Simulation and Resampling

Maribel Borrajo García and Mercedes Conde Amboage

maribelborrajo@uniovi.es; mercedes.amboage@usc.es

Universidad de Oviedo
DEPARTAMENTO DE ESTADÍSTICA
E INVESTIGACIÓN OPERATIVA
Y DIDÁCTICA DE LA MATEMÁTICA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA
DEPARTAMENTO DE ESTATÍSTICA,
ANÁLISE MATEMÁTICA E OPTIMIZACIÓN

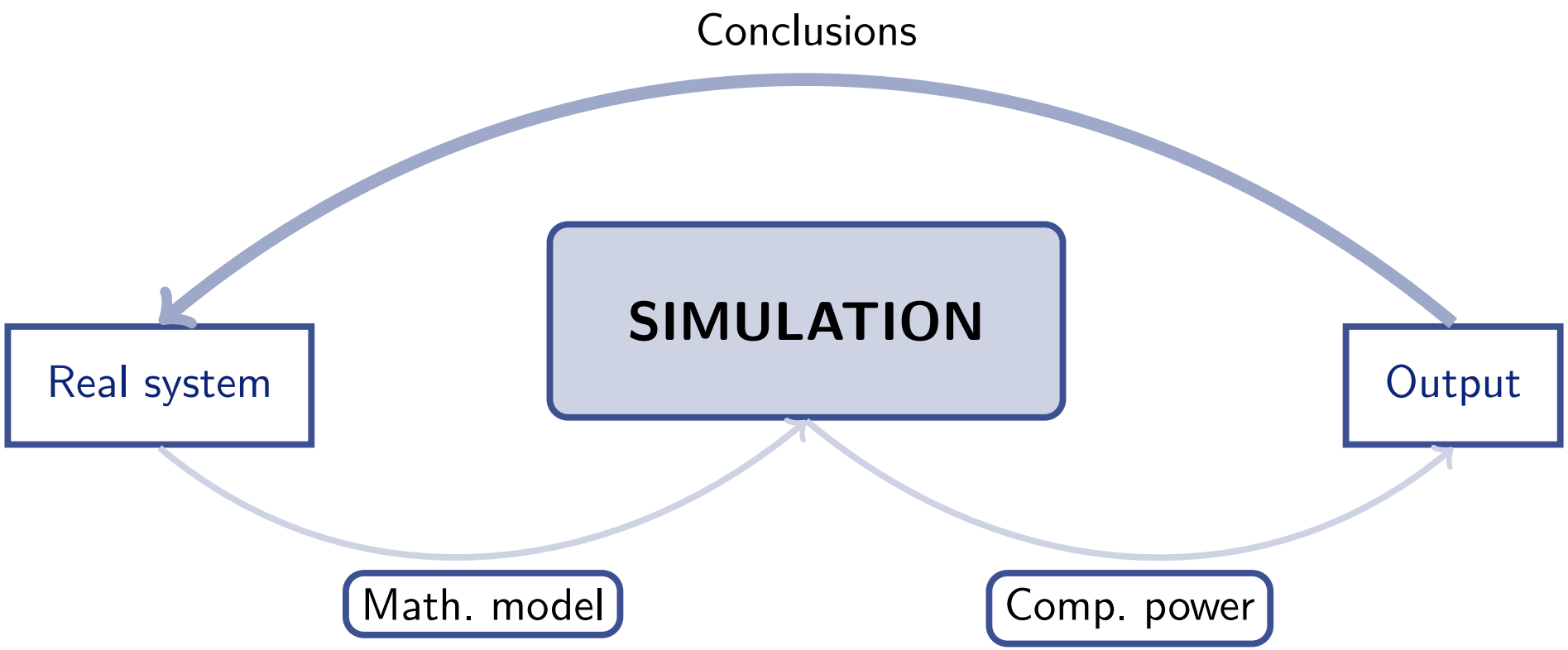
modestya
Modelos de Optimización, Decisión, Estadística y Aplicaciones

CiTiUS
Centro Singular de Investigación
en Tecnoloxías da
Información

Introduction

What is simulation?

The imitation of a real world process or system using mathematical models and computational resources.



What is simulation?

The imitation of a real world process or system using mathematical models and computational resources

- **Natural sample:** set of observations from the population obtained through field work.
- **Artificial sample (lab sample):** set of observations from the population NOT obtained through field work.

To obtain artificial samples we need to know the population.

What is simulation?

Advantages

- The only tool if mathematical methods are not available.
- Existing mathematical models are too complex.
- Allows comparison of alternative designs.

Drawbacks

- Computational power.
- For stochastic models, simulation estimates the output, while the analytical method, if available, produces exact solutions.
- Not appropriate models may result in wrong conclusions.

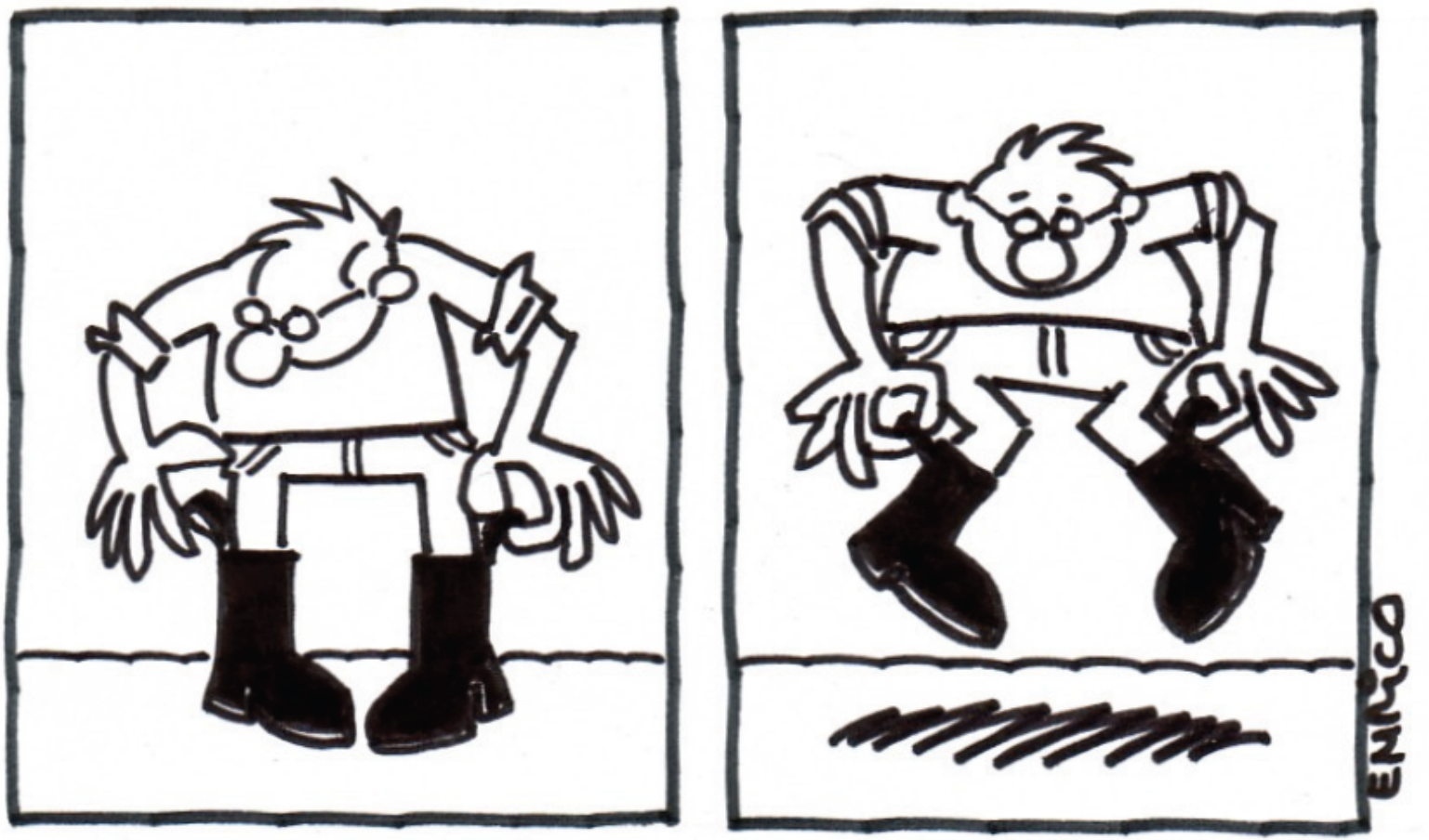
Applications of simulation

- Approximation of the distribution (or characteristic) of a certain statistic (bias and variability) with Monte Carlo methods.
- Comparison of two confidence intervals approximating with Monte Carlo their coverage.
- Comparison of two hypothesis tests approximating with Monte Carlo their power functions.

There exist a wider variety of applications based on these ones.

What is bootstrap?

"Pull yourself up by your own bootstraps"



What is bootstrap?

- Bradley Efron (Stanford University, 1979). He defined the method by mixing Monte Carlo with problem resolution in a very general way.
- Peter Hall (1951-2016). He was one of the most prolific nowadays statisticians and he devoted a huge part of his work to bootstrap from the 80's.

Hypothesis on the population distribution are not required.

Part I: Simulation

Part I: Simulation

Module I: Simulation of univariate distributions

Univariate simulation

Continuos variables

- The inverse transform method.
- Acceptance-rejection methods.
- ...

Discrete variables

- The generalized inverse method or quantile transformation method.
- Truncation methods.
- ...

Cao, R. (2002) Introducción a la simulación y a la teoría de colas. NetBiblio.

The inverse transform method

Probability Integral Transform

Any random variable can be transformed into a uniform random variable and, more importantly, vice versa.

That is, if a random variable X has density f and cumulative distribution function F, then it follows that

$$F(x) = \int_{-\infty}^{x} f(t)dt,$$

and if we set $U = F(X)$, then U is a random variable distributed from a uniform $U(0, 1)$.

The inverse transform method

Probability Integral Transform

Any random variable can be transformed into a uniform random variable and, more importantly, vice versa.

That is, if a random variable X has density f and cumulative distribution function F, then it follows that

$$F(x) = \int_{-\infty}^{x} f(t)dt, \qquad \mathbf{X = F^{-1}(U)}$$

and if we set $U = F(X)$, then U is a random variable distributed from a uniform $U(0, 1)$.

The inverse transform method

Example: Let us consider a random variable that follows a Maxwell distribution which density is given by

$$f(x) = xe^{-x^2/2} \quad \text{if} \quad x > 0,$$

and as a consequence

$$F(x) = \int_{-\infty}^{x} f(t)dt = \int_{0}^{x} te^{-t^2/2}dt = 1 - e^{-x^2/2} \quad \text{con} \quad x > 0.$$

In this case, it is easy to compute the inverse of this distribution function as

$$y = F(x) \Leftrightarrow y = 1 - e^{-x^2/2} \Leftrightarrow x = \sqrt{-2\ln(1-y)}.$$

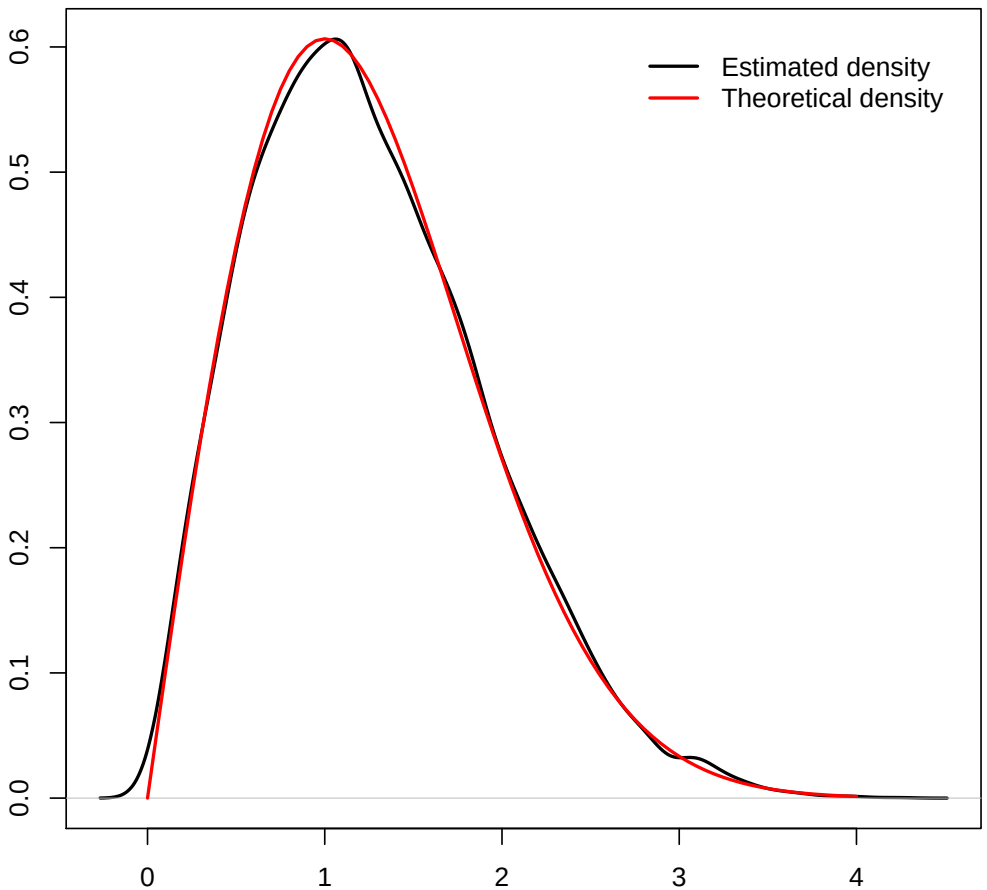
Continuous univariate simulation

```
> n=10000
> u<-runif(n)
> maxw<-sqrt(-2*log(u))

> fmaxw<-function(x){
+     x*exp(-x^2/2)
}

> sec=seq(0,4,by=0.05)
> fsec=fmaxw(sec)

> plot(density(maxw),lwd=2)
> lines(sec,fsec,col=2,lwd=2)
```



Acceptance-rejection methods

Assumptions: The functional form of the density f of interest (called the **target density**) up to a multiplicative constant is known.

Idea: We use a simpler (to simulate) density g, called the **instrumental or candidate density**, to generate the random variable for which the simulation is actually done.

Constraints:

- f and g have compatible supports, i.e., $g(x) > 0$ when $f(x) > 0$.
- There is a constant M with $\frac{f(x)}{g(x)} \leq M$ for all x.

Acceptance-rejection methods

Algorithm

1. Generate $Y \sim g$ and $U \sim U(0,1)$.
2. Accept $X = Y$ if $U \leq \frac{f(Y)}{Mg(Y)}$.
3. Return to Step 1 otherwise.

Acceptance-rejection methods

Algorithm

1. Generate $Y \sim g$ and $U \sim U(0,1)$.
2. Accept $X = Y$ if $U \leq \frac{f(Y)}{Mg(Y)}$.
3. Return to Step 1 otherwise.

```
> u=runif(1)*M
> y=randg(1)
> while (u>f(y)/g(y)){
+     u=runif(1)*M
+     y=randg(1)
}
```

Acceptance-rejection methods

Algorithm

1. Generate $Y \sim g$ and $U \sim U(0,1)$.
2. Accept $X = Y$ if $U \leq \frac{f(Y)}{Mg(Y)}$.
3. Return to Step 1 otherwise.

```
> u=runif(1)*M
> y=randg(1)
> while (u>f(y)/g(y)){
+     u=runif(1)*M
+     y=randg(1)
}
```

randg is a function that delivers generations from the density g.

Generalized inverse method

Idea: Given a random variable with distribution function F and quantile function $Q(u) = \inf\{x_j / \sum_{i=1}^{j} p_i \geq u\}$. If we consider a random variable $U \sim U(0,1)$ then the variable $Q(U)$ has the same distribution as X. This method is also called **quantile transformation**.

Problem: Compute the quantile function Q is not a easy problem in some cases. It is needed to use a **sequential search**.

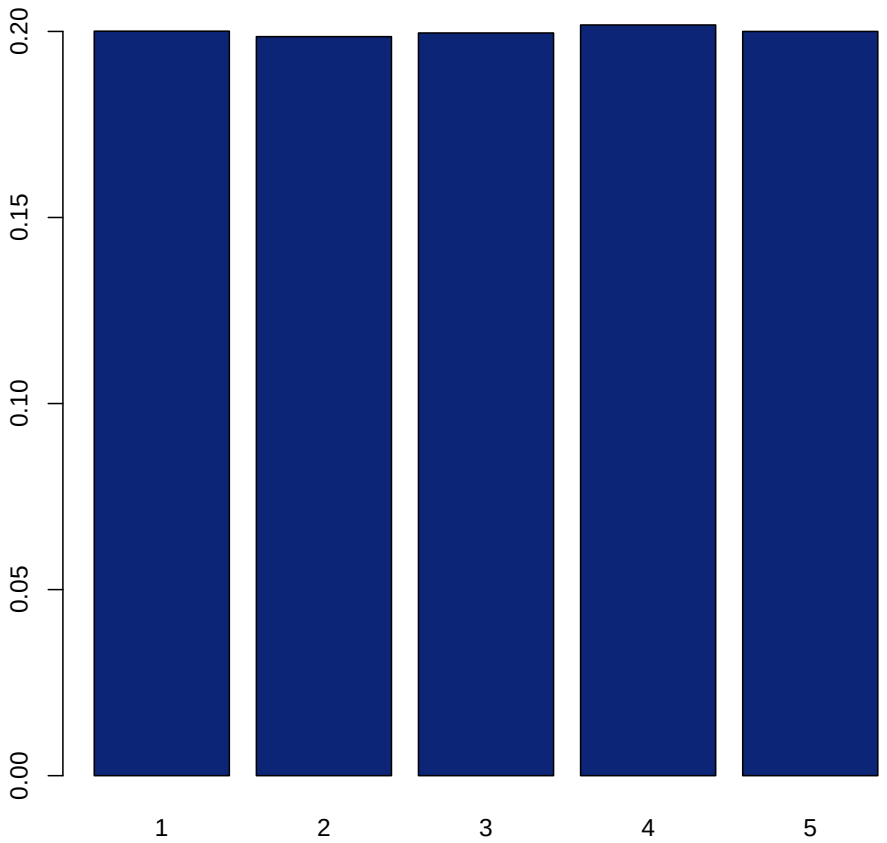
Algorithm:

1. Generate $U \sim U(0,1)$.
2. Put $I = 1$ and $S = p_1$.
3. While $U > S$ compute $I = I + 1$ and $S = S + p_I$.
4. Return $X = x_I$.

Generalized inverse method

Example: Let us consider a discrete variable that takes values $x_1, \ldots, x_n$ with the same probability $1/n$.

```
> gdv<-function(x, prob) {
+    i <- 1
+    Fx <- prob[1]
+    U <- runif(1)
+    while (Fx < U) {
+       i <- i + 1
+       Fx <- Fx + prob[i]
+    }
+    return(x[i])
+ }
```



Truncation methods

Use a continuous distribution similar to the discrete one

$$X \left| \begin{array}{ccccc} x_1 & x_2 & x_3 & \cdots & x_n \\ p_1 & p_2 & p_3 & \cdots & p_n \end{array} \right. \qquad F(x) = \sum_{x_i \leq x} p_i.$$

We define $Y \sim G$ and values $-\infty = a_0 < a_1 < a_2 < \cdots < a_{n-1} < a_n = \infty$ such that

$$F(x_i) - F(x_i^-) = p_i = G(a_i) - G(a_{i-1}) \quad i = 1, \ldots, n.$$

Then $P(Y \in [a_{i-1}, a_i)) = p_i$. If the continuous r.v. Y is easy to simulate, we can generate values from it and transform them into values of X.

Cao, R. (2002) Introducción a la simulación y a la teoría de colas. NetBiblio.

Truncation methods

Algorithm

1. Generate $Y \sim G$.
2. Find $i / a_{i-1} < Y < a_i$.
3. Return x_i.

Cao, R. (2002) Introducción a la simulación y a la teoría de colas. NetBiblio.

Part I: Simulation

Module II: Simulation of Multidimensional Distributions

Introduction

Given a pair of variables (X, Y), let us define the distribution function as

$$F_{X,Y}(x, y) = P(X \leq x, Y \leq y)$$

Introduction

Given a pair of variables (X, Y), let us define the distribution function as

$$F_{X,Y}(x, y) = P(X \leq x, Y \leq y)$$

Discrete case

$$F_{X,Y}(x, y) = \sum_{x_i \leq x} \sum_{y_j \leq y} P(X = x_i, Y = y_j) = \sum_{x_i \leq x} \sum_{y_j \leq y} p_{ij}.$$

Introduction

Given a pair of variables (X, Y), let us define the distribution function as

$$F_{X,Y}(x, y) = P(X \leq x, Y \leq y)$$

Discrete case

$$F_{X,Y}(x, y) = \sum_{x_i \leq x} \sum_{y_j \leq y} P(X = x_i, Y = y_j) = \sum_{x_i \leq x} \sum_{y_j \leq y} p_{ij}.$$

Continuous case

$$F_{X,Y}(x, y) = P(X \leq x, Y \leq y) = \int_{-\infty}^{x} \int_{-\infty}^{y} f_{X,Y}(u, v) du dv.$$

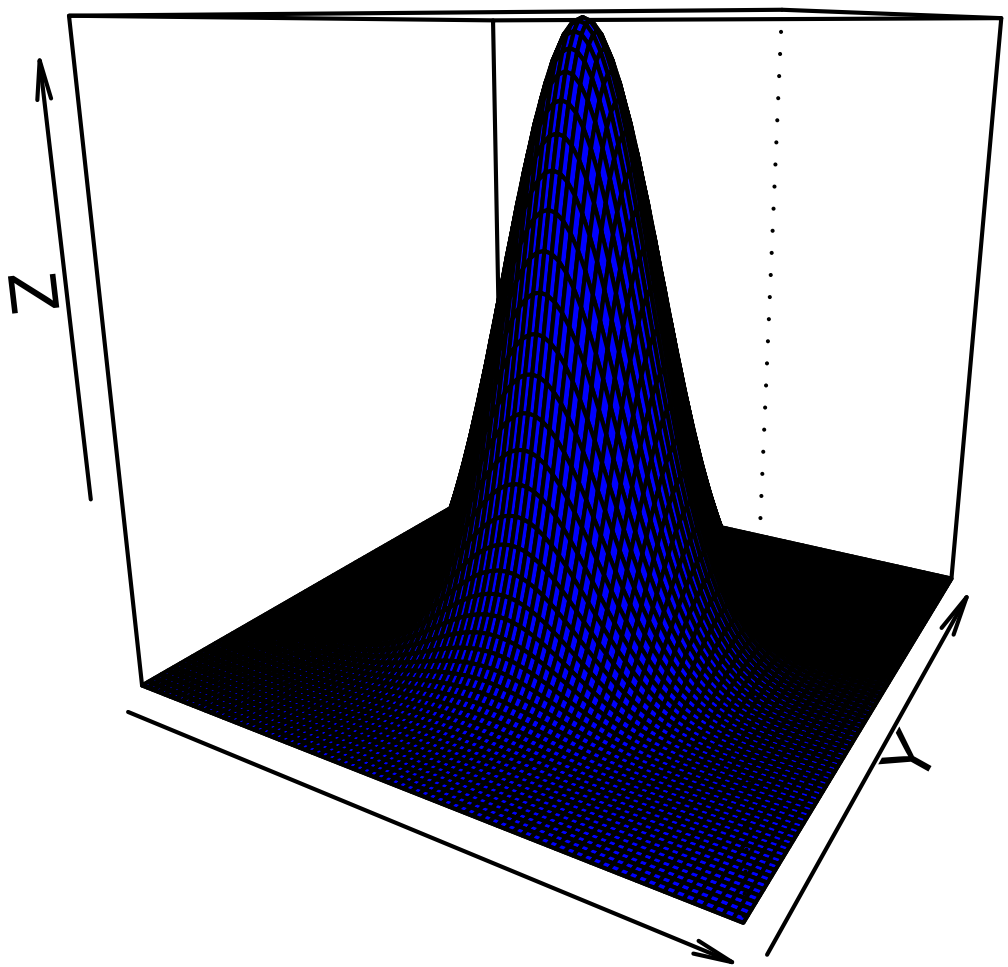
Introduction

Definition

Given a pair of variables X and Y of continuous variables, if X and Y are **independent** then it follows that

$$f_{X,Y}(x,y) = f_X(x) \cdot f_Y(y)$$

for all $x, y \in \mathbb{R}$.



Introduction

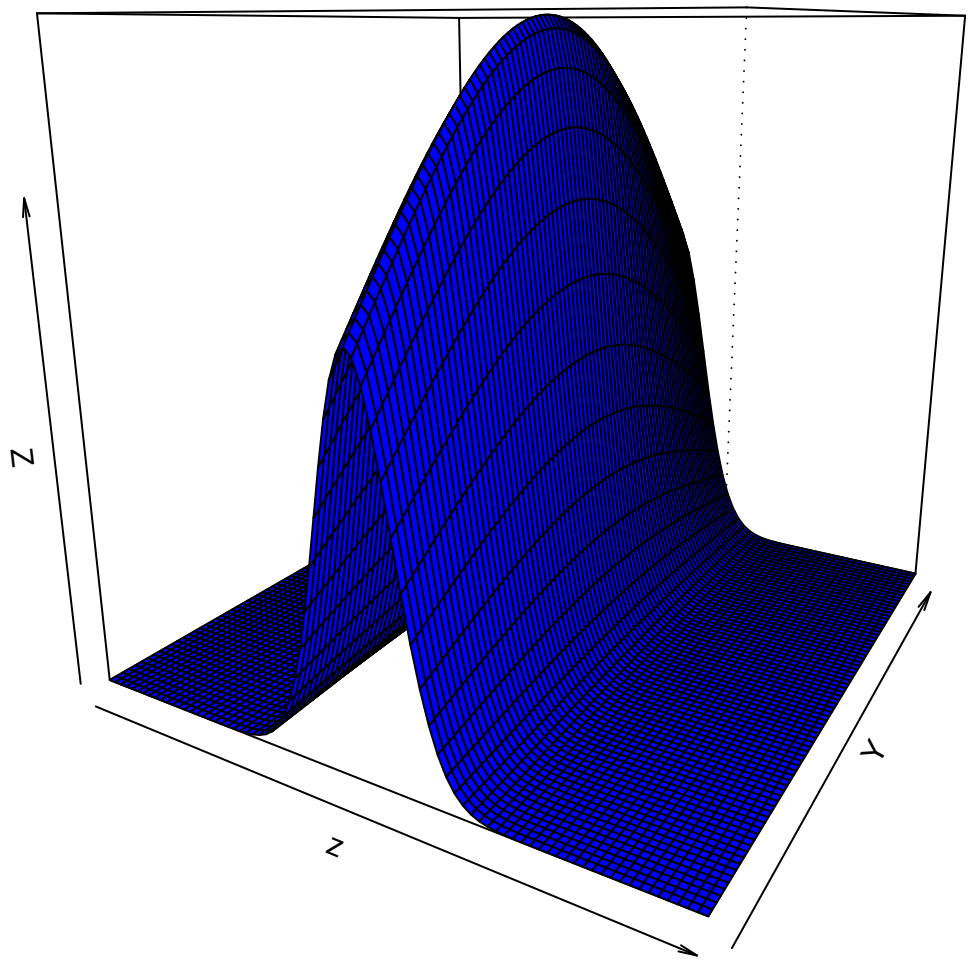
Definition

Given a pair of continuous variables X and Y, it follows that

$$f_{X,Y}(x,y) = f_X(x) \cdot f_{Y|X}(y|x) = f_Y(y) \cdot f_{X|Y}(x|y)$$

where

$$f_{Y|X}(y \mid X = x) = \frac{f_{X,Y}(x,y)}{f_X(x)}$$



Conditional distribution methods

Idea: this method is based on the following decomposition of the density function

$$f(x_1, x_2, \ldots, x_d) = f_1(x_1) \cdot f_2(x_2|x_1) \cdot \ldots \cdot f_d(x_d|x_1, x_2, \ldots, x_{d-1}),$$

where the conditional densities can be obtained as a function of the marginal densities

$$f_i(x_i|x_1, x_2, \ldots, x_{i-1}) = \frac{f_{1,\ldots,i}(x_1, x_2, \ldots, x_i)}{f_{1,\ldots,i-1}(x_1, x_2, \ldots, x_{i-1})}.$$

Conditional distribution methods

Idea: this method is based on the following decomposition of the density function

$$f(x_1, x_2, \ldots, x_d) = f_1(x_1) \cdot f_2(x_2|x_1) \cdot \ldots \cdot f_d(x_d|x_1, x_2, \ldots, x_{d-1}),$$

where the conditional densities can be obtained as a function of the marginal densities

$$f_i(x_i|x_1, x_2, \ldots, x_{i-1}) = \frac{f_{1,\ldots,i}(x_1, x_2, \ldots, x_i)}{f_{1,\ldots,i-1}(x_1, x_2, \ldots, x_{i-1})}.$$

Algorithm:

1. Generate $X_1 \sim f_1$.
2. From $i = 2$ to d, generate $X_i \sim f_i(\cdot|X_1, X_2, \ldots, X_{i-1})$.
3. Return $\mathbf{X} = (X_1, X_2, \ldots, X_d)$.

Conditional distribution methods

Example: Generate 1000 values from a bivariate Gaussian distribution with the following covariance matrix

$$\Sigma = \begin{pmatrix} 4 & 1 \\ 1 & 9 \end{pmatrix}$$

using conditional distribution method.

Conditional distribution methods

Example: Generate 1000 values from a bivariate Gaussian distribution with the following covariance matrix

$$\Sigma = \begin{pmatrix} 4 & 1 \\ 1 & 9 \end{pmatrix}$$

using conditional distribution method.

Theoretical ideas

- Given

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N_2\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}\begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}\right)$$

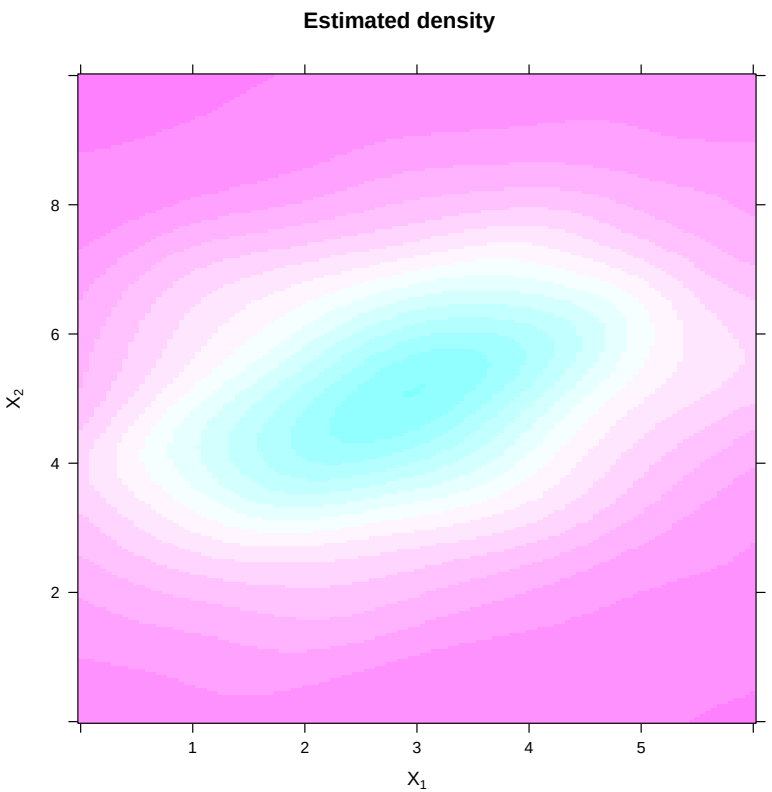
  then $X_1 \sim N(0, \sigma_1)$ and $X_2|X_1 \sim N\left(\frac{\sigma_{12}}{\sigma_1^2}X_1, \sqrt{\frac{\sigma_1^2\sigma_2^2 - \sigma_{12}^2}{\sigma_1^2}}\right)$.

- If $X \sim N(0, \sigma)$ then $X + \mu \sim N(\mu, \sigma)$.

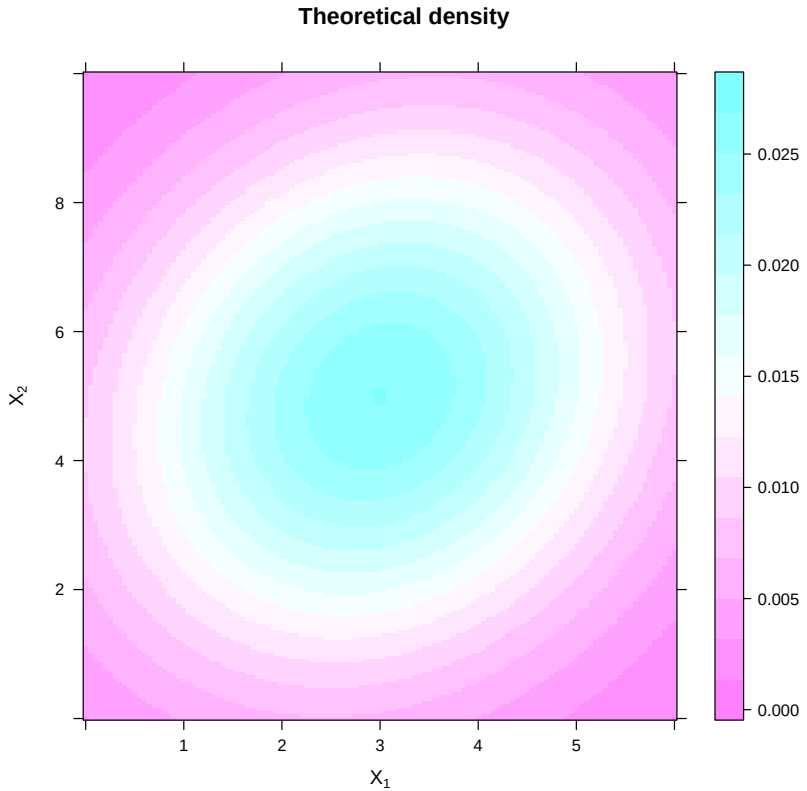
Conditional distribution methods

```
> library(ks); library(mvtnorm); library(lattice)
> set.seed(1234)
> n=1000; sigma1=2; mu1=3; sigma2=3; mu2=5; sigma12=1
> sigma=matrix(c(sigma1^2,sigma12,sigma12,sigma2^2),ncol=2)
> z1=rnorm(n); z2=rnorm(n)
> y1=sigma1*z1
> y2=sigma12/sigma1^2*y1+z2*sqrt((sigma1^2+sigma2^2
+   -sigma12^2)/sigma1^2)
> x1=y1+mu1; x2=y2+mu2
>
> d <- expand.grid('x3'=seq(0,6,.05),'x4'=seq(0,10,.05))
> fhat <- kde(cbind(x1,x2), eval.points= d)
> levelplot(fhat$estimate ~ d$x3*d$x4,main='Estimated
+    density',xlab=expression(X[1]),ylab=expression(X[2]))
>
> d$dens<-dmvnorm(as.matrix(d), mean=c(mu1,mu2), sigma=sigma)
> levelplot(dens ~ x3*x4, data=d, xlab=expression(X[1]),
+    ylab=expression(X[2]), main='Real density')
```

Conditional distribution methods



(a) Estimation



(b) Theory

Acceptance-rejection methods

Idea: Simulate random variable X from distribution with density f and distribution function F.
We start with a random variable Y with the density g such that $f(x) \leq Mg(x)$, $M < \infty$ for all x. Them, given $Y = x$, one accepts Y and lets $X = Y$ with probability $f(x)/Mg(x)$. Otherwise, a new Y is generated and one continues until eventual acceptance.

Acceptance-rejection methods

Idea: Simulate random variable X from distribution with density f and distribution function F.
We start with a random variable Y with the density g such that $f(x) \leq Mg(x)$, $M < \infty$ for all x. Them, given $Y = x$, one accepts Y and lets $X = Y$ with probability $f(x)/Mg(x)$. Otherwise, a new Y is generated and one continues until eventual acceptance.

Algorithm:

1. Generate $Y \sim g$ and $U \sim U(0,1)$.
2. Accept $X = Y$ if $U \leq \frac{f(Y)}{Mg(Y)}$.
3. Return to Step 1 otherwise.

Acceptance-rejection methods

Idea: Simulate random variable X from distribution with density f and distribution function F.
We start with a random variable Y with the density g such that $f(x) \leq Mg(x)$, $M < \infty$ for all x. Them, given $Y = x$, one accepts Y and lets $X = Y$ with probability $f(x)/Mg(x)$. Otherwise, a new Y is generated and one continues until eventual acceptance.

Algorithm:

1. Generate $Y \sim g$ and $U \sim U(0,1)$.
2. Accept $X = Y$ if $U \leq \frac{f(Y)}{Mg(Y)}$.
3. Return to Step 1 otherwise.

Remark! When the dimension increases, it is difficult to find the auxiliary density g and the iterations of the method grows dramatically.

Acceptance-rejection methods

Idea: Simulate random variable X from distribution with density f and distribution function F.
We start with a random variable Y with the density g such that $f(x) \leq Mg(x)$, $M < \infty$ for all x. Them, given $Y = x$, one accepts Y and lets $X = Y$ with probability $f(x)/Mg(x)$. Otherwise, a new Y is generated and one continues until eventual acceptance.

Algorithm:

1. Generate $Y \sim g$ and $U \sim U(0,1)$.
2. Accept $X = Y$ if $U \leq \frac{f(Y)}{Mg(Y)}$.
3. Return to Step 1 otherwise.

Example: In Cao (2002), we can find an example that show how to generate points uniformly distributed on a d-dimensional sphere.

Cao, R. (2002) Introducción a la simulación y a la teoría de colas. NetBiblio.

Factorization of the covariance matrix

It is well know that if $cov(X) = \Sigma$ entonces $cov(AX) = A\Sigma A^t$.

Idea: simulate independent data and then transform them linearly in order to obtain the wanted covariance.

Factorization of the covariance matrix

It is well know that if $cov(X) = \Sigma$ entonces $cov(AX) = A\Sigma A^t$.

Idea: simulate independent data and then transform them linearly in order to obtain the wanted covariance.

This method is used to simulate data from a multidimensional Gaussian or t-Student distribution. If $X \sim N_d(\mu, \Sigma)$ and A is a matrix of dimension $p \times d$, then $AX \sim N_p(A\mu, A\Sigma A^t)$.

Factorization of the covariance matrix

It is well know that if $cov(X) = \Sigma$ entonces $cov(AX) = A\Sigma A^t$.

Idea: simulate independent data and then transform them linearly in order to obtain the wanted covariance.

This method is used to simulate data from a multidimensional Gaussian or t-Student distribution. If $X \sim N_d(\mu, \Sigma)$ and A is a matrix of dimension $p \times d$, then $AX \sim N_p(A\mu, A\Sigma A^t)$.

- If $Z \sim N_d(0, I_d)$ and $\Sigma = HAH^t = HA^{1/2}(HA^{1/2})^t$, then

$$Y = \mu + HA^{1/2}Z \sim N_d(\mu, \Sigma).$$

- If $Z \sim N_d(0, I_d)$ and $\Sigma = LL^t$ using a Cholesky factorization, then

$$Y = \mu + LZ \sim N_d(\mu, \Sigma).$$

Factorization of the covariance matrix

Algorithm:

1. Obtain the Cholesky factorization $\Sigma = LL^t$.
2. Simulate $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_d)$ a random sample of a standard Gaussian distribution.
3. Compute $X = \mu + LZ$.
4. Repeat Step 2 and Step 3 several times.

Remark! Note that the algorithm depends on the Cholesky factorization. For example, if using a Cholesky factorization we obtain $\Sigma = U^t U$ then we have to use $L = U^t$.

Factorization of the covariance matrix

Example: Let us consider the functional variable

$$Y(x) = \sin(2\pi x) + \varepsilon(x),$$

where $0 \leq x \leq 1$ and $\text{Cov}(\varepsilon(x), \varepsilon(y)) = e^{-\|x-y\|}$. Obtain a sample of this variable (sample size $n = 100$) using 50 discretization points.

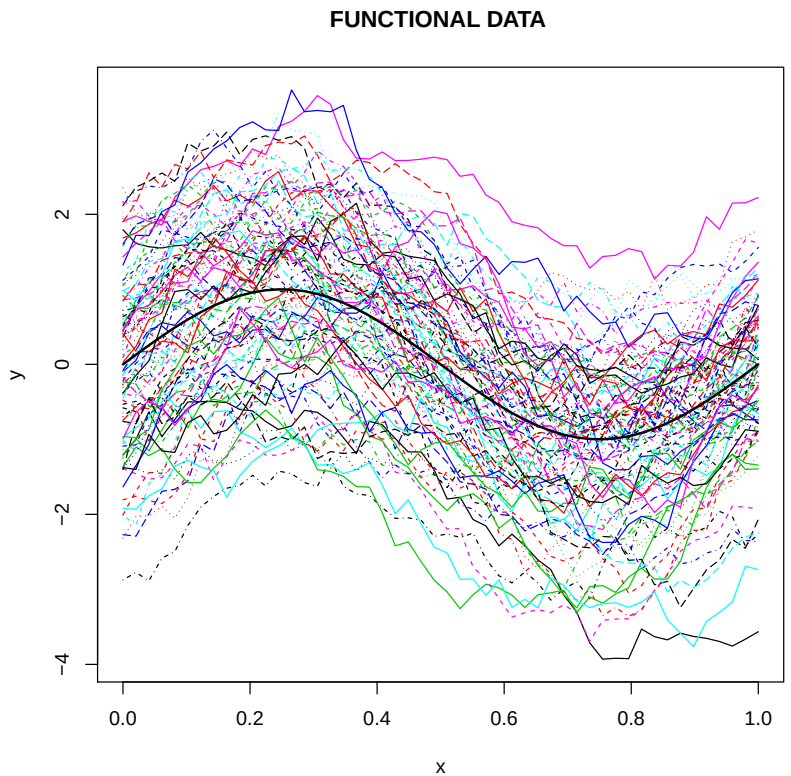
Factorization of the covariance matrix

Example: Let us consider the functional variable

$$Y(x) = \sin(2\pi x) + \varepsilon(x),$$

where $0 \leq x \leq 1$ and $\text{Cov}(\varepsilon(x), \varepsilon(y)) = e^{-\|x-y\|}$. Obtain a sample of this variable (sample size $n = 100$) using 50 discretization points.

```
> n <- 100; p <- 50
> x <- seq(0, 1, length = p)
> mu <- sin(2*pi*x)
> x.dist<-as.matrix(dist(x))
> x.cov <- exp(-x.dist)
> # Cholesky factorization
> U <- chol(x.cov)
> L <- t(U)
> set.seed(1234) # Simulation
> z <- matrix(rnorm(n*p), nrow=p)
> y <- mu + L %*% z
> matplot(x, y, type = "l")
> lines(x, mu, lwd=2)
```



Simulation based on copulas

Definition

A **copula** is a multidimensional distribution function that verifies that each marginal distribution follows a uniform distribution.

This kind of functions is used to build multivariate distributions based on marginal distributions.

Simulation based on copulas

Definition

A **copula** is a multidimensional distribution function that verifies that each marginal distribution follows a uniform distribution.

This kind of functions is used to build multivariate distributions based on marginal distributions.

Sklar Theorem (1959)

Given (X, Y) a random variable with distribution $F(\cdot, \cdot)$, and $F_1(\cdot)$ and $F_2(\cdot)$ represent the corresponding marginal distributions. Then, there exists a copula for which

$$F(x, y) = C(F_1(x), F_2(y)), \quad \forall x, y \in \mathbb{R}.$$

Moreover, if $F_1(\cdot)$ and $F_2(\cdot)$ are continuous, then $C(\cdot, \cdot)$ is unique. The reciprocal is true.

Nelsen, R.B.(2006). An introduction to copulas, 2a ed., Springer.

Simulation based on copulas

Idea: If $(U, V) \sim C(\cdot, \cdot)$ (uniform marginal distributions):

$$\left(F_1^{-1}(U)\ ,\ F_2^{-1}(V)\right) \sim F(\cdot, \cdot).$$

In most cases, we know explicit expression of $C_u(v) = C_2(v|u)$ and of its inverse $C_u^{-1}(w)$. As a consequence, it is easy to generate values from (U, V) using the conditional distributions methods.

Simulation based on copulas

Idea: If $(U, V) \sim C(\cdot, \cdot)$ (uniform marginal distributions):

$$\left(F_1^{-1}(U)\ ,\ F_2^{-1}(V)\right) \sim F(\cdot, \cdot).$$

In most cases, we know explicit expression of $C_u(v) = C_2(v|u)$ and of its inverse $C_u^{-1}(w)$. As a consequence, it is easy to generate values from (U, V) using the conditional distributions methods.

Algorithm

1. Generate $U, W \sim U(0, 1)$.
2. Obtain $V = C_u^{-1}(W)$.
3. Return $(F_1^{-1}(U), F_2^{-1}(V))$.

Simulation based on copulas

Example: Let us consider a random bidimensional variable with marginal distributions that follows an Exp(1) and an Exp(2), respectively. Moreover the joint distribution is determined by a Clayton copula

$$C_\alpha(u,v) = \max\left\{\left(u^{-\alpha}+v^{-\alpha}-1\right)^{-1/\alpha}, 0\right\},$$

where $\alpha \in [-1,+\infty] \setminus \{0\}$. In this case, it follows that

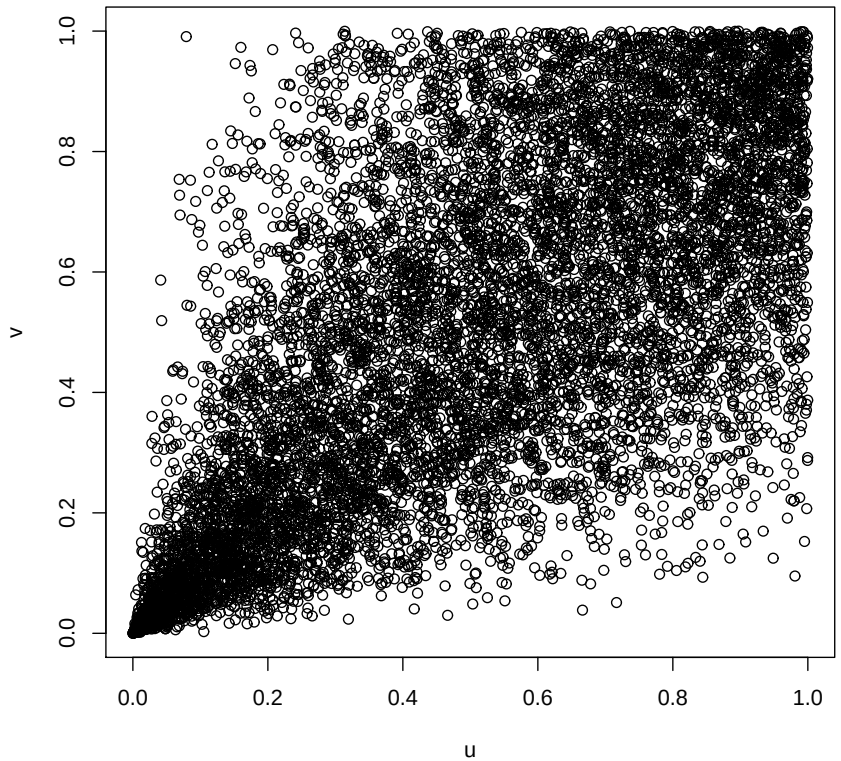
$$C_u^{-1}(w) = \left(u^{-\alpha}\left(w^{-\frac{\alpha}{\alpha+1}}-1\right)+1\right)^{-\frac{1}{\alpha}}.$$

Taking $\alpha = 2$, design an R code to genera samples from this distribution.

Simulation based on copulas

- Generate values from the copula.

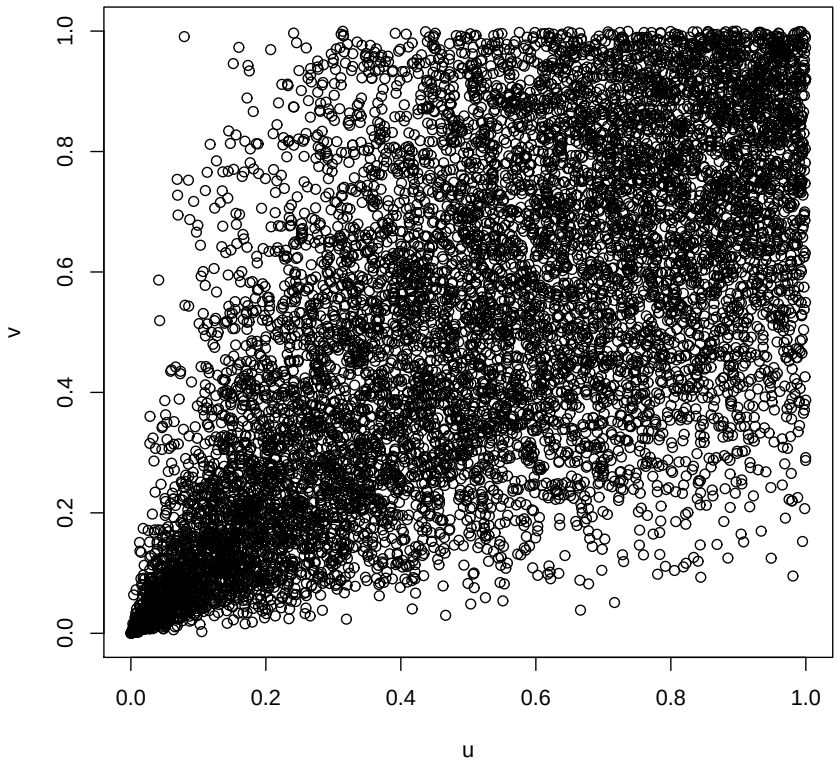
```
> set.seed(1234)
> rcclayton <- function(alpha, n) {
+    val <- cbind(runif(n), runif(n))
+    val[, 2] <- (val[, 1]^(-alpha)
+    * (val[, 2]^(-alpha/(alpha
+    + 1)) - 1) + 1)^(-1/alpha)
+    return(val)
+ }
> rcunif <- rcclayton(2,10000)
> plot(rcunif, xlab = 'u', ylab = 'v')
```



Simulation based on copulas

- Generate values from the copula: other way.

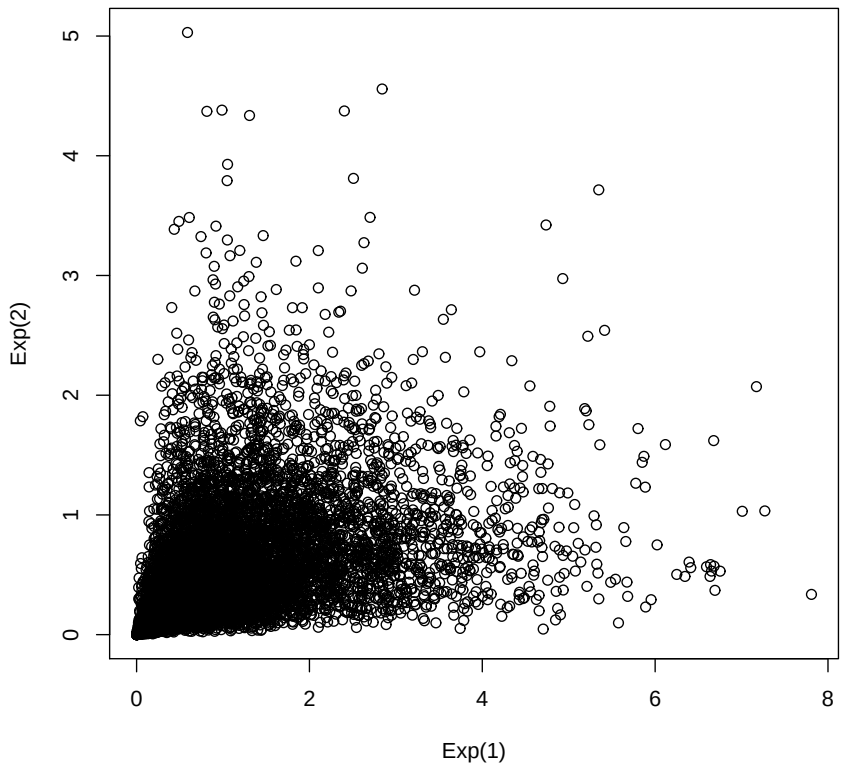
```
> library(copula)
> clayton.cop<-claytonCopula(2, dim=2)
> y <- rCopula(10000, clayton.cop)
> plot(y,xlab='u',ylab='v')
```



Simulation based on copulas

- Generate the bidimensional distribution.

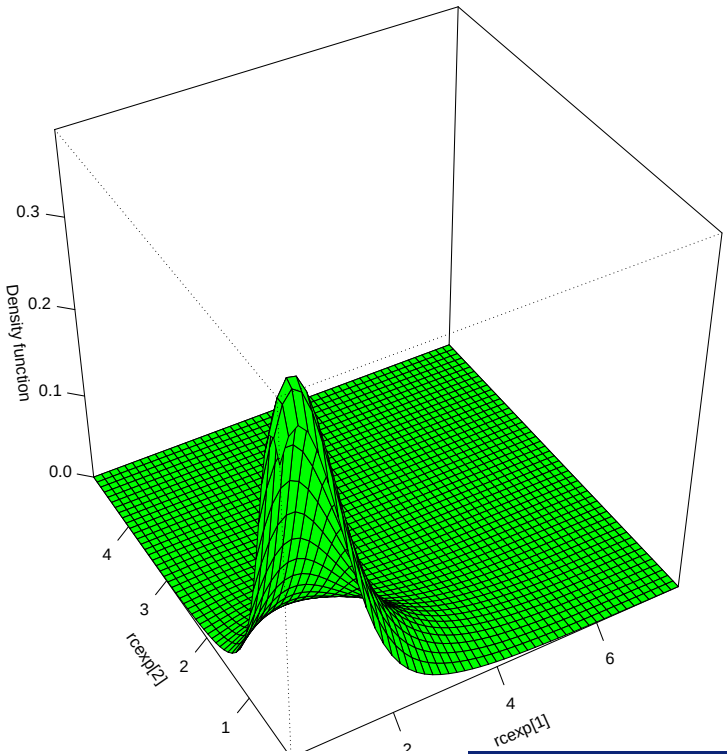
```
> rcexp <- cbind(qexp(rcunif[,1], 1), qexp(rcunif[,2], 2))
> plot(rcexp, xlab = 'Exp(1)', ylab = 'Exp(2)')
```



Simulation based on copulas

- Estimated density.

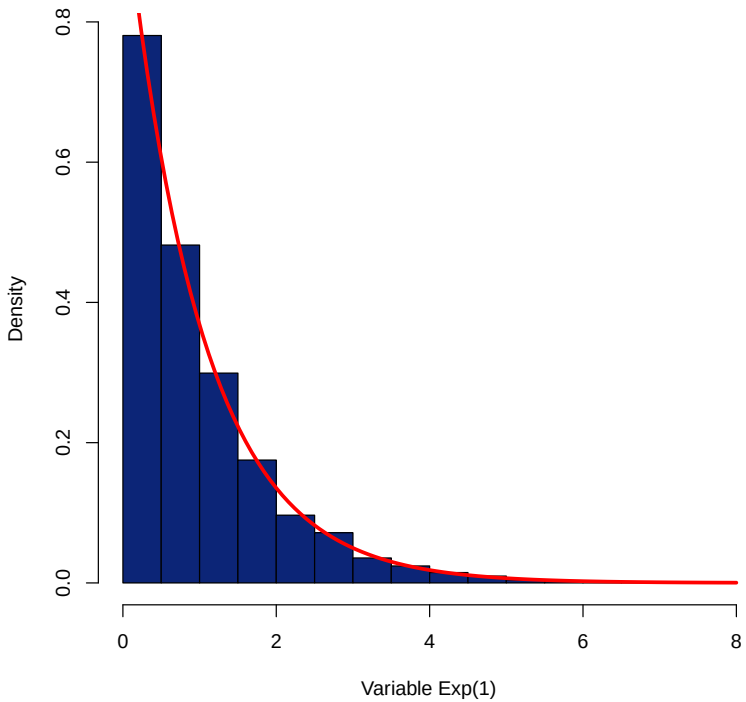
```
> library(sm)
> sm.density(rcexp)
```

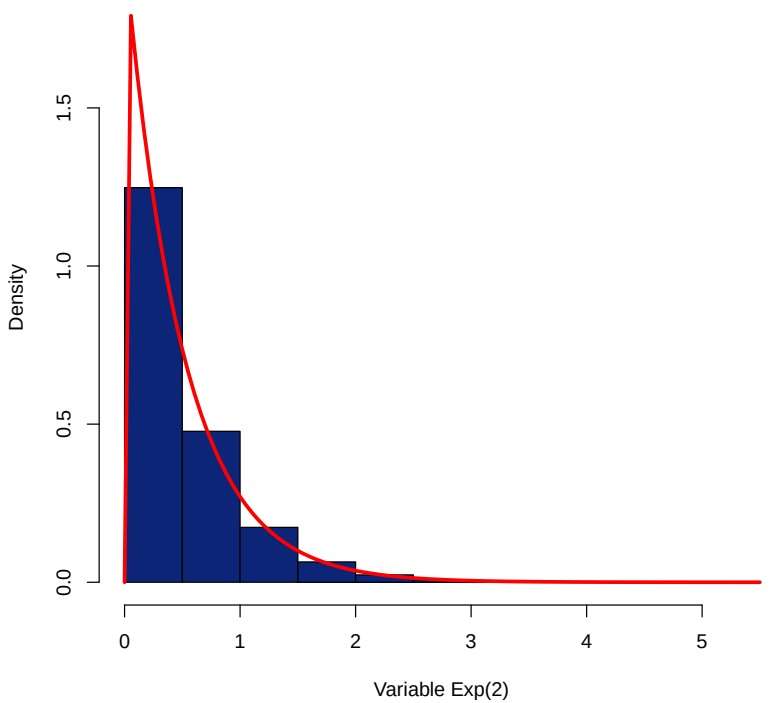


Simulation based on copulas

- Marginal distributions.

```
> hist(rcexp[,1], freq = FALSE,main='', xlab='Variable Exp(1)', col=4)
> curve(dexp(x,1), add = TRUE,lwd=3,col=2)
>
> hist(rcexp[,2], freq = FALSE,main='', xlab='Variable Exp(2)',col=4)
> curve(dexp(x,2), add = TRUE,lwd=3,col=2)
```





Discrete multivariate simulation

Idea: reduce the problem to simulate several one-dimensional discrete random variable.

Discrete multivariate simulation

Idea: reduce the problem to simulate several one-dimensional discrete random variable.

- Without dependence structure.
- With a certain dependence structure.

Conditional distributions

Let $\mathbf{X}$ a discrete d-dimensional random vector with joint probability function $p(x_1, \ldots, x_d)$, we can write

$$p(x_1, \ldots, x_d) = p_1(x_1)p_2(x_2|x_1) \ldots p_n(x_d|x_1, \ldots, x_{d-1}).$$

1. Generate X_1 from p_1.
2. From $i = 2...d$ generate X_i from $p_i(\cdot|X_1, \ldots, X_{i-1})$.
3. Return $(X_1, \ldots, X_d)$.

Remark! It is useful to note that $p_i(x_i|x_1, \ldots, x_{i-1}) = \frac{p_i(x_1,\ldots,x_i)}{p_{i-1}(x_1,\ldots,x_{i-1})}$.

Code-labelling methods

Build a function h (bijective) codifying all the possible d-plas of the range of the random vector, making a correspondence with a different natural number for each of them.

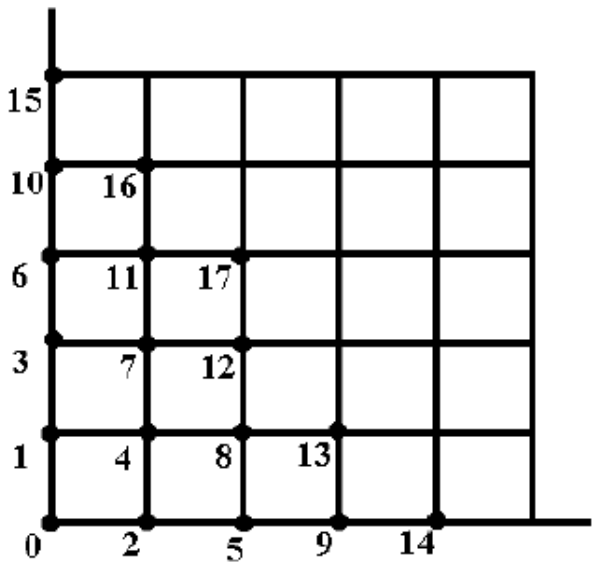
Example

$$\mathbf{X} = (X_1, X_2) \in \mathbb{Z}^+ \times \mathbb{Z}^+$$

$$h(i,j) = \frac{(i+j)(i+j+1)}{2} + i$$

Remark!

Computational cost of h^{-1}.



Part I: Simulation

Module III: Applications of Simulation

Application to the statistical inference

- Distribution of statistics or point estimators.
  - Distribution approximations.
    - Approximation of characteristics of the distribution.
    - Validity of the asymptotic distribution.
  - Comparison of estimators.
- Estimation base on confidence intervals.
  - Obtain confidence intervals.
  - Analysis of an estimator using confidence intervals.
- Hypothesis contrasts.
  - Approximation of the p-value.
  - Analysis of the hypothesis contrast.
- Bootstrap methods.

Application to the statistical inference

Example: Given $\{X_1, \ldots, X_n\}$ a random sample of a variable $X \sim N(\mu, \sigma)$, then the sampling distribution of the sample mean is

$$\widehat{\mu} = \overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \sim N\left(\mu, \frac{\sigma}{\sqrt{n}}\right).$$

Check this result using a simulation study.

Application to the statistical inference

Example: Given $\{X_1, \ldots, X_n\}$ a random sample of a variable $X \sim N(\mu, \sigma)$, then the sampling distribution of the sample mean is

$$\widehat{\mu} = \overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \sim N\left(\mu, \frac{\sigma}{\sqrt{n}}\right).$$

Check this result using a simulation study.

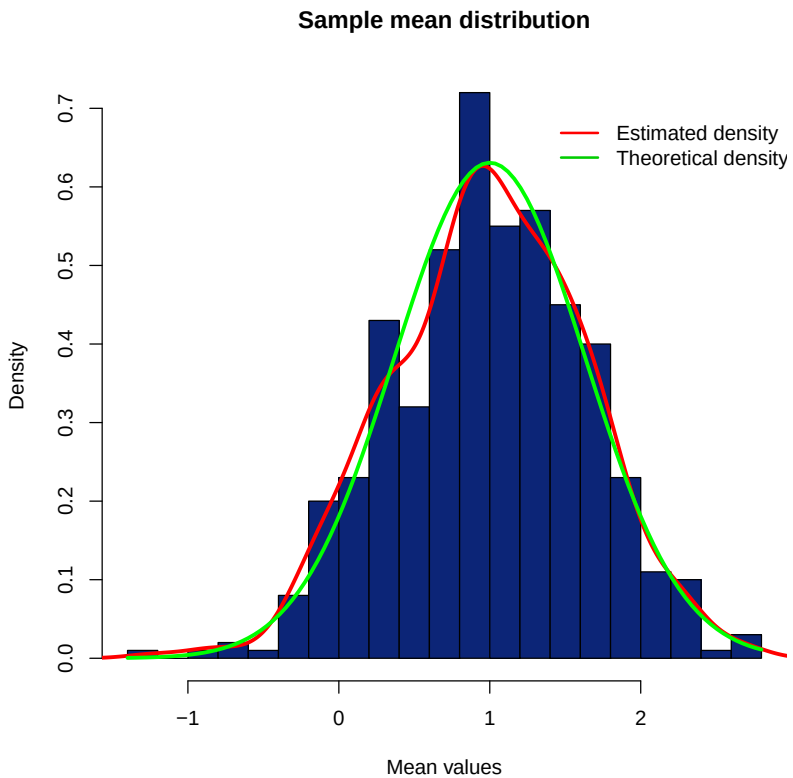
```
> set.seed(1234); nsim <- 500 ; nx <- 10; mux <- 1; sdx <- 2
> samples <- as.data.frame(matrix(rnorm(nsim*nx, mean=mux,
+    sd=sdx), ncol=nx))
> rownames(samples) <- paste('sample', 1:nsim, sep='')
> colnames(samples) <- paste('obs', 1:nx, sep='')
> samples$mean <- rowMeans(samples[,1:nx])
> samples$sd <- apply(samples[,1:nx], 1, sd)
> hist(samples$mean, freq=FALSE, breaks='FD', main='Sample mean
+    distribution', xlab='Mean values', ylab='Density', col=4))
> lines(density(samples$mean),lwd=3,col=2)
> curve(dnorm(x,mux,sdx/sqrt(nx)), lwd=3, col='green', add=TRUE)
```

Application to the statistical inference

Example: Given $\{X_1, \ldots, X_n\}$ a random sample of a variable $X \sim N(\mu, \sigma)$, then the sampling distribution of the sample mean is

$$\widehat{\mu} = \overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \sim N\left(\mu, \frac{\sigma}{\sqrt{n}}\right).$$

Check this result using a simulation study.



Monte Carlo Integration

Monte Carlo integration is design in order to compute multidimensional integrals as

$$I = \int \cdots \int h(x_1, \ldots, x_n) dx_1 \ldots dx_n.$$

Example: We are interested in computing

$$I = \int_a^b h(x)dx = (b-a)\int_a^b h(x)\frac{1}{b-a}dx.$$

If $x_1, x_2, \ldots, x_n$ represents a random sample that follows an $U(a,b)$ then

$$I = E(h(U(a,b))) \cdot (b-a) \simeq \frac{1}{n}\sum_{i=1}^{n} h(x_i) \cdot (b-a).$$

Monte Carlo Integration

Example: Estimate the integral $\int_0^1 4x^3 dx = 1$.

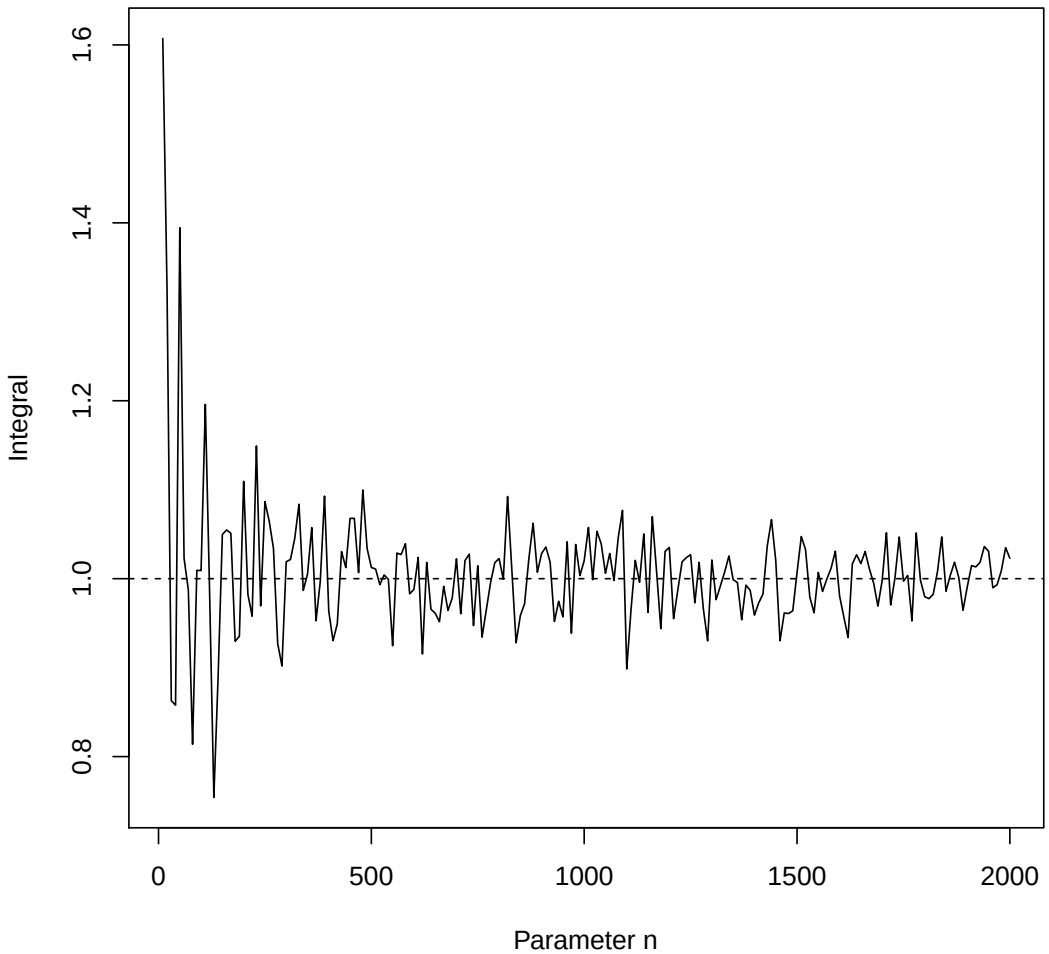
Monte Carlo Integration

Example: Estimate the integral $\int_0^1 4x^3 dx = 1$.

```
> mc.integral <- function(fun, a, b, n) {
+    x <- runif(n, a, b)
+    fx <- sapply(x, fun)
+    return(mean(fx) * (b - a))
+ }
> fun<-function(x) ifelse((x > 0)& (x < 1), 4*x^3, 0)
> set.seed(1234)
> mc.integral(fun, 0, 1, 100)
[1] 0.760426
> mc.integral(fun, 0, 1, 1000)
[1] 1.057621
> mc.integral(fun, 0, 1, 10000)
[1] 0.9964717
```

Monte Carlo Integration

Example: Estimate the integral $\int_0^1 4x^3 dx = 1$.



Importance sampling

Given $X \sim f$, we are interested in computing

$$\theta = E(h(X)) = \int h(x)f(x)dx.$$

Then, if $x_1, x_2, \ldots, x_n$ represents a random sample of X

$$\widehat{\theta} \simeq \frac{1}{n}\sum_{i=1}^{n} h(x_i).$$

Idea

In order to approximate the integral $\theta = E(h(X))$, sometimes, it could be simpler to generate observations from a density g that is similar to the product hf. That is, if $Y \sim g$

$$\theta = \int h(x)f(x)dx = \int \frac{h(x)f(x)}{g(x)}g(x)dx = E(q(Y)),$$

where $q(x) = \frac{h(x)f(x)}{g(x)}$.

Importance sampling

Procedure

- Given $Y_1, Y_2, \ldots, Y_n$ a random sample of $Y \sim g$ then

$$\theta \simeq \frac{1}{n}\sum_{i=1}^{n} q(Y_i) = \frac{1}{n}\sum_{i=1}^{n} w(Y_i)\, h(Y_i) = \widehat{\theta}_g$$

  where $w(x) = \frac{f(x)}{g(x)}$.

- In this case $\mathrm{Var}(\widehat{\theta}_g) = \mathrm{Var}(q(Y))/n$, and this variance can be reduced significantly with respect to the classical method if

$$g(x) \underset{\text{approx.}}{\propto} h(x)\, f(x).$$

How to select the density g?

Condition 1: The variance of the estimator $\widehat{\theta}_g$ should be finite to apply the central limit theorem, that is,

$$E(q^2(Y)) = \int \frac{h^2(x)\, f^2(x)}{g(x)} dx = \int h^2(x)\, \frac{f^2(x)}{g(x)} dx < \infty.$$

Condition 2: The density g should be more heavy-tails than the density f, because this could lead to estimators with finite variance.

Condition 3: The distribution of the weights $w(Y_i)$ should be homogeneous in order to avoid influential observations.

How to select the density g?

Condition 4: If f and/or g are quasi-densities, then the following approximation is used

$$\theta \simeq \frac{\sum_{i=1} w(Y_i)\, h(Y_i)}{\sum_{i=1} w(Y_i)},$$

in order to avoid the normalized constants.

Condition 5: If we simulated samples of $\{Y_1, Y_2, \ldots, Y_n\}$ weighted by $w(Y_i)$, then we will obtain an approximation of the density f.

Rubin, D. B. (1987).The calculation of posterior distributions by data augmentation: Comment: A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: The SIR algorithm. Journal of the American Statistical Association, 82(398), 543-546.

How to select the density g?

Example: Generate 1000 values from a standard Gaussian distribution using importance sampling methods based on 10^5 values from a Cauchy$(0, 1)$ distribution.

How to select the density g?

Example: Generate 1000 values from a standard Gaussian distribution using importance sampling methods based on 10^5 values from a Cauchy$(0, 1)$ distribution.

```
> nsim <- 10^3
> nsim2 <- 10^5
> set.seed(1234)
> y <- rcauchy(nsim2)
> w <- dnorm(y)/dcauchy(y)
>
> rx <- sample(y, nsim, prob = w/sum(w))
>
> hist(rx, freq = FALSE,col='gray', xlab='',
+    main='Histogram of the simulated data')
> curve(dnorm, add = TRUE,lwd=3,col=2)
```

How to select the density g?

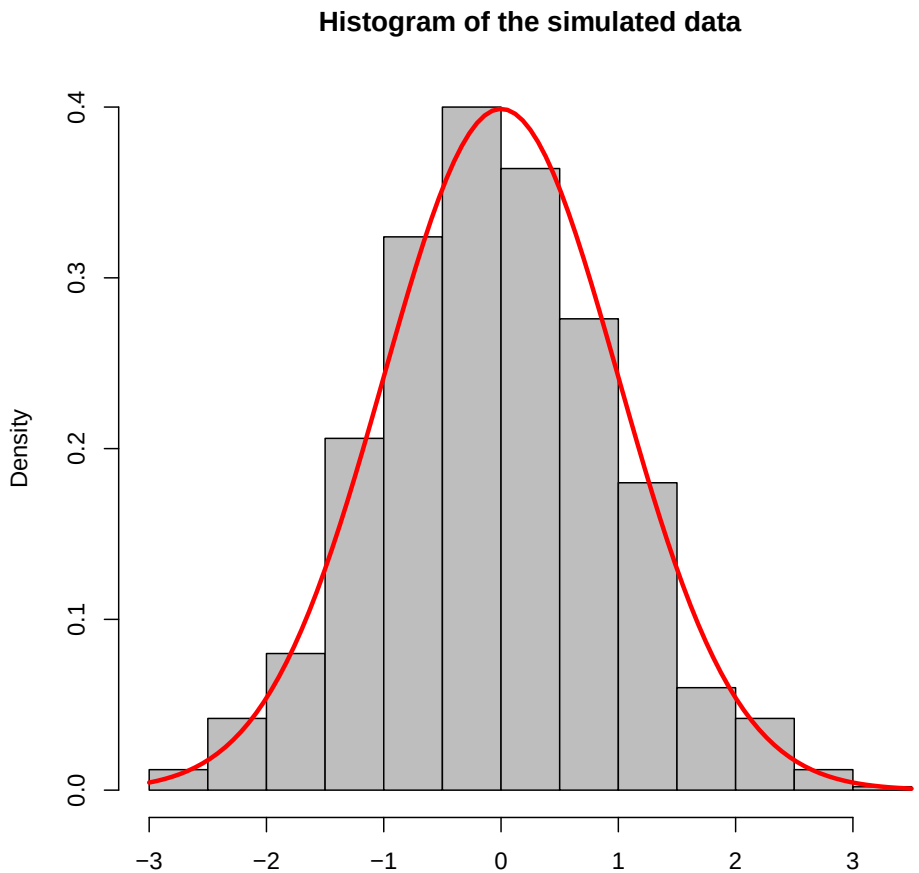
Example: Generate 1000 values from a standard Gaussian distribution using importance sampling methods based on 10^5 values from a Cauchy$(0, 1)$ distribution.

```
> nsim <- 10^3
> nsim2 <- 10^5
> set.seed(1234)
> y <- rcauchy(nsim2)
> w <- dnorm(y)/dcauchy(y)
>
> rx <- sample(y, nsim, prob = w/sum(w))
>
> hist(rx, freq = FALSE,col='gray', xlab='',
+    main='Histogram of the simulated data')
> curve(dnorm, add = TRUE,lwd=3,col=2)
```

nsim2 should be bigger than **nsim**.

How to select the density g?

Example: Generate 1000 values from a standard Gaussian distribution using importance sampling methods based on 10^5 values from a Cauchy$(0, 1)$ distribution.



Monte Carlo Optimization

Goal: we are interested in the following optimization problem

$$\min_{x \in D} f(x).$$

There exists several methods in the literature to solve this kind of problems like Newton-Raphson methods.

Original idea: We focus on looking for zeros of the first derivative of f using an iterative approximation

$$x_{i+1} = x_i - [Hf(xi)]^{-1} \nabla f(xi),$$

where $Hf(x_i)$ represents the Hessian matrix (second derivatives) and $\nabla f(xi)$ the gradient vector (first derivatives).

Monte Carlo Optimization

Problem: These methods works really well when the objective function do not have local minimums (i.e. maximum likelihood estimator, where the objective function can be multimodal). In other cases, the election of the initial point is crucial.

Alternative idea: we can try to generate random values in the following way: the regions where the objective function is lower were high probability and low probability the regions where the objective function is higher.

- Methods using random gradient.
- Simulated annealing.
- Genetic algorithms.
- ...

Simulated annealing (SA)

SA is based on physical/chemical processes referred to as tempering certain alloys of metal, glass, or crystal by heating above its melting point, holding its temperature, and then cooling it very slowly until it solidifies into a perfect crystalline structure.

SA is basically composed of two stochastic processes: one process for the generation of solutions and the other for the acceptance of solutions. The generation temperature is responsible for the correlation between generated probing solutions and the original solution.

SA is a descent algorithm modified by random ascent moves in order to escape local minima which are not global minima. The annealing algorithm simulates a nonstationary finite state Markov chain whose state space is the domain of the cost function to be minimized. Importance sampling is the main principle that underlies SA.

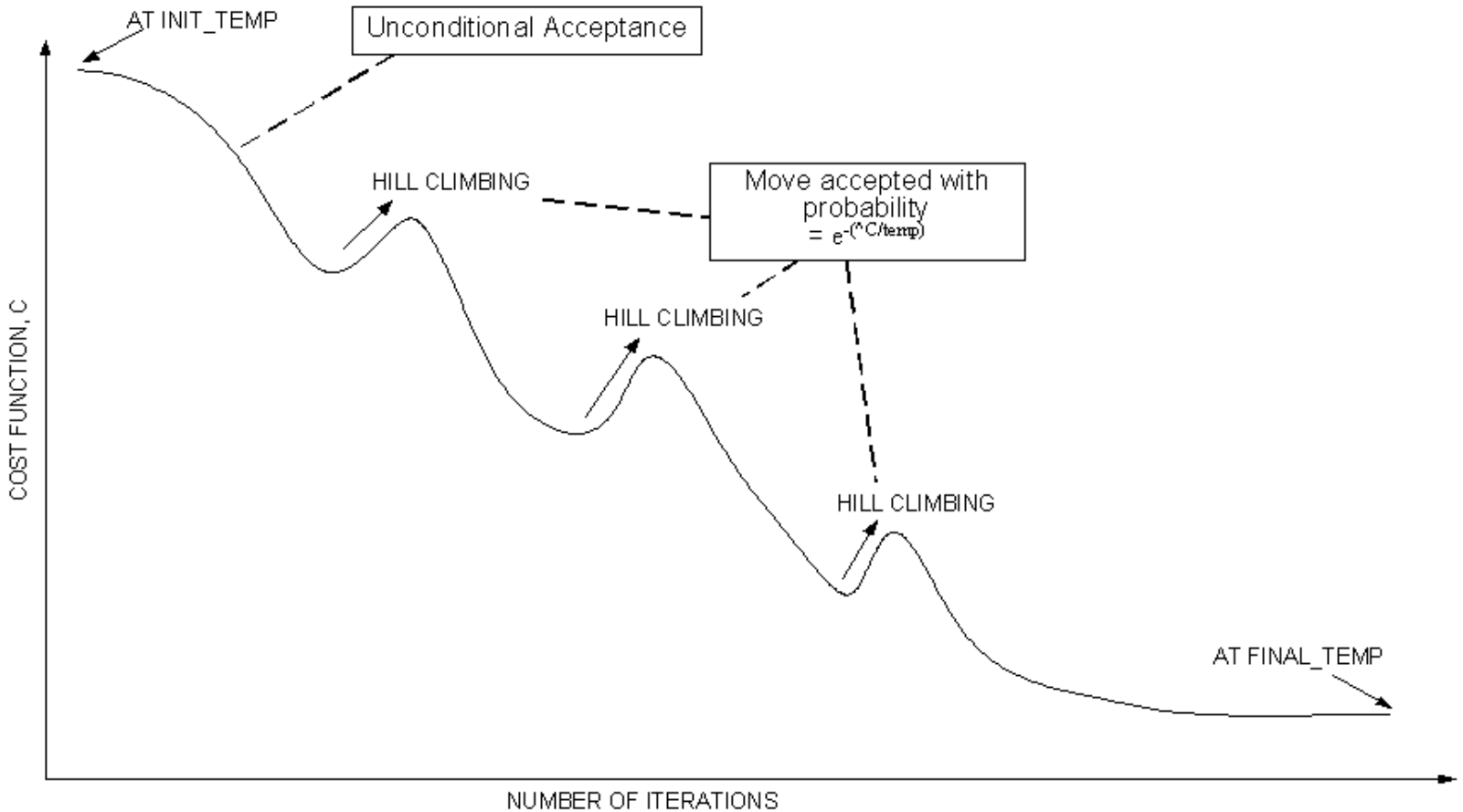
Simulated annealing (SA)

Algorithm

1. temp = init-temp;
2. place = init-placement;
3. while (temp > final-temp) do
4. while (inne_loop_criterion = FALSE) do
5. new_place = PERTURB(place,temp);
6. ∇C = COST(new_place) - COST(place);
7. if (∇C < 0) then
8. place = new_place;
9. else if (RANDOM(0,1)> e-(∇C/temp)) then
10. place = new_place;
11. temp = SCHEDULE(temp);

Robert, C. P., and Casella, G. (1999). The Metropolis-Hastings Algorithm. In Monte Carlo Statistical Methods (pp. 231-283). Springer, New York, NY.

Simulated annealing (SA)



http://www.ecs.umass.edu/ece/labs/vlsicad/ece665/slides/SimulatedAnnealing.ppt

Simulated annealing (SA)

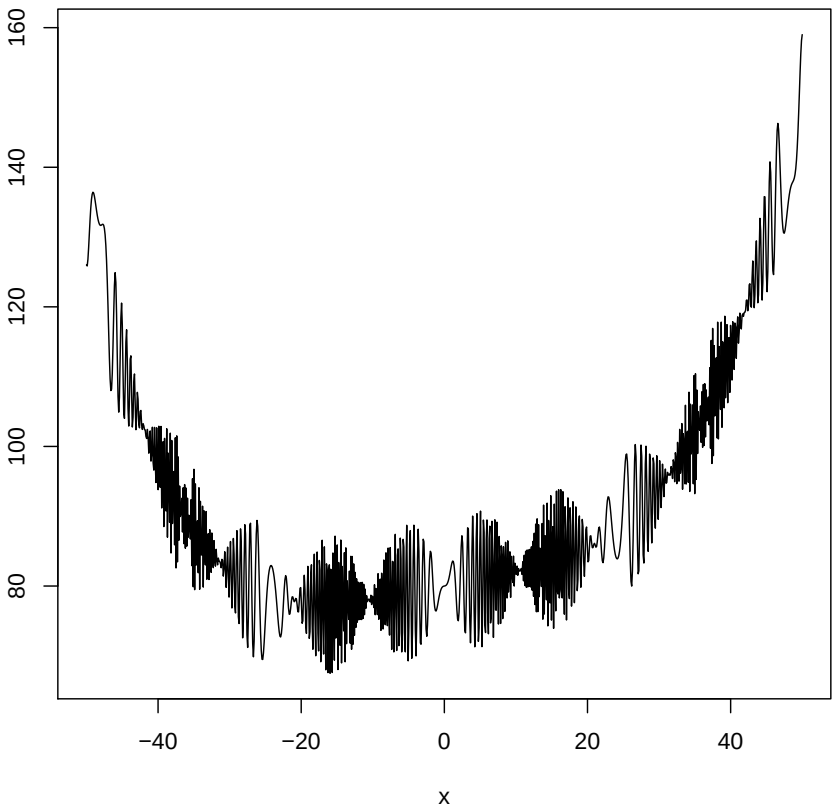
Example: Find the minimum of the function

$$f(x) = 10\sin(0.3x)\ \sin(1.3x^2) + 0.00001x^4 + 0.2x + 80.$$

Simulated annealing (SA)

Example: Find the minimum of the function

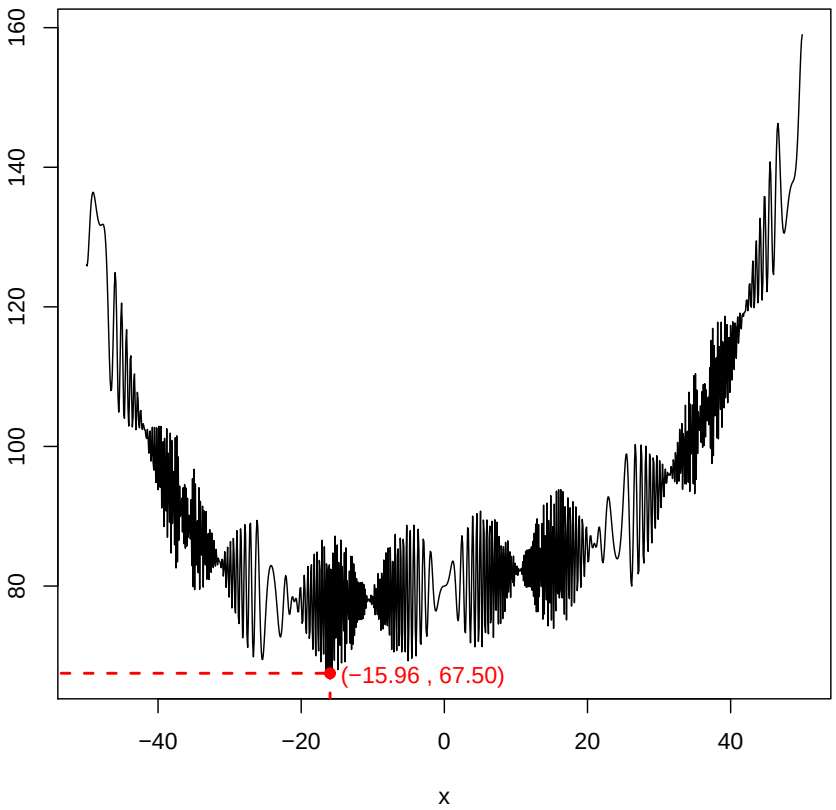
$$f(x) = 10\sin(0.3x)\ \sin(1.3x^2) + 0.00001x^4 + 0.2x + 80.$$



Simulated annealing (SA)

Example: Find the minimum of the function

$$f(x) = 10\sin(0.3x)\ \sin(1.3x^2) + 0.00001x^4 + 0.2x + 80.$$




```
> fw <- function (x){
+    10*sin(0.3*x)*sin(1.3*x^2)
+   + 0.00001*x^4 + 0.2*x+80
+ }
> optim(50, fw, method = "SANN",
+    control = list(maxit = 20000,
+    temp = 20, parscale = 20))
$par                    $value
[1] -15.81429           [1] 67.47401

$counts                 $convergence
function gradient       [1] 0
20000 NA

$message
NULL
```

Genetic algorithms

This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

- The **population** is $\{x_1, \ldots, x_n\}$. An individual is characterized by a set of parameters (variables) known as Genes. Genes are joined into a string to form a Chromosome (solution).
- The **fitness** function determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.
- The idea of **selection** phase is to select the fittest individuals and let them pass their genes to the next generation. Two pairs of individuals (parents) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.

Genetic algorithms

- **Crossover** is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes.
- In certain new offspring formed, some of their genes can be subjected to a **mutation** with a low random probability. This implies that some of the bits in the bit string can be flipped.
- The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation). Then it is said that the genetic algorithm has provided a set of solutions to our problem.

Remark! The R packages **DEOptim** and **gafit** implement this kind of methods.

Genetic algorithms

Algorithm:

1. START.
2. Generate the initial population.
3. Compute fitness.
4. REPEAT.
5. Selection.
6. Crossover.
7. Mutation.
8. Compute fitness.
9. UNTIL population has converged.
10. STOP.

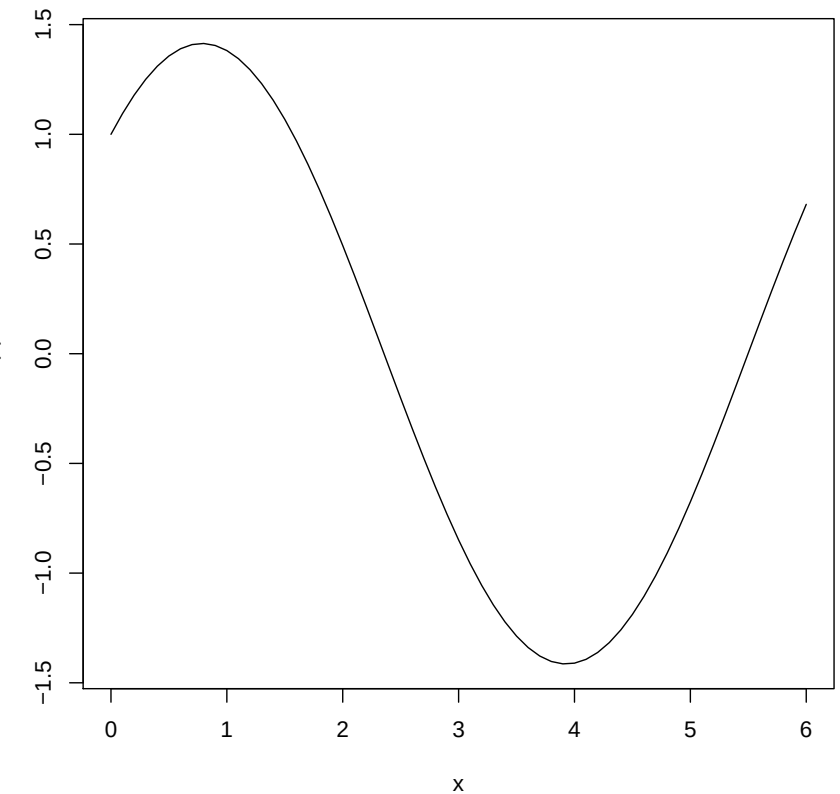
Mitchell, M. (1998). An Introduction to Genetic Algorithms. The MIT Press. ISBN 0262631857.

Genetic algorithms

Example: Find the mimimum of the function $f(x) = \cos(x) + \sin(x)$ on the interval $[0, 6]$.

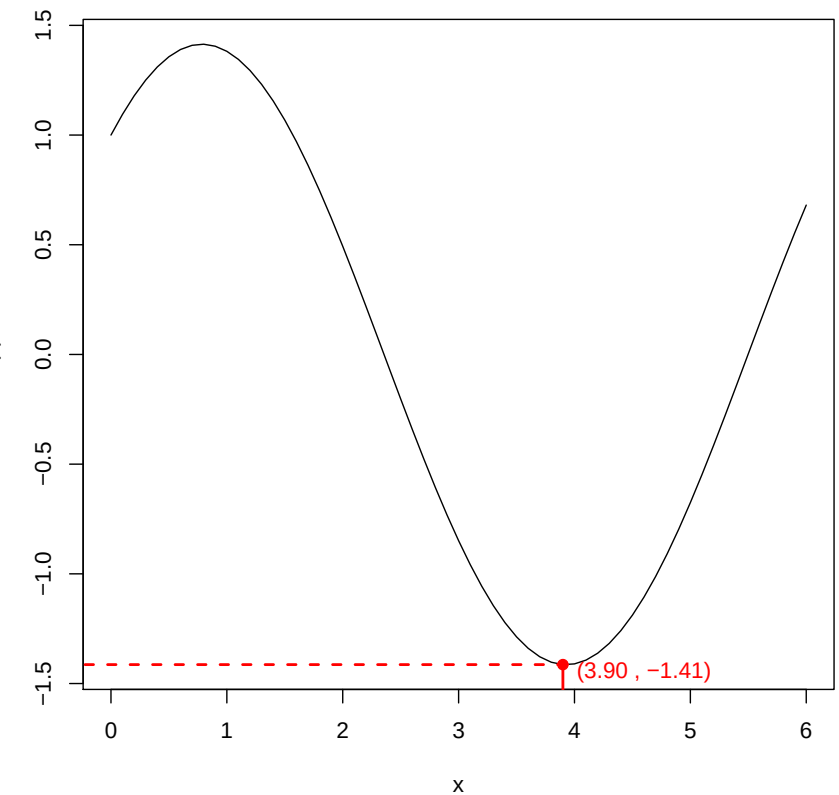
Genetic algorithms

Example: Find the mimimum of the function $f(x) = \cos(x) + \sin(x)$ on the interval $[0, 6]$.



Genetic algorithms

Example: Find the mimimum of the function $f(x) = \cos(x) + \sin(x)$ on the interval $[0, 6]$.




```
library(gafit)
> e<-expression(cos(theta)
+    +sin(theta))
> guess.1 <- list( theta=3 )
> guess.2 <- gafit( e, guess.1,
+    step=1e-3 )
> guess.2
$theta
[1] 3.927359
> gafit( e, guess.2, step=1e-5)
$theta
[1] 3.926993
```

Part I: Simulation

Module IV: Variance reduction techniques

Variance reduction

These techniques are used when we want to offer the most precise answers possible with low computational cost.

Never use these techniques if we are trying to **estimate the variability**.

There are a lot of different procedures, we will detail the following:

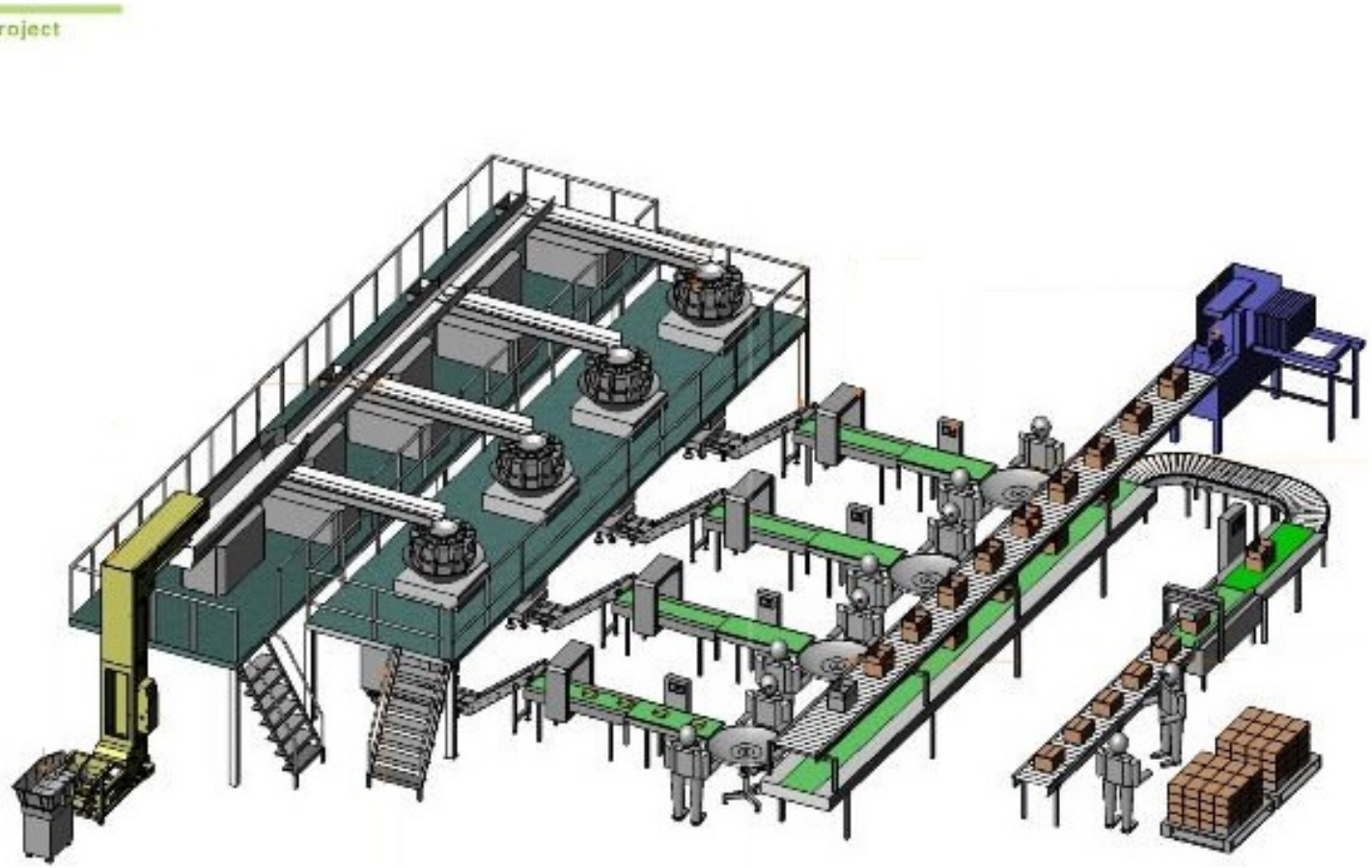
- Common random numbers.
- Antithetic variables.
- Stratified sampling.
- Control variables.
- Conditioning.

Var reduction - Common random numbers

Idea: we have two different procedures to perform the same tasks and we want to compare them, for example, in terms of mean time.

Var reduction - Common random numbers

Idea: we have two different procedures to perform the same tasks and we want to compare them, for example, in terms of mean time.



Var reduction - Common random numbers

Obective: we have changed the production system at a certain point and we want to compare the new method with the old one, for example, in terms of mean time.

- T_{NM} time for processing one product with the new method.
- T_{OM} time for processing one product with the old method.
- Both are random variables depending on several elements.
- For simplification: imagine the final time depends on the arriving times of the products to process.

Var reduction - Common random numbers

Obective: we have changed the production system at a certain point and we want to compare the new method with the old one, for example, in terms of mean time.

- T_{NM} time for processing one product with the new method.
- T_{OM} time for processing one product with the old method.
- Both are random variables depending on several elements.
- For simplification: imagine the final time depends on the arriving times of the products to process.

$$\theta = E[T_{OM}] - E[T_{NM}] \Rightarrow \widehat{\theta} = \overline{T}_{OM} - \overline{T}_{NM}$$

Shall we generate the two vectors of arriving times of the products independently or not?

Let compute the variance of the statistic of interest:

$$Var(\widehat{\theta}) = Var(\overline{T}_{\text{OM}}) + Var(\overline{T}_{\text{NM}}) - 2Cov(\overline{T}_{\text{OM}}, \overline{T}_{\text{NM}}).$$

If we generate the sequences independently, then $Cov(\overline{T}_{\text{OM}}, \overline{T}_{\text{NM}}) = 0$, and

$$Var(\overline{T}_{\text{OM}} - \overline{T}_{\text{NM}}) = Var(\overline{T}_{\text{OM}}) + Var(\overline{T}_{\text{NM}}).$$

If we generate the sequences with $Cov(\overline{T}_{\text{OM}}, \overline{T}_{\text{NM}}) > 0$, then

$$Var(\widehat{\theta})_{\text{dependence}} \leq Var(\widehat{\theta})_{\text{independence}}$$

Var reduction - Common random numbers

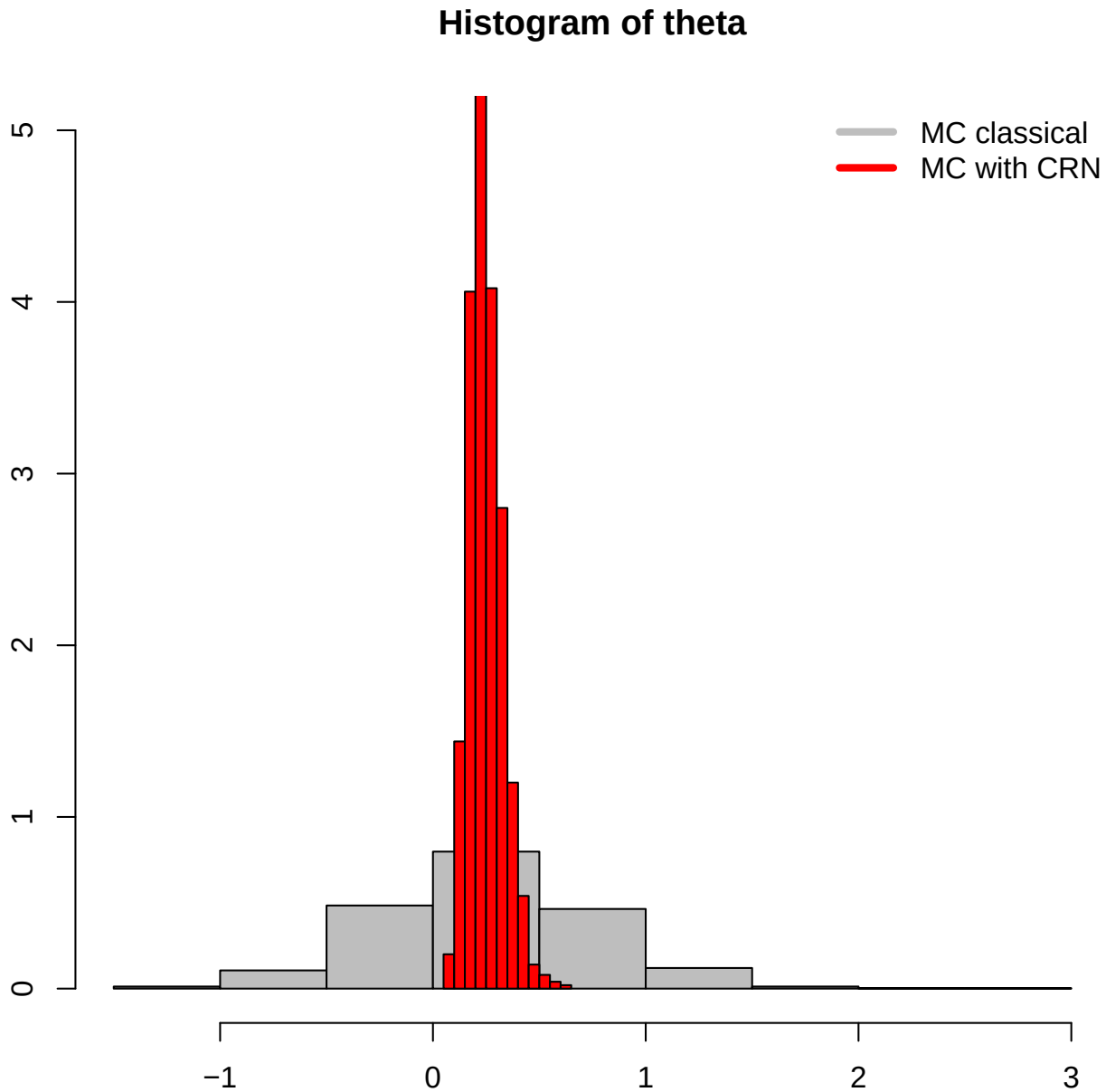

```
> nsim<-1000 #number of simulations
> N<-10 #number of products to be processed
> #New method: the processing times are exponential with rate=1
> u<-runif(N);t1<-qexp(u,rate=1)
> #Old method: the processing times are exponential with rate=0.8
> v<-runif(N);t2<-qexp(v,rate=0.8);theta<-mean(t2)-mean(t1)
> #Simulate NOT using common random numbers
> theta.not<-numeric(nsim)
> for(i in 1:nsim){
+     u<-runif(N);t1<-qexp(u,rate=1)
+     v<-runif(N);t2<-qexp(v,rate=0.8)
+     theta.not[i]<-mean(t2)-mean(t1)}
> #Simulate using common random numbers
> theta.yes<-numeric(nsim)
> for(i in 1:nsim){
+     u<-runif(N)
+     t1<-qexp(u,rate=1);t2<-qexp(u,rate=0.8)
+     theta.yes[i]<-mean(t2)-mean(t1)}
> #Summary of the results
> m<-c(mean(theta.not),mean(theta.yes))
> d<-c(sd(theta.not),sd(theta.yes))
> res<-cbind(m,d);colnames(res)<-c('Mean','Std Dev')
> rownames(res)<-c('Not CRN','CRN')
> res
```

Var reduction - Common random numbers


```
> nsim<-1000 #number of simulations
> N<-10 #number of products to be processed
> #New method: the processing times are exponential with rate=1
> u<-runif(N);t1<-qexp(u,rate=1)
> #Old method: the processing times are exponential with rate=0.8
> v<-runif(N);t2<-qexp(v,rate=0.8);theta<-mean(t2)-mean(t1)
> #Simulate NOT using common random numbers
> theta.not<-numeric(nsim)
> for(i in 1:nsim){
+     u<-runif(N);t1<-qexp(u,rate=1)
+     v<-runif(N);t2<-qexp(v,rate=0.8)
+     theta.not[i]<-mean(t2)-mean(t1)}
> #Simulate using common random numbers
> theta.yes<-numeric(nsim)
> for(i in 1:nsim){
+     u<-runif(N)
+     t1<-qexp(u,rate=1);t2<-qexp(u,rate=0.8)
+     theta.yes[i]<-mean(t2)-mean(t1)}
> #Summary of the results
> m<-c(mean(theta.not),mean(theta.yes))
> d<-c(sd(theta.not),sd(theta.yes))
> res<-cbind(m,d);colnames(res)<-c('Mean','Std Dev')
> rownames(res)<-c('Not CRN','CRN')
> res
```

	Mean	Std Dev
Not CRN	0.2447	0.5150
CRN	0.2479	0.0774

Var reduction - Common random numbers



Variance reduction - Antithetic variables

Problem: we want study a quantity $\theta = h(X_1, \ldots, X_n)$ of our population distributed following F.

Variance reduction - Antithetic variables

Problem: we want study a quantity $\theta = h(X_1, \ldots, X_n)$ of our population distributed following F.

Initial idea: we simulate a sample from our population (needs to be known) using one of the methods for simulation previously explained, for example, inverse method:

1. Generate $U_1 \sim U(0, 1)$.
2. Compute $X_1 = F^{-1}(U_1)$.
3. Return X_1.
4. Repeat Step 1-3 M times.

Variance reduction - Antithetic variables

Problem: we want study a quantity $\theta = h(X_1, \ldots, X_n)$ of our population distributed following F.

Initial idea: we simulate a sample from our population (needs to be known) using one of the methods for simulation previously explained, for example, inverse method:

1. Generate $U_1 \sim U(0,1)$.
2. Compute $X_1 = F^{-1}(U_1)$.
3. Return X_1.
4. Repeat Step 1-3 M times.

Introduction of antithetic variables: to reduce the variance we can generate a sample with double size of the previous one in the following way:

1. Generate $U_1 \sim U(0,1)$.
2. Compute $X_1 = F^{-1}(U_1)$.
3. Compute $X_2 = F^{-1}(1 - U_1)$ instead of using a new $U_2 \sim U(0,1)$.
4. Return X_1 and X_2.
5. Repeat Step 1-4 M/2 times.

Variance reduction - Antithetic variables

Justification: we want to estimate by simulation a quantity $\theta = h(X_1, \ldots, X_n)$, for example, the sample mean.

We generate n pairs $(X_1, Y_1), \ldots, (X_n, Y_n)$ from $X \sim F$ and $Y \sim F$.

The combined estimator $\widehat{\theta} = \frac{\overline{X}+\overline{Y}}{2}$ is our proposal of statistic where

$$Var(\widehat{\theta}) = \frac{1}{4}\left(Var(\overline{X}) + Var(\overline{Y}) + 2Cov(\overline{X}, \overline{Y})\right) = \frac{\sigma^2}{2n}(1 + \rho(\overline{X}, \overline{Y})),$$

with ρ denoting the correlation.

Then, with a negative correlation or covariance between X and Y, we obtain a **reduction of** $-100\rho(\overline{X}, \overline{Y})\%$ using a $2n$ size sample with antithetic variables instead of a $2n$ size sample in the classical way.

Variance reduction - Antithetic variables

Example: dice toss

```
> nsim<-5000 #number of simulations
#Naive calculation
> toss <- runif(nsim) < 1/6
> mu=mean(toss);SE=sd(toss)
> print(c(Mean=mu,SE=SE))
   Mean      SE
0.1660000 0.3721179
#Antithetic variates
> unif.vec<-runif(nsim/2)
#Normal process
> x_1<- unif.vec<1/6
#Antithetic process
> x_2<- (1-unif.vec)<1/6
> toss.av <- (x_1 + x_2)/2
> mu=mean(toss.av);SE=sd(toss.av)
> print(c(Mean=mu,SE=SE))
   Mean      SE
0.1730000 0.2378942
```

Variance reduction - Antithetic variables

Exercise

Use this technique to approximate with MC $I = \int_a^b g(x)dx$ and apply it to approximate for example $E(e^{U(0,2)}) = \int_0^2 \frac{1}{2} e^x dx$.

Variance reduction - Stratifying sampling

Idea: divide the population in strata and obtain our sample with a number of observations in each stratum (proportional to their probability). We need to assure that the domain is covered.

Problem: we want to estimate by simulation a quantity $\theta = E(X)$ and suppose there is a discrete r.v. Y with values y_i and probabilities p_i where $i = 1, \ldots, k$ such that

- $p_i = P(Y = y_i)$ with $i = 1, \ldots, k$ are known.
- for each i we can simulate $X|Y = y_i$.

Variance reduction - Stratifying sampling

Simple random sampling

- $E(X)$.
- $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$.
- $Var(\overline{X}) = \frac{1}{n} Var(X)$.

Stratifying sampling

- $E(X) = \sum_{i=1}^{k} E(X|Y = y_i)p_i$.
- $\varepsilon_{\overline{X}} = \sum_{i=1}^{k} \overline{X}_i p_i$
  with $\overline{X}_i$ is the average of np_i values of X generated conditional on $Y = y_i$.
- $Var(\varepsilon_{\overline{X}}) = \sum_{i=1}^{k} p_i^2 Var(\overline{X}_i)) = \frac{1}{n} E(Var(X|Y))$.

Variance reduction - Stratifying sampling

Simple random sampling

- $E(X)$.
- $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$.
- $Var(\overline{X}) = \frac{1}{n} Var(X)$.

Stratifying sampling

- $E(X) = \sum_{i=1}^{k} E(X|Y = y_i)p_i$.
- $\varepsilon_{\overline{X}} = \sum_{i=1}^{k} \overline{X}_i p_i$
  with $\overline{X}_i$ is the average of np_i values of X generated conditional on $Y = y_i$.
- $Var(\varepsilon_{\overline{X}}) = \sum_{i=1}^{k} p_i^2 Var(\overline{X}_i)) = \frac{1}{n} E(Var(X|Y))$.

$$Var(X) = E(Var(X|Y)) + Var(E(X|Y))$$

$$\Downarrow$$

$$Var(\overline{X}) - Var(\varepsilon_{\overline{X}}) = \frac{1}{n} Var(E(X|Y)$$

Variance reduction - Stratifying sampling

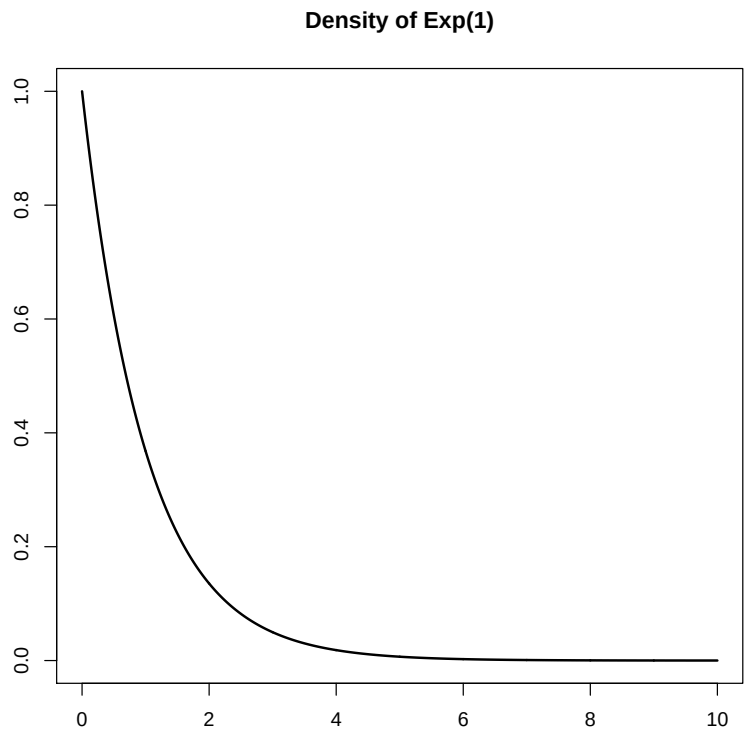
Example

We want to approximate the theoretical mean of $X \sim Exp(1)$ with a sample of size $n = 10$ using 3 strata: 40% of the smaller values, 50% of the intermediate and 10% highest.

Variance reduction - Stratifying sampling

Example

We want to approximate the theoretical mean of $X \sim Exp(1)$ with a sample of size $n = 10$ using 3 strata: 40% of the smaller values, 50% of the intermediate and 10% highest.



Variance reduction - Stratifying sampling

Example

We want to approximate the theoretical mean of $X \sim Exp(1)$ with a sample of size $n = 10$ using 3 strata: 40% of the smaller values, 50% of the intermediate and 10% highest.

Recall: how to simulate Exp(1)

1. $U \sim U(0, 1)$.
2. $X = -\ln(U)$.

Variance reduction - Stratifying sampling

Example

We want to approximate the theoretical mean of $X \sim Exp(1)$ with a sample of size $n = 10$ using 3 strata: 40% of the smaller values, 50% of the intermediate and 10% highest.

Step 1: use stratifying sampling

1. Simulate 4 values $U[0.6, 1)$ for the first stratum.
2. Simulate 5 values $U[0.1, 0.6)$ for the second stratum.
3. Simulate 1 value $U[0, 0.1)$ for the third stratum.

Variance reduction - Stratifying sampling

Example

We want to approximate the theoretical mean of $X \sim Exp(1)$ with a sample of size $n = 10$ using 3 strata: 40% of the smaller values, 50% of the intermediate and 10% highest.

Step 2: algorithm

For $i = 1, \ldots, 10$

1. Generate U_i
   - 1a. $U \sim U(0,1)$.
   - 1b. If $i \leq 4$ compute $U_i = 0.4U + 0.6$.
   - 1c. If $4 < i \leq 9$ compute $U_i = 0.5U + 0.1$.
   - 1d. If $i = 10$ compute $U_i = 0.1U$.
2. $X_i = -\ln(U_i)$

$$Var(\overline{X}) = \frac{1}{10^2} \sum_{i=1}^{10} Var(X_i) = 0.022338$$

Variance reduction - Stratifying sampling

Example

We want to approximate the theoretical mean of $X \sim Exp(1)$ with a sample of size $n = 10$ using 3 strata: 40% of the smaller values, 50% of the intermediate and 10% highest.

Step 2: algorithm

For $i = 1, \ldots, 10$

1. Generate U_i
   - 1a. $U \sim U(0,1)$.
   - 1b. If $i \leq 4$ compute $U_i = 0.4U + 0.6$.
   - 1c. If $4 < i \leq 9$ compute $U_i = 0.5U + 0.1$.
   - 1d. If $i = 10$ compute $U_i = 0.1U$.
2. $X_i = -\ln(U_i)$

Variance in simple random sampling

$$Var(\overline{X}) = \frac{1}{10^2} \sum_{i=1}^{10} Var(X_i) = 0.022338 \leq \mathbf{1}$$

Variance reduction - Control variables

We want to estimate by simulation a quantity $\theta = E(X)$. Suppose we have another variable Y which mean μ_Y is known.

$T = X + c(Y - \mu_Y)$ is an unbiased estimator of θ.

Variance reduction - Control variables

We want to estimate by simulation a quantity $\theta = E(X)$. Suppose we have another variable Y which mean μ_Y is known.

$T = X + c(Y - \mu_Y)$ is an unbiased estimator of θ.

c?

Variance reduction - Control variables

We want to estimate by simulation a quantity $\theta = E(X)$. Suppose we have another variable Y which mean μ_Y is known.

$T = X + c(Y - \mu_Y)$ is an unbiased estimator of θ.

c? $\min_c Var(X + c(Y - \mu_Y)) = \min_c Var(X) + c^2 Var(Y) + 2cCov(X, Y)$,

$$c^* = \frac{-Cov(X, Y)}{Var(Y)} \Rightarrow Var(T) = Var(X)\left(1 - \rho^2(X, Y)\right).$$

Variance reduction - Control variables

We want to estimate by simulation a quantity $\theta = E(X)$. Suppose we have another variable Y which mean μ_Y is known.

$T = X + c(Y - \mu_Y)$ is an unbiased estimator of θ.

c? $\min_c Var(X + c(Y - \mu_Y)) = \min_c Var(X) + c^2 Var(Y) + 2cCov(X, Y)$,

$$c^* = \frac{-Cov(X, Y)}{Var(Y)} \Rightarrow Var(T) = Var(X)\left(1 - \rho^2(X, Y)\right).$$

Y is called a **control variable** and the use of it provides with a reduction in the variance of the $100\rho^2(X, Y)\%$.
(not known in advance, computed through simulation)

How do we compute c in practice?

Adjust a linear regression model of X over Y with the simulated data (X_i, Y_i), we get $\widehat{x} = \widehat{\beta}_0 + \widehat{\beta}_1 y$, and then

$$\widehat{\beta}_0 = -c^* \mu_Y \text{ and } \widehat{\beta}_1 = -c^*.$$

Hence,

$$c^* = -\widehat{\beta}_1.$$

How do we compute c in practice?

Adjust a linear regression model of X over Y with the simulated data (X_i, Y_i), we get $\widehat{x} = \widehat{\beta}_0 + \widehat{\beta}_1 y$, and then

$$\widehat{\beta}_0 = -c^* \mu_Y \text{ and } \widehat{\beta}_1 = -c^*.$$

Hence,

$$c^* = -\widehat{\beta}_1.$$

Additionally, to approximate θ we can proceed as follows

$$\widehat{\theta} = \overline{T} = \overline{X} - \widehat{\beta}_1(\overline{Y} - \mu_Y) = \widehat{\beta}_0 + \widehat{\beta}_1 \mu_Y.$$

Variance reduction - Control variables

Approximate the mean of $e^{U(0,2)}$ using control variables

```
> a <- 0; b <- 2; mean.theo <- (exp(b)-exp(a))/(b-a)
> mean.theo
[1] 3.194528 #Theoretical mean
#Classical Monte Carlo simulation approximation
> set.seed(54321);nunif<- 10000; u <- runif(nunif, a, b)
> expu <- exp(u)
> theta1<-mean(expu);theta1
[1] 3.200529
#With control variable
> mod <- lm(expu ~ u)
> mod.coef=mod$coef
> theta2<-mod.coef[1]+mod.coef[2]*1;theta2
[1] 3.193739
#estimation of the variance reduction
> expuc <- expu - mod.coef[2]*(u-1)
> 100*(var(expu)-var(expuc))/var(expu)
[1] 93.8147
```

Variance reduction - Control variables

Approximate the mean of $e^{U(0,2)}$ using control variables

```
#simulation proof of the variance reduction
> nsim<-1000
> theta1<-numeric(nsim);theta2<-numeric(nsim)
> for(i in 1:nsim){
+    u <- runif(nunif, a, b);expu <- exp(u)
+    theta1[i]<-mean(expu)
+    mod <- lm(expu ~ u);mod.coef=mod$coef
+    theta2[i]<-mod.coef[1]+mod.coef[2]*1}
#std dev approximation without control variable
> sd(theta1)
[1] 0.01683754
> sd(theta2)
#std dev approximation with control variable
[1] 0.00438511
```

Variance reduction - Conditioning

We want to estimate by simulation a quantity $\theta = E(X)$.

We select Y r.v. such that $E(X|Y)$ takes on a value that can be determine by simulation.

Variance reduction - Conditioning

We want to estimate by simulation a quantity $\theta = E(X)$.

We select Y r.v. such that $E(X|Y)$ takes on a value that can be determine by simulation.

Justification:

$$Var(X) = E(Var(X|Y)) + Var(E(X|Y))$$

$$\Downarrow$$

$$Var(X) \geq Var(E(X|Y)).$$

Variance reduction - Conditioning

We want to estimate by simulation a quantity $\theta = E(X)$.

We select Y r.v. such that $E(X|Y)$ takes on a value that can be determine by simulation.

Justification:

$$Var(X) = E(Var(X|Y)) + Var(E(X|Y))$$

$$\Downarrow$$

$$Var(X) \geq Var(E(X|Y)).$$

$E(X|Y)$ is an unbiased estimator of θ with less variance.

Variance reduction - Conditioning

Example

Approximate the value of π using the variable $X = \begin{cases} 1 & V_1^2 + V_2^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$

where V_1 and V_2 follow a $U(-1, 1)$.

First of all we need to realise that $E(X) = \pi/4$, so we are interested on estimating the mean of X.

Option 1: traditional Monte Carlo approximation

1. Generate $X_i \sim X$.
2. Repeat Step 1 M times.
3. Approximate the mean of our variable by $\frac{X_1 + \ldots + X_M}{M}$.

Variance reduction - Conditioning

Option 2: Monte Carlo approximation with conditioning

We can derive that $E(X|V_1) = (1 - V_1^2)^{1/2}$.

This estimator verifies that $E((1 - V_1^2)^{1/2}) = \pi/4$. Moreover,

$$Var(X) = \frac{\pi}{4}\left(1 - \frac{\pi}{4}\right) \simeq 0.1686,$$

$$Var((1 - V_1{}^2)^{1/2}) = \frac{2}{3} - \left(\frac{\pi}{4}\right)^2 \simeq 0.0498.$$

1. Generate $V_i \sim V_1$.
2. Compute $T_i = (1 - V_i^2)^{1/2}$.
3. Repeat Step 1 and Step 2 M times.
4. Approximate the mean of I by $\frac{T_1 + \ldots + T_M}{M}$.

Variance reduction - Conditioning

Example of π approximation

```
#Traditional MC approximation
> M=10000
> v1=runif(M,-1,1)
> v2=runif(M,-1,1)
> aux=v1^2+v2^2
> aux[aux<=1]=1
> aux[aux>1]=0
> X=mean(aux)*4
 [1] 3.1348
#MC approximation with conditioning
> T=(1-v1^2)^0.5
> mean(T)*4
 [1] 3.132865
```

Variance reduction - Conditioning

Example of π approximation

```
#proof through simulation of the variance reduction
> nsim<-10000
> X<-numeric(nsim);T<-numeric(nsim)
> for (i in 1:nsim){
+    v1=runif(M,-1,1);v2=runif(M,-1,1)
+    aux=v1^2+v2^2
+    aux[aux<=1]=1;aux[aux>1]=0
+    X[i]=mean(aux)*4
+    T[i]=mean((1-v1^2)^0.5)*4}
#estimation of the variance classical MC
> sd(X)
[1] 0.01655588
#estimation of the variance conditioning
> sd(T)
[1] 0.008933938
```

Summary of R functions for simulation

Discrete distributions

```
rbinom(n,size,prob) #Ber(prob) or Bi(size,prob)
rnbinom(n,size,prob) #NB(size,prob) or Ge(prob)
rpois(n,lambda) #Pois(λ)
rmultinom(n,size,prob) #Multinom(size,prob)
```

Continuous distributions

```
runif(n,min,max) #U(min,max)
rnorm(n,mu,sigma) #N(μ,σ)
rexp(n,rate) #Exp(1/rate)
rgamma(n,shape,rate=1,scale=1/rate) #Gamma(shape,scale)
rbeta(n,shape1,shape2) #Be(shape1,shape2)
rweibull(n,shape,scale) #We(shape,scale)
library(mvtnorm); rmvnorm(n,mu,sigma) #N_d(μ,Σ)
library(mnormt); rmt(n,mean,S=df Σ/(df − 2) #Multiv T-Stud
```

Part II: Resampling

Part II: Resampling

Module I: Introduction to the bootstrap. Uniform bootstrap

Introduction to the bootstrap

Definition

Statistical procedure that try to **model a population** through resampling on just **one available sample**.

Scenario

$(X_1, \ldots, X_n)$ a sample, we want to analyse different characteristics of a statistic $T \equiv T(X_1, \ldots, X_n)$.

Introduction to the bootstrap

Real world

- $\boldsymbol{X} = (X_1, \ldots, X_n)$ independent random sample with distribution F.
- We are interested on making inference on $\theta = \theta(F)$.
- We want to know the distribution $T(\boldsymbol{X})$.
- Sometimes this distribution can be computed exactly.
- Others, we can only approximate it when $n \to \infty$.

Bootstrap world

- Theoretical distribution (unknown) is replaced by an estimation $\widehat{F}$.
- We obtain, conditional on the original sample, resamples $\boldsymbol{X}^* = (X_1^*, \ldots, X_n^*)$.
- We consider the distribution of $T^* = T(\boldsymbol{X}^*)$.
- We approximate the distribution of T by the distribution of T^*.
- Rarely, the bootstrap distribution is directly computed, but it used to be approximated using Monte Carlo methods.

General algorithm

1. Select a sample $\boldsymbol{X} = (X_1, \ldots, X_n)$.
2. Construct an estimator of the distribution function $\widehat{F}$ (many possibilities).
3. Draw a sample from $\widehat{F}$, $\boldsymbol{X}^* = (X_1^*, \ldots, X_n^*)$ (resample).
4. Calculate the statistic of interest $\widehat{T}^* = \widehat{T}(X_1^*, \ldots, X_n^*)$.
5. Compute the exact distribution of the statistic in the bootstrap world.

5.* Repeat Step 3 and Step 4 B times, where B is as large as possible, $B = 500, 1000, \ldots$

6. Analyse the distribution of the statistic of interest in the bootstrap world $\widehat{T}_1^*, \ldots, \widehat{T}_n^*$.

Introduction to the bootstrap

Advantages

- No hypothesis on the unknown distribution (unknown population).
- Simplicity.
- Adaptability.
- Reduction of the field work and costs.

Drawbacks

- Based on independent samples.
- Computational cost.
- Computers have built-in error.
- Large sample sizes are generated.

A precedent to the bootstrap: the Jackknife

- First introduced by Quenouille in 1949 and named by Tukey in 1958.
- Is was used not to approximate the distribution of a certain statistic but some of its characteristics; mainly bias and variance.
- The resamples are obtained from the original one by taking out one value.
- In the Jackknife framework there are n different resamples while in the bootstrap there are $\binom{2n-1}{n}$.


Tukey, J. (1958) Bias and confidence in not quite large samples. *Annals of Mathematical Statistics*, 29, 614


Quenouille, M. (1949) Approximate test of correlation in time series. *Journal of the Royal Statistical Society: Series B*, 11, 18-84.

Uniform bootstrap (naïve bootstrap)

$\widehat{F}$ is done by the **empirical distribution function**

$$F_n = \frac{1}{n}\sum_{i=1}^{n} 1_{\{X_i \leq x\}}.$$

Algorithm

1. For every i=1,. . . ,n draw X_i^* from F_n, i.e., $P(X_i^* = X_j) = \frac{1}{n}$, $j = 1, \ldots, n$.
2. Obtain the resample $\mathbf{X}^* = (X_1^*, \ldots, X_n^*)$.
3. Compute the statistic of interest in the bootstrap world $T^* = T(X_1^*, \ldots, X_n^*)$.

Uniform bootstrap (naïve bootstrap)

$\widehat{F}$ is done by the **empirical distribution function**

$$F_n = \frac{1}{n}\sum_{i=1}^{n} 1_{\{X_i \leq x\}}.$$

Algorithm

1. For every i=1,...,n draw X_i^* from F_n, i.e., $P(X_i^* = X_j) = \frac{1}{n}$, $j = 1, \ldots, n$.
2. Obtain the resample $\mathbf{X}^* = (X_1^*, \ldots, X_n^*)$.
3. Compute the statistic of interest in the bootstrap world $T^* = T(X_1^*, \ldots, X_n^*)$.

Remark! In Step 1, we need to simulate a univariate discrete distribution. There are many possibilities, an efficient one (due to the equiprobability of the values) is the quantile transformation with direct search. In this case, the step will be rewritten as

1. For every i=1,...,n draw $U_i \sim U(0,1)$ and compute $X_i^* = X_{\lceil nU_i \rceil}$.

Uniform bootstrap (naïve bootstrap)

Example 1 (part I): Inference on the mean with known variance.

```
> sample.orig<-c(0.52, 0.61, -0.07, -1.23, 0.55, -1.34,
-0.10, 1.33, 1.52, 0.79, 1.14, 0.06, -0.74, 0.42, 1.08)
> n=length(sample.orig)
> sigma=0.5 #assumed to be known
> x.bar=mean(sample.orig) #statistic in the original
sample
> B=10000 #number of Bootstrap replicates
> resample<-numeric(n)
> stat.boot.unif<-numeric(B) #T*
>for (k in 1:B) {
+    resample<-sample(sample.orig,n,replace=T) #resamples
+    stat.boot.unif[k]=mean(resample)}
```

Uniform bootstrap (naïve bootstrap)

Example 1 (part I): Inference on the mean with known variance.

```
> sample.orig<-c(0.52, 0.61, -0.07, -1.23, 0.55, -1.34,
-0.10, 1.33, 1.52, 0.79, 1.14, 0.06, -0.74, 0.42, 1.08)
> n=length(sample.orig)
> sigma=0.5 #assumed to be known
> x.bar=mean(sample.orig) #statistic in the original
sample
> B=10000 #number of Bootstrap replicates
> resample<-numeric(n)
> stat.boot.unif<-numeric(B) #T*
>for (k in 1:B) {
+    resample<-sample(sample.orig,n,replace=T) #resamples
+    stat.boot.unif[k]=mean(resample)}
```

Uniform bootstrap (naïve bootstrap)

Example 1 (part II): Inference on the mean with known variance.

```
> sample.orig<-c(0.52, 0.61, -0.07, -1.23, 0.55, -1.34,
-0.10, 1.33, 1.52, 0.79, 1.14, 0.06, -0.74, 0.42, 1.08)
> n=length(sample.orig)
> sigma=1 #assumed to be known
>stat<-sqrt(length(sample.orig))*(mean(sample.orig)-0)/sigma
> B=10000 #number of Bootstrap replicates
> resample<-numeric(n)
> stat.boot.unif<-numeric(B) #T*
>for (k in 1:B) {
+    resample<-sample(sample.orig,n,replace=T) #resamples
+    stat.boot.unif[k]=sqrt(n)*(x.bar.boot-stat)/sigma}
```

Exact bootstrap distribution

In principle, it is always possible to compute the exact distribution of the bootstrap statistic in the bootstrap world, at least for uniform bootstrap that is the most popular.

Exact bootstrap distribution

In principle, it is always possible to compute the exact distribution of the bootstrap statistic in the bootstrap world, at least for uniform bootstrap that is the most popular.

Reason: In the bootstrap world, the distribution of the original sample is discrete and take values $X_1, X_2, \ldots, X_n$. Then, each bootstrap observation X_i^* will take one of these n values and as a consequence the number of possible bootstrap samples $\mathbf{X}^* = (X_1^*, X_2^*, \ldots, X_n^*)$ is n^n (it allows repeated values).

Exact bootstrap distribution

In principle, it is always possible to compute the exact distribution of the bootstrap statistic in the bootstrap world, at least for uniform bootstrap that is the most popular.

Reason: In the bootstrap world, the distribution of the original sample is discrete and take values $X_1, X_2, \ldots, X_n$. Then, each bootstrap observation X_i^* will take one of these n values and as a consequence the number of possible bootstrap samples $\mathbf{X}^* = (X_1^*, X_2^*, \ldots, X_n^*)$ is n^n (it allows repeated values).

Remark! The number of possible bootstrap sample is finite, but it is really big even for small sample sizes. For instance, for $n = 10$ we will have 10^{10} possible bootstrap samples and for $n = 20$, this number will increase up to $20^{20} = 10.4857 \times 10^{25}$. That means, that computing the exact bootstrap distribution is an intractable problem.

Exact bootstrap distribution

Example: Let us consider a random sample of a population with distribution F. If the sample size is $n = 3$, the parameter of interest if $\theta(F) = \mu$ and the statistic if $T = T(X_1, \ldots, X_n) = \overline{X}$. Then the distribution of the bootstrap statistic $T^* = T(X_1^*, \ldots, X_n^*) = \overline{X}^*$ can be computed as follows

$\mathbf{X}^*$ (+permutations)	(m_1, m_2, m_3)	(p_1, p_2, p_3)	$\frac{3!}{m_1!\, m_2!\, m_3!\, 3^3}$	$\overline{X}^*$
(X_1, X_1, X_1)	$(3, 0, 0)$	$(1, 0, 0)$	$1/27$	X_1
(X_2, X_2, X_2)	$(0, 3, 0)$	$(0, 1, 0)$	$1/27$	X_2
(X_3, X_3, X_3)	$(0, 0, 3)$	$(0, 0, 1)$	$1/27$	X_3
(X_1, X_1, X_2)	$(2, 1, 0)$	$(2/3, 1/3, 0)$	$1/9$	$(2X_1 + X_2)/3$
(X_1, X_1, X_3)	$(2, 0, 1)$	$(2/3, 0, 1/3)$	$1/9$	$(2X_1 + X_3)/3$
(X_1, X_2, X_2)	$(1, 2, 0)$	$(1/3, 2/3, 0)$	$1/9$	$(2X_2 + X_3)/3$
(X_2, X_2, X_3)	$(0, 2, 1)$	$(0, 2/3, 1/3)$	$1/9$	$(2X_2 + X_3)/3$
(X_1, X_3, X_3)	$(1, 0, 2)$	$(1/3, 0, 2/3)$	$1/9$	$(X_1 + 2X_3)/3$
(X_2, X_3, X_3)	$(0, 1, 2)$	$(0, 1/3, 2/3)$	$1/9$	$(X_2 + 2X_3)/3$
(X_1, X_2, X_3)	$(1, 1, 1)$	$(1/3, 1/3, 1/3)$	$2/9$	$(X_1 + X_2 + X_3)/3$

Exact bootstrap distribution

Example: Let us consider a random sample of a population with distribution F. If the sample size is $n = 3$, the parameter of interest if $\theta(F) = \mu$ and the statistic if $T = T(X_1, \ldots, X_n) = \overline{X}$. Then the distribution of the bootstrap statistic $T^* = T(X_1^*, \ldots, X_n^*) = \overline{X}^*$ can be computed as follows

$\mathbf{X}^*$ (+permutations)	(m_1, m_2, m_3)	(p_1, p_2, p_3)	$\frac{3!}{m_1!\, m_2!\, m_3!\, 3^3}$	$\overline{X}^*$
(X_1, X_1, X_1)	$(3, 0, 0)$	$(1, 0, 0)$	$1/27$	X_1
(X_2, X_2, X_2)	$(0, 3, 0)$	$(0, 1, 0)$	$1/27$	X_2
(X_3, X_3, X_3)	$(0, 0, 3)$	$(0, 0, 1)$	$1/27$	X_3
(X_1, X_1, X_2)	$(2, 1, 0)$	$(2/3, 1/3, 0)$	$1/9$	$(2X_1 + X_2)/3$
(X_1, X_1, X_3)	$(2, 0, 1)$	$(2/3, 0, 1/3)$	$1/9$	$(2X_1 + X_3)/3$
(X_1, X_2, X_2)	$(1, 2, 0)$	$(1/3, 2/3, 0)$	$1/9$	$(2X_2 + X_3)/3$
(X_2, X_2, X_3)	$(0, 2, 1)$	$(0, 2/3, 1/3)$	$1/9$	$(2X_2 + X_3)/3$
(X_1, X_3, X_3)	$(1, 0, 2)$	$(1/3, 0, 2/3)$	$1/9$	$(X_1 + 2X_3)/3$
(X_2, X_3, X_3)	$(0, 1, 2)$	$(0, 1/3, 2/3)$	$1/9$	$(X_2 + 2X_3)/3$
(X_1, X_2, X_3)	$(1, 1, 1)$	$(1/3, 1/3, 1/3)$	$2/9$	$(X_1 + X_2 + X_3)/3$

Approximated bootstrap distribution

Idea: In a bootstrap scenario, the mechanism to generate the data is known and as a result it can be simulated using Monte Carlo methods.

Algorithm: general structure for a uniform bootstrap

1. For each $i = 1, \ldots, n$ generate $X_i^\star$ from F_n.
2. Obtain $\mathbf{X}^* = (X_1^*, \ldots, X_n^*)$.
3. Compute $T^* = T(X_1^*, \ldots, X_n^*)$.
4. Repeat Step 1-3 B times in order to obtain the bootstrap replications $T^{*(1)}, \ldots, T^{*(B)}$.
5. Use the bootstrap replications to approximate the distribution of T.

Approximated bootstrap distribution

Idea: In a bootstrap scenario, the mechanism to generate the data is known and as a result it can be simulated using Monte Carlo methods.

Algorithm: general structure for a uniform bootstrap

1. For each $i = 1, \ldots, n$ generate $X_i^\star$ from F_n.
2. Obtain $\mathbf{X}^* = (X_1^*, \ldots, X_n^*)$.
3. Compute $T^* = T(X_1^*, \ldots, X_n^*)$.
4. Repeat Step 1-3 B times in order to obtain the bootstrap replications $T^{*(1)}, \ldots, T^{*(B)}$.
5. Use the bootstrap replications to approximate the distribution of T.

Remark 1! In order to complete Step 1, we can use the algorithms seen to generate discrete random variables. In R can be solved thanks to the function `sample`.

Remark 2! The number B should be big in order to obtain a good approximation. Example: $B = 100, 200, 500$ to estimate bias or variance and $B = 500, 1000, 5000$ for build confidence intervals or testing hypothesis.

Approximated bootstrap distribution

The distribution function of the estimator of interest

$$\psi(u) = P(T \leq u)$$

can be estimated using the empirical distribution of the bootstrap replications,

$$\widehat{\psi}_B(u) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{T^{*(i)} \leq u\}}.$$

The empirical distribution $\widehat{\psi}_B(u)$ is an estimator of the exact bootstrap distribution $\widehat{\psi}_B(u) = P^*(T^* \leq u)$.

Approximated bootstrap distribution

The distribution function of the estimator of interest

$$\psi(u) = P(T \leq u)$$

can be estimated using the empirical distribution of the bootstrap replications,

$$\widehat{\psi}_B(u) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{T^{*(i)} \leq u\}}.$$

The empirical distribution $\widehat{\psi}_B(u)$ is an estimator of the exact bootstrap distribution $\widehat{\psi}_B(u) = P^*(T^* \leq u)$.

$$E^{\text{MC}}\left(\widehat{\psi}_B(u)\right) = \widehat{\psi}(u)$$

$$\text{Var}^{\text{MC}}\left(\widehat{\psi}_B(u)\right) = \frac{1}{B}\,\widehat{\psi}(u)\,(1 - \widehat{\psi}(u))$$

Approximated bootstrap distribution

The distribution function of the estimator of interest

$$\psi(u) = P(T \leq u)$$

can be estimated using the empirical distribution of the bootstrap replications,

$$\widehat{\psi}_B(u) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{T^{*(i)} \leq u\}}.$$

The empirical distribution $\widehat{\psi}_B(u)$ is an estimator of the exact bootstrap distribution $\widehat{\psi}_B(u) = P^*(T^* \leq u)$.

$$E^{\text{MC}}\left(\widehat{\psi}_B(u)\right) = \widehat{\psi}(u)$$

$$\text{Var}^{\text{MC}}\left(\widehat{\psi}_B(u)\right) = \frac{1}{B}\,\widehat{\psi}(u)\,(1 - \widehat{\psi}(u))$$

Remark! The Monte Carlo error, that is given by the Monte Carlo variance, can be constrained by $\frac{1}{2\sqrt{B}}$.

Approximated bootstrap distribution

Example: Let us consider a random sample of a variable X with distribution F. The distribution of the parameter $\theta = \text{median}(X)$ can be estimated using a bootstrap approximation.

In a simple simulation example, let us consider the random sample

$$\begin{aligned}\mathbf{X} =&(0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1.0,\\ &1.1,\ 1.2,\ 1.3,\ 1.4,\ 1.5)\end{aligned}$$

and a uniform bootstrap procedure.

Approximated bootstrap distribution

```
> set.seed(1234)
> ori.sample<-seq(0.1,1.5,length=15);
> B=1000000 ; boot.median<-numeric(B)
> for (k in 1:B){
+    boot.sample<-sample(ori.sample,n,replace=T)
+    sort.boot.sample=sort(boot.sample)
+    boot.median[k]=sort.boot.sample[8]
+ }
> var.boot.median=(1/B)*sum((boot.median-
+    mean(boot.median))^2);
> var.boot.median
[1] 0.03395916
> bias.boot.median=mean(boot.median)-mean(ori.sample)
> bias.boot.median
[1] -9.63e-05
> ECM.median=(bias.boot.median)^2+var.boot.median
> ECM.median
[1] 0.03395917
```

Part II: Resampling

Module II: Bootstrap variations

Bootstrap variations

- Uniform bootstrap uses the empirical distribution.
- We may have some information on the population.
- Modifications of the bootstrap: take profit of this information to improve the results.

Parametric bootstrap

$F \in \{F_\theta / \theta \in \Theta \subset \mathbb{R}^d\}$, we estimate θ from the sample, $\widehat{\theta}$ (using for example max likelihood) and we obtain resamples from $F_{\widehat{\theta}}$.

Algorithm

1. Given the sample $\boldsymbol{X} = (X_1, \ldots, X_n)$, estimate $\widehat{\theta}$.
2. For every i=1,. . . ,n draw X_i^* from $F_{\widehat{\theta}}$.
3. Obtain the resample $\mathbf{X}^* = (X_1^*, \ldots, X_n^*)$.
4. Compute the statistic of interest in the bootstrap world $T^* = T(X_1^*, \ldots, X_n^*)$.

Parametric bootstrap

$F \in \{F_\theta / \theta \in \Theta \subset \mathbb{R}^d\}$, we estimate θ from the sample, $\widehat{\theta}$ (using for example max likelihood) and we obtain resamples from $F_{\widehat{\theta}}$.

Algorithm

1. Given the sample $\boldsymbol{X} = (X_1, \ldots, X_n)$, estimate $\widehat{\theta}$.
2. For every i=1,...,n draw X_i^* from $F_{\widehat{\theta}}$.
3. Obtain the resample $\mathbf{X}^* = (X_1^*, \ldots, X_n^*)$.
4. Compute the statistic of interest in the bootstrap world $T^* = T(X_1^*, \ldots, X_n^*)$.

Remark! In Step 2, we need to simulate values from $F_{\widehat{\theta}}$. We may use inversion method (if possible) and in this case, the step will be rewritten as

2. For every i=1,...,n draw $U_i \sim U(0,1)$ and compute $X_i^* = F_{\widehat{\theta}}^{-1}(U_i)$.

When the inverse function can not be computed (Gaussian distribution for example) statistical software used to have specific generating functions or we can use other methods.

Parametric bootstrap

Example 1 (part III): Inference on the mean with known variance.

```
> sample.orig<-c(-1.21,0.28,1.08,-2.35,0.43,0.51,-0.57,
+ -0.55,-0.56,-0.89,-0.48,-1.00,-0.78,0.06,0.96)
> n=length(sample.orig)
> sigma=1 #assumed to be known
> x.bar=mean(sample.orig)
> B=10000 #number of Bootstrap replicates
> resample<-numeric(n)
> stat.boot<-numeric(B) #T*
> for (k in 1:B) {
+    u=rnorm(n)
+    resample=u*sigma+x.bar
#equivalent: resample=rnorm(n,x.bar,sigma)
+    stat.boot[k]=mean(resample)
#stat.boot[k]=sqrt(n)*(x.bar.boot-x.bar)/sigma}
```

Symmetrised bootstrap

We know that the distribution function of the population is **symmetric**.

There is a value c such that $F(c-h) = F(c+h)\ \forall h > 0$, and it can be proved that $c = \mu$ the mean of the distribution (whenever it exists). Equivalently, this symmetric property can be expressed as $F(x) = 1 - F(2\mu - x)\ \forall x \in \mathbb{R}$.

The distribution in the bootstrap world is then done with a symmetrised version of the empirical distribution function

$$\widehat{F} = F_n^{\text{sim}}.$$

Symmetrised bootstrap

How can we define this estimator?

We first build a new sample by symmetrising the original one around the sample mean:

$$Y_i = \begin{cases} X_i & i = 1, \ldots, n, \\ 2\overline{X} - X_{i-n} & i = n+1, \ldots, 2n. \end{cases}$$

And then F_n^{sim} is the empirical distribution of this new sample, i.e, it asigns an equal probability of $1/2n$ to each data Y_i.

$$F_n^{\text{sim}}(x) = \frac{1}{2n} \sum_{i=1}^{2n} 1_{\{Y_i \leq x\}} \Longleftrightarrow F_n^{\text{sim}}(x) = \frac{1}{2} \left(F_n(x) + 1 - F_n(2\overline{X} - x) \right).$$

Symmetrised bootstrap

Algorithm

1. For every i=1,. . . ,n draw X_i^* from F_n^{sim}, i.e., $P(X_i^* = Y_j) = \frac{1}{2n}$, $j = 1, \ldots, 2n$.
2. Obtain the resample $\mathbf{X}^* = (X_1^*, \ldots, X_n^*)$.
3. Compute the statistic of interest in the bootstrap world $T^* = T(X_1^*, \ldots, X_n^*)$.

Symmetrised bootstrap

Algorithm

1. For every i=1,...,n draw X_i^* from F_n^{sim}, i.e., $P(X_i^* = Y_j) = \frac{1}{2n}$, $j = 1, \ldots, 2n$.
2. Obtain the resample $\mathbf{X}^* = (X_1^*, \ldots, X_n^*)$.
3. Compute the statistic of interest in the bootstrap world $T^* = T(X_1^*, \ldots, X_n^*)$.

Remark! In Step 1, we may proceed in two different ways: compute the symmetrised version of the original sample and draw values from there with equiprobability, and then it would be rewritten as

1. For every i=1,...,n draw $U_i \sim U(0,1)$ and compute $X_i^* = X_{\lceil 2nU_i \rceil}$.

or use the fact that F_n^{sim} is the distribution of a random variable built in two steps, in the first one sampling from F_n and in the second decide with equiprobability if we remain with that value or its symmetric around the sample mean. In this case it may be rewritten as

1. For every i=1,...,n draw $U_i, V_i \sim U(0,1)$. If $V_i \leq 0.5$ then $X_i^* = X_{\lceil nU_i \rceil}$ and if not, $X_i^* = 2\overline{X} - X_{\lceil nU_i \rceil}$.

Symmetrised bootstrap

Example 1 (part IV): Inference on the mean with known variance.

```
> sample.orig<-c(0.52, 0.61, -0.07, -1.23, 0.55, -1.34,
-0.10, 1.33, 1.52, 0.79, 1.14, 0.06, -0.74, 0.42, 1.08)
> n=length(sample.orig)
> sigma=0.5 #assumed to be known
> x.bar=mean(sample.orig)
> sample.symm<-c(sample.orig,2*x.bar-sample.orig)
> B=10000 #number of Bootstrap replicates
> resample<-numeric(n)
> stat.boot.symm<-numeric(B) #T*
>for (k in 1:B) {
+    resample<-sample(sample.symm,n,replace=T) #resamples
+    stat.boot.symm[k]=mean(resample)
}
```

Smooth bootstrap

We know that the distribution function of the population is **continuous**.

This means that the distribution has an associated density function, $f(x) = F'(x)$. So we need to use a bootstrap resampling from a continuous framework, for which we will use kernel estimators.

The estimator of the distribution function is based on the well known kernel density estimator

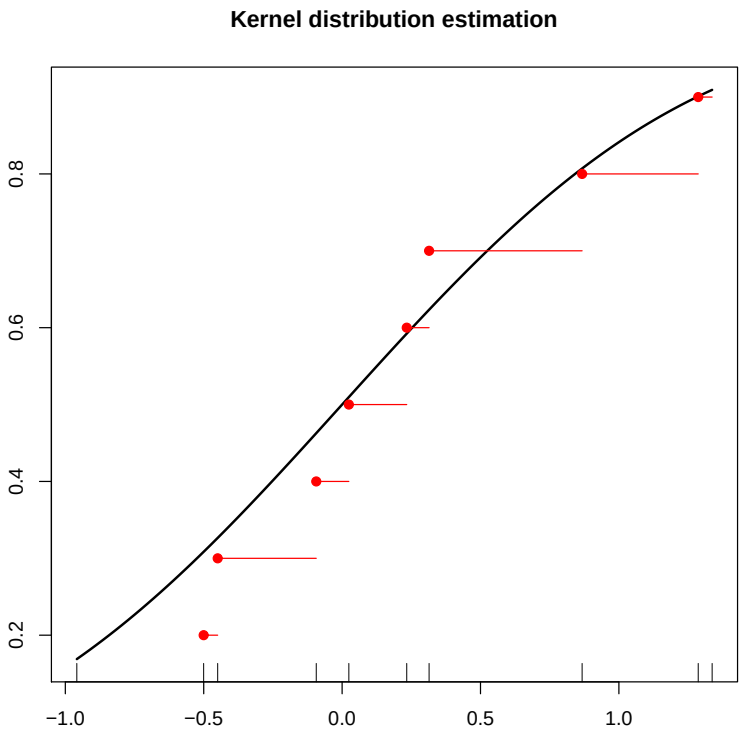
$$F_h(x) = \int_{-\infty}^{x} \widehat{f}_h(t)dt = \int_{-\infty}^{x} \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) dt$$

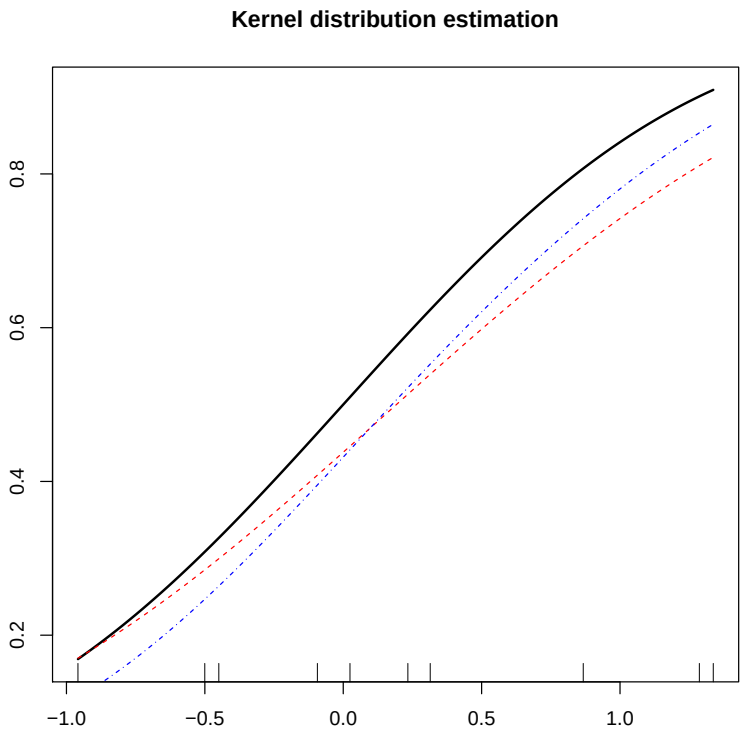
$$= \frac{1}{nh}\sum_{i=1}^{n} \int_{-\infty}^{\frac{x-X_i}{h}} K(u)du = \frac{1}{n}\sum_{i=1}^{n} \mathbb{K}\left(\frac{x - X_i}{h}\right),$$

KDE

where h is the bandwidth parameter and $\mathbb{K}(t) = \int_{-\infty}^{t} K(u)du$.

Smooth bootstrap





Smooth bootstrap

Algorithm

1. From the original sample $(X_1, \ldots, X_n)$ using a value $h > 0$ compute KDE $\widehat{f}_h$.
2. Obtain the resample $\mathbf{X}^* = (X_1^*, \ldots, X_n^*)$ from $\widehat{f}_h$.
3. Compute the statistic of interest in the bootstrap world $T^* = T(X_1^*, \ldots, X_n^*)$.

Remark! To draw values from $\widehat{f}_h$ we may think it as a linear combination of density functions $K_h(x - X_i)$ with coefficients $1/n$, and then do the simulation in two steps: first select randomly and equiprobably $i \in \{1, \ldots, n\}$ and then draw a value from $K_h(\cdot - X_i)$ which is equivalent to draw V from K and $X_i + hV$. Step 2 may be rewritten as

2. For every i=1,. . . ,n draw $U_i \sim U(0,1)$, $V_i \sim K$ and compute $X_i^* = X_{\lceil nU_i \rceil} + hV_i$.

Smooth bootstrap

Example 1 (part V): Inference on the mean with known variance.

```
> sample.orig<-c(0.52, 0.61, -0.07, -1.23, 0.55, -1.34,
-0.10, 1.33, 1.52, 0.79, 1.14, 0.06, -0.74, 0.42, 1.08)
> n=length(sample.orig)
> sigma=0.5 #assumed to be known
> x.bar=mean(sample.orig)
> sample.symm<-c(sample.orig,2*x.bar-sample.orig)
> B=10000 # number of Bootstrap replicates
> h=0.0001 #bandwidth value
> resample<-numeric(n)
> stat.bootsmooth<-numeric(B) #T*
> for (k in 1:B) {
+    resample<-sample(sample.orig,n,replace=T) #resamples
from the empirical distribution
+    resample=resample+h*rnorm(n,0,1)
+    stat.boot.smooth[k]=mean(resample)
}
```

Smooth bootstrap

Example 1 (part V): Inference on the mean with known variance.

```
> sample.orig<-c(0.52, 0.61, -0.07, -1.23, 0.55, -1.34,
-0.10, 1.33, 1.52, 0.79, 1.14, 0.06, -0.74, 0.42, 1.08)
> n=length(sample.orig)
> sigma=0.5 #assumed to be known
> x.bar=mean(sample.orig)
> sample.symm<-c(sample.orig,2*x.bar-sample.orig)
> B=10000 # number of Bootstrap replicates
> h=0.0001 #bandwidth value
> resample<-numeric(n)
> stat.bootsmooth<-numeric(B) #T*
> for (k in 1:B) {
+    resample<-sample(sample.orig,n,re
from the empirical distribution
+    resample=resample+h*rnorm(n,0,1)
+    stat.boot.smooth[k]=mean(resample
}
```

rnorm is a function that draw values from a Gaussian distribution (K is Gaussian).

Weighted and biased bootstrap

The distribution used in the resampling procedure is discrete and assign probabilities only to the data of the sample

$$\widehat{F}(X_i) - \widehat{F}(X_i^-) = p_i,\ i = 1, \ldots, n$$

with $p_i \geq 0$ and $\sum_{i=1}^{n} p_i = 1$.

- Particularly when $p_i = \frac{1}{n}\ \forall i \in \{i, \ldots, n\}$ we get uniform bootstrap.
- **Applications:** censored data and also dependent data.

Weighted and biased bootstrap

The distribution used in the resampling procedure is discrete and assign probabilities only to the data of the sample

$$\widehat{F}(X_i) - \widehat{F}(X_i^-) = p_i,\ i = 1, \ldots, n$$

with $p_i \geq 0$ and $\sum_{i=1}^{n} p_i = 1$.

- Particularly when $p_i = \frac{1}{n}\, \forall i \in \{i, \ldots, n\}$ we get uniform bootstrap.
- **Applications:** censored data and also dependent data.

Weighted bootstrap derives in **biased bootstrap** when the vector of weights $\boldsymbol{p} = (p_1, \ldots, p_n)$ minimises the distance to the uniform bootstrap $\left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ under some restrictions derived from the problem we are analysing.

Hall, P., and Presnell, B. (1999) Intentionally biased bootstrap methods. *Journal of the Royal Statistical Society: Series B*, 61(1), 143-158

Bootstrap validation

Uniform bootstrap method FAILS: "Frankfurt taxis example"

In Frankfurt taxis are presumably numbered from 1 to N. After arriving at the train station you observe n taxi cabs numbered by $(X_1, \ldots, X_n)$. The maximum likelihood estimate for the total number of taxis N is $\widehat{\theta}_{\text{MLE}} = \widehat{\theta} = max(X_1, \ldots, X_n)$.

Bootstrap validation

Uniform bootstrap method FAILS: "Frankfurt taxis example"

In Frankfurt taxis are presumably numbered from 1 to N. After arriving at the train station you observe n taxi cabs numbered by $(X_1, \ldots, X_n)$. The maximum likelihood estimate for the total number of taxis N is $\widehat{\theta}_{\text{MLE}} = \widehat{\theta} = max(X_1, \ldots, X_n)$.

It is straightforward to show that the distribution of $n(\theta - \widehat{\theta})$ is exponential with parameter $1/\theta$ which implies in particular that, as for any continuous distribution, the value 0 is assumed with zero probability.

Bootstrap validation

Uniform bootstrap method FAILS: "Frankfurt taxis example"

In Frankfurt taxis are presumably numbered from 1 to N. After arriving at the train station you observe n taxi cabs numbered by $(X_1, \ldots, X_n)$. The maximum likelihood estimate for the total number of taxis N is $\widehat{\theta}_{\text{MLE}} = \widehat{\theta} = max(X_1, \ldots, X_n)$.

It is straightforward to show that the distribution of $n(\theta - \widehat{\theta})$ is exponential with parameter $1/\theta$ which implies in particular that, as for any continuous distribution, the value 0 is assumed with zero probability.

The bootstrap distribution for this statistic $n(\theta - \widehat{\theta})$ is the distribution of $n(\widehat{\theta} - \widehat{\theta}^*)$ where $\widehat{\theta} = X_i$ for a certain i.

Bootstrap validation

Uniform bootstrap method FAILS: "Frankfurt taxis example"

In Frankfurt taxis are presumably numbered from 1 to N. After arriving at the train station you observe n taxi cabs numbered by $(X_1, \ldots, X_n)$. The maximum likelihood estimate for the total number of taxis N is $\widehat{\theta}_{\text{MLE}} = \widehat{\theta} = max(X_1, \ldots, X_n)$.

It is straightforward to show that the distribution of $n(\theta - \widehat{\theta})$ is exponential with parameter $1/\theta$ which implies in particular that, as for any continuous distribution, the value 0 is assumed with zero probability.

The bootstrap distribution for this statistic $n(\theta - \widehat{\theta})$ is the distribution of $n(\widehat{\theta} - \widehat{\theta}^*)$ where $\widehat{\theta} = X_i$ for a certain i.

Then, we obtain a value of 0 with that probability for which the element X_i is being resampled.

Bootstrap validation

Uniform bootstrap method FAILS: "Frankfurt taxis example"

In Frankfurt taxis are presumably numbered from 1 to N. After arriving at the train station you observe n taxi cabs numbered by $(X_1, \ldots, X_n)$. The maximum likelihood estimate for the total number of taxis N is $\widehat{\theta}_{\text{MLE}} = \widehat{\theta} = max(X_1, \ldots, X_n)$.

It is straightforward to show that the distribution of $n(\theta - \widehat{\theta})$ is exponential with parameter $1/\theta$ which implies in particular that, as for any continuous distribution, the value 0 is assumed with zero probability.

The bootstrap distribution for this statistic $n(\theta - \widehat{\theta})$ is the distribution of $n(\widehat{\theta} - \widehat{\theta}^*)$ where $\widehat{\theta} = X_i$ for a certain i.

Then, we obtain a value of 0 with that probability for which the element X_i is being resampled.

It is well known that this probability converges toward $1 - \frac{1}{e}$ hence not towards the value 0 provided by the exponential distribution.

Bootstrap validation

Suppose that we want to estimate the distribution function

$$G_{\widehat{F},n}(q) = P[Q(Y_1, \ldots, Y_n; F) \leq q | F]$$

where the conditioning on F indicates that $Y_1, \ldots, Y_n$ is a random sample from F. The bootstrap estimate of $G_{\widehat{F},n}$ is

$$G_{\widehat{F},n}(q) = P[Q(Y_1^*, \ldots, Y_n^*; \widehat{F}) \leq q | \widehat{F}].$$

Bootstrap validation

Suppose that we want to estimate the distribution function

$$G_{\widehat{F},n}(q) = P[Q(Y_1, \ldots, Y_n; F) \leq q|F]$$

where the conditioning on F indicates that $Y_1, \ldots, Y_n$ is a random sample from F. The bootstrap estimate of $G_{\widehat{F},n}$ is

$$G_{\widehat{F},n}(q) = P[Q(Y_1^*, \ldots, Y_n^*; \widehat{F}) \leq q|\widehat{F}].$$

In order for $G_{\widehat{F},n}$ to approach $G_{F,n}$ as $n \to \infty$, three conditions must hold.

Bootstrap validation

Suppose that we want to estimate the distribution function

$$G_{\widehat{F},n}(q) = P[Q(Y_1, \ldots, Y_n; F) \leq q|F]$$

where the conditioning on F indicates that $Y_1, \ldots, Y_n$ is a random sample from F. The bootstrap estimate of $G_{\widehat{F},n}$ is

$$G_{\widehat{F},n}(q) = P[Q(Y_1^*, \ldots, Y_n^*; \widehat{F}) \leq q|\widehat{F}].$$

In order for $G_{\widehat{F},n}$ to approach $G_{F,n}$ as $n \to \infty$, three conditions must hold. Suppose that the true distribution F is surrounded by a neighbourhood $\mathcal{N}$ in a suitable space of distributions, and that as $n \to \infty$, $\widehat{F}$ eventually falls into $\mathcal{N}$ with probability one. Then the conditions are:

1. For any $A \in \mathcal{N}$, $G_{A,n}$ must converges weakly to a limit $G_{A,\infty}$.
2. This convergence must be uniform on $\mathcal{N}$.
3. The function mapping A to $G_{A,\infty}$ must be continuous.

Bootstrap validation

Suppose that we want to estimate the distribution function

$$G_{\widehat{F},n}(q) = P[Q(Y_1, \ldots, Y_n; F) \leq q|F]$$

where the conditioning on F indicates that $Y_1, \ldots, Y_n$ is a random sample from F. The bootstrap estimate of $G_{\widehat{F},n}$ is

$$G_{\widehat{F},n}(q) = P[Q(Y_1^*, \ldots, Y_n^*; \widehat{F}) \leq q|\widehat{F}].$$

In order for $G_{\widehat{F},n}$ to approach $G_{F,n}$ as $n \to \infty$, three conditions must hold. Suppose that the true distribution F is surrounded by a neighbourhood $\mathcal{N}$ in a suitable space of distributions, and that as $n \to \infty$, $\widehat{F}$ eventually falls into $\mathcal{N}$ with probability one. Then the conditions are:

1. For any $A \in \mathcal{N}$, $G_{A,n}$ must converges weakly to a limit $G_{A,\infty}$.
2. This convergence must be uniform on $\mathcal{N}$.
3. The function mapping A to $G_{A,\infty}$ must be continuous.

Under this conditions the bootstrap is **consistent**, meaning that for any q and ε it follows that

$$P\left(\left|G_{\widehat{F},n}(q) - G_{F,\infty}(q)\right|\right) \longrightarrow 0 \quad \text{as } n \to \infty.$$

Bootstrap validation

- Consistency is a weak property, but standard normal approximation methods are consistent in this sense.

Bootstrap validation

- Consistency is a weak property, but standard normal approximation methods are consistent in this sense.
- Once consistency is established, meaning that the resampling method is 'valid', we need to know whether the method is 'good' relative to other possible methods.

Bootstrap validation

- Consistency is a weak property, but standard normal approximation methods are consistent in this sense.
- Once consistency is established, meaning that the resampling method is 'valid', we need to know whether the method is 'good' relative to other possible methods.
- This involves looking at the rate of convergence to nominal properties.

Bootstrap validation

- Consistency is a weak property, but standard normal approximation methods are consistent in this sense.
- Once consistency is established, meaning that the resampling method is 'valid', we need to know whether the method is 'good' relative to other possible methods.
- This involves looking at the rate of convergence to nominal properties.

Bootstrap validation

- Consistency is a weak property, but standard normal approximation methods are consistent in this sense.
- Once consistency is established, meaning that the resampling method is 'valid', we need to know whether the method is 'good' relative to other possible methods.
- This involves looking at the rate of convergence to nominal properties.

Asymptotic accuracy

Bootstrap validation

Idea: to achieve valid conclusions, F and $\widehat{F}$ (in any of its forms) must be close to each other, at least in some asymptotic sense.

Bootstrap validation

Idea: to achieve valid conclusions, F and $\widehat{F}$ (in any of its forms) must be close to each other, at least in some asymptotic sense. There are several theoretical procedures to prove this idea:

- Proving that the bootstrap distribution converge to the same limit as the theoretical one we are interested in.

Bootstrap validation

Idea: to achieve valid conclusions, F and $\widehat{F}$ (in any of its forms) must be close to each other, at least in some asymptotic sense. There are several theoretical procedures to prove this idea:

- Proving that the bootstrap distribution converge to the same limit as the theoretical one we are interested in.

Bickel, P. J., and Freedman, D. A. (1981) Some asymptotic theory for the bootstrap. *The Annals of Statistics*, 9(6), 1196-1217

Bootstrap validation

Idea: to achieve valid conclusions, F and $\widehat{F}$ (in any of its forms) must be close to each other, at least in some asymptotic sense. There are several theoretical procedures to prove this idea:

- Proving that the bootstrap distribution converge to the same limit as the theoretical one we are interested in.
- Proving that a functional distance between both distributions converges to zero.

Bootstrap validation

Idea: to achieve valid conclusions, F and $\widehat{F}$ (in any of its forms) must be close to each other, at least in some asymptotic sense. There are several theoretical procedures to prove this idea:

- Proving that the bootstrap distribution converge to the same limit as the theoretical one we are interested in.
- Proving that a functional distance between both distributions converges to zero.

Bickel, P. J., and Freedman, D. A. (1981) Some asymptotic theory for the bootstrap. *The Annals of Statistics*, 9(6), 1196-1217

Bootstrap validation

Idea: to achieve valid conclusions, F and $\widehat{F}$ (in any of its forms) must be close to each other, at least in some asymptotic sense. There are several theoretical procedures to prove this idea:

- Proving that the bootstrap distribution converge to the same limit as the theoretical one we are interested in.
- Proving that a functional distance between both distributions converges to zero.
- Building Edgeworth expansions of both distributions and proving that the differences between them converges to zero.

Bootstrap validation

Idea: to achieve valid conclusions, F and $\widehat{F}$ (in any of its forms) must be close to each other, at least in some asymptotic sense. There are several theoretical procedures to prove this idea:

- Proving that the bootstrap distribution converge to the same limit as the theoretical one we are interested in.
- Proving that a functional distance between both distributions converges to zero.
- Building Edgeworth expansions of both distributions and proving that the differences between them converges to zero.

HALL, P. (1988) Theoretical comparison of bootstrap confidence intervals. *The Annals of Statistics*, 16(3), 927-953

Bootstrap validation

Idea: to achieve valid conclusions, F and $\widehat{F}$ (in any of its forms) must be close to each other, at least in some asymptotic sense. There are several theoretical procedures to prove this idea:

- Proving that the bootstrap distribution converge to the same limit as the theoretical one we are interested in.
- Proving that a functional distance between both distributions converges to zero.
- Building Edgeworth expansions of both distributions and proving that the differences between them converges to zero.
- Proving that analogous characteristics of both distributions have the same structure, replacing population moments by empirical ones (imitation).

Bootstrap validation

Idea: to achieve valid conclusions, F and $\widehat{F}$ (in any of its forms) must be close to each other, at least in some asymptotic sense. There are several theoretical procedures to prove this idea:

- Proving that the bootstrap distribution converge to the same limit as the theoretical one we are interested in.
- Proving that a functional distance between both distributions converges to zero.
- Building Edgeworth expansions of both distributions and proving that the differences between them converges to zero.
- Proving that analogous characteristics of both distributions have the same structure, replacing population moments by empirical ones (imitation).

Cao, R. and Prada, J.M. (1993) Bootstrapping the mean of a symmetric population. *Statistics and Probabililty Letters*, 17, 43-48

Bootstrap validation

We illustrate the **Edgeworth expansions method** with the statistic

$$T = \sqrt{n}\,\frac{\overline{X} - \mu}{\sigma},$$

where $\boldsymbol{X} = (X_1, \ldots, X_n)$ is a random sample from a distribution F with mean μ and standard deviation σ.

Applying the *Central Limit Theorem* we may obtain the asymptotic distribution (so when $n \to \infty$) of T

$$\lim_{n\to\infty} P(T \leq u) = \Phi(u),\ \forall u \in \mathbb{R},$$

with Φ the distribution function of a standard Gaussian and its density will be denoted bu ϕ.

Cramer's theorem

Let consider the random variables $X_1, \ldots, X_n, \ldots$ independent identically distributed from a distribution F with mean μ and standard deviation σ. Let assume there is $j \in \mathbb{N}$ such that $E(|X|^{j+2}) < \infty$ and $\lim_{|t|\to\infty} |\alpha(t)| < 1$ where $\alpha(t) = E(e^{itX})$ the characteristic function of the population. Then

$$P(T \leq u) = P\left(\sqrt{n}\frac{\overline{X} - \mu}{\sigma} \leq u\right)$$
$$= \Phi(u) + n^{-1/2}p_1(u)\phi(u) + \cdots + n^{-(j-1)/2}p_{j-1}(u)\phi(u) + O\left(n^{-j/2}\right),$$

where $p_i(u)$ are the polynomials of degree $3i-1$ with coefficients depending on the moments of order less than $i+2$ of X.

	CLT approximation	Uniform bootstrap
Order of convergence	$O\left(n^{-1/2}\right)$	$O\left(n^{-1}\right)$

Notation

Bootstrap validation

The results above, based on Edgeworth expansions, can be applied to many common statistics:

- Smooth functions of sample moments, such as means, variances, and higher moments.
- Smooth functions of solutions to smooth estimating equations, such as most maximum likelihood estimators, estimators in linear and generalized linear models.
- Some robust estimators.
- Many statistics calculated from time series.

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Applications*, Cambridge University Press.

Part II: Resampling

Module III: Bootstrap applications to build confidence intervals

Definition

A confidence interval of a parameter θ will be defined by limits $\widehat{\theta}_{\alpha_1}$ and $\widehat{\theta}_{1-\alpha_2}$, such that for any α

$$P(\theta < \widehat{\theta}_{\alpha}) = \alpha.$$

The coverage of the interval $(\widehat{\theta}_{\alpha_1}, \widehat{\theta}_{1-\alpha_2})$ is $1-(\alpha_1+\alpha_2)$, and α_1 and α_2 are respectively the left- and right-tail error probabilities.

In principle we can choose α_1 and α_2, so long as they sum to the overall error probability α. The simplest way to do this, which we adopt for general discussion, is to set $\alpha_1 = \alpha_2 = \alpha/2$. Then the interval is equi-tailed with coverage probability $1-\alpha$.

Classical method

Scenario: Let us consider a random variable X with mean μ and variance σ^2. We want to build a two sided confidence interval for μ when σ^2 is known.

Statistic: The statistic used in this case is

$$T_1 = \sqrt{n}\,\frac{\overline{X} - \mu}{\sigma} \longrightarrow N(0,1),$$

when $n \to \infty$.

Classical method

Scenario: Let us consider a random variable X with mean μ and variance σ^2. We want to build a two sided confidence interval for μ when σ^2 is known.

Statistic: The statistic used in this case is

$$T_1 = \sqrt{n}\,\frac{\overline{X} - \mu}{\sigma} \longrightarrow N(0,1),$$

when $n \to \infty$.

Confidence interval

The confidence interval for μ of level $(1-\alpha)$ based on asymptotic gaussianity is given by

$$I_1 = \left(\overline{X} - z_{\alpha/2}\,\frac{\sigma}{\sqrt{n}}\,,\ \overline{X} + z_{\alpha/2}\,\frac{\sigma}{\sqrt{n}}\right),$$

where $z_{\alpha/2}$ represents the quantile $1-\alpha/2$ of a standard Gaussian distribution.

Classical method

Generalization

Given a random variable X with distribution function F, a confidence interval for a parameter $\theta = \theta(F)$ of level $(1-\alpha)$ is given by

$$I_1 = \left(\widehat{\theta} - z_{\alpha/2} \frac{\widehat{\sigma}_\theta}{\sqrt{n}} \, , \, \widehat{\theta} + z_{\alpha/2} \frac{\widehat{\sigma}_\theta}{\sqrt{n}} \right),$$

where $z_{\alpha/2}$ represents the quantile $1-\alpha/2$ of a standard Gaussian distribution.

Percentile method

This method is based on the statistic

$$T_2 = \sqrt{n}\,\left(\widehat{\theta} - \theta\right).$$

Once we have performed a bootstrap procedure, given $\widehat{F}$ an estimator of F, the distribution of T_2 can be approximated by

$$T_2^* = \sqrt{n}\,\left(\widehat{\theta}^* - \theta(\widehat{F})\right).$$

Then, if us denote by q_β the β-quantile of T_2^*, that is, $P^*(T_2^* \leq q_\beta) = \beta$, it follows that

$$1 - \alpha = P^*\left(q_{\alpha/2} < T_2^* < q_{1-\alpha/2}\right) \simeq P\left(q_{\alpha/2} < T_2 < q_{1-\alpha/2}\right).$$

Percentile method

This method is based on the statistic

$$T_2 = \sqrt{n}\,\left(\widehat{\theta} - \theta\right).$$

Once we have performed a bootstrap procedure, given $\widehat{F}$ an estimator of F, the distribution of T_2 can be approximated by

$$T_2^* = \sqrt{n}\,\left(\widehat{\theta}^* - \theta(\widehat{F})\right).$$

Then, if us denote by q_β the β-quantile of T_2^*, that is, $P^*(T_2^* \leq q_\beta) = \beta$, it follows that

$$1-\alpha = P^*\left(q_{\alpha/2} < T_2^* < q_{1-\alpha/2}\right) \simeq P\left(q_{\alpha/2} < T_2 < q_{1-\alpha/2}\right).$$

Confidence interval of level $(1-\alpha)$

$$\widehat{I}_2 = \left(\widehat{\theta} - \frac{q_{1-\alpha/2}}{\sqrt{n}}\,,\, \widehat{\theta} - \frac{q_{\alpha/2}}{\sqrt{n}}\right)$$

Percentile method

This method is based on the statistic

$$T_2 = \sqrt{n}\,\left(\widehat{\theta} - \theta\right).$$

Once we have performed a bootstrap procedure, given $\widehat{F}$ an estimator of F, the distribution of T_2 can be approximated by

$$T_2^* = \sqrt{n}\,\left(\widehat{\theta}^* - \theta(\widehat{F})\right).$$

Then, if us denote by q_β the β-quantile of T_2^*, that is, $P^*(T_2^* \leq q_\beta) = \beta$, it follows that

$$1-\alpha = P^*\left(q_{\alpha/2} < T_2^* < q_{1-\alpha/2}\right) \simeq P\left(q_{\alpha/2} < T_2 < q_{1-\alpha/2}\right).$$

Confidence interval of level $(1-\alpha)$

$$\widehat{I}_2 = \left(\widehat{\theta} - \frac{q_{1-\alpha/2}}{\sqrt{n}}\,,\, \widehat{\theta} - \frac{q_{\alpha/2}}{\sqrt{n}}\right)$$

Percentile-t methods

This method is based on the statistic

$$T_3 = \sqrt{n}\left(\frac{\widehat{\theta} - \theta}{\widehat{\sigma}_\theta}\right).$$

Once we have performed a bootstrap procedure, given $\widehat{F}$ an estimator of F, the distribution of T_3 can be approximated by

$$T_3^* = \sqrt{n}\left(\frac{\widehat{\theta}^* - \theta(\widehat{F})}{\widehat{\sigma}_\theta^*}\right).$$

Then, if us denote by q_β the β-quantile of T_3^*, it follows that

$$1 - \alpha = P^*\left(q_{\alpha/2} < T_3^* < q_{1-\alpha/2}\right) \simeq P\left(q_{\alpha/2} < T_3 < q_{1-\alpha/2}\right).$$

Percentile-t methods

This method is based on the statistic

$$T_3 = \sqrt{n}\left(\frac{\widehat{\theta} - \theta}{\widehat{\sigma}_\theta}\right).$$

Once we have performed a bootstrap procedure, given $\widehat{F}$ an estimator of F, the distribution of T_3 can be approximated by

$$T_3^* = \sqrt{n}\left(\frac{\widehat{\theta}^* - \theta(\widehat{F})}{\widehat{\sigma}_\theta^*}\right).$$

Then, if us denote by q_β the β-quantile of T_3^*, it follows that

$$1 - \alpha = P^*\left(q_{\alpha/2} < T_3^* < q_{1-\alpha/2}\right) \simeq P\left(q_{\alpha/2} < T_3 < q_{1-\alpha/2}\right).$$

Confidence interval of level $(1-\alpha)$

$$\widehat{I}_3 = \left(\widehat{\theta} - \frac{\widehat{\sigma}_\theta}{\sqrt{n}}\, q_{1-\alpha/2}\,,\ \widehat{\theta} - \frac{\widehat{\sigma}_\theta}{\sqrt{n}}\, q_{\alpha/2}\right)$$

Adjusted Percentile-t methods

This method is quite similar to the previous one, but in this case the choice of the quantiles is different because now the quantiles have to be symmetric. Given the statistic T_3 and its bootstrap version T_3^*, let us consider the quantile $q_{1-\alpha}$ that verifies

$$\begin{aligned} 1-\alpha &= P^*(|T_3^*| \leq q_{1-\alpha}) = P^*(-q_{1-\alpha} \leq T_3^* \leq q_{1-\alpha}) \\ &= P^*\left(-q_{1-\alpha} \leq \sqrt{n}\left(\frac{\widehat{\theta}^* - \theta(\widehat{F})}{\widehat{\sigma}_\theta^*}\right) \leq q_{1-\alpha}\right) \\ &\simeq P^*(-q_{1-\alpha} \leq T_3 \leq q_{1-\alpha}). \end{aligned}$$

Adjusted Percentile-t methods

This method is quite similar to the previous one, but in this case the choice of the quantiles is different because now the quantiles have to be symmetric. Given the statistic T_3 and its bootstrap version T_3^*, let us consider the quantile $q_{1-\alpha}$ that verifies

$$\begin{aligned} 1-\alpha &= P^*(|T_3^*| \leq q_{1-\alpha}) = P^*(-q_{1-\alpha} \leq T_3^* \leq q_{1-\alpha}) \\ &= P^*\left(-q_{1-\alpha} \leq \sqrt{n}\left(\frac{\widehat{\theta}^* - \theta(\widehat{F})}{\widehat{\sigma}_\theta^*}\right) \leq q_{1-\alpha}\right) \\ &\simeq P^*(-q_{1-\alpha} \leq T_3 \leq q_{1-\alpha}). \end{aligned}$$

Confidence interval of level $(1-\alpha)$

$$\widehat{I}_4 = \left(\widehat{\theta} - \frac{\widehat{\sigma}_\theta}{\sqrt{n}}\, q_{1-\alpha}\ ,\ \widehat{\theta} + \frac{\widehat{\sigma}_\theta}{\sqrt{n}}\, q_{1-\alpha}\right).$$

Comparison

Coverage error

		Bootstrap		
	Classical	Percentile	Percentile-t	Adj. Percentile-t
Two sided	$O\left(n^{-1}\right)$	$O_p\left(n^{-1/2}\right)$	$O_p\left(n^{-1}\right)$	$O_p\left(n^{-3/2}\right)$
One sided	$O\left(n^{-1/2}\right)$	$O_p\left(n^{-1/2}\right)$	$O_p\left(n^{-1}\right)$	$O_p\left(n^{-1}\right)$

- Bhattacharya-Ghosh's theorem.
- Edgeworth expansions.

Efron, B. and Tibshirani, R.J. (1993). An Introduction to the Bootstrap. Chapman and Hall.

Example

Simulation study

Design a Monte Carlo simulation study in order to check the coverage error associated with the percentile-t method supposing that the original variable follows an exponential distribution with rate 0.01. Fix the confidence level equal to 0.90.

Example

```
> set.seed(1234)
> alpha=0.1 ; n=100 ; N=100 ; B=100
> percentilet<-numeric(B) ; coverageI3=numeric(N)
> for (z in 1:N){
+    ori.sample<-rexp(n,rate=0.01)
+    bar.x=mean(ori.sample) ; hat.sd=sd(ori.sample)
+    for (k in 1:B){
+      boot.sample<-sample(ori.sample,n,replace=T)
+      percentilet[k]=sqrt(n)*(mean(boot.sample)-bar.x)
+      /sd(boot.sample)
+    }
+    percentiletord=sort(percentilet)
+    quantile1=percentiletord[floor(B*(1-(alpha/2)))]
+    quantile2=percentiletord[floor(B*(alpha/2))]
+    I3=c(bar.x-hat.sd*quantile1/sqrt(n),
+      bar.x-hat.sd*quantile2/sqrt(n))
+    if((I3[1]<100)&(100<I3[2])){coverageI3[z]=1}
+ }
> mean(coverageI3)
```

Example

```
> set.seed(1234)
> alpha=0.1 ; n=100 ; N=100 ; B=100
> percentilet<-numeric(B) ; coverageI3=numeric(N)
> for (z in 1:N){
+    ori.sample<-rexp(n,rate=0.01)
+    bar.x=mean(ori.sample) ; hat.sd=sd(ori.sample)
+    for (k in 1:B){
+      boot.sample<-sample(ori.sample,n,replace=T)
+      percentilet[k]=sqrt(n)*(mean(boot.sample)-bar.x)
+      /sd(boot.sample)
+    }
+    percentiletord=sort(percentilet)
+    quantile1=percentiletord[floor(B*(1-(alpha/2)))]
+    quantile2=percentiletord[floor(B*(alpha/2))]
+    I3=c(bar.x-hat.sd*quantile1/sqrt(n),
+      bar.x-hat.sd*quantile2/sqrt(n))
+    if((I3[1]<100)&(100<I3[2])){coverageI3[z]=1}
+ }
> mean(coverageI3)
```

Empirical coverage: 0.88

Bootstrap in R

library(bootstrap)

```
bootstrap(x,nboot,theta,func=NULL)
boott(x,theta,sdfun=sdfunb,nbootsd=25,
nboott=200,...)
jackknife(x,theta)
```

Efron, B., and Tibshirani, R. J. (1993) *An introduction to the bootstrap*, CRC Press.

Bootstrap in R

library(boot)

```
boot(data,statistic,sim,strata,weights,...)
boot.ci(boot.out,conf=0.95,type=‘‘all’’,...)
censboot(data,statistic,R,F.surv,G.surv,...)
envelope(boot.out,level=0.95,...)
norm.ci(boot.out,conf=0.95,...)
plot.boot(boo.object); print.boot(boot.object)
print.bootci(bootci.object)
```

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Applications*, Cambridge University Press.

Bootstrap in R

Example

```
#Bootstrap 95 CI for regression coefficients
#function to obtain regression weights
> bs <- function(formula, data, indices) {
+    d <- data[indices,]
+    fit <- lm(formula, data=d)
+    return(coef(fit))}
#bootstrapping with 1000 replications
> results<-boot(data=mtcars,statistic=bs,R=1000,
+ formula=mpg~wt+disp)
#view results
> plot(results, index=1) #intercept
> plot(results, index=2) #wt
> plot(results, index=3) #disp
#get 95% confidence intervals
> boot.ci(results, type=``all'', index=1) #intercept
> boot.ci(results, type=``all'', index=2) #wt
> boot.ci(results, type=``all'', index=3) #disp
```

Simulation and Resampling

Maribel Borrajo García and Mercedes Conde Amboage

maribelborrajo@uniovi.es; mercedes.amboage@usc.es

Universidad de Oviedo
DEPARTAMENTO DE ESTADÍSTICA
E INVESTIGACIÓN OPERATIVA
Y DIDÁCTICA DE LA MATEMÁTICA

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA
DEPARTAMENTO DE ESTATÍSTICA,
ANÁLISE MATEMÁTICA E OPTIMIZACIÓN

modestya
Modelos de Optimización, Decisión, Estadística y Aplicaciones

CiTiUS
Centro Singular de Investigación
en Tecnoloxías da
Información

A brief introduction to

What is R ?

R is a free software environment for statistical computing and graphics.

- Main webpage: `www.r-project.org`.
- For almost any operative system.
- Collaborative program.
- Has a `base` package with simple statistical functionalities.
- Extra implementations: packages (free to be downloaded).
- Last released version 3.4.4.

Main characteristics of R

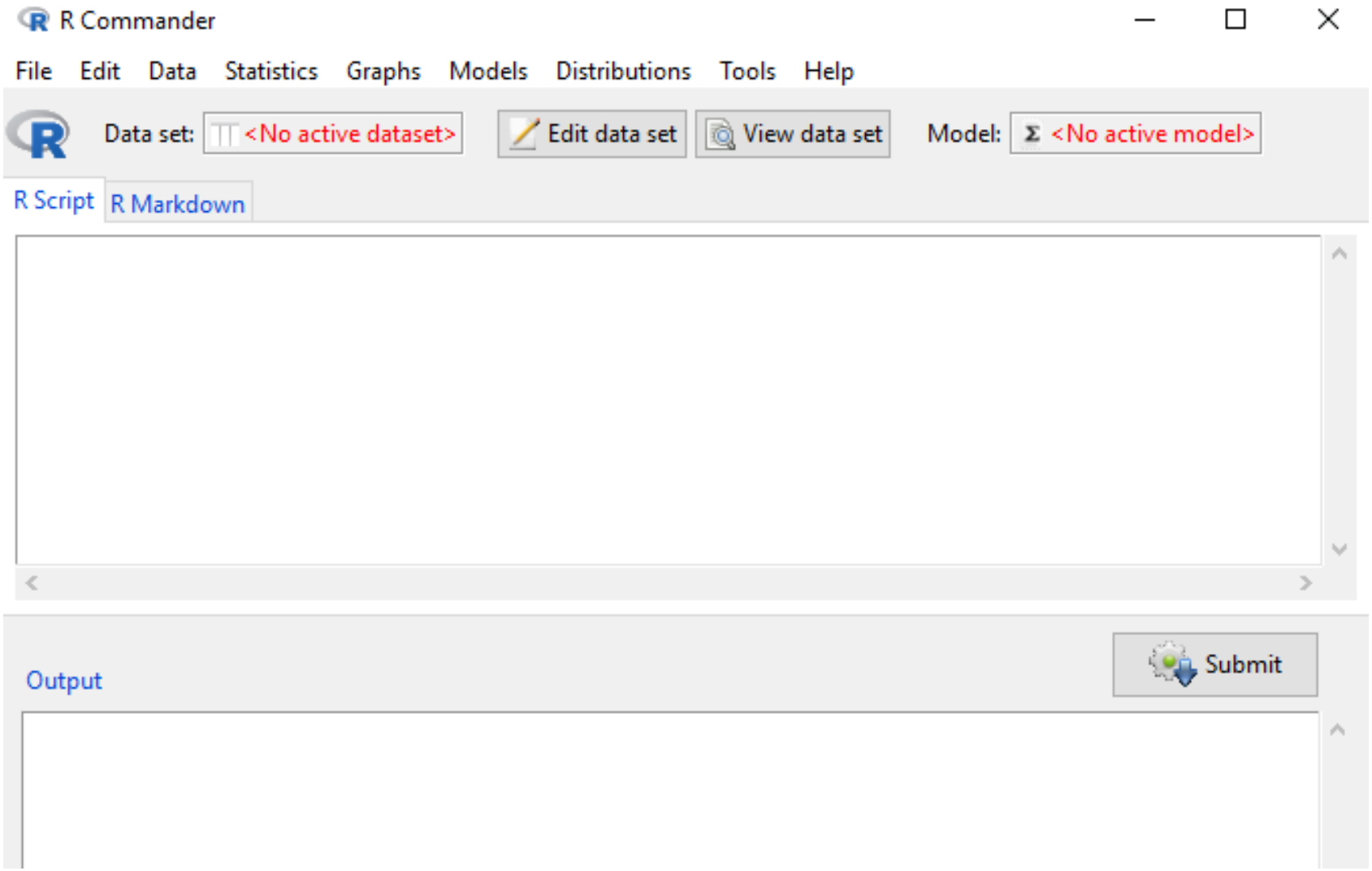
- Object oriented programming language.
- Distinguish small and capital letters.
- <- asign operator.
- # for comments.
- ; separates commands in one line.
- + let continue the command in next line.
- ? or `help(function.name)` or `heal.start()`.

Some curiosities to work with R

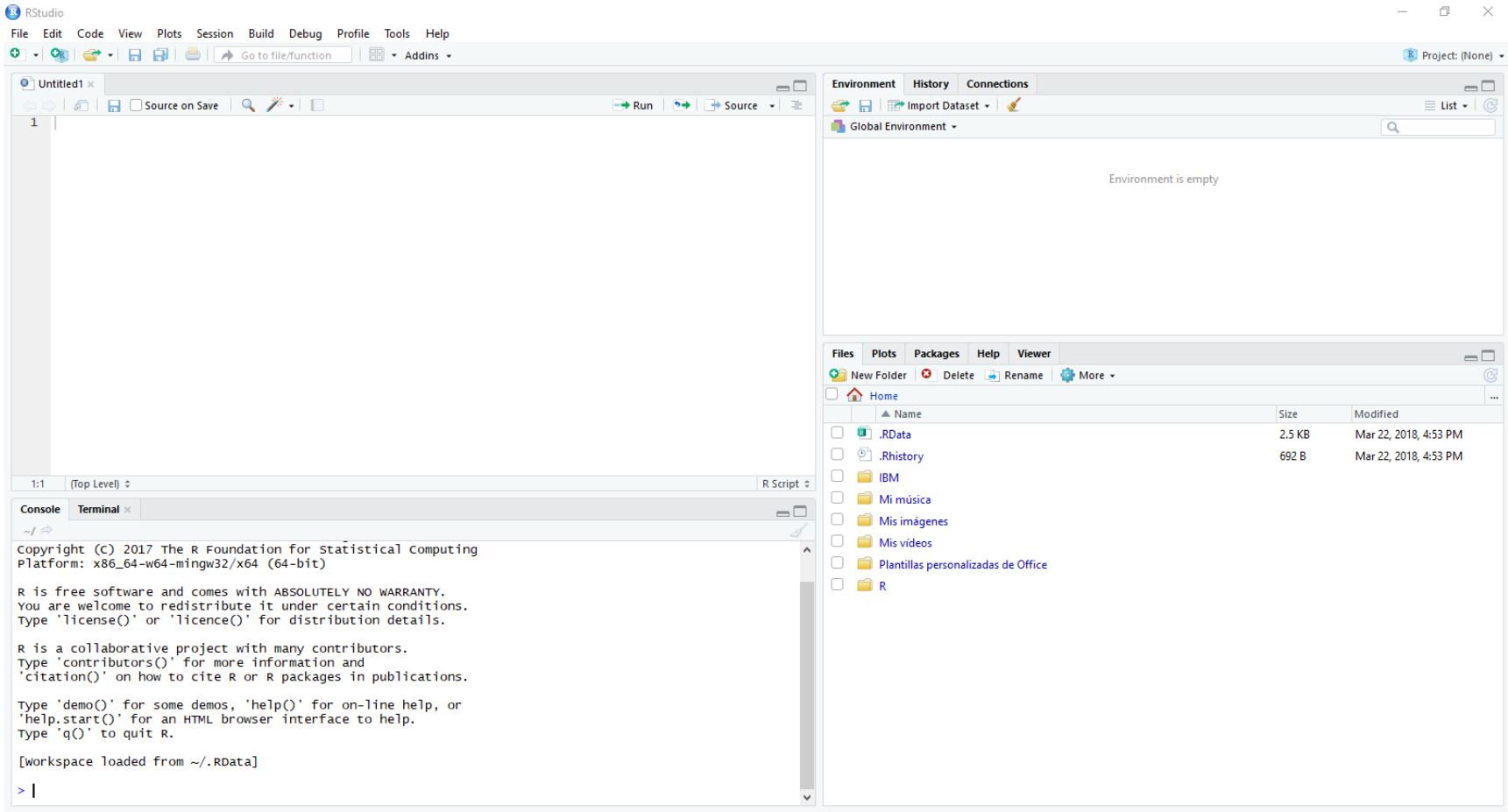
- **Aritmetic/logic operations:** element by element except %*%.
- **Loops:** `for` and `while`.
- **Functions:**

```
function.name<-function(arg1,arg2,...){
expr
return(argument.name.to.be.returned)}.
```

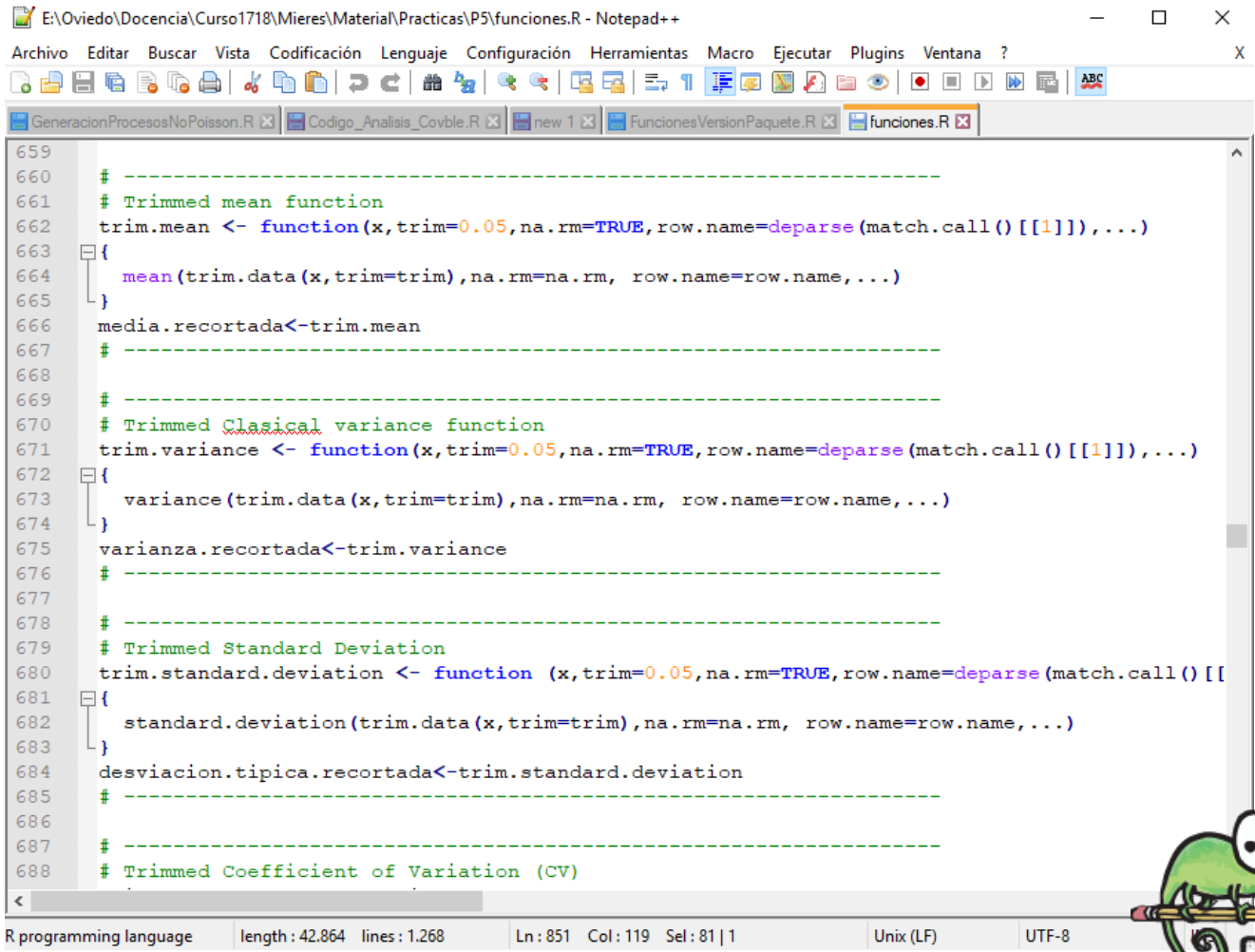
R Commander


R Commander
File Edit Data Statistics Graphs Models Distributions Tools Help
Data set: <No active dataset>
Edit data set
View data set
Model: <No active model>
R Script
R Markdown
Output
Submit

R Studio


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Console
Terminal
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
[Workspace loaded from ~/.RData]
>
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home
Name Size Modified
.RData 2.5 KB Mar 22, 2018, 4:53 PM
.Rhistory 692 B Mar 22, 2018, 4:53 PM
IBM
Mi música
Mis imágenes
Mis vídeos
Plantillas personalizadas de Office
R

Notepad++


E:\Oviedo\Docencia\Curso1718\Mieres\Material\Practicas\P5\funciones.R - Notepad++
Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins Ventana ?
GeneracionProcesosNoPoisson.R
Codigo_Analisis_Covble.R
new 1
FuncionesVersionPaquete.R
funciones.R
659
660 # --
661 # Trimmed mean function
662 trim.mean <- function(x,trim=0.05,na.rm=TRUE,row.name=deparse(match.call()[[1]]),...)
663 {
664 mean(trim.data(x,trim=trim),na.rm=na.rm, row.name=row.name,...)
665 }
666 media.recortada<-trim.mean
667 # --
668
669 # --
670 # Trimmed Clasical variance function
671 trim.variance <- function(x,trim=0.05,na.rm=TRUE,row.name=deparse(match.call()[[1]]),...)
672 {
673 variance(trim.data(x,trim=trim),na.rm=na.rm, row.name=row.name,...)
674 }
675 varianza.recortada<-trim.variance
676 # --
677
678 # --
679 # Trimmed Standard Deviation
680 trim.standard.deviation <- function (x,trim=0.05,na.rm=TRUE,row.name=deparse(match.call()[[
681 {
682 standard.deviation(trim.data(x,trim=trim),na.rm=na.rm, row.name=row.name,...)
683 }
684 desviacion.tipica.recortada<-trim.standard.deviation
685 # --
686
687 # --
688 # Trimmed Coefficient of Variation (CV)
R programming language
length : 42.864 lines : 1.268
Ln : 851 Col : 119 Sel : 81 | 1
Unix (LF)
UTF-8
Notepad++

Some ideas on Random Variables

Random variables

- **Qualitative variables:**
  - **Ordinal variables**
    - Level of education.
    - Social economic class.
  - **Nominal variables**
    - Gender.
    - Eye color.
- **Quantitative variables:**
  - **Discrete variables**
    - Result of flipping a coin.
    - The number of patients in a doctor's surgery.
  - **Continuous variables**
    - Height of students in class.
    - Time it takes a computer to complete a task.

Discrete random variables

A **discrete random variable** (denoted by X) is one which may take on only a countable number of distinct values.

- **Support:**

$$Sup(X) = \{x_1, \ldots, x_n\}.$$

- **Probability mass function:**

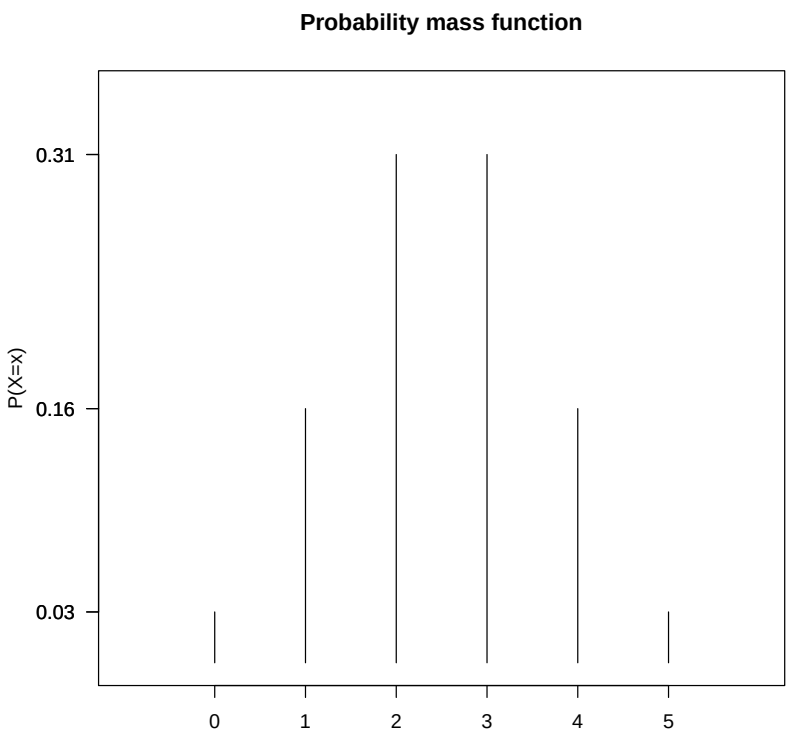
$$p_i = P(X = x_i).$$

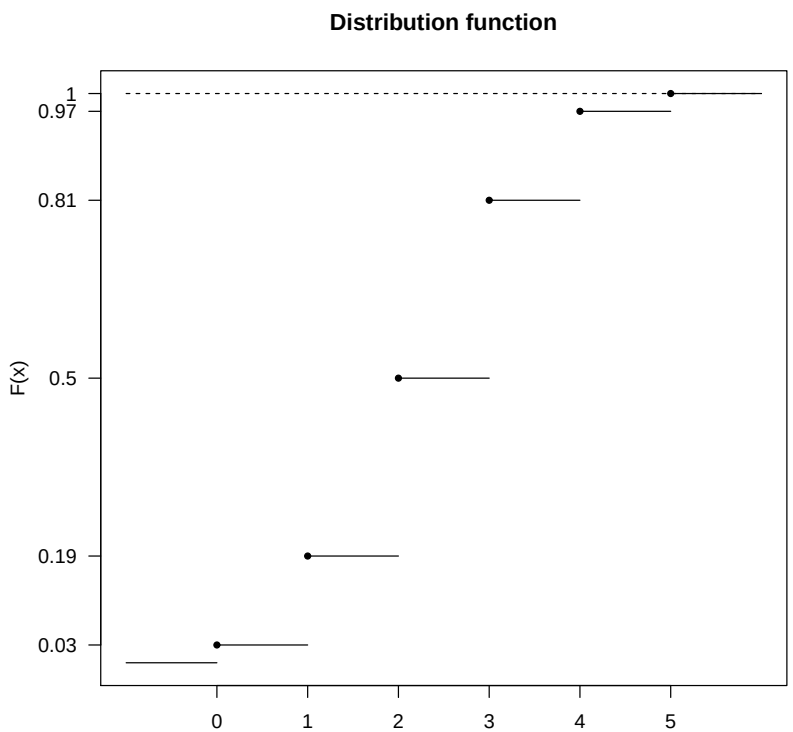
- **Cumulative distribution function:**

$$F(x) = \sum_{x_i \leq x} P(X = x) = \sum_{x_i \leq x} p_i.$$

Discrete random variables: examples

- Be(p): Bernoulli distribution with success probability p.
- Bi(n, p): binomial distribution with parameters n and p.
- Ge(p): geometric distribution with success probability p.
- Pois(λ): Poisson distribution with parameter λ.
- ...





Continuous random variables

A **continuous random variable** (denoted by X) is one which takes an infinite number of possible values.

- **Support:**

$$Sup(X) = (a, b) \text{ or } (a, b) \cup (c, d) \text{ or } \mathbb{R}.$$

- **Cumulative distribution function:**

$$F(x) = P(X \leq x)$$

- **Density function:**

$$f(x) = F'(x) \text{ or } F(x) = \int_{-\infty}^{x} f(z)dz.$$

Continuous random variables

A **continuous random variable** (denoted by X) is one which takes an infinite number of possible values.

- **Support:**

$$Sup(X) = (a, b) \text{ or } (a, b) \cup (c, d) \text{ or } \mathbb{R}.$$

- **Cumulative distribution function:**

$$F(x) = P(X \leq x)$$

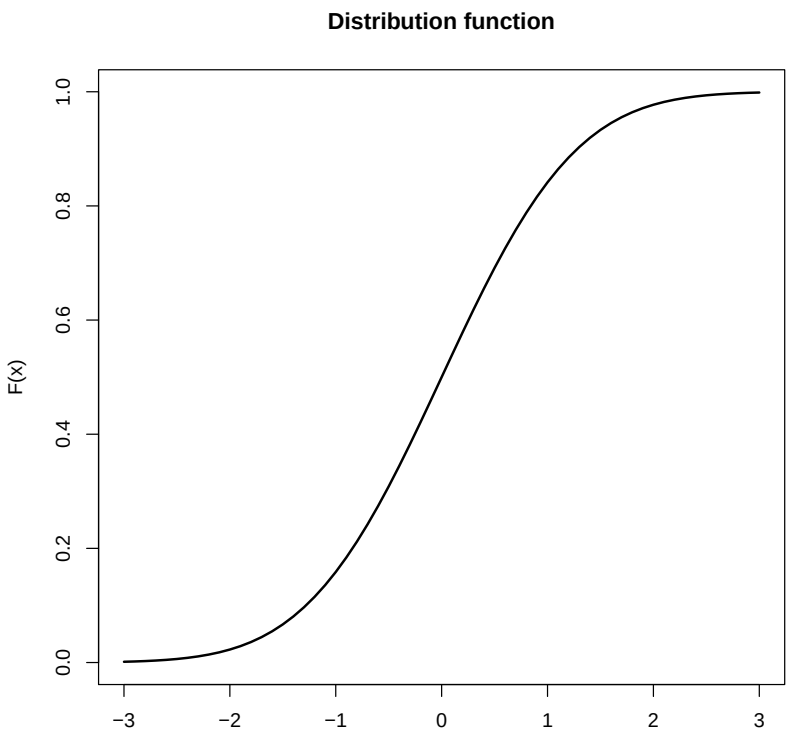
- **Density function:**

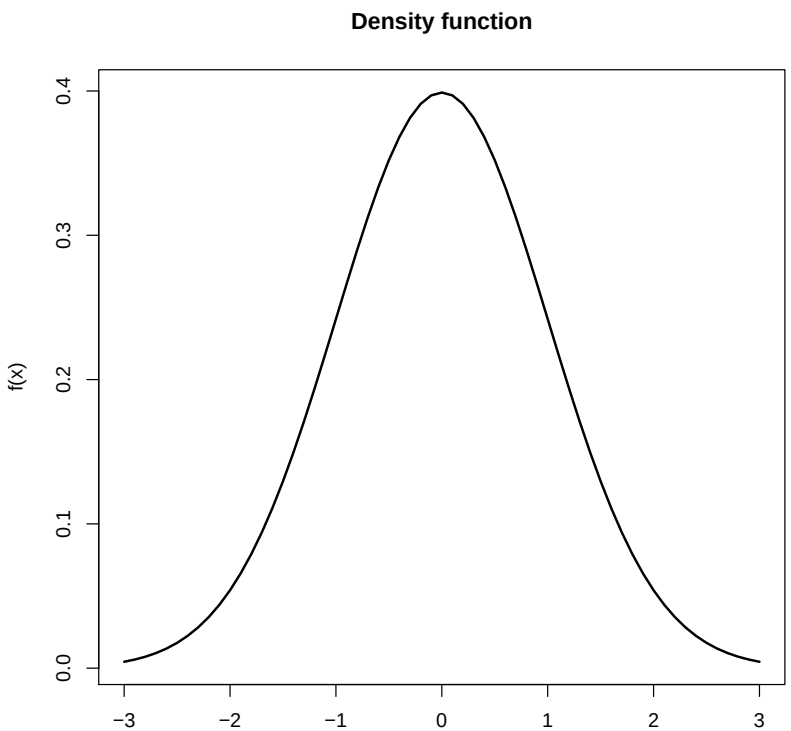
$$f(x) = F'(x) \text{ or } F(x) = \int_{-\infty}^{x} f(z)dz.$$

Remark! $P(X = x_0) = 0$ for all $x_0 \in \mathbb{R}$.

Continuous random variables: examples

- $U(a, b)$: uniform distribution on interval (a, b).
- $N(\mu, \sigma)$: Gaussian distribution with mean μ and variance σ.
- $\text{Exp}(\lambda)$: exponential distribution with mean $\frac{1}{\lambda}$.
- $\Gamma(k, \lambda)$: gamma distribution with parameters k and λ.
- ...



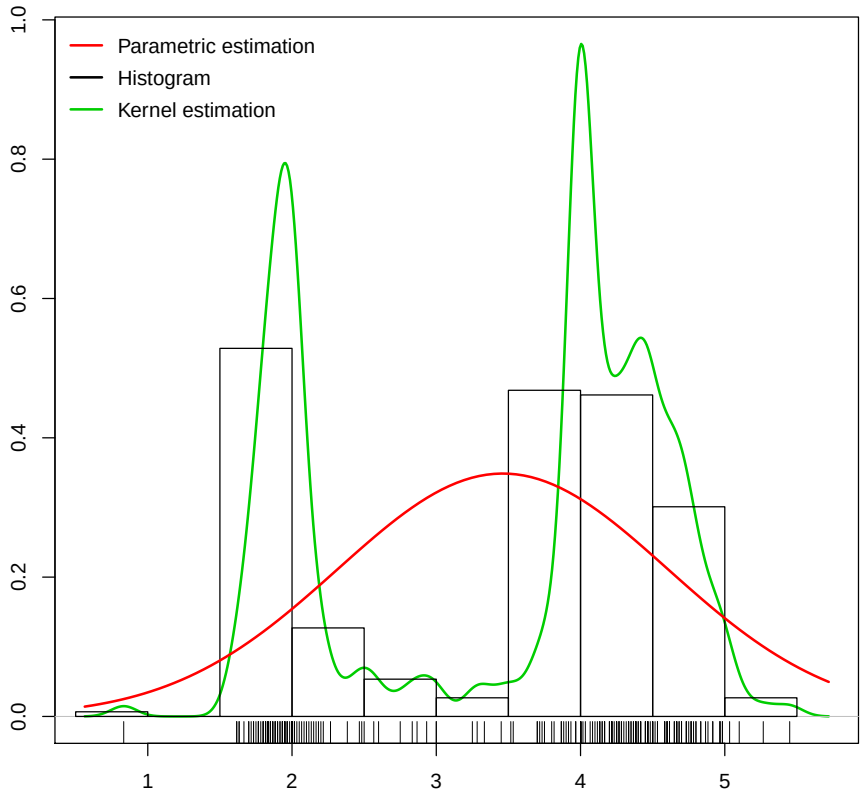


Back

Some ideas on Kernel Density Estimation

Parametric vs Nonparametric estimation

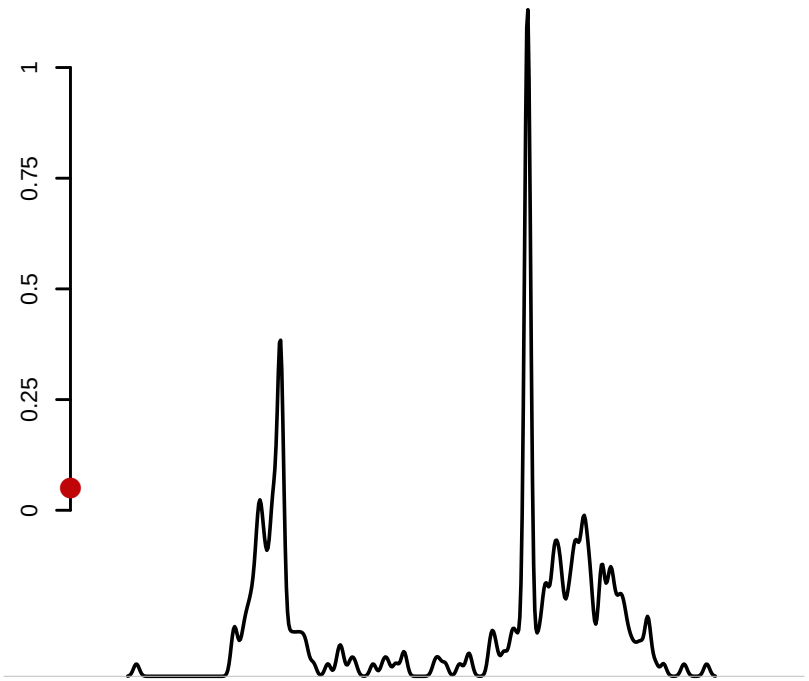
Let start with an example...



Kernel density estimation

$$\widehat{f}_{nh}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)$$

- Fixed kernel function, K.
- Bandwidth parameter, h.
  - Silverman (1986).
  - Bowman (1984).
  - Sheather & Jones (1991).



Back

Generate random numbers

Random numbers are numbers that occur in a sequence such that two conditions are met:

1. the values are uniformly distributed over a defined interval or set.
2. it is impossible to predict future values based on past or present ones.

Different methods

- Lehmer generator.
- Mid square method.
- Combined generators.
- Shuffled generators.

Gentle J. E. (2003). Random Number Generation and Monte Carlo Methods. Springer.

Back

Convergence notation

- $A_n = O(a_n)$ if there exists constants C and n_0 such that $|a_n| \leq Ca_n$ for $n \geq n_0$. Equivalently,

$$\limsup_{n\to\infty} \frac{|A_n|}{a_n} < \infty$$

- $A_n = o(a_n)$ if for every $\varepsilon > 0$ there exists n_ε such that

$$|A_n| \leq \varepsilon a_n$$

for $n \geq n_\varepsilon$.

- $X_n = O_p(a_n)$ if for every $\varepsilon > 0$ there exists constants C_ε and n_ε such that

$$P(|X_n| \leq C_\varepsilon a_n) > 1 - \varepsilon$$

for every $n \geq n_\varepsilon$.

- $X_n = o_p(a_n)$ if for every $\varepsilon > 0$ there exists n_ε such that

$$P(|X_n| \leq \varepsilon a_n) > 1 - \varepsilon$$

for every $n \geq n_\varepsilon$.

Back