Estimativas - Bayesianas

André

12/07/2022

Relatório de Estimativas Bayesianas (conjunto TGCA)

```
log_veros = function(par, tempos, cens){
  return(sum(cens*dgompertz(tempos, par[1], par[2], ln = T) +
               (1 - cens)*pgompertz(tempos, par[1], par[2], ln = T, lower.tail = F)))
}

log_veros_mix = function(par, tempos, cens){
  return(sum(cens*log(1-par[3]) + cens*dgompertz(tempos, par[1], par[2], ln = T) +
               (1- cens)*log(par[3]  + (1-par[3])*(1 - pgompertz(tempos, a = par[1], b = par[2])))))
}

log_veros_def = function(par, tempos, cens){
  return(sum(cens*flexsurv::dgompertz(tempos, par[1], par[2], log = T) +
               (1 - cens)*flexsurv::pgompertz(tempos, par[1], par[2], log = T, lower.tail = F)))
}

calcula_dic = function(tempos, cens, log_veros, cadeia_a, cadeia_b, cadeia_p = NULL){
  a = mean(cadeia_a)
  b = mean(cadeia_b)

  if(!is.null(cadeia_p)){
    p = mean(cadeia_p)
    par = c(a,b,p)
    aux = numeric(length = length(cadeia_a))

    for(i in 1:length(cadeia_a)){
      aux[i] = log_veros(c(cadeia_a[i],cadeia_b[i], cadeia_p[i]), tempos, cens)
    }

    p_dic = 2*(log_veros(par, tempos, cens) - mean(aux))

    dic = -2*log_veros(par, tempos, cens) + 2*p_dic
  }
  else{
    par = c(a,b)
    aux = numeric(length = length(cadeia_a))

    for(i in 1:length(cadeia_a)){
      aux[i] = log_veros(c(cadeia_a[i],cadeia_b[i]), tempos, cens)
    }

    p_dic = 2*(log_veros(par, tempos, cens) - mean(aux))

    dic = -2*log_veros(par, tempos, cens) + 2*p_dic
  }
  return(dic)
}
```

TGCA

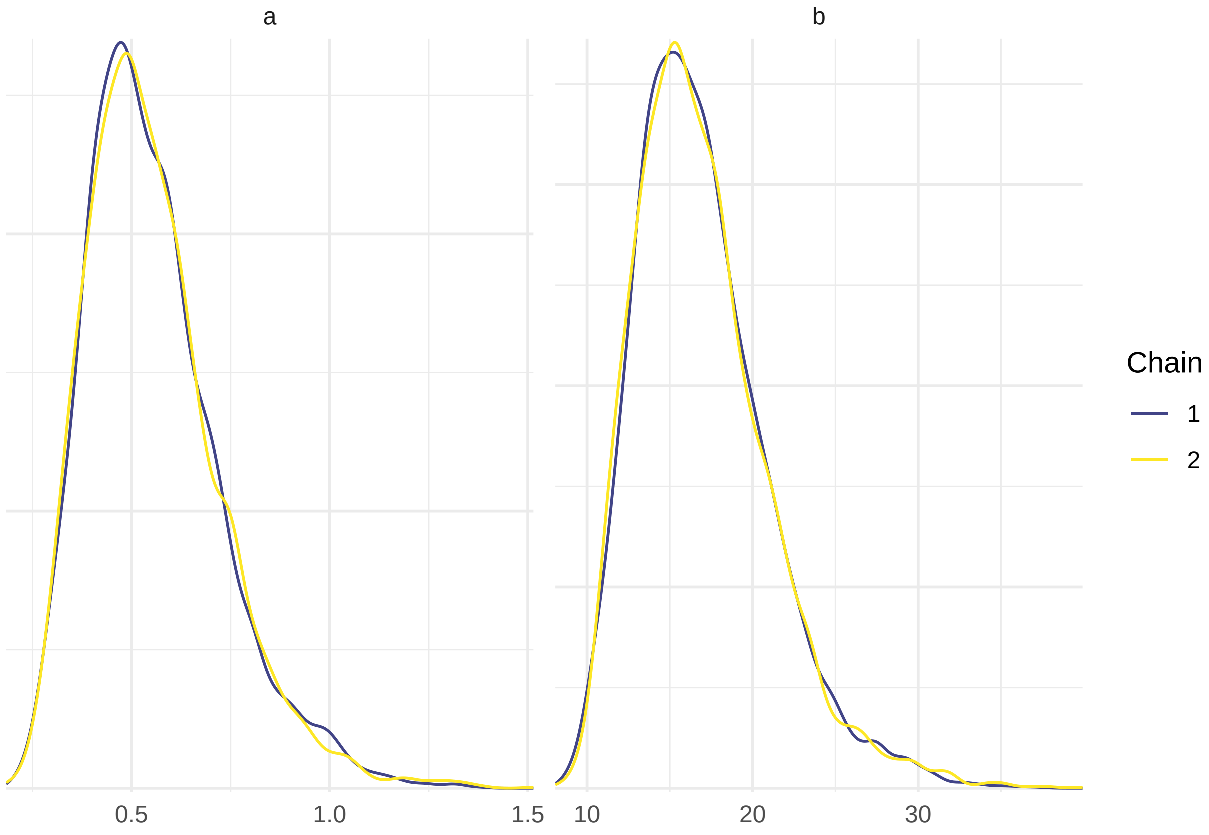
```
dados_tg = read.csv('dados_tg.csv')

#### Modelo usual

fit_usual_1_cadeia = stan(file = 'codigos_stan/usu_tgca.stan',
            data = list(N = nrow(dados_tg), T = dados_tg$tempo/365, D = dados_tg$cens),
            iter = 11000, warmup = 1000, chains = 1, cores = 1, seed = 154)
```

```
##
## SAMPLING FOR MODEL 'usu_tgca' NOW (CHAIN 1).
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1:
## Chain 1: Gradient evaluation took 0.002 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 20 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
```

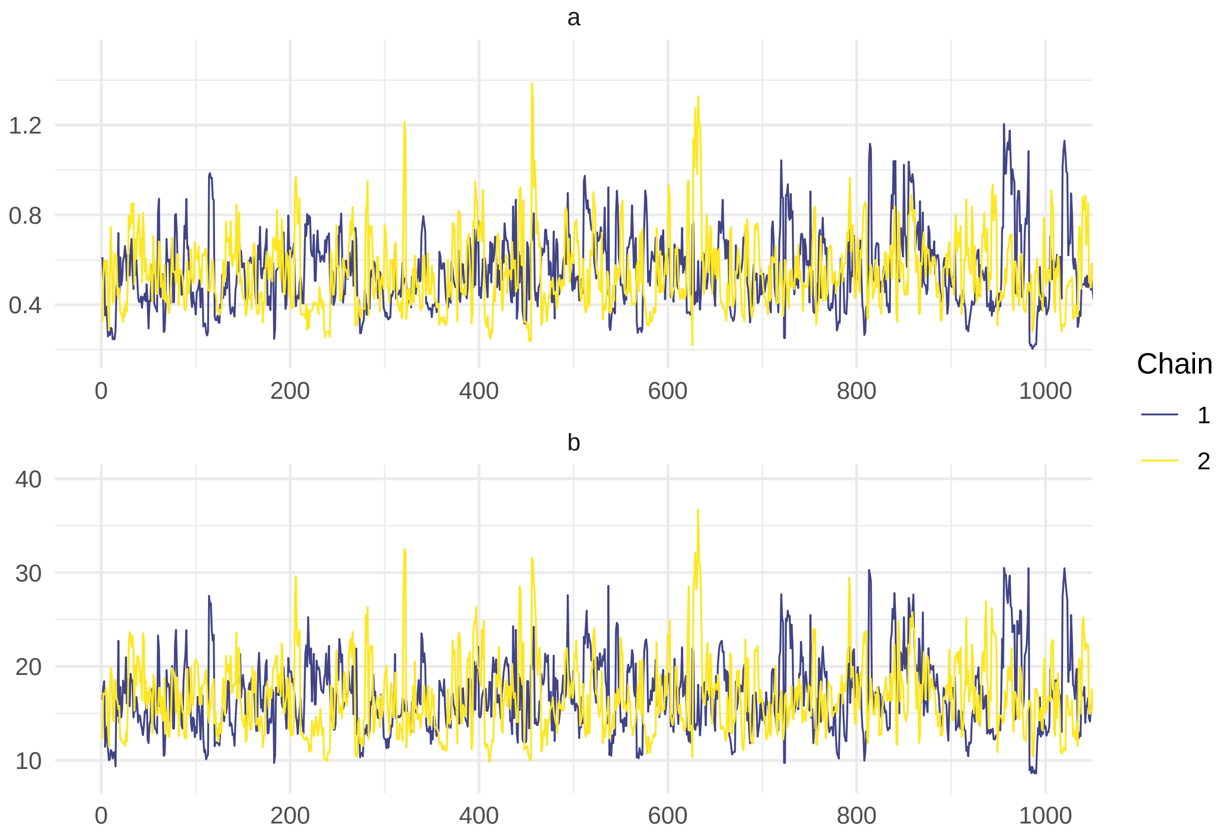
```
## Chain 1: Iteration:     1 / 11000 [  0%]  (Warmup)
## Chain 1: Iteration:  1001 / 11000 [  9%]  (Sampling)
## Chain 1: Iteration:  2100 / 11000 [ 19%]  (Sampling)
## Chain 1: Iteration:  3200 / 11000 [ 29%]  (Sampling)
## Chain 1: Iteration:  4300 / 11000 [ 39%]  (Sampling)
## Chain 1: Iteration:  5400 / 11000 [ 49%]  (Sampling)
## Chain 1: Iteration:  6500 / 11000 [ 59%]  (Sampling)
## Chain 1: Iteration:  7600 / 11000 [ 69%]  (Sampling)
## Chain 1: Iteration:  8700 / 11000 [ 79%]  (Sampling)
## Chain 1: Iteration:  9800 / 11000 [ 89%]  (Sampling)
## Chain 1: Iteration: 10900 / 11000 [ 99%]  (Sampling)
## Chain 1: Iteration: 11000 / 11000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 9.123 seconds (Warm-up)
## Chain 1:                92.892 seconds (Sampling)
## Chain 1:                102.015 seconds (Total)
## Chain 1:
```

```
fit_usual = stan(file = 'codigos_stan/usu_tgca.stan',
            data = list(N = nrow(dados_tg), T = dados_tg$tempo/365, D = dados_tg$cens),
            iter = 6000, warmup = 1000, chains = 2, cores = 2, seed = 154)
```

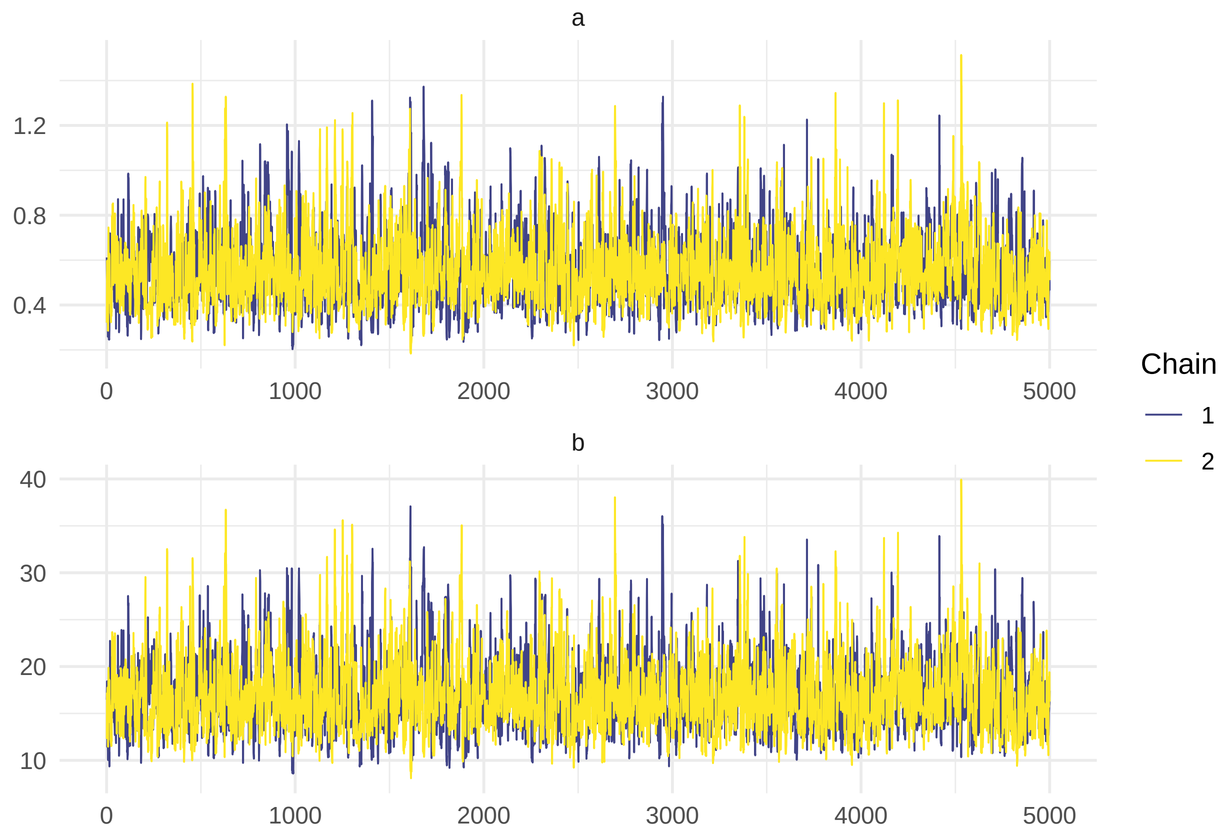
```
color_scheme_set("viridis")
mcmc_dens_overlay(fit_usual, pars = c("a", "b"))
```



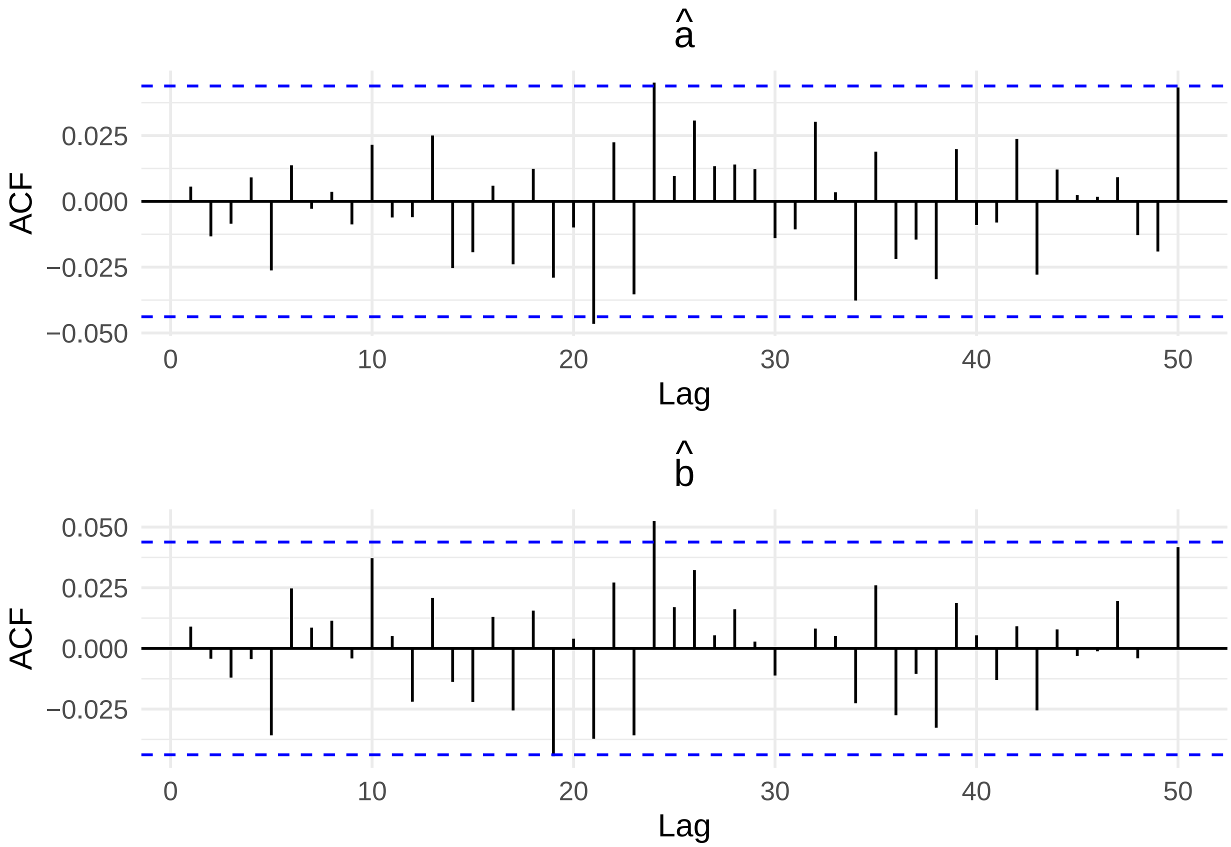
```
color_scheme_set("viridis")
mcmc_trace(fit_usual, window = c(1,1000), pars = c("a", "b"),
           facet_args = list(nrow = 2))
```



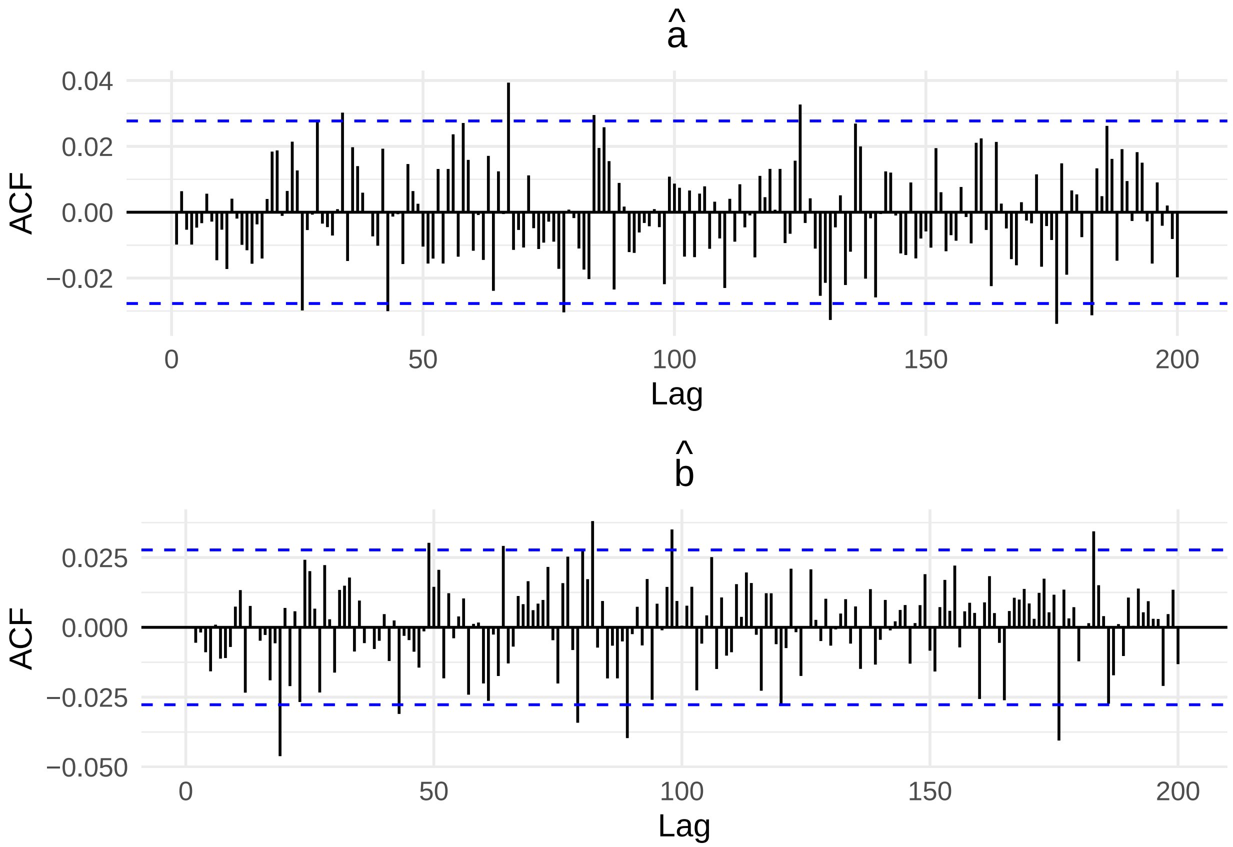
```
color_scheme_set("viridis")
mcmc_trace(fit_usual, pars = c("a", "b"),
           facet_args = list(nrow = 2))
```



```
cadeias_df = data.frame(index = 1:length(extract(fit_usual, "a")), a = extract(fit_usual, "a"), b = ext

n = 2000

acf_a = ggAcf(cadeias_df[1:n,]$a, lag.max = 50) +
  theme_minimal() +
  labs(title = expression(hat(a))) +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size=12))

acf_b = ggAcf(cadeias_df[1:n,]$b, lag.max = 50) +
  theme_minimal() +
  labs(title = expression(hat(b))) +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size=12))

gridExtra::grid.arrange(acf_a, acf_b)
```

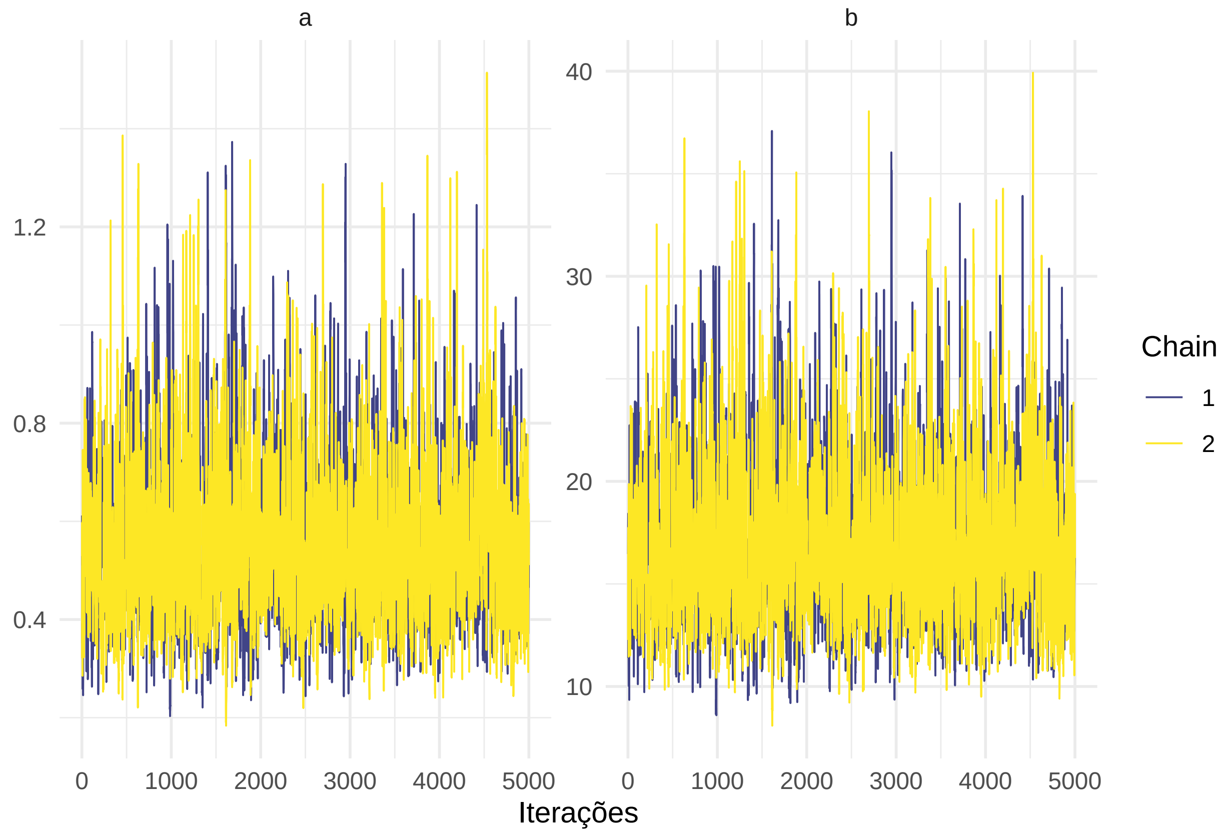


```
cadeias_df = data.frame(index = 1:length(extract(fit_usual_1_cadeia, "a")), a = extract(fit_usual_1_cade

n = 5000

acf_a = ggAcf(cadeias_df[1:n,]$a, lag.max = 200) +
  theme_minimal() +
  labs(title = expression(hat(a))) +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size=12))

acf_b = ggAcf(cadeias_df[1:n,]$b, lag.max = 200) +
  theme_minimal() +
  labs(title = expression(hat(b))) +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size=12))

gridExtra::grid.arrange(acf_a, acf_b)
```

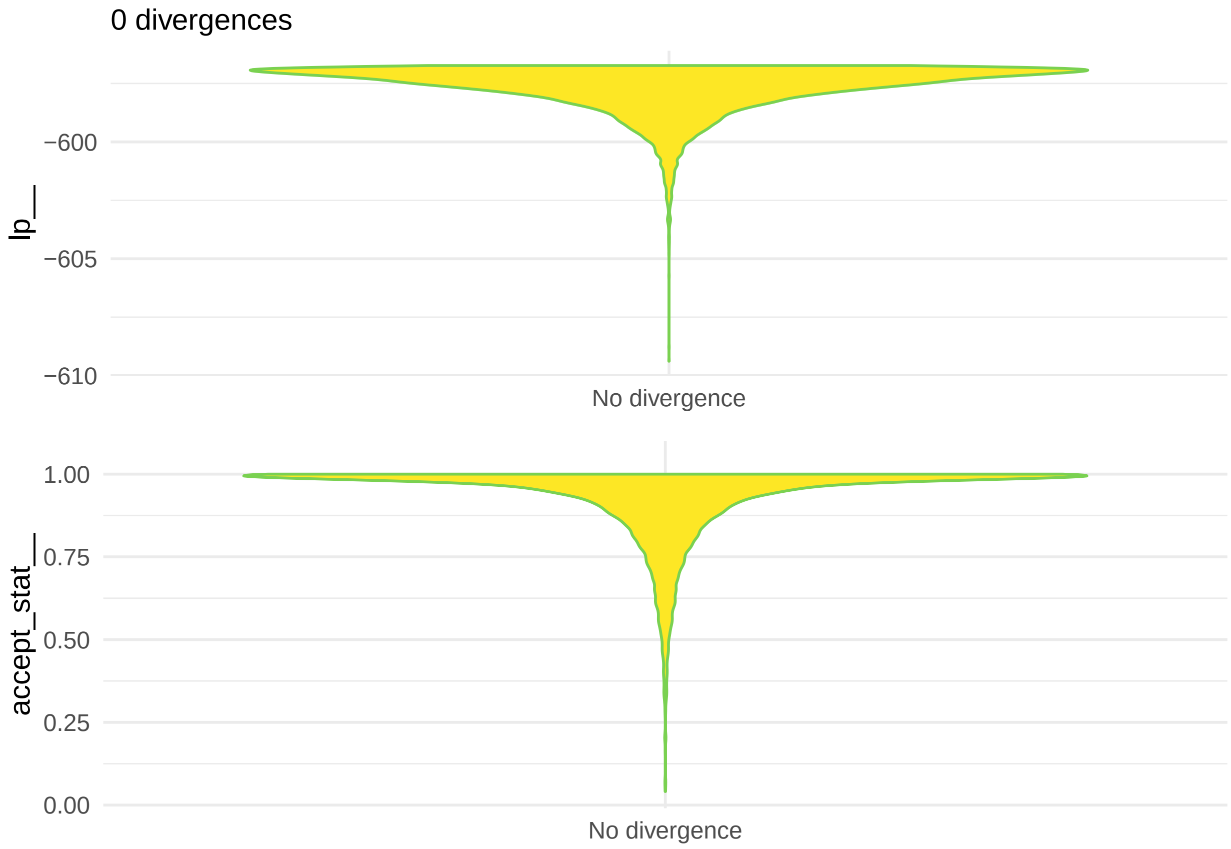


```
lp_cp = log_posterior(fit_usual)
np_cp = nuts_params(fit_usual)

#mcmc_parcoord(as.array(fit_usual), pars = c("a","b"), np = np_cp)

mcmc_trace(fit_usual, pars = c("a","b"), np = np_cp) +
  xlab("Iterações")
```

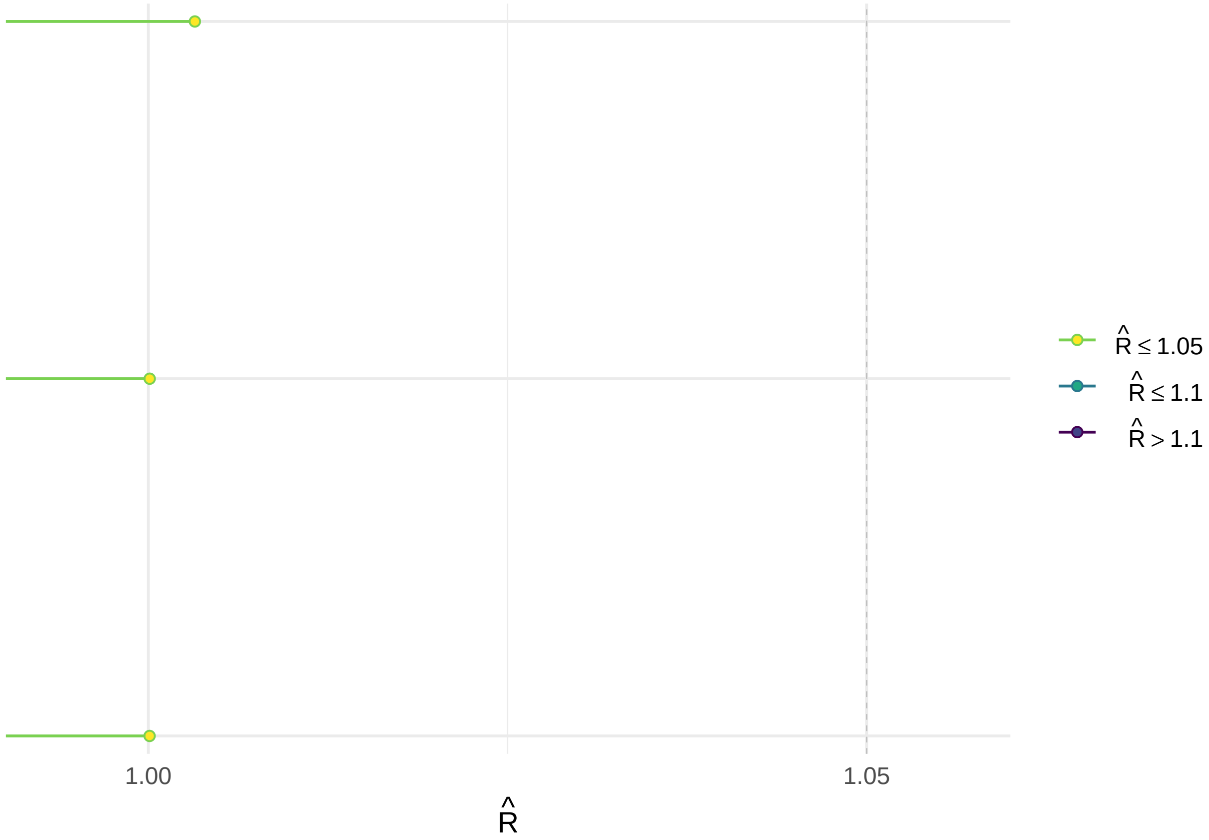
```
## No divergences to plot.
```



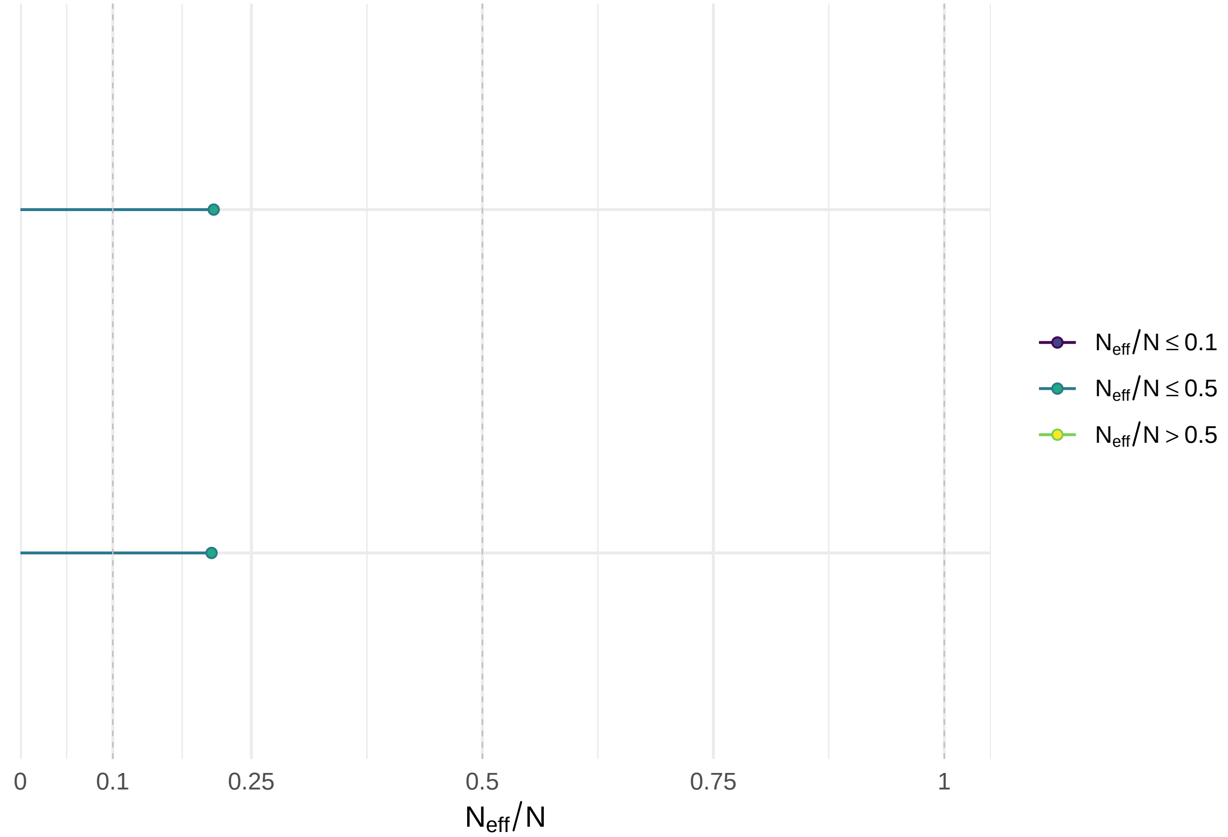
```
mcmc_nuts_divergence(np_cp, lp_cp)
```



```
rhats <- rhat(fit_usual)
mcmc_rhat(rhats)
```



```r
neff_ncp = neff_ratio(fit_usual, pars = c("a", "b"))
mcmc_neff(neff_ncp)
```



Recuperando as estimativas e calculando as medidas:

```
fit_usual_summary = summary(fit_usual)

fit_usual_summary$summary
```

```
##               mean     se_mean        sd         2.5%          25%          50%
## a        0.5540881 0.003701192 0.1692786    0.2983317    0.4337424    0.5283417
## b       16.9437765 0.088320502 4.0171780   10.8781972   14.0441733   16.3479516
## lp__  -597.7297692 0.022767083 1.0234818 -600.4872953 -598.0879006 -597.4181400
##                75%       97.5%    n_eff     Rhat
## a        0.6455283    0.9642769 2091.804 1.000084
## b       19.1345575   26.8031764 2068.803 1.000091
## lp__  -597.0160757 -596.7603169 2020.903 1.003236
```

```
fit_usual_summary$summary[,'50%']
```

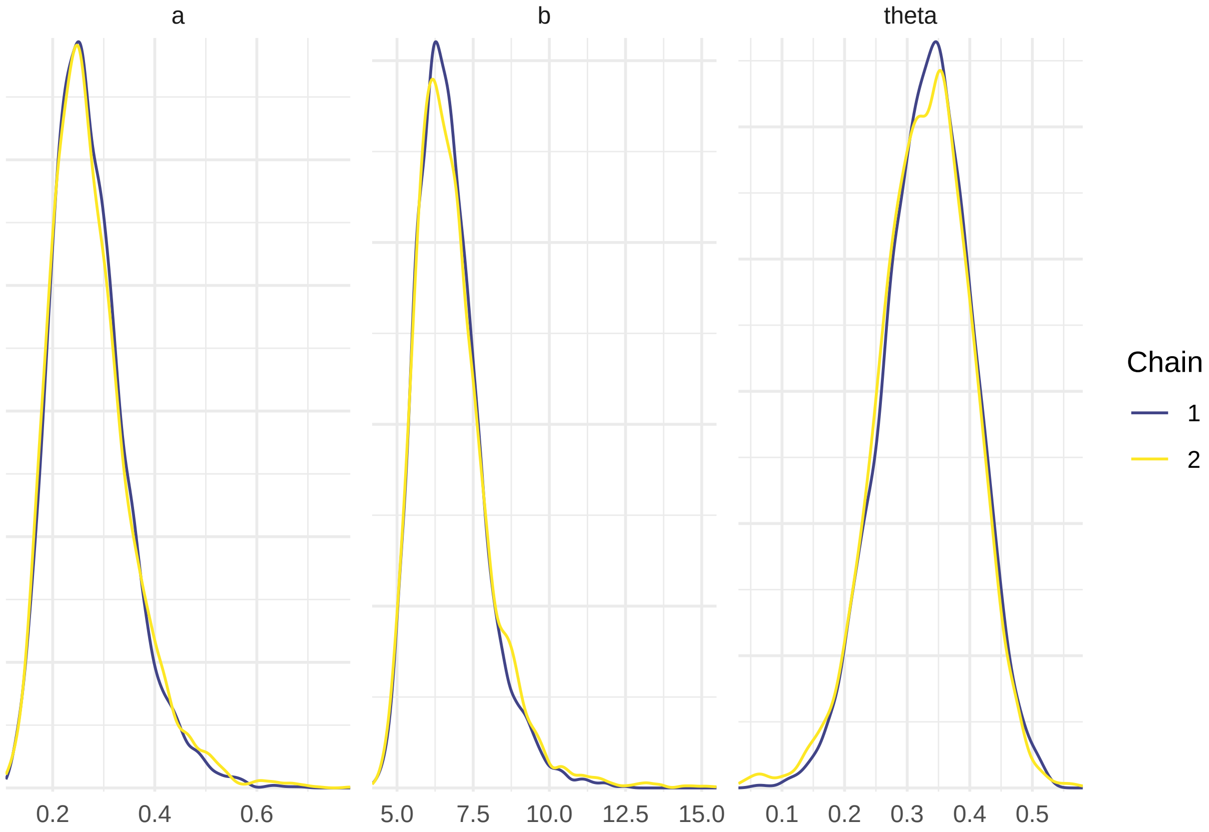
```
##            a            b         lp__
##    0.5283417   16.3479516 -597.4181400
```

```
tgca_a = fit_usual_summary$summary[,'50%']['a']
tgca_b = fit_usual_summary$summary[,'50%']['b']

calcula_dic(dados_tg$tempo/365, dados_tg$cens, log_veros,
           extract(fit_usual, "a") |> unlist(),
          extract(fit_usual, "b") |> unlist(), NULL) |>
  print(digits = 22)
```

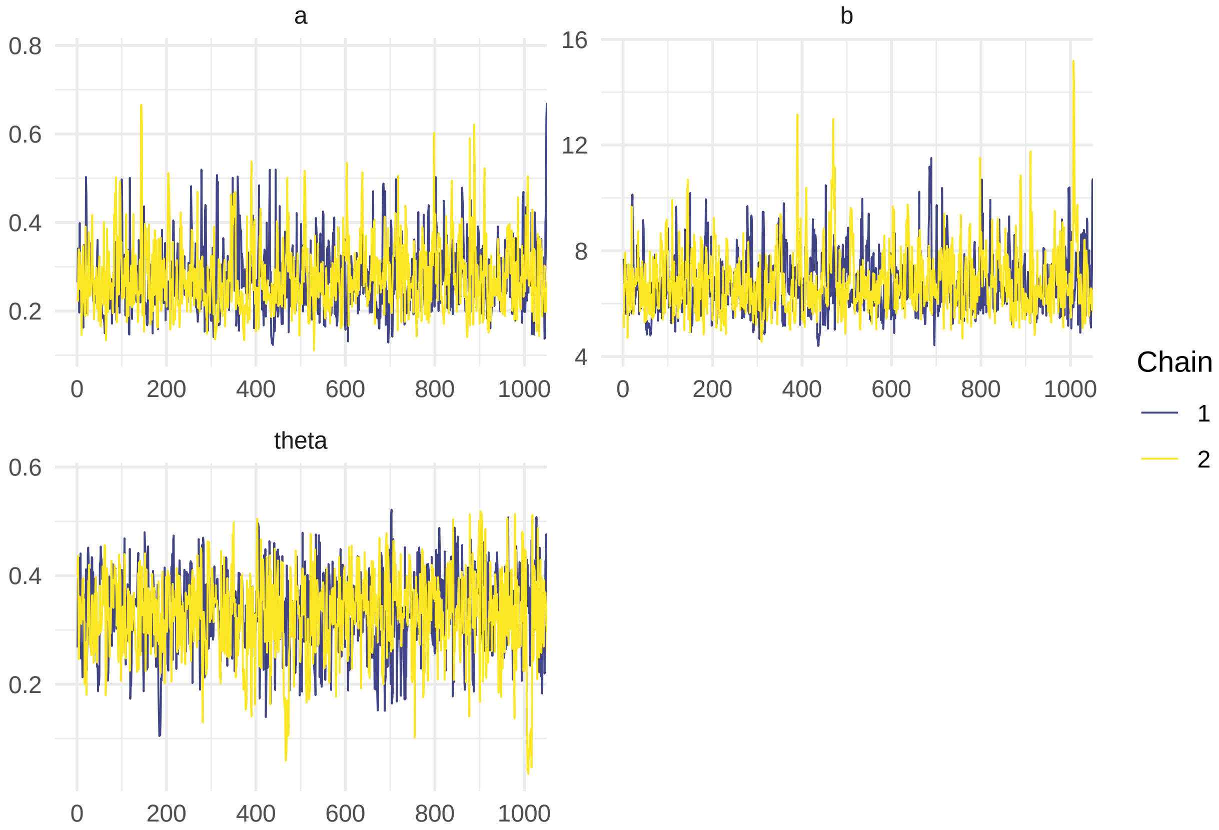
```
## [1] 1222.4336344764645
```

```
fit = stan( file = 'codigos_stan/mix_tgca.stan',
            data = list(N = nrow(dados_tg), T = dados_tg$tempo/365, D = dados_tg$cens),
            warmup = 1000, iter = 6000, chains = 2, cores = 2, seed = 154)
```

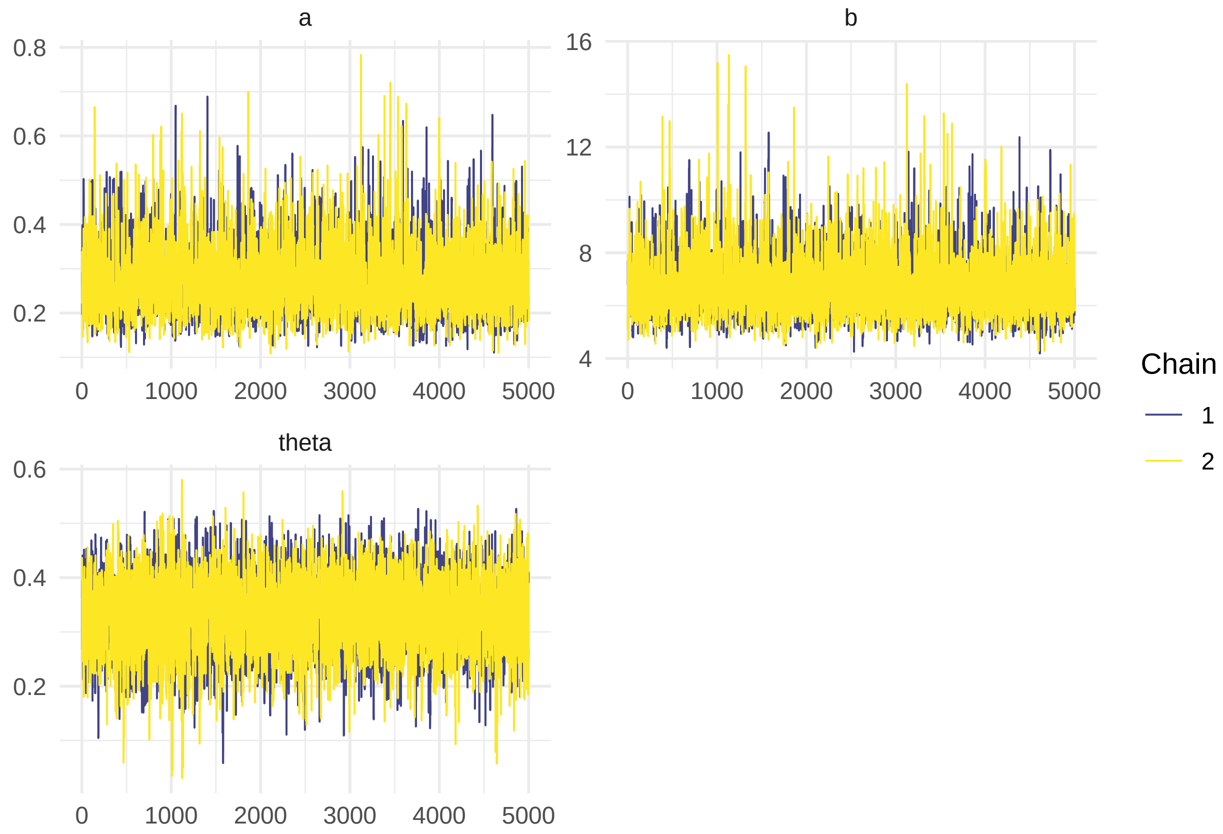
```
color_scheme_set("viridis")
mcmc_dens_overlay(fit, pars = c("a", "b", "theta"))
```



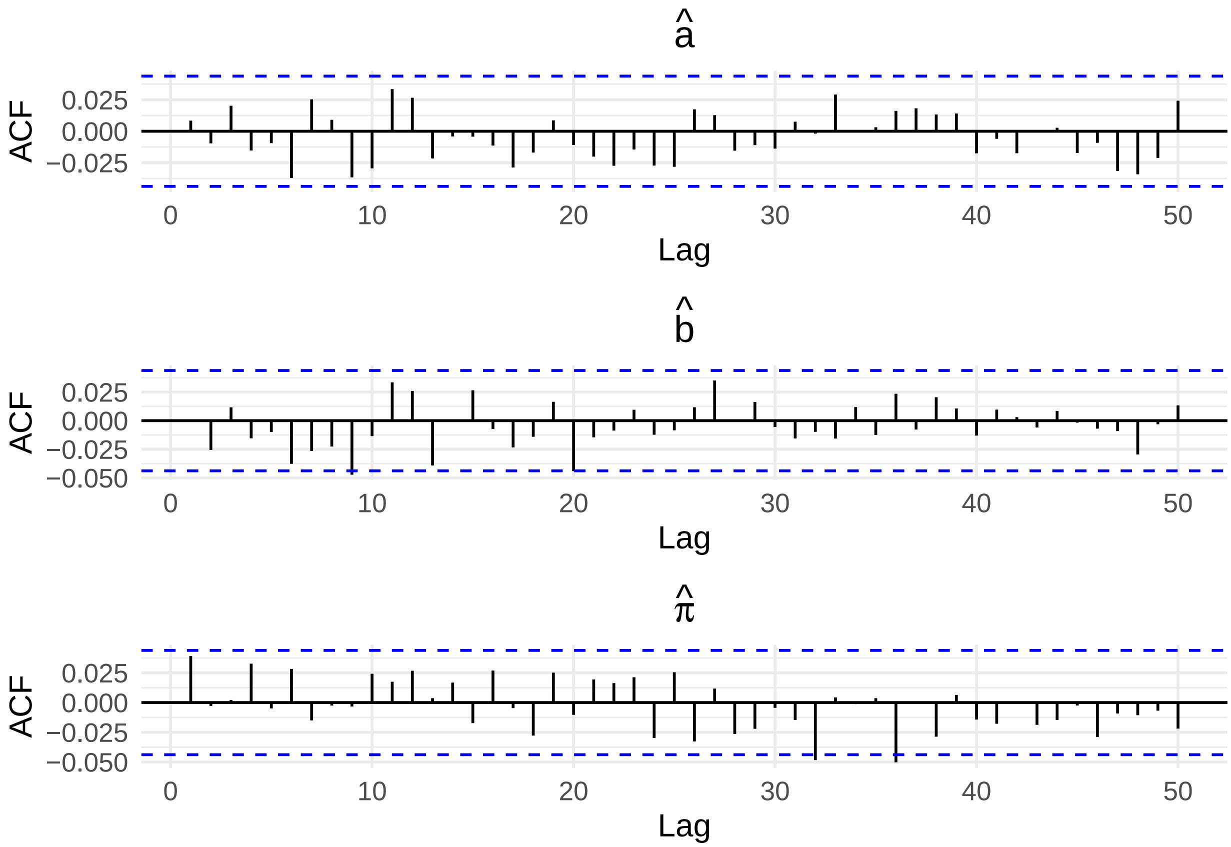
```
color_scheme_set("viridis")
mcmc_trace(fit, window = c(1,1000), pars = c("a", "b", "theta"),
           facet_args = list(nrow = 2))
```



```
color_scheme_set("viridis")
mcmc_trace(fit, pars = c("a", "b", "theta"),
           facet_args = list(nrow = 2))
```

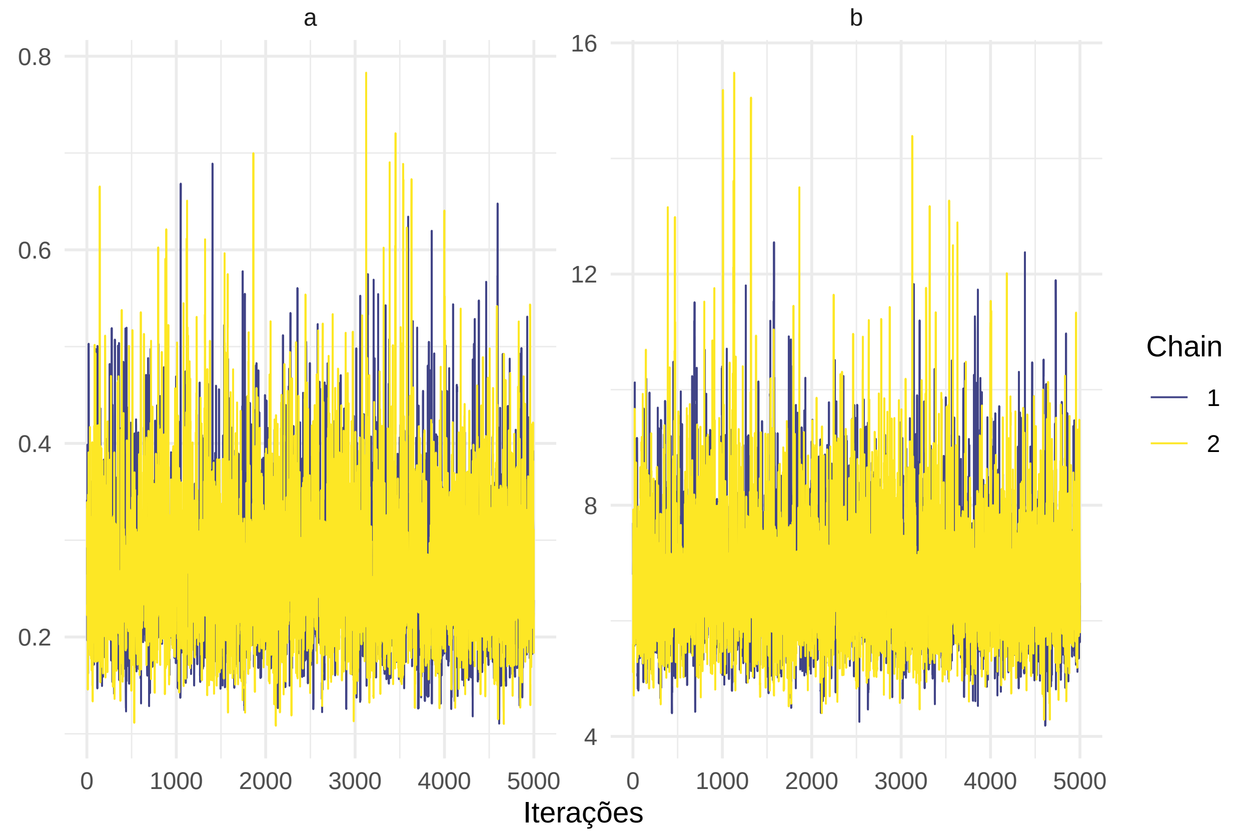


```
cadeias_df = data.frame(index = 1:length(extract(fit, "a")), a = extract(fit, "a"), b = extract(fit, "b

n = 2000

acf_a = ggAcf(cadeias_df[1:n,]$a, lag.max = 50) +
  theme_minimal() +
  labs(title = expression(hat(a))) +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size=12))

acf_b = ggAcf(cadeias_df[1:n,]$b, lag.max = 50) +
  theme_minimal() +
  labs(title = expression(hat(b))) +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size=12))

acf_theta = ggAcf(cadeias_df[1:n,]$theta, lag.max = 50) +
  theme_minimal() +
  labs(title = expression(hat(pi))) +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size=12))

gridExtra::grid.arrange(acf_a, acf_b, acf_theta)
```

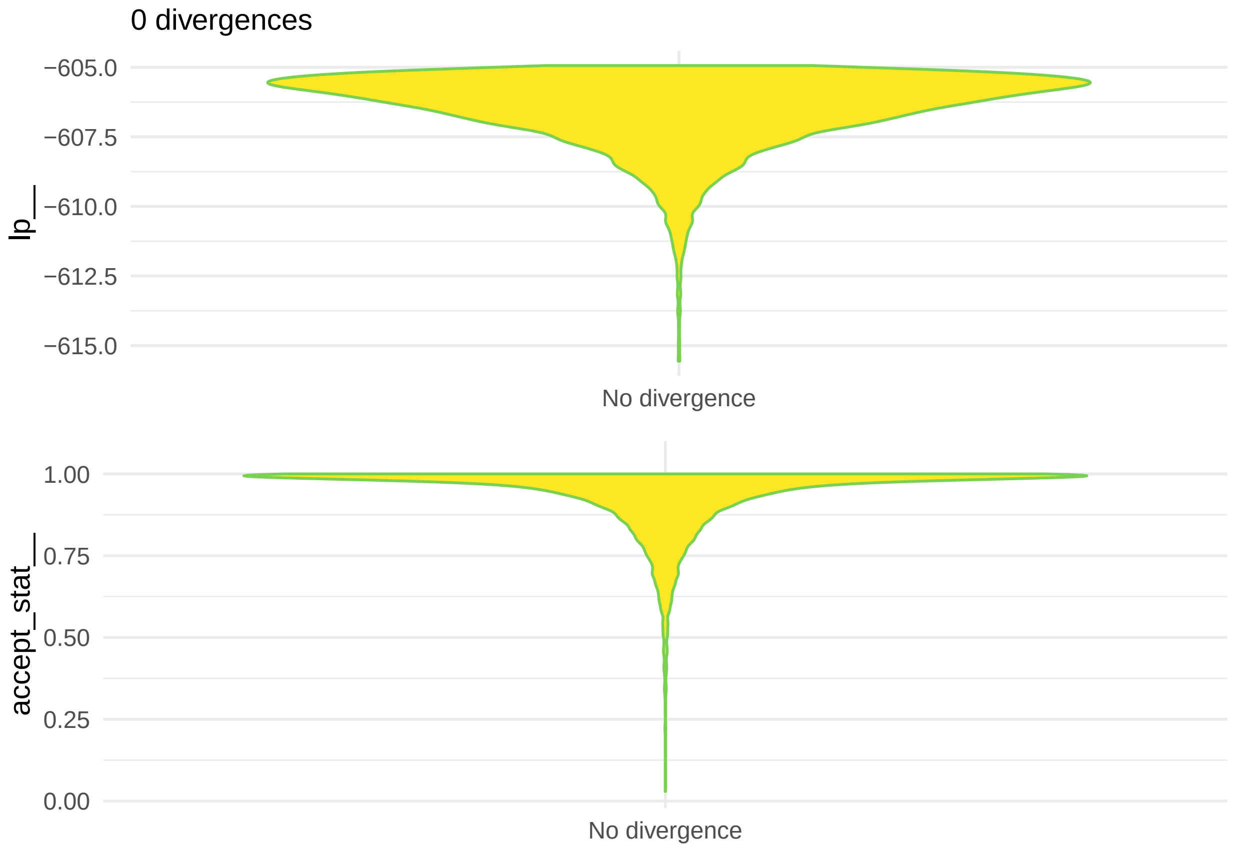


```r
lp_cp = log_posterior(fit)
np_cp = nuts_params(fit)

#mcmc_parcoord(as.array(fit), pars = c("a","b"), np = np_cp)

mcmc_trace(fit, pars = c("a","b"), np = np_cp) +
  xlab("Iterações")
```

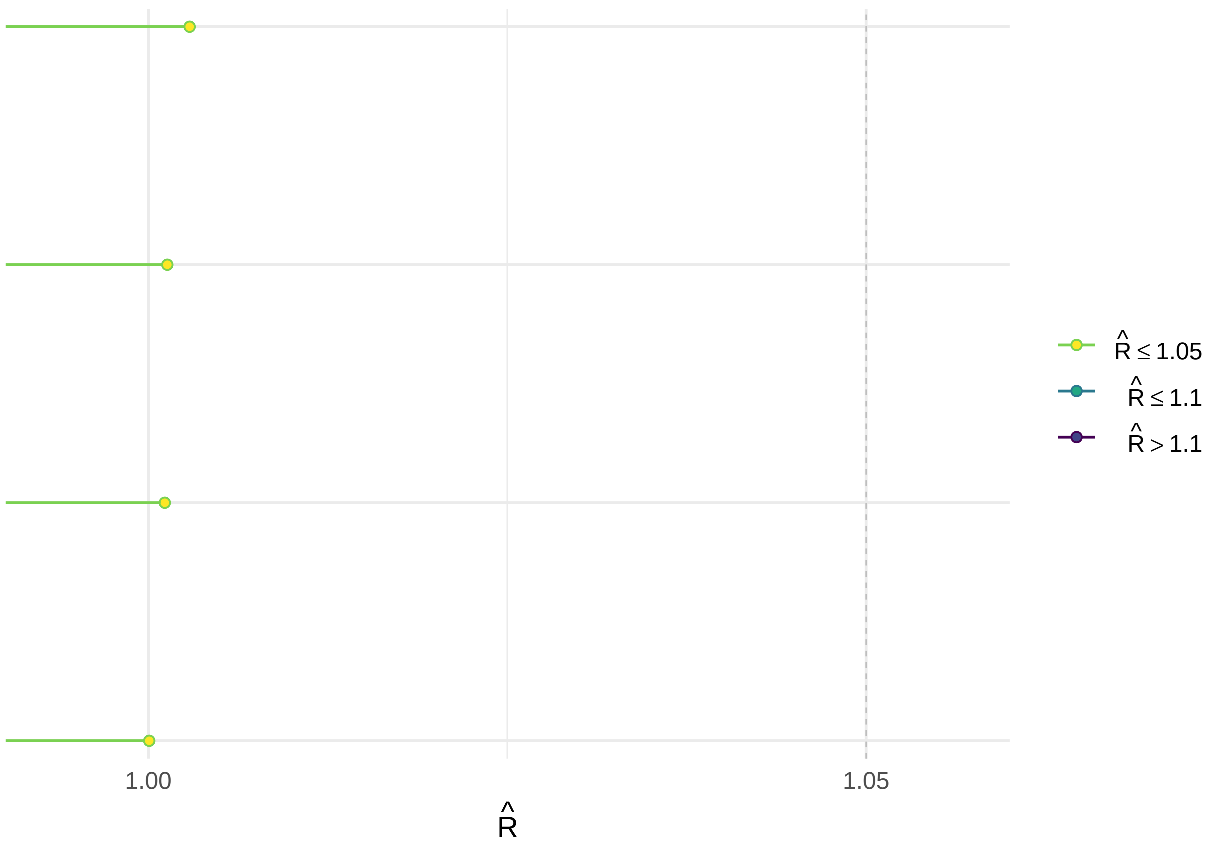
```
## No divergences to plot.
```



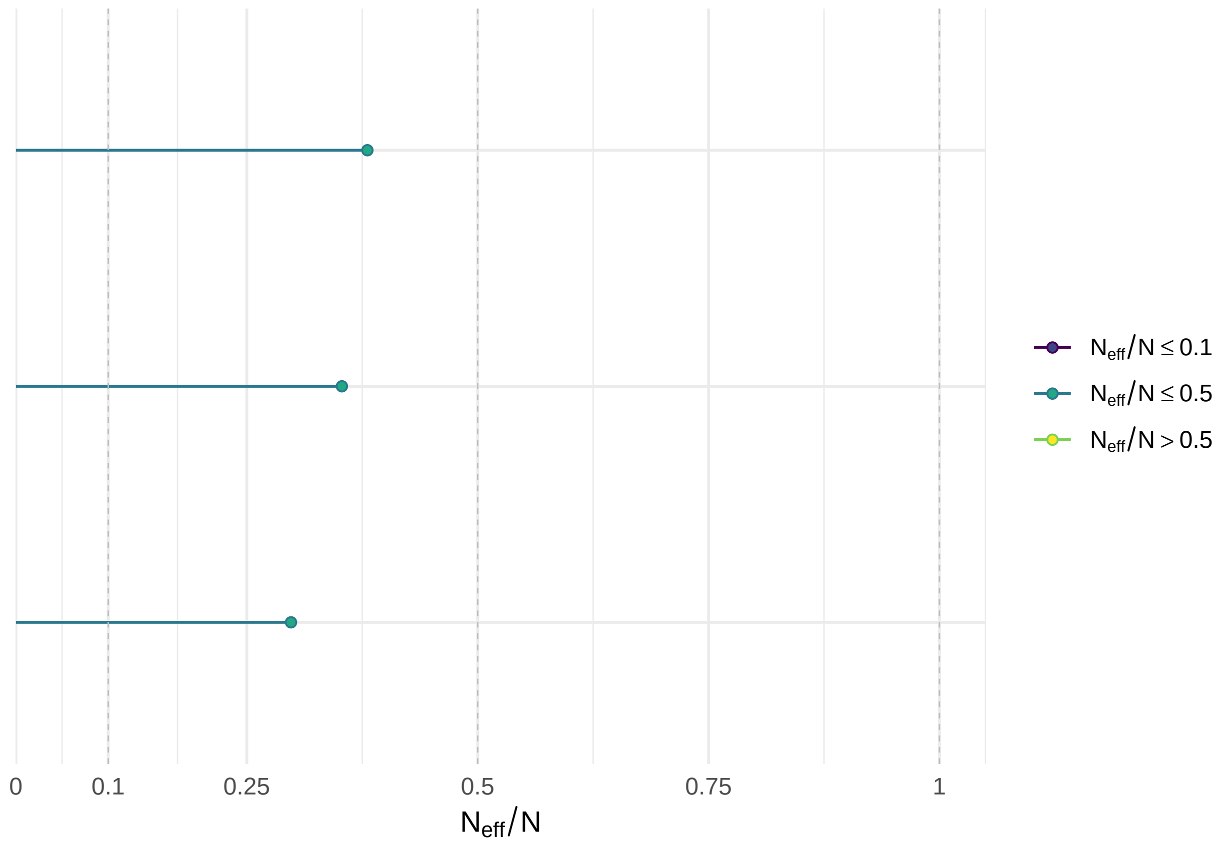
```
mcmc_nuts_divergence(np_cp, lp_cp)
```



```r
rhats <- rhat(fit)
mcmc_rhat(rhats)
```



```r
neff_ncp = neff_ratio(fit, pars = c("a", "b", "theta"))
mcmc_neff(neff_ncp)
```



Recuperando as estimativas e calculando as medidas:

```r
fit_summary = summary(fit)

fit_summary$summary
```

```
##                 mean     se_mean         sd         2.5%          25%
## a          0.2742000 0.001311849 0.07793910    0.1564256    0.2193026
## b          6.7705991 0.021574312 1.17754239    5.0413187    5.9596265
## theta      0.3302775 0.001172791 0.07235323    0.1821547    0.2824036
## lp__    -606.5328357 0.029117997 1.33415210 -609.9924344 -607.1023526
##                  50%          75%        97.5%    n_eff     Rhat
## a          0.2621206    0.3153845    0.4623192 3529.741 1.000063
## b          6.5949526    7.3659231    9.5345281 2979.060 1.001146
## theta      0.3336474    0.3804155    0.4628844 3806.054 1.001323
## lp__    -606.1789377 -605.5711757 -605.0459993 2099.364 1.002877
```

```r
fit_summary$summary[,'50%']
```

```
##            a            b        theta         lp__
##    0.2621206    6.5949526    0.3336474 -606.1789377
```

```r
tgca_a2 = fit_summary$summary[,'50%']['a']
tgca_b2 = fit_summary$summary[,'50%']['b']
tgca_p2 = fit_summary$summary[,'50%']['theta']
```

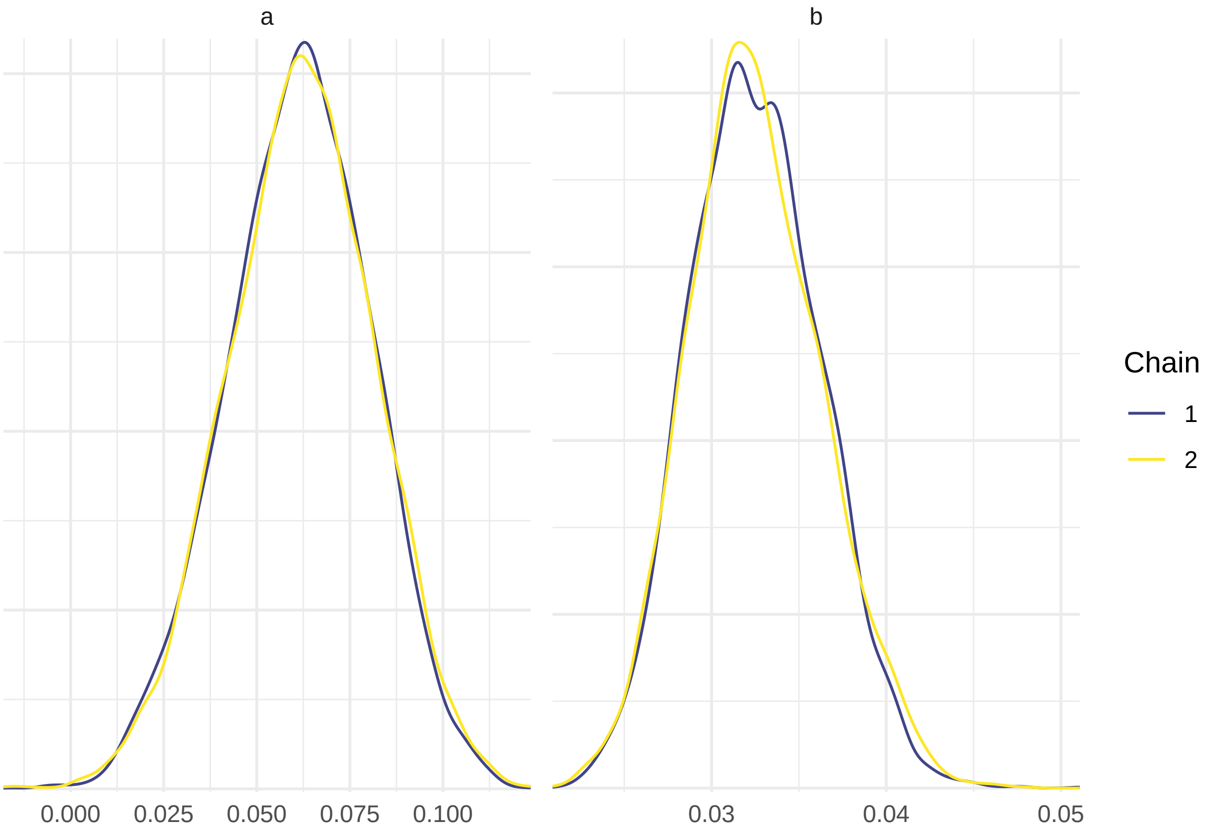
```
calcula_dic(dados_tg$tempo/365, dados_tg$cens, log_veros_mix,
           extract(fit, "a") |> unlist(),
           extract(fit, "b") |> unlist(),
           extract(fit, "theta") |> unlist()) |>
  print(digits = 22)
```

```
## [1] 1214.9003339179947
```

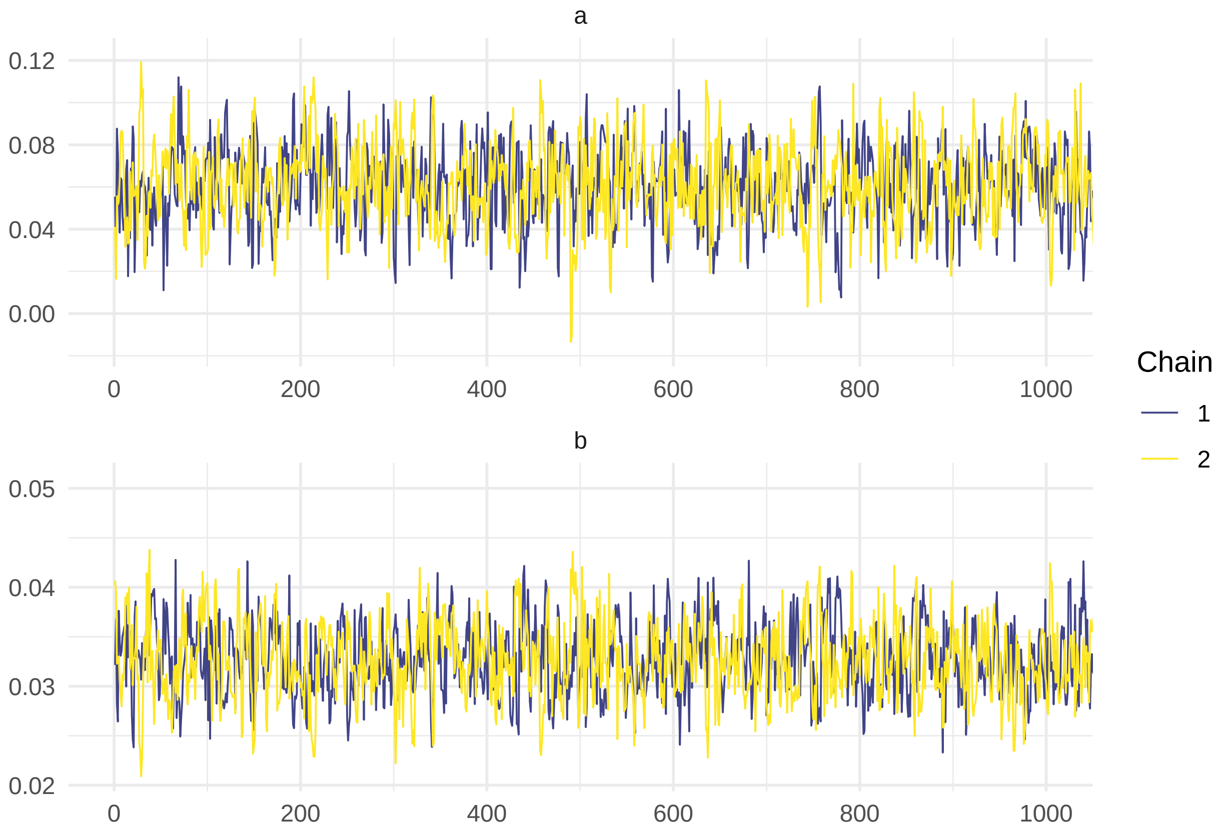
```
fit_def = stan( file = 'codigos_stan/def_tgca.stan',
            data = list(N = nrow(dados_tg), T = dados_tg$tempo/365, D = dados_tg$cens),
            warmup = 1000, iter = 6000, chains = 2, cores = 2, seed = 154)
```

```
## Warning in readLines(file, warn = TRUE): linha final incompleta encontrada em
## 'C:\Users\oandr\Documents\TG\codigos_stan\def_tgca.stan'
```

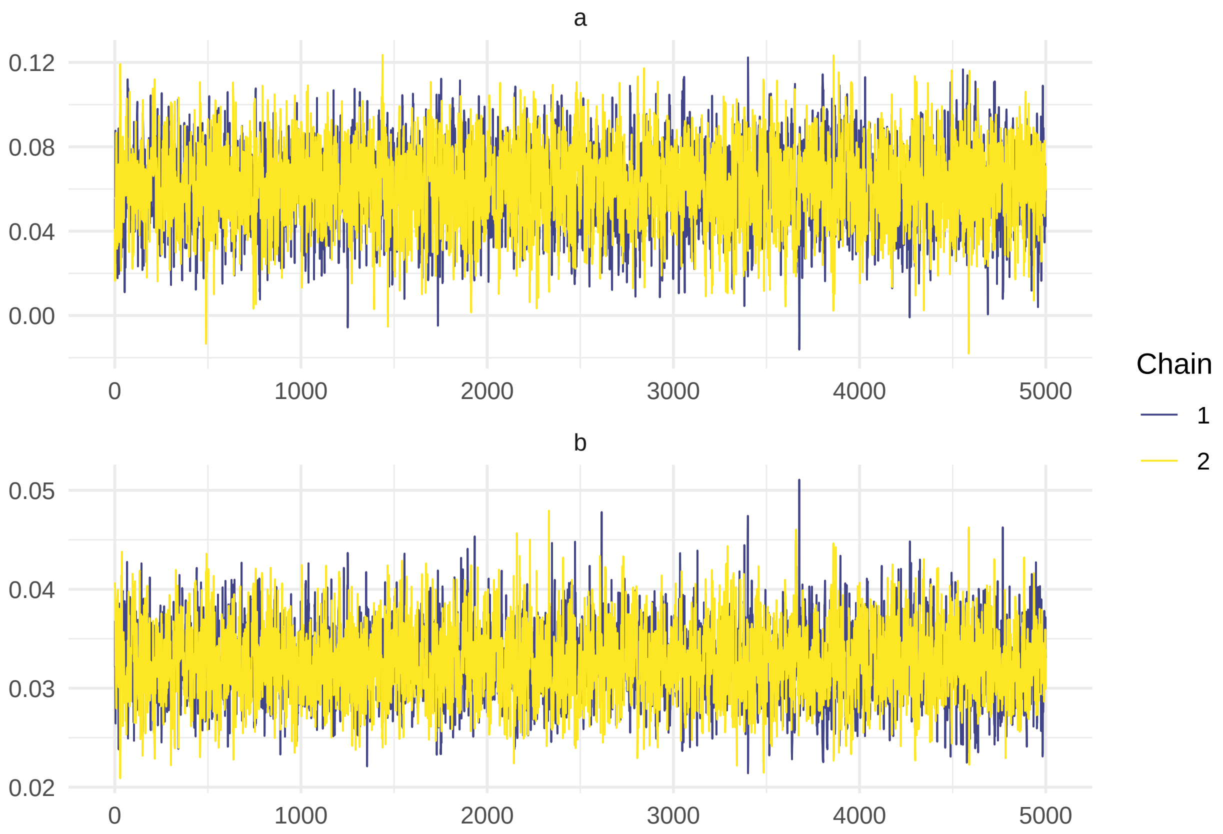
```
color_scheme_set("viridis")
mcmc_dens_overlay(fit_def, pars = c("a", "b"))
```



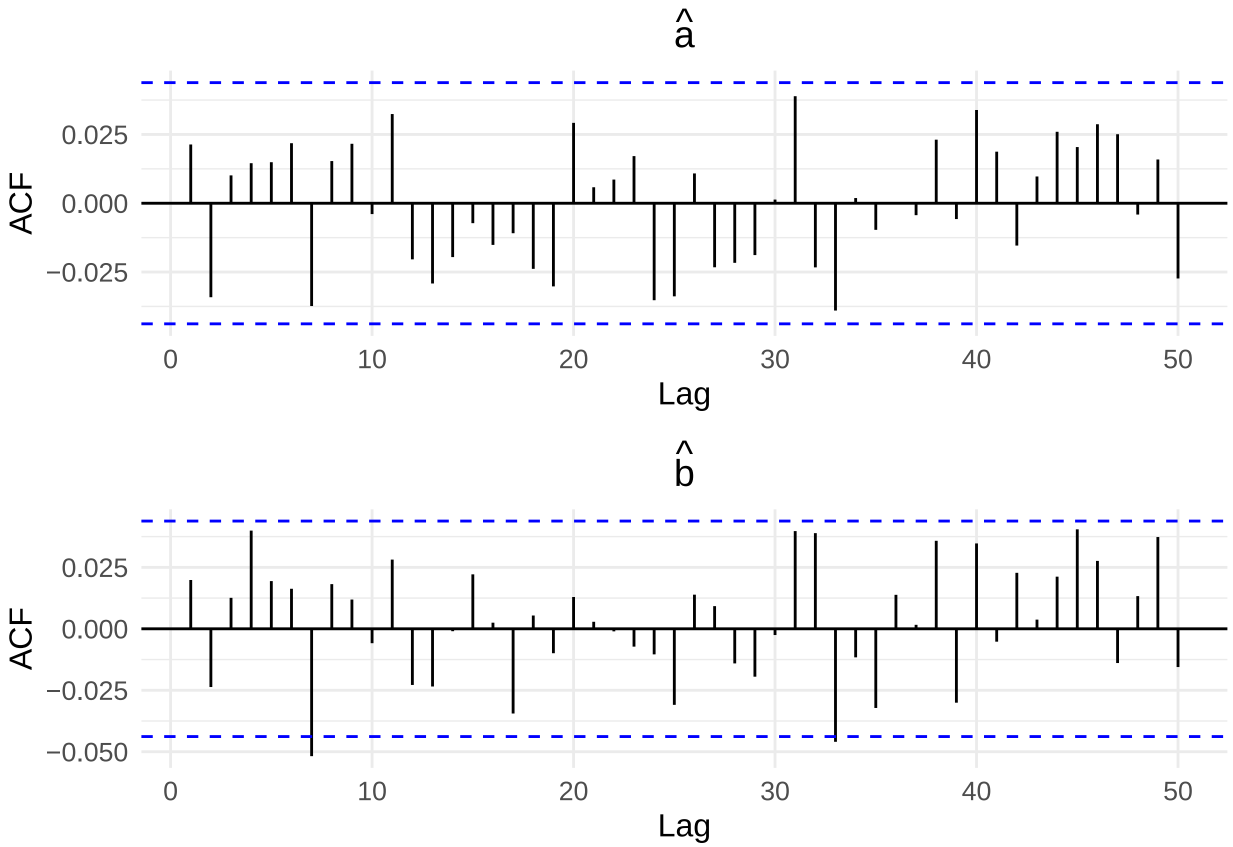
```
color_scheme_set("viridis")
mcmc_trace(fit_def, window = c(1,1000), pars = c("a", "b"),
           facet_args = list(nrow = 2))
```



```
color_scheme_set("viridis")
mcmc_trace(fit_def, pars = c("a", "b"),
           facet_args = list(nrow = 2))
```

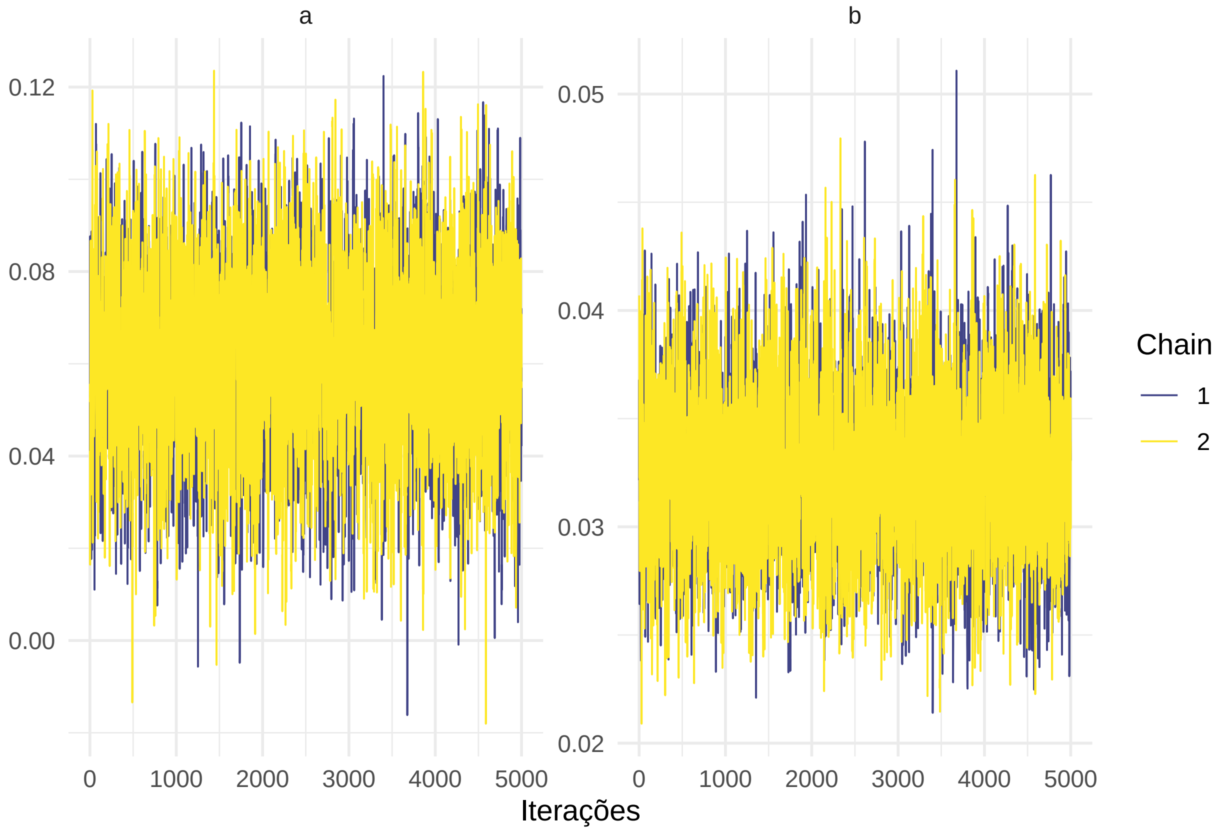


```
cadeias_df = data.frame(index = 1:length(extract(fit_def, "a")), a = extract(fit_def, "a"), b = extract

n = 2000

acf_a = ggAcf(cadeias_df[1:n,]$a, lag.max = 50) +
  theme_minimal() +
  labs(title = expression(hat(a))) +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size=12))

acf_b = ggAcf(cadeias_df[1:n,]$b, lag.max = 50) +
  theme_minimal() +
  labs(title = expression(hat(b))) +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size=12))

gridExtra::grid.arrange(acf_a, acf_b)
```

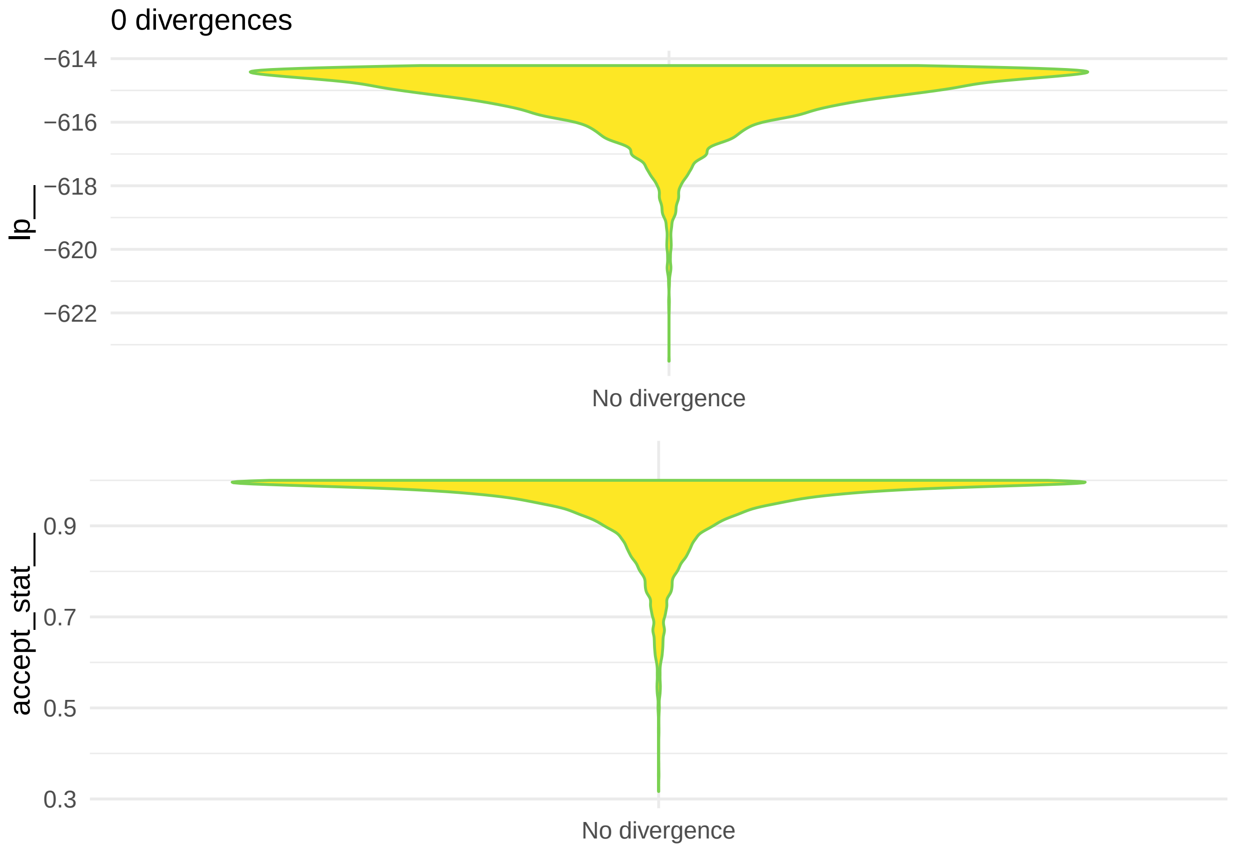


```
lp_cp = log_posterior(fit_def)
np_cp = nuts_params(fit_def)

#mcmc_parcoord(as.array(fit_def), pars = c("a","b"), np = np_cp)

mcmc_trace(fit_def, pars = c("a","b"), np = np_cp) +
  xlab("Iterações")
```

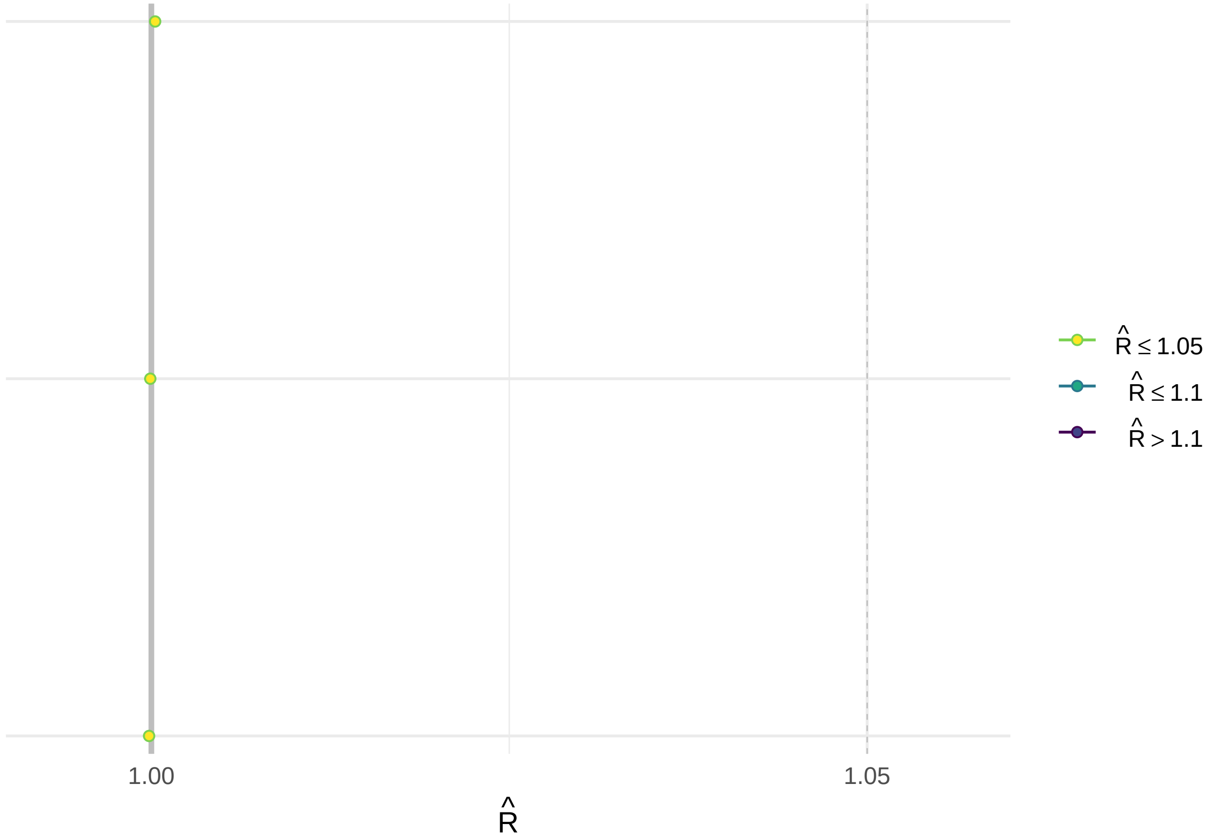
```
## No divergences to plot.
```



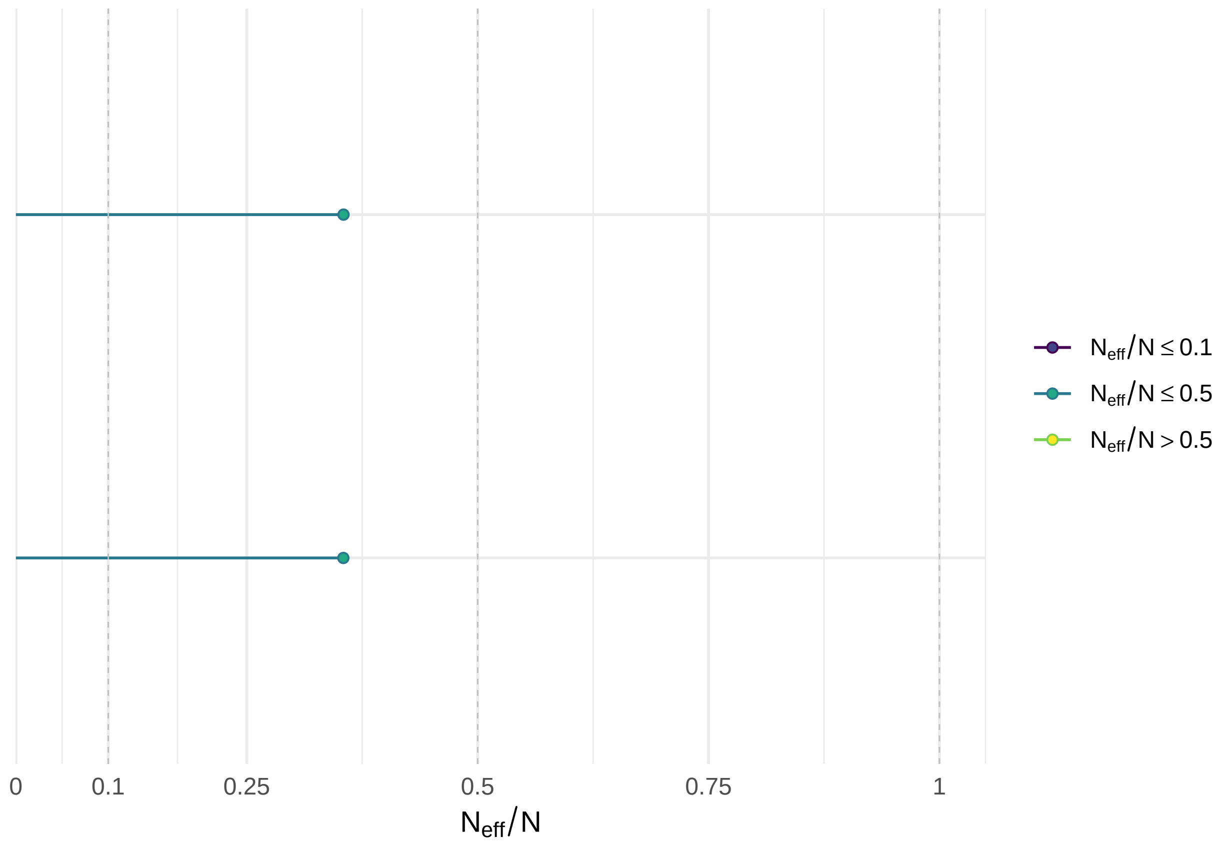
```
mcmc_nuts_divergence(np_cp, lp_cp)
```



```
rhats <- rhat(fit_def)
mcmc_rhat(rhats)
```



```
neff_ncp = neff_ratio(fit_def, pars = c("a", "b"))
mcmc_neff(neff_ncp)
```



Recuperando as estimativas e calculando as medidas:

```
fit_summary_def = summary(fit_def)

fit_summary_def$summary
```

```
##                mean      se_mean          sd          2.5%           25%
## a        0.06150619 3.268103e-04 0.019459508    0.02206330    0.04854018
## b        0.03253568 6.333458e-05 0.003772216    0.02563339    0.02987902
## lp__  -615.23118612 1.644536e-02 0.987111066 -617.86532703 -615.62566009
##                 50%           75%        97.5%    n_eff      Rhat
## a        0.06193181    0.07498323    0.09809185 3545.458 0.9999253
## b        0.03233709    0.03504657    0.04026717 3547.408 0.9998415
## lp__  -614.94205479 -614.51766552 -614.24081054 3602.842 1.0002653
```

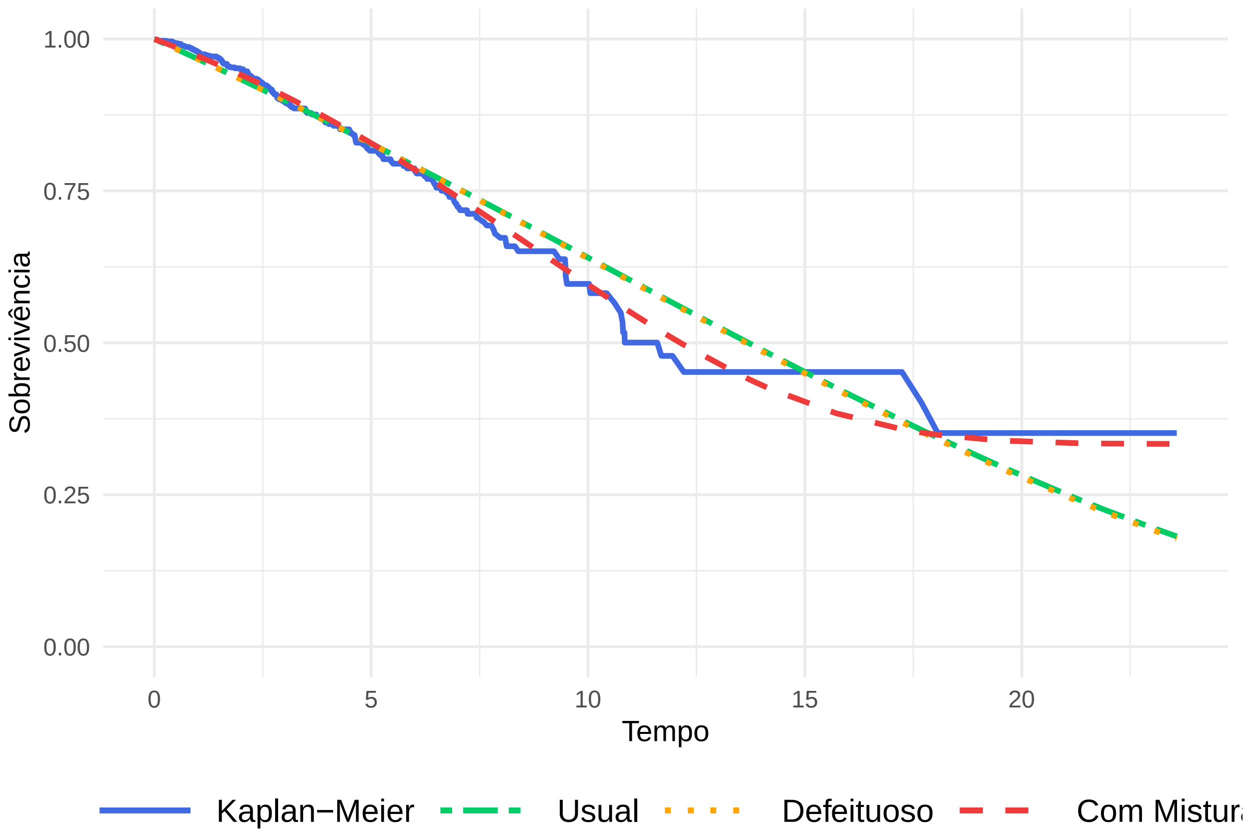
```
fit_summary_def$summary[,'50%']
```

```
##             a             b          lp__
##    0.06193181    0.03233709 -614.94205479
```

```
tgca_a3 = fit_summary_def$summary[,'50%']['a']
tgca_b3 = fit_summary_def$summary[,'50%']['b']



calcula_dic(dados_tg$tempo/365, dados_tg$cens, log_veros_def,
           extract(fit_def, "a") |> unlist(),
           extract(fit_def, "b") |> unlist(), NULL) |>
  print(digits = 22)
```

```
## [1] 1223.5034654373019
```

```
### Figura

x = dados_tg$tempo/365

kaplan_meier_s = survfit(Surv(tempo/365, cens) ~ 1, data = dados_tg)

dados_km = data.frame(kaplan_meier_s$time, kaplan_meier_s$surv, kaplan_meier_s$n.event)
colnames(dados_km) = c('Tempo', 'Sobrevivência', 'Evento')


ggplot() +
  geom_line(aes(x = Tempo, y = Sobrevivência, colour = "a", linetype = "a"),
            data = dados_km, size = 1) +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(x = 'Tempo') +
  theme_minimal() +
  geom_line(
    mapping=aes(x=x, y = pgompertz(x,tgca_a , #fit_usual_summary$summary[,1]['a']
                                   tgca_b, lower.tail = F), #fit_usual_summary$summary[,1]['b']
                colour = "b", linetype = "b"),
    size = 1) +
  geom_line(
    mapping=aes(x=x, y = flexsurv::pgompertz(x, tgca_a3, #fit_summary_def$summary[,1]['a']
                                   tgca_b3, lower.tail = F), #fit_summary_def$summary[,1]['b']
                colour = "c", linetype = "c"),
    size = 1) +
  geom_line(
    mapping=aes(x=x, y= tgca_p2 + (1 - tgca_p2)*pgompertz(x, tgca_a2, tgca_b2, lower.tail = F), #fit_su
                colour = "d", linetype = "d"),
    size = 1) +
  ylim(0,1) +
  theme(legend.position = 'bottom', legend.text = element_text(size=12), legend.key.width= unit(1.5, 'c
  scale_color_manual(name = "",
                     values = c(
                       "royalblue",
                       "springgreen3",
                       'orange',
                       "brown2"),
                     labels = c("Kaplan-Meier","Usual",
                                "Defeituoso",
                                "Com Mistura")) +
  scale_linetype_manual(name = "", values=c("solid", "twodash",
                                            "dotted", "dashed"),
                        labels = c("Kaplan-Meier",
                                   "Usual",
                                   "Defeituoso",
                                   "Com Mistura"))
```



```r
ggsave(filename = 'figuras/tgca_bayes.pdf', units = 'in', width = 7, height = 5)
```