107.330 Statistische Simulation und computerintensive Methoden

Monte Carlo Methods

Alexandra Posekany

WS 2020

Monte Carlo Methods

Monte Carlo integration is just one computational tool from the family of Monte Carlo methods. In general Monte Carlo methods may refer to any method in statistical inference or numerical analysis where simulations are used.

Monte Carlo methods rely on simulation by repeated random sampling to obtain numerical results for various estimation problems, such as

- optimisation
- numerical integration
- sampling from complex probability distributions (Markov Chain Monte Carlo simulation)

Notion of Monte Carlo Methods

Originally, simulations tested a previously understood deterministic problem, and applied statistical sampling to estimate uncertainties in the simulations. (Confidence bands etc.)

Monte Carlo simulations invert this approach. They often solve deterministic problems by applying probabilistic simulations.

History of Monte Carlo Methods

In the 1940s mathematician Stanislaw Ulam was on sick leave and played Solitaire to pass his time. “How often do I set it up and lose?”

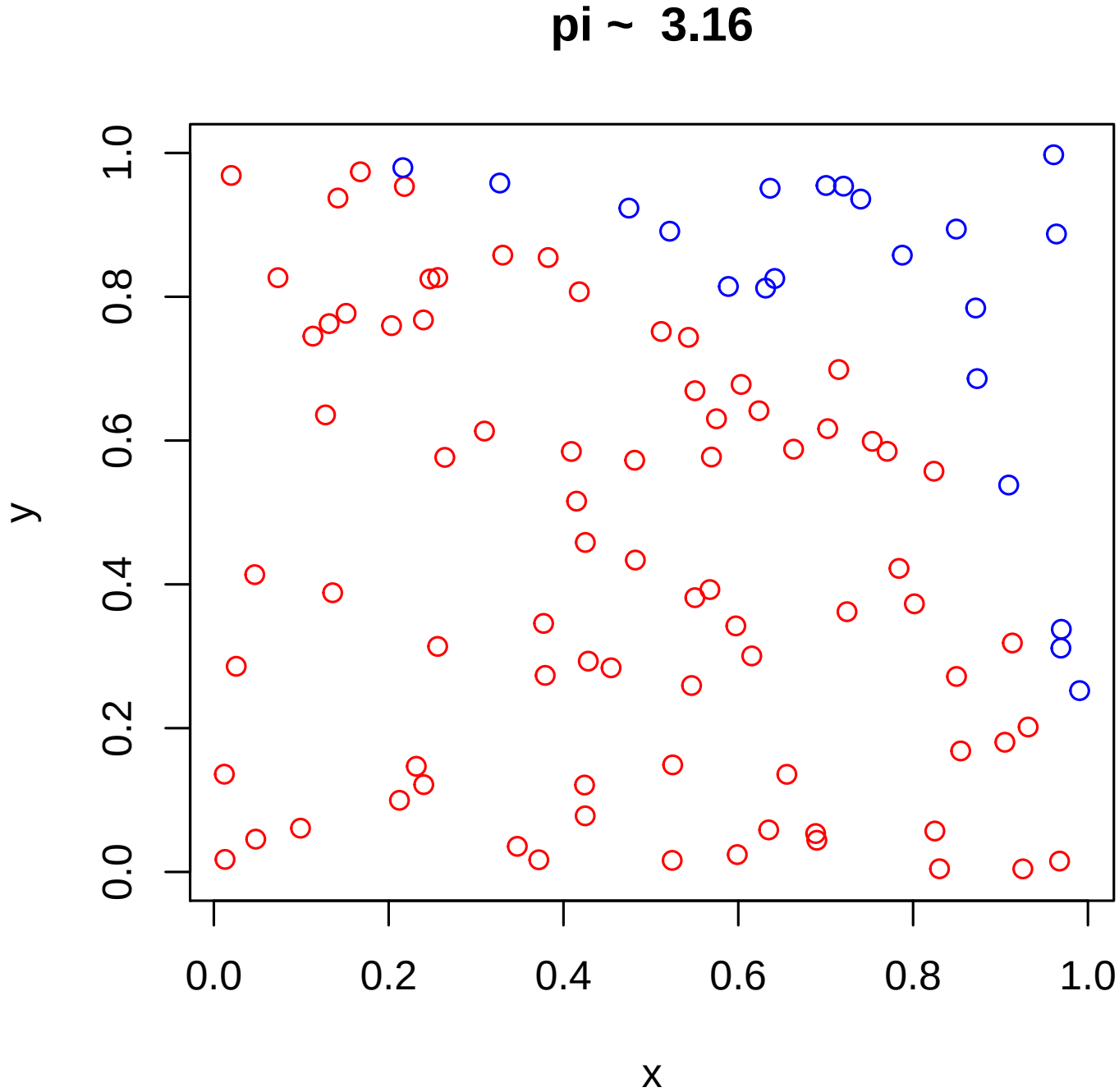
He wanted to calculate this probability, but the combinatorics of winning Solitaire game turned out to be too difficult for an analytical solution. So, he decided to try a physical simulation by playing enough games to estimate probability. This also turned out to be infeasible

Finally, he turned to his friend John von Neumann with whom he worked on the Manhattan project and discussed the idea to simulate it on their computer. This bore the idea which would be utilised further during the project.

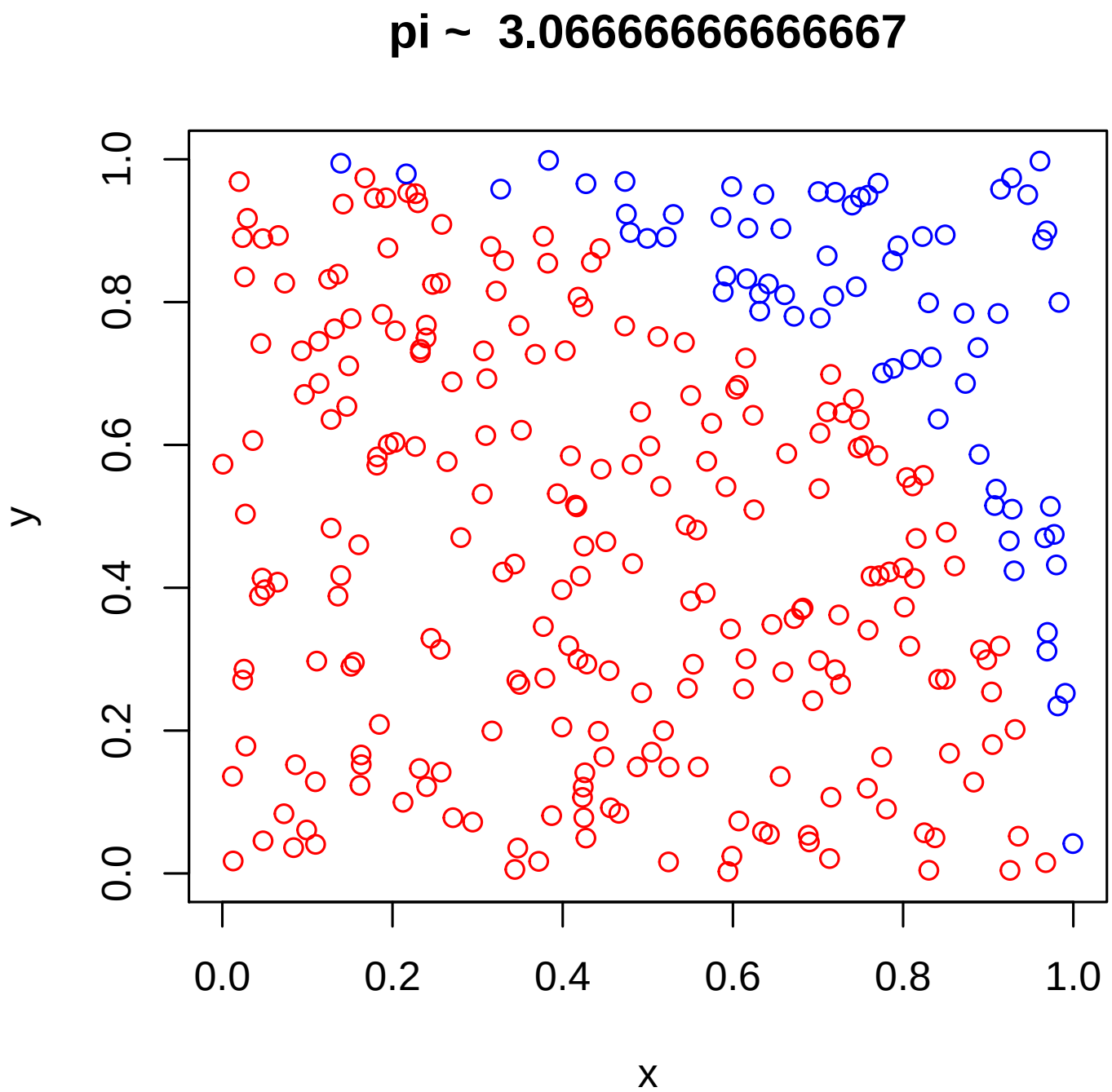
Backbone of Monte Carlo simulation

1. Define a domain of possible inputs for the problem
2. Generate inputs randomly from a probability distribution over the domain
3. Perform a deterministic computation on the inputs
4. Aggregate the results

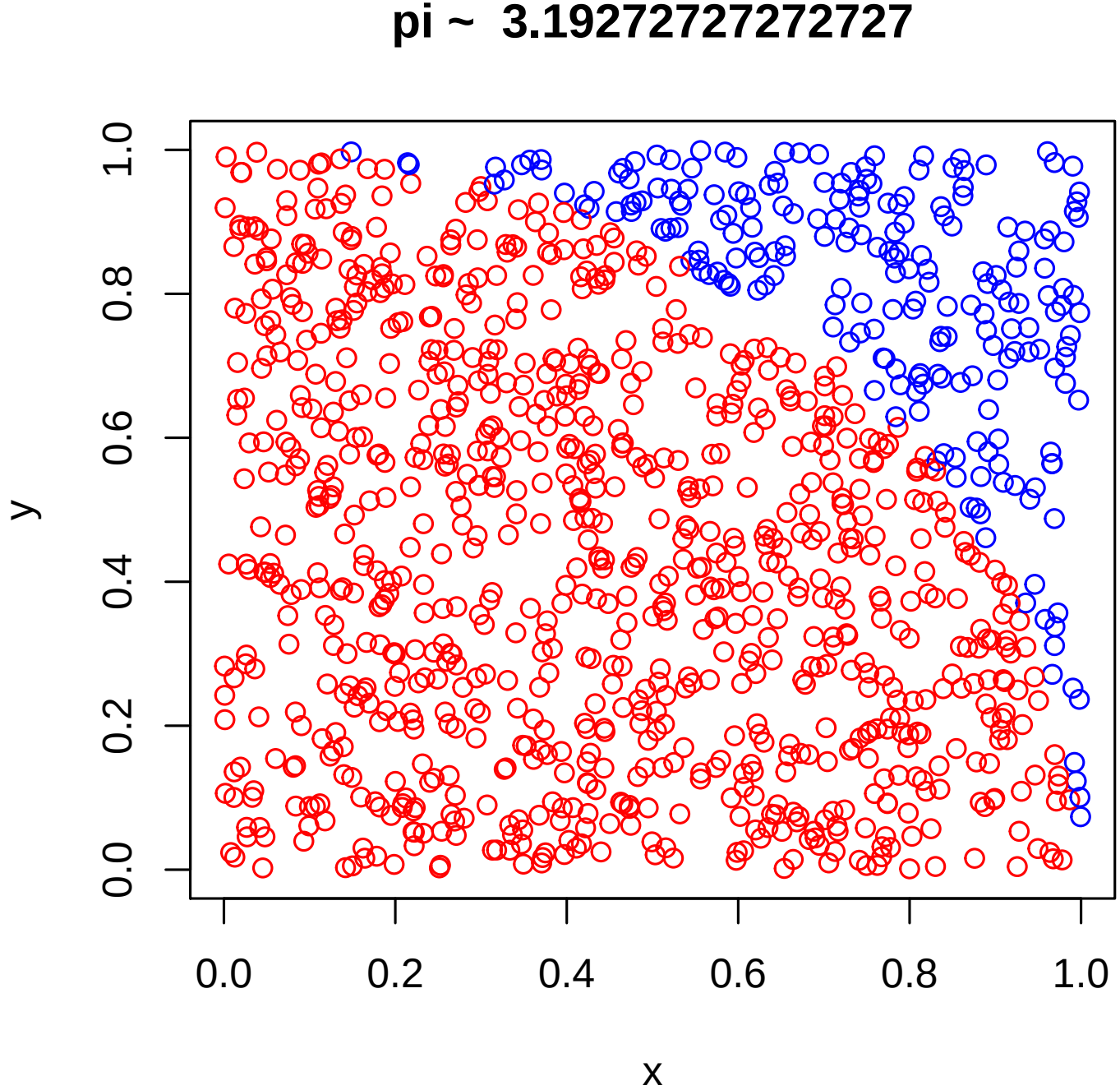
Monte Carlo visualised



Monte Carlo Simulation visualised

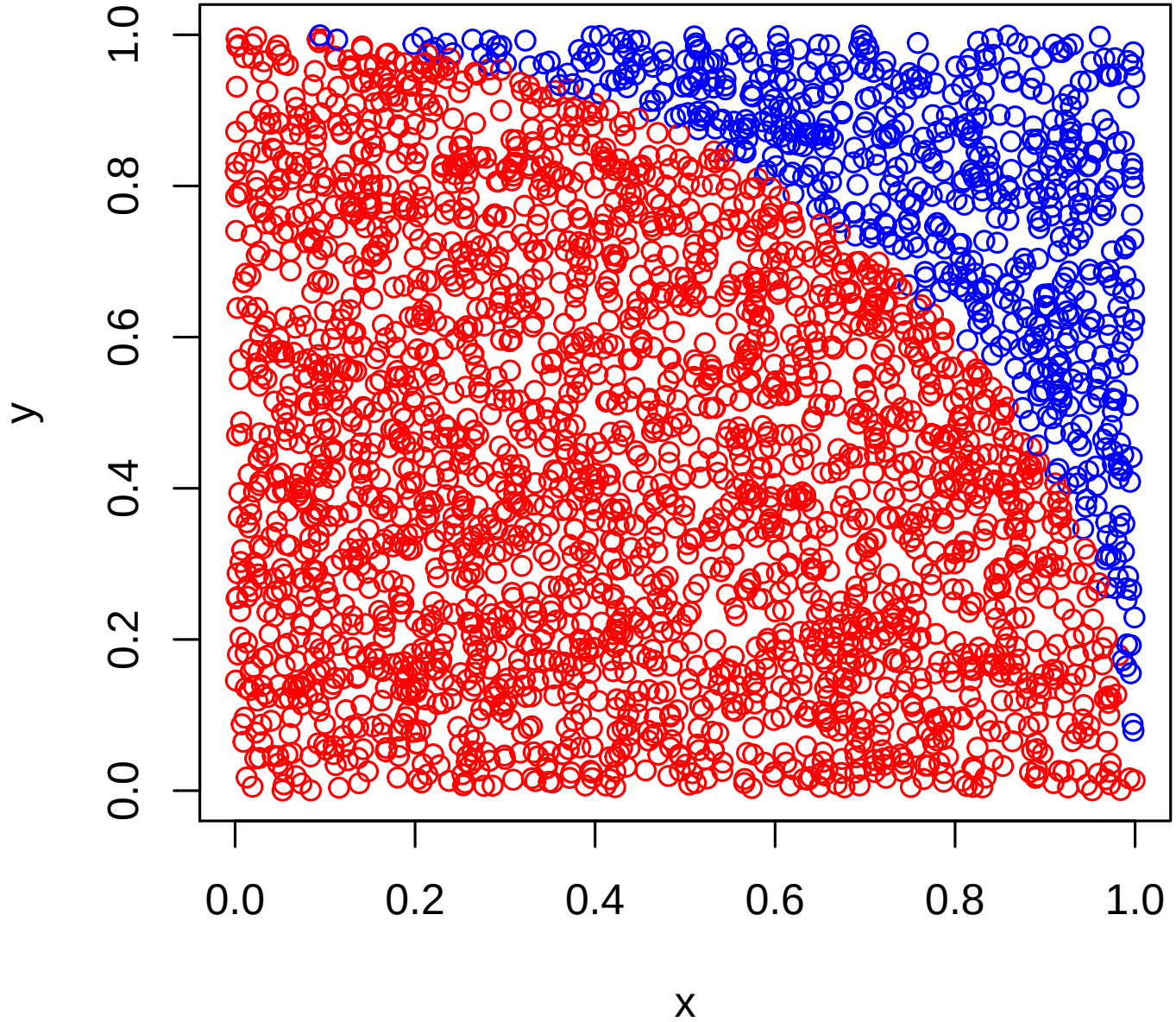


Monte Carlo Simulation visualised



Monte Carlo Simulation visualised

pi ~ 3.16774193548387



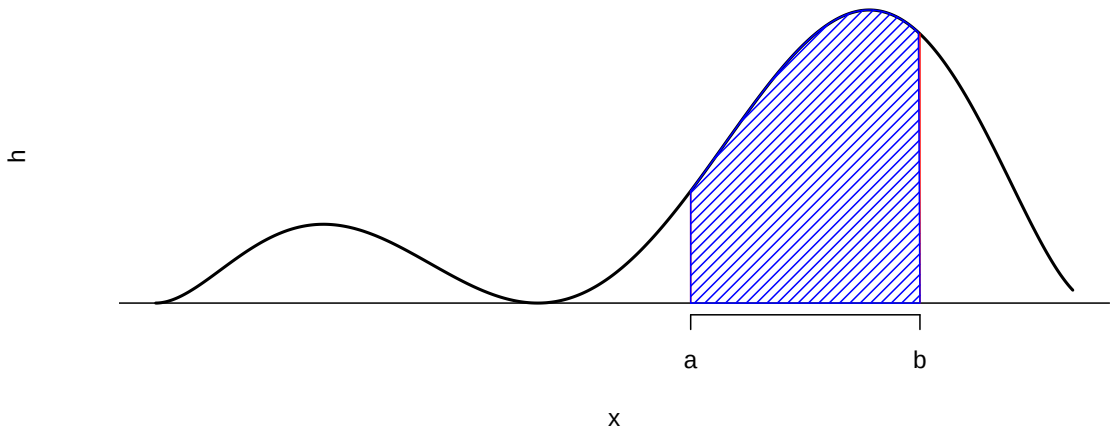
Monte Carlo integration: Motivation

Let x be a random variable having density $f(x)$ and assume h to be an arbitrary function.

The problem of interest is now to compute integrals of the form

$$\int_a^b h(x)dx$$

(always under the assumption that the integral exists).



Since x is a random variable the expectation of $Y = h(X)$ is

$$E_f[h(X)] = \int_X h(x)f(x)dx.$$

Monte Carlo integration

To approximate the $E_f[h(X)]$ Monte Carlo integration uses then random samples $x_1, \ldots, x_m$ coming from f to approximate it with the empirical average

$$\bar{h}_m = \frac{1}{m} \sum_{i=1}^{m} h(x_i).$$

Due to the Strong Law of Large Numbers $\bar{h}_m$ then converges almost surely towards $E_f[h(X)]$.

Large sample theory for Monte Carlo integration

Assume m is large, then according to the central limit theorem

$$\frac{\bar{h}_m - E_f[h(X)]}{s_m} \sim N(0,1).$$

And therefore it is possible to construct convergence tests and confidence bounds on the approximation of $E_f[h(X)]$.

MSE estimation

Let θ be a parameter of interest and $\hat{\theta}$ an estimate of θ. The mean squared error (MSE) of $\hat{\theta}$ is defined as

$$MSE(\hat{\theta}) = E((\theta - \hat{\theta})^2)$$

The Monte Carlo way to estimate the MSE is

$$\hat{MSE} = \frac{1}{m} \sum_{i=1}^{m} (\theta - \hat{\theta}^i)^2$$

where basically m samples $X^i = (x_1^i, \ldots, x_n^i)$, $i = 1, \ldots, m$ are drawn from the sampling distribution and for each sample X^i the corresponding estimate $\hat{\theta}^i$ is computed.

MSE for the median for normal data

Let $x \sim N(0, 1)$. Of interest is then the MSE for μ which is estimated by the median, given the sample size $n = 30$.

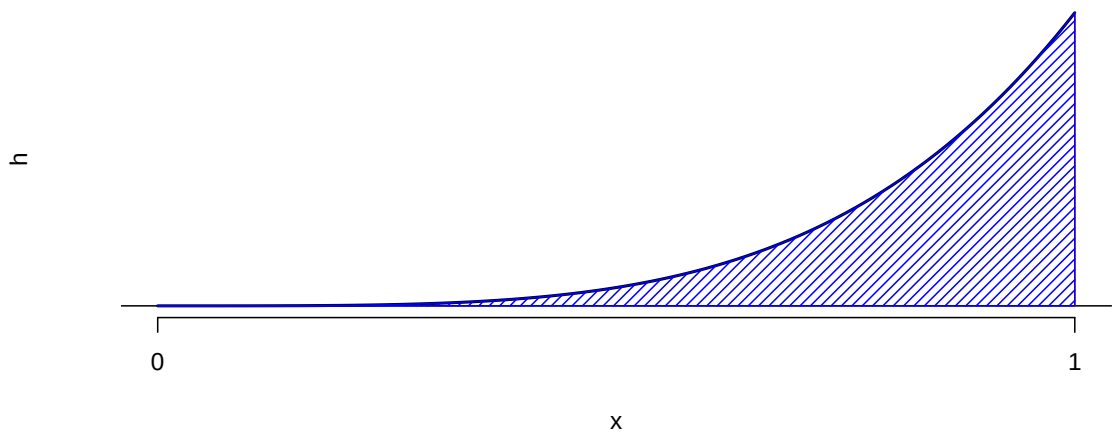
Hence

```
> n <- 30
> m <- 1000
> ThetaTrue <- 0
> Thetas <- replicate(m, median(rnorm(n)))
> MSE <- mean((Thetas-ThetaTrue)^2)
> MSE
[1] 0.04611623
```

Monte Carlo integration example

Assume $x \sim unif\,[0,1]$ and the integral of interest is

$$\int_0^1 x^4 dx.$$



Example in R

Hence in R this is simply:

```
> us <- runif(100000)
> mean(us^4) #Monte Carlo Integral
[1] 0.19917
> #Numerical Integral:
> integrate(f = function(x){x^4},lower = 0,upper = 1)
0.2 with absolute error < 2.2e-15
> 1^5/5 #Exact Integral
[1] 0.2
```

```
Monte Carlo Integral: 0.200247336092673
Numerical Integral:
0.2 with absolute error < 2.2e-15
Exact Integral: 0.2
```

Monte Carlo integration example II

To approximate then

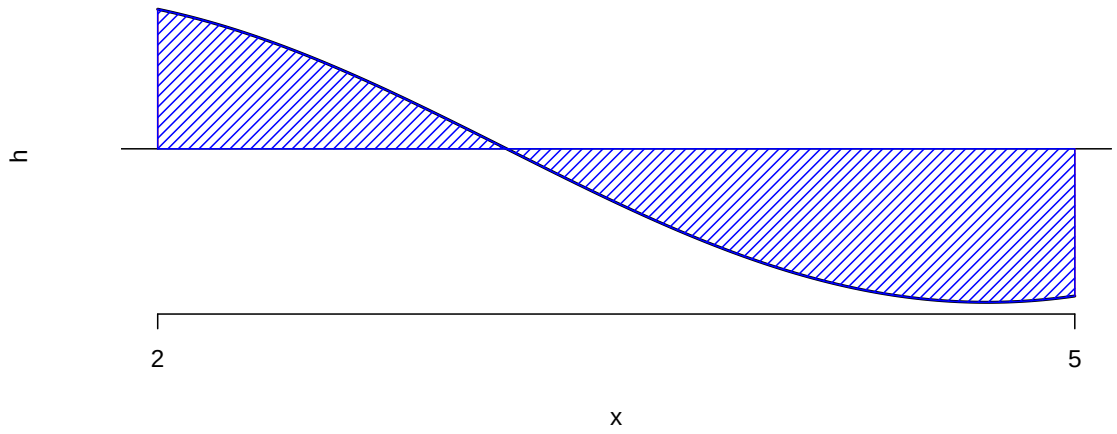
$$\int_2^5 \sin(x)dx.$$

We use

$$E(h(X)) = \int_2^5 \underbrace{\sin(x)}_{h(x)} \cdot \underbrace{\frac{1}{5-2}}_{f(x)} dx.$$

to get

$$\int_2^5 \sin(x)dx = (5-2) \cdot E[h(X)]$$



Example in R

Hence in R this is:

```
> us <- runif(100000, min=2, max=5)
> cat(paste("Monte Carlo Integral:",mean(sin(us)*(5-2))))
Monte Carlo Integral: -0.701770894174566
> cat(paste("Numerical Integral:"))
Numerical Integral:
> integrate(f = function(x){sin(x)},lower = 2,upper = 5)
-0.699809 with absolute error < 2.1e-14
> cat(paste("Exact Integral:",-cos(5)+cos(2)))
Exact Integral: -0.699809022010369
```

Monte Carlo integration example III

The methodology also works with more variables. For example let U and V be independent and $unif\,[0,1]$, then

$$\int_0^1 \int_0^1 h(x,y)dxdy \approx \frac{1}{m}\sum_{i=1}^{m} h(u_i, v_i).$$

So if the integral of interest is:

$$\int_3^{10} \int_1^7 \sin(x-y)dxdy$$

we use

```
> cat(paste("Monte Carlo Integral:",
+     mean(sin(us-vs)*(7-1)*(10-3))))
Monte Carlo Integral: 0.126590044466954
> cat(paste("Exact Integral:",
+       sin(7-10)-sin(1-10)-sin(7-3)+sin(1-3)))
Exact Integral: 0.118503545664136
```

Monte Carlo integration example IV

However, often other densities than the uniform are used.

For example to approximate the value of the integral

$$\int_1^\infty e^{-x^2} dx$$

a density with the same support is needed. Therefore, we rewrite

$$\int_1^\infty e^{-x^2} dx = \int_0^\infty e^{-(x+1)^2} dx$$

and then the exponential distribution can be used.

```
> X <- rexp(100000)
> cat(paste("Monte Carlo Integral:",
+     mean(exp(-(X+1)^2)/dexp(X))))
Monte Carlo Integral: 0.139080112913746
> cat(paste("Numerical Integral:"))
Numerical Integral:
>     integrate(function(x){exp(-x^2)},1,Inf)
0.1394028 with absolute error < 8.4e-05
```

Convergence and variance of Monte Carlo integration

Moreover, if h^2 has a finite expectation under f, then the speed of convergence of $\bar{h}_m$ to $E_f(h(X))$ can be assessed since the variance

$$var(\bar{h}_m) = \frac{1}{m} \int_X (h(x) - E_f(h(X)))^2 f(x) dx$$

can also be estimated from the sample $x_1, \ldots, x_m$ through

$$v_m = \frac{1}{m^2} \sum_{i=1}^{m} (h(x_i) - \bar{h}_m)^2$$

and a standard deviation of

$$s_m = \sqrt{v_m} = \frac{1}{\sqrt{m}} \cdot Var(h(X))$$

Importance sampling algorithm

Importance sampling is a variance reduction technique for MC simulation. Instead of sampling the points randomly, they are sampled more frequently where the integrand is large. By focusing more on these "important" values the variance of the estimate can be reduced.

Summary Monte Carlo integration

The main goal of Monte Carlo integration is usually to approximate an integral of the form

$$\int_a^b h(x)dx$$

The strategy is usually to choose a density f with the same support (transform the integral if needed appropriately). Then the average of $h(x)/f(x)$ is an approximation for the integral.

Naturally the more samples are used the better the approximation.

Nevertheless Monte Carlo integration is not always successful and the convergence depends on the variation of the ratio $h(x)/f(x)$.

The best choice for f is such that the ratio is roughly constant and situations where it can get arbitrarily large should be avoided.

Monte Carlo methods for hypothesis tests and the construction of confidence intervals

In statistics Monte Carlo methods are often used to

- estimate parameters of sampling distributions like test statistics, mean squared errors (MSE) or percentiles.
- evaluate coverage probabilities of confidence intervals.
- estimate the power of a test.
- compare the performance of different methods.

In the following a few examples are given.

Hypothesis Tests

Hypothesis testing is a method of making decisions using data.

Hypothesis tests

hypothesis: assumption about the underlying structure of the population and distribution from which the sample is drawn

- **null hypothesis (H_0)**: basic structure and distribution assumed for the data, if nothing interesting occurs ('standard')
- **alternative hypothesis (H_A)**: interesting case being reasonably different from the standard case ('something happens')

test statistic: value/estimate caluculated from the sample under the distribution assumption of H_0

Significance

Significance

When assuming the standard structure under H_0, we obtain probabilities for every observed sample estimate (independent of the alternative hypothesis).

Significance level α: If the probability of this estimate falls beneath a certain predefined (!) level (=significance level), the null hypothesis is rejected.

Confidence level: This is $1 - \alpha$. This is probability of the data to be generated under the given Null hypothesis scenario. For estimating confidence regions and errors bands we use this instead of significance.

p-value: the probability of observing the sample estimate (=test statistics) or a more extreme value, if drawing randomly from the distribution defined by the null hypothesis. This is the probability is falsely rejecting the null hypothesis.

Confidence Interval coverage

Many statistical methods are derived under the assumption that the data is normal or asymptotically under large sample assumptions.

Often the data is however not normal and the data size is small. In such cases the performance of the method can be evaluated using Monte Carlo methods.

Consider the estimate of a one-sided confidence interval (CI) for the variance σ^2 which is under normality for a sample of size n:

$$[0, (n-1)s^2/\chi^2_{n-1,\alpha}],$$

where s^2 is the sample variance and the significance level α leads to a confidence level $(100(1-\alpha)\%)$.

Confidence Interval coverage II

```
> set.seed(1234)
> n <- 30
> m <- 1000
> alpha <- 0.05
> quant <- qchisq(alpha, df=n-1)
> quant
[1] 17.70837
> sigma2 <- 1
```

Confidence Interval coverage III

```
> s2 <- replicate(m, var(rnorm(n)))
> LCI <- rep(0,m)
> UCL <- (n-1) * s2 / quant
> head(RES <- cbind(LCI,s2,UCL),3)
     LCI        s2      UCL
[1,]   0 0.8153784 1.335300
[2,]   0 0.9196803 1.506109
[3,]   0 0.9591891 1.570810
> colMeans(RES)
      LCI        s2       UCL
0.0000000 0.9962934 1.6315740
> coverage <- mean(UCL > sigma2)
> coverage
[1] 0.956
```

Statistical Errors

Error Table

	H_0 TRUE	H_0 FALSE
keep H_0	✓	Type II error
reject H_0	Type I error	✓

- **False positive rate** = significance level (α): probability of committing Type I error, incorrectly rejecting H_0 (positive outcome)
- **False negative rate** (β): probability to commit Type II error, incorrectly accepting H_0 (negative outcome)
- **Power** ($1 - \beta$): probability of correctly rejecting H_0

Statistical Errors

The **significance level** (α) is chosen by the statistician *before* the analysis and typically values such as $\alpha = 0.05$, $\alpha = 0.01$ or $\alpha = 0.001$ are chosen

The **Power** $1 - \beta$ is unknown, but affected by:

- **sample size**: the more observations, the larger the power
- **effect size**: the larger the effect (difference of values) to be detected, the larger the power
- **significance level**: the smaller the significance level, the smaller the power

The smaller the α error is chosen, the larger the β error becomes!

Power of a test

The type 1 error is usually controlled by the level α. The power of the test is measured as a function giving the probability of rejecting correctly the null as a function of the magnitude of the alternative Δ.

The power function can be estimated using Monte Carlo methods where repeatedly for a series of fixed values of Δ the number of rejections is counted. Usually the null case is included in power curve estimations because it is only valid if the level of the test is correct.

Power of the one sample t-test

```
> n <- 30; m <- 1000; alpha <- 0.05
> Delta1 <- 0; Delta2 <- 0.1; Delta3 <- 0.2; Delta4 <- 0.3
> set.seed(1)
> PvalueD1 <- replicate(m,t.test(rnorm(n)+Delta1)$p.value)
> set.seed(1)
> PvalueD2 <- replicate(m,t.test(rnorm(n)+Delta2)$p.value)
> set.seed(1)
> PvalueD3 <- replicate(m,t.test(rnorm(n)+Delta3)$p.value)
> set.seed(1)
> PvalueD4 <- replicate(m,t.test(rnorm(n)+Delta4)$p.value)
> POWER <-colMeans(cbind(PvalueD1,PvalueD2,PvalueD3,
+                        PvalueD4)<alpha)
> names(POWER) <- c("D0","D1","D3","D4")
> POWER
    D0     D1     D3     D4
0.042  0.072  0.158  0.350
```