R Notebook

```
library(spdep)
library(Matrix)
library(dplyr)
library(tidyr)
library(spatialreg)
```

Générer des données (modèle latent)

Les le processus de génération des données pour le modèle latent est simplement un modèle SAR dont les données sont ensuite censurées.

```
source("simul data.R")

df_censure_latent <- gen_data_latent(n = 100,
                                     beta_reel = c(1, 1),
                                     lambda_reel = 0.6,
                                     sigma2_reel = 1,
                                     seed = 6390)

df_latent <- df_censure_latent$donnee

W_latent <- df_censure_latent$W |>
  as.matrix()
W_latent <- W_latent |>
  mat2listw()
```

```
## Warning in mat2listw(W_latent): style is M (missing); style should be set to a
## valid value
```

```
# On réestime le modèle de base selon sa bonne forme
fit1.1 <- lagsarlm(y_latent ~ X1, data = df_latent, listw = W_latent)
summary(fit1.1)
```

Estimation d'un modèle SAR classique (approche fréquentiste)

```
##
## Call:lagsarlm(formula = y_latent ~ X1, data = df_latent, listw = W_latent)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
```

```
## -2.772394 -0.550517 -0.013323  0.640189  2.463999
##
## Type: lag
## Coefficients: (asymptotic standard errors)
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 1.036663   0.463657  2.2358  0.02536
## X1          1.023647   0.031623 32.3707  < 2e-16
##
## Rho: 0.62783, LR test value: 9.0694, p-value: 0.0025993
## Asymptotic standard error: 0.16711
##     z-value: 3.757, p-value: 0.00017197
## Wald statistic: 14.115, p-value: 0.00017197
##
## Log likelihood: -141.0658 for lag model
## ML residual variance (sigma squared): 0.97068, (sigma: 0.98523)
## Number of observations: 100
## Number of parameters estimated: 4
## AIC: 290.13, (AIC for lm: 297.2)
## LM test for residual autocorrelation
## test value: 4.4639, p-value: 0.034618
```

```
# On estime le modèle original, mais sur les données censurées
fit1.2 <- lagsarlm(y_obs ~ X1, data = df_latent, listw = W_latent)
summary(fit1.2)
```

```
##
## Call:lagsarlm(formula = y_obs ~ X1, data = df_latent, listw = W_latent)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.13285 -0.55986 -0.16580  0.47492  6.07382
##
## Type: lag
## Coefficients: (asymptotic standard errors)
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept) 2.030666   0.748006  2.7148  0.006632
## X1          0.856929   0.038496 22.2600 < 2.2e-16
##
## Rho: 0.32577, LR test value: 1.3272, p-value: 0.2493
## Asymptotic standard error: 0.24817
##     z-value: 1.3127, p-value: 0.18928
## Wald statistic: 1.7232, p-value: 0.18928
##
## Log likelihood: -159.9689 for lag model
## ML residual variance (sigma squared): 1.4312, (sigma: 1.1963)
## Number of observations: 100
## Number of parameters estimated: 4
## AIC: 327.94, (AIC for lm: 327.26)
## LM test for residual autocorrelation
## test value: 7.1722, p-value: 0.0074043
```

Estimation d'un modèle SAR classique (approche bayésienne)

Estimation du modèle SAR avec Gibbs-MH

Méthode basée sur celle présentée dans "Introduction to spatial econometrics" de LeSage. Pour l'instant, l'estimation est seulement pour le modèle SAR avec une variable y non censurée.

Ajout d'un module pour la censure (modèle latent, estimation des variables latentes avec l'algo de Gibbs présenté par Geweke) à venir. Référence exacte à ajouter.

- Essentiellement, on rajoute une étape qui est un échantillonneur de gibbs pour chaque y latent estimé à partir des autres observations tel que $p(y_j|y_1, ..., y_{j-1}, y_{j+1}, ..., y_n)$

```
source("MH_Gibbs_latent.R")

fit1.1_gibbs <- MH_Gibbs_latent(
  y = df_latent$y_latent,
  X = as.matrix(df_latent[,c("X0", "X1")]),
  W = listw2mat(W_latent),
  c_beta = c(0, 0),
  c_lambda = 0.01,
  h_T = 1e+10,
  N = 100000,
  h_igamma = c(a = 0, b = 0),
  theta_0 = c(beta0 = 0, beta1 = 0,
              sigma2 = 0.5, lambda = 0.3)
  )
```

```
## ================================================================================
```

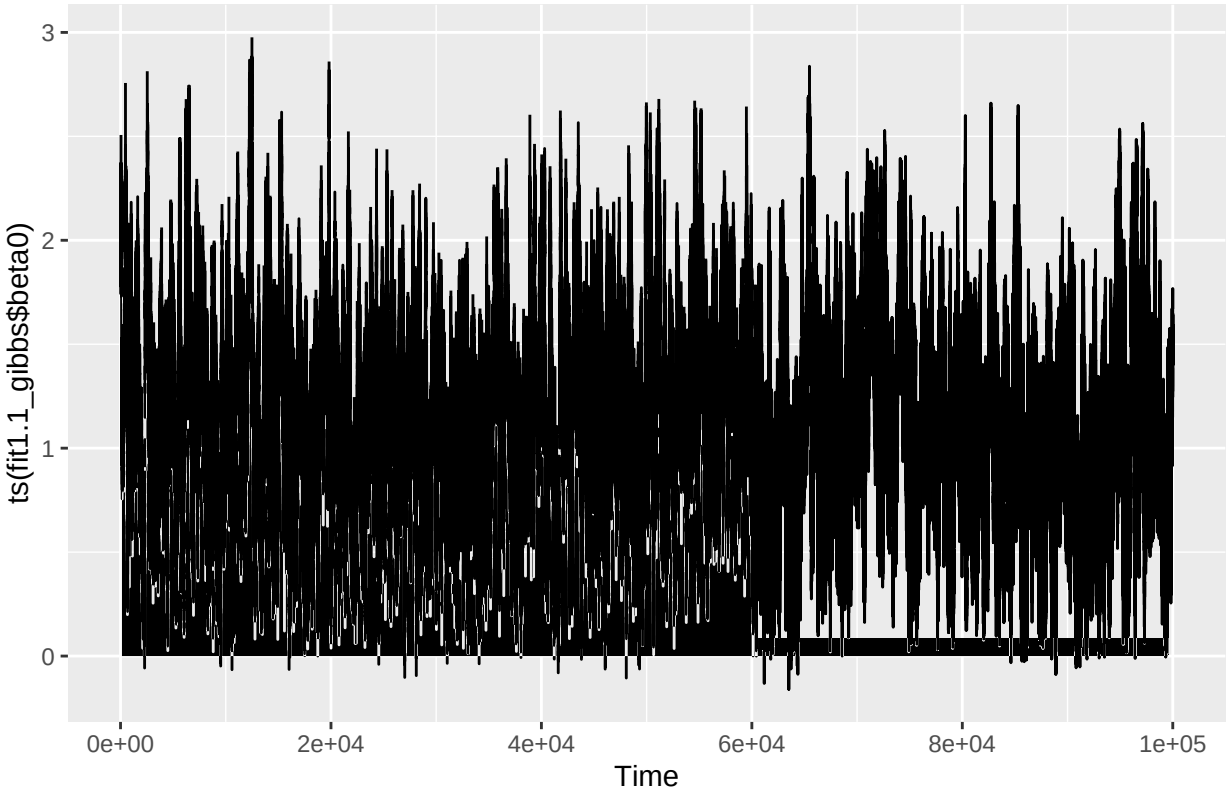
```
fit1.1_gibbs$accept |>
  mean(na.rm = TRUE)
```

```
## [1] 0.56277
```

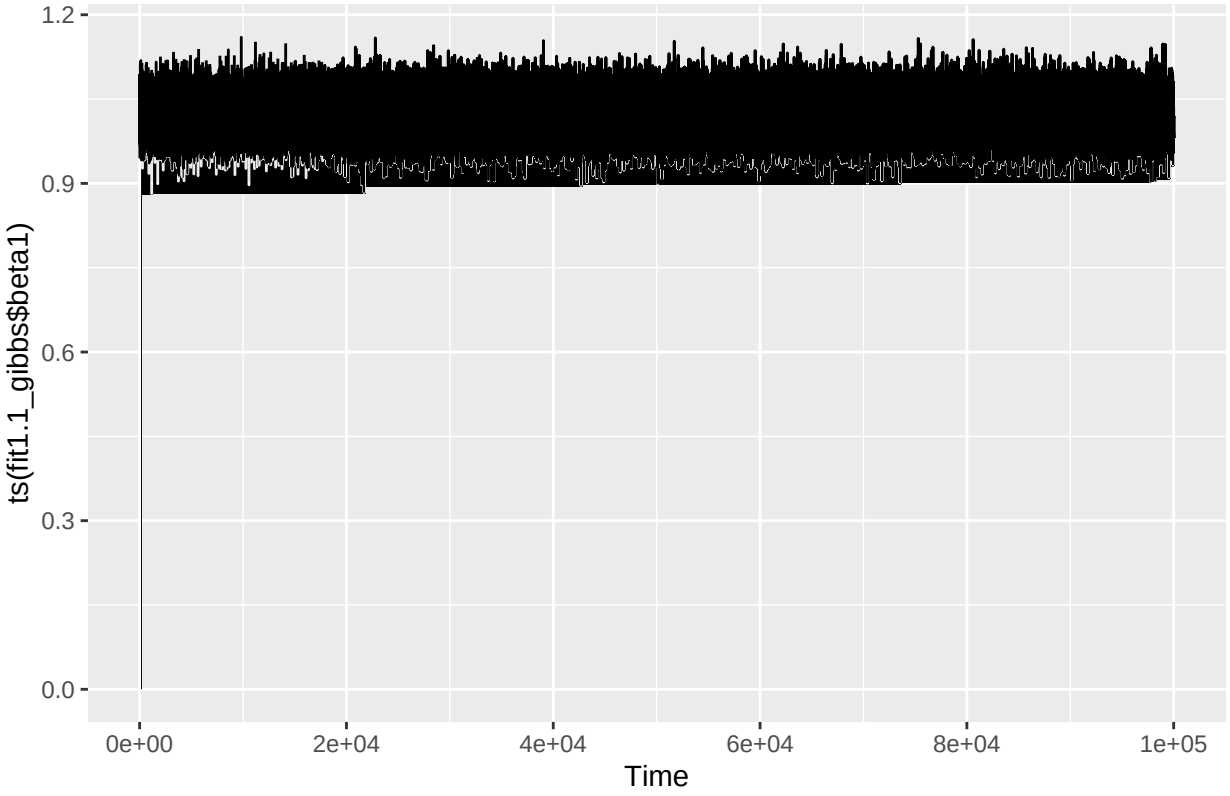
```
fit1.1_gibbs$beta0 |> ts() |> forecast::autoplot()
```

Complet (accept et rejet)

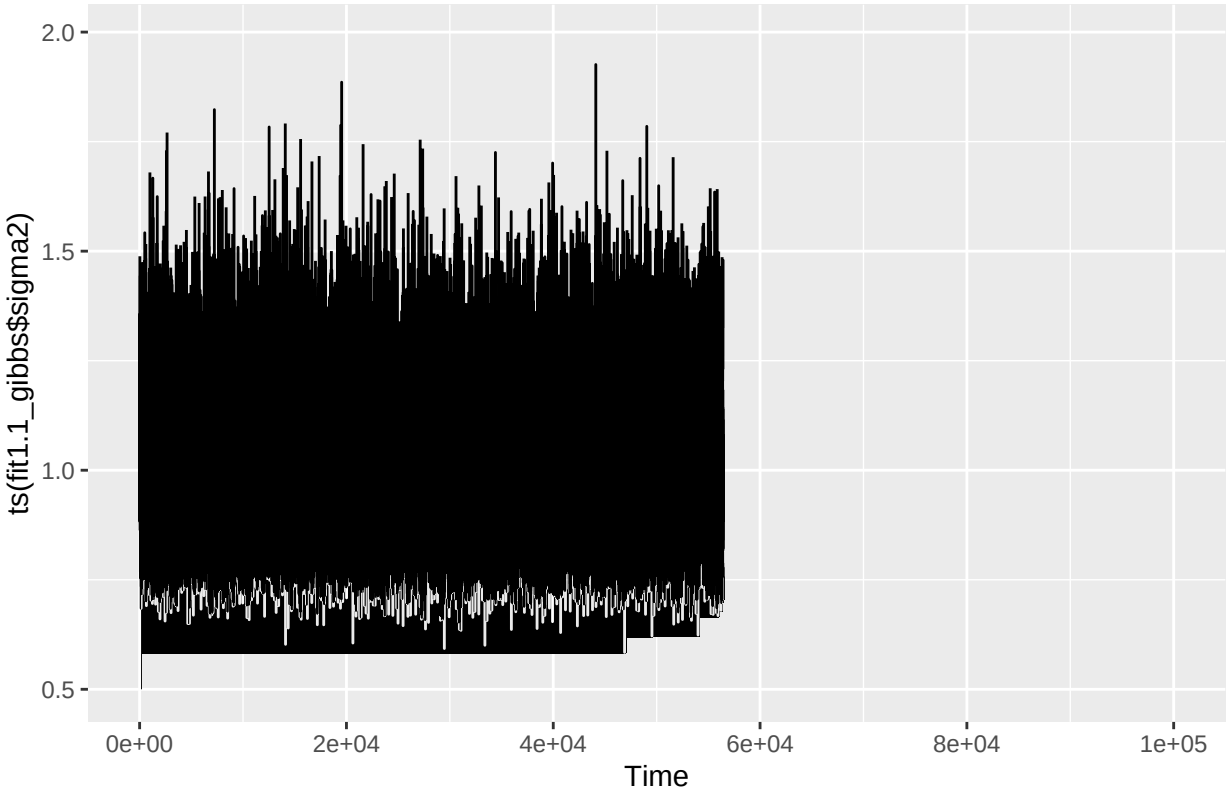
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```



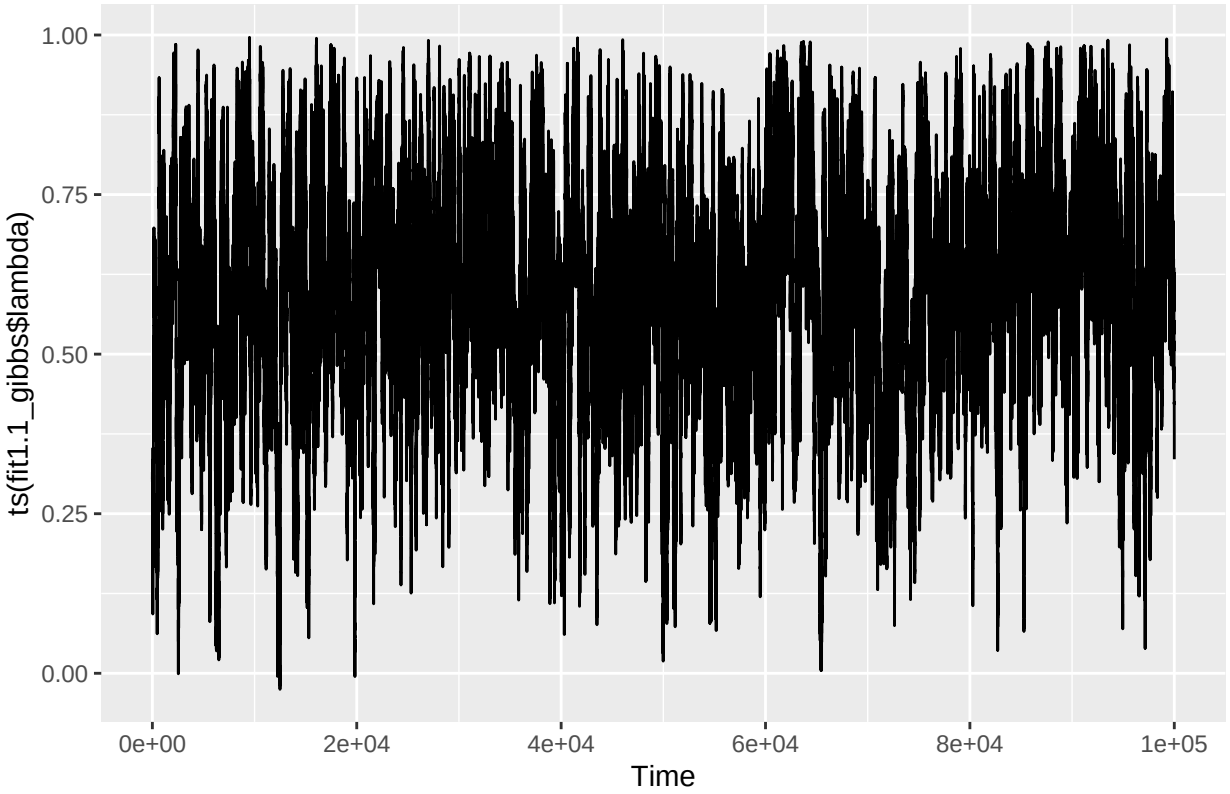

```
fit1.1_gibbs$beta1 |> ts() |> forecast::autoplot()
```



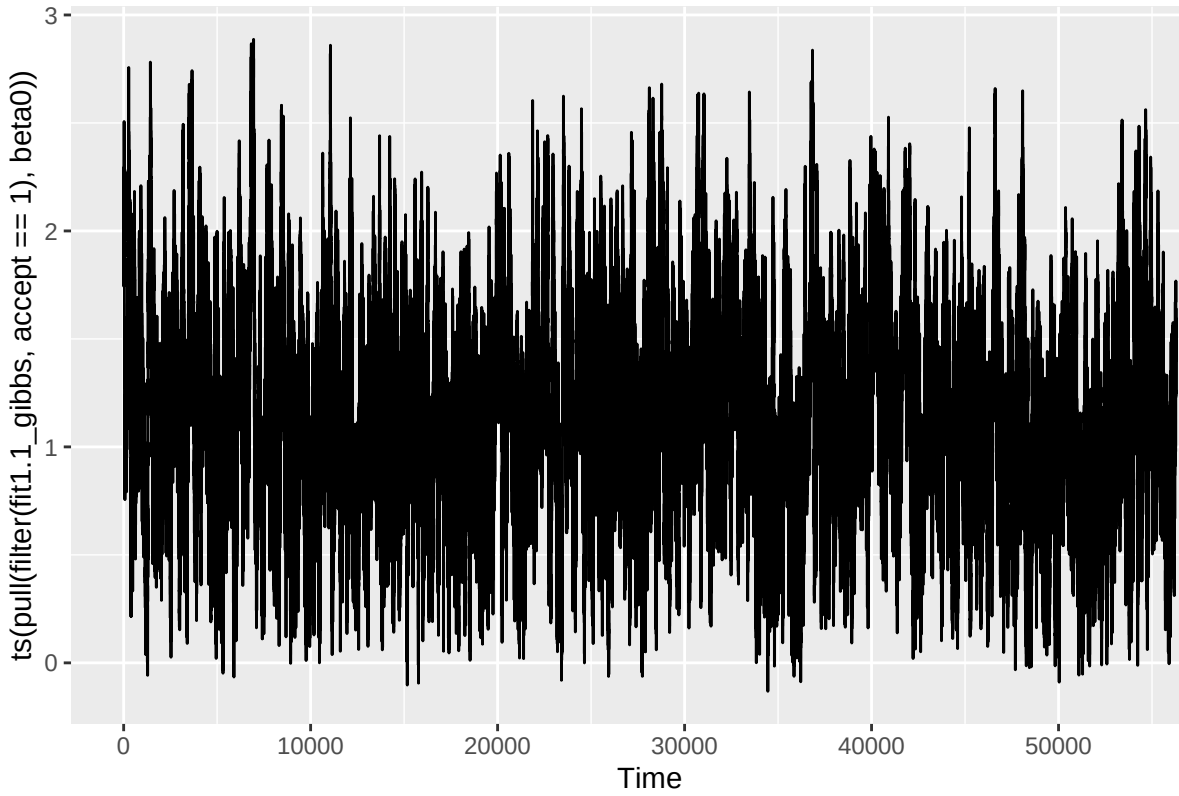

```
fit1.1_gibbs$sigma2 |> ts() |> forecast::autoplot()
```




```
fit1.1_gibbs$lambda |> ts() |> forecast::autoplot()
```

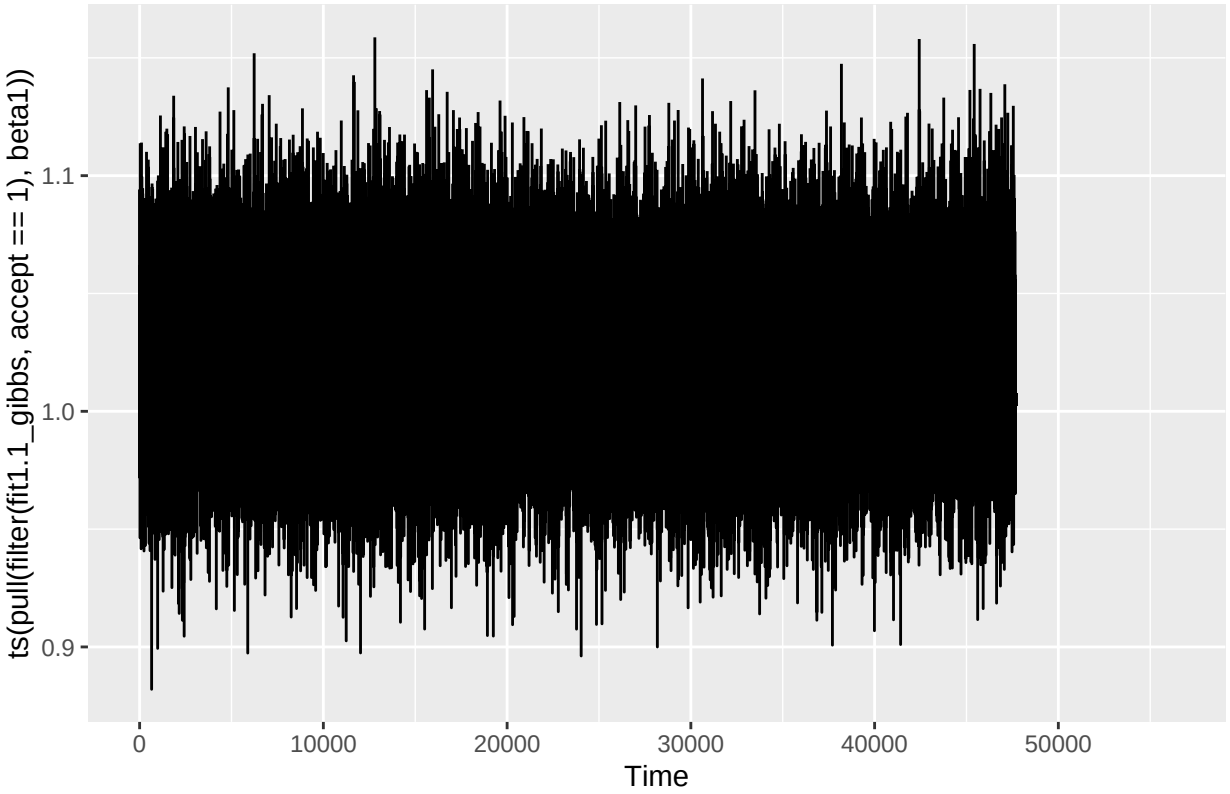



```
fit1.1_gibbs |>
  filter(accept == 1) |>
  pull(beta0) |> ts() |> forecast::autoplot()
```

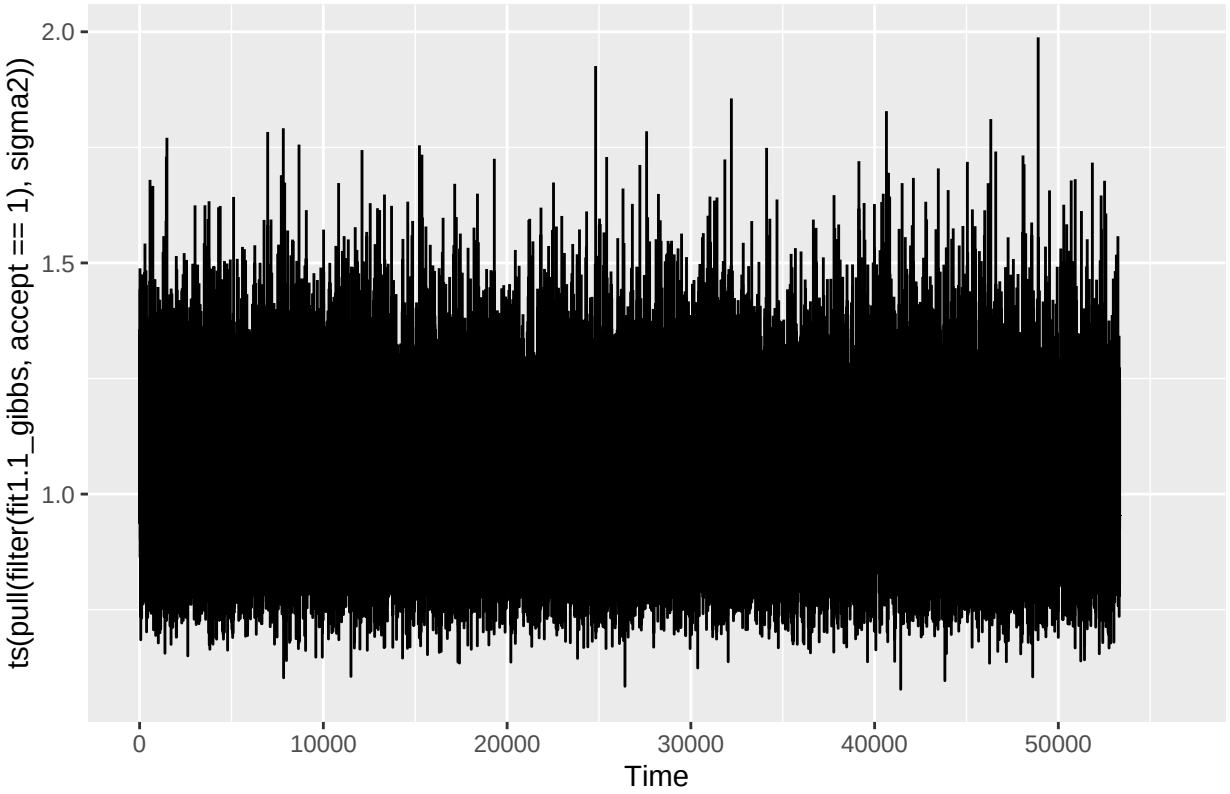


Complet (accept)

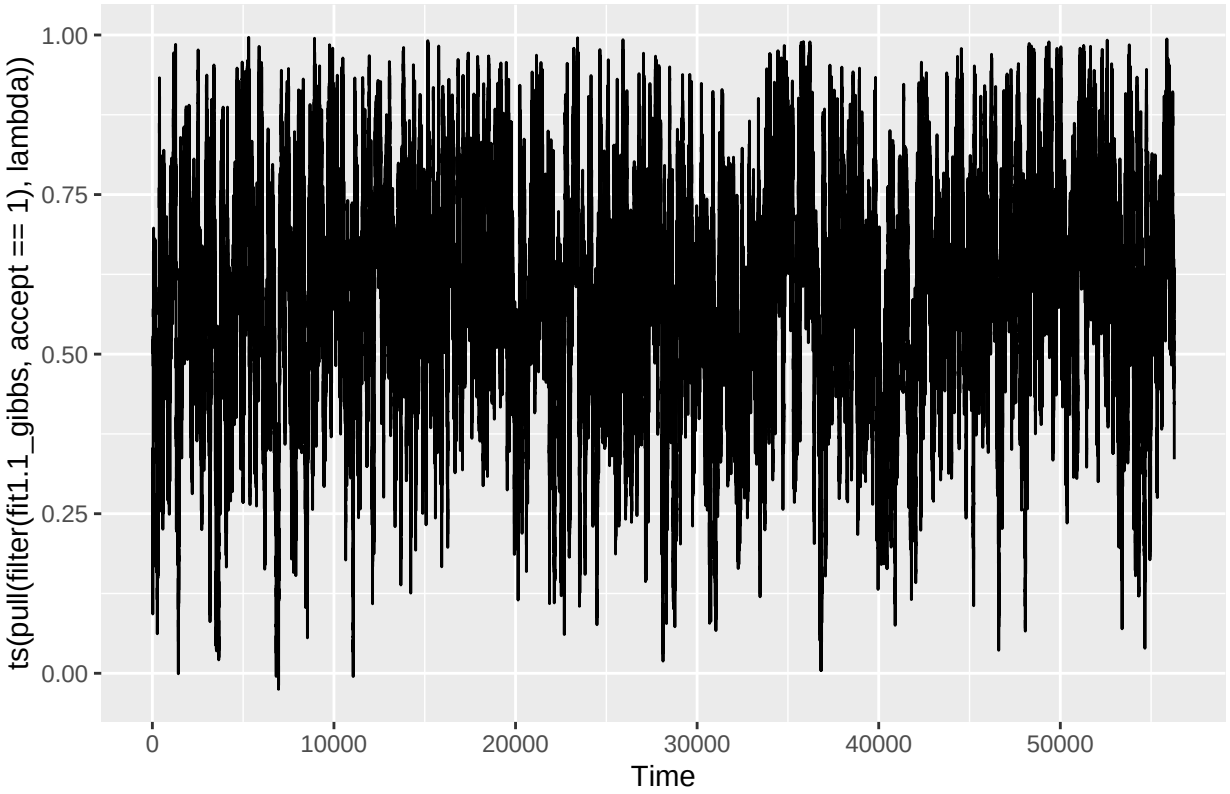
```
fit1.1_gibbs |>
  filter(accept == 1) |>
  pull(beta1) |> ts() |> forecast::autoplot()
```



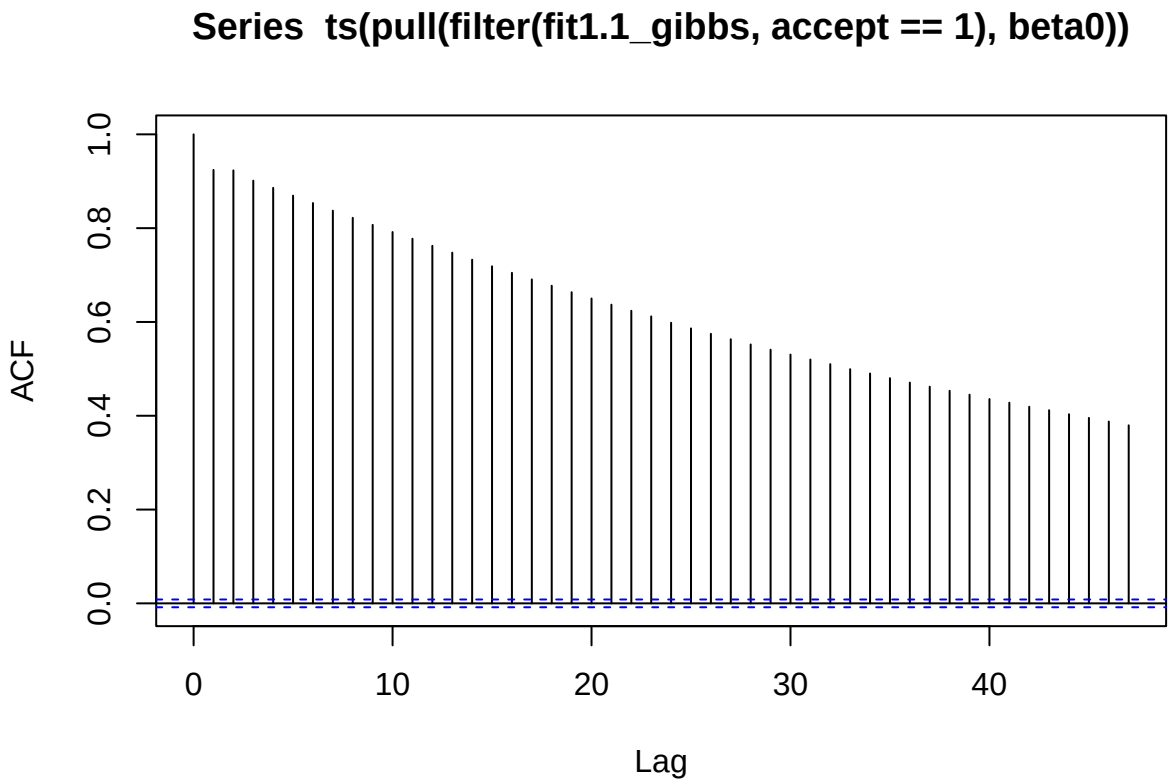

```
fit1.1_gibbs |>
  filter(accept == 1) |>
  pull(sigma2) |> ts() |> forecast::autoplot()
```




```
fit1.1_gibbs |>
  filter(accept == 1) |>
  pull(lambda) |> ts() |> forecast::autoplot()
```

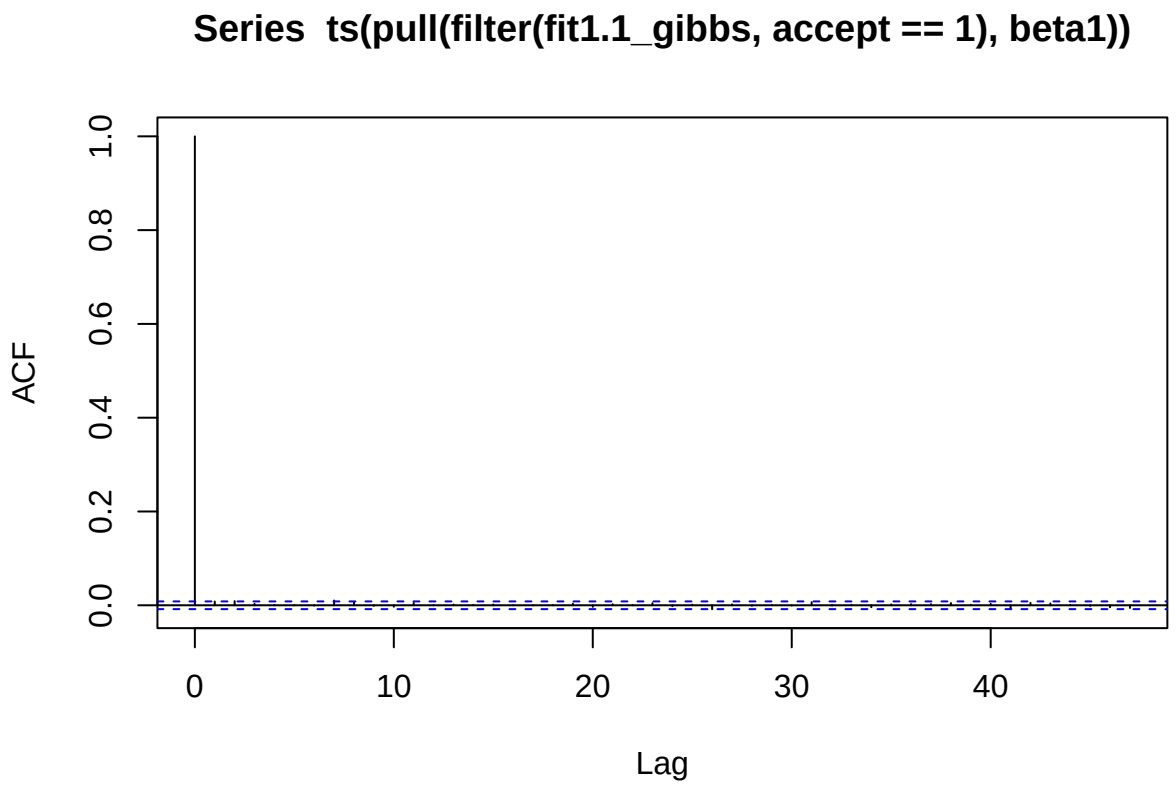



```
fit1.1_gibbs |>
  filter(accept == 1) |>
  pull(beta0) |> ts() |> acf()
```

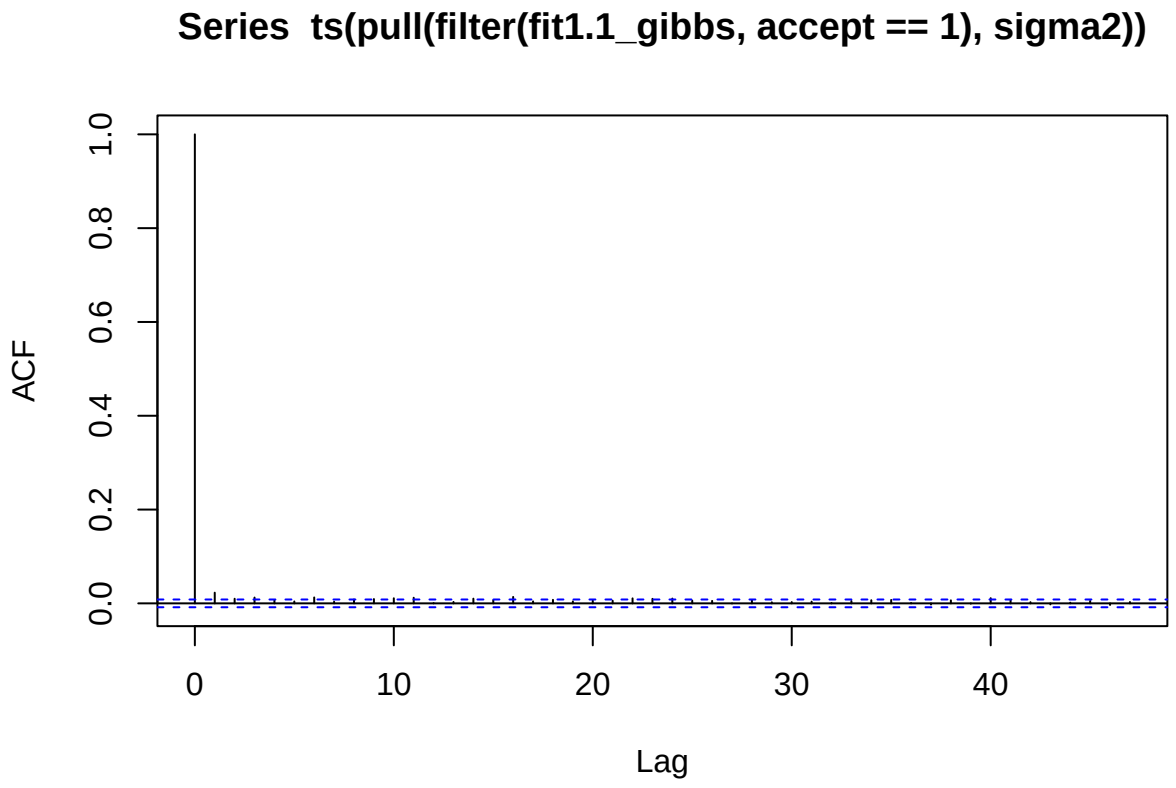


ACF accept

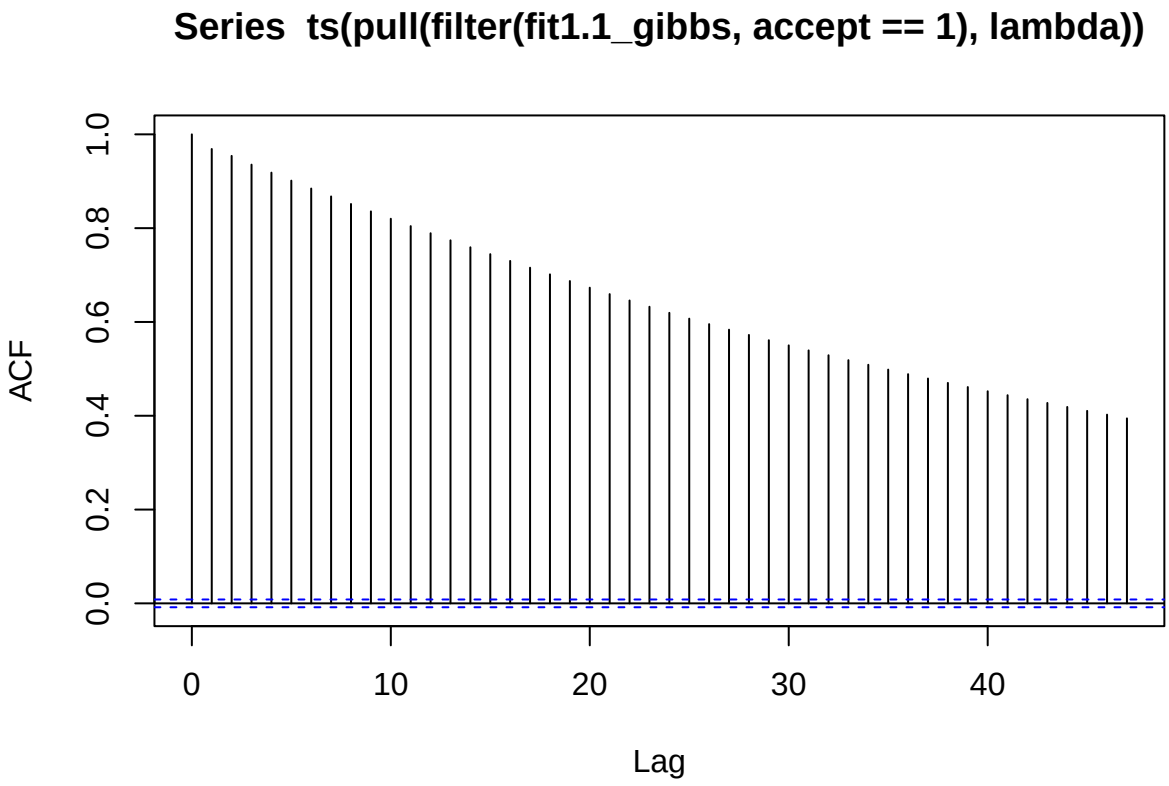
```
fit1.1_gibbs |>
  filter(accept == 1) |>
  pull(beta1) |> ts() |> acf()
```




```
fit1.1_gibbs |>
  filter(accept == 1) |>
  pull(sigma2) |> ts() |> acf()
```




```
fit1.1_gibbs |>
  filter(accept == 1) |>
  pull(lambda) |> ts() |> acf()
```


Series ts(pull(filter(fit1.1_gibbs, accept == 1), lambda))
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
Lag