Monte Carlo

Ignacio Cascos

2019

Outline

Introduction

- *Simulation* **Monte Carlo** techniques: computation algorithms that rely on random sampling.
- *Resampling* **Bootstraping**: random sampling with replacement from an original sample.

1.1 Probability and inference refresher

Consider a random variable X and a random sample from it $X_1, X_2, \ldots, X_n$, we define:

- The *sample mean* as

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

- The *sample variance* as

$$S_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X}_n)^2$$

- The *sample proportion* as the fraction of sample observations with a given characteristic,

$$\hat{p}_n = \frac{\#\{X_i \text{ with the characteristic}\}}{n}.$$

Laws of Large Numbers

If $\{X_n\}_n$ is a sequence of *independent and identically distributed* random variables with $\mathbb{E}[X_i] = \mu$, then

- **WLLN**: for any $\varepsilon > 0$, $\lim_{n\to\infty} P(|\overline{X}_n - \mu| \geq \varepsilon) = 0$;
- **SLLN**: $P\left(\lim_{n\to\infty} \overline{X}_n = \mu\right) = 1$,

where $\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$.

Observe that if $X_i \sim \mathrm{B}(1, p)$, then $\overline{X}_n = \hat{p}_n$ and $\mu = p$.

Central Limit Theorem

- **Lyapunov** If $\{X_n\}_n$ is a sequence of *iid* r.v.s with mean μ and variance $\sigma^2 < \infty$, then

$$\frac{\sum_{i=1}^{n} X_i - n\mu}{\sigma\sqrt{n}} = \frac{\overline{X}_n - \mu}{\sigma/\sqrt{n}} \xrightarrow{d} Z\,,$$

  where $Z \sim \mathrm{N}(0,1)$. If needed, we can substitute σ by is estimate S_n.
- **DeMoivre-Laplace** If $X_n \sim \mathrm{B}(n,p)$, then

$$\frac{X_n - np}{\sqrt{np(1-p)}} \xrightarrow{d} Z\,,$$

  where, we can divide numerator and denominator by n to obatin

$$\frac{\hat{p}_n - p}{\sqrt{\frac{p(1-p)}{n}}} \xrightarrow{d} Z\,,$$

  and here we can substitute the p's in the denominator by $\hat{p}_n$ if needed.

Statistical analysis of simulated data (Conf. Intervals)

- Approximate $(1-\alpha)100\%$ CI on μ (the mean)

$$\left[\overline{x}_n - z_{\alpha/2}\frac{s_n}{\sqrt{n}}\,,\ \overline{x}_n + z_{\alpha/2}\frac{s_n}{\sqrt{n}}\right]$$

- Approximate $(1-\alpha)100\%$ CI on p (a proportion)

$$\left[\hat{p}_n - z_{\alpha/2}\sqrt{\frac{\hat{p}_n(1-\hat{p}_n)}{n}}\,,\ \hat{p}_n + z_{\alpha/2}\sqrt{\frac{\hat{p}_n(1-\hat{p}_n)}{n}}\right]$$

$z_{\alpha/2} = \texttt{qnorm}(1-\alpha/2)$

1.2 Statistical validation techniques (GoF tests)

- **Kolmogorov-Smirnoff Goodness-of-Fit test** (continuous data, given F_0)

Test. $H_0 : F = F_0$ vs. $H_1 : F \neq F_0$

Test statistic. $D = \max_x |\hat{F}_n(x) - F_0(x)|$, with $\hat{F}_n$ empirical cdf.

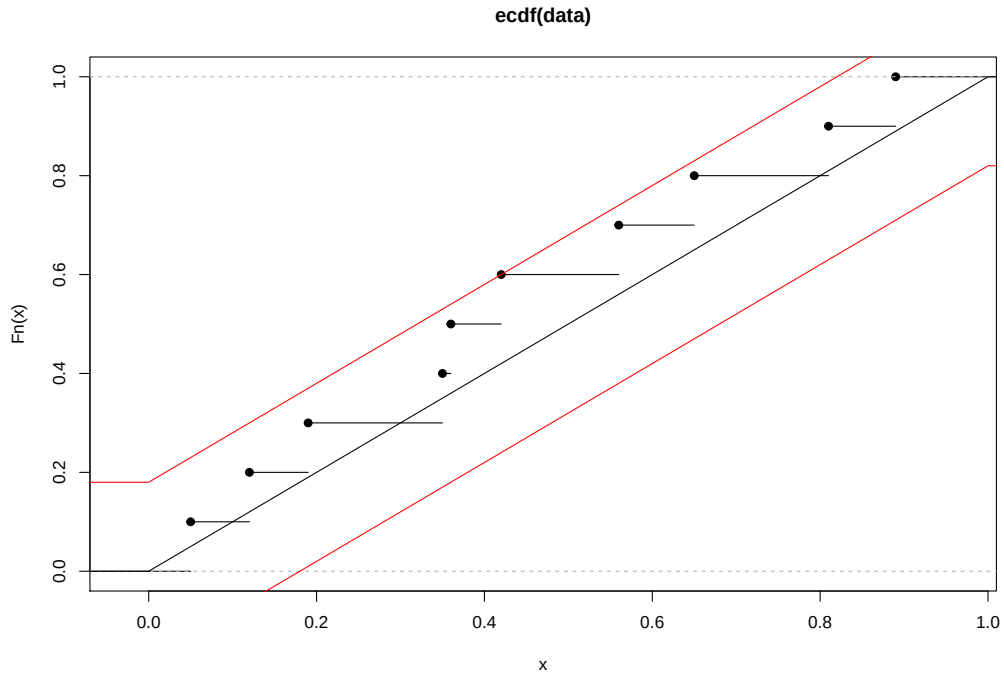
The distribution of D (under H_0) does NOT depend on F_0 (continuous).

```
data <- c(.56,.35,.89,.81,.36,.19,.65,.42,.12,.05)
ks.test(data,"punif")
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  data
## D = 0.18, p-value = 0.8473
## alternative hypothesis: two-sided
```

Statistical validation techniques (GoF tests)

```
plot(ecdf(data))
aux <- seq(-.1,1.1,0.01)
points(aux,punif(aux),type="l")
D <- ks.test(data,"punif")$statistic
points(aux,punif(aux)+D,type="l",col="red")
points(aux,punif(aux)-D,type="l",col="red")
```



Statistical validation techniques (GoF tests)

- **Chi-Square Goodness-of-Fit test** (discrete data, given F_0)

Test. $H_0 : F = F_0$ vs. $H_1 : F \neq F_0$

Test statistic. $X^2 = \sum_{i=1}^{k} \frac{(N_i - np_i)^2}{np_i}$,

k levels of the variable, N_i observed counts, and np_i expected counts.

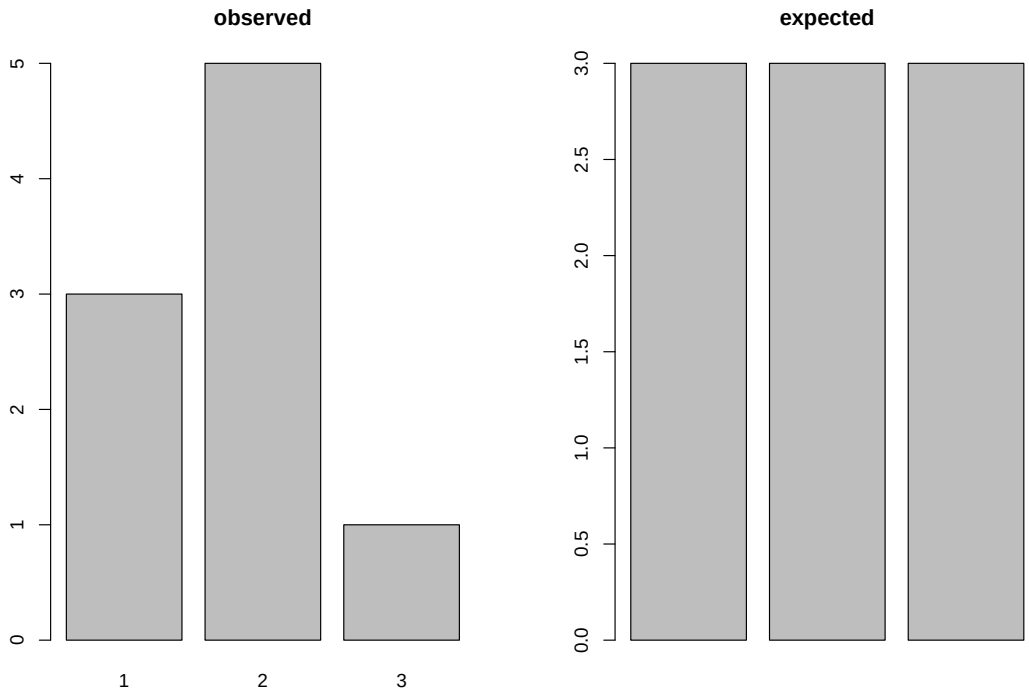
Under H_0 $X^2 \sim \chi^2_{k-1}$.

```
observed <- table(c(3,2,1,2,2,1,2,1,2))
probs <- rep(1/3,3)
chisq.test(x=observed,p=probs)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  observed
## X-squared = 2.6667, df = 2, p-value = 0.2636
```

Statistical validation techniques (GoF tests)

```
n <- sum(observed)
par(mfrow=c(1,2))
barplot(observed,main="observed")
barplot(probs*n,main="expected")
```



Statistical validation (GoF tests, unspecified parameters)

- For specific distributions (e.g. normal), we can use specific tests (e.g. Shapiro-Wilk normality test) or graphical tools (e.g. qq-plots).
- When some parameters are unespecified:
  - Continuous distribution, use KS test and simulate to estimate critical value or p-value.
  - Discrete distribution, use Chi-square test and substract the number of estimated parameters from the degrees of freedom.

Randomness tests

- **Wald-Wolfowitz runs test** of randomness for continuous data

Test. H_0 : i.i.d. observations vs. H_1 : not i.i.d. observations

Test statistic. U: # runs (group of consecutive values either above or below threshold – median), N_1 (resp. N_2): # values above (below) threshold

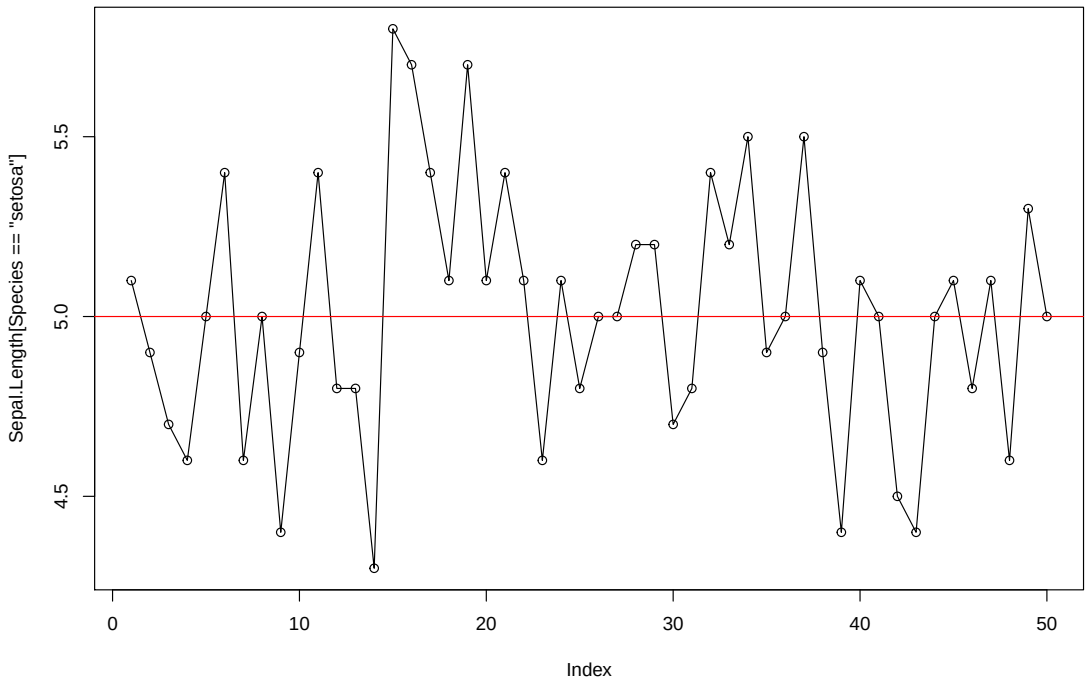
$$\frac{U - \left(\frac{2N_1N_2}{N_1+N_2} + 1\right)}{\sqrt{\frac{2N_1N_2(2N_1N_2-N_1-N_2)}{(N_1+N_2)^2(N_1+N_2-1)}}}$$

```
# install.packages("randtests")
require(randtests) ; data("iris")
runs.test(iris$Sepal.Length[iris$Species=="setosa"])$p.value
```

```
## [1] 0.7428503
```

Randomness tests

```
attach(iris)
plot(Sepal.Length[Species=="setosa"],type="l")
points(Sepal.Length[Species=="setosa"])
abline(h=median(Sepal.Length[Species=="setosa"]),col="red")
```



1.3 Random numbers

Prima materia

The first matter in simulation are random numbers in the $(0, 1)$ interval (indepent random variables with a $\mathrm{U}(0, 1)$ distribution).

Is it possible to build such random numbers?

If fair coins do exist, it is.

Consider a sequence of tosses of a fair coin $x_1, x_2, x_3, \ldots$, where

$$x_i = \begin{cases} 1 & \text{if Head on } i\text{-th toss} \\ 0 & \text{if Tail on } i\text{-th toss} \end{cases}$$

We can interpret the sequence as the binary expansion of a number in the $[0, 1]$ interval,

$$\sum_{i=1}^{\infty} \frac{x_i}{2^i} \, .$$

Prima materia

```
MC <- 200
simul.bin <- vector(length=MC)
set.seed(1)
for(i in 1:MC){
  samp <- sample(c(0,1),replace=T,size=10)
  simul.bin[i] <- sum(samp/2^((1:10)*samp))
}
ks.test(simul.bin,punif)$p.value
```

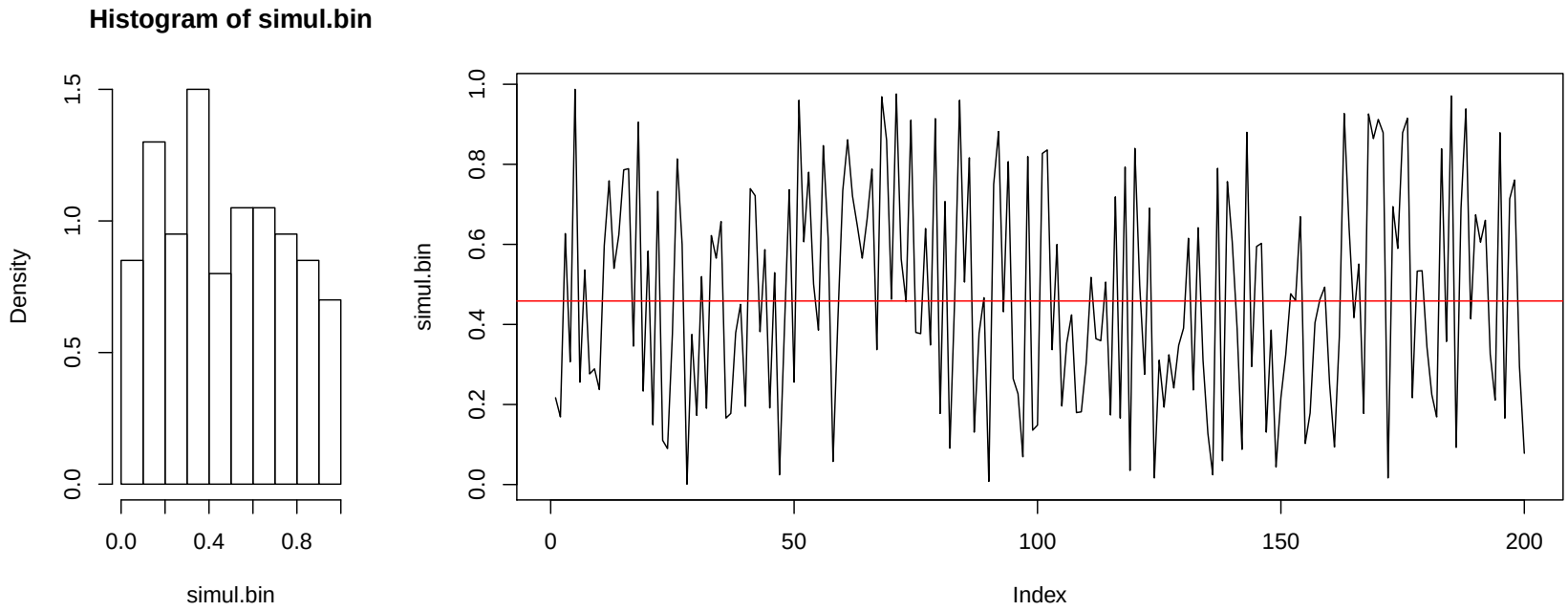
```
## [1] 0.2908259
```

```
runs.test(simul.bin)$p.value
```

```
## [1] 0.2567005
```

Prima materia

```
layout(matrix(c(1,2),1,2,byrow=T),widths=c(1,3))
hist(simul.bin,probability=T)
plot(simul.bin,type="l")
abline(h=median(simul.bin),col="red")
```



Pseudorandom numbers

The numbers we work with have been determiniscally generated, so they are not truely random.

A *multiplicative congruential generator* works as follows:

1. Set m (prime) large.
2. Set $a < m$.
3. Set seed x_0 (initial value).
4. Set $x_n \equiv ax_{n-1}\ (m)$, which means x_n is the remainder of ax_{n-1}/m.
5. For every n return x_n/m.

Pseudorandom numbers

```
MC <-1000 ; seed <- 1 ; m <- 2^35-31 ; a <- 5^5
simul <- vector(length=MC)
simul[1] <- (a*seed)%%m
for(i in 2:MC) simul[i] <- (a*simul[i-1])%%m
simul <- simul/m
ks.test(simul,"punif")$p.value
```

```
## [1] 0.9174393
```

```
runs.test(simul)$p.value
```

```
## [1] 0.3113298
```

Replicability and efficiency vs. true randomness

The methods we use do not produce truely random numbers, but they have two properties that are essential in Statistics:

- We can **duplicate** our results. By fixing a seed (initial value) we always obtain the same sequence of (pseudo)random numbers.
- They are **efficient**. Fast and simple to implement.

Replicability and efficiency vs. true randomness

```
MC<-1000000 ; ptm <- proc.time() ; set.seed(1)
simul.unif <- runif(MC)
proc.time()-ptm
```

```
##    user  system elapsed
##    0.03    0.00    0.03
```

```
ptm <- proc.time()
seed <- 1 ; m <- 2^35-31 ; a <- 5^5
simul <- vector(length=MC)
simul[1] <- (a*seed)%%m
for(i in 2:MC) simul[i] <- (a*simul[i-1])%%m
simul <- simul/m ; proc.time()-ptm
```

```
##    user  system elapsed
##    0.27    0.01    0.28
```

1.4 Approximation of probabilities and volumes

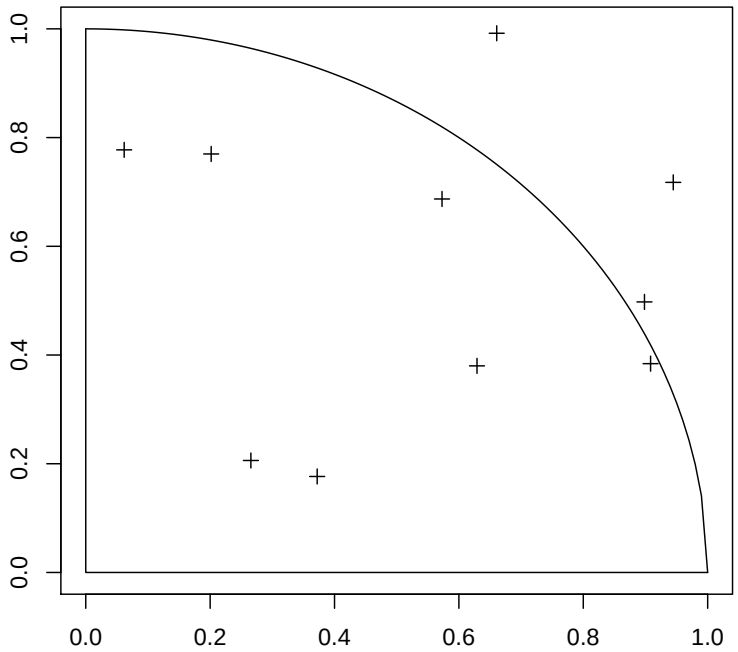
We can approximate the volue of some given compact set $C \subset \mathbb{R}^d$ by considering a d-dimensional box B containing it, $C \subset B$. We can simulate points from the box at random, the volume of C is approximately the fraction of points in C times the volume of B.

Consider $B = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d]$

1. Set $inside = 0$.
2. Repeat Steps 3 to 4 n times.
3. Generate U_1 random number in $[a_1, b_1]$, U_2 random number in $[a_2, b_2]$, $\ldots$, U_d random number in $[a_d, b_d]$.
4. If $(U_1, \ldots, U_d) \in C$, then $inside = inside + 1$
5. Set $vol = (inside/n) * (b_1 - a_1) * \cdots * (b_d - a_d)$ and stop.

Approximate the area of a circle of radius 1

```
aux <- seq(0,1,.01)
plot(aux,sqrt(1-aux^2),type="l",xlab="",ylab="")
lines(c(1,0,0),c(0,0,1))
MC <- 10 ; set.seed(1)
points(runif(MC),runif(MC),pch=3)
```



Approximate the area of a circle of radius 1

```
n <- 1000 ; set.seed(1)
x <- runif(n)
y <- runif(n)
sum(x^2+y^2<=1)/n*4
```

```
## [1] 3.148
```

```
prop.test(sum(x^2+y^2<=1),n=n,conf.level=0.95)$conf.int*4
```

```
## [1] 3.040123 3.246918
## attr(,"conf.level")
## [1] 0.95
```

Approximation of probabilities by simulation

For each random experiment, we can simulate a random number in the interval $(0, 1)$ and assume that an event with assigned probability p occurs if it is less than p.

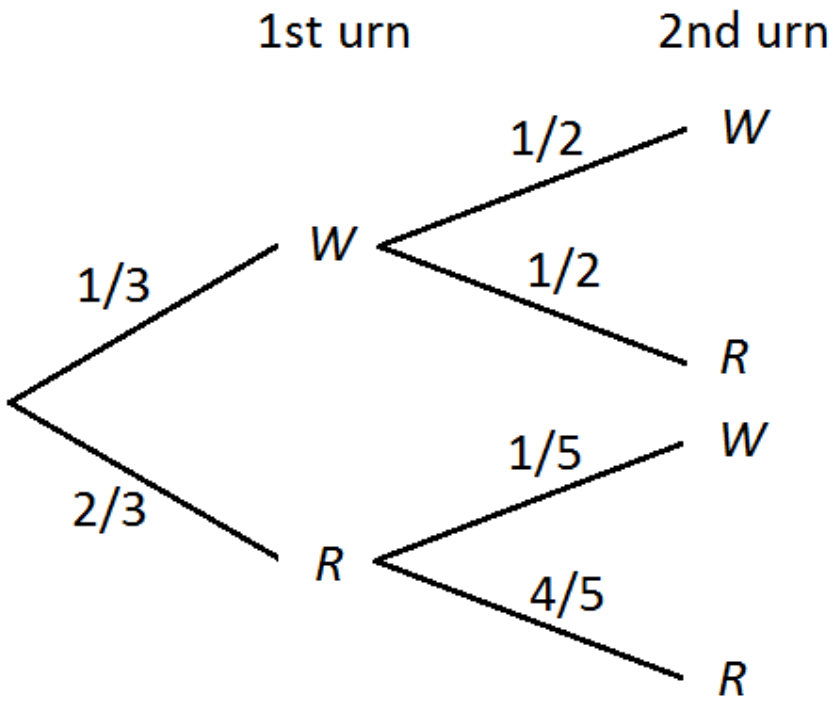
The expression `(runif(1)<p)` is a logical value (TRUE or FALSE) that is interpreted as numerical value 1 (TRUE) or 0 (FALSE) when operating.

The intesection of two events occurs whenever both of them do simulataneously ocurr, logical operator 'AND', product as numbers.

The union occurs whenever at least one of them occurs, logical operator 'OR', maximum as numbers.

Approximation of probabilities by simulation (example)

An urn contains 1 white ball and 2 red balls. One ball is taken from the urn at random. If it is a white ball, it is returned into the urn together with another white ball. If it is a red ball, it is returned together with other 2 red balls. In a second stage, another ball is taken from the urn at random.



Approximation of probabilities by simulation (example)

An urn contains 1 white ball and 2 red balls. One ball is taken from the urn at random. If it is a white ball, it is returned into the urn together with another white ball. If it is a red ball, it is returned together with other 2 red balls. In a second stage, another ball is taken from the urn at random.

Events

- $W_1 \equiv$'first ball is white'
- $R_1 \equiv$'first ball is red'
- $W_2 \equiv$'second ball is white'
- $R_2 \equiv$'second ball is red'

Probabilities

- $P(W_1) = 1/3;\ P(R_1) = 2/3$
- $P(W_2|W_1) = 1/2;\ P(R_2|W_1) = 1/2$
- $P(W_2|R_1) = 1/5;\ P(R_2|R_1) = 4/5$

Approximation of probabilities by simulation (example)

- **Total probability rule** What is the probability that the second ball extracted from the urn is red?

$$P(R_2) = P(W_1)P(R_2|W_1) + P(R_1)P(R_2|R_1) = \frac{1}{3}\cdot\frac{1}{2} + \frac{2}{3}\cdot\frac{4}{5} = 0.7\,.$$

Bayes' rule

If the second ball extracted from the urn is red, what is the probability that the first one was alse red?

$$P(R_1|R_2) = \frac{P(R_1)P(R_2|R_1)}{P(R_2)} = \frac{\frac{2}{3}\cdot\frac{4}{5}}{0.7} = 0.762\,.$$

Approximation of probabilities by simulation (example)

```
MC <- 1000 ; set.seed(1)
ball.1 <- runif(MC)
ball.2 <- runif(MC)
R1 <- (ball.1<2/3)
R2 <- R1*(ball.2<4/5)+(1-R1)*(ball.2<1/2)
p1 <- sum(R2)/MC ; p1
```

```
## [1] 0.703
```

```
p2 <- sum(R1*R2)/sum(R2) ; p2
```

```
## [1] 0.7496444
```

95% CIs on the probabilities

```
p1-sqrt(p1*(1-p1)/MC)*qnorm(.975)
```

```
## [1] 0.6746793
```

```
p1+sqrt(p1*(1-p1)/MC)*qnorm(.975)
```

```
## [1] 0.7313207
```

```
n2 <- sum(R2)
p2-sqrt(p2*(1-p2)/n2)*qnorm(.975)
```

```
## [1] 0.7176203
```

```
p2+sqrt(p2*(1-p2)/n2)*qnorm(.975)
```

```
## [1] 0.7816685
```

Integrals over $(0, 1)$

For any function g, the integral $\int_0^1 g(x)\,\mathrm{d}x$ is the expectation of random variable $U \sim \mathrm{U}(0, 1)$

$$\int_0^1 g(x)\,\mathrm{d}x = \int_0^1 g(x) f_U(x)\,\mathrm{d}x = \mathbb{E}[g(U)]\,,$$

After LLN, the sample mean of $g(U_1), \ldots, g(U_n)$ can be used to approximate the mean of randon variable $g(U)$:

1. Generate $U_1, \ldots, U_n$ random numbers in $(0, 1)$.
2. Transform each U_i into $Y_i = g(U_i)$.
3. Return $n^{-1} \sum_{i=1}^n Y_i$ and stop.

Asymptotic confidence inteval of an integral

Approximate $(1-\alpha)100\%$ CI on μ (the mean)

$$\left[\overline{x}_n - z_{\alpha/2}\frac{s_n}{\sqrt{n}}\,,\, \overline{x}_n + z_{\alpha/2}\frac{s_n}{\sqrt{n}}\right]$$

with $z_{\alpha/2} = \texttt{qnorm}(1-\alpha/2)$.

Take each X_i as $Y_i = g(U_i)$, and use is average and sample variance.

1. Generate $U_1, \ldots, U_n$ random numbers in $(0, 1)$.
2. Transform each U_i into $Y_i = g(U_i)$.
3. Obtain sample mean and variance, $\overline{y}_n$ and s_n^2.
4. Return $\overline{y}_n \pm z_{\alpha/2} s_n/\sqrt{n}$ and stop.

Area of a quarter circle

$$\int_0^1 \sqrt{1-x^2}\,\mathrm{d}x$$

```
MC <-1000 ; set.seed(1) ; y <- sqrt(1-runif(MC)^2) ; mean(y)
```

```
## [1] 0.7845459
```

```
t.test(y)$conf.int
```

```
## [1] 0.7704274 0.7986644
## attr(,"conf.level")
## [1] 0.95
```

```
4*t.test(y)$conf.int
```

```
## [1] 3.081710 3.194658
## attr(,"conf.level")
```

Integrals over (a, b)

Transform to integral over $(0, 1)$ as

$$\int_a^b g(x)\,\mathrm{d}x = (b-a)\int_0^1 g\left(a+(b-a)t\right)\,\mathrm{d}t\,,$$

with the change of variable $t = (x-a)/(b-a)$.

If $U \sim \mathrm{U}(0,1)$, then $a+(b-a)U \sim \mathrm{U}(a,b)$, so alternatively to generating randdom numbers in $(0,1)$ and transforming, it is possible to generate random numbers in (a,b) as `runif(MC,min=a,max=b)`.

Integrals over (a, ∞)

Transform to integral over $(0, 1)$ as

$$\int_0^{+\infty} g(x)\,\mathrm{d}x = \int_0^1 t^{-2} g\left(t^{-1} - 1\right)\,\mathrm{d}t\,,$$

where the change of variable is $t = (1 + x)^{-1}$.

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\,\mathrm{d}x = 2\int_0^1 \frac{t^{-2}}{\sqrt{2\pi}} e^{-(t^{-1}-1)^2/2}\,\mathrm{d}t$$

```
MC <-10000 ; set.seed(1) ; u <- runif(MC)
y <- 2*exp(-(1/u-1)^2/2)/(u^2*sqrt(2*pi))
t.test(y)$conf.int
```

```
## [1] 0.9803571 1.0074944
## attr(,"conf.level")
## [1] 0.95
```

Multiple integrals

Consider now a function of two (or more) variables, $g : \mathbb{R}^2 \mapsto \mathbb{R}$, the integral $\int_0^1 \int_0^1 g(x_1, x_2)\,\mathrm{d}x_1\mathrm{d}x_2$ is thus the expectation of two independent random variables in the unit interval, $X_1, X_2 \sim \mathrm{U}(0,1)$

$$\int_0^1 \int_0^1 g(x_1, x_2)\,\mathrm{d}x_1\mathrm{d}x_1 = \int_0^1 \int_0^1 g(x_1, x_2) f_{X_1,X_2}(x_1, x_2)\,\mathrm{d}x = \mathbb{E}[g(X_1, X_2)]\,,$$

We can approximate the integral using the LLN over rv $g(X_1, X_2)$:

1. Generate $U_1, \ldots, U_n$ and $U'_1, \ldots, U'_n$ random numbers in $(0,1)$.
2. Transform each pair (U_i, U'_i) into $Y_i = g(U_i, U'_i)$.
3. Return $n^{-1} \sum_{i=1}^n Y_i$ and stop.

Multiple integrals

If the multiple integral is over a region different from the unit square (hypercube) we can transform it.

Alternatively, we can consider a wider region and discard those generated points that do not lie in it. In such a case, we should keep on simulating until we have a fixed number of points inside the region.

A new step should be introduced between Septs 1 and 2 of the previous algorithm.

- If some (U_i, U_i') does not lie inside the region, generate a new pair.