Runs Test

Markus Kalisch, Lukas Meier

Einführung

Die Schüler einer Klasse bekommen die Hausaufgabe, eine zufällige Sequenz der Länge 200 aus den Ziffern 0 und 1 zu bilden. Damit die Sequenz auch wirklich zufällig ist, soll mit einer Münze geworfen werden (Kopf = 0, Zahl = 1). Zweihundertmal. Puhh, nicht nur bei schönem Wetter verliert man hier die Lust. Pragmatische Schüler setzen sich nun an den Computer und tippen in schneller Folge und rein willkürlich zweihundertmal auf die Tasten '0' und '1'. Willkürlich ist doch eigentlich das gleiche wie zufällig, oder etwa nicht?

"Echte" Zufallszahlen sind schwierig zu produzieren. Der Computer produziert in der Tat auch "nur" deterministische Pseudo-Zufallszahlen (das ist ein eigenes Kapitel für sich). Die werden aber dermassen schlau produziert, dass sie für die meisten Anwendungen "zufällig genug" sind. Wir gehen im Folgenden daher davon aus, dass der Computer "echte" Zufallszaheln produzieren kann. Stimmt das auch für einen Schüler, der versucht eine zufällige Sequenz von 0ern und 1ern zu erzeugen? Falls die Sequenz sehr kurz ist, ist das sicher kein Problem (z.B. wenn man nur eine Zahl aus 0 oder 1 wählen soll). Wenn die Sequenz aber lang wird, könnte es sein, dass der Mensch deutlich vom Zufall abweicht.

Um solche Abweichungen zu prüfen, gibt es Testprozeduren für (Pseudo-) Zufallszahlengeneratoren. Eine verbreitete solche Prozedur sind die "Diehard Tests" (https://en.wikipedia.org/wiki/Diehard_tests). Diese Prozedur besteht aus einigen statistischen Tests. Einen davon, den Runs-Test, schauen wir genauer an.

Schauen wir uns mal die Sequenz 00111011 an. Erst kommen zwei 0er, der "erste Run (=gleiche Ziffer)". Dann kommen drei 1er; das ist der zweite Run. Dann kommt eine 0; das ist der dritte Run. Und schliesslich kommen noch zwei 1er; das ist der vierte und letzte Run. Die Sequenz hat also 4 Runs. Schauen wir uns nun die gleich lange Sequenz 00000000 an. Diese Sequenz hat nur einen Run. Die gleich lange Sequenz 01010101 hat acht Runs. An diesem Beispiel kann man schon erahnen, dass eine zufällige Sequenz wahrscheinlich nicht zu viele und nicht zu wenige Runs aufweist. Die Anzahl Runs gibt uns also eine Möglichkeit, die Zufälligkeit einer Sequenz zu testen. Soweit die Intuition. Jetzt versuchen wir das Problem gemäss dieser Intuition etwas präziser zu lösen.

Beobachteter Wert: Anzahl Runs

Ein Schüler schickt mir per Email eine recht lange willkürlich getippte Sequenz aus 0ern und 1ern ohne Trennzeichen (das geht viel schneller zu tippen und die Schüler machen eher mit). Die Sequenz muss mind. 200 Zeichen enthalten (evtl. muss man den Schülern hier eine Vorgabe machen, z.B. zwei Zeilen voll, oder so, damit sie nicht zählen müssen). Diese Zeichenkette kopiere ich via Copy/Paste nach R):

```
x <-
"00101001011101010101111000110101001000101111010111001010101010100100111110010101111011101100100100010101101010010011100101100011100101010100001110101011011100001010011111000110010100110011001110011001010011001"
```

Meine Zeichenkette enthält 206 Ziffern. Als Service für die Lehrer kommt hier eine Funktion, die diese Zeichenkette in einen Vektor mit 0ern und 1ern umwandelt und auch prüft, ob mind. 200 Elemente vorhanden sind:

```
string2vec <- function(x, n) { ## n: min. Anzahl Ziffern in Sequenz
  nx <- nchar(x) ## Anzahl Ziffern in x
  if (nx < n) {
    stop("Nicht genug Ziffern in Squenz")
  }
  tmp <- substring(x, 1:nx, 1:nx)

  if (!all( unique(tmp) %in% c("0", "1") ) )
    stop("Ungültige Ziffern in Sequenz!")

  zTmp <- as.numeric(tmp) ## Zeichenkette als Vektor
  z <- zTmp[1:n] ## Vektor auf Laenge n gekuerzt
  z
}
```

Nun zählen wir die Runs in dieser Sequenz. Dazu definieren wir zunächst eine Funktion:

```
runs <- function(x) {
  ## Zur Sicherheit nochmals ein Check:
  if (!all(unique(x) %in% c(0,1)))
    stop("x darf nur die Werte 0 oder 1 enthalten")
  cnt <- 0 ## Zaehlvariable fuer runs
  last <- -1000 ## Wert, der in x sicher nicht vorkommt

  for (i in 1:length(x)) {
    if (x[i] != last) {
      cnt <- cnt + 1
      last <- x[i]
    }
  }
  cnt
}
```

Nun können wir die beiden Funktionen auf unsere Zeichenkette anwenden:

```
nZ <- 200 ## Verlangen 200 Zeichen pro Sequenz
tmp <- string2vec(x, n = nZ)
nmbRuns <- runs(tmp)
```

Wir haben in unserer Zeichensequenz 125 runs. Wie vergleicht sich das mit den Anzahlen von Runs in echten Zufallssequenzen?

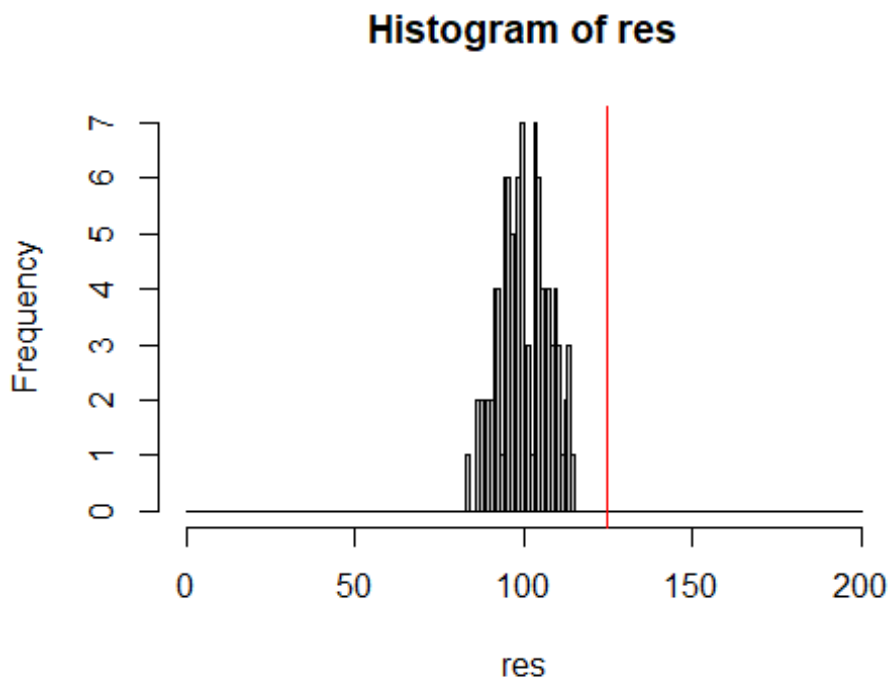
Was erwarten wir, wenn nur der Zufall am Werk ist?

Wir simulieren viele Zufallssequenzen. Dazu verwenden wir die Funktion `rbinom` (siehe `?rbinom`).

```
set.seed(123) ## Damit Schüler das Ergebnis reproduzieren können, setze ich
einen Random Seed
nreps <- 100 ## Anzahl Zufallssequenzen
res <- vector("numeric", nreps) ## Bereite Vektor fuer Ergebnis vor
for (i in 1:nreps) {
  x <- rbinom(n = nZ, size = 1, prob = 0.5) ## nZ Münzwürfe
  res[i] <- runs(x) ## Zähle Anzahl runs und speichere ab
}
```

Min Runs: 84, Max Runs: 115. Jetzt kommt der eigentliche Runs-Test: Wir stellen die gefundenen Anzahlen der Runs aus den Zufallssequenzen mit einem Histogramm dar und zeichnen die Anzahl Runs aus unserer eigenen Sequenz mit einer vertikalen Linie ein:

```
## Zeige Histogramm der Anzahl Runs in den simulierten Sequenzen
hist(res, breaks = 0:nZ)
## Zeige Anzahl Runs in der selbst erzeugten Sequenz
abline(v = nmbRuns, col = "red")
```



Schlussfolgerung

Wir sehen, dass unsere selbst erzeugte Sequenz nicht sehr gut zu den echten Zufallssequenzen passt. Nur 0 der 100 Zufallssequenzen hatten mehr runs als die von uns erzeugte Sequenz.

Ausblick

Bei kurzen Sequenzen (<100) ist es relativ leicht, eine Sequenz zu erzeugen, die im Runs-Test nicht auffällt. Je länger die Sequenz, desto mehr Macht hat der Runs-Test, d.h., desto eher kann er Abweichungen vom echten Zufall entdecken. Kann jemand eine Sequenz der Länge 1000 eintippen, die nicht auffällig ist?

Echte Test-Prozeduren für Pseudo-Zufallszahlengenertoren enthalten mehrere solche Tests und sind daher noch viel schwieriger zu schlagen.