Bayesian Modelling - Homework 2

Sebastian Kimm-Friedenberg

2022-11-23

Load packages:

```
library(bayesm)
library(data.table)
library(knitr)
library(flextable)
```

```
## Warning: Paket 'flextable' wurde unter R Version 4.2.2 erstellt
```

```
library(tidyverse)
library(plotly)
```

```
## Warning: Paket 'plotly' wurde unter R Version 4.2.2 erstellt
```

```
library(MASS)
set.seed(60323)

library(compiler)
```

Task 1

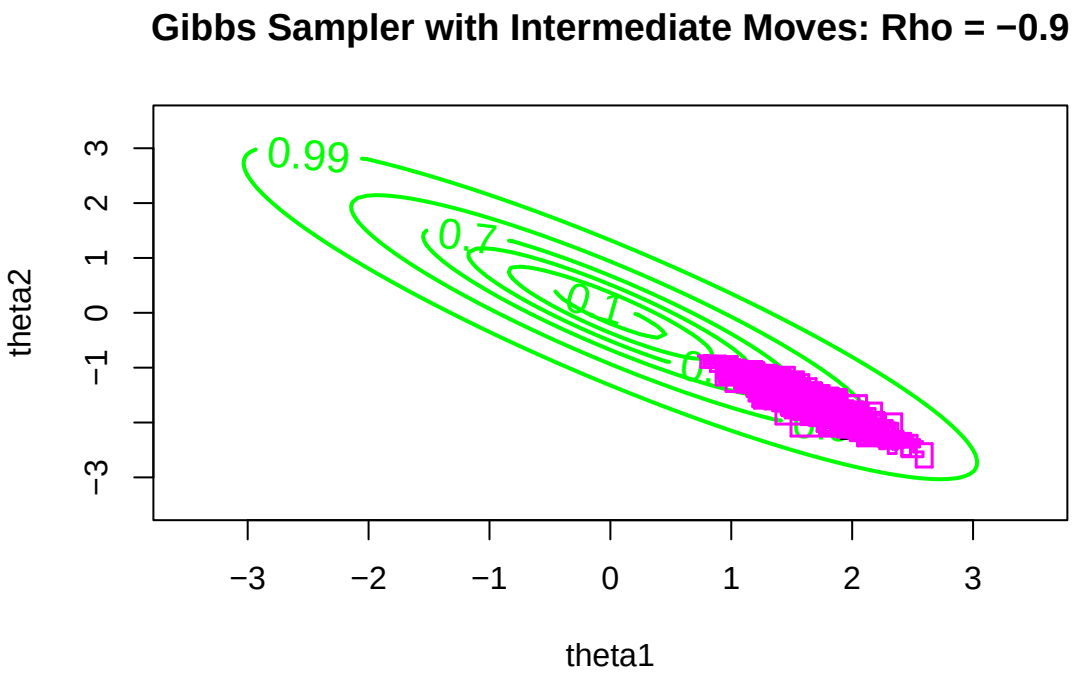
For this task I use the function `rbiNormGibbs` from the `bayesm` package. Instead of drawing from a normal distribution, I draw from a truncated normal distribution. Further, I "commented out" the print statements of the original function in order to let it run in Rmarkdown.

```
## code based on rbiNormGibbs
bi_trun_norm_gibbs <- function(initx = 2, inity = -2, rho, burnin = 100, R = 500, mu=NULL, sigma=NULL,l
    ######## newly added
    ## define mean and sd of normal, if not given by user
    if(is.null(mu)){
      mu = c(0, 0)
    } else if(length(mu)!=2) {
      stop("mu is not a 2x1 vector")
    }

    if(is.null(sigma)){
      sigma = matrix(c(1, rho, rho, 1), ncol = 2)
    } else if(length(sigma)!=4) {
      stop("sigma is not a 2x2 matrix")
```

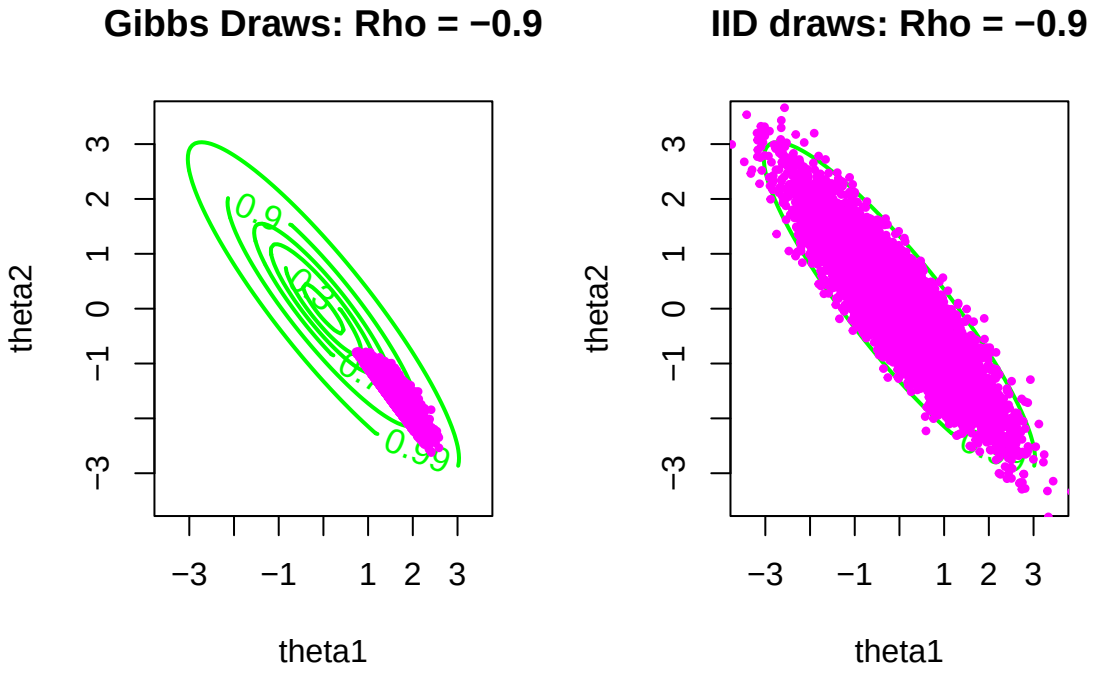
```
}

kernel = function(x, mu, rooti) {
    z = as.vector(t(rooti) %*% (x - mu))
    (z %*% z)
}
if (missing(rho)) {
    pandterm("Requires rho argument ")
}
#cat("Bivariate Normal Gibbs Sampler", fill = TRUE)
#cat("rho= ", rho, fill = TRUE)
#cat("initial x,y coordinates= (", initx, ",", inity, ")",
 #   fill = TRUE)
#cat("burn-in= ", burnin, " R= ", R, fill = TRUE)
#cat(" ", fill = TRUE)
#cat(" ", fill = TRUE)
sd = (1 - rho^2)^(0.5)
rooti = backsolve(chol(sigma), diag(2))

x = seq(-3.5, 3.5, length = 100)
y = x
z = matrix(double(100 * 100), ncol = 100)
for (i in 1:length(x)) {
    for (j in 1:length(y)) {
        z[i, j] = kernel(c(x[i], y[j]), mu, rooti)
    }
}
prob = c(0.1, 0.3, 0.5, 0.7, 0.9, 0.99)
lev = qchisq(prob, 2)
par(mfrow = c(1, 1))
contour(x, y, z, levels = lev, labels = prob, xlab = "theta1",
    ylab = "theta2", drawlabels = TRUE, col = "green", labcex = 1.3,
    lwd = 2)
title(paste("Gibbs Sampler with Intermediate Moves: Rho =",
    rho))
points(initx, inity, pch = "B", cex = 1.5)
oldx = initx
oldy = inity
continue = "y"
r = 0
draws = matrix(double(R * 2), ncol = 2)
draws[1, ] = c(initx, inity)
#cat(" ")
#cat("Starting Gibbs Sampler ....", fill = TRUE)
#cat("(hit enter or y to display moves one-at-a-time)", fill = TRUE)
#cat("('go' to paint all moves without stopping to prompt)",
 #   fill = TRUE)
#cat(" ", fill = TRUE)

while (r<R){ #(continue != "n" && r < R) {
   #if (continue != "go")
    #  continue = readline("cont?")
  ##########################################################################
```

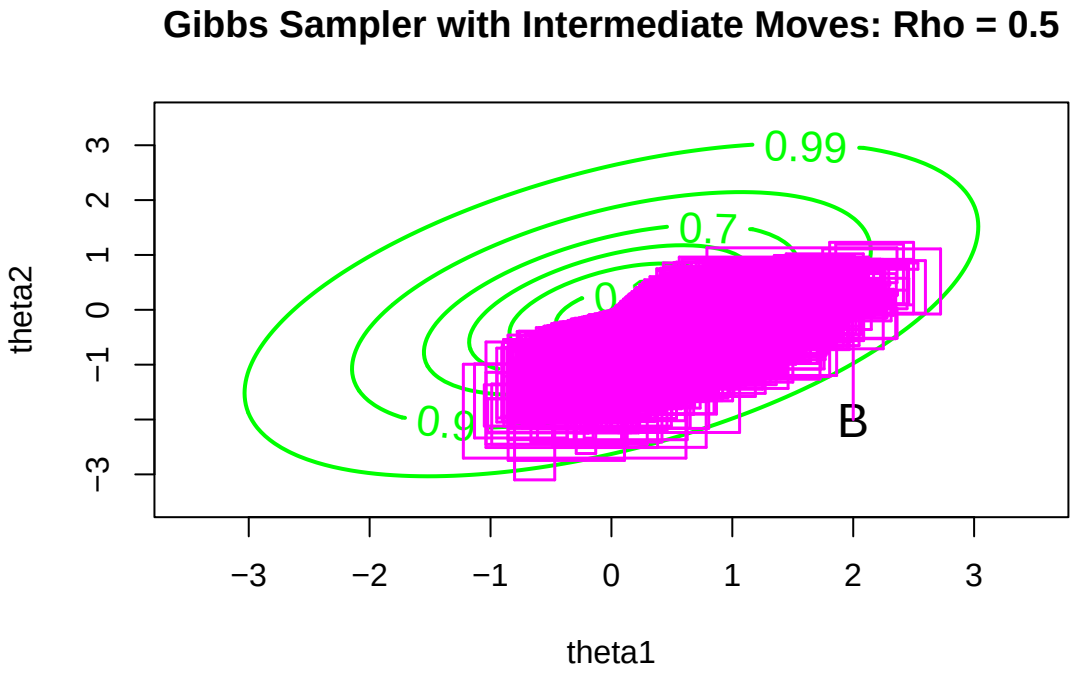
```
      ## only modified lines--> replaced rnorm(1)
        newy = sd * rtrun(mu=mu[1], sigma=sd, a=lower1, b=upper1) + rho * oldx
        lines(c(oldx, oldx), c(oldy, newy), col = "magenta",
            lwd = 1.5)
        newx = sd * rtrun(mu=mu[2], sigma=sd, a=lower2, b=upper2) + rho * newy
        lines(c(oldx, newx), c(newy, newy), col = "magenta",
            lwd = 1.5)
        oldy = newy
        oldx = newx
        r = r + 1
        draws[r, ] = c(newx, newy)

    }
    #continue = readline("Show Comparison to iid Sampler?")
    #if (continue != "n" & continue != "No" & continue != "no") {
        par(mfrow = c(1, 2))
        contour(x, y, z, levels = lev, xlab = "theta1", ylab = "theta2",
            drawlabels = TRUE, labels = prob, labcex = 1.1, col = "green",
            lwd = 2)
        title(paste("Gibbs Draws: Rho =", rho))
        points(draws[(burnin + 1):R, ], pch = 20, col = "magenta",
            cex = 0.7)
        idraws = t(chol(sigma)) %*% matrix(rnorm(2 * (R - burnin)),
            nrow = 2)
        idraws = t(idraws)
        contour(x, y, z, levels = lev, xlab = "theta1", ylab = "theta2",
            drawlabels = TRUE, labels = prob, labcex = 1.1, col = "green",
            lwd = 2)
        title(paste("IID draws: Rho =", rho))
        points(idraws, pch = 20, col = "magenta", cex = 0.7)
    #}
    attributes(draws)$class = c("bayesm.mat", "mcmc")
    attributes(draws)$mcpar = c(1, R, 1)
    return(draws)
}
```

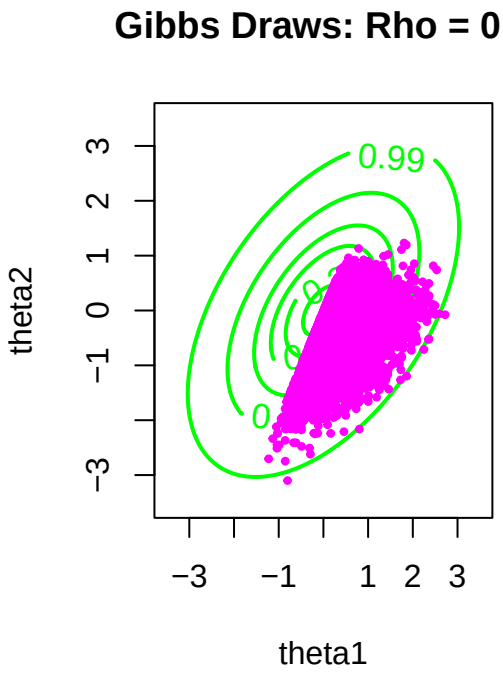
The code that produces the plots below is from the `rbiNormGibbs` function from the `bayesm` package. Every second plot shows draws from an IID sampler, without the truncation. It becomes clear, that a higher degree of correlation among the variables leads to a greater concentration of the probability mass. Also, the truncated draws lose the ergodicity property as opposed to the IID sam-

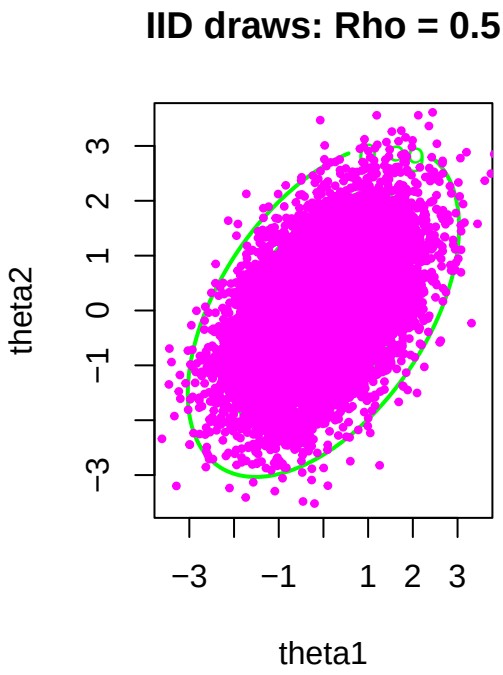

Gibbs Sampler with Intermediate Moves: Rho = −0.9
0.99
0.7
0.1
theta2
theta1
−3
−2
−1
0
1
2
3

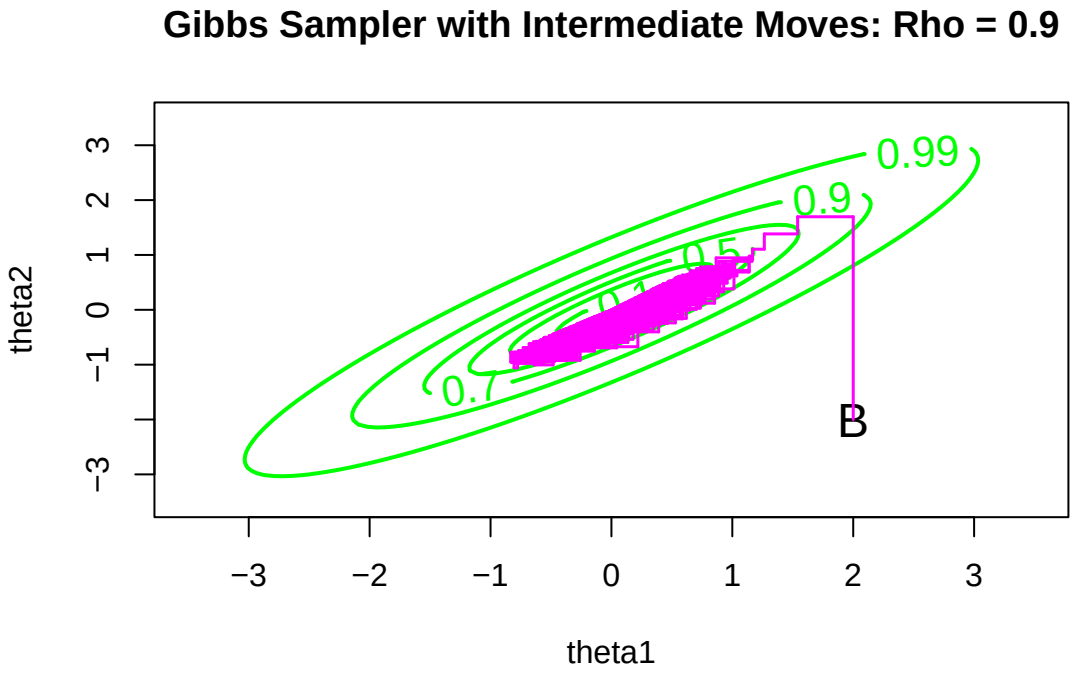
pler.

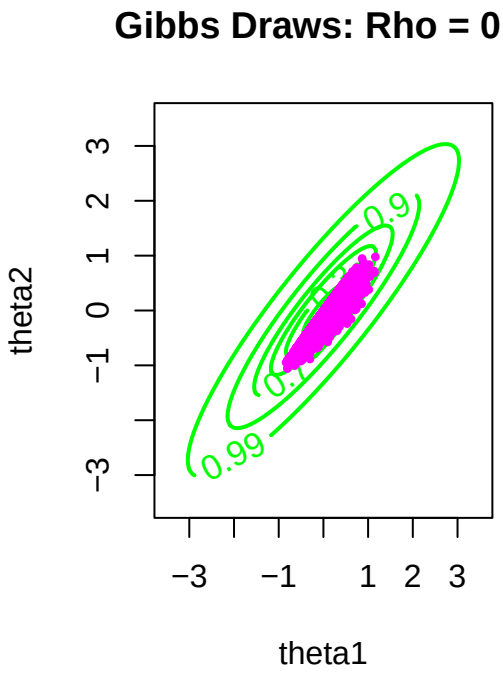

Gibbs Draws: Rho = −0.9
IID draws: Rho = −0.9
theta2
theta1
0.9
0.99
−3
−1
1
2
3

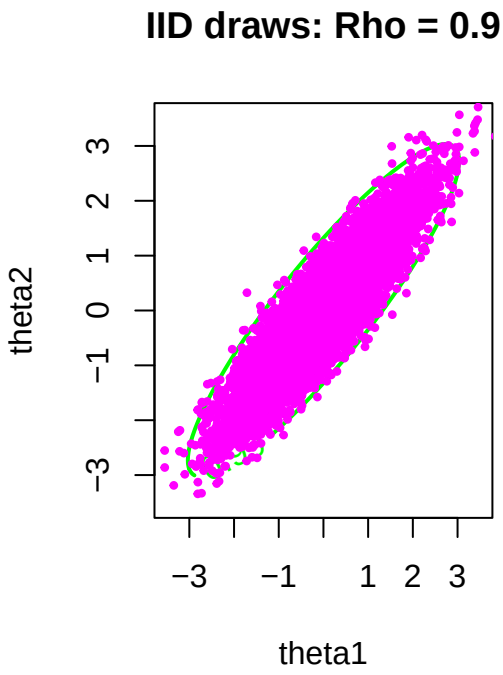

Gibbs Sampler with Intermediate Moves: Rho = 0.5
0.99
0.7
0
0.9
B
theta2
theta1
−3
−2
−1
0
1
2
3


Gibbs Draws: Rho = 0.5
0.99
theta2
theta1
−3
−1
1
2
3


IID draws: Rho = 0.5
theta2
theta1
−3
−1
1
2
3


Gibbs Sampler with Intermediate Moves: Rho = 0.9
0.99
0.9
0.5
0.1
0.7
B
theta2
theta1
−3
−2
−1
0
1
2
3


Gibbs Draws: Rho = 0.9
0.9
0.99
theta2
theta1
−3
−1
1
2
3

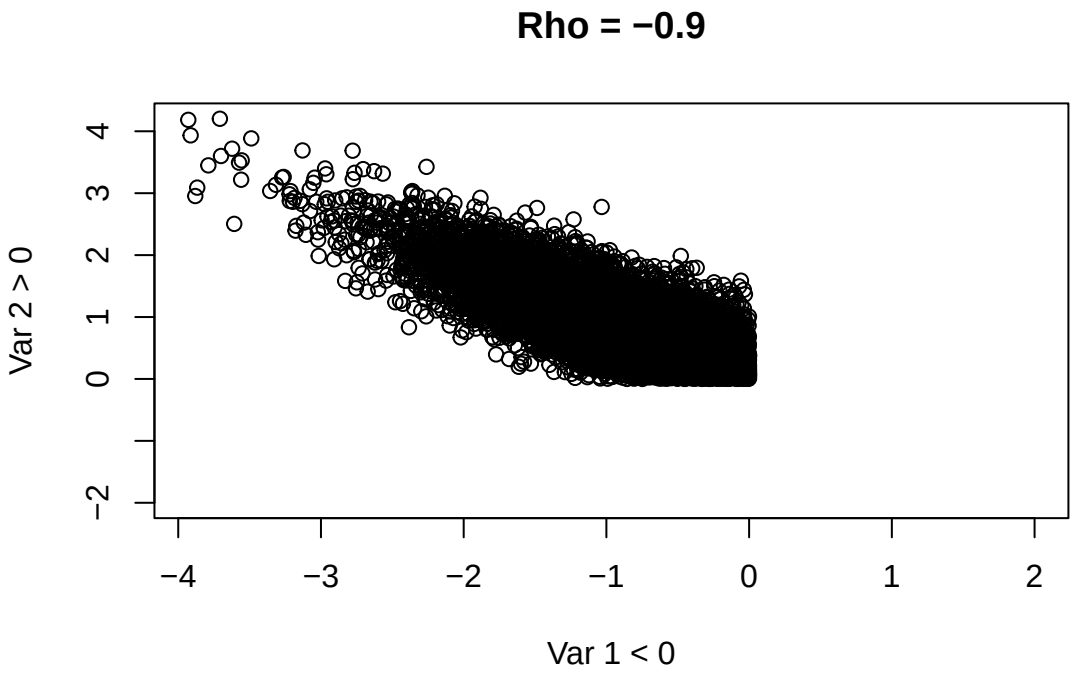

IID draws: Rho = 0.9
theta2
theta1
−3
−1
1
2
3

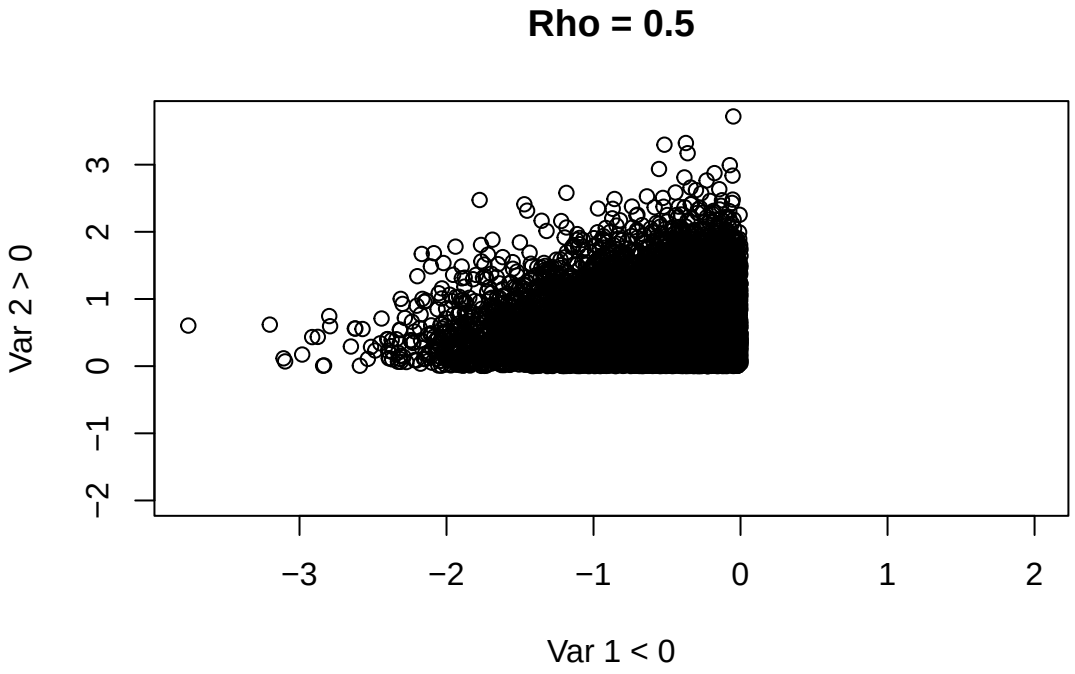
Task 2

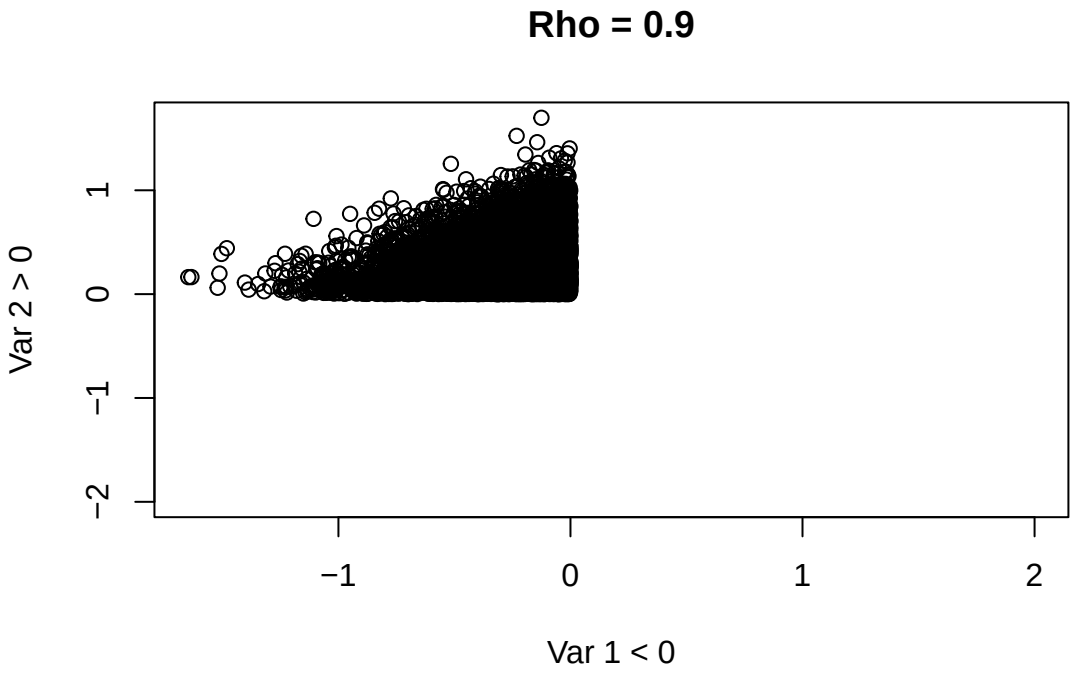
```r
# based on MASS
# draw from mvnorm until draw is in limits, repeat n times
simple_rejection <- function(n, mu=c(0,0), sigma=diag(2), lower1, upper1, lower2, upper2){
  count <- 0
  draw <- matrix(NA,n,2)

  for (i in 1:n){
    in_limits <- FALSE
    while (in_limits==FALSE){
      draw[i,] <- mvrnorm(1, mu, sigma)
      in_limits <- ifelse((draw[i,1] >= lower1 & draw[i,1] <= upper1) & (draw[i,2] >= lower2 & draw[i,2
      count <- count +1
    }
  }

  return(list(draw=draw, count=count))
}

simple_rejection(5, lower1 = -Inf, upper1 = 0, lower2 = 0, upper2 = Inf) #works
```

```r
## define parameters
# mean of normal
mu <- c(0,0)
rho <- c(-0.9, 0.5, 0.9)

# truncation --> one negative, one positive
lower1 <- -Inf
upper1 <- 0
lower2 <- 0
upper2 <- Inf
n <- 10000

## run rejection sampling

res_draw <- list()
res_count <- list(NA, NA, NA)

## allocate memeory
for (i in 1:length(rho)){
  res_draw[[i]] <- matrix(NA, n, 2)
}
# draw
for (i in 1:length(rho)){
  sigma <- matrix(c(1, rho[i], rho[i], 1), ncol = 2)
  tmp <- simple_rejection(n, mu = mu, sigma =  sigma, lower1 = lower1, upper1 = upper1, lower2 = lower2
  res_draw[[i]][,1] <- tmp$draw[,1]
  res_draw[[i]][,2] <- tmp$draw[,2]
  res_count[[i]] <- tmp$count

  # messi plot, but works
  plot(res_draw[[i]][,1], res_draw[[i]][,2],xlim = c(min(res_draw[[i]][,1]), 2), ylim=c(-2,max(res_draw
```

```
        xlab = "Var 1 < 0", ylab = "Var 2 > 0", main = paste("Rho =", as.character(rho[i])))
}
```








```
print(rho)
```

[1] -0.9 0.5 0.9

```
print(res_count)
```

[[1]] [1] 23414

[[2]] [1] 59271

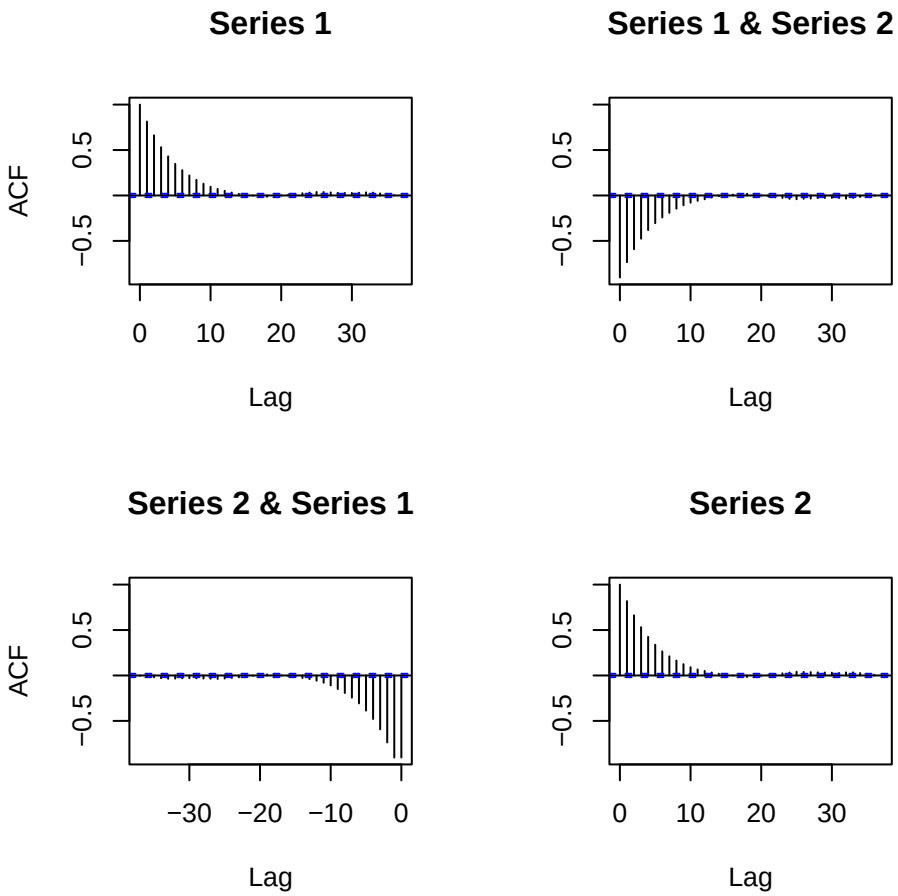
[[3]] [1] 141052

Even though the plots are not the same, we can see a small difference between the two methods. The Gibbs sampler draws not exactly from the truncated distribution but also "off the limits". That leads to a more "scattered" image for the Gibbs sampler, whereas the rejection methods keeps the proposed limits.

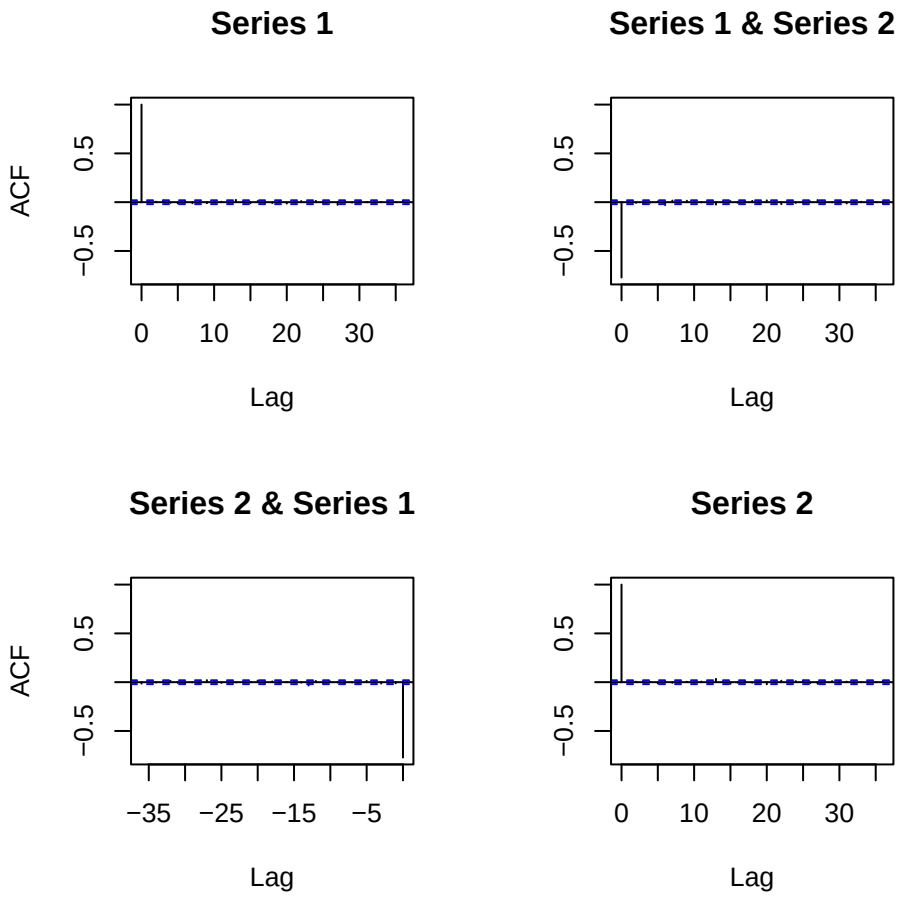
The number of "rejected" draws increases with the value (not the absolute value!) of rho, suggesting a relation among those two statistics.

Task 3

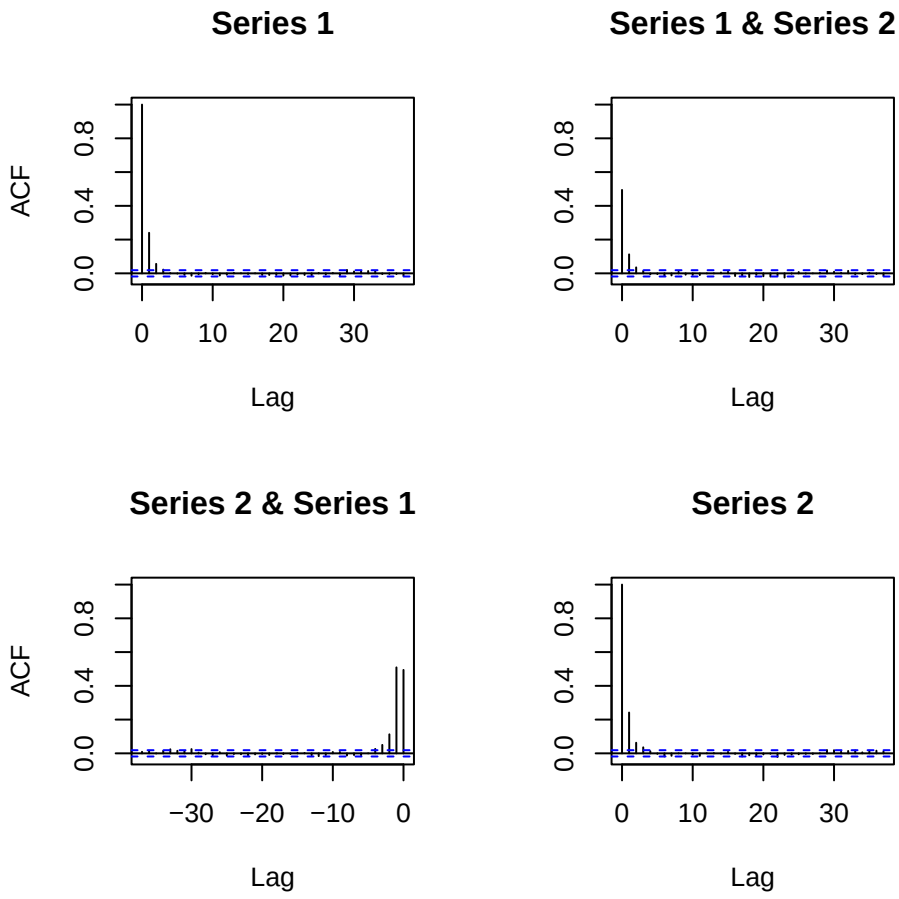
```
for (i in 1:length(rho)){
  print(paste("Gibbs sampler for rho=", as.character(rho[i])))
  acf(draw[[i]])
  print(paste("Rejection sampler for rho=", as.character(rho[i])))
  acf(res_draw[[i]])
}
```



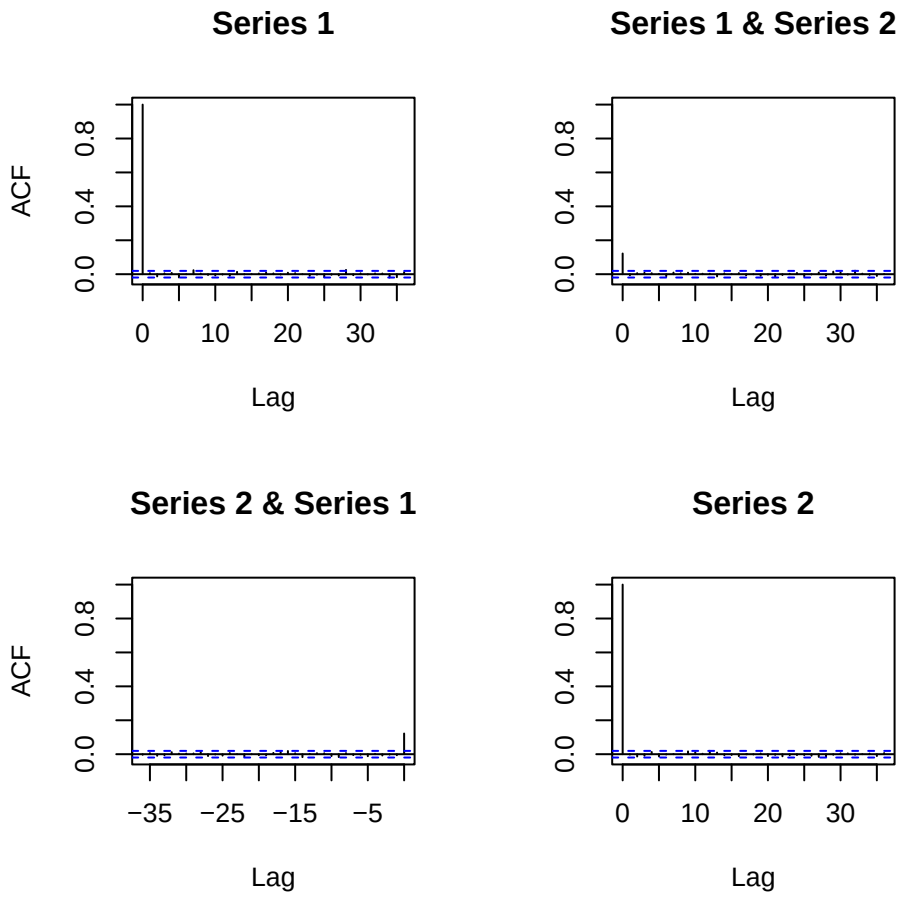
[1] "Gibbs sampler for rho= -0.9"



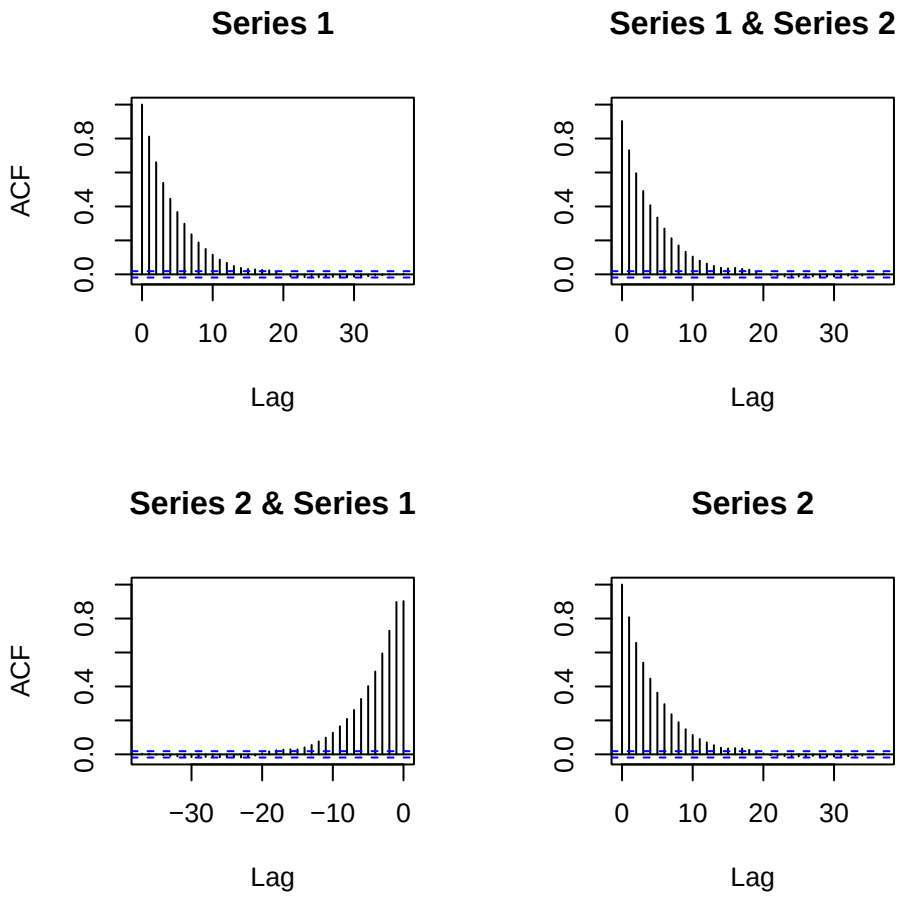
[1] "Rejection sampler for rho= -0.9"



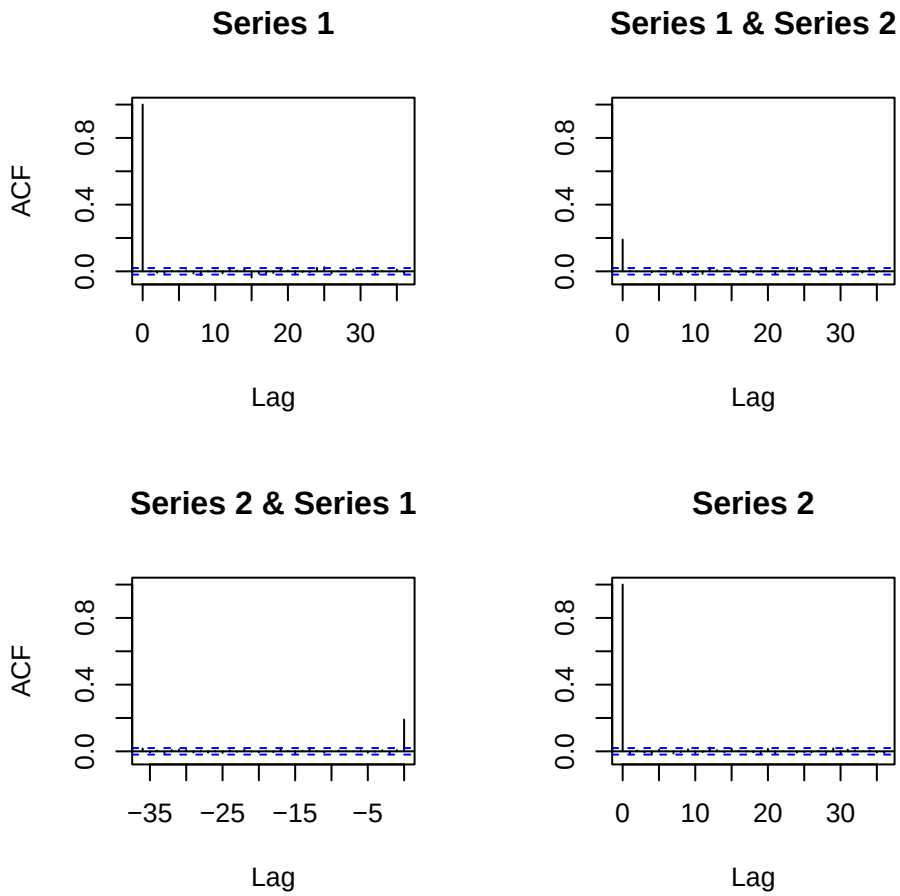
[1] "Gibbs sampler for rho= 0.5"



[1] "Rejection sampler for rho= 0.5"



[1] "Gibbs sampler for rho= 0.9"



[1] "Rejection sampler for rho= 0.9"
As expected we see more autocorrelation in the plots with higher absolute values of rho for the Gibbs sampler. Somehow, the rejection sampler (every second plot) does not reveal any autocorrelation. Maybe, due to the "rejection" of many draws, the variables loose their dependence over time, i.e. the rho is "thrown away".

The result surprises me, since the scatter plots do not reveal much difference. However, the different plotting styles might hide some important information regarding the distributions.