Programação Estatística

Introdução ao R - Simulação de dados

Rachid Muleia, PhD in Statistics

Universidade Eduardo Mondlane
Faculdade de Ciências
Departamento de Matemática e Informática

2023-04-21

Números aleatórios

Estudos de simulação são experimentos de computador que envolvem a criação de dados por amostragem pseudo-aleatória a partir de distribuições de probabilidade conhecidas.

- Simulação de dados é um tópico bastante importante na estatística
- Pode ser usado a avaliar a performance de um modelo estatístico/matemático
- Construir a distribuição empírica
- Nem sempre é possível ter uma solução analítica para uma distribuição de probabilidade
- R oferece vários recursos para fazer simulação ou gerar números aleatórios

Números aleatórios

Para gerar números aleatórios de uma distribuição de probabilidades pode-se usar as seguintes funções

- `rnorm` : gera variáveis aleatórias Normal com uma determinada média e desvio padrão
- `pnorm`: avalia a função de distribuição cumulativa para uma distribuição Normal
- `dnorm`: avalia a densidade da distribuição normal de probabilidade
- `rpois`: gera valores de uma distribuição de Poisson com uma determinada taxa

Números aleatórios

As distribuições de probabilidade geralmente são acompanhadas de quatro funções. Essas funções levam os seguintes prefixos:

- `d` para densidade
- `r` para gerar números aleatórios
- `p` para distribuição cumulativa
- `q` devolve o valor de z para uma determinada probabilidade

Números aleatórios- Distribuição normal

```
> x <- rnorm(20)
> x
##  [1] -0.86178300 -1.08529260 -0.05082519  1.99831483  1.01857129 -0.40461906
##  [7]  0.94965034 -1.14411739 -1.66841514 -0.74350959  0.53708728  0.99408728
## [13]  0.56651479  1.47355181  1.14080714  0.02639987 -0.07287376 -0.85293799
## [19]  2.18233831 -1.77865588
> x <- rnorm(20, 20, 2)
> x
##  [1] 18.87576 21.23584 21.95492 20.11536 19.95713 19.98438 19.79428 20.53959
##  [9] 19.52306 19.39213 22.45404 20.54159 22.95826 17.16114 13.45543 20.43130
## [17] 17.53185 19.08953 20.58143 23.65813
```

Números aleatórios- Distribuição normal

Ao gerar números aleatórios é importante especificar a semente usando a função `set.seed`. Isto garante a "reprodutibilidade"

```
> set.seed(2)
> x <- rnorm(20)
> x
##  [1] -0.89691455  0.18484918  1.58784533 -1.13037567 -0.08025176  0.13242028
##  [7]  0.70795473 -0.23969802  1.98447394 -0.13878701  0.41765075  0.98175278
## [13] -0.39269536 -1.03966898  1.78222896 -2.31106908  0.87860458  0.03580672
## [19]  1.01282869  0.43226515
> x <- rnorm(20, 20, 2)
> x
##  [1] 24.18164 17.60015 23.17928 23.90930 20.00988 15.09659 20.95447 18.80688
##  [9] 21.58441 20.57927 21.47788 20.63792 22.15233 19.43168 18.44665 18.80868
## [17] 16.54804 18.19483 18.88188 19.50697
> summary(x)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   15.10   18.72   19.76   20.00   21.50   24.18
```

Números aleatórios- Distribuição normal

Calculo da probabilidade: $X \sim N(100, 2^2)$

- Calcule $P(92 < X < 105) = P(X < 105) - P(X < 92)$

```
> pnorm(105, mean = 100, sd=2)- pnorm(92,mean = 100,sd=2)
## [1] 0.9937587
```

- Calcule $P(X > 110) = 1 - P(X < 110)$

```
> 1- pnorm(110, mean=100, sd=2) # 1-P(X<110)
## [1] 2.866516e-07
> pnorm(110, mean=100, sd=2, lower.tail = FALSE) # P(X>110)
## [1] 2.866516e-07
```

Números aleatórios - Distribuição de Poisson

```
> rpois(10, 1)
##  [1] 0 1 0 1 0 2 3 0 1 0
> rpois(10, 2)
##  [1] 5 1 1 2 2 1 2 0 0 2
> rpois(20,10)
##  [1]  7 13 10  9  6  8 12 13 15  4 13 12  8  8 14 15 11 12  9
```

Distribuição de probabilidade cumulativa

```
> ppois(2,2) # P( X <=2)
## [1] 0.6766764
> ppois(4,2)
## [1] 0.947347
```

Números aleatórios- modelo de regressão linear

Suponha que queira simular dados a partir do seguinte modelo linear

$$y = \beta_0 + \beta_1 + \epsilon$$

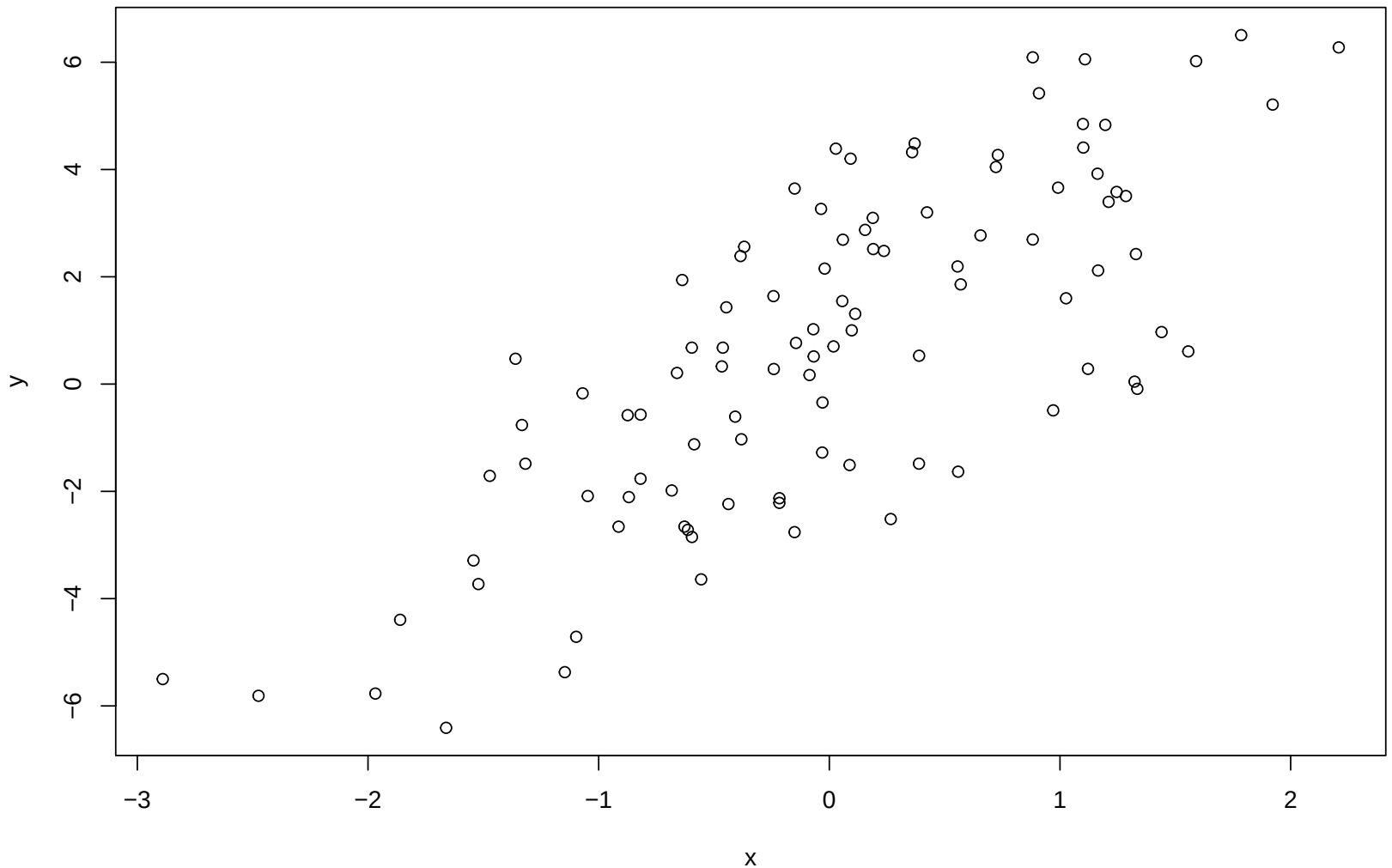
,

onde $\epsilon \sim N(0, 2^2)$. Assume-se igualmente que $x \sim N(0, 1^2)$, $\beta_0 = 0.5$ e $\beta_1 = 2$

```
> set.seed(20)
> x <- rnorm(100)
> e <- rnorm(100, 0, 2)
> y <- 0.5 + 2 * x + e
> summary(y)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -6.4084 -1.5402  0.6789  0.6893  2.9303  6.5052
```

Números aleatórios- modelo de regressão linear

```
> plot(x, y)
```



Números aleatórios- modelo de regressão linear

Suponha que queira simular dados a partir do seguinte modelo linear

$$y = \beta_0 + \beta_1 + \epsilon$$

,

onde $\epsilon \sim N(0, 2^2)$. Assume-se igualmente que $x \sim N(0, 1^2)$, $\beta_0 = 0.5$ e $\beta_1 = 2$

```
> sim_model <- lm(y~x)
> summary(sim_model)
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9170 -1.3303  0.1328  1.5261  3.6446
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.6777     0.1983   3.417 0.000922 ***
## x             2.3596     0.2013  11.719  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Números aleatórios a partir de um conjunto de dados

Para gerar números aleatórios a partir de um conjunto de dados pode-se usar a função `sample`

```
> set.seed(10)
> sample(1:10, 4)
## [1] 9 7 8 6
> sample(letters, 5)
## [1] "g" "s" "x" "o" "w"
> sample(1:10) ## permutacao
##  [1] 10  7  2  8  5  3  9  4  1  6
> sample(1:10)
##  [1]  5  9  4  2 10  7  1  6  8  3
> sample(1:10, replace = TRUE) # com resposicao
##  [1] 10  2  9  4  2  8  3  6  2  1
```

Bootstrap

Geralmente é usado quando a distribuição de uma determinada estatística é desconhecida

```
> data(iris)
> attach(iris)
> summary(Sepal.Length)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.300   5.100   5.800   5.843   6.400   7.900
> sample.size <- dim(iris)[1]
> se.est <- sd(Sepal.Length)/sqrt(sample.size)
> t.test(with(iris, Sepal.Length))
##
##  One Sample t-test
##
## data:  with(iris, Sepal.Length)
## t = 86.425, df = 149, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  5.709732 5.976934
## sample estimates:
## mean of x
##  5.843333
```

Bootstrap - Média

Geralmente é usado quando a distribuição de uma determinada estatística é desconhecida

```
> data(iris)
> attach(iris)
## The following objects are masked from iris (pos = 3):
##
##     Petal.Length, Petal.Width, Sepal.Length, Sepal.Width, Species
> sample.size <- dim(iris)[1]
> nr.bootstrap.samples <- 500
> set.seed(111)
> b.est <- matrix(nrow=nr.bootstrap.samples,ncol=1)
> for (i in 1:nr.bootstrap.samples){
+ # print(i)
+ sel <- sample(1:sample.size,sample.size,replace=T)
+ b.est[i,] <- mean(Sepal.Length[sel])
+ }
>
> # comprar os erros- padrao
> se.boot <- sd(b.est)
> c(se.est, se.boot)
## [1] 0.06761132 0.06738434
```

Bootstrap- Mediana

```
> data(iris)
> attach(iris)
## The following objects are masked from iris (pos = 3):
##
##     Petal.Length, Petal.Width, Sepal.Length, Sepal.Width, Species
## The following objects are masked from iris (pos = 4):
##
##     Petal.Length, Petal.Width, Sepal.Length, Sepal.Width, Species
> sample.size <- dim(iris)[1]
> nr.bootstrap.samples <- 500
> set.seed(111)
> b.est <- matrix(nrow=nr.bootstrap.samples,ncol=1)
> for (i in 1:nr.bootstrap.samples){
+ # print(i)
+ sel <- sample(1:sample.size,sample.size,replace=T)
+ b.est[i,] <- median(Sepal.Length[sel])
+ }
>
> # erro padrao
>
> se.mediana <-sd(b.est)
> # intervalo de confianca
> quantile(b.est, prob=c(0.025, 0.975))
##  2.5% 97.5%
##   5.6   6.0
```