

# Trabalho6

June 12, 2025

```
[1]: library(pacman)
     p_load(dplyr,glarma,boot)
```

```
[2]: setwd('C:\\Users\\Marcelo\\OneDrive\\Área de Trabalho\\ts\\glarma')
     getwd()
```

'C:/Users/Marcelo/OneDrive/Área de Trabalho/ts/glarma'

## 1 Simulação

Criando a função que simula o processo:

```
[3]: simulate_glarma <- function(T = 100, beta = 1, theta = 0.3, x = NULL) {
     if (is.null(x)) x <- rep(1, T)

     z <- numeric(T)
     mu <- numeric(T)
     y <- numeric(T)
     e <- numeric(T)

     for (t in 1:T) {
       z_t <- 0
       if (t > 1) z_t <- + e[t - 1] * theta

       eta <- beta * x[t] + z_t
       mu[t] <- exp(eta)
       y[t] <- rpois(1, lambda = mu[t])
       e[t] <- (y[t] - mu[t]) / sqrt(mu[t])
       z[t] <- z_t
     }

     return(data.frame(time = 1:T, x = x, y = y, mu = mu, z = z,e=e))
  }
```

Vou conferir se a simulação está correta:

```
[4]: Amostra_original<-simulate_glarma(T=100,beta=1,theta=0.3,x=NULL)
```

```
[5]: length(Amostra_original$y)
```

100

```
[6]: summary(Amostra_original$y)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.00	2.00	3.00	3.37	4.25	11.00

```
[7]: X <- Amostra_original$x  
X <- as.matrix(X)  
colnames(X) <- "Intercept"
```

```
[8]: result=glarma(Amostra_original$y, X, phiLags = NULL ,thetaLags = c(1), type =  
  ↪"Poi")
```

```
[9]: Theta<-result$delta[2]  
Theta
```

**theta\\_1:** 0.338920651879532

```
[10]: Beta<-result$delta[1]  
Beta
```

**Intercept:** 1.13435742153011

Conseguimos recuperar os parâmetros bem, está tudo certo.

## 2 Bootstrap paramétrico:

Antes de criar as funções para rodar o experimento Monte Carlo, vou ver se a minha ideia do bootstrap está funcionando.

```
[17]: T=length(Amostra_original$y)  
B=100
```

gerando as amostras bootstrap:

```
[18]: mu<-Amostra_original$mu
```

```
[19]: length(mu)
```

100

```
[20]: Y_Bootstrap <- matrix(NA, nrow = T, ncol = B)
```

```
[22]: for (b in 1:B){  
  r = simulate_glarma(T=T,beta=1,theta=Theta,x=NULL)  
  Y_Bootstrap[,b] <- r$y  
}
```

criando uma função que retorna os parâmetros estimados a partir de uma amostra:

```
[23]: recupera_par <- function(Y_col,X) {  
  fit <- tryCatch({  
    glarma(Y_col, X, phiLags = NULL, thetaLags = c(1), type = "Poi")  
  }, error = function(e) {  
    message("GLARMA failed to converge: ", e$message)  
    return(NULL)  
  })  
  
  if (!is.null(fit) && !is.null(fit$delta) && length(fit$delta) >= 2 && is.  
→numeric(fit$delta)) {  
    return(c(beta = fit$delta[1],theta = fit$delta[2])  
  )  
  } else {  
    return(c(beta = NA,theta = NA)  
  )  
  }  
}
```

estimando os parâmetros para B amostras bootstrap.

```
[24]: par_estimates <- t(apply(Y_Bootstrap, 2, function(y_col) recupera_par(y_col, X)))
```

```
[25]: beta=par_estimates[,1]  
theta=par_estimates[,2]
```

```
[26]: par_estimates=list('beta'=beta[!is.na(beta)], 'theta'=theta[!is.na(theta)])
```

A média (nas B amostras) dos parâmetros está perto do valor dos parâmetros para amostra original.

```
[27]: mean(par_estimates$beta,na.rm=TRUE)  
mean(par_estimates$theta,na.rm=TRUE)
```

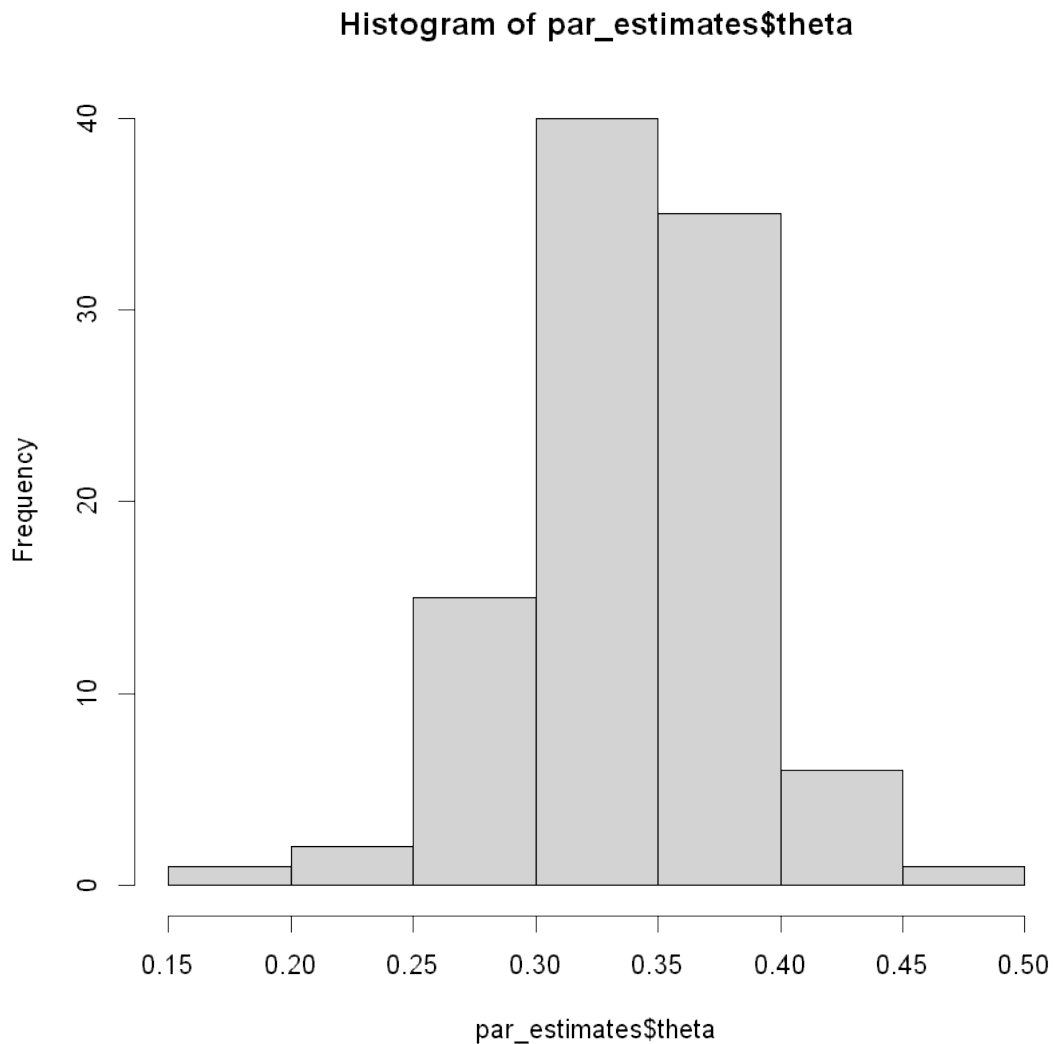
1.00196906653669

0.337134396509231

```
[28]: quantile(par_estimates$theta,probs=c(0.025,0.975))
```

2.5\%	0.240781370041432	97.5\%	0.416456947646565
-------	-------------------	--------	-------------------

```
[29]: hist(par_estimates$theta)
```



Nossa ideia para o bootstrap está funcionando, vamos fazer o experimento monte carlo:

### 2.0.1 Questão 1

Criando a função que faz o Bootstrap paramétrico. Ela retorna o valor dos parâmetros estimados na amostra original. Além disso, retorna os vetores Beta\_bootstrap e Theta\_bootstrap com as estimativas para os parâmetros na amostra bootstrap. Ela também retorna a média do Beta\_bootstrap e Theta\_bootstrap e seus intervalos de confiança. Finalmente, ela também retorna a Amostra original.

```
[31]: bootstrap_parametrico <- function(B=100,T=100,beta=1,theta=0.3)
{
```

```

Amostra_original<-simulate_glarma(T=100,beta=1,theta=0.3,x=NULL)
X <- Amostra_original$x
X <- as.matrix(X)
colnames(X) <- "Intercept"

result=glarma(Amostra_original$y, X, phiLags = NULL ,thetaLags = c(1), type_
→= "Poi")
beta_estimado<-result$delta[1]
theta_estimado<-result$delta[2]

T=length(Amostra_original$y)
mu<-result$mu

Y_Bootstrap <- matrix(NA, nrow = T, ncol = B)

for (b in 1:B)
{
  s = simulate_glarma(T=T,beta=beta_estimado,theta=theta_estimado,x=NULL)
  Y_Bootstrap[,b] <- s$y
}

par_bootstrap <- t(apply(Y_Bootstrap, 2, function(y_col) recupera_par(y_col,
→X)))

beta_bootstrap=par_bootstrap[,1]
theta_bootstrap=par_bootstrap[,2]

par_bootstrap=list('beta'=beta_bootstrap[!is.
→na(beta_bootstrap)], 'theta'=theta_bootstrap[!is.na(theta_bootstrap)])

mean_beta <- mean(par_bootstrap$beta)
mean_theta <- mean(par_bootstrap$theta)

IC_theta<-quantile(par_bootstrap$theta, probs = c(0.025, 0.975))
IC_beta<-quantile(par_bootstrap$beta, probs = c(0.025, 0.975))

return(list('beta_estimado'=beta_estimado,
  'Beta_bootstrap'=par_bootstrap[1],
  'theta_estimado'=theta_estimado,
  'Theta_bootstrap'=par_bootstrap[2],
  'mean_beta'=mean_beta,
  'mean_theta'=mean_theta,
  'IC_theta'=IC_theta,
  'IC_beta'=IC_beta,

```

```
}
      'Amostra_original' = Amostra_original$y))
```

Para ficar claro, vou rodar a função 1 vez com B=10 para ser possível ver o output:

```
[34]: bootstrap_parametrico(B=10)
```

```
$beta_estimado Intercept: 0.964179131401806
```

```
$Beta_bootstrap $beta = 1. 0.945538212925242 2. 1.21973334584034 3. 1.03639181295438
4. 0.947817952520812 5. 0.995444435156517 6. 1.12120861055112 7. 0.9130322361958
8. 1.01476887581031 9. 0.976672541823167 10. 1.10089475610305
```

```
$theta_estimado theta_1: 0.339725213744957
```

```
$Theta_bootstrap $theta = 1. 0.403055697943423 2. 0.379898228610576 3. 0.330324618826487
4. 0.349029714203789 5. 0.3057417983432 6. 0.365773922398388 7. 0.385603703485939
8. 0.384991197758038 9. 0.359958406777127 10. 0.244649198880401
```

```
$mean_beta 1.02715027798807
```

```
$mean_theta 0.350902648722737
```

```
$IC_theta 2.5\% 0.258395033759531 97.5\% 0.399128999190489
```

```
$IC_beta 2.5\% 0.920346080959924 97.5\% 1.19756528040027
```

```
$Amostra_original 1. 0 2. 0 3. 1 4. 1 5. 0 6. 3 7. 3 8. 4 9. 3 10. 3 11. 5 12. 4 13. 5 14. 2 15. 0 16. 1 17. 2
18. 0 19. 0 20. 2 21. 6 22. 5 23. 3 24. 2 25. 1 26. 1 27. 5 28. 7 29. 9 30. 6 31. 1 32. 5 33. 5 34. 3 35. 1
36. 1 37. 3 38. 3 39. 2 40. 1 41. 3 42. 3 43. 1 44. 3 45. 2 46. 4 47. 5 48. 5 49. 2 50. 1 51. 2 52. 3 53. 5
54. 1 55. 2 56. 3 57. 3 58. 3 59. 3 60. 3 61. 1 62. 2 63. 2 64. 7 65. 9 66. 2 67. 1 68. 1 69. 4 70. 3 71. 2
72. 3 73. 1 74. 2 75. 4 76. 5 77. 0 78. 4 79. 7 80. 2 81. 1 82. 1 83. 0 84. 1 85. 3 86. 4 87. 4 88. 7 89. 5
90. 1 91. 3 92. 5 93. 5 94. 1 95. 1 96. 2 97. 4 98. 3 99. 0 100. 2
```

A função Questao1 faz o experimento Monte Carlo. Ela retorna:

um vetor de tamanho R com os parâmetros estimados da Amostra original de cada iteração (r in 1:R).

um vetor de tamanho R (r 1:R) com a média (em b 1:B) dos parâmetros bootstrap em cada iteração r.

uma lista de tamanho R com de intervalo de confiança bootstrap para cada parâmetro.

Além disso retorna quatro escalares:

mean\_theta\_dif\_boot\_est <- média (em r 1:R) de (mean\_theta\_bootstrap[r] - theta\_estimado[r])^2 ( erro quadrático médio entre o theta estimado original e a média do theta\_bootstrap)

theta\_dif\_boot\_true; média (em r 1:R) de (mean\_theta\_bootstrap[r] - theta)^2 ( erro quadrático médio entre o theta verdadeiro e a média do theta\_bootstrap)

Raciocínio análogo para o beta.

```
[38]: Questao1 <- function(B = 20, T = 100, beta = 1, theta = 0.3, R = 2) {
```

```

theta_estimado <- numeric(R)
beta_estimado <- numeric(R)
mean_theta_bootstrap <- numeric(R)
mean_beta_bootstrap <- numeric(R)
IC_theta_bootstrap <- vector("list", R)
IC_beta_bootstrap <- vector("list", R)
theta_dif_boot_true <- numeric(R)
theta_dif_boot_est <- numeric(R)
beta_dif_boot_true <- numeric(R)
beta_dif_boot_est <- numeric(R)

for (r in 1:R) {
  result <- tryCatch({
    bootstrap_parametrico(B = B, T = T, beta = beta, theta = theta)
  }, error = function(e) {
    message(sprintf("Erro na repetição %d: %s", r, e$message))
    return(NULL)
  })

  if (!is.null(result)) {
    mean_theta_bootstrap[r] <- result$mean_theta
    mean_beta_bootstrap[r] <- result$mean_beta
    IC_theta_bootstrap[[r]] <- result$IC_theta
    IC_beta_bootstrap[[r]] <- result$IC_beta
    theta_estimado[r] <- result$theta_estimado
    beta_estimado[r] <- result$beta_estimado

    theta_dif_boot_est[r] <- (mean_theta_bootstrap[r] - theta_estimado[r])^2
    theta_dif_boot_true[r] <- (mean_theta_bootstrap[r] - theta)^2

    beta_dif_boot_est[r] <- (mean_beta_bootstrap[r] - beta_estimado[r])^2
    beta_dif_boot_true[r] <- (mean_beta_bootstrap[r] - beta)^2

  } else {
    mean_theta_bootstrap[r] <- NA
    mean_beta_bootstrap[r] <- NA

    IC_theta_bootstrap[[r]] <- c(NA, NA)
    IC_beta_bootstrap[[r]] <- c(NA, NA)
    theta_estimado[r] <- NA
    beta_estimado[r] <- NA

    theta_dif_boot_est[r] <- NA
    theta_dif_boot_true[r] <- NA

    beta_dif_boot_est[r] <- NA
  }
}

```

```

    beta_dif_boot_true[r] <- NA
  }

}

return(list(
  'theta_estimado' = theta_estimado,
  'beta_estimado' = beta_estimado,
  'mean_theta_bootstrap' = mean_theta_bootstrap,
  'mean_beta_bootstrap' = mean_beta_bootstrap,
  'IC_theta_bootstrap' = IC_theta_bootstrap,
  'IC_beta_bootstrap' = IC_beta_bootstrap,
  'mean_theta_dif_boot_est' = mean(theta_dif_boot_est, na.rm = TRUE),
  'mean_theta_dif_boot_true' = mean(theta_dif_boot_true, na.rm = TRUE),
  'mean_beta_dif_boot_est' = mean(beta_dif_boot_est, na.rm = TRUE),
  'mean_beta_dif_boot_true' = mean(beta_dif_boot_true, na.rm = TRUE)
))
}

```

Para ficar mais claro, vamos rodar a função com B=20 e R=2:

```
[39]: MM_bootstrap_parametrico<-Questao1(B=20,R=2)
```

GLARMA failed to converge: Fisher Scoring fails to converge from the initial estimates.

```
[40]: MM_bootstrap_parametrico
```

```

$theta_estimado 1. 0.373595101786257 2. 0.293284732046803
$beta_estimado 1. 1.01385123199714 2. 0.972661390434401
$mean_theta_bootstrap 1. 0.37490588714954 2. 0.285038538497943
$mean_beta_bootstrap 1. 0.37490588714954 2. 0.285038538497943
$IC_theta_bootstrap 1. 2.5\% 0.339499279926336 97.5\% 0.410908269162711
2. 2.5\% 0.205846749715939 97.5\% 0.367576003965858
$IC_beta_bootstrap 1. 2.5\% 0.846446291604389 97.5\% 1.19658680583726
2. 2.5\% 0.829021125027402 97.5\% 1.10518669946315
$mean_theta_dif_boot_est 3.48589331569297e-05
$mean_theta_dif_boot_true 0.00291736862996855
$mean_beta_dif_boot_est 0.440538170103822
$mean_beta_dif_boot_true 0.45095627067673

```



Respondendo a questão:

A função `conferindo_cobertura_bootstrap` responde a nossa questão 1) - ver texto abaixo do output.

```
[43]: conferindo_cobertura_bootstrap <- function(B = 20, T = 100, beta = 1, theta = 0.
→3, R = 100) {

  resultado <- Questao1(B = B, T = T, beta = beta, theta = theta, R = R)

  theta_estimado <- resultado$theta_estimado
  beta_estimado <- resultado$beta_estimado

  IC_theta_bootstrap <- resultado$IC_theta_bootstrap
  mean_theta_boot <- resultado$mean_theta_boot

  IC_beta_bootstrap <- resultado$IC_beta_bootstrap
  mean_beta_boot <- resultado$mean_beta_boot

  dentro_do_intervalo_theta <- logical(R)
  dentro_do_intervalo_beta <- logical(R)

  for (r in 1:R) {
    intervalo_theta <- IC_theta_bootstrap[[r]]
    intervalo_beta <- IC_beta_bootstrap[[r]]

    if (any(is.na(intervalo_theta))) {
      dentro_do_intervalo_theta[r] <- NA
      dentro_do_intervalo_beta[r] <- NA
    } else {
      dentro_do_intervalo_theta[r] <- (theta >= intervalo_theta[1]) && (theta <=
→intervalo_theta[2])
      dentro_do_intervalo_beta[r] <- (beta >= intervalo_beta[1]) && (beta <=
→intervalo_beta[2])
    }
  }

  cobertura_percentual_theta <- mean(dentro_do_intervalo_theta, na.rm = TRUE) *
→100
  cobertura_percentual_beta <- mean(dentro_do_intervalo_beta, na.rm = TRUE) * 100

  return(list(
    cobertura_percentual_theta = cobertura_percentual_theta,
    cobertura_percentual_beta = cobertura_percentual_beta,

    total_dentro_theta = sum(dentro_do_intervalo_theta, na.rm=TRUE),
```

```

total_dentro_beta = sum(dentro_do_intervalo_beta,na.rm=TRUE),

total_r = sum(!is.na(dentro_do_intervalo_theta)),

mean_dif_theta_boot_true = resultado$mean_theta_dif_boot_true,
mean_dif_theta_boot_est=resultado$mean_theta_dif_boot_est,
mean_dif_beta_boot_true = resultado$mean_beta_dif_boot_true ,
mean_dif_beta_boot_est=resultado$mean_beta_dif_boot_est
))
}

```

[44]: `conferindo_cobertura_bootstrap(R=100,B=100)`

GLARMA failed to converge: Fisher Scoring fails to converge from the initial estimates.

GLARMA failed to converge: Fisher Scoring fails to converge from the initial estimates.

GLARMA failed to converge: Fisher Scoring fails to converge from the initial estimates.

GLARMA failed to converge: Fisher Scoring fails to converge from the initial estimates.

GLARMA failed to converge: Fisher Scoring fails to converge from the initial estimates.

GLARMA failed to converge: Fisher Scoring fails to converge from the initial estimates.

GLARMA failed to converge: Fisher Scoring fails to converge from the initial estimates.

GLARMA failed to converge: Fisher Scoring fails to converge from the initial estimates.

`$cobertura_percentual_theta` 92

`$cobertura_percentual_beta` 92

`$total_dentro_theta` 92

`$total_dentro_beta` 92

`$total_r` 100

`$mean_dif_theta_boot_true` 0.00217628359507018

`$mean_dif_theta_boot_est` 2.41416906439163e-05

```
$mean_dif_beta_boot_true 0.494160383951601
```

```
$mean_dif_beta_boot_est 0.483344169615911
```

O parâmetro verdadeiro está contido em 92% dos 100 intervalos de confiança. Também podemos notar que o erro quadrático médio é mais baixo quando comparamos a média do bootstrap com o parâmetro estimado do que com o parâmetro verdadeiro. Logo a distância é menor em relação ao parâmetro estimado.

### 3 Bootstrap Não Paramétrico

Vamos ver o que acontece no bootstrap não paramétrico:

```
[45]: e<-Amostra_original$e
```

```
[46]: y<-Amostra_original$y
```

```
[47]: mu<-Amostra_original$mu
```

```
[48]: X <- Amostra_original$x  
X <- as.matrix(X)  
colnames(X) <- "Intercept"
```

```
[49]: e_boot <- replicate(B, sample(e, size = T, replace = TRUE))
```

```
[50]: gera_y_boot <- function(e,mu)  
{  
  
  T <- length(e)  
  y <- numeric(T)  
  
  for (t in 1:T)  
  {  
    y[t] = mu[t] + e[t]*sqrt(mu[t])  
    y[t] = round(y[t])  
  }  
  y[y < 1e-12] <- 0  
  return(data.frame(y=y))  
}
```

```
[51]: est_bootstrap <- function(y,X)  
{  
  fit <- tryCatch({  
    glarma(y, X, phiLags = NULL, thetaLags = c(1), type = "Poi")  
  }, error = function(e) {  
    message("GLARMA failed to converge: ", e$message)  
    return(NULL)  
  })  
}
```

```

if (!is.null(fit) && !is.null(fit$delta) && length(fit$delta) >= 2 && is.
→numeric(fit$delta)) {
  return(c(beta = fit$delta[1], theta = fit$delta[2])
)
} else {
  return(c(beta = NA, theta = NA)
)
}
}

```

```

[52]: gera_mu_boot <- function(e,y,x,par)
{
  z <- numeric(T)

  beta <- par[1]
  theta <- par[2]

  for (t in 1:T) {
    z_t <- 0
    if (t > 1) z_t <- e[t - 1] * theta

    eta <- beta * x[t] + z_t
    mu[t] <- exp(eta)
    z[t] <- z_t
  }
  return(mu)
}

```

```

[53]: bootstrap_n_parametrico <- function(e_boot, mu_init, y, X, B)
{
  T <- length(y)
  Y_Bootstrap <- matrix(NA, nrow = T, ncol = B)
  theta_boot <- numeric(B)
  beta_boot <- numeric(B)

  mu_current <- mu_init

  for (b in 1:B) {
    e <- e_boot[, b]

    y_boot <- gera_y_boot(e, mu_current)
    y_boot <- y_boot$y
    Y_Bootstrap[, b] <- y_boot

    par <- est_bootstrap(y_boot, X)
  }
}

```

```

    theta_boot[b] <- par[2]
    beta_boot[b] <- par[1]

    mu_current <- gera_mu_boot(e=e, y=y_boot, x=X, par=par)
  }

  return(cbind(beta_boot, theta_boot))
}

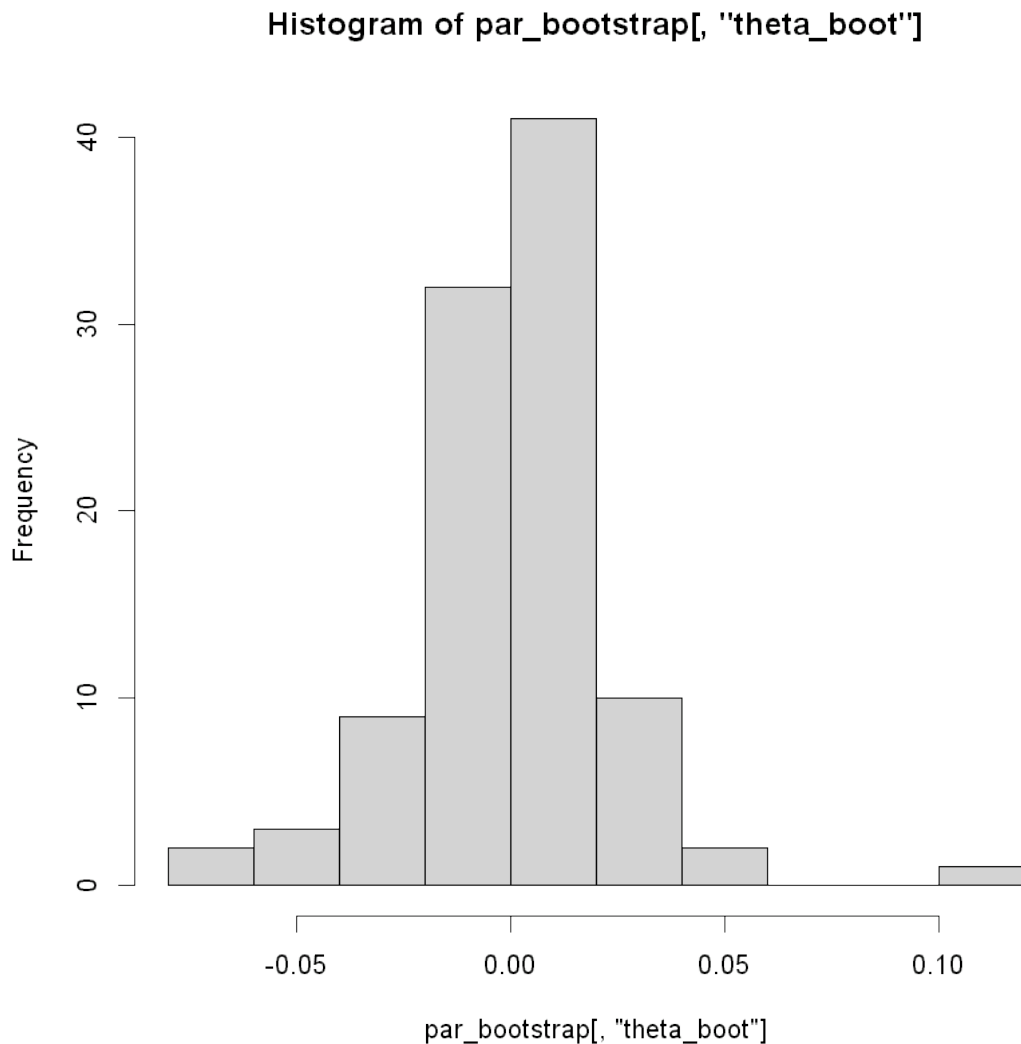
```

```
[54]: par_bootstrap <- bootstrap_n_parametrico(e_boot=e_boot, mu=mu, y=y, X=X, B=B)
```

```
[55]: mean(par_bootstrap[, 'theta_boot'])
```

```
-0.000370618399197457
```

```
[56]: hist(par_bootstrap[, 'theta_boot'])
```



O bootstrap não paramétrico deu errado, como o esperado. Eu montei o experimento Monte Carlo para o código ficar completo:

### 3.0.1 Questão 2

```
[59]: Questao2 <- function(B=20,T=100,beta=1,theta=0.3,R=2)
  {

    theta_estimado <- numeric(R)
    beta_estimado <- numeric(R)
    mean_theta_bootstrap <- numeric(R)
    mean_beta_bootstrap <- numeric(R)
    IC_theta_bootstrap <- vector("list", R)
    IC_beta_bootstrap <- vector("list", R)
    theta_dif_boot_true <- numeric(R)
    theta_dif_boot_est <- numeric(R)
    beta_dif_boot_true <- numeric(R)
    beta_dif_boot_est <- numeric(R)

    for (r in 1:R)
    {

      Amostra_original<-simulate_glarma(T=100,beta=1,theta=0.3,x=NULL)
      X <- Amostra_original$x
      X <- as.matrix(X)
      colnames(X) <- "Intercept"

      result=glarma(Amostra_original$y, X, phiLags = NULL ,thetaLags = c(1),
      ↪type = "Poi")
      beta_estimado[r]<-result$delta[1]
      theta_estimado[r]<-result$delta[2]

      e<-Amostra_original$e
      e_boot <- replicate(B, sample(e, size = T, replace = TRUE))

      ↪
      ↪par_bootstrap<-bootstrap_n_parametrico(e_boot=e_boot,B=20,mu_init=Amostra_original$mu,y=Amostra_original$y)

      Theta_bootstrap<-par_bootstrap[, 'theta_boot']
      Beta_bootstrap<-par_bootstrap[, 'beta_boot']

      mean_theta_bootstrap[r] <- mean(Theta_bootstrap)
      mean_beta_bootstrap[r] <- mean(Beta_bootstrap)
```

```

        IC_theta_bootstrap[[r]]<-quantile(Theta_bootstrap, probs = c(0.025, 0.
→975))
        IC_beta_bootstrap[[r]]<-quantile(Beta_bootstrap, probs = c(0.025, 0.975))

        theta_dif_boot_est[r] <- (mean_theta_bootstrap[r] - theta_estimado[r])^2
        theta_dif_boot_true[r] <- (mean_theta_bootstrap[r] - theta)^2

        beta_dif_boot_est[r] <- (mean_beta_bootstrap[r] - beta_estimado[r])^2
        beta_dif_boot_true[r] <- (mean_beta_bootstrap[r] - beta)^2

    }
    return(list('theta_estimado'=theta_estimado,
               'beta_estimado'=beta_estimado,
               'mean_theta_bootstrap'=mean_theta_bootstrap,
               'mean_beta_bootstrap'=mean_beta_bootstrap,
               'IC_theta_bootstrap'=IC_theta_bootstrap,
               'IC_beta_bootstrap'=IC_beta_bootstrap,
               'mean_theta_dif_boot_est' = mean(theta_dif_boot_est, na.rm =_
→TRUE),
               'mean_theta_dif_boot_true' = mean(theta_dif_boot_true, na.rm =_
→TRUE),
               'mean_beta_dif_boot_est' = mean(beta_dif_boot_est, na.rm = TRUE),
               'mean_beta_dif_boot_true' = mean(beta_dif_boot_true, na.rm =_
→TRUE)
               ))
  }

```

Testando as funções:

```
[60]: MM_bootstrap_n_parametrico<-Questao2(B=20,R=2)
```

```
[61]: MM_bootstrap_n_parametrico
```

```

$theta_estimado 1. 0.282816431072546 2. 0.325890607844324
$beta_estimado 1. 1.04409512264816 2. 0.949878856913491
$mean_theta_bootstrap 1. -0.0133797731679488 2. -0.00509248055837412
$mean_beta_bootstrap 1. 1.18326431256418 2. 0.94886120057529
$IC_theta_bootstrap 1. 2.5\% -0.158034161411147 97.5\% 0.0688101305732844
                2. 2.5\% -0.128559988372725 97.5\% 0.101332194159946
$IC_beta_bootstrap 1. 2.5\% 0.944477458455162 97.5\% 1.46121032147708
                2. 2.5\% 0.832989816411914 97.5\% 1.09740571529093
$mean_theta_dif_boot_est 0.0986409981075326
$mean_theta_dif_boot_true 0.0956441519620284

```

`$mean_beta_dif_boot_est 0.00968454952315114`

`$mean_beta_dif_boot_true 0.0181004925331102`

```
[62]: conferindo_cobertura_bootstrap2 <- function(B = 20, T = 100, beta = 1, theta = 0.
→3, R = 100) {

  resultado <- Questao2(B = B, T = T, beta = beta, theta = theta, R = R)

  theta_estimado <- resultado$theta_estimado
  beta_estimado <- resultado$beta_estimado

  IC_theta_bootstrap <- resultado$IC_theta_bootstrap
  mean_theta_boot <- resultado$mean_theta_boot

  IC_beta_bootstrap <- resultado$IC_beta_bootstrap
  mean_beta_boot <- resultado$mean_beta_boot

  dentro_do_intervalo_theta <- logical(R)
  dentro_do_intervalo_beta <- logical(R)

  for (r in 1:R) {
    intervalo_theta <- IC_theta_bootstrap[[r]]
    intervalo_beta <- IC_beta_bootstrap[[r]]

    if (any(is.na(intervalo_theta))) {
      dentro_do_intervalo_theta[r] <- NA
      dentro_do_intervalo_beta[r] <- NA
    } else {
      dentro_do_intervalo_theta[r] <- (theta >= intervalo_theta[1]) && (theta <=
→intervalo_theta[2])
      dentro_do_intervalo_beta[r] <- (beta >= intervalo_beta[1]) && (beta <=
→intervalo_beta[2])
    }
  }

  cobertura_percentual_theta <- mean(dentro_do_intervalo_theta, na.rm = TRUE) *
→100
  cobertura_percentual_beta <- mean(dentro_do_intervalo_beta, na.rm = TRUE) * 100

  return(list(
    cobertura_percentual_theta = cobertura_percentual_theta,
    cobertura_percentual_beta = cobertura_percentual_beta,

    total_dentro_theta = sum(dentro_do_intervalo_theta, na.rm=TRUE),
    total_dentro_beta = sum(dentro_do_intervalo_beta, na.rm=TRUE),
```



```

total_r = sum(!is.na(dentro_do_intervalo_theta)),

mean_dif_theta_boot_true = resultado$mean_theta_dif_boot_true,
mean_dif_theta_boot_est=resultado$mean_theta_dif_boot_est,
mean_dif_beta_boot_true = resultado$mean_beta_dif_boot_true ,
mean_dif_beta_boot_est=resultado$mean_beta_dif_boot_est
))
}

```

```
[63]: conferindo_cobertura_bootstrap2(R=2)
```

```

$cobertura_percentual_theta 0
$cobertura_percentual_beta 0
$total_dentro_theta 0
$total_dentro_beta 0
$total_r 2
$mean_dif_theta_boot_true 0.0849217372298995
$mean_dif_theta_boot_est 0.0927827690486409
$mean_dif_beta_boot_true 0.596613767025563
$mean_dif_beta_boot_est 0.45780750192877

```

Já sabemos que o não paramétrico está errado, logo fiz poucas iterações.

## 4 Questão 3

criando a função que faz previsões no modelo glarma:

```
[64]: prev_glarma <- function(y, x, beta, theta, seed = NULL) {

  futuro_x <- rep(mean(x[, 1], na.rm = TRUE), 1)

  if (!is.null(seed)) set.seed(seed)

  T <- length(y)
  mu <- numeric(T)
  e <- numeric(T)
  z <- numeric(T)

  #passo 1, reconstruir z,e,mu

  for (t in 1:T) {

    z_t <- if (t > 1) e[t - 1] * theta else 0

```

```

    eta <- beta * x[t] + z_t
    mu[t] <- exp(eta)
    e[t] <- (y[t] - mu[t]) / sqrt(mu[t])
    z[t] <- z_t
  }

  e_t <- e[T]

  z_t <- e_t * theta

  eta <- beta * futuro_x + z_t

  y_pred <- exp(eta)

  return(list('y_pred' = y_pred))
}

```

criando uma função que faz o Intervalo preditivo bootstrap utilizando o bootstrap paramétrico já apresentado.

```

[69]: intervalo_preditivo_bootstrap <- function(T=100,B=100, beta = 1, theta = 0.
  ↪3,R=2){

  mean_Y_pred <- numeric(R)
  IC_pred <- vector("list", R)
  Y_teste <- numeric(R)

  x_train <- rep(1,(T-1))
  x_train <- as.matrix(x_train)
  colnames(x_train) <- "Intercept"

  for (r in 1:R)
  {

    Y_pred<-numeric(B)
    beta_boot<-numeric(B)
    theta_boot<-numeric(B)

    boot <- bootstrap_parametrico(T=100,B=20, beta = 1, theta = 0.3)

    Y <- boot$Amostra_original
    Y_train = Y[1:(T-1)]
    Y_test = Y[T]

    theta_boot<-boot$Theta_bootstrap$theta
  }
}

```

```

beta_boot<-boot$Beta_bootstrap$beta

for (b in 1:B)
{
  ↵
  ↪prev<-prev_glarma(y=Y_train,x=x_train,beta=beta_boot[b],theta=theta_boot[b])
  Y_pred[b]<-prev$y_pred
}

mean_Y_pred[r]=mean(Y_pred,na.rm=TRUE)
IC_pred[[r]]<-quantile(mean_Y_pred,probs = c(0.025, 0.975))
Y_teste[r]<-Y_test
}

return(list('Y_test'=Y_teste,'IC_pred'=IC_pred))
}

```

rodando a função com R=10 para verificar o output:

```
[71]: intervalo_preditivo_bootstrap(R=10,B=100)
```

```

$Y_test 1. 2 2. 2 3. 2 4. 3 5. 7 6. 3 7. 0 8. 1 9. 1 10. 4

$IC_pred  1. 2.5\%           0 97.5\%           1.6324996379393

      2. 2.5\%           0 97.5\%           2.19759296685041
      3. 2.5\%           0 97.5\%           2.21597746189282
      4. 2.5\%           0 97.5\%           2.53347829393443
      5. 2.5\%           0 97.5\%           3.45836002412046
      6. 2.5\%           0 97.5\%           3.73917218778078
      7. 2.5\%           0 97.5\%           3.73917218778078
      8. 2.5\%           0 97.5\%           3.73917218778078
      9. 2.5\%           0.473951507788828 97.5\%           3.73917218778078
     10. 2.5\%           2.11367830253043 97.5\%           3.73917218778078

```

conferindo\_IP verifica o percentual de intervalos que contém o Y\_teste:

```

[72]: conferindo_IP <- function(B = 20, T = 100, beta = 1, theta = 0.3, R=2) {

  resultado <- intervalo_preditivo_bootstrap(T=T,B=B, beta = beta, theta = ↵
  ↪theta,R=R)

  Y_test <- resultado$Y_test
  IC_pred <- resultado$IC_pred

```

```
dentro_do_intervalo <- logical(R)

for (r in 1:R) {
  intervalo <- IC_pred[[r]]
  if (any(is.na(intervalo))) {
    dentro_do_intervalo[r] <- NA
  } else {
    dentro_do_intervalo[r] <- (Y_test[r] >= intervalo[1]) && (Y_test[r] <=
→intervalo[2])
  }
}

cobertura_percentual <- mean(dentro_do_intervalo, na.rm = TRUE) * 100

return(list(
  Intervalos=resultado,
  total_dentro = sum(dentro_do_intervalo, na.rm = TRUE),
  cobertura_percentual = cobertura_percentual,
  total_R = sum(!is.na(dentro_do_intervalo))
))
}
```

[73]: `conferindo_IP(R=100,B=100)`

GLARMA failed to converge: Fisher Scoring fails to converge from the initial estimates.

GLARMA failed to converge: Fisher Scoring fails to converge from the initial estimates.

```
$Intervalos $Y_test 1. 0 2. 6 3. 6 4. 6 5. 4 6. 5 7. 2 8. 6 9. 0 10. 2 11. 1 12. 5 13. 2 14. 3 15. 2 16. 1 17. 3
18. 0 19. 3 20. 4 21. 3 22. 4 23. 3 24. 4 25. 5 26. 1 27. 2 28. 1 29. 4 30. 1 31. 7 32. 2 33. 2 34. 0
35. 2 36. 1 37. 4 38. 3 39. 3 40. 3 41. 3 42. 2 43. 3 44. 2 45. 3 46. 3 47. 1 48. 1 49. 4 50. 2 51. 1
52. 2 53. 2 54. 4 55. 1 56. 2 57. 5 58. 5 59. 2 60. 1 61. 3 62. 4 63. 2 64. 4 65. 1 66. 3 67. 2 68. 3
69. 0 70. 0 71. 3 72. 4 73. 2 74. 3 75. 5 76. 7 77. 1 78. 3 79. 4 80. 2 81. 4 82. 0 83. 3 84. 2 85. 3
86. 1 87. 2 88. 4 89. 1 90. 2 91. 2 92. 3 93. 4 94. 2 95. 3 96. 2 97. 3 98. 4 99. 1 100. 3

$IC_pred 1. 2.5\% 0 97.5\% 0
2. 2.5\% 0 97.5\% 0
3. 2.5\% 0 97.5\% 1.17995248110578
4. 2.5\% 0 97.5\% 2.83391731500919
5. 2.5\% 0 97.5\% 3.32480844463045
6. 2.5\% 0 97.5\% 3.32480844463045
7. 2.5\% 0 97.5\% 3.32480844463045
```

8. 2.5\%	0 97.5\%	3.32480844463045
9. 2.5\%	0 97.5\%	3.32480844463045
10. 2.5\%	0 97.5\%	3.32480844463045
11. 2.5\%	0 97.5\%	3.32480844463045
12. 2.5\%	0 97.5\%	3.57523401356289
13. 2.5\%	0 97.5\%	3.57523401356289
14. 2.5\%	0 97.5\%	3.57523401356289
15. 2.5\%	0 97.5\%	3.57523401356289
16. 2.5\%	0 97.5\%	3.57523401356289
17. 2.5\%	0 97.5\%	3.57523401356289
18. 2.5\%	0 97.5\%	3.57523401356289
19. 2.5\%	0 97.5\%	3.97385251849517
20. 2.5\%	0 97.5\%	4.16139073582548
21. 2.5\%	0 97.5\%	4.16139073582548
22. 2.5\%	0 97.5\%	4.20333539811624
23. 2.5\%	0 97.5\%	4.20333539811624
24. 2.5\%	0 97.5\%	4.20333539811624
25. 2.5\%	0 97.5\%	4.20333539811624
26. 2.5\%	0 97.5\%	4.20333539811624
27. 2.5\%	0 97.5\%	4.20333539811624
28. 2.5\%	0 97.5\%	4.20333539811624
29. 2.5\%	0 97.5\%	4.20333539811624
30. 2.5\%	0 97.5\%	4.20333539811624
31. 2.5\%	0 97.5\%	4.20333539811624
32. 2.5\%	0 97.5\%	4.20333539811624
33. 2.5\%	0 97.5\%	4.20333539811624
34. 2.5\%	0 97.5\%	4.20333539811624
35. 2.5\%	0 97.5\%	4.20333539811624
36. 2.5\%	0 97.5\%	4.20333539811624
37. 2.5\%	0 97.5\%	4.27833997345539
38. 2.5\%	0 97.5\%	4.27833997345539
39. 2.5\%	0 97.5\%	4.27833997345539

40. 2.5\%	0 97.5\%	4.27833997345539
41. 2.5\%	0 97.5\%	4.27833997345539
42. 2.5\%	0 97.5\%	4.27833997345539
43. 2.5\%	0 97.5\%	4.27833997345539
44. 2.5\%	0 97.5\%	4.27833997345539
45. 2.5\%	0 97.5\%	4.27833997345539
46. 2.5\%	0 97.5\%	4.27833997345539
47. 2.5\%	0 97.5\%	4.27833997345539
48. 2.5\%	0 97.5\%	4.27833997345539
49. 2.5\%	0 97.5\%	4.27833997345539
50. 2.5\%	0 97.5\%	4.27833997345539
51. 2.5\%	0 97.5\%	4.27833997345539
52. 2.5\%	0 97.5\%	4.27833997345539
53. 2.5\%	0 97.5\%	4.27833997345539
54. 2.5\%	0 97.5\%	4.27833997345539
55. 2.5\%	0 97.5\%	4.27833997345539
56. 2.5\%	0 97.5\%	4.27833997345539
57. 2.5\%	0 97.5\%	4.27833997345539
58. 2.5\%	0 97.5\%	4.27833997345539
59. 2.5\%	0 97.5\%	4.27833997345539
60. 2.5\%	0 97.5\%	4.27833997345539
61. 2.5\%	0 97.5\%	4.27833997345539
62. 2.5\%	0 97.5\%	4.27833997345539
63. 2.5\%	0 97.5\%	4.27833997345539
64. 2.5\%	0 97.5\%	4.27833997345539
65. 2.5\%	0 97.5\%	4.27833997345539
66. 2.5\%	0 97.5\%	4.34559241471176
67. 2.5\%	0 97.5\%	4.34559241471176
68. 2.5\%	0 97.5\%	4.34559241471176
69. 2.5\%	0 97.5\%	4.34559241471176
70. 2.5\%	0 97.5\%	4.40105303455639
71. 2.5\%	0 97.5\%	4.40105303455639

72. 2.5\%	0 97.5\%	4.40105303455639
73. 2.5\%	0 97.5\%	4.40105303455639
74. 2.5\%	0 97.5\%	4.40105303455639
75. 2.5\%	0 97.5\%	4.40105303455639
76. 2.5\%	0 97.5\%	4.51340245746593
77. 2.5\%	0 97.5\%	4.51340245746593
78. 2.5\%	0 97.5\%	4.51340245746593
79. 2.5\%	0 97.5\%	4.51340245746593
80. 2.5\%	0 97.5\%	4.51340245746593
81. 2.5\%	0 97.5\%	4.51340245746593
82. 2.5\%	0 97.5\%	4.51340245746593
83. 2.5\%	0 97.5\%	4.51340245746593
84. 2.5\%	0 97.5\%	4.51340245746593
85. 2.5\%	0 97.5\%	4.51340245746593
86. 2.5\%	0 97.5\%	4.51340245746593
87. 2.5\%	0 97.5\%	4.51340245746593
88. 2.5\%	0 97.5\%	4.51340245746593
89. 2.5\%	0 97.5\%	4.51340245746593
90. 2.5\%	0 97.5\%	4.51340245746593
91. 2.5\%	0 97.5\%	4.51340245746593
92. 2.5\%	0 97.5\%	4.51340245746593
93. 2.5\%	0 97.5\%	4.52549761765428
94. 2.5\%	0 97.5\%	4.52549761765428
95. 2.5\%	0 97.5\%	4.52549761765428
96. 2.5\%	0 97.5\%	4.52549761765428
97. 2.5\%	0.657865607622061 97.5\%	4.52549761765428
98. 2.5\%	1.42365517438767 97.5\%	4.52549761765428
99. 2.5\%	1.50103048029173 97.5\%	4.52549761765428
100. 2.5\%	1.63547183253908 97.5\%	4.52549761765428

**\$total\_dentro 86**

**\$cobertura\_percentual 86**

**\$total\_R 100**

Das 100 previsões, apenas 86 estão no intervalo preditivo bootstrap. Pode ser falta de sorte ou há algo errado na minha previsão.