Quincunx

A. Blejec

December 14, 2009

Abstract

Functions for dynamic presentation of Quincunx are given

1 Functions

```
> quincunx <- function(n = 80, mp = 8, sleep = 0.1, bins = NULL,
+     cexball = 2.5, cexpin = 1.2, prn = FALSE, seed = NULL) {
+     ball <- function(x, y, whipe = TRUE, sleep = 0.2) {
+         points(x, y, pch = pchball, cex = cexball, bg = bgball,
+             fg = fgball)
+         Sys.sleep(sleep)
+         if (whipe)
+             points(x, y, pch = 16, cex = cexball, col = "white")
+     }
+     odd <- function(x) {
+         x != as.integer(x/2) * 2
+     }
+     even <- function(x) {
+         x == as.integer(x/2) * 2
+     }
+     if (is.null(seed))
+         my.seed <- round(as.numeric(Sys.time()))
+     else my.seed = seed
+     set.seed(my.seed)
+     oldpar <- par(mar = c(1, 1, 1, 1))
+     on.exit(par(oldpar))
+     np <- 10
+     dx <- 0.5
+     uy <- 4
+     ly <- 8
+     colpin <- 1
+     pchpin <- 21
+     bgball <- "grey"
+     fgball <- "blue"
+     pchball <- 21
+     coltrace <- "red"
+     collwd <- 1
+     const <- 1
+     dy = 0.5
+     eps <- dy/4
```

```
+     x1 <- seq(1, np, 2 * dx)
+     y1 <- seq(mp, 1, -1) * const
+     xs <- rep(c(x1 + 0.5, x1), mp/2)
+     ys <- rep(y1, each = length(x1))
+     par(pty = "s")
+     plot(xs, ys, type = "n", xlim = c(0, np + 1.5), ylim = c(-ly,
+         mp * const + uy), axes = FALSE, xlab = "", ylab = "")
+     box()
+     rug(seq(0, np + 1, 1), 0.3, col = "grey", lwd = 2)
+     points(xs, ys, pch = pchpin, cex = cexpin, bg = colpin, fg = colpin,
+         ylim = c(-mp, mp))
+     xd <- seq(0, np + 1, 0.1)
+     yd <- dnorm(xd, mean(x1), sqrt(n/4)/3) * n/3.5 - ly - 1
+     lines(5.5 - c(2, 0.3, 0.3), c(12.4, 9.9, 9), lwd = 2, col = "black")
+     lines(5.5 + c(0.3, 0.3), c(12.5, 9), lwd = 2, col = "black")
+     ball(5.5, 9.4, FALSE)
+     for (j in 0:3) for (i in 0:j) ball(5.5 - i * 0.5, 10 + j *
+         0.7, FALSE)
+     if (is.null(bins)) {
+         bins <- numeric(np + 1)
+         for (i in 1:n) {
+             dy <- 0.5
+             starty <- mp * const + dy
+             midx <- mean(x1)
+             x <- midx
+             xt <- x
+             yt <- starty + dy
+             for (y in seq(starty, 1, -dy)) {
+                 if (y%%1 == 0)
+                   x = x + sign(runif(1, -1, 1)) * dx
+                 xt <- c(xt, x)
+                 yt <- c(yt, y)
+                 ball(x, y + eps, sleep = sleep)
+                 lines(xt, yt, col = coltrace, lwd = collwd)
+             }
+             binx <- x + 0.5
+             bins[binx] = bins[binx] + 1
+             while (y > -ly - 1 + bins[binx] * dy) {
+                 ball(x, y, sleep = sleep)
+                 y <- y - dy
+             }
+             y <- -ly - 1 + bins[binx] * dy
+             epsx <- sign(5 - x) * (odd(bins[binx]) - 0.5) * 0.5
+             epsy <- even(bins[binx]) * 0.5
+             epsy2 <- ((bins[binx] - 1)%/%2) * 0.35
+             ball(x + epsx, y - epsy - epsy2, FALSE)
+             lines(xt, yt, col = "white", lwd = collwd)
+         }
+     }
+     else {
+         for (binx in 1:length(bins)) {
+             x <- binx - 0.5
+             bins.old <- bins
```

```
+             bins[binx] = 0
+             while (bins[binx] < bins.old[binx]) {
+                 bins[binx] = bins[binx] + 1
+                 y <- -ly - 1 + bins[binx] * dy
+                 epsx <- sign(5 - x) * (odd(bins[binx]) - 0.5) *
+                   0.5
+                 epsy <- even(bins[binx]) * 0.5
+                 epsy2 <- ((bins[binx] - 1)%/%2) * 0.35
+                 ball(x + epsx, y - epsy - epsy2, FALSE)
+             }
+         }
+     }
+     dy <- 0.5
+     starty <- mp * const + dy
+     midx <- mean(x1)
+     x <- midx
+     xt <- x
+     yt <- starty + dy
+     X <- rep(1:length(bins), bins) - 0.5
+     m <- mean(X)
+     s <- sd(X)
+     n <- sum(bins)
+     cat("Mean=", m, "\nSD=", s, "\n")
+     lines(xd, dnorm(xd, m, s) * n/3 - ly - 1, col = "green",
+         lwd = 3)
+     x <- midx
+     xt <- x
+     yt <- starty + dy
+     for (y in seq(starty, 4 - dy, -dy)) {
+         if (y%%1 == 0)
+             x = x + sign(runif(1, -1, 1)) * dx
+         xt <- c(xt, x)
+         yt <- c(yt, y)
+         ball(x, y + eps, sleep = sleep)
+         lines(xt, yt, col = coltrace, lwd = collwd)
+     }
+     ball(x, y + eps, FALSE)
+     lfn <- paste("Q", my.seed, ".WMF", sep = "", collapse = "")
+     cat("Bins:\n", bins, "\n")
+     return(list(file = lfn, seed = my.seed, bins = bins))
+ }
```

2 Examples

SLOW MOTION DEMO

```
> quincunx(n = 10)
```

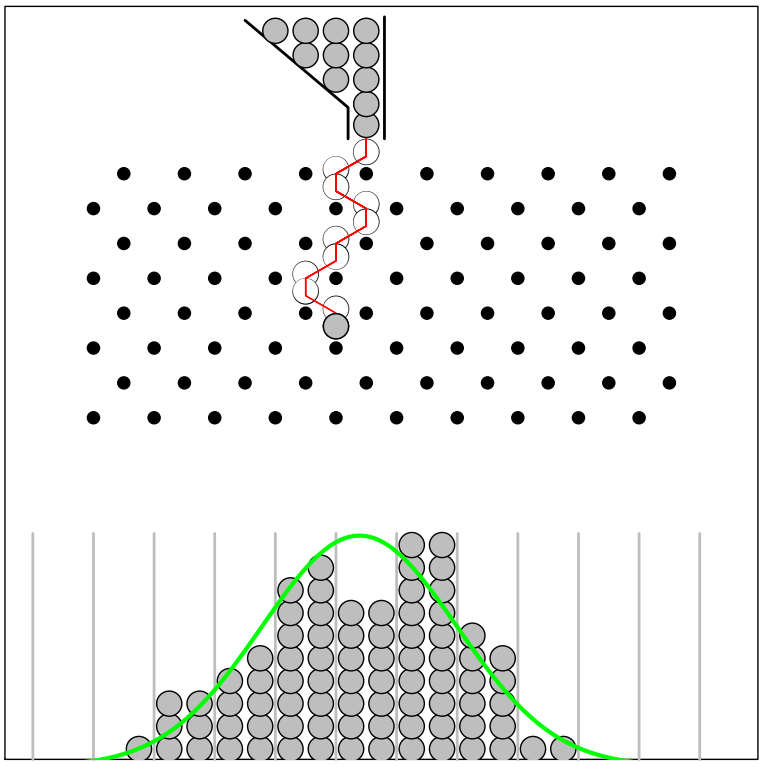
Fast demo, result saved into x for later use

```
> x = quincunx(sleep = 0)
```

```
Mean= 5.3875
SD= 1.606898
Bins:
 0 1 6 9 17 14 20 11 2 0 0
> x
$file
[1] "Q1260798682.WMF"

$seed
[1] 1260798682

$bins
 [1]  0  1  6  9 17 14 20 11  2  0  0
```

Replot saved simulation

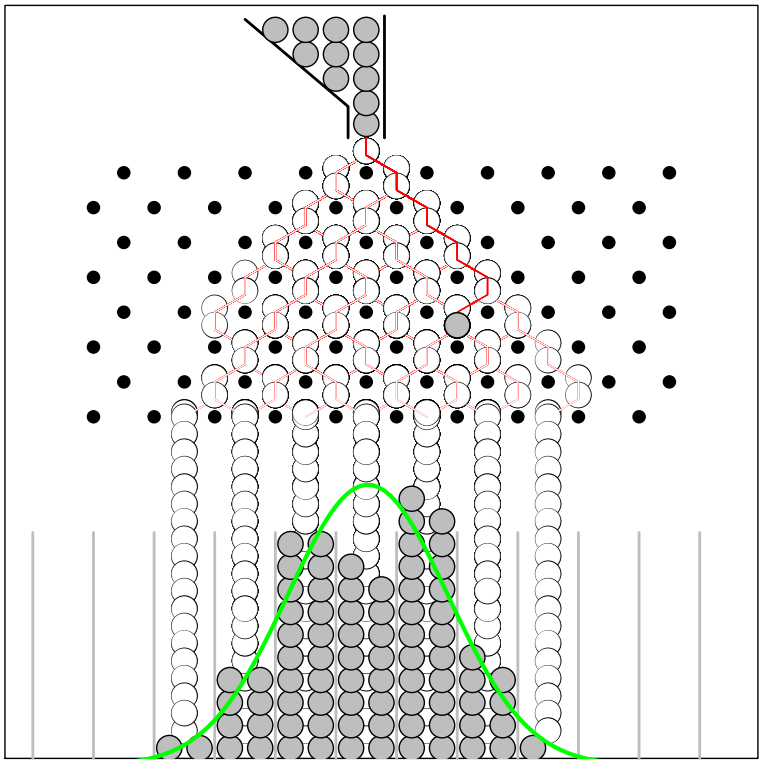
```
> quincunx(bins = x$bins)
Mean= 5.3875
SD= 1.606898
Bins:
 0 1 6 9 17 14 20 11 2 0 0
$file
[1] "Q1260798702.WMF"

$seed
[1] 1260798702

$bins
 [1]  0  1  6  9 17 14 20 11  2  0  0
```



Reset random number generator to make two identical runs

```
> set.seed(1234)
> x = quincunx(sleep = 0)
```

```
Mean= 5.6
SD= 1.506106
Bins:
 0 1 2 11 12 17 25 10 1 1 0
```



```
> set.seed(1234)
> x = quincunx(sleep = 0)
```