Cellular Automata

Generating semi-random numbers

Thomas Fischer
09-930-090

17 September 2018

Contents

1 Code for simulating cellular automata

```
library(raster)
```

```
## Loading required package: sp
```

```
library(sp)

# Convert Rule Nr (Integer) to bits
ruleToBits <- function(int) {
  if (int > 255 | int < 0) stop("Invalid Rule (0-255)")

  bits <- vector(length = 8)
  for (i in 7:0) {
    tmp <- int %% 2^i
    if (tmp != int) {
      bits[7 - i + 1] <- 1
      int <- tmp
      next
    }
    else {
      bits[7 - i + 1] <- 0
    }
  }
  return(bits)
}

# Compare Neighborhood to Ruleset
rule <- function(nb, bits) {
  int <- sum(nb * 2^(2:0))
  return(ifelse(bits[7 - int + 1] == 1, 1, 0))
}

# Horizontal wrap, pass neighborhoods to rule function
applyRule <- function(seq, bits) {
  new <- vector(length = length(seq))
  tmp <- seq[c(length(seq), 1:length(seq), 1)]

  for (i in (1:length(seq))) {
```

```
    nb <- tmp[c(i + 0:2)]
    new[i] <- rule(nb, bits)
  }
  return(new)
}

# Run Cellular Automata
sim <- function(seed, n, rule) {
  grid <- matrix(0, nr = n, nc = length(seed))
  grid[1,] <- seed
  bits <- ruleToBits(rule)

  for (i in 2:n) {
    seed <- applyRule(seed, bits)
    grid[i,] <- seed
  }
  g <- raster(grid)
  plot(g, xaxt = "n", yaxt = "n", axes = FALSE,
       legend = F, bty = "l", box = FALSE,
       mar = c(0,0,0,0))
  return(g)
}

# Initial values (seed)
seq <- c(rep(0, 30), 1, rep(0, 30))
```

```
sim(seed = seq, n = 30, rule = 30)
```

```
sim(seed = seq, n = 30, rule = 150)
```

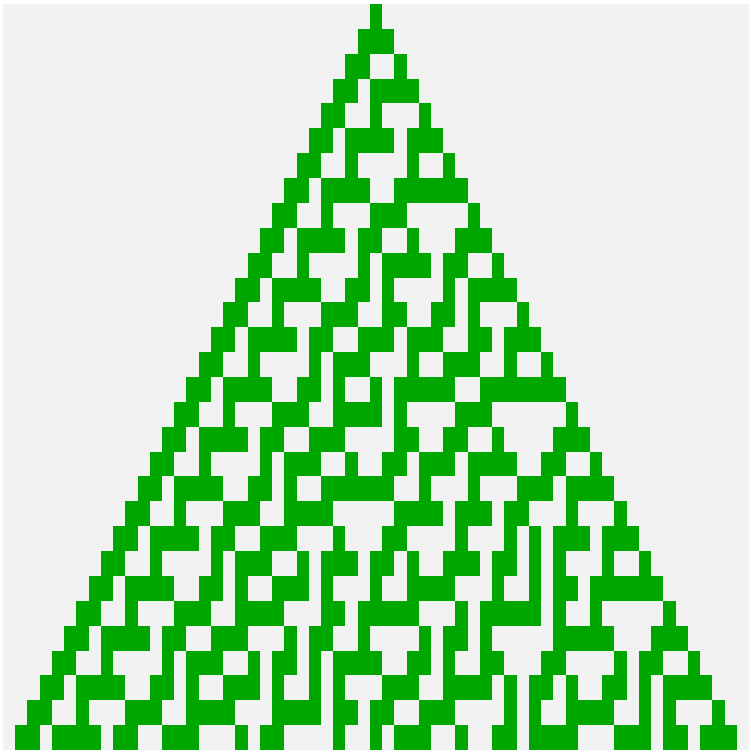

Figure 1: Rule 30

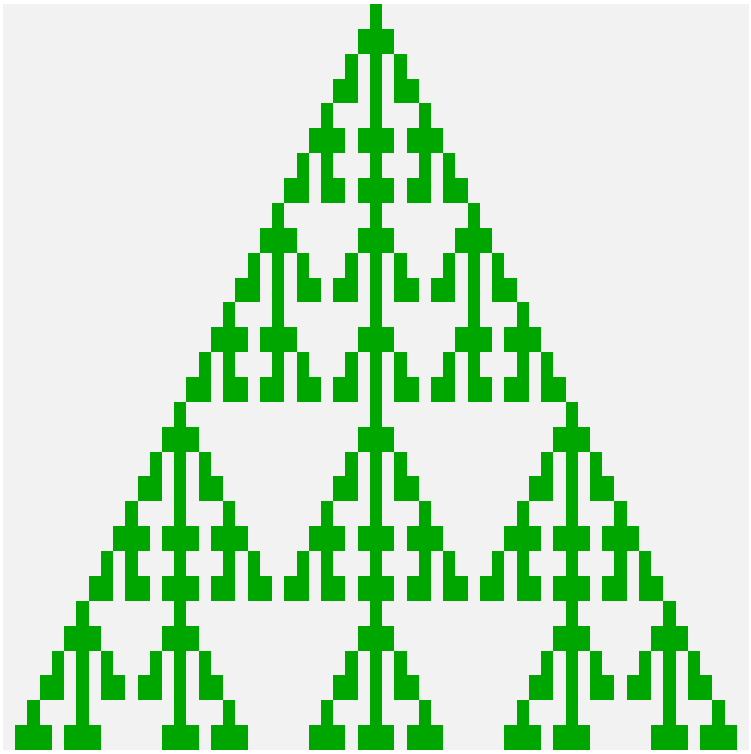

Figure 2: Rule 150