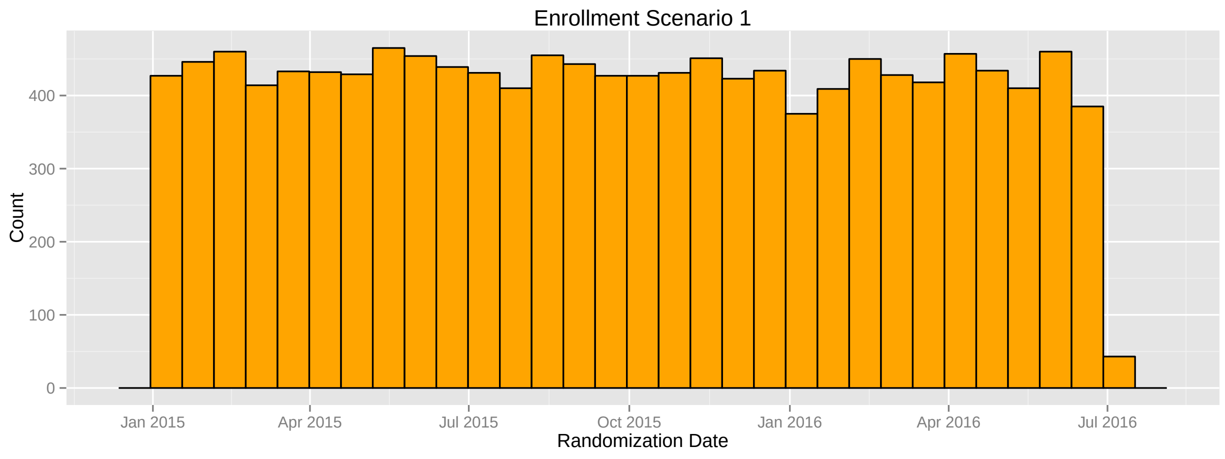
Assume knowledge of hard enrollment start and stop dates:

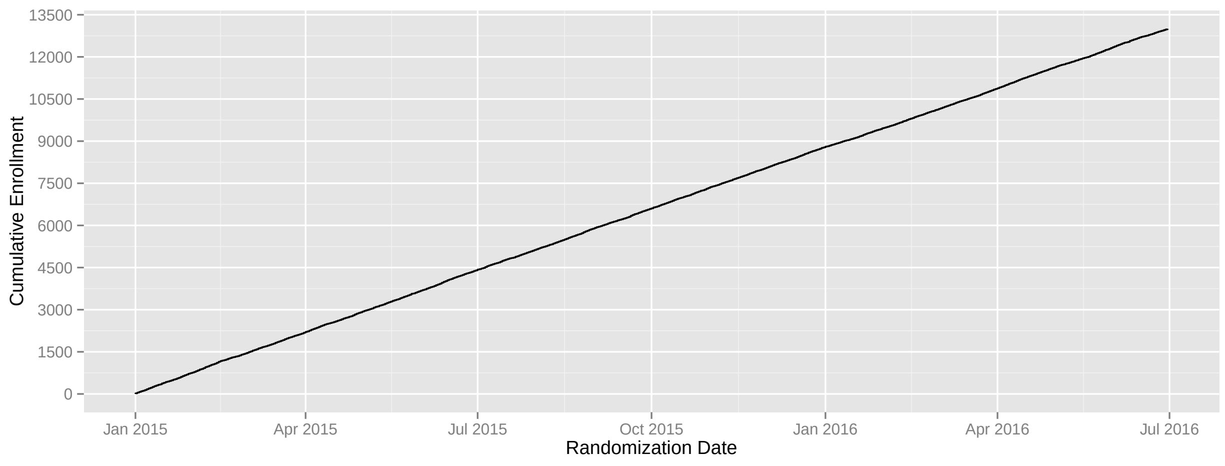
```r
enroll.start <- as.Date("2015-01-01")
enroll.stop <- as.Date("2015-01-01") + months(18) - 1
```

5.1 Uniform enrollment

The chron package is used in conjunction with base functions sort and sample.

```r
holdit <- data.frame(enrollment=as.Date(chron(sort(
  sample(x=enroll.start:enroll.stop, size=13000, replace=TRUE)))))

p1 <- ggplot(data=holdit, aes(x=enrollment)) +
  geom_histogram(color="black", fill="orange") +
  labs(x="Randomization Date", y="Count", title="Enrollment Scenario 1")

p2 <- ggplot(data=holdit, aes(x=enrollment, y= 1:nrow(holdit))) +
  geom_step() +
  labs(x="Randomization Date", y="Cumulative Enrollment") +
  scale_y_continuous(breaks=seq(0,15000, 1500))

grid.arrange(p1,p2, ncol=1)
```



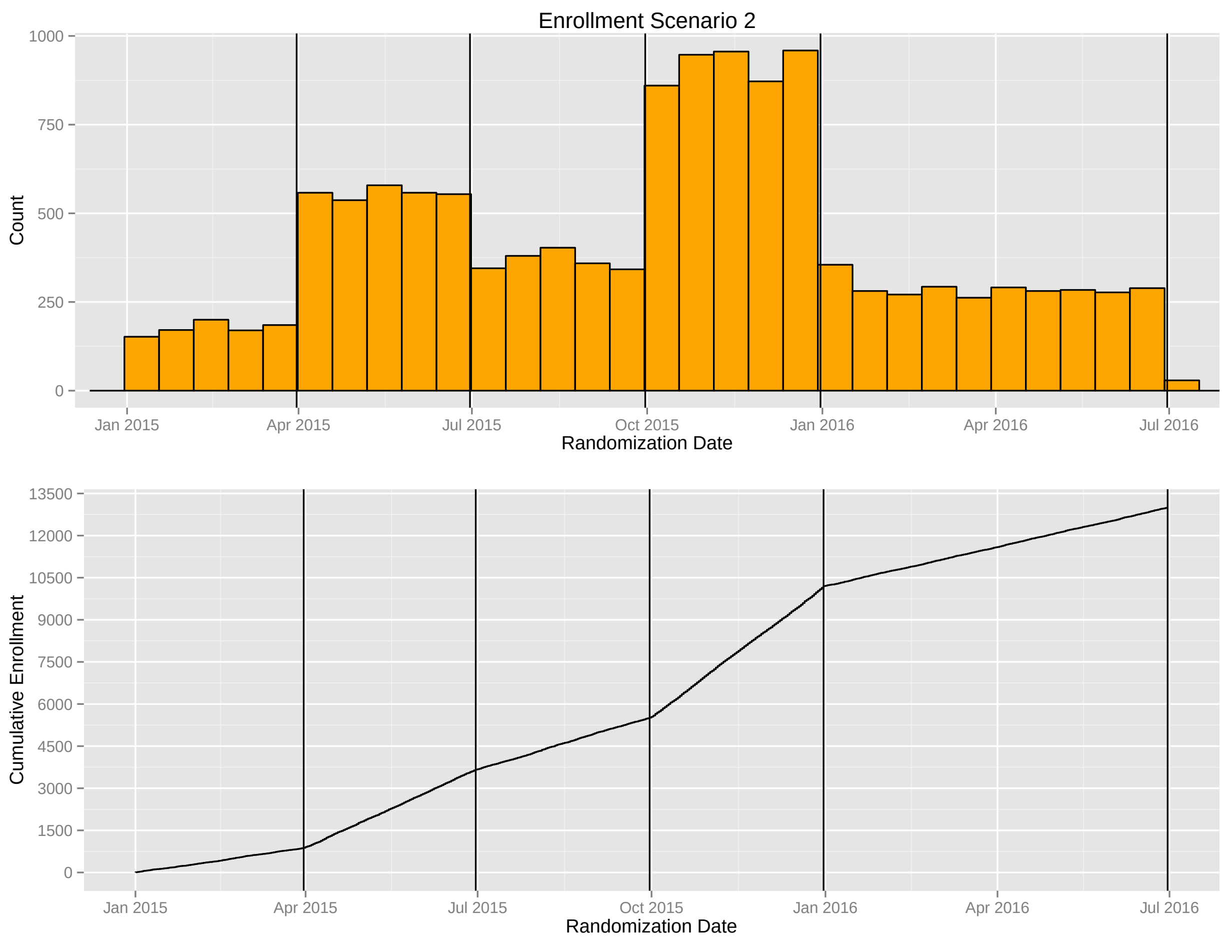


5.2 Stepwise uniform enrollment

```
enroll.start1 <- as.Date("2015-01-01")
enroll.stop1 <- as.Date("2015-01-01") + months(3) - 1
enroll.start2 <- enroll.stop1 + 1
enroll.stop2 <- as.Date("2015-01-01") + months(6) - 1
enroll.start3 <- enroll.stop2 + 1
enroll.stop3 <- as.Date("2015-01-01") + months(9) - 1
enroll.start4 <- enroll.stop3 + 1
enroll.stop4 <- as.Date("2015-01-01") + months(12) - 1
enroll.start5 <- enroll.stop4 + 1
enroll.stop <- as.Date("2015-01-01") + months(18) - 1

start.dates <- c(enroll.start1, enroll.start2, enroll.start3,
                 enroll.start4, enroll.start5)
stop.dates <- c(enroll.stop1, enroll.stop2, enroll.stop3,
                enroll.stop4, enroll.stop)

period.dist <- sample(paste("Period", 1:5),
                      size=13000, prob=c(1, 3, 2, 5, 3), replace=T)
holdit <- c()
```

```
for(i in 1:length(table(period.dist))){
  holdit <- c(holdit, sample(start.dates[i]:stop.dates[i],
                             size=table(period.dist)[i], replace=TRUE))
  }
holdit <- data.frame(enrollment=as.Date(chron(sort(holdit))))
```

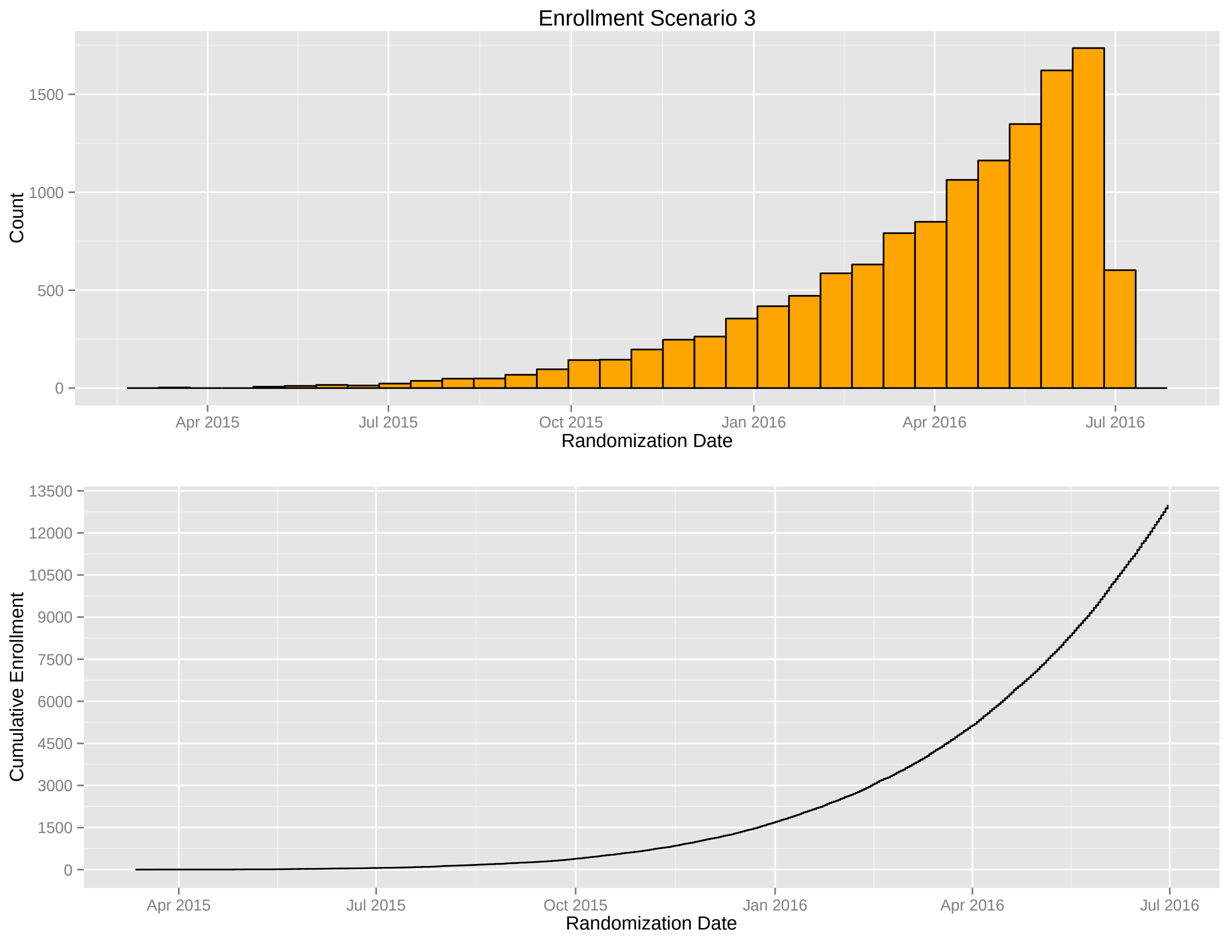
```
p1 <- ggplot(data=holdit, aes(x=enrollment)) +
  geom_histogram(color="black", fill="orange") +
  labs(x="Randomization Date", y="Count", title="Enrollment Scenario 2") +
  geom_vline(xintercept=as.numeric(stop.dates)) +
  scale_x_date(limits=c(min(holdit$enrollment), max(holdit$enrollment)))

p2 <- ggplot(data=holdit, aes(x=enrollment, y= 1:nrow(holdit))) +
  geom_step() +
  labs(x="Randomization Date", y="Cumulative Enrollment") +
  scale_y_continuous(breaks=seq(0,15000, 1500)) +
  geom_vline(xintercept=as.numeric(stop.dates)) +
  scale_x_date(limits=c(min(holdit$enrollment), max(holdit$enrollment)))

grid.arrange(p1,p2, ncol=1)
```



5.3 Scaled beta enrollment

The base::sample function is used sample herem but the probability argument appeals to the density of the beta distribution.

```r
holdit <- data.frame(enrollment = as.Date(chron(sort(
  sample(x=enroll.start:enroll.stop, size=13000, replace=TRUE, prob =
           dbeta(seq(0,1,length.out = length(enroll.start:enroll.stop)),5,1)))))
  )

p1 <- ggplot(data=holdit, aes(x=enrollment)) +
  geom_histogram(color="black", fill="orange") +
  labs(x="Randomization Date", y="Count", title="Enrollment Scenario 3")

p2 <- ggplot(data=holdit, aes(x=enrollment, y= 1:nrow(holdit))) +
  geom_step() +
  labs(x="Randomization Date", y="Cumulative Enrollment") +
  scale_y_continuous(breaks=seq(0,15000, 1500))

grid.arrange(p1,p2, ncol=1)
```



5.4 Centre-based enrollment

Without loss of generality, assume

- 9 Countries are to participate
- Each country plans to have 10 centres
- First Centre Initiated (FCI) for countries ~ Discrete Uniform over 1st 180 days of the year
- Remaining centres come online uniformly over 90-day period following country's FCI
- Overall rates for countries sample from Uniform(.2, .8)
- Given the overall country rate, sample from a Beta with mean equal to that rate
- Given centre-level rates, model enrollment at the centre level with a Poisson process
- Ignore the centre-level shutdown issue

```r
set.seed(2468)
# The enroll.stop time well exceeds my inital rought estimate
enroll.start <- 1
enroll.stop <- 365*2

# Initializing a data.frame to hold information about centres
centres <- data.frame(COUNTRY=sort(rep(paste("Country",1:9))))

# Sampling to manufacture First Centre Initiated dates for the 9 countries
centres$FCI <- sample(enroll.start:(enroll.start+180), size=9,
                      replace=TRUE)
# Initializing columns intended to hold the center initiating dates for the remaining
# 9 centres of each country
centres$centre2 <- centres$centre3 <- centres$centre4 <- centres$centre5 <-
  centres$centre6 <- centres$centre7 <-centres$centre8 <- centres$centre9 <-
  centres$centre10 <- 0

# Populating the centre start-up by adding random number of days to the country FCI date.
for(i in 3:11) {
  centres[,i] <- centres$FCI + sample(x = 1:90, size=nrow(centres),
                                      replace=TRUE)
}
```

5.4.1 Aside: Sifting through beta distributions

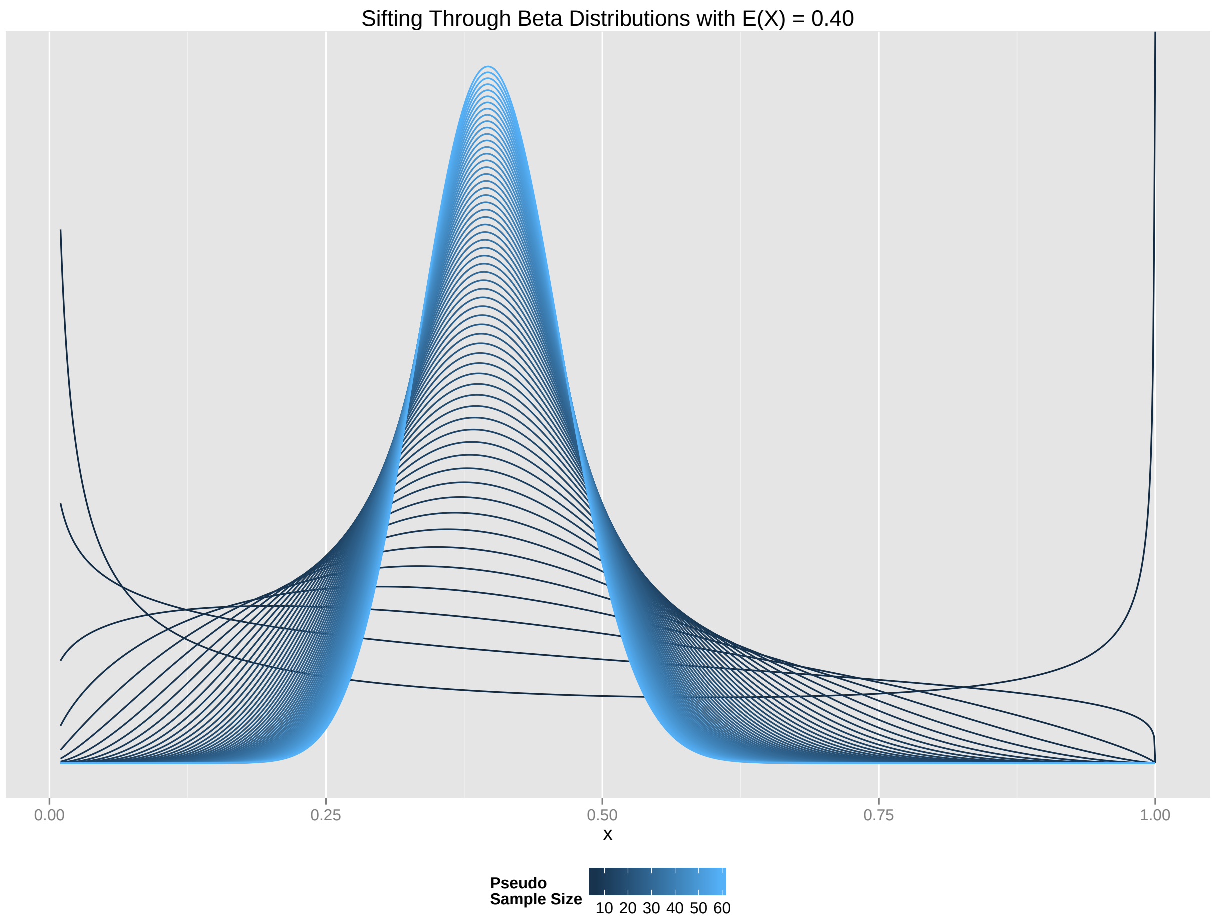
Consider the following code that re-parameterizes the beta distribution allowing us isolate betas sharing expected value 0.40.

```r
big.df <- c()
for(i in 1:60){
  big.df <- rbind(big.df,
                  data.frame(i=rep(i, length(seq(0.01,1,.001))),
                             x=seq(0.01,1,.001),
                             y=dbeta(seq(0.01,1,.001),.4*i, i - .4*i)))
  }

head(big.df)
```

```
  i     x     y
```

```
1 1 0.010 4.817
2 1 0.011 4.551
3 1 0.012 4.322
4 1 0.013 4.121
5 1 0.014 3.943
6 1 0.015 3.785
```

```
ggplot(data=big.df, aes(x=x,y=y, color=i, group=i)) +
  geom_line() +
  theme(legend.position="bottom") +
  labs(x="x", y=NULL,
       title="Sifting Through Beta Distributions with E(X) = 0.40",
       color="Pseudo\nSample Size") +
  scale_y_continuous(breaks=NULL)
```



Back to the main topic: At this point the data.frame, centres, holds a 9 rows, one for each country, with randomized start dates for each country's 10 centres.

```
head(centres)
```

```
    COUNTRY FCI centre10 centre9 centre8 centre7 centre6 centre5 centre4
1 Country 1  84      118     105     157     147     116     100     108
```

```
2 Country 2 104      109      194      120      145      180      114      180
3 Country 3  68       70       95      145       82       92      146      140
4 Country 4 164      176      224      175      244      194      176      172
5 Country 5  41      113       54       93       87       62       71       82
6 Country 6  62       66       84       66       69      146      123      139
  centre3 centre2
1     140     149
2     108     144
3     112      81
4     175     212
5     126     106
6     110      82
```

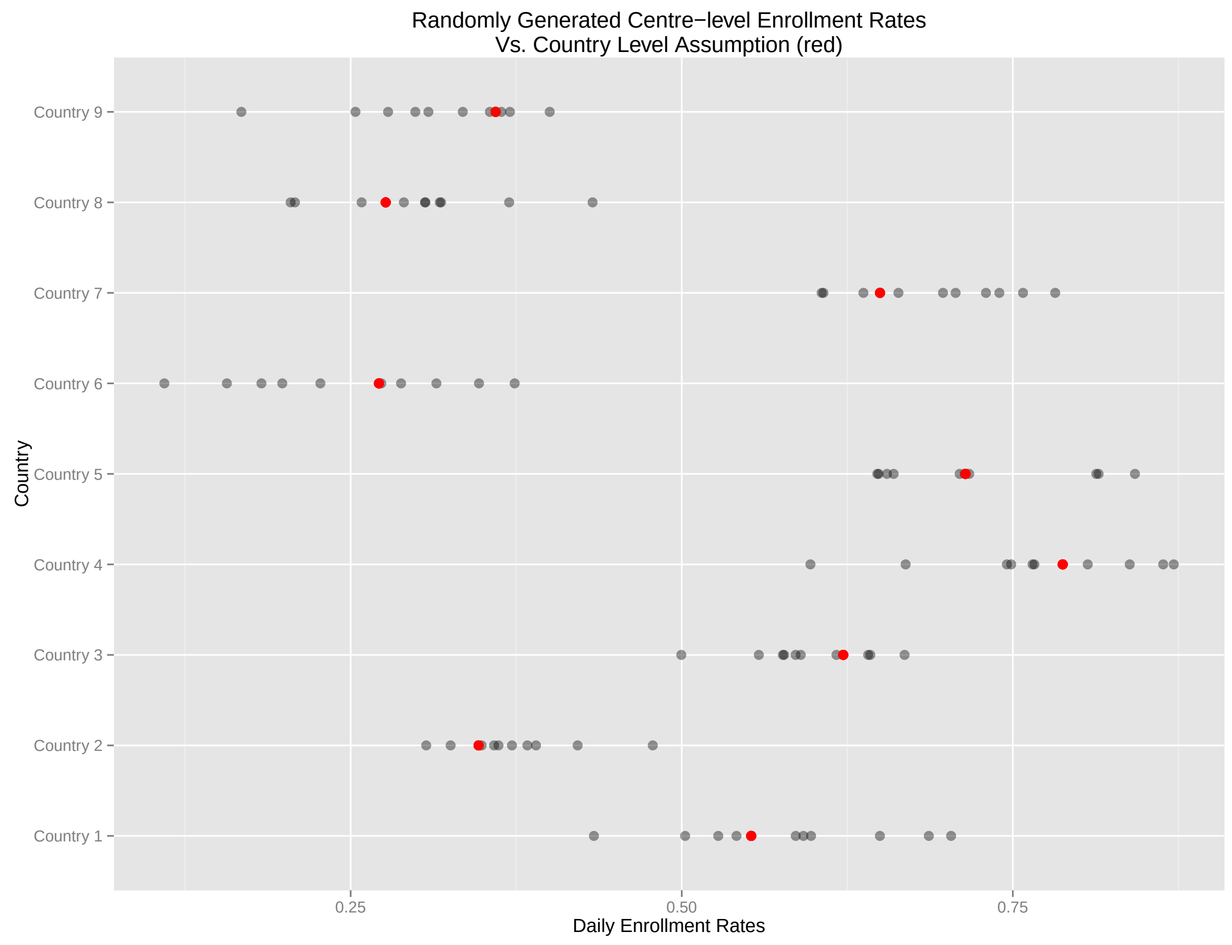
In this next chunk of code:

- the data.frame is reorganized using reshape2:melt
- country level rates are sampled from Uniform(.2, .8) using plyr::ddply
- centre level rates are sample from a beta distributions with mean equal to the sampled country level rate and variance controlled through the value 20. (See aside)

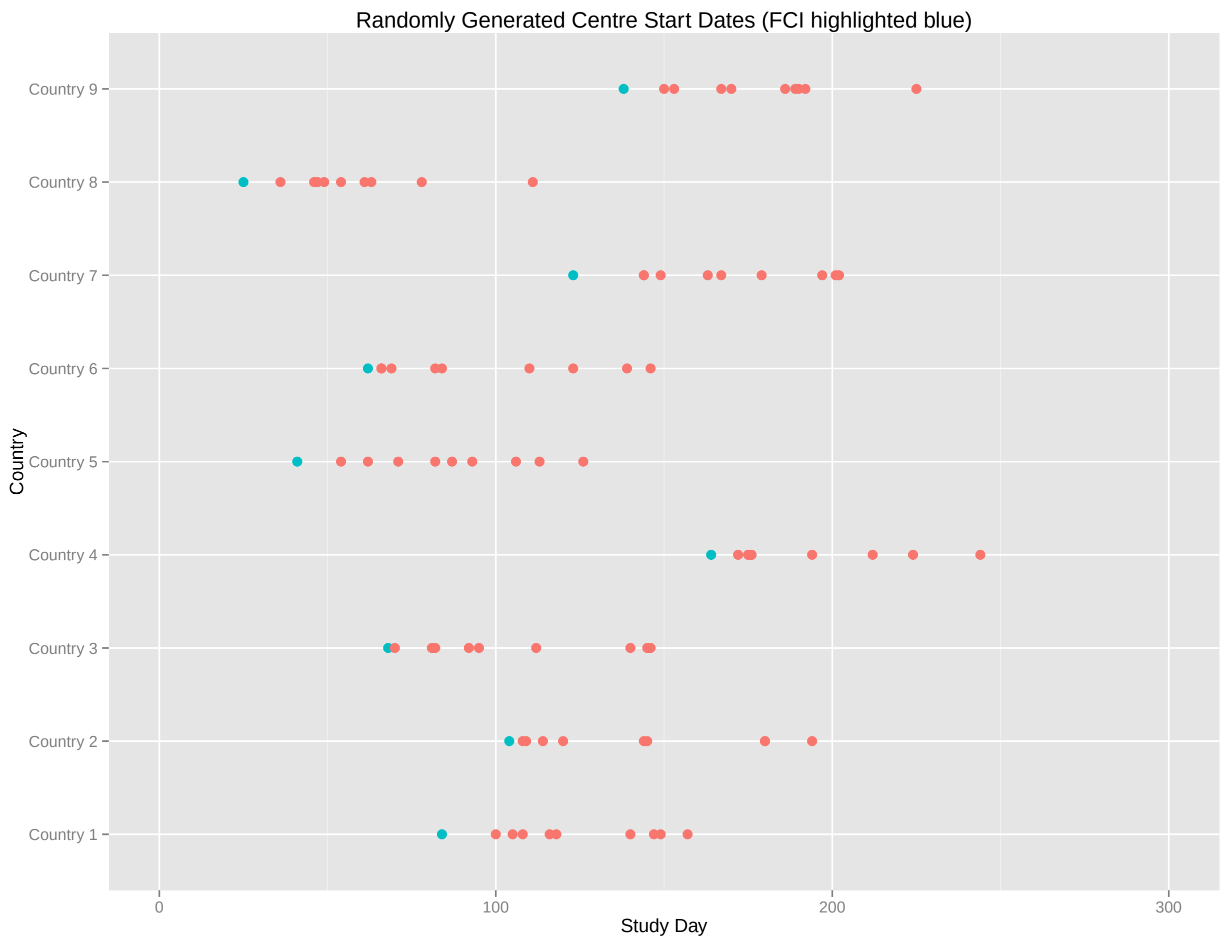
```r
centres.melt <- melt(centres, id=1)
centres.melt<- ddply(centres.melt, .(COUNTRY), mutate,
                     country.rate = runif(1, .2, .8))
centres.melt <- ddply(centres.melt, .(COUNTRY), mutate, centre.rate =
                        rbeta(length(value), country.rate*20,
                              20 - country.rate*20))
head(centres.melt)
```

```
    COUNTRY variable value country.rate centre.rate
1 Country 1      FCI    84       0.5524      0.6866
2 Country 1 centre10   118       0.5524      0.5976
3 Country 1  centre9   105       0.5524      0.5413
4 Country 1  centre8   157       0.5524      0.6496
5 Country 1  centre7   147       0.5524      0.5919
6 Country 1  centre6   116       0.5524      0.5276
```

```r
ggplot(data=centres.melt, aes(x=centre.rate, y=COUNTRY)) +
  geom_point(alpha=.4, size=3) +
  labs(x="Daily Enrollment Rates", y="Country",
       title=paste0("Randomly Generated Centre-level Enrollment Rates\nVs.",
                    " Country Level Assumption (red)")) +
  geom_point(aes(x=country.rate, y=COUNTRY), color="red", size=3)
```



```
ggplot(data=centres.melt, aes(x=value, y=COUNTRY,
                              color=factor(variable=="FCI"))) +
  geom_point(size=3) +
  labs(color="FCI", x="Study Day", y="Country",
       title="Randomly Generated Centre Start Dates (FCI highlighted blue)") +
  theme(legend.position="bottom") +
  xlim(0,300) +
  guides(color=FALSE)
```



The following function simulates enrollment at the centre-level:

```
enroll.centre <- function(enroll.duration=365*2,
                          centre.start=sample(size=1, x=1:(364/2)),
                          rate=rbeta(1,.1*50,1*50)){
  x <- rep(0, enroll.duration)
  x[centre.start:enroll.duration] <- rpois(enroll.duration-centre.start+1,
                                           lambda=rate)
  return(x)
  }
```

Testing the function using the first row of centres.melt:

```
enroll.centre(centre.start=centres.melt$value[1],
                   rate=centres.melt$centre.rate[1])
```

```
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 1 2 2 1 0 0 1
[106] 0 0 0 2 2 0 1 1 1 1 0 0 1 1 2 0 0 0 1 1 1 1 0 1 1 0 0 0 0 1 0 0 1 0 1
[141] 0 1 0 0 0 2 1 1 0 0 0 2 1 0 0 1 0 0 0 0 2 0 0 0 0 0 3 0 0 0 0 0 0 0 0
[176] 0 0 1 1 1 0 0 2 0 0 0 0 1 1 2 2 0 2 1 0 1 1 2 1 3 1 0 2 2 0 0 0 0 1 0
```

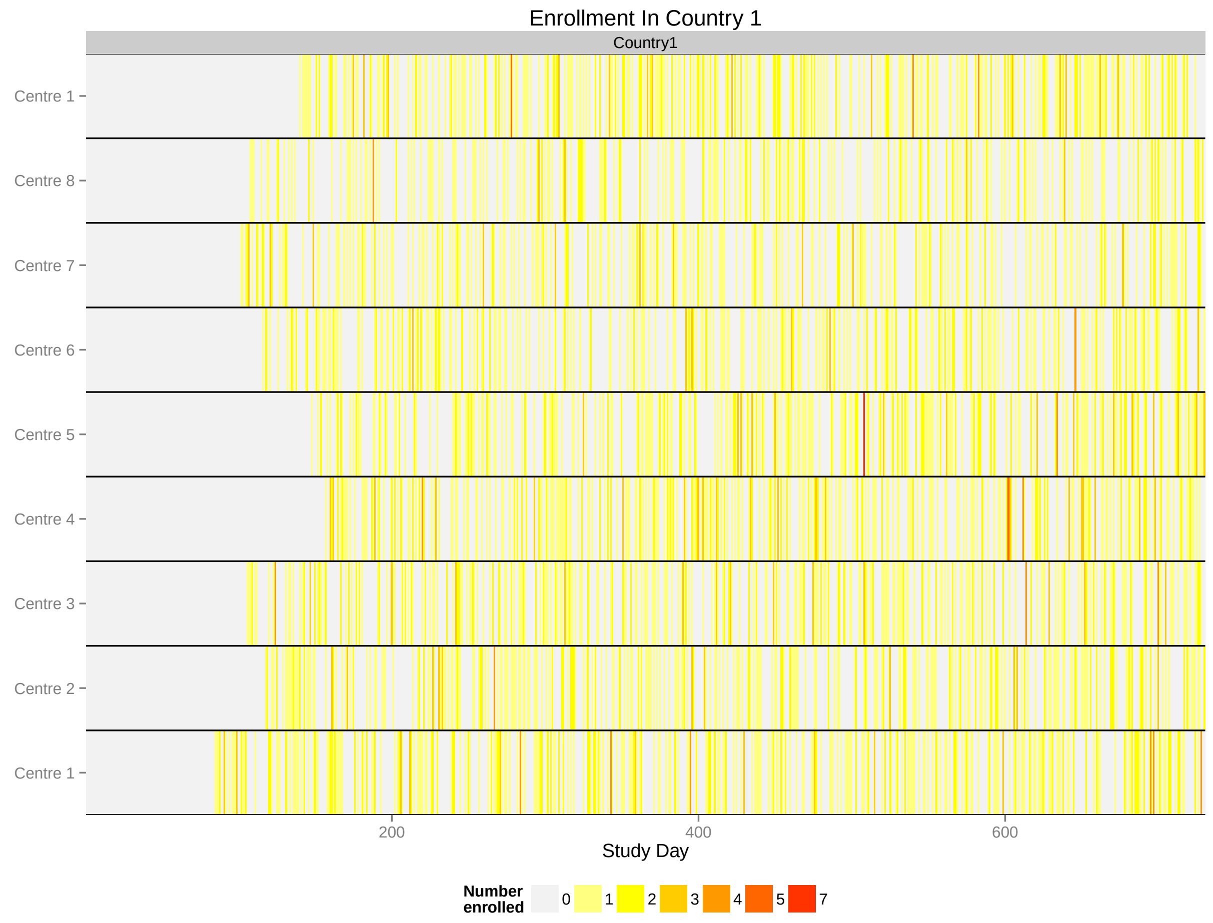
```
[211] 0 0 0 2 0 0 1 0 1 0 0 1 0 0 2 1 1 0 0 2 0 0 0 1 0 0 1 0 0 1 1 0 1 2 0
[246] 0 2 0 0 1 1 0 0 0 1 1 1 0 1 0 2 0 1 0 1 0 1 0 1 0 0 0 1 1 0 1 0 0 0 0
[281] 3 1 0 1 0 0 0 1 1 0 2 0 0 0 0 0 0 0 1 0 1 0 0 3 0 1 1 0 3 1 0 2 0 1 1
[316] 0 0 0 0 1 0 1 1 0 0 0 1 1 0 2 0 0 0 4 1 2 0 1 0 0 2 0 1 0 0 2 0 0 0 1
[351] 1 1 0 0 0 1 2 1 0 1 0 1 2 0 1 0 0 0 0 0 2 1 1 1 1 2 2 0 0 0 0 1 0 0 1
[386] 2 0 0 0 0 0 0 0 0 0 0 2 0 1 2 0 1 1 0 0 2 1 0 1 2 0 1 0 1 2 0 0 0 0 1
[421] 2 1 0 0 0 0 0 2 1 1 0 1 1 2 2 0 0 2 0 3 0 1 1 2 0 0 2 0 0 1 0 0 1 1 1
[456] 2 1 1 2 0 0 0 1 2 3 0 2 0 2 1 1 1 0 1 0 0 0 1 2 0 2 2 1 0 0 2 0 0 1 0
[491] 1 0 0 0 1 0 1 2 0 0 0 0 1 1 1 0 3 1 1 0 0 1 0 0 1 3 1 1 1 1 2 1 0 1 0
[526] 0 2 2 0 0 0 0 2 2 1 0 0 1 1 1 0 0 0 2 0 1 1 1 1 2 0 0 1 0 2 0 1 1 0 0
[561] 1 2 0 0 0 1 1 0 0 1 0 1 1 0 1 1 1 2 0 0 0 1 2 0 1 0 0 0 2 1 0 0 0 0 1
[596] 1 1 0 1 1 1 1 0 0 2 0 1 1 0 0 0 0 1 0 1 2 4 0 1 1 0 0 1 0 0 1 1 1 1 0
[631] 0 1 0 0 0 1 1 0 1 0 1 2 0 0 2 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 2 2 1 0
[666] 2 0 0 0 0 0 0 0 1 1 0 1 0 2 1 1 0 0 0 0 0 1 2 1 1 1 0 0 0 2 0 1 1 1 0
[701] 0 2 1 0 0 1 0 1 0 0 2 1 1 1 0 0 0 1 0 0 0 2 4 0 0 0 1 0 0 2
```

The enroll.centre function is applied to each row of centre.melt. The data.frame holdit and holdit.melt are used to collect results.

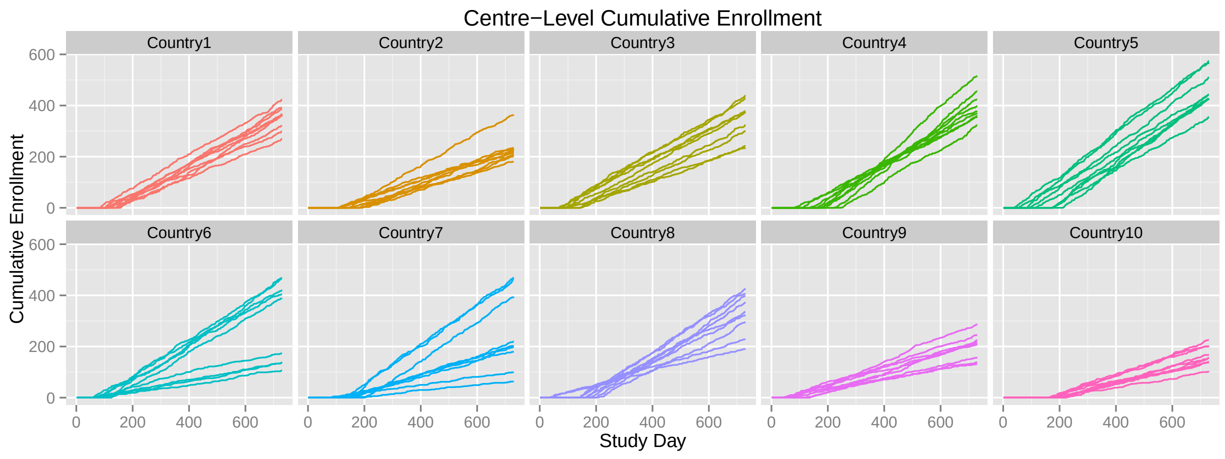
```r
holdit <- c()
for(i in 1:nrow(centres.melt)){
  holdit <- rbind(holdit,
                  enroll.centre(centre.start=centres.melt$value[i],
                                rate=centres.melt$centre.rate[i]))
  }

holdit <- t(holdit)
holdit <- data.frame(holdit)
holdit$day <- 1:nrow(holdit)
holdit.melt <- melt(holdit, id="day")
holdit.melt$COUNTRY <-factor(
  paste0("Country",
         as.numeric(cut(as.numeric(holdit.melt$variable),breaks = 10)))
  )
```

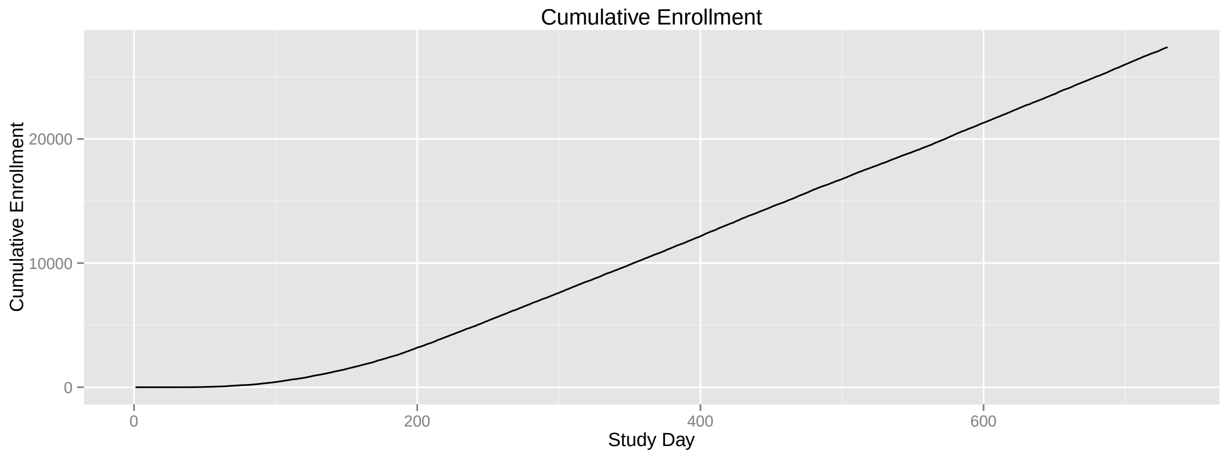
The next chunk of code provides a way to visualize Poisson processes occuring at each of the centres in Country1.

```r
ggplot(data=subset(holdit.melt, COUNTRY=="Country1"),
       aes(x=day, y=(variable), fill=as.factor(value))) +
  geom_tile() +
  facet_wrap(~COUNTRY, scales="free_y", ncol=3) +
  theme(legend.position="bottom") +
  scale_fill_manual(breaks=0:8, values=c("grey95", rev(heat.colors(7)))) +
  scale_x_continuous(expand=c(0,0)) +
  scale_y_discrete(expand=c(0,0), labels=paste("Centre", c(1,2,3,4,5,6,7,8))) +
  labs(x="Study Day", y="", title=("Enrollment In Country 1"),
       fill="Number\nenrolled") +
  geom_hline(yintercept=1:10-.5)
```



```
holdit.melt2 <- ddply(holdit.melt, .(COUNTRY, variable), mutate,
                      csum=cumsum(value))
holdit.melt2$COUNTRY <- factor(
  holdit.melt2$COUNTRY,
  c("Country1", "Country2",  "Country3",  "Country4" , "Country5",
    "Country6" , "Country7",  "Country8",  "Country9", "Country10"))
p1 <-ggplot(data=subset(holdit.melt2),
            aes(x=day, y=csum, group=variable, color=COUNTRY)) +
  geom_line() +
  facet_wrap(~COUNTRY, ncol=5) +
  guides(color=FALSE) +
  labs(x="Study Day", y="Cumulative Enrollment",
       title="Centre-Level Cumulative Enrollment")

p2 <- ggplot(data=ddply(holdit.melt[order(holdit.melt$day),], .(day),
                        summarize, sum=sum(value)), aes(x=day, y=cumsum(sum))) +
  geom_line() +
  labs(x="Study Day", y="Cumulative Enrollment", title="Cumulative Enrollment")

grid.arrange(p1,p2)
```





6 Simulating correlated data

6.1 Simulating positively correlated data

```
set.seed(8675309)
my.df <- data.frame(observation=1:10000)
my.df$start.point <- runif(10000, 20, 30)
my.df$variation1 <- my.df$start.point + runif(10000, min=-5, max=5)
my.df$variation2 <- my.df$start.point + runif(10000, min=-5, max=5)
```

A check is conducted to ensure simulated values are in line with expectations.

```
my.df.melt <- melt(my.df, id=1:2)
ddply(my.df.melt, .(variable), summarize, n=length(value),
      mean=mean(value), sd=sd(value))
```

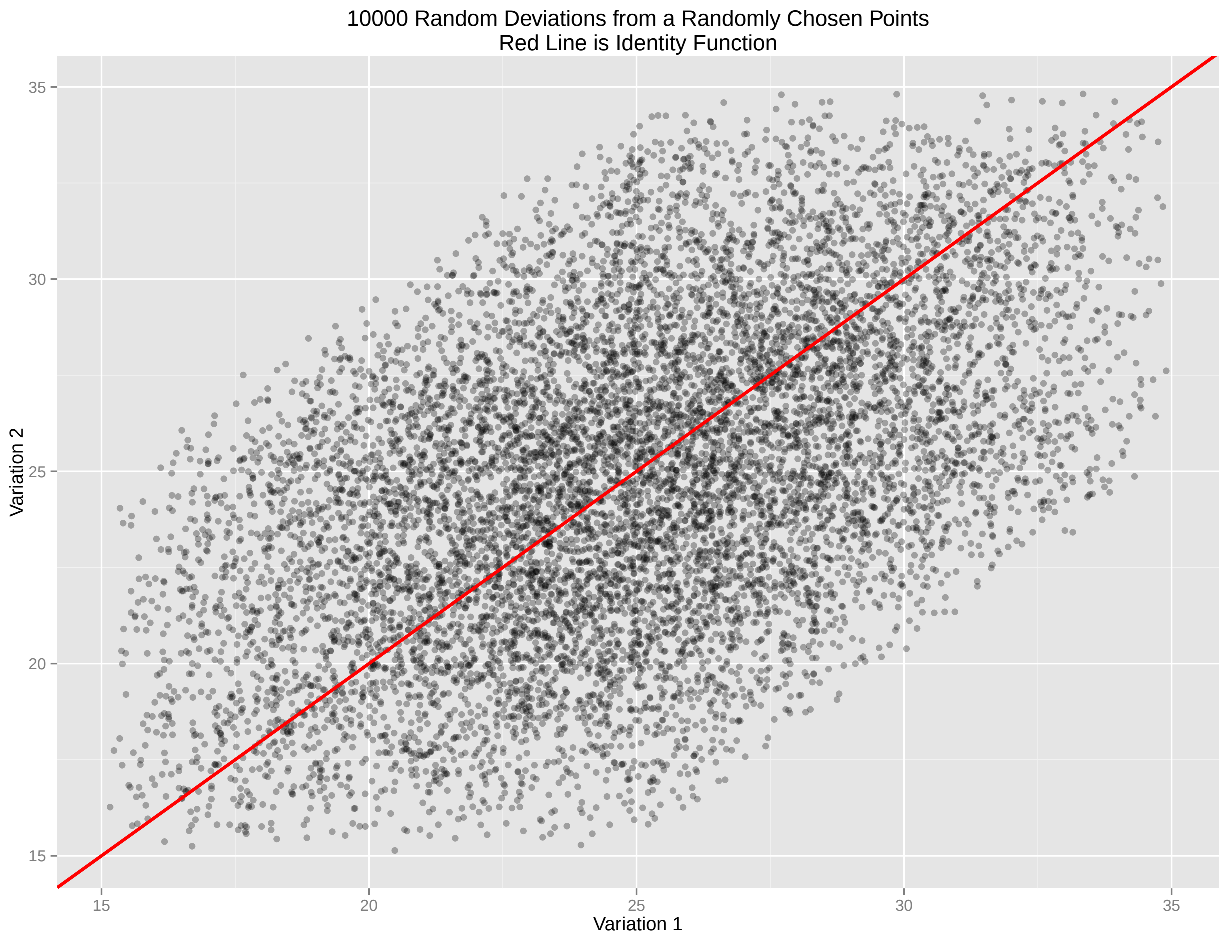
```
    variable     n  mean    sd
1 variation1 10000 24.99 4.056
2 variation2 10000 24.92 4.062
```

```
cor(my.df[,3:4])
```

```
           variation1 variation2
variation1     1.0000     0.4985
variation2     0.4985     1.0000
```

The following produces a scatter plot of the simulated correlated data.

```
ggplot(data=my.df, aes(x=variation1, y=variation2)) +
  geom_point(alpha=.3) +
  geom_abline(slope=1, intercept=0,color="red",size=1) +
  labs(x="Variation 1",
       y="Variation 2",
       title=paste0("10000 Random Deviations from a Randomly Chosen Points\n",
                    "Red Line is Identity Function"))
```



6.2 Simulating negatively correlated data

Introduce some linear algebra to get negative correlation:

```
for.rotating <- my.df[,c("variation1", "variation2")]

rotate.data <- function(theta=-pi/2, df=for.rotating){
  save.means <- colMeans(df)
  # Centers to origin
  df[,1] <- df[,1] - save.means[1]
  df[,2] <- df[,2] - save.means[2]
  # Rotates counter clockwise
  df <- as.matrix(df) %*% matrix(rbind(c(cos(theta), sin(-theta)),
                                       c(sin(theta), cos(theta))),ncol=2)
  # Reposition center
  df[,1] <- df[,1] + save.means[1]
  df[,2] <- df[,2] + save.means[2]
  df <- data.frame(df);
  names(df) <-c("variation1", "variation2")
  return(df)
}

for.rotating <- rotate.data(theta=-pi/2, df=for.rotating)

for.rotating.melt <- melt(for.rotating[,c("variation1", "variation2")])
```

We consider summary statistics to explore what's been created:

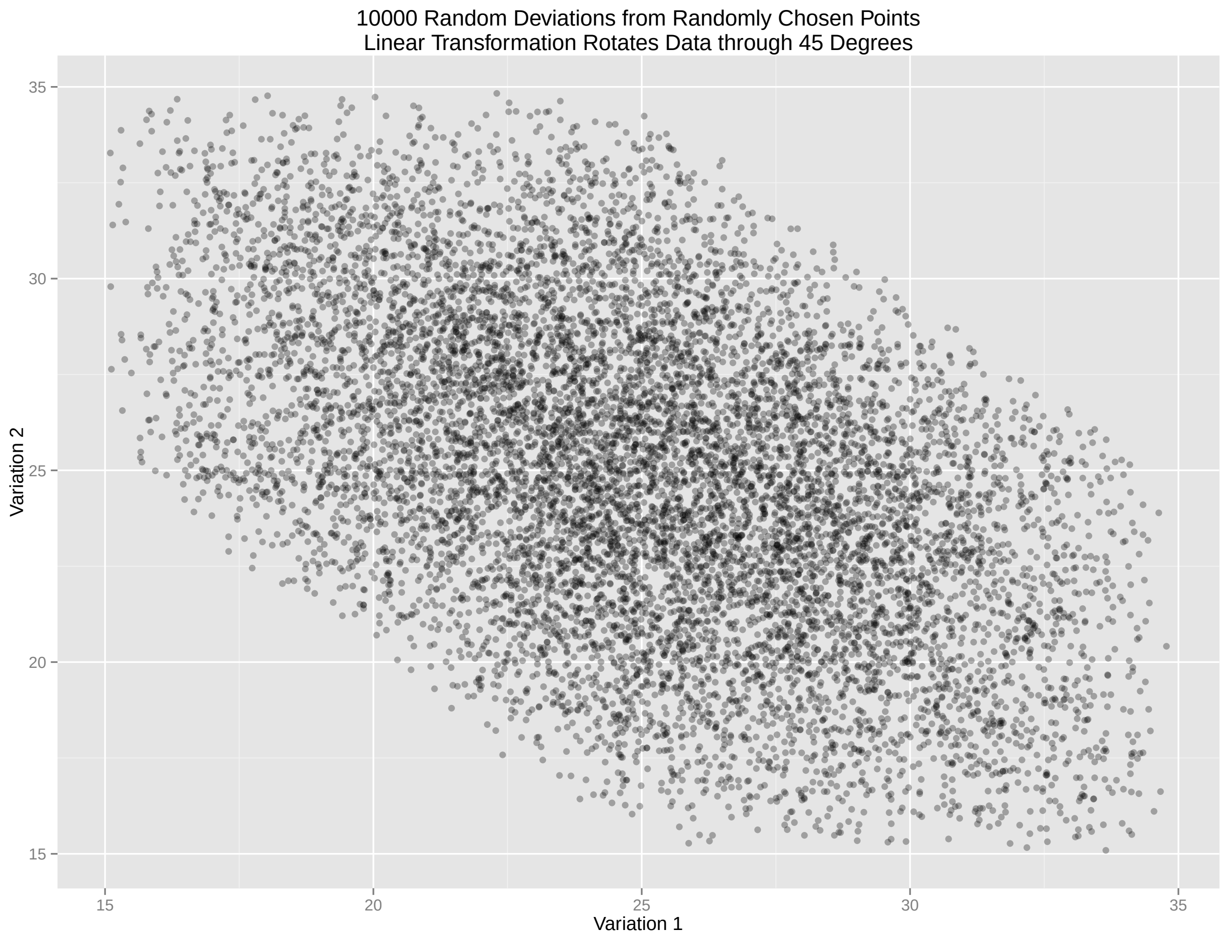
```
ddply(for.rotating.melt, .(variable), summarize,
      n=length(value), mean=mean(value), sd=sd(value))
```

```
    variable     n  mean    sd
1 variation1 10000 24.99 4.062
2 variation2 10000 24.92 4.056
```

```
cor(for.rotating$variation1, for.rotating$variation2)
```

```
[1] -0.4985
```

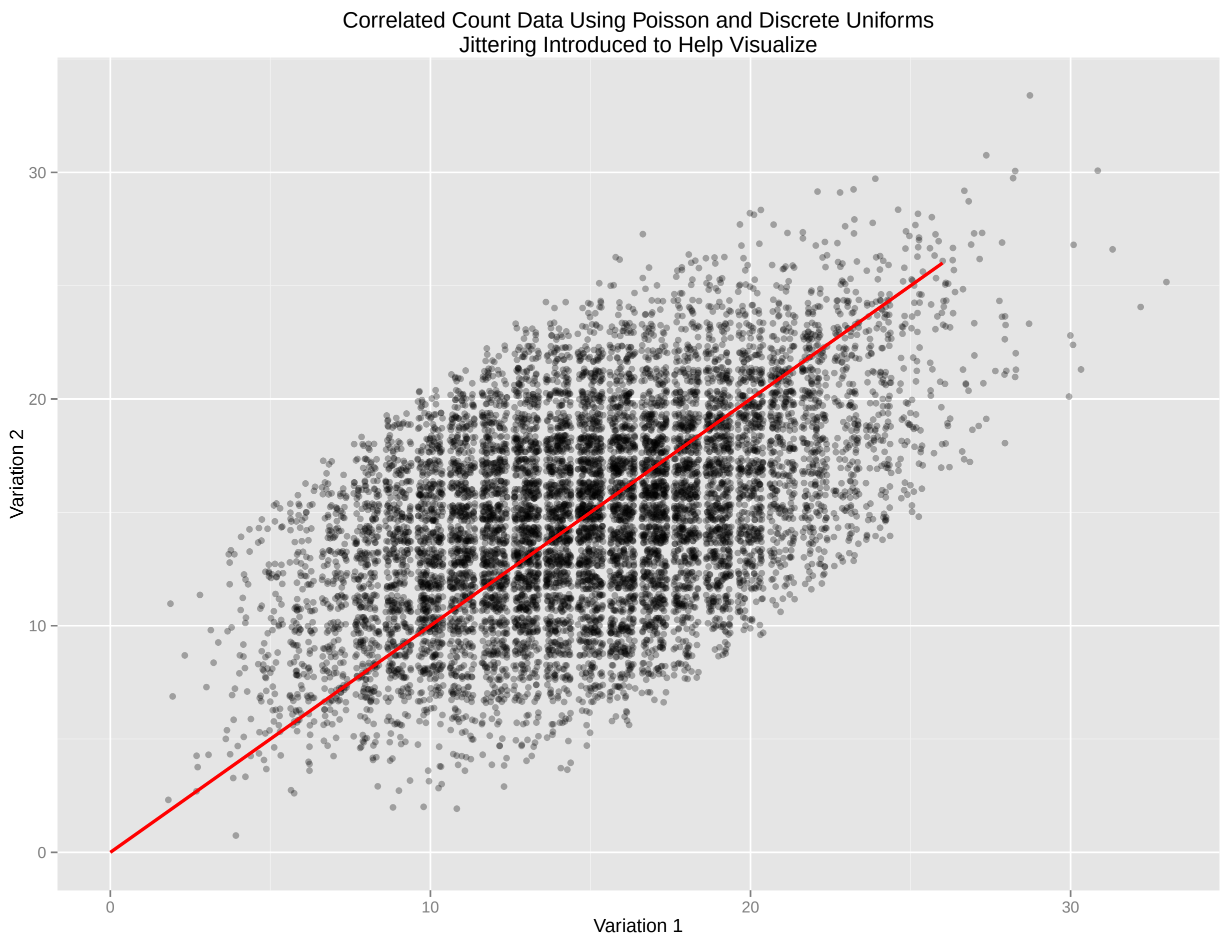
```
ggplot(data=for.rotating, aes(x=variation1, y=variation2)) +
  geom_point(alpha=.3) +
  labs(x="Variation 1",y="Variation 2",
       title=paste0("10000 Random Deviations from Randomly Chosen Points\n",
                    "Linear Transformation Rotates Data through 45 Degrees"))
```



6.3 Simulating correlated count data

The Poisson and discrete uniform distributions are used in this example.

```
my.df <- data.frame(observation=1:10000)
my.df$start.point <- rpois(10000, 10)
my.df$variation1 <- my.df$start.point + sample(x=0:10, size=10000, replace=TRUE)
my.df$variation2 <- my.df$start.point + sample(x=0:10, size=10000, replace=TRUE)

p1 <- ggplot(data=my.df, aes(x=variation1, y=variation2)) +
  geom_point(alpha=.3, position="jitter") +
  geom_line(aes(x=start.point,y=start.point),color="red",size=1) +
  labs(x="Variation 1",y="Variation 2",
       title=paste0("Correlated Count Data Using Poisson and Discrete",
                    " Uniforms\nJittering Introduced to Help Visualize"))
print(p1)
```



7 A post-mortem simulation

7.1 The setting

Trial conclusion results in retaining the null hypothesis associated with the primary analysis. A colleague poses a follow-up question: Of the 15 primary and secondary endpoints, 12 endpoints had treatment effect estimates on the favorable side of the null. What is the likelihood of that occurring.

7.2 A quick approach to the problem

Suppose the null hypothesis is true and the likelihood of observing an point estimate to the left or right of the null is 0.50. Then, treating the scenario as a binomial experiment, where outcomes (i.e., observing point estimates falling on the favorable side of the null) are considered to be independent, the probability of 12 or more ‘successes’ is:

```r
1 - pbinom(p = .5, size = 15, q = 11)
```

```
[1] 0.01758
```

7.3 Working towards a simulation that investigates the impact of correlation

Suppose the setting is an outcomes trial where the 15 observed point estimates are observed hazard ratios associated with time to event analyses using treatment as the only covariate. Further assume these hazard ratio estimates are based on the 15 event counts being simulated here:

```r
set.seed(1234)
observed.events <- rpois(15, lambda=seq(1000, 2000, length.out=15))
observed.events
```

```
 [1]  961 1081 1155 1188 1303 1335 1465 1475 1552 1571 1716 1826 1852 1929
[15] 2031
```

7.4 Motivating a multivariate normal model

In 1-dimimension the log of the observed hazard ratio is often modeled using a normal distribution with mean equal to the log of the true hazard ratio and variance equal to 4/(number of events). In what follows we generalize this idea. Suppose a vector of log observed hazard ratios is distributed according to a multivariate normal distribution. In this exercise, we will simulate under the null hypothsis and assume that true hazard ratios associated with the 15 time to event endpoints of interest are 1. Therefore we will take the a 15-tuple zero vector for the mean of the distribution. The variance matrix will be built by stipulating a diagonal matrix, **D**, that holds 4/(number of events), using the values generated above. For simplicity sake, we will assume a simple correlation structure by using matrix **R** having with 1's along the diagonal and a common correlation on the off-diagonal. The variance matrix will then be taken to be **RDR'**. The simulation will explore the role of increasing the common correlation from 0 through 0.9.

7.5 Building the simulation

Create a 15x15 diagonal matrix the individual variance values, 4/(number of events), to be married later with a correlation matrix, and a 15-tuple zero vector for the mean of the multivariate normal distribution.

```r
my.diag <- diag(4/observed.events)
my.mean <- rep(0, 15)
```

We introduce a helper function that produces correlation matrices with 1 on the diagonal and common off diagonal entries.

```r
get.corr <- function(n=15, rho=.2) {
  matrix(rep(rho, n*n), ncol=n) - diag(rho,n) +diag(1,n)}
```

The object simulated.count.data is initialized; the for–loop handle storage. Note that the mvtnorm::rmvnorm function is making use of the mean vector, the diagonal matrix and the loop–specific correlation matrix.

```r
simulated.count.data <- c()
for(i in seq(0, .9,.05)){
  my.corr <- get.corr(n=15, rho=i)
  simualted.HRs <- rmvnorm(100000, mean=my.mean,
                           sigma=my.diag%*%my.corr%*%my.diag)
  simulated.count.data <- cbind(simulated.count.data,
                                rowSums(sign(simualted.HRs)==-1))
  }

head(simulated.count.data)
```

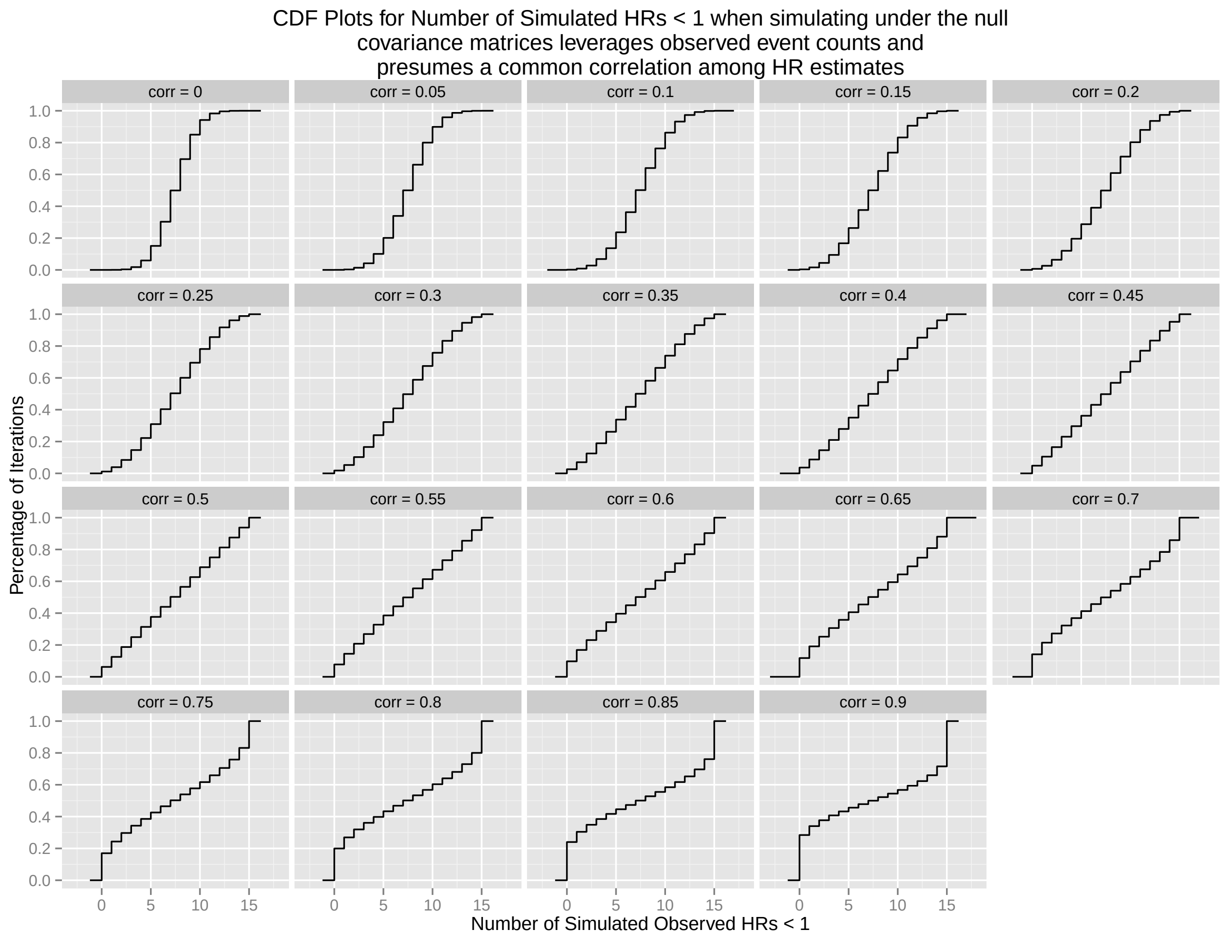
```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[1,]    5   13    8    6    8    8   12    5    7     5    13    14    12
[2,]    4    7   10    9   12    8    1    4   11    14     7     3     1
[3,]    9    6   10    8    8    4    1    7   13    13     3     4    15
[4,]    5    8    7   14    6    8   12   13   12    13     3     7     0
[5,]    8    6    3    9    9    2   10   11    9     5    15     8     2
[6,]    2    9   10   12    2    6    7    9    2     1     3     0     8
     [,14] [,15] [,16] [,17] [,18] [,19]
[1,]     0     3    15     5     8    15
[2,]     0     0    13     4    15    10
[3,]    15     3     0     0    15     6
[4,]    12     4    15     4     1    14
[5,]     0     2     4     0     1     1
[6,]     1     4    13    12     0     9
```

```
simulated.count.data <- data.frame(simulated.count.data)
names(simulated.count.data) <- paste0("corr = ", seq(0, .9,.05))
simulated.count.data$iteration <- 1:nrow(simulated.count.data)
melted.data <- melt(data=simulated.count.data, id="iteration")
head(melted.data)
```

```
  iteration variable value
1         1 corr = 0     5
2         2 corr = 0     4
3         3 corr = 0     9
4         4 corr = 0     5
5         5 corr = 0     8
6         6 corr = 0     2
```

7.6 Visualizing the results

```
ggplot(data=melted.data,
       aes(x=value, group=variable)) +
  stat_ecdf() +
  scale_y_continuous(breaks=seq(0,1,.05)) +
  facet_wrap(~variable) +
  scale_y_continuous(breaks=seq(0,1,.2)) +
  labs(x="Number of Simulated Observed HRs < 1", y="Percentage of Iterations",
       title=paste0("CDF Plots for Number of Simulated HRs < 1 when simulating",
                    " under the null\ncovariance matrices leverages observed",
                    " event counts and\npresumes a common correlation among HR",
                    " estimates"))
```



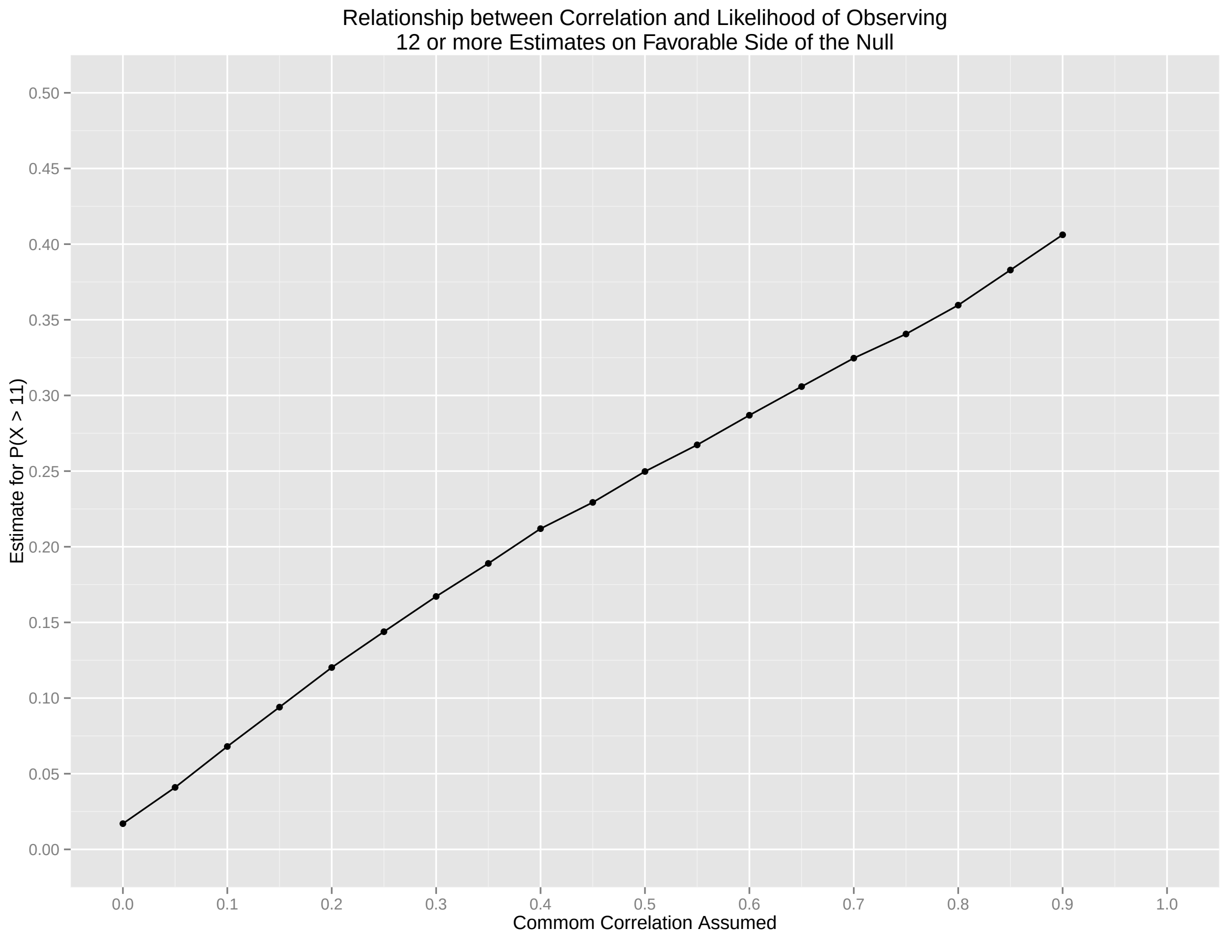
Here's a summary of the simulation results. Note that the first row provides a sanity check. This should match with the binomial case.

```
my.summary <- ddply(melted.data, .(variable),
                    summarize, "P(X > 11)"= mean(value >= 12))
print(my.summary)
```

```
      variable P(X > 11)
1      corr = 0   0.01700
2   corr = 0.05   0.04097
3    corr = 0.1   0.06800
4   corr = 0.15   0.09400
5    corr = 0.2   0.12019
6   corr = 0.25   0.14385
7    corr = 0.3   0.16715
8   corr = 0.35   0.18898
9    corr = 0.4   0.21196
10  corr = 0.45   0.22932
11   corr = 0.5   0.24975
12  corr = 0.55   0.26733
13   corr = 0.6   0.28691
14  corr = 0.65   0.30586
15   corr = 0.7   0.32464
```

```
16 corr = 0.75   0.34061
17  corr = 0.8   0.35967
18 corr = 0.85   0.38289
19  corr = 0.9   0.40617
```

```
ggplot(data=ddply(melted.data, .(variable), summarize, prop= mean(value >= 12)),
       aes(x=seq(0,.9,.05), y=prop)) +
  geom_point() +
  geom_line() +
  labs(x="Commom Correlation Assumed",
       y="Estimate for P(X > 11)",
       title=paste0("Relationship between Correlation and Likelihood of ",
                    "Observing\n12 or more Estimates on Favorable Side of the",
                    " Null")) +
  scale_y_continuous(limits=c(0,.5), breaks=seq(0,.5,.05)) +
  scale_x_continuous(limits=c(0,1), breaks=seq(0,1,.1))
```



8 Simulation to explore the robustness of efficacy to early withdrawals in an outcomes trial

8.1 Assumptions

All assumptions in this section are fabricated for the purpose of illustration. Let us take for granted the following trial design parameters and build some toy study data:

- Trial enrolls 16000 uniformly over 365 days
- Subjects are randomized 1:1 to placebo and treatment
- Placebo subjects have an annualized event rate of 4%
- Treated subjects are expected to have an annualized event rate of 3.2% (a 20% risk reduction)
- Time to event is exponentially distributed. Setting up an integral and solving $P(X < 365) = .04$ suggests;
- Exponential distribution with mean of -log(1 - .04/365) for placebo
- Similarly, an exponential distribution with mean -log(1 - .032/365) for treated

We will use the exponential distribution to assist in creating time to withdrawal, but expect this to be distorted since censoring is based on min(time to event, time to withdrawal, occurrence of the 1450th event of trial).

8.2 Preparing our fake trial data

8.2.1 Initialize a data.frame to build toy dataset

```r
set.seed(1234)
sim.trial <- data.frame(SUBJID = 1:16000,
                        TREATMENT = sample(size=16000,
                                           x=c("Placebo", "Treatment"),
                                           replace=TRUE),
                        event.time=0,
                        rand.day = sample(size=16000, x=1:365, replace=TRUE))

sim.trial$event.time <- floor(
  rexp(nrow(sim.trial),-log(1 -.04/365)))*(sim.trial$TREATMENT=="Placebo") +
  floor(rexp(nrow(sim.trial),-log(1 -.032/365)))*(sim.trial$TREATMENT=="Treatment")

sim.trial$withdraw.time <- floor(
  rexp(nrow(sim.trial),-log(1 -.20/365)))*(sim.trial$TREATMENT=="Placebo") +
  floor(rexp(nrow(sim.trial),-log(1 -.25/365)))*(sim.trial$TREATMENT=="Treatment")

head(sim.trial)
```

```
  SUBJID TREATMENT event.time rand.day withdraw.time
1      1   Placebo      14353      140           135
2      2 Treatment      19440      238            10
3      3 Treatment      20638      131           224
4      4 Treatment      16900      123           127
5      5 Treatment       3731       65           181
6      6 Treatment       1960      257           583
```

8.2.2 Event and withdrawal times

The exponentially distributed event times need to be shifted based on randomization date.

```
sim.trial$withdraw.day <- sim.trial$rand.day + sim.trial$withdraw.time
sim.trial$event.day <- sim.trial$rand.day + sim.trial$event.time
```

8.2.3 A flag variable

Add a flag variable to indicate when simulated event.day precedes simulated withdrawal day.

```
sim.trial$event.first <-  sim.trial$event.day < sim.trial$withdraw.day
```

8.2.4 Identify last event for trial

Determine the study day associated with the 1450th event. In real-world application, this will never happen. An added task is to estimate the overshoot (or undershoot) based on expectations regarding trial closeout.

```
# Subset, sort and identify the study day of the 1450th
sim.trial.observed.events <- subset(sim.trial, event.first==TRUE)
sim.trial.observed.events <- sim.trial.observed.events[
  order(sim.trial.observed.events$event.day),]
event.1450 <- sim.trial.observed.events[1450,]$event.day
event.1450
```

```
[1] 1657
```

8.2.5 Censoriong

Initialize columns to hold censored event time and censor for event and withdrawal.

```
head(sim.trial)
```

```
  SUBJID TREATMENT event.time rand.day withdraw.time withdraw.day
1      1   Placebo      14353      140           135          275
2      2 Treatment      19440      238            10          248
3      3 Treatment      20638      131           224          355
4      4 Treatment      16900      123           127          250
5      5 Treatment       3731       65           181          246
6      6 Treatment       1960      257           583          840
  event.day event.first
1     14493       FALSE
2     19678       FALSE
3     20769       FALSE
4     17023       FALSE
5      3796       FALSE
6      2217       FALSE
```

```
sim.trial$centime <- sim.trial$censor <- sim.trial$withdraw.censor <- 0
```

Some preparatory checks to ensure upcoming logic appropriately partitions the data set.

```
# These should match.
nrow(sim.trial)
```

```
[1] 16000
```

```
sum(sim.trial$event.first == TRUE & sim.trial$event.day <= event.1450) +
sum(sim.trial$event.first == TRUE & sim.trial$event.day > event.1450) +
sum(sim.trial$event.first == FALSE & sim.trial$event.day <= event.1450) +
sum(sim.trial$event.first == FALSE & sim.trial$event.day > event.1450)
```

```
[1] 16000
```

Code to populate censored time and censored variables

If the event occurs BEFORE withdrawal and the event occurs PRIOR to final event, then event is observed and centime is the # of days between randomization and event day + 1.

```
sim.trial[sim.trial$event.first == TRUE  &
            sim.trial$event.day <= event.1450,]$censor <- 1
sim.trial[sim.trial$event.first == TRUE &
            sim.trial$event.day <= event.1450,]$centime <-
  sim.trial[sim.trial$event.first ==TRUE &
              sim.trial$event.day <= event.1450,]$event.day -
  sim.trial[sim.trial$event.first == TRUE &
              sim.trial$event.day <= event.1450,]$rand.day + 1
```

If the event occurs BEFORE withdrawal and the event occurs AFTER final event, then the event is unobserved and centime is the # of days between randomization and the date of event.1450 occurs + 1.

```
sim.trial[sim.trial$event.first ==TRUE &
            sim.trial$event.day > event.1450,]$censor <- 0
sim.trial[sim.trial$event.first == TRUE &
            sim.trial$event.day > event.1450,]$centime <-
  event.1450 - sim.trial[sim.trial$event.first &
                           sim.trial$event.day > event.1450,]$rand.day
```

If the event occurs AFTER withdrawal and event occurs PRIOR to final event, then the withdrawal date is observed but event is not observed. Centime is the number of days between randomization day and withdrawal day + 1.

```
sim.trial[(sim.trial$event.first == FALSE) &
            sim.trial$event.day <= event.1450,]$censor <- 0
sim.trial[(sim.trial$event.first == FALSE) &
            sim.trial$event.day <= event.1450,]$withdraw.censor <- 1
sim.trial[(sim.trial$event.first == FALSE) &
            sim.trial$event.day <= event.1450,]$centime <-
  sim.trial[(sim.trial$event.first == FALSE) &
              sim.trial$event.day <= event.1450,]$withdraw.day -
  sim.trial[(sim.trial$event.first == FALSE) &
              sim.trial$event.day <= event.1450,]$rand.day + 1
```

If the event occurs AFTER withdrawal and event occurs AFTER final event, then neither event nor withdrawal is observed, centime is elapsed time between randomization date and day of event.1450 + 1.

```r
sim.trial[(sim.trial$event.first == FALSE) &
            sim.trial$event.day > event.1450,]$censor <- 0
sim.trial[(sim.trial$event.first == FALSE) &
            sim.trial$event.day > event.1450,]$withdraw.censor <- 0
sim.trial[(sim.trial$event.first == FALSE) &
            sim.trial$event.day > event.1450,]$centime <-
  event.1450 - sim.trial[(sim.trial$event.first == FALSE) &
                           sim.trial$event.day > event.1450,]$rand.day + 1
```

Here's where our fabricated trial data stands:

```r
head(sim.trial)
```

```
  SUBJID TREATMENT event.time rand.day withdraw.time withdraw.day
1      1   Placebo      14353      140           135          275
2      2 Treatment      19440      238            10          248
3      3 Treatment      20638      131           224          355
4      4 Treatment      16900      123           127          250
5      5 Treatment       3731       65           181          246
6      6 Treatment       1960      257           583          840
  event.day event.first withdraw.censor censor centime
1     14493       FALSE               0      0    1518
2     19678       FALSE               0      0    1420
3     20769       FALSE               0      0    1527
4     17023       FALSE               0      0    1535
5      3796       FALSE               0      0    1593
6      2217       FALSE               0      0    1401
```

8.3 Review of fabricated trial data

Some rudimentary checks.

```r
sum(sim.trial$censor)
```

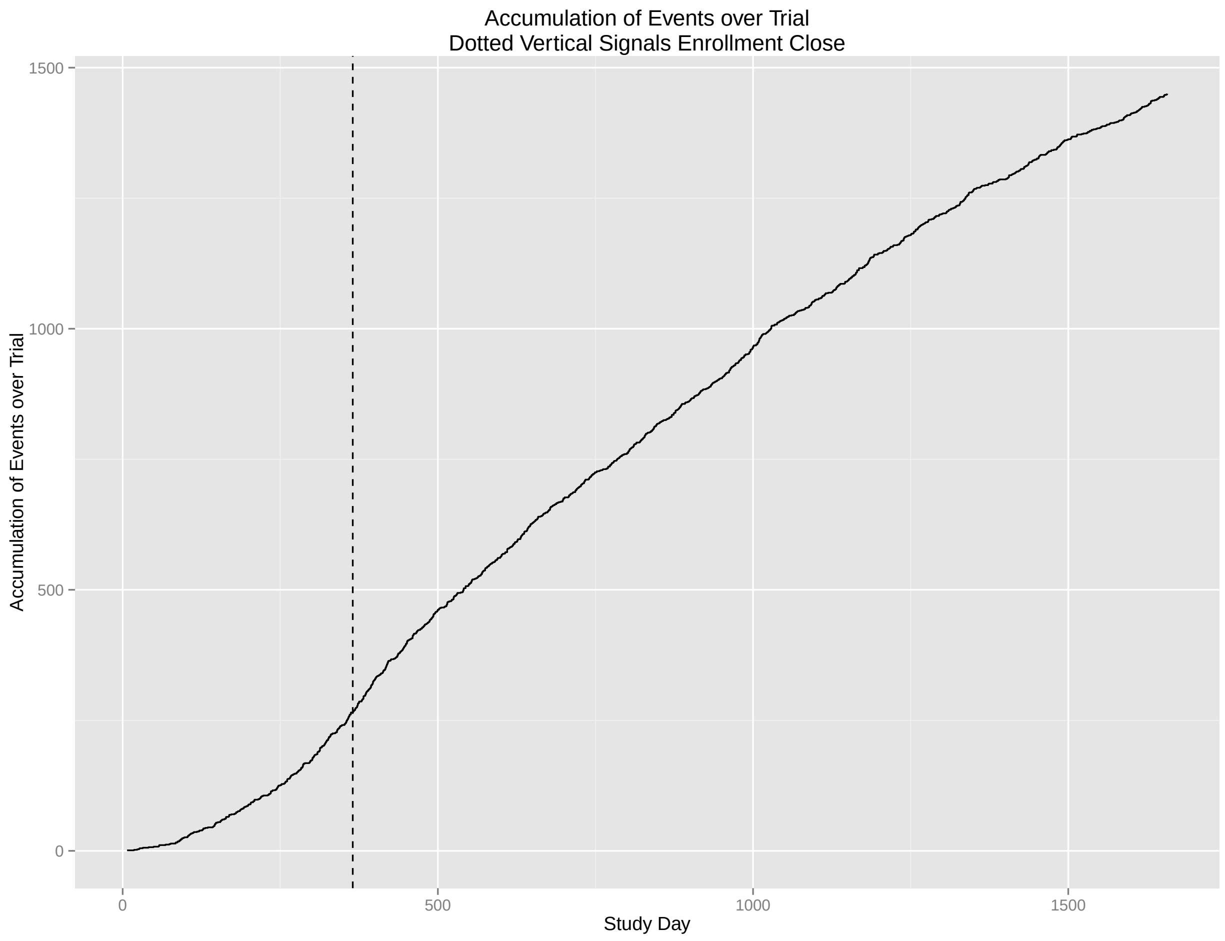
```
[1] 1450
```

```r
sum(sim.trial$withdraw.censor)
```

```
[1] 718
```

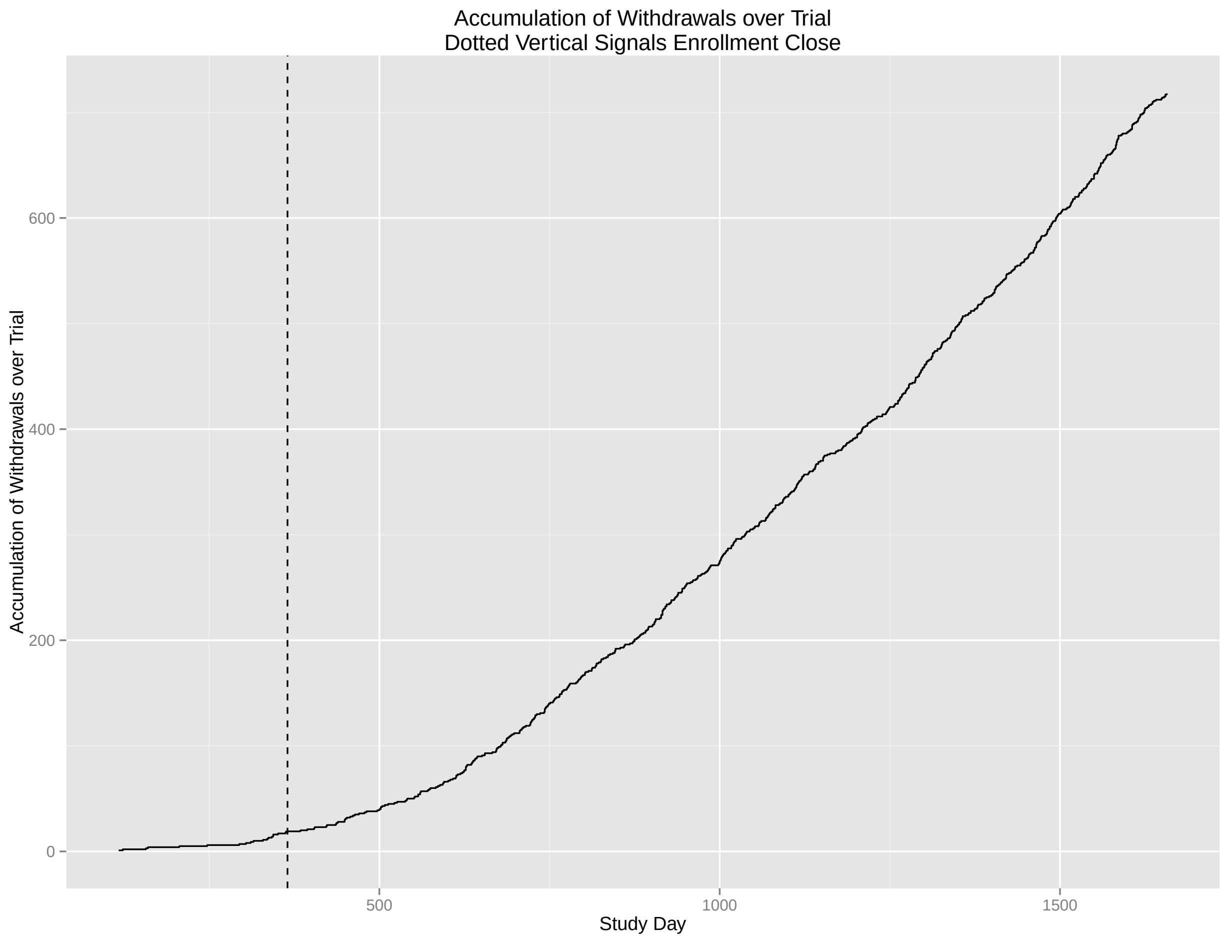
A check on how events accumulate by study day:

```r
sim.trial <- sim.trial[order(sim.trial$censor, sim.trial$event.day),]

ggplot(data=subset(sim.trial, censor==1),
       aes(x=event.day, y=1:nrow(subset(sim.trial, censor==1)))) +
  geom_step() +
  labs(x="Study Day",
       title=paste0("Accumulation of Events over Trial\nDotted Vertical Signals",
                    " Enrollment Close"),
       y="Accumulation of Events over Trial") +
  geom_vline(xintercept=365, linetype="dashed")
```



A check on how withdrawals accumulate by study day:

```
ggplot(data=subset(sim.trial, withdraw.censor==1),
       aes(x=event.day, y=1:nrow(subset(sim.trial, withdraw.censor==1)))) +
  geom_step() +
  labs(x="Study Day",
       title=paste0("Accumulation of Withdrawals over Trial\nDotted Vertical ",
                    "Signals Enrollment Close"),
       y="Accumulation of Withdrawals over Trial") +
  geom_vline(xintercept=365, linetype="dashed")
```



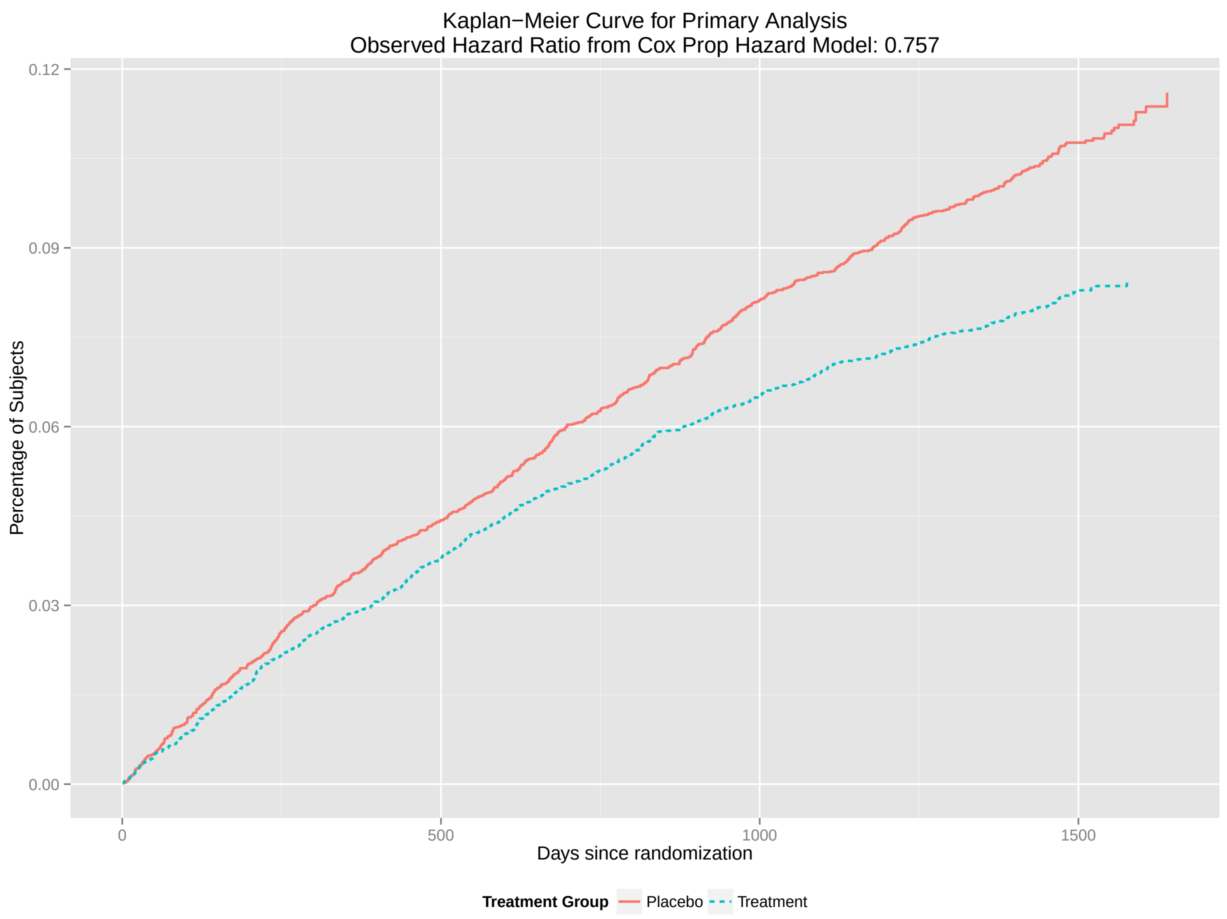
The primary analysis yields the following summary:

```
summary(coxph(data = sim.trial,Surv(centime,censor)~TREATMENT))$coefficients
```

```
                        coef exp(coef)  se(coef)      z  Pr(>|z|)
TREATMENTTreatment -0.2786    0.7569  0.05298 -5.258 1.457e-07
```

```
kmfit <- survfit(data=sim.trial, Surv(centime, censor) ~ TREATMENT)
kmfit.out <- data.frame(X = summary(kmfit)$time,
                        Y = 1 - summary(kmfit)$surv,
                        AT.RISK = summary(kmfit)$n.risk,
                        CATEGORY = factor(summary(kmfit)$strata))
levels(kmfit.out$CATEGORY) <- unlist(strsplit(x = levels(kmfit.out$CATEGORY),
    split = "="))[seq(2, nlevels(kmfit.out$CATEGORY) * 2, 2)]
```

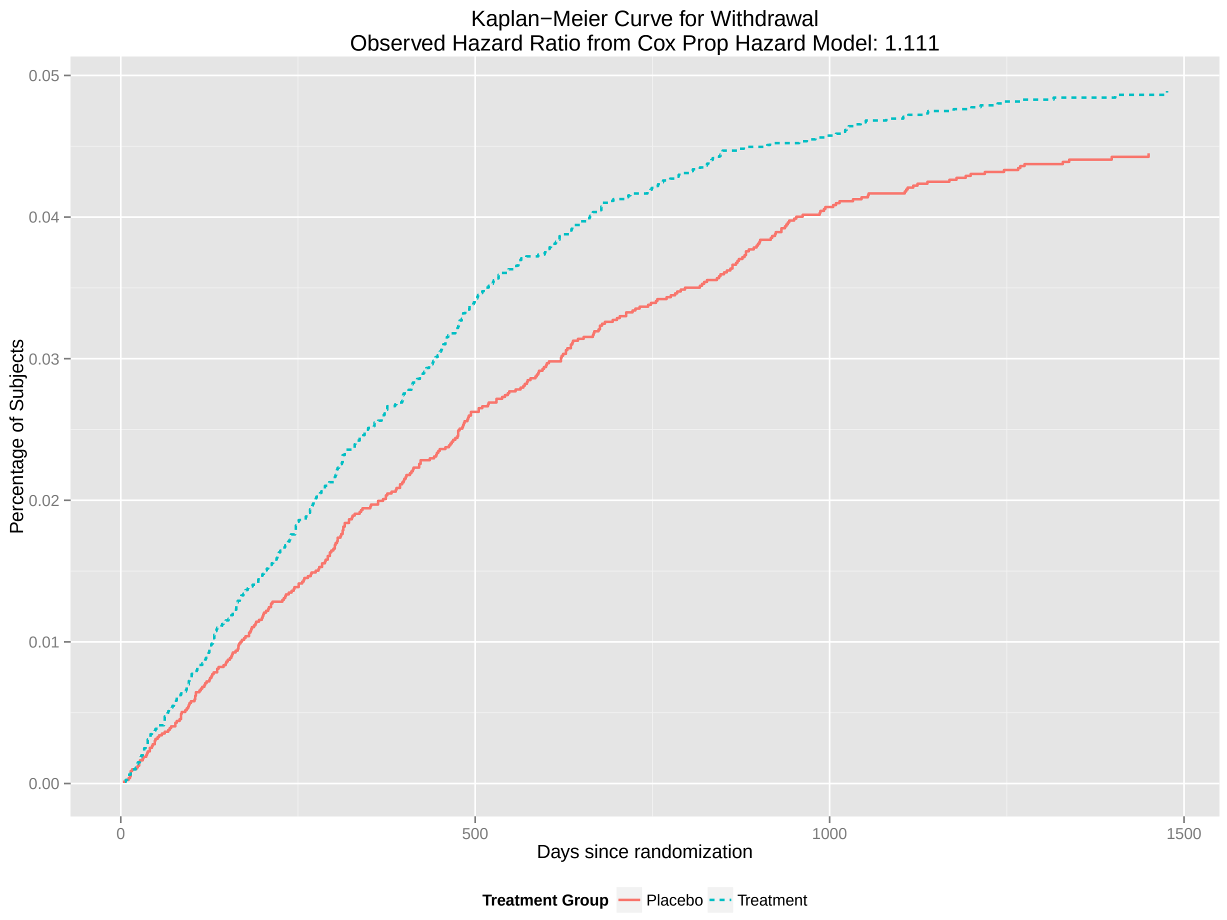
```
ggplot(data = kmfit.out, aes(x = X, y = Y, colour = CATEGORY,
                             linetype = CATEGORY)) +
  geom_step(size = .75) +
  theme(legend.position="bottom") +
  labs(x="Days since randomization",
       y="Percentage of Subjects",
       color="Treatment Group", linetype="Treatment Group",
       title=paste0("Kaplan-Meier Curve for Primary Analysis\nObserved Hazard ",
                    "Ratio from Cox Prop Hazard Model: ",
                    round(summary(coxph(data = sim.trial,Surv(centime,censor)~
                                          TREATMENT))$coefficients[2],3)))
```



The simulated withdrawal data and censoring imparted by code above yields:

```
kmfit <- survfit(data=sim.trial, Surv(centime, withdraw.censor) ~ TREATMENT)
kmfit.out <- data.frame(X = summary(kmfit)$time,
                        Y = 1 - summary(kmfit)$surv,
                        AT.RISK = summary(kmfit)$n.risk,
                        CATEGORY = factor(summary(kmfit)$strata))
levels(kmfit.out$CATEGORY) <- unlist(strsplit(x = levels(kmfit.out$CATEGORY),
    split = "="))[seq(2, nlevels(kmfit.out$CATEGORY) *
    2, 2)]
```

```
ggplot(data = kmfit.out, aes(x = X, y = Y, colour = CATEGORY,
                             linetype = CATEGORY)) +
  geom_step(size = .75) +
  theme(legend.position="bottom") +
  labs(x="Days since randomization",
      y="Percentage of Subjects",
       color="Treatment Group", linetype="Treatment Group",
       title=paste0("Kaplan-Meier Curve for Withdrawal\nObserved Hazard Ratio",
                    " from Cox Prop Hazard Model: ",
                    round(summary(coxph(
                      data = sim.trial,Surv(centime,withdraw.censor)~
                        TREATMENT))$coefficients[2],3)))
```



8.4 Imputation Step

Now onto our investigation: Suppose the withdrawers had remained under observation. Moreover, suppose some extreme assumptions for the sake of illustration: Placebo withdrawers have an (4*1.25)% annualized event rate and treated subjects have a (4*2)% annualized event rate.

Preparation includes partitioning the data set based on withdrawal censor:

```
sim.trial.A <- subset(sim.trial, withdraw.censor==0)
sim.trial.B <- subset(sim.trial, withdraw.censor==1)
```

An iteration will

- simulate and populate new values for event.time, event.day, centime and censor

- merge simulated data with original data from non-withdrawers
- re-run analysis on the merged data

```
sim.trial.B$censor<-  sim.trial.B$centime <- sim.trial.B$event.day <-
  sim.trial.B$event.time <- 0
sim.trial.B$event.time <-  floor(rexp(nrow(sim.trial.B),
       -log(1 -.04*1.25/365)))*(sim.trial.B$TREATMENT=="Placebo") +
 floor(rexp(nrow(sim.trial.B),-log(1 -.04*2/365)))*
  (sim.trial.B$TREATMENT=="Treatment")
sim.trial.B$event.day <- sim.trial.B$rand.day + sim.trial.B$event.time
sim.trial.B$centime <- ifelse(sim.trial.B$event.day <= event.1450,
                              sim.trial.B$event.day-sim.trial.B$rand.day + 1,
                              event.1450 - sim.trial.B$rand.day + 1)
sim.trial.B$censor <- (sim.trial.B$event.day <= event.1450)*1
mergeback <- rbind(sim.trial.A, sim.trial.B)
summary(coxph(data = mergeback,Surv(centime,censor)~TREATMENT))$coefficients
```

```
                     coef exp(coef) se(coef)      z  Pr(>|z|)
TREATMENTTreatment -0.201    0.8179  0.05027 -3.999 6.368e-05
```

8.5 Replication

The previous code is wrapped in a function:

```
my.iteration <- function(){
sim.trial.B$censor<-  sim.trial.B$centime <- sim.trial.B$event.day <-
  sim.trial.B$event.time <- 0
sim.trial.B$event.time <-  floor(
  rexp(nrow(sim.trial.B),log(1 -.04*1.25/365)))*
  (sim.trial.B$TREATMENT=="Placebo") +
  floor(rexp(nrow(sim.trial.B),
             -log(1 -.04*2/365)))*(sim.trial.B$TREATMENT=="Treatment")
sim.trial.B$event.day <- sim.trial.B$rand.day + sim.trial.B$event.time
sim.trial.B$centime <- ifelse(sim.trial.B$event.day <= event.1450,
                              sim.trial.B$event.day-sim.trial.B$rand.day + 1,
                              event.1450 - sim.trial.B$rand.day + 1)
sim.trial.B$censor <- (sim.trial.B$event.day <= event.1450)*1
mergeback <- rbind(sim.trial.A, sim.trial.B)
summary(coxph(data = mergeback,Surv(centime,censor)~TREATMENT))$coefficients
}
```

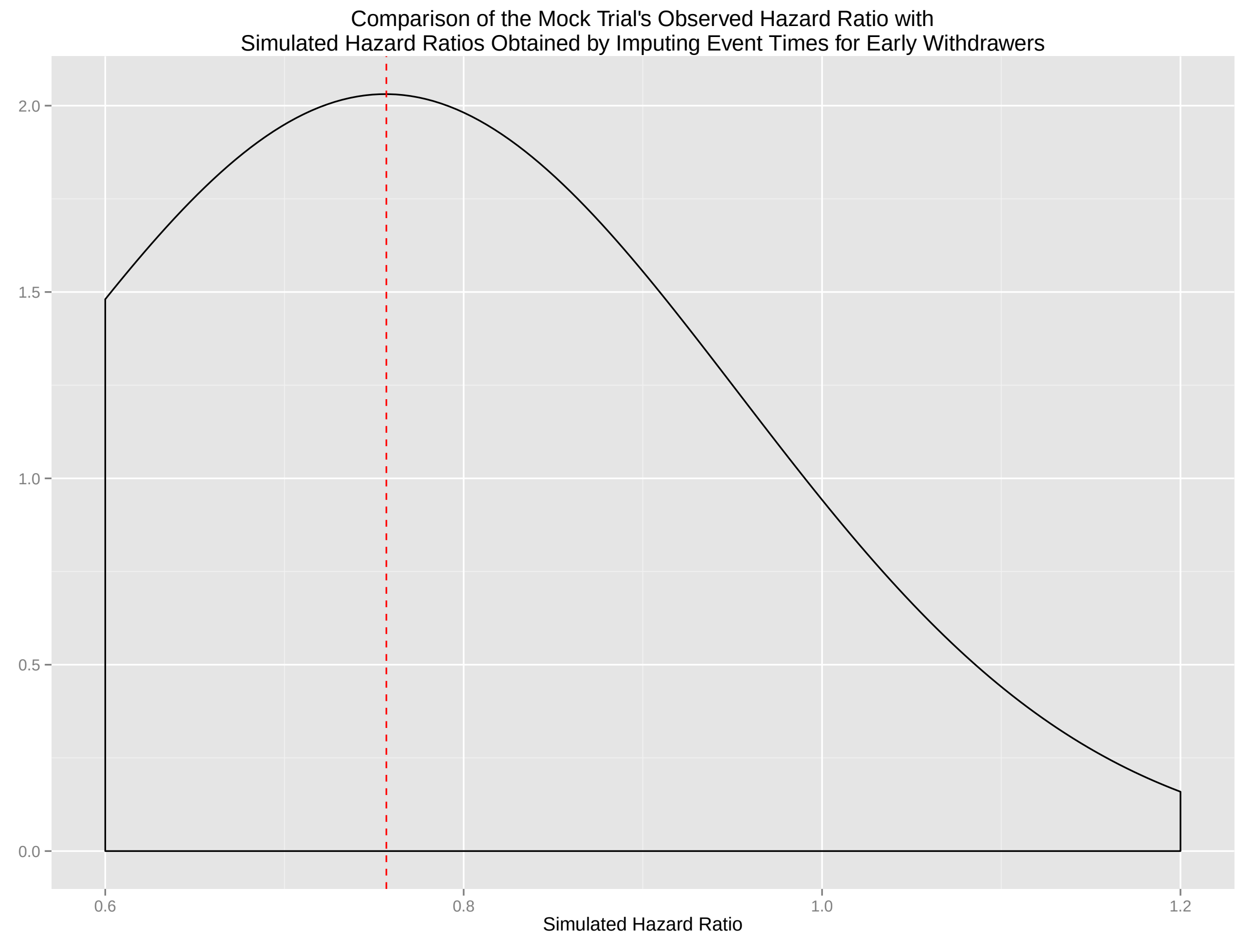
The function is replicated 1000 times. (Remove the .progress argument if you're running outside of Windows.)

```
coxph.results <- rdply(.n = 500, my.iteration,
                       .progress = progress_win(title="Working..."))
```

9 Visualizing the simulation results

Contrast the simulated results exploring the robust of primary efficacy analysis under the extreme assumptions made for withdrawers:

```r
names(coxph.results)[3] <- "HR"
ggplot(data=coxph.results, aes(x=HR)) +
  geom_density() +
  geom_vline(xintercept=summary(coxph(data = sim.trial,Surv(centime,censor)~
                                        TREATMENT))$coefficients[2],
             color="red", linetype="dashed") +
  xlim(.6, 1.2) +
  labs(x="Simulated Hazard Ratio", y=NULL, title=paste0(
    "Comparison of the Mock Trial's Observed Hazard Ratio with\n",
    "Simulated Hazard Ratios Obtained by Imputing Event Times for",
    " Early Withdrawers"))
```



10 References for tutorial

10.1 Recurrent event modeling

- Andersen, P. K. and Gill, R. D. (1982), Cox's Regression Model Counting Process: A Large Sample Study, Annals of Statistics, 10, 1100 1120.
- Prentice, R. L., Williams, B. J., and Peterson, A. V. (1981), On the Regression Analysis of Multivariate Failure Time Data, Biometrika, 68, 373 379.

- Wei, L. J., Lin, D. Y., and Weissfeld, L. (1989), Regression Analysis of Multivariate Incomplete Failure Time Data by Modeling Marginal Distribution, Journal of the American Statistical Association, 84, 1065 1073.

10.2 Enrollment modeling

- Anisimov, V., and Fedorov, V. Modelling, prediction, and adaptive adjustment of recruitment in multicenter trials. Stats in Medicine. 2007, 26. 4958–4975
- Katharine D Barnard, Louise Dent, Andrew Cook. A systematic review of models to predict recruitment to multicentre clinical trials. BMC Medical Research Methodology 2010, 10:63

10.3 Distributional relationships

- Bender R, Augustin T, Blettner M. Generating survival times to simulate Cox proportional hazards models. Statistics in Medicine 2005; 24:1713 1723.
- Leemis, L., McQueston, J. Univariate distribution relationships. The American Statistician, 2008. Vol 6. No. 1, 47.

10.4 Simulation size from a sequential estimation perspective

- Mukhopadhyay, N., Cicconetti, G. 2004. How Many Simulations Should One Run? In Applied Sequential Methodologies Real World Examples with Data Analysis (N. Mukhopadhyay, S. Datta, S. Chattopadhyay, eds.), 261–292. Marcel Dekker: New York
- Paper s Goal: Advertise 2–stage sequential sampling methodology in the setting of simulation size determination.

10.5 Assurance

- O'Hagan, A., Stevens, J. W. and Campbell, M. J. (2005), Assurance in clinical trial design. Pharmaceut. Statist., 4: 187 201. doi: 10.1002/pst.175
- Chuang–Stein, C. (2006), Sample size and the probability of a successful trial. Pharmaceut. Statist., 5: 305 309. doi: 10.1002/pst.232
- Carroll KJ. Decision Making from Phase II to Phase III and the Probability of Success: Reassured by 'Assurance'? 2013 Journal of Biopharmaceutical Statistics Vol 23(5): 1188–1200

10.6 R packages used for this document

10.6.1 For Deming document:

- H. Wickham. ggplot2: elegant graphics for data analysis. Springer New York, 2009.
- Hadley Wickham (2007). Reshaping Data with the reshape Package. Journal of Statistical Software, 21(12), 1–20. URL http://www.jstatsoft.org/v21/i12/.
- Hadley Wickham (2014). scales: Scale functions for graphics.. R package version 0.2.4. http://CRAN.R–project.org/package=scales
- Hadley Wickham (2011). The Split–Apply–Combine Strategy for Data Analysis. Journal of Statistical Software, 40(1), 1–29. URL http://www.jstatsoft.org/v40/i01/
- Hadley Wickham (2012). gtable: Arrange grobs in tables. R package version 0.1.2. http://CRAN.R-project.org/package=gtable

- Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1–25. URL http://www.jstatsoft.org/v40/i03/

- R Core Team (2014). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.
- Baptiste Auguie (2012). gridExtra: functions in Grid graphics. R package version 0.9.1. http://CRAN.R-project.org/package=gridExtra
- Erich Neuwirth (2011). RColorBrewer: ColorBrewer palettes. R package version 1.0–5. http://CRAN.R-project.org/package=RColorBrewer
- Therneau T (2014). A Package for Survival Analysis in S. R package version 2.37–7, .
- Venables, W. N. & Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0–387–95457–0 (MASS package)