Non-Uniform Random Number Generation

Tobias Flörchinger

Humboldt-Universität zu Berlin

Motivation and Study Design

For generating streams of uniform pseudo-random numbers, random number generators (RNGs) are used. $V, U \sim Unif(0,1)$ are basis to numerous methods for generating random variates (not uniformly distributed random numbers). Random variate generators (RVGs) are relevant e.g. in data generation for simulation studies or Bayesian Inference for non-standard posterior distributions. $\Rightarrow$ Hence RVGs are widely used but seldom in the spotlight.

Therefore, 5 methods with candidate RVGs are presented and compared regarding efficiency, universality and quality:

- Proposed RVGs are compared to the R implementations
- If applicable, RVGs sample from the target distributions N(0,1), Exp(1), Ga(2,1), Bin(40,$\frac{1}{2}$), Pois(10) and Tri(-1,1,0).
- Efficiency is evaluated by comparing run-times of RVGs
- and Quality by goodness-of-fit tests (χ^2, Kolmogorov-Smirgov (KS) and Anderson-Darling (AD))[3] as well as the Wald-Wolfowitz Runs Test[5] for correlation in realizations.

Note: RVGs need i.i.d U(0,1) random numbers from RNGs. $\Rightarrow$ The simulation relies on the Mersenne-Twister (standard in R) to yield sufficiently good pseudo-random numbers.

Alias Method

Idea: Randomly select discrete events with prespecified probabilities (q_K) with rejection sampling. In case of rejection, replace the rejected event with an alias a_K.[6]

Theoretical Background: *"An arbitrary n-point distribution can be represented as an equally weighted [at 1/n] mixture of n two-point distributions."* (Kroese, Taimre, Botev, 2011, p. 56)

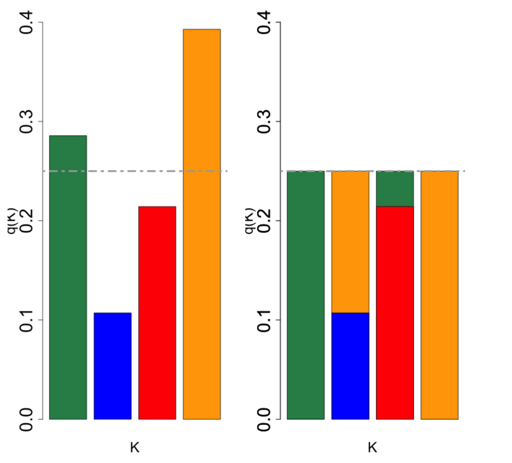

Fig. 1: Redistribution of probability mass to create an equally probable mixture of two-point distributions from a discrete distribution.

Pseudo-Code:

1. Compute two tables containing a_K, q_K (for all two-point distributions)
2. Generate $U \sim Unif(0,1)$ and $V \sim Unif(0, \frac{1}{n})$
3. Set $K = \lceil n \times U \rceil$
4. If $V \leq q_K$: return value of K
5. Else: return a_K

Conditions: Two tables needed, discrete distributions.
Strengths: Each run leads to a generated number in constant time; very fast for a lot of numbers.

Numerical Inversion (NINV)

Idea: Use the inverse cumulative density function (cdf) to generate random variates.

Theoretical Background: Approximate the inverse cdf as computing F^{-1} is often difficult or slow for most standard distributions. The approximate $G^{-1}(U) = X$ can be determined using numerical root finding (Bisection, Regular falsi, Newton-Raphson). The error $e = |F(X) - U|$ is minimized up to a tolerance level.[7]

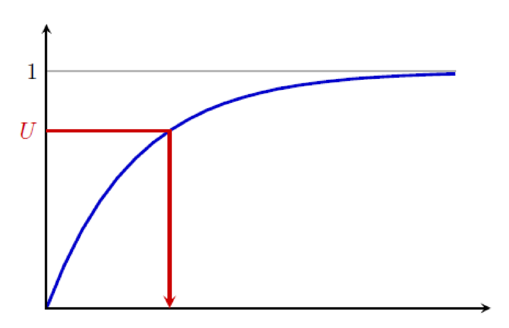

Fig. 2: Cdf of exponential distribution with $\lambda = 1$. (Leydold, Hörmann, Sak 2012)

Pseudo-Code:

1. Draw $U \sim Unif(0,1)$
2. Shift the cdf with $F(x) - U$ $\Rightarrow$ the root provides X
3. Generate $X = G^{-1}(U)$ with numerical root finding
   3.1 Choose starting values
   3.2 Iterate until e exceeds the tolerance level
4. return X

Condition: Monotonically increasing cdf.
Strengths: Works everywhere with controlled accuracy.

Composition Method (Comp)

Idea: Split the area under the probability distribution function (pdf), choose area randomly by proportion. Use inversion to generate a random variate from this area.[1]

Theoretical Background: The pdf can be written as

$$f(x) = \sum_{i=1}^{n} f_i(x) p_i.$$

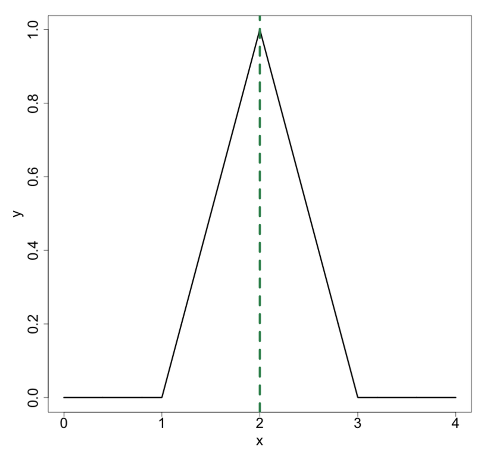

Fig. 3: The triangular distribution density is split into two areas, both have the probability of p=0.5.

Pseudo-Code:

Let $F_i = \sum p_i$

1. Draw $U \sim Unif(0,1)$
2. Find f_J by summing as long as $0 < U - \sum_0^{J-1} p_i = U_J$
3. Use integration and inversion to get $X = F_J^{-1}(U_J)$
4. return X

Condition: Pdf can be written as sum.
Strengths: Mixtures of distributions and compound distributions.

Transformed Density Rejection (TDR)

Idea: Partition tranformed pdf into intervals to construct the hat locally then use composition and acceptance-rejection.

Theoretical Background: For all continous unimodal pdfs exists a strictly monotonically increasing transformation T that ensures concavity (e.g. $ln(\cdot)$). Approximate concave density functions both from above $\tilde{h}_i$ (hat) and from below $\tilde{s}_i$ (squeeze) using tangents and secants.[3]

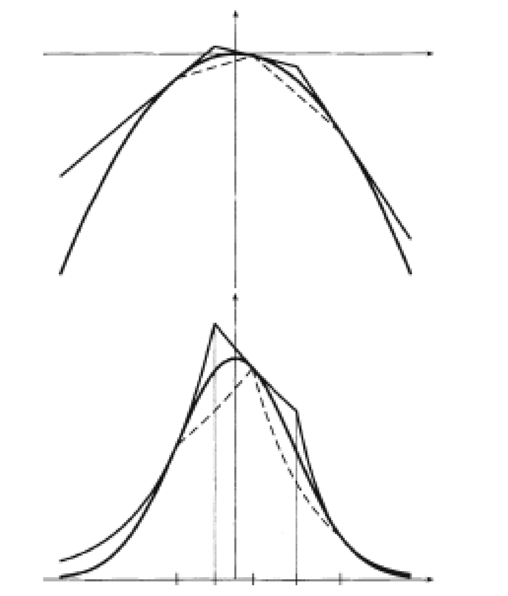

Fig. 4: Approximation in $ln(f(x))$ (upper graph) and with $exp(\cdot)$ in $f(x)$ (lower graph) for N(0,1). (Hörman, Leydold, Derflinger, 2004)

Pseudo-Code:

1. $T(f(x)) \Rightarrow$ with $q_i \in (b_l, b_r)$ construct $\tilde{h}(x)$ and $\tilde{s}(x)$ $\Rightarrow$ in preimage with $T^{-1}(\cdot)$
2. Draw from $V \sim Unif(0, A)$ ($\int h(x)dx = \sum_{i=1}^{N} A_i = A$; $A_i \mathrel{\hat{=}}$ area of interval i)
3. Find interval J and h_J with $\min\{J : A_1 + \dots + A_J \geq V\}$
4. Recycle $V_J \sim Unif(0, A_J)$ to generate $X = H_J^{-1}(V_J)$
5. Accept or reject X with $U \sim Unif(0,1)$ by evaluating squeeze and density

Conditions: Bounded, unimodal, differentiable pdf.
Strengths: T-concave and truncated distributions.

Ratio-of-Uniform Method (RoU)

Idea: Generate the random variable as the ratio of two uniformly distributed random numbers, which are sampled using the rejection method.[2]

Theoretical Background: *"To generate a random variable X with density $f(x)$ let $C_f = \{(u,v) : 0 \leq u \leq \sqrt{f(\frac{v}{u})}\}$, then if (U,V) is a pair of random variables distributed uniformly over C_f, $X = \frac{V}{U}$ has the desired density f."* (Kinderman, Monahan, 1977, p. 258)

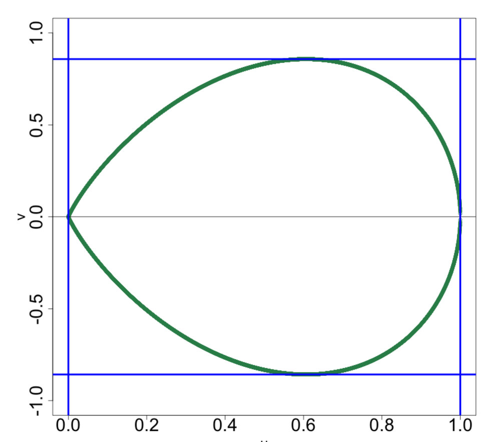

Fig. 5: Enclosing rectangle (blue) and boundaries of C_f (green) for N(0,1).

Pseudo-Code:

1. Define enclosing rectangle (b, a_-, a_+) for $\sqrt{f(x)}$
2. Generate $U \sim Unif(0, b)$ and $V \sim Unif(a_-, a_+)$
3. Define X as $\frac{V}{U}$
4. If $U^2 \leq f(X)$ return X

Conditions: Density has to be bounded.
Strengths: Bell-shaped densities and densities with an infinite but at least subquadratic tail.

Efficiency and Universality

Due to uninformative differences between distributions, for efficiency NINV, TDR, RoU for $N(0,1)$, Alias for $Bin(40, \frac{1}{2})$, NINV, Composition for $Tri(-1,1,0)$ and the R RVGs are shown.

Efficiency

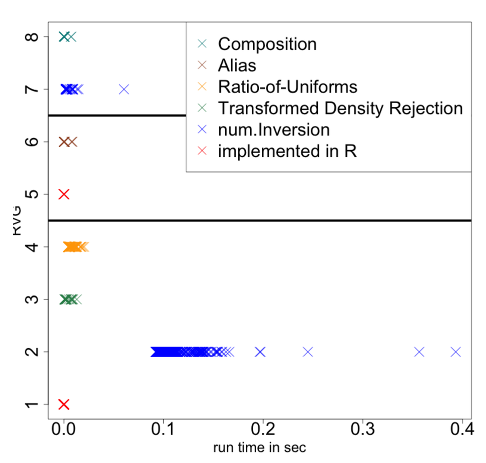

Fig. 6: Run time of RVGs with n=10,000.

Except for numerical Inversion, there are few differences between the RVGs in runtime. However comparing $n = 1,000$ with $n = 10,000$ shows that increasing the sample size is less costly for rejection methods than for the others, so TDR and RoU eventually outperform R as n increases.

Universality

Universality corresponds to the size of the family of distributions that can be sampled from. It is defined by the "Conditions" part of the RVGs. In R implemented RVGs are distribution specific approximations.

RVG	continous	discrete
NINV	+ + +	(+ + +)
TDR	+ +	-
RoU	+ +	-
Comp	+	+
Alias	-	+ + +

Tab. 1 :Ratings of universality of different RVGs. For specifics: see RVGs "Conditions".

$\Rightarrow$ For $n = 10,000$ the RVGs in R are the most efficient ones, but lack universality. Efficiency can often be improved and universality is attained by generalization.

Quality

The tests yield minor differences between distributions, hence the selection as for efficiency is used.

Runs-Test

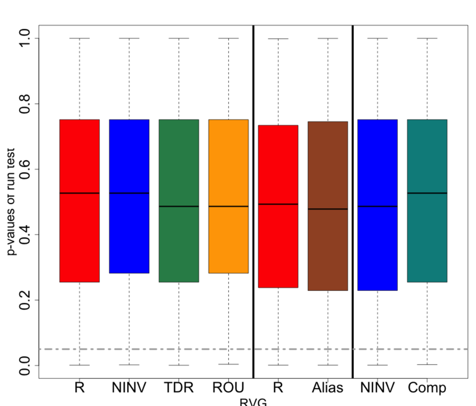

Fig. 7: p-values for RVGs with n=10,000 and T=1,000.

H_0: Random realization of runs around the median.

For all RVGs H_0 cannot be rejected on average, hence no correlation could be found. There is no systematic difference between RVGs regrading the p-values of the Runs-Test.

Goodness-of-Fit Tests and Non-Rejection Rates

Test	R	NINV	TDR	RoU	R	Alias	NINV	Comp
χ^2	0.50	0.52	0.50	0.49	0.49	0.48	0.49	0.50
KS	0.50	0.52	0.50	0.49	-	-	0.50	0.49
χ^2	95.2	95.4	94.0	95.6	94.4	94.0	95.1	94.7
KS	95.1	95.7	94.9	94.7	-	-	94.8	93.7

Tab. 2: EPV (upper columns) and non-rejection rates in % (lower columns) for $\alpha = 5\%$ using n=10,000 and T=1,000 iterations.

H_0: The variates edf corresponds to the distribution cdf.

On average no goodness-of-fit test rejects H_0. There is no clear difference between the non-rejection rates of RVGs visible (AD results are consistent to those of KS and χ^2).

$\Rightarrow$ The underlying RNG is the main driver for the quality.

Conclusion

- **Results**: RVGs differ in terms of efficiency and universality, whereas their generated variates are of consistent quality. R RVGs are sufficient for their corresponding distributions. However sampling from non-standard distributions requires other RVGs. Here universality ensures applicability.
- **Implications**: Depending on the target distribution and the purposes of the simulation a suiting RVG has to be chosen by considering the candidate RVGs strengths, conditions and efficiency. One has to assure that a state-of-the-art RNG is underlying to ensure the quality of variates.
- **Outlook**: To assess quality further an analysis of RVGs with varying RNGs is conceivable. Efficiency can be refined by setup and sampling times. Proposed RVGs are usable for special purposes like variance reduction (importance sampling) and sampling from truncated distributions.

References

- [1] Cheng, R.C.H. (1998). Random Variate Generation. In J. Banks (Hsrg.) Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice , pp. 139-172.
- [2] Devroye, L. (1986). Non-Uniform Random Variate Generation. New York, Springer Science and Business Media.
- [3] Hörmann, W., Leydold, J. Derflinger, G. (2004). Automatic Nonuniform Random Variate Generation. Heidelberg, Springer.
- [4] Kinderman, A.J. Monahan, J.F (1977). Computer Generation of Random Variables Using the Ratio of Uniform Deviates. ACM Transaction on Mathematical Software, 3(3), pp.257-260.
- [5] Kroese, D.P., Taimre, T. Botev, Z.I. (2011). Handbook of Monte Carlo Methods. New York, Wiley Series in Probability and Statistics, John Wiley Sons.
- [6] Kronmal, R.A. Petersin Jr, A.V. (1979). On the Alias Method for Generating Random Variables from a Discrete Distribution. The American Statistician, 33(4), pp. 214-218.
- [7] Leydold, J., Hörmann, W., Sak, H. (2012). An R Interface to the UNU. RAN Library for Universal Random Variate Generators.

For Additional Information

Universität Trier

Authors:
Tobias Flörchinger (floercht@hu-berlin.de)