

Disciplina: Estatística Computacional II

Lista: Simulado de Prova

Data da lista:	08/10 - 16/10
Preceptor(a):	Zamir Jose Rodriguez Ortega
Curso(s) atendido(s):	Estatística
Orientador(a):	Marcos Vinicius de Oliveira Peres

Exercício 1: Distribuição de Gompertz

ITEM (a)

```
set.seed(123)
```

```
# Parâmetros
```

```
n <- 300
```

```
b <- 0.5
```

```
c <- 2
```

```
# Função quantílica da Gompertz
```

```
qgompertz <- function(u, b, c) {  
  (1 / c) * log(1 - (c / b) * log(1 - u))  
}
```

```
# Simulação pelo método da inversa
```

```
u <- runif(n)
```

```
amostra <- qgompertz(u, b, c)
```

```
# Resumo da amostra
```

```
summary(amostra)
```

```
sd(amostra)
```

```
# Visualização
```

```
hist(amostra,  
  probability = TRUE,  
  col = "lightblue",  
  border = "white",  
  main = "Amostra da distribuição de Gompertz",  
  xlab = "Valores simulados",  
  ylab = "Densidade")
```

ITEM (b)

```
library(maxLik)

# Função de log-verossimilhança
log_veross <- function(par) {

  b <- exp(par[1])
  c <- exp(par[2])

  n <- length(amostra)

  logL <- n * log(b) +
    c * sum(amostra) -
    (b / c) * sum(expm1(c * amostra))

  return(logL)
}

# Estimação por máxima verossimilhança
ajuste <- maxLik(
  logLik = log_veross,
  start = log(c(b = 0.5, c = 2)),
  method = "BFGS"
)

# Resultados
summary(ajuste)

# Parâmetros estimados
estimativas <- exp(coef(ajuste))
names(estimativas) <- c("b", "c")

print(estimativas)
```

ITEM (c)

```
# Parâmetros estimados
b_est <- estimativas["b"]
c_est <- estimativas["c"]

# Função densidade de Gompertz
dgomperz <- function(x, b, c) {
  b * exp(c * x) *
    exp(-(b / c) * expm1(c * x))
}
```

```

# Histograma
hist(amostra,
     probability = TRUE,
     col = "lightblue",
     border = "white",
     main = "Ajuste da distribuição de Gompertz",
     xlab = "Valores de X",
     ylab = "Densidade")

# Curva ajustada
curve(dgompertz(x, b_est, c_est),
      add = TRUE,
      col = "red",
      lwd = 2)

# Curva verdadeira
curve(dgompertz(x, 0.5, 2),
      add = TRUE,
      col = "blue",
      lwd = 2,
      lty = 2)

legend("topright",
      legend = c("Densidade estimada",
                  "Densidade verdadeira"),
      col = c("red", "blue"),
      lty = c(1, 2),
      lwd = 2)

```

Exercício 2: Curva de crescimento de Gompertz

ITEM (a)

```

set.seed(123)

# Parâmetros verdadeiros
A <- 10
B <- 2
C <- 0.6

# Valores da variável explicativa
n <- 100
x <- seq(0, 10, length.out = n)

# Erros aleatórios
erro <- rnorm(n, mean = 0, sd = 1)

# Simulação da resposta
y <- A * exp(-B * exp(-C * x)) + erro

```

```

# Banco de dados
dados <- data.frame(x, y)

# Resumo
summary(dados)

# Gráfico de dispersão
plot(x, y,
     pch = 19,
     col = "darkblue",
     main = "Crescimento pela curva de Gompertz",
     xlab = "Tempo",
     ylab = "Crescimento observado")

```

ITEM (b)

```

# Ajuste do modelo não linear
modelo <- nls(
  y ~ A * exp(-B * exp(-C * x)),
  data = dados,
  start = list(A = 9, B = 1.5, C = 0.5),
  algorithm = "port",
  lower = c(A = 0, B = 0, C = 0)
)

# Resultados do ajuste
summary(modelo)

# Parâmetros estimados
coef(modelo)

# Gráfico com curva ajustada
plot(x, y,
     pch = 19,
     col = "gray50",
     main = "Ajuste da curva de Gompertz",
     xlab = "Tempo",
     ylab = "Crescimento")

# Valores ajustados
lines(x, predict(modelo),
     col = "red",
     lwd = 2)

legend("bottomright",
     legend = c("Dados observados",
                 "Curva ajustada"),
     col = c("gray50", "red"),
     pch = c(19, NA),

```

```
lty = c(NA, 1),  
lwd = c(NA, 2))
```

ITEM (c)

```
# Parâmetros verdadeiros  
verdadeiros <- c(A = 10, B = 2, C = 0.6)  
  
# Parâmetros estimados  
estimados <- coef(modelo)  
  
# Tabela comparativa  
comparacao <- data.frame(  
  Parametro = names(verdadeiros),  
  Verdadeiro = as.numeric(verdadeiros),  
  Estimado = as.numeric(estimados),  
  Diferenca = as.numeric(estimados - verdadeiros),  
  Erro_Relativo = as.numeric(  
    abs(estimados - verdadeiros) / verdadeiros * 100  
  )  
)  
  
print(round(comparacao[, -1], 4))  
  
# Tabela completa  
print(comparacao)
```

Exercício 3: Análise de sobrevivência com dados colon

ITEM (a)

```
library(survival)  
  
# Carregar os dados  
data("colon", package = "survival")  
  
# Selecionar eventos observados  
dados_colon <- subset(  
  colon,  
  status == 1 & !is.na(time) & !is.na(node4)  
)  
  
# Manter variáveis de interesse  
dados_colon <- dados_colon[, c("time", "node4")]  
  
# Resumo dos dados  
summary(dados_colon)
```

```

# Quantidade por grupo
table(dados_colon$node4)

# Estatísticas descritivas por grupo
aggregate(time ~ node4,
  data = dados_colon,
  FUN = function(x) {
    c(n = length(x),
      media = mean(x),
      mediana = median(x),
      dp = sd(x),
      minimo = min(x),
      maximo = max(x))
  })

# Boxplot por grupo
boxplot(time ~ node4,
  data = dados_colon,
  col = c("lightblue", "lightgreen"),
  main = "Tempo observado por grupo",
  xlab = "Mais de 4 linfonodos afetados",
  ylab = "Tempo de sobrevivência",
  names = c("Não (0)", "Sim (1)"))

# Histograma por grupo
par(mfrow = c(1, 2))

hist(dados_colon$time[dados_colon$node4 == 0],
  col = "lightblue",
  main = "Grupo node4 = 0",
  xlab = "Tempo",
  ylab = "Frequência")

hist(dados_colon$time[dados_colon$node4 == 1],
  col = "lightgreen",
  main = "Grupo node4 = 1",
  xlab = "Tempo",
  ylab = "Frequência")

par(mfrow = c(1, 1))

```

ITEM (b)

```

library(maxLik)

# Dados
tempo <- dados_colon$time
grupo <- dados_colon$node4

```

```

# -----
# MODELO GOMPERTZ
# -----

loglik_gomp <- function(par) {

  beta0 <- par[1]
  beta1 <- par[2]
  c <- exp(par[3])

  b <- exp(beta0 + beta1 * grupo)

  logf <- log(b) + c * tempo -
    (b / c) * expm1(c * tempo)

  return(logf)
}

# Valores iniciais
taxa0 <- 1 / mean(tempo[grupo == 0])
taxa1 <- 1 / mean(tempo[grupo == 1])

inicio <- c(
  beta0 = log(taxa0),
  beta1 = log(taxa1 / taxa0),
  logc = log(0.0001)
)

# Ajuste Gompertz
ajuste_gomp <- maxLik(
  logLik = loglik_gomp,
  start = inicio,
  method = "BFGS"
)

summary(ajuste_gomp)

# -----
# MODELO EXPONENCIAL
# -----

loglik_exp <- function(par) {

  beta0 <- par[1]
  beta1 <- par[2]

  lambda <- exp(beta0 + beta1 * grupo)

  logf <- log(lambda) - lambda * tempo

```

```

    return(logf)
  }

# Ajuste Exponencial
ajuste_exp <- maxLik(
  logLik = loglik_exp,
  start = inicio[1:2],
  method = "BFGS"
)

summary(ajuste_exp)

# Parâmetros estimados
coef(ajuste_gomp)
coef(ajuste_exp)

# ITEM (c)

# Coeficientes Gompertz
beta_gomp <- coef(ajuste_gomp)

# Coeficientes Exponencial
beta_exp <- coef(ajuste_exp)

# Razão de taxas
RT_gomp <- exp(beta_gomp[2])
RT_exp <- exp(beta_exp[2])

# Tabela comparativa
resultados <- data.frame(
  Modelo = c("Gompertz", "Exponencial"),
  Beta0 = c(beta_gomp[1], beta_exp[1]),
  Beta1 = c(beta_gomp[2], beta_exp[2]),
  Razao_Taxas = c(RT_gomp, RT_exp)
)

print(resultados)

# Interpretação automática
interpretar <- function(rt, modelo) {

  if (rt > 1) {
    cat(modelo, ": o grupo node4 = 1 apresenta taxa",
        round((rt - 1) * 100, 2),
        "% maior que o grupo node4 = 0.\n")
  } else {
    cat(modelo, ": o grupo node4 = 1 apresenta taxa",
        round((1 - rt) * 100, 2),
        "% menor que o grupo node4 = 0.\n")
  }
}

```

```
}  
}
```

```
interpretar(RT_gomp, "Gompertz")  
interpretar(RT_exp, "Exponencial")
```

ITEM (d)

```
# Função densidade Gompertz
```

```
dgomp <- function(x, b, c) {  
  b * exp(c * x - (b / c) * expm1(c * x))  
}
```

```
# Parâmetros Gompertz
```

```
b0_g <- exp(beta_gomp[1])  
b1_g <- exp(beta_gomp[1] + beta_gomp[2])  
c_g <- exp(beta_gomp[3])
```

```
# Parâmetros Exponencial
```

```
lambda0 <- exp(beta_exp[1])  
lambda1 <- exp(beta_exp[1] + beta_exp[2])
```

```
# Configuração dos gráficos
```

```
par(mfrow = c(1, 2))
```

```
for (g in 0:1) {
```

```
  # Dados do grupo
```

```
  tempos <- tempo[grupo == g]
```

```
  # Parâmetros do grupo
```

```
  b <- if (g == 0) b0_g else b1_g  
  lambda <- if (g == 0) lambda0 else lambda1
```

```
  # Histograma
```

```
  hist(tempos,  
       probability = TRUE,  
       col = if (g == 0) "lightblue" else "lightgreen",  
       border = "white",  
       main = paste("Grupo node4 =", g),  
       xlab = "Tempo de sobrevivência",  
       ylab = "Densidade")
```

```
  # Densidade Gompertz
```

```
  curve(dgomp(x, b, c_g),  
        add = TRUE,  
        col = "red",  
        lwd = 2)
```

```

# Densidade Exponencial
curve(dexp(x, rate = lambda),
      add = TRUE,
      col = "blue",
      lwd = 2,
      lty = 2)

legend("topright",
      legend = c("Gompertz", "Exponencial"),
      col = c("red", "blue"),
      lty = c(1, 2),
      lwd = 2,
      cex = 0.8)
}

par(mfrow = c(1, 1))

# ITEM (e)

# -----
# COMPARAÇÃO PELO AIC
# -----

# Log-verossimilhanças máximas
LL_gomp <- as.numeric(logLik(ajuste_gomp))
LL_exp <- as.numeric(logLik(ajuste_exp))

# Número de parâmetros
k_gomp <- 3
k_exp <- 2

# Cálculo do AIC
AIC_gomp <- -2 * LL_gomp + 2 * k_gomp
AIC_exp <- -2 * LL_exp + 2 * k_exp

# Tabela comparativa
comparacao <- data.frame(
  Modelo = c("Gompertz", "Exponencial"),
  LogVeross = c(LL_gomp, LL_exp),
  Parametros = c(k_gomp, k_exp),
  AIC = c(AIC_gomp, AIC_exp)
)

print(comparacao)

# Melhor modelo pelo AIC
melhor <- comparacao$Modelo[
  which.min(comparacao$AIC)
]

```

```
cat("Melhor modelo pelo AIC:", melhor, "\n")
```

```
# -----  
# COMPARAÇÃO PELO KS  
# -----
```

```
# Função acumulada Gompertz  
pgomp <- function(x, b, c) {  
  -expm1(-(b / c) * expm1(c * x))  
}
```

```
# Estatística KS descritiva  
ks_estatistica <- function(x, Fajustada) {
```

```
  x <- sort(x)  
  n <- length(x)  
  Fval <- Fajustada(x)
```

```
  Dmais <- max(seq_len(n) / n - Fval)  
  Dmenos <- max(Fval - (seq_len(n) - 1) / n)
```

```
  max(Dmais, Dmenos)  
}
```

```
# Cálculo por grupo  
ks_resultados <- data.frame()
```

```
for (g in 0:1) {
```

```
  xg <- tempo[grupo == g]
```

```
  bg <- if (g == 0) b0_g else b1_g  
  lg <- if (g == 0) lambda0 else lambda1
```

```
  ks_g <- ks_estatistica(  
    xg, function(x) pgomp(x, bg, c_g)  
  )
```

```
  ks_e <- ks_estatistica(  
    xg, function(x) pexp(x, rate = lg)  
  )
```

```
  ks_resultados <- rbind(  
    ks_resultados,  
    data.frame(  
      Grupo = g,  
      KS_Gompertz = ks_g,  
      KS_Exponencial = ks_e  
    )  
  )
```

```
)  
}
```

```
print(ks_resultados)
```

ITEM (f)

```
# Função para calcular a mediana  
mediana_gomp <- function(b, c) {  
  (1 / c) * log(1 - (c / b) * log(0.5))  
}
```

```
# Mediana para node4 = 0  
mediana0 <- mediana_gomp(b0_g, c_g)
```

```
# Mediana para node4 = 1  
mediana1 <- mediana_gomp(b1_g, c_g)
```

```
# Tabela de resultados  
medianas <- data.frame(  
  Grupo = c("node4 = 0", "node4 = 1"),  
  Mediana = c(mediana0, mediana1)  
)
```

```
print(medianas)
```

```
# Diferença entre as medianas  
diferenca <- mediana1 - mediana0
```

```
cat("Diferença entre as medianas:",  
    round(diferenca, 2), "\n")
```

```
# Comparação gráfica  
barplot(  
  c(mediana0, mediana1),  
  names.arg = c("node4 = 0", "node4 = 1"),  
  col = c("lightblue", "lightgreen"),  
  main = "Medianas estimadas - Gompertz",  
  xlab = "Grupo",  
  ylab = "Tempo mediano"  
)
```

OBS: O modelo Gompertz possui três parâmetros estimados, enquanto o Exponencial possui dois. Essa diferença deve ser considerada na comparação por AIC. Lembrem do princípio da parcimônia.