INTRODUCCIÓN A LA COMPUTACIÓN BAYESIANA CON R

Big & Open Data: Análisis y Programación con R

José Manuel Velasco Cabo
mvelascc@ucm.es
http://github.com/manelvcmb

Taller del Curso Big & Open Data: Análisis y Programación con R
Escuela Complutense de Verano 2014-2015
Beatriz González Pérez, Victoria López López, Guadalupe Miñana Ropero, José Manuel Velasco Cabo
http://www.tecnologiaucm.es

Instalación OpenBugs

www.openbugs.net/w/Downloads

Linux Download

Source package

Should be usable on any x86 (PC) Linux platform. On 64 bit Linux, the necessary 32-bit C development packages are required.

Compilation has been successful on 64-bit Ubuntu using the g++-multilib package,

Download: OpenBUGS-3.2.3.tar.gz

To install this, unpack by typing

```
tar zxvf OpenBUGS-3.2.3.tar.gz
cd OpenBUGS-3.2.3
```

then compile and install by typing

```
./configure
make
sudo make install
```

Instalación JAGS

To install this, unpack by typing

```
tar zxvf OpenBUGS-3.2.3.tar.gz
cd OpenBUGS-3.2.3
```

then compile and install by typing

```
./configure
make
sudo make install
```

Instalación RSTAN

Dentro de Rstudio, en la consola de R:

```
source('http://mc-stan.org/rstan/install.R', echo = TRUE, max.deparse.length = 2000)
install_rstan()
```

Índice

- Teorema de Bayes. Inferencia Bayesiana
- Posterior y Prior Conjugados
- MCMC
  - BUGS
  - JAGS
  - STAN
  - MCMCPACK
- Diagnosis de Convergencia

Índice

- Posterior y Prior Conjugados
- Computación Bayesiana
  - Metodos Montecarlo
  - Metropolis
  - Gibbs
- MCMC
  - BUGS
  - JAGS
  - STAN
  - MCMCPACK
- Diagnosis de Convergencia

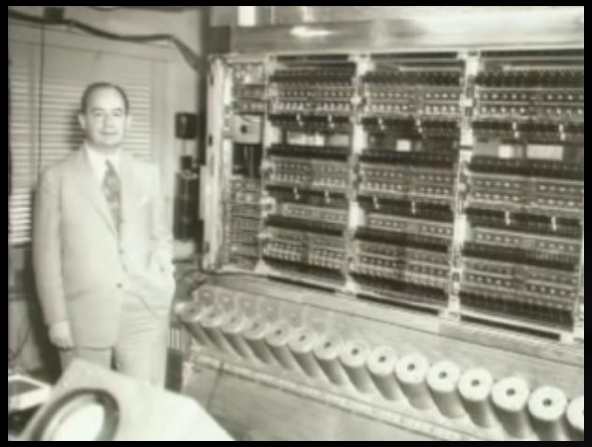
Stanislaw (Stan) Ulam

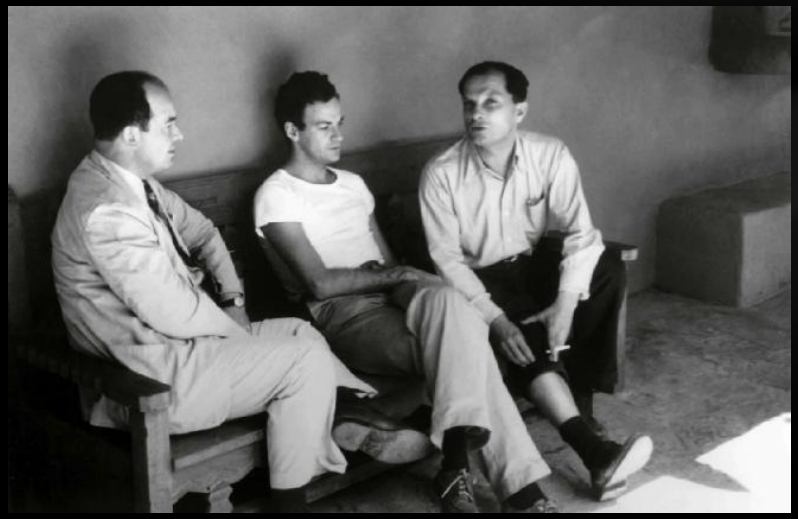
Once in my life I had a mathematical dream which proved correct. I was twenty years old. I thought, my God, this is wonderful, I won't have to work, it will all come in dreams. But it never happened again. . .

All my father does is think, think, think . . .

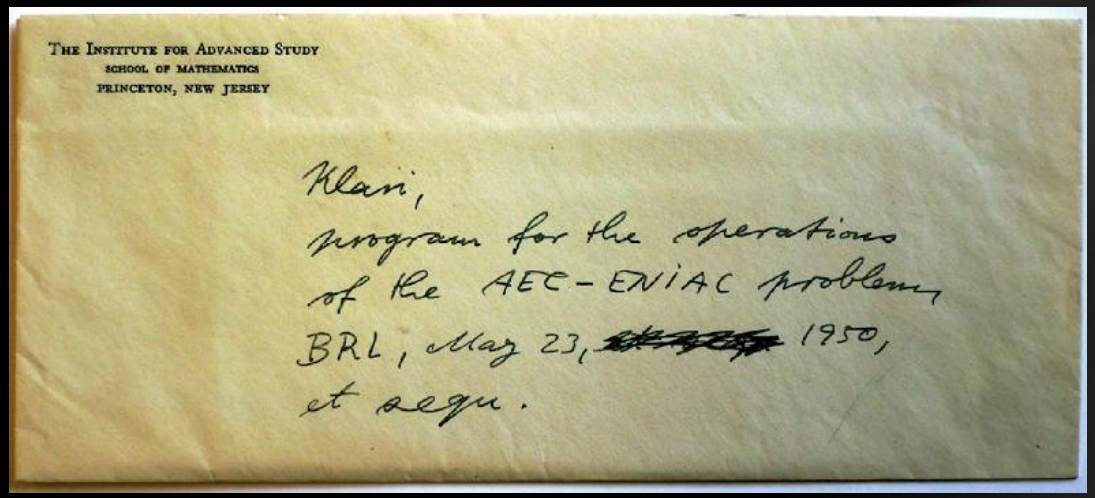
— Claire Ulam

John von Neumann





Klara von Neumann (Dan Eckart)



WOMAN SCIENTIST REPORTED A SUICIDE

SAN DIEGO, Calif., Nov. 11 (UPI)—Coroner's deputies listed as probable suicide today the drowning of Mrs. Klara Dan Eckart, 52 years old, who helped develop the atomic bomb.

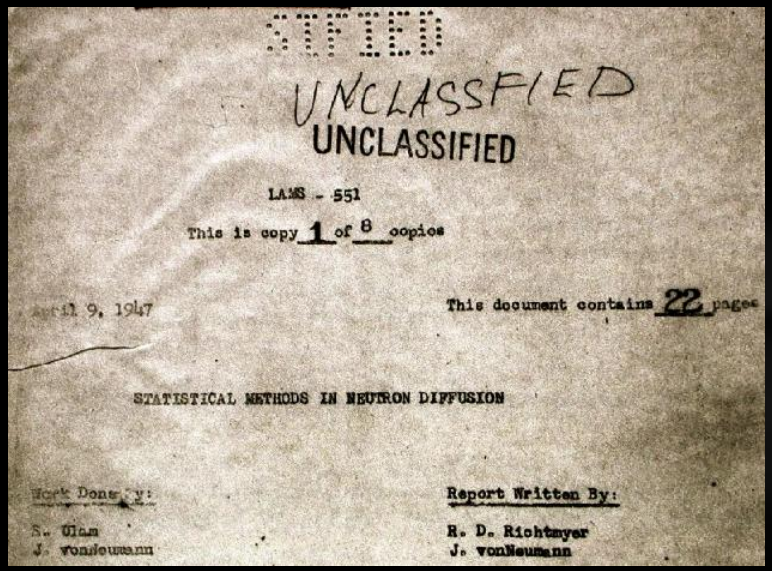
Authorities said that Mrs. Eckart, a native of Budapest, waded into the surf off San Diego yesterday.

Mrs. Eckart was the widow of John Von Neumann and the wife of Dr. Carl H. Eckart, physics professor at the University of California at San Diego.

Dr. Von Neumann and Mrs. Eckart worked on theoretical problems at the Aberdeen, Md., Proving Grounds and at the Los Alamos, N. M., atomic laboratory in World War II.

Klara Dan Eckart's education did not extend beyond the standard secondary school of Europe. However, through diligent self-

Statistical Methods in Neutron Diffusion



UNCLASSIFIED

UNCLASSIFIED

LAMS - 551

This is copy 1 of 8 copies

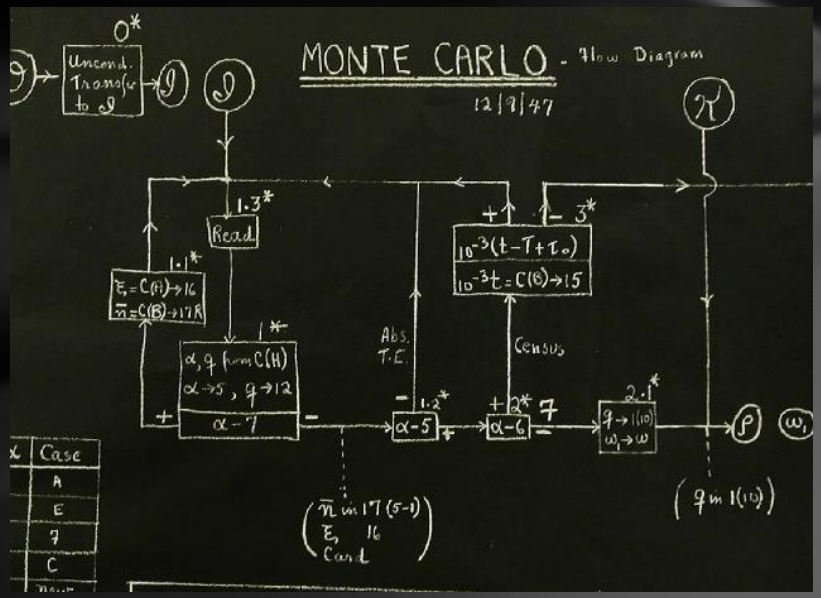
April 9, 1947

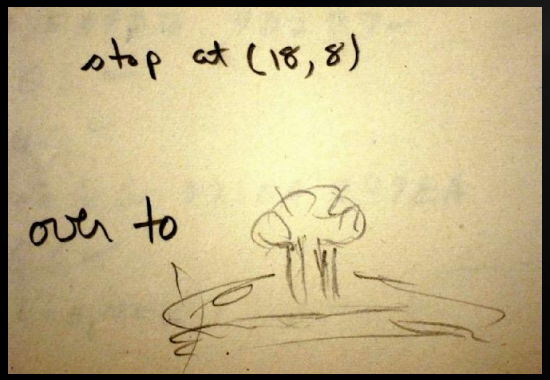
This document contains 22 pages

STATISTICAL METHODS IN NEUTRON DIFFUSION

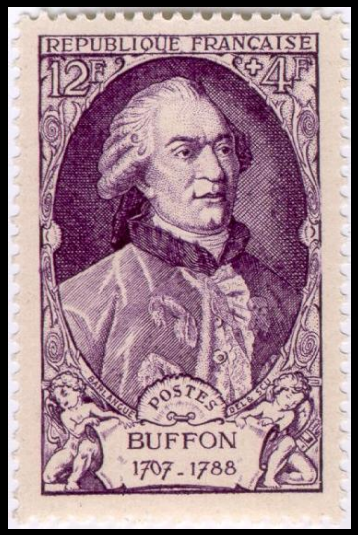
Work Done By:
S. Ulam
J. vonNeumann

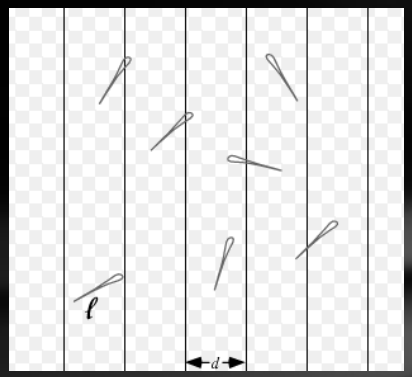
Report Written By:
R. D. Richtmyer
J. vonNeumann

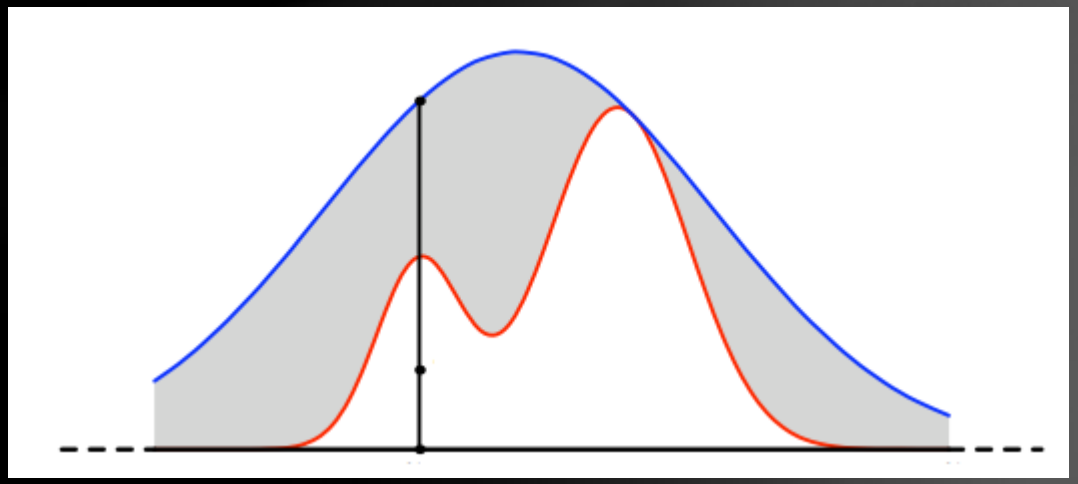




Aguja de Bufón - Método GRID







Metropolis

Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller (1953) Equations of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, **21**, 1087–1092.

Paseo Aleatorio por las Canarias

- Unos políticos quieren hacer campaña en las islas Canarias.
- Quieren dedicar a cada isla un tiempo proporcional a su población
- Sin embargo, estudiaron estadística en un par de tardes y no les dio tiempo a aprender a normalizar

- ¿Qué pueden hacer?

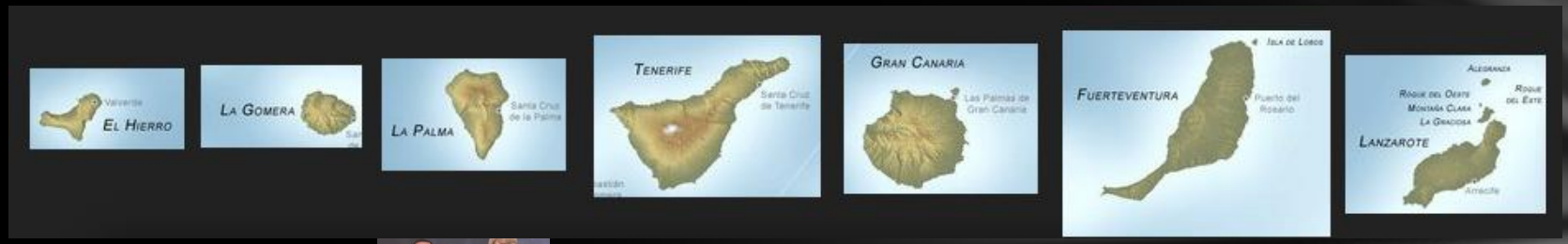
Idea de John Kruschke

Paseo Aleatorio por las Canarias

Paseo Aleatorio por las Canarias

Empezamos en una isla cualquiera. Por ejemplo: La Palma.

Paseo Aleatorio por las Canarias



Elegimos al azar una de las islas adyacentes. Por ejemplo: Tenerife

La población de Tenerife es mayor que la de La Palma → Nos movemos a Tenerife

Paseo Aleatorio por las Canarias

De nuevo, elegimos al azar una de las islas adyacentes.
Por ejemplo: Gran Canaria

La población de Tenerife es mayor que la de Gran Canaria →

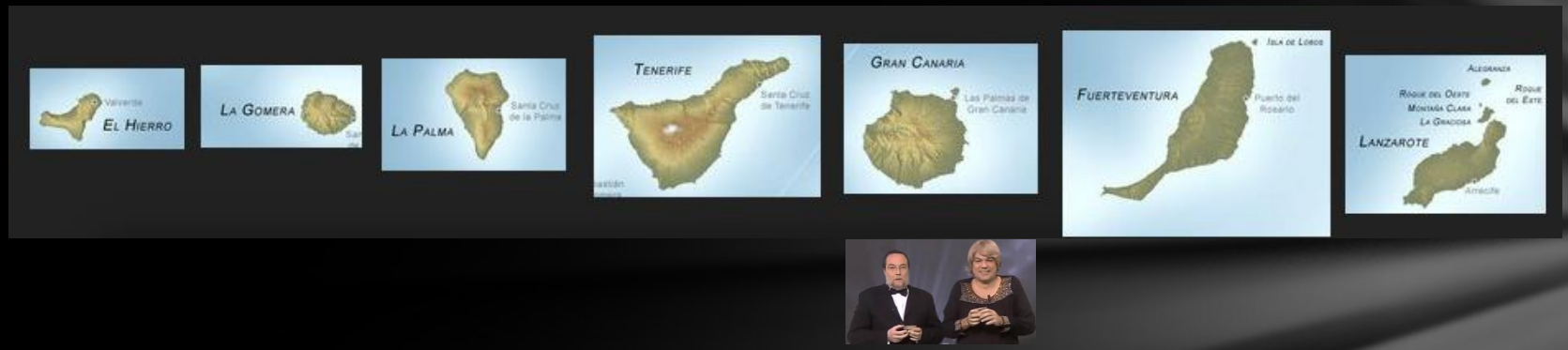
$$\frac{\text{Población de Gran Canaria}}{\text{Población de Tenerife}} = 0.9$$

Paseo Aleatorio por las Canarias

Elegimos un número aleatorio entre 0 y 1 → 0.7

$$\frac{\text{Población de Gran Canaria}}{\text{Población de Tenerife}} = 0.9 > 0.7 \rightarrow \text{Nos movemos a Gran Canaria}$$

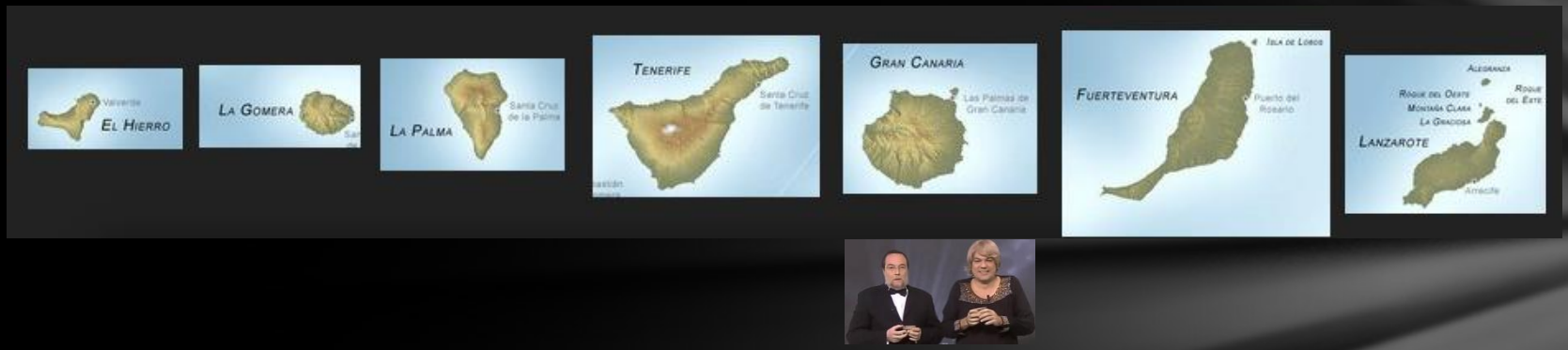
Paseo Aleatorio por las Canarias



Elegimos una isla adyacente al azar → Fuerteventura

De nuevo, la población de Gran Canaria es mayor que la de Fuerteventura →

Paseo Aleatorio por las Canarias

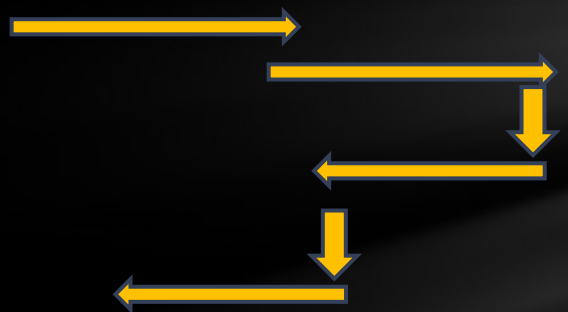


Elegimos un número aleatorio entre 0 y 1 → 0.4

$$\frac{\text{Población de Fuerteventura}}{\text{Población de Gran Canaria}} = 0.1 < 0.4 \rightarrow \text{Nos quedamos en Gran Canaria}$$

¡Rechazamos el movimiento a Fuerteventura!

Paseo Aleatorio por las Canarias



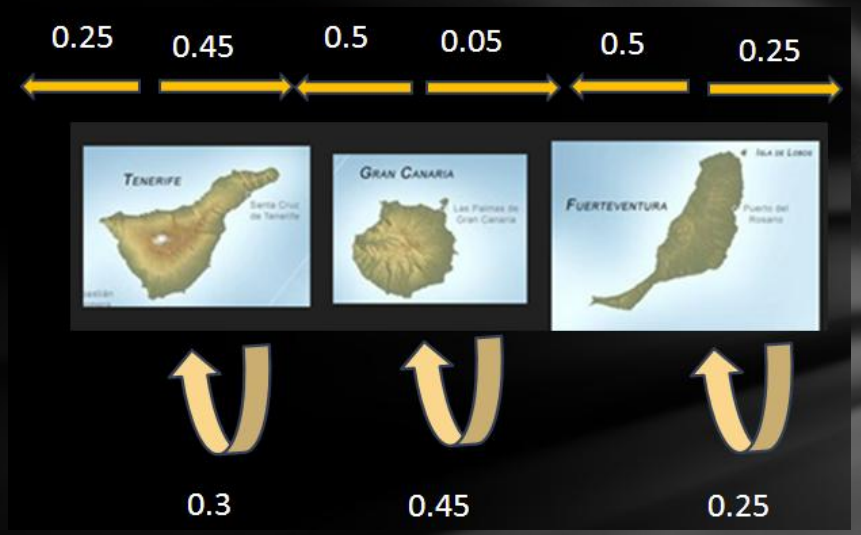
Paseo Aleatorio por las Canarias

Si el paseo es lo suficientemente largo, al final el tiempo que pasan en cada isla es proporcional a su población relativa

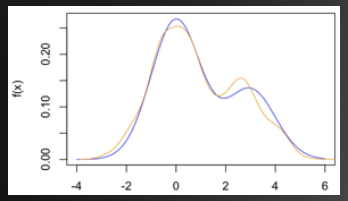
Cadena de Markov

0.3 0.45 0.25

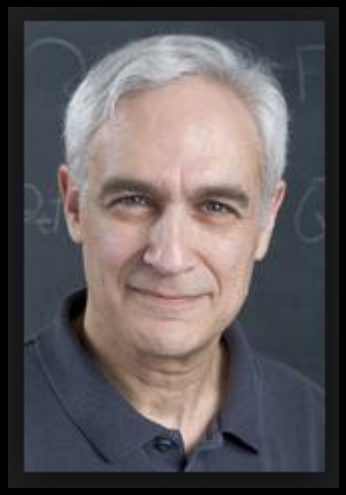
Ergodicidad

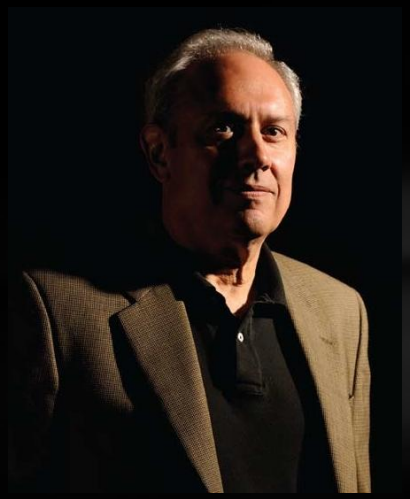


$$\frac{1}{n}\sum_{j=1}^{n} f \circ T^{j}(x) \to \int f \, d\, m$$

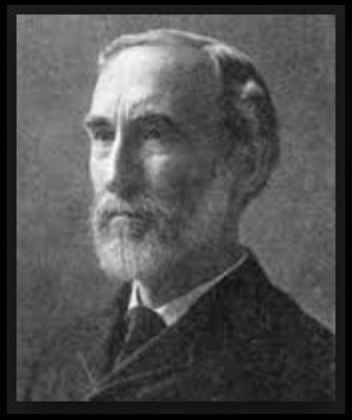


Gibbs Sampling



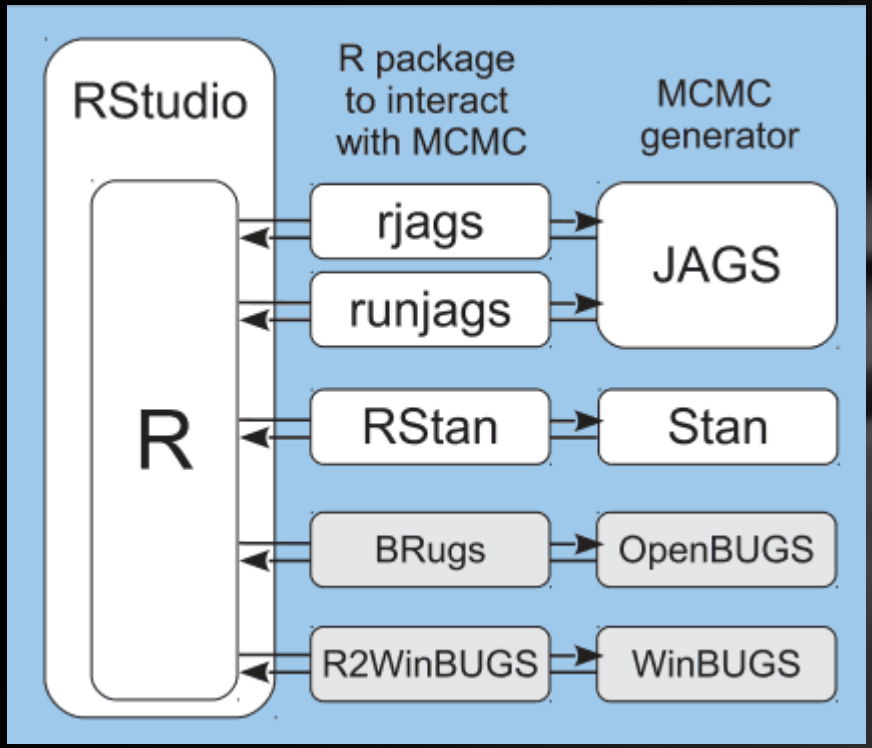


Geman and Geman (1984) Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Trans. Pattern Anal. Machine Intelligence*, **6**, 721–741.*



$$\theta_1^{(j)} \sim \pi\left(\theta_1 \middle| \theta_2^{(j-1)}, \theta_3^{(j-1)}, \ldots, \theta_k^{(j-1)}\right)$$
$$\theta_2^{(j)} \sim \pi\left(\theta_2 \middle| \theta_1^{(j)}, \theta_3^{(j-1)}, \ldots, \theta_k^{(j-1)}\right)$$
$$\theta_3^{(j)} \sim \pi\left(\theta_3 \middle| \theta_1^{(j)}, \theta_2^{(j)}, \theta_3^{(j-1)}, \ldots, \theta_k^{(j-1)}\right)$$
$$\vdots$$
$$\theta_k^{(j)} \sim \pi\left(\theta_k \middle| \theta_1^{(j)}, \theta_2^{(j)}, \ldots, \theta_{k-1}^{(j)}\right)$$

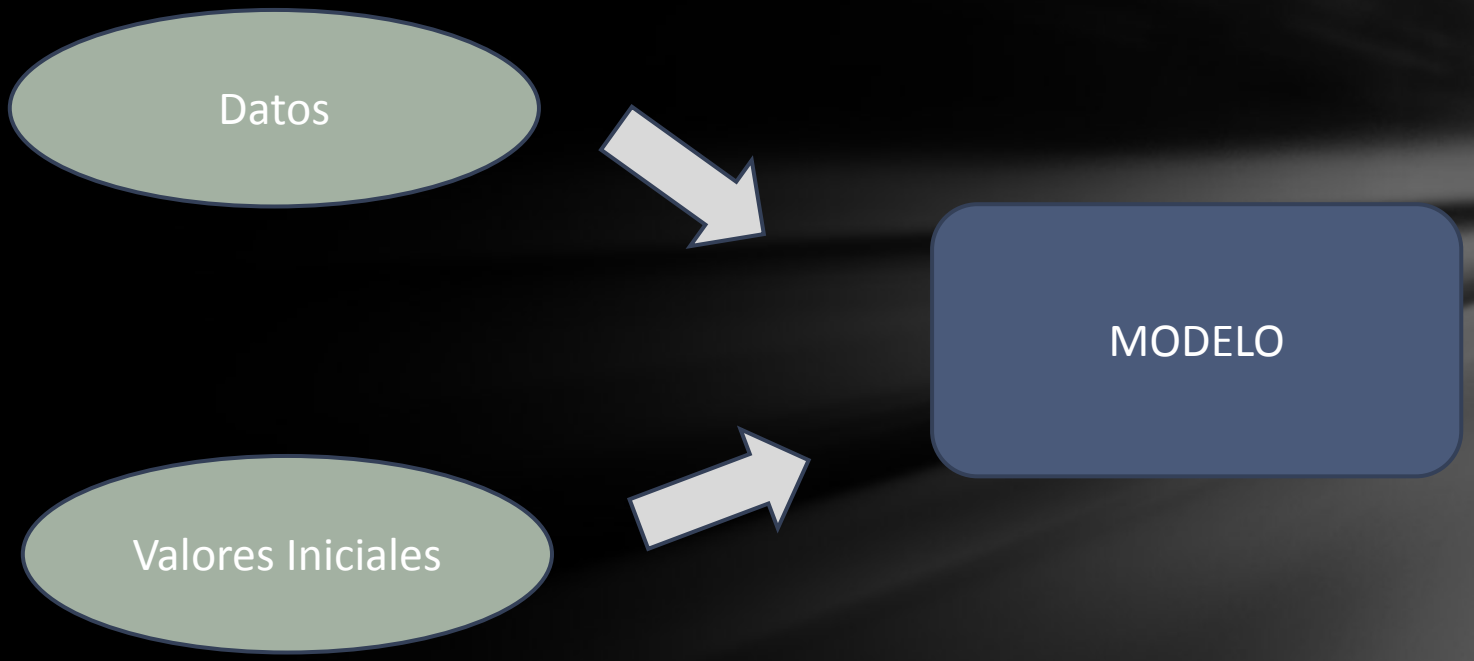
Interfaz entre R y librerias externas



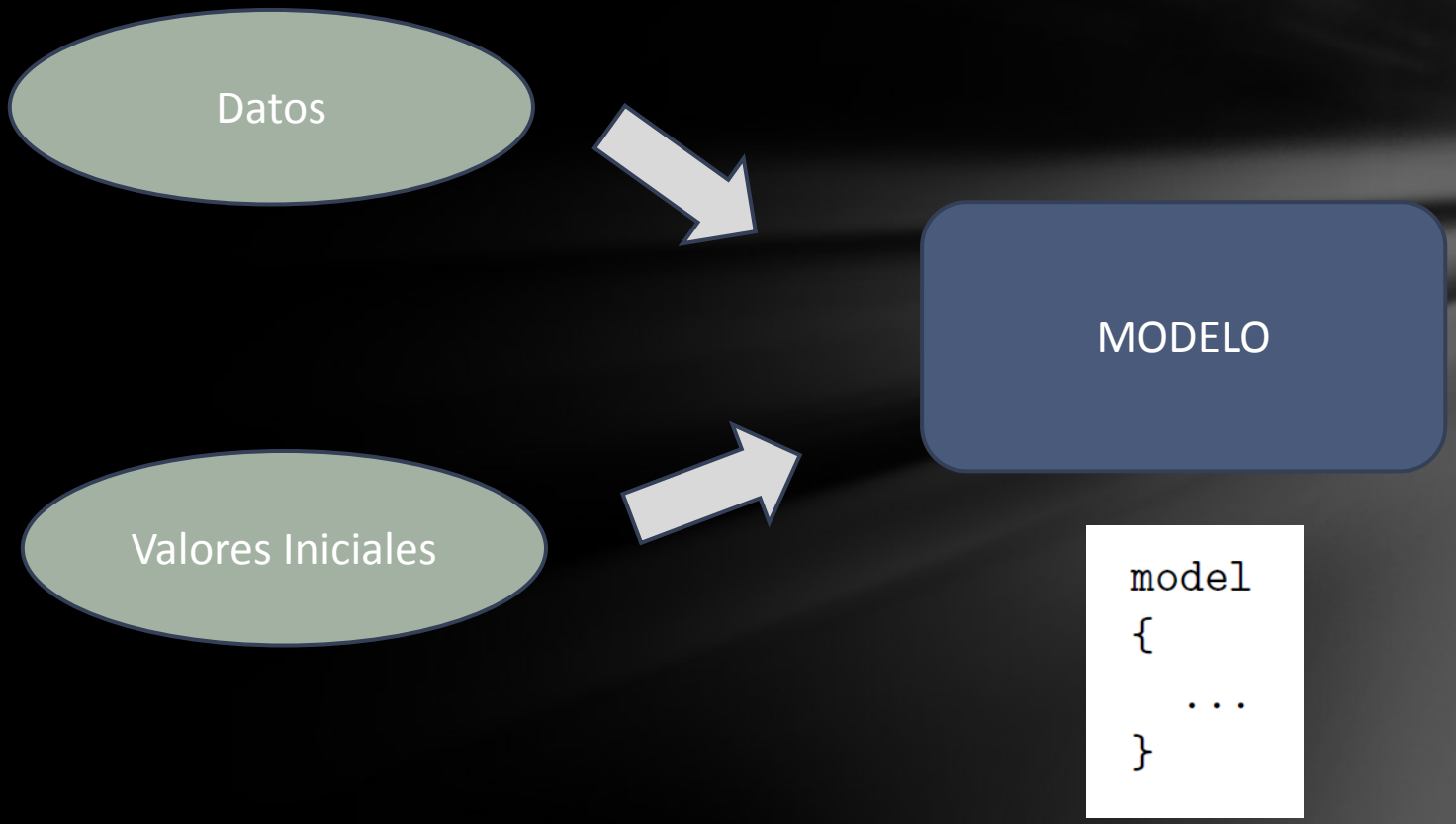
BUGS - JAGS

- BUGS → Bayesian inference Using Gibbs Sampling
- JAGS → Just Another Gibbs Sampler
- Stan → Stan (Ulam) → Hamiltonian Monte Carlo

BUGS - JAGS



BUGS - JAGS



BUGS - JAGS

MODELO

```
model
{
   ...
}
```

```
alpha <- 1
```

Asignación determinística

BUGS - JAGS

MODELO

```
model
{
   ...
}
```

```
alpha <- 1
```

Asignación determinística

```
p ~ dbeta(1, 1)
```

Asignación estocástica

BUGS - JAGS

MODELO

```
model
{
   ...
}
```

```
alpha <- 1
```

Asignación determinística

```
p ~ dbeta(1, 1)
```

Asignación estocástica

```
for (i in 1:N)
{
  x[i] ~ dbern(p)
}
```

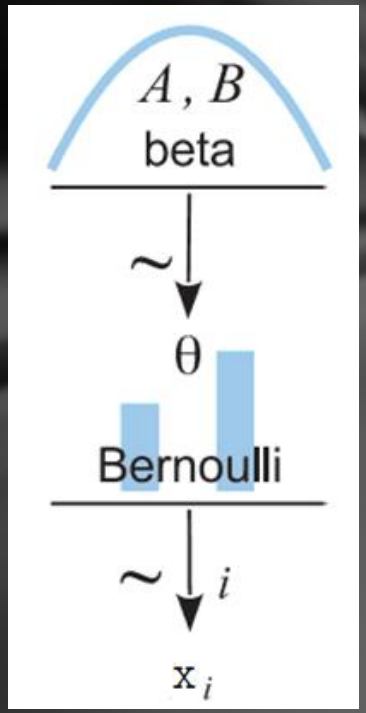
Bucles

BUGS - JAGS

MODELO

```
model
{
   ...
}
```

```
model
{
  alpha <- 1
  beta <- 1

  θ ~ dbeta(alpha, beta)

  for (i in 1:N)
  {
    x[i] ~ dbern(p)
  }
}
```



¡BUGS es un lenguaje descriptivo!

BUGS - JAGS

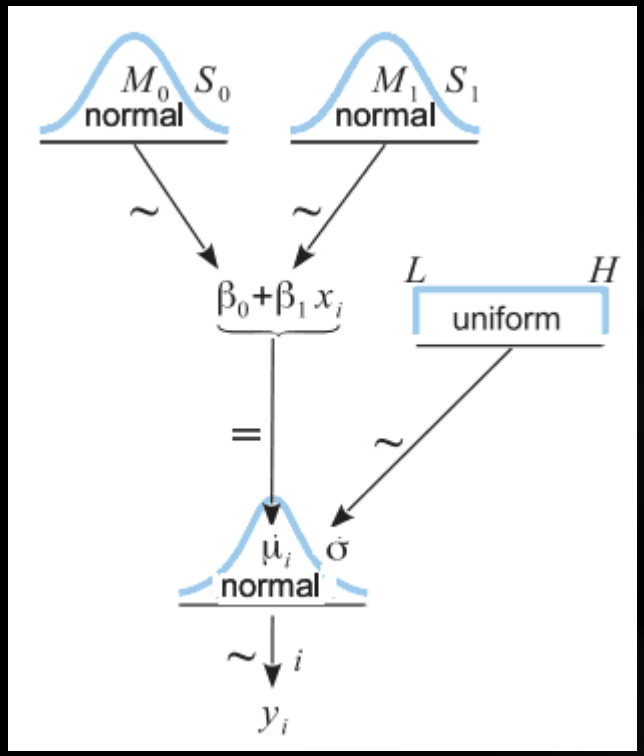
$$y_i = ax_i + b + \epsilon_i$$

$$\epsilon_i \sim \mathcal{N}(0, 1)$$

$$y_i \sim \mathcal{N}(ax_i + b, 1)$$

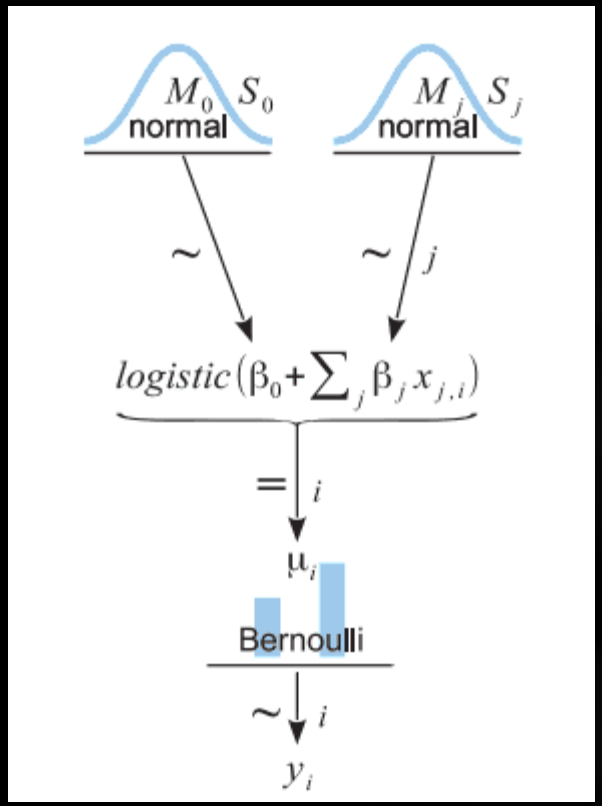
```
y[i] ~ dnorm(a * x[i] + b, 1)
```

JAGS – Regresión Lineal



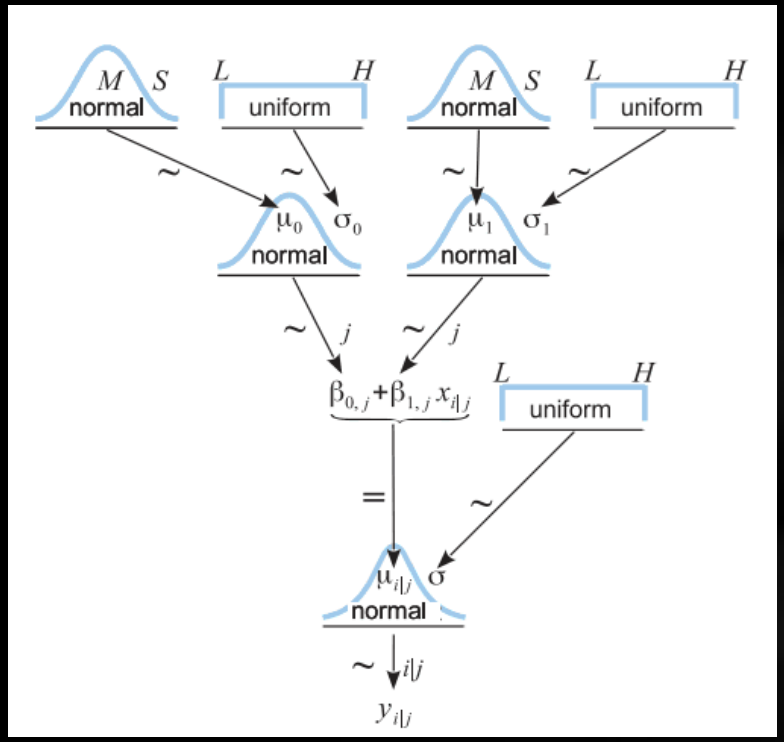

```
model
{
  a ~ dnorm(0, 0.0001)
  b ~ dnorm(0, 0.0001)

  tau <- pow(sigma, -2)
  sigma ~ dunif(0, 100)

  for (i in 1:N)
  {
    mu[i] <- a * x[i] + b
    y[i] ~ dnorm(mu[i], tau)
  }
}
```

JAGS – Regresión Logística



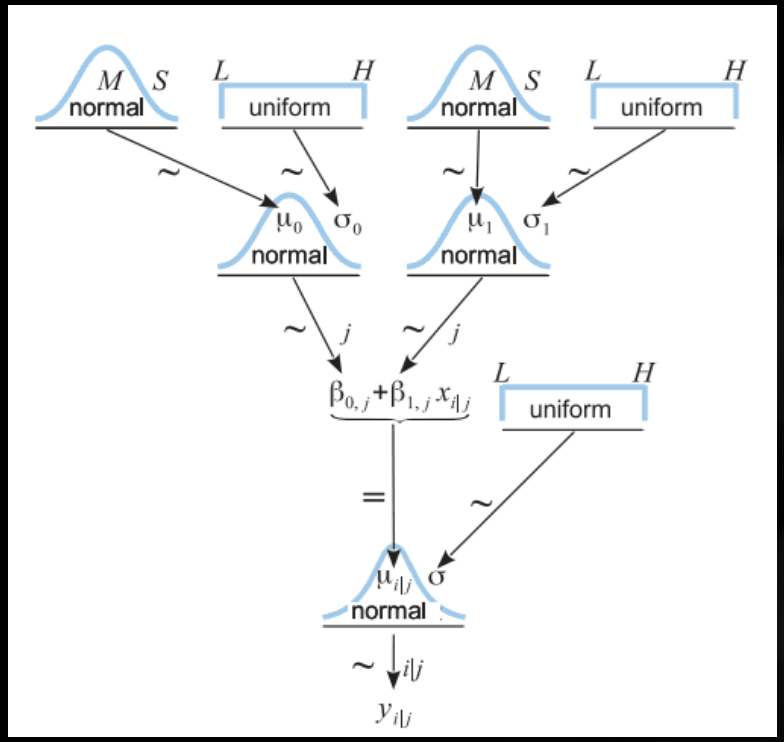

```
model
{
  a ~ dnorm(0, 0.0001)
  b ~ dnorm(0, 0.0001)

  for (i in 1:N)
  {
    y[i] ~ dbern(p[i])
    logit(p[i]) <- a * x[i] + b
  }
}
```

JAGS – Regresión Lineal Jerárquica




```
model
{
  mu.a ~ dnorm(0, 0.0001)
  mu.b ~ dnorm(0, 0.0001)
  ...

  for (j in 1:K)
  {
    a[j] ~ dnorm(mu.a, tau.a)
    b[j] ~ dnorm(mu.b, tau.b)
  }
  for (i in 1:N)
  {
    mu[i] <- a[g[i]] * x[i] + b[g[i]]
    y[i] ~ dnorm(mu[i], tau)
  }
}
```

JAGS – Regresión Lineal Jerárquica

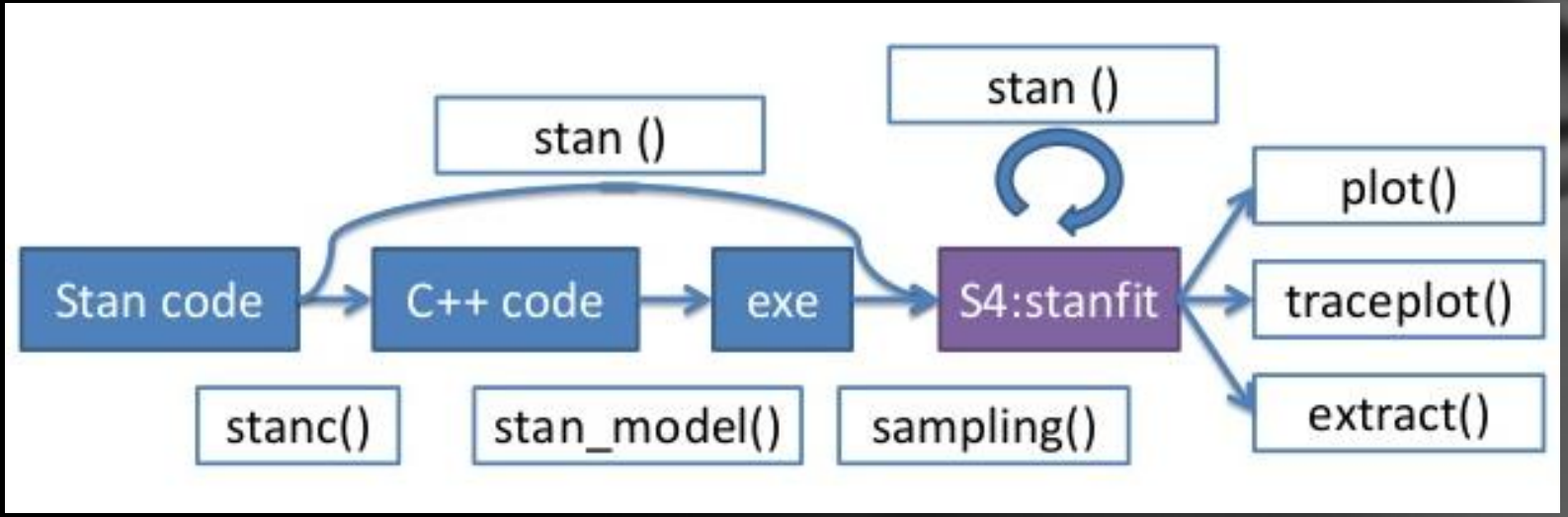



```
model
{
  for (i in 1:N)
  {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- a[g[i]] * x[i] + b[g[i]]
  }

  for (j in 1:K)
  {
    a[j] ~ dnorm(mu.a, tau.a)
    b[j] ~ dnorm(mu.b, tau.b)
  }
     mu.a ~ dnorm(0, 0.0001)
     mu.b ~ dnorm(0, 0.0001)

     tau <- pow(sigma, -2)
     sigma ~ dunif(0, 1000)

     tau.a <- pow(sigma.a, -2)
     tau.b <- pow(sigma.b, -2)
     sigma.a ~ dunif(0, 1000)
     sigma.b ~ dunif(0, 1000)
  }
```

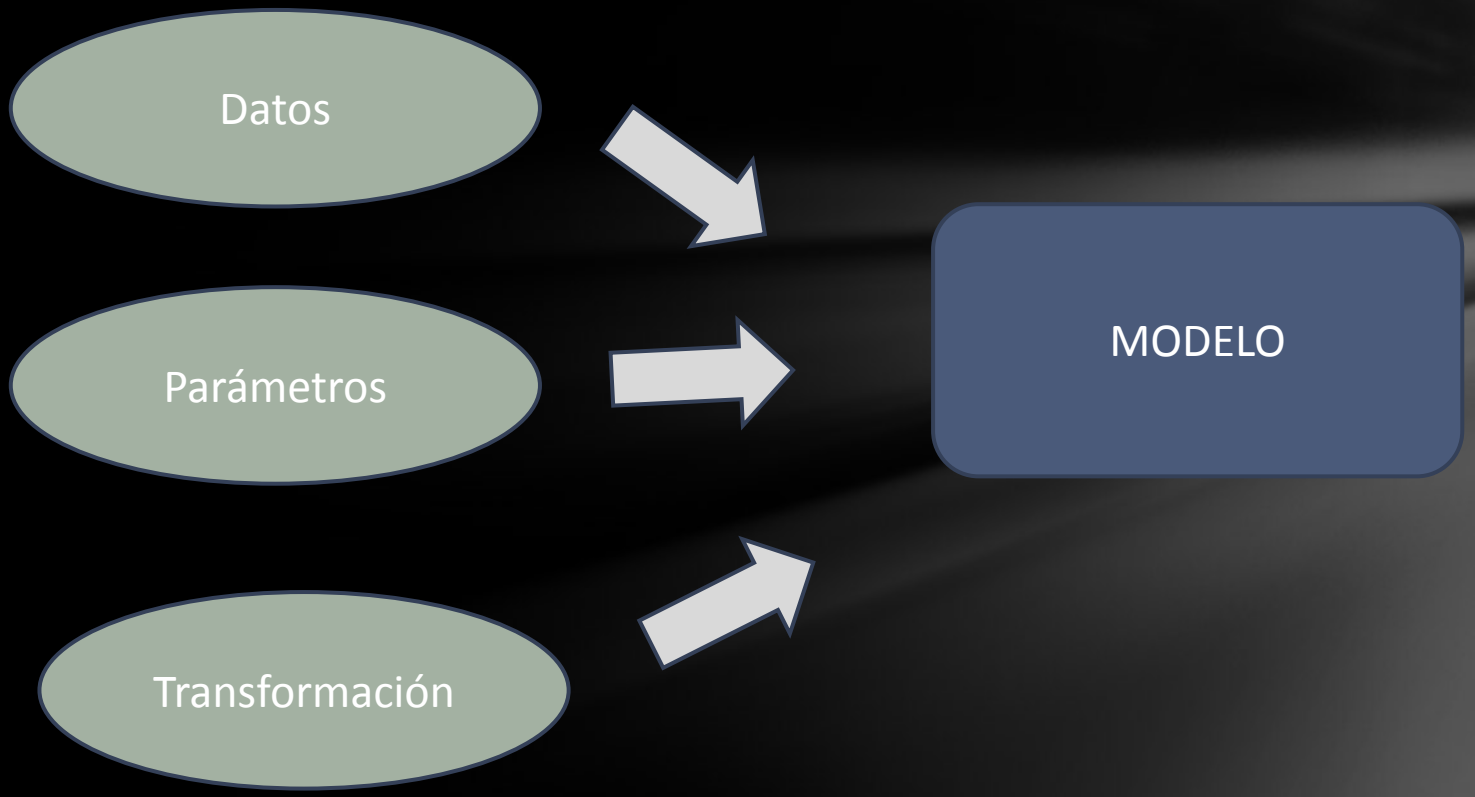
Recopilación de funciones

Distribution	Density	Distribution	Quantile
Bernoulli	dbern	pbern	qbern
Beta	dbeta	pbeta	qbeta
Binomial	dbin	pbin	qbin
Chi-square	dchisqr	pchisqr	qchisqr
Double exponential	ddexp	pdexp	qdexp
Exponential	dexp	pexp	qexp
F	df	pf	qf
Gamma	dgamma	pgamma	qgamma
Generalized gamma	dgen.gamma	pgen.gamma	qgen.gamma
Noncentral hypergeometric	dhyper	phyper	qhyper
Logistic	dlogis	plogis	qlogis
Log-normal	dlnorm	plnorm	qlnorm
Negative binomial	dnegbin	pnegbin	qnegbin
Noncentral Chi-square	dnchisqr	pnchisqr	qnchisqr
Normal	dnorm	pnorm	qnorm
Pareto	dpar	ppar	qpar
Poisson	dpois	ppois	qpois
Student t	dt	pt	qt
Weibull	dweib	pweib	qweib

STAN

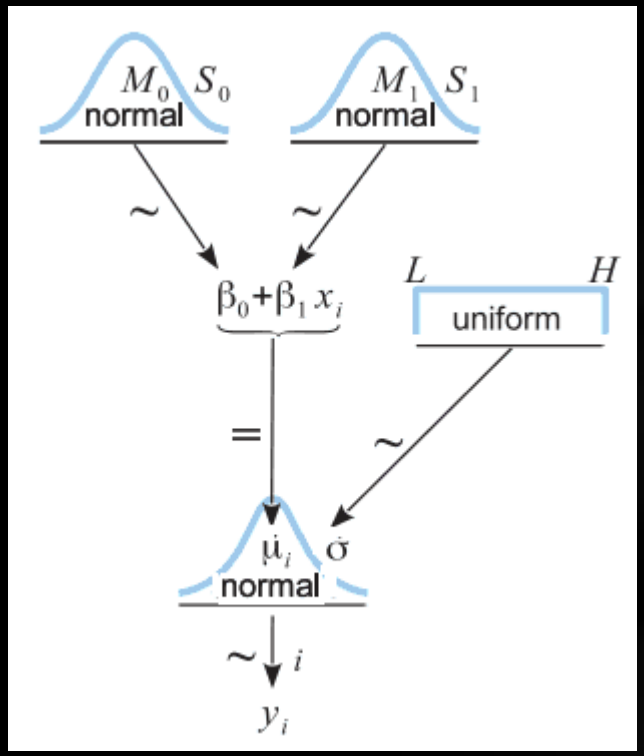

stan ()
stan ()
Stan code
C++ code
exe
S4:stanfit
plot()
traceplot()
extract()
stanc()
stan_model()
sampling()

STAN



¡Stan es un lenguaje imperativo!

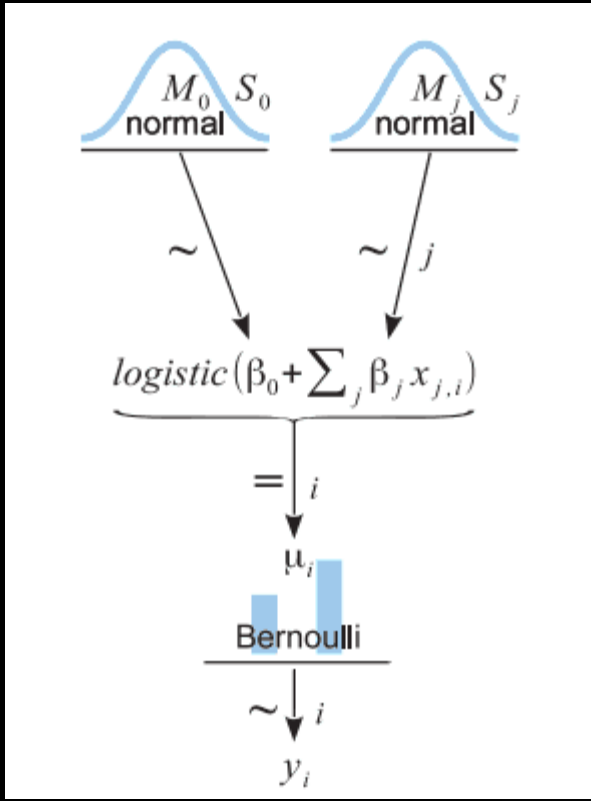
STAN – Regresión Lineal




```
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;
}
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma;
}
model {
  for (n in 1:N)
    y[n] ~ normal(alpha + beta * x[n], sigma);
}
```

```
model {
  y ~ normal(alpha + beta * x, sigma);
}
```

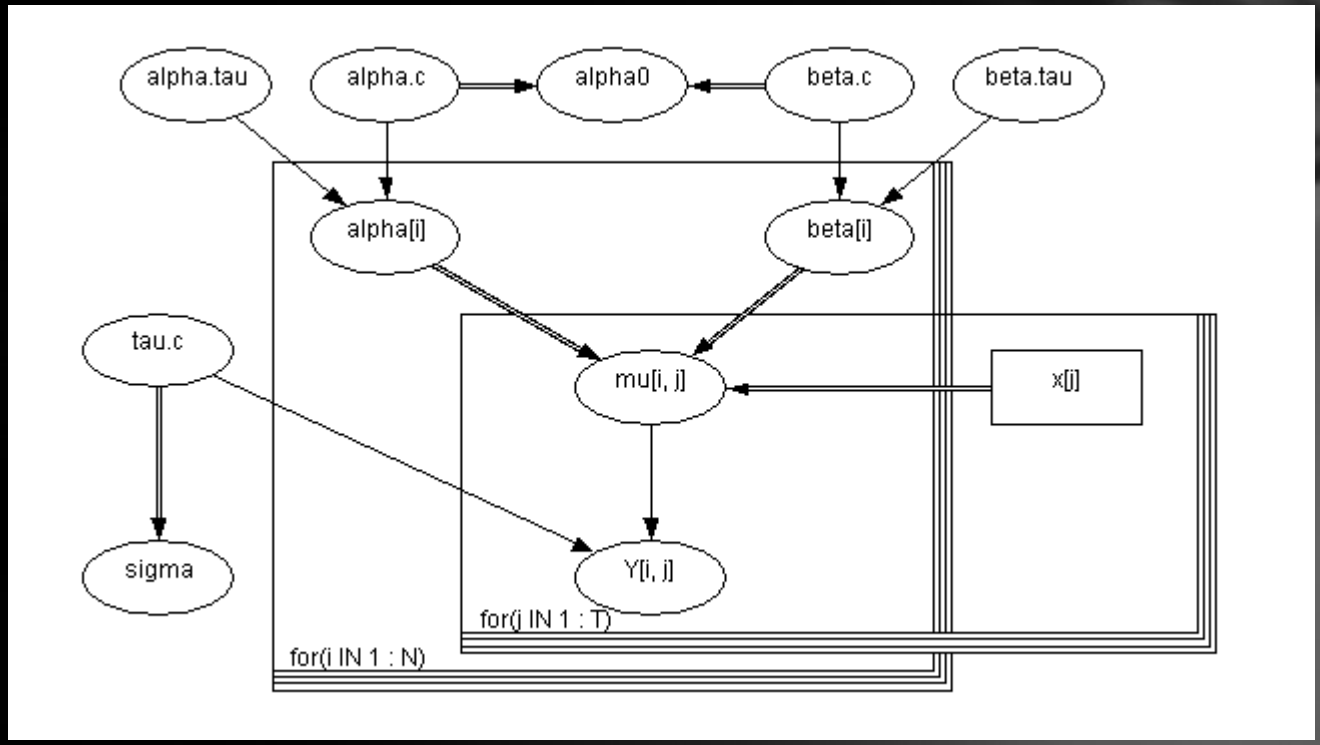
STAN – Regresión Logística




```
data {
  int<lower=0> N;
  vector[N] x;
  int<lower=0,upper=1> y[N];
}
parameters {
  real alpha;
  real beta;
}
model {
  y ~ bernoulli_logit(alpha + beta * x);
}
```

Ratas – Regresión Lineal jerárquica

30 ratas son pesadas durante cinco semanas



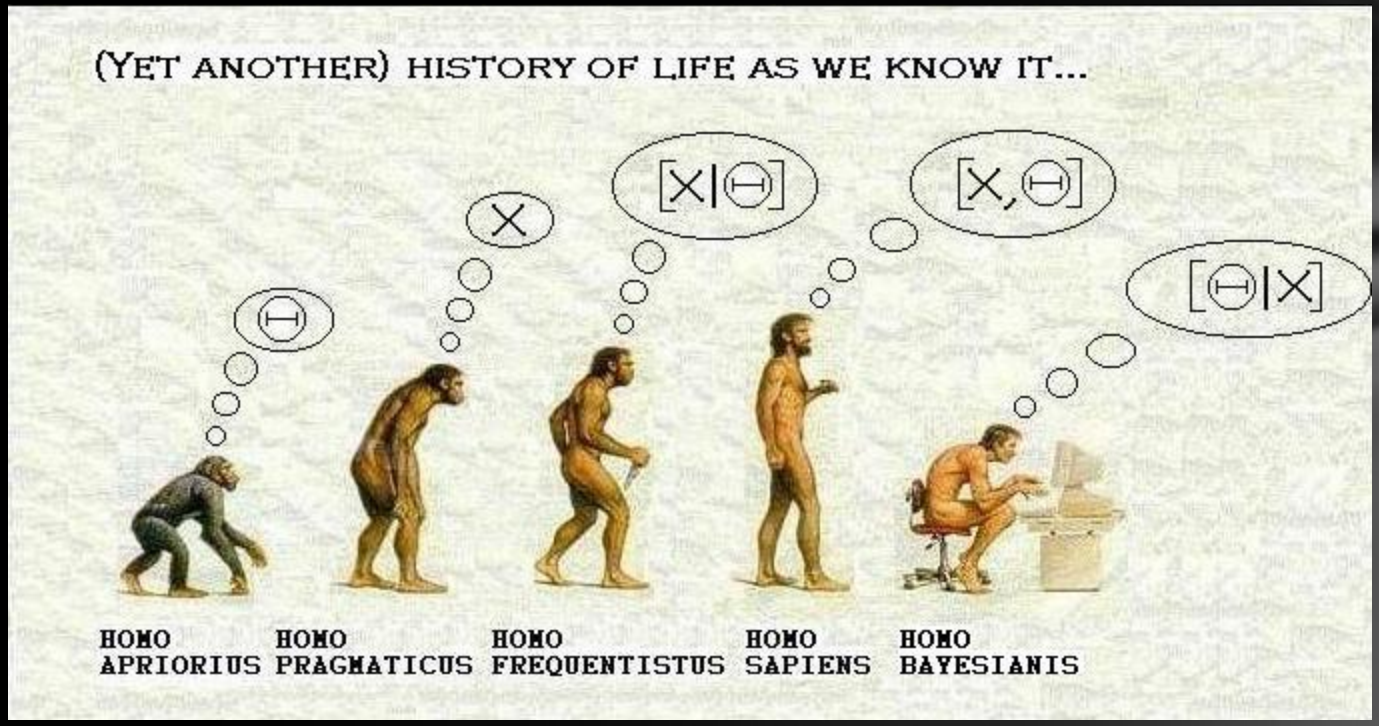
BUGS

```
model
{
  for( i in 1 : N ) {
    for( j in 1 : T ) {
      Y[i , j] ~ dnorm(mu[i , j],tau.c)
      mu[i , j] <- alpha[i] + beta[i] * (x[j] - xbar)
      culmative.Y[i , j] <- culmative(Y[i , j], Y[i , j])
      post.pv.Y[i , j] <- post.p.value(Y[i , j])
      prior.pv.Y[i , j] <- prior.p.value(Y[i , j])
      replicate.post.Y[i , j] <- replicate.post(Y[i , j])
      pv.post.Y[i , j] <- step(Y[i , j] - replicate.post.Y[i , j])
      replicate.prior.Y[i , j] <- replicate.prior(Y[i , j])
      pv.prior.Y[i , j] <- step(Y[i , j] - replicate.prior.Y[i , j])
    }
    alpha[i] ~ dnorm(alpha.c,alpha.tau)
    beta[i] ~ dnorm(beta.c,beta.tau)
  }
  tau.c ~ dgamma(0.001,0.001)
  sigma <- 1 / sqrt(tau.c)
  alpha.c ~ dnorm(0.0,1.0E-6)
  alpha.tau ~ dgamma(0.001,0.001)
  beta.c ~ dnorm(0.0,1.0E-6)
  beta.tau ~ dgamma(0.001,0.001)
  alpha0 <- alpha.c - xbar * beta.c
}
```

STAN

```
data {
  int<lower=0> N;
  int<lower=0> T;
  real x[T];
  real y[N,T];
  real xbar;
}
parameters {
  real alpha[N];
  real beta[N];

  real mu_alpha;
  real mu_beta;

  real<lower=0> sigmasq_y;
  real<lower=0> sigmasq_alpha;
  real<lower=0> sigmasq_beta;
}
transformed parameters {
  real<lower=0> sigma_y;
  real<lower=0> sigma_alpha;
  real<lower=0> sigma_beta;

  sigma_y <- sqrt(sigmasq_y);
  sigma_alpha <- sqrt(sigmasq_alpha);
  sigma_beta <- sqrt(sigmasq_beta);
}
model {
  mu_alpha ~ normal(0, 100);
  mu_beta ~ normal(0, 100);
  sigmasq_y ~ inv_gamma(0.001, 0.001);
  sigmasq_alpha ~ inv_gamma(0.001, 0.001);
  sigmasq_beta ~ inv_gamma(0.001, 0.001);
  alpha ~ normal(mu_alpha, sigma_alpha); // vectorized
  beta ~ normal(mu_beta, sigma_beta);  // vectorized
  for (n in 1:N)
    for (t in 1:T)
      y[n,t] ~ normal(alpha[n] + beta[n] * (x[t] - xbar), sigma_y);
}
```

Conclusión


(YET ANOTHER) HISTORY OF LIFE AS WE KNOW IT...
[X|Θ]
[X,Θ]
[Θ|X]
X
Θ
HOMO APRIORIUS
HOMO PRAGMATICUS
HOMO FREQUENTISTUS
HOMO SAPIENS
HOMO BAYESIANIS