Statistical Programming with R
Assignment 2

Tove Henning
Carl Munkby
Johannes Zetterberg

```r
library(StatProg)
library(tidyverse)

## -- Attaching packages ----------------------------------------------------------------------
tidyverse 1.3.0 --
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## -- Conflicts ------------------------------------------------------------------------------- tidyverse_conf
--
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

#### OLS
olsFun <- function(data){
  ### set column names and add intercept column to X
  Y <- data[,1]
  N <- nrow(data)
  X <- cbind(rep(1, N), data[,2])

  ### calculate the formel and extract the Beta coefficient
  beta_ols=(solve(t(X)%*% X) %*% (t(X) %*% Y))[2,1]

  return(beta_ols)
}

#Här är en kommentar om OLS. blabla bla blabakljdgklajkl

testData <- cbind( c(0.62, 0.18, 3.92, 0.80, -5.15),
c(0.44, 1.49, 0.69, 0.13, 1.90) )

olsFun(data = testData)

## [1] -3.092464
```

```r
##### weighted least squares
wlsFun <- function(data, lambda){
  if (is.numeric(lambda)==FALSE){print("lambda is not numeric") }
  else{
    ### create variable for the number of observations in the dataset
    N <- nrow(data)

    ### set column names and add intercept column for X
    Y <- data[,1]
    X <- cbind(rep(1,N), data[,2])

    ### create a zero matrix N x N
    Z <- matrix(0, N, N)

    ## make a forloop to put in the error terms on the diagonal
    ## to create the error covariance matrix
    er <- NULL
    for (i in 1:N) {
```

```r
      er[i] <- exp(X[i,2]*lambda)
      Z[i,i] <- er[i]
    }
    ### calculate and extract the Beta coefficient
    beta_wls = ((solve(t(X)%*%(solve(Z))%*%X)%*%t(X)%*%
                 (solve(Z))%*%Y))[2,1]
  return(beta_wls)
  }
}

wlsFun(data = testData, lambda = 2)
```

```
## [1] 0.007392226
```

```r
#### FWLS
# H<U+653C><U+3E34>r g<U+663C><U+3E36>r jag lite kommentarer
# Bla bla bla
fwlsFun <- function(data, trueVar){
  y = data[,1]
  N <- nrow(data)
  X = cbind(rep(1,N), data[,2:ncol(data)])

  mod = lm(y ~ -1 +X)
  res = mod$residuals
  res2 = res^2

  ln_res2 = lm(log(res2) ~  -1 +X)
  lambda_hat = ln_res2$coefficients[2]

  if(trueVar == TRUE){
    error_cov = matrix(0, N, N)
    for(i in 1:N){
      error_cov[i,i] <- exp(X[i,2]*lambda_hat)
    }

  }
  else if(trueVar == FALSE)
  {
    error_cov = matrix(0, N, N)
    for(i in 1:N){
      error_cov[i,i] <- 1 + X[i,2]*lambda_hat
    }

  }
beta_fwls = (solve(t(X)%*%solve(error_cov)%*%X)%*%t(X)%*%solve(error_cov)%*%y)[2,1]

return(beta_fwls)
}
fwlsFun(data = testData, trueVar = TRUE)
```

```
## [1] -2.615266
```

```r
fwlsFun(data = testData, trueVar = FALSE)
```

```
## [1] -2.750474
```

First we created the OLS fun to...

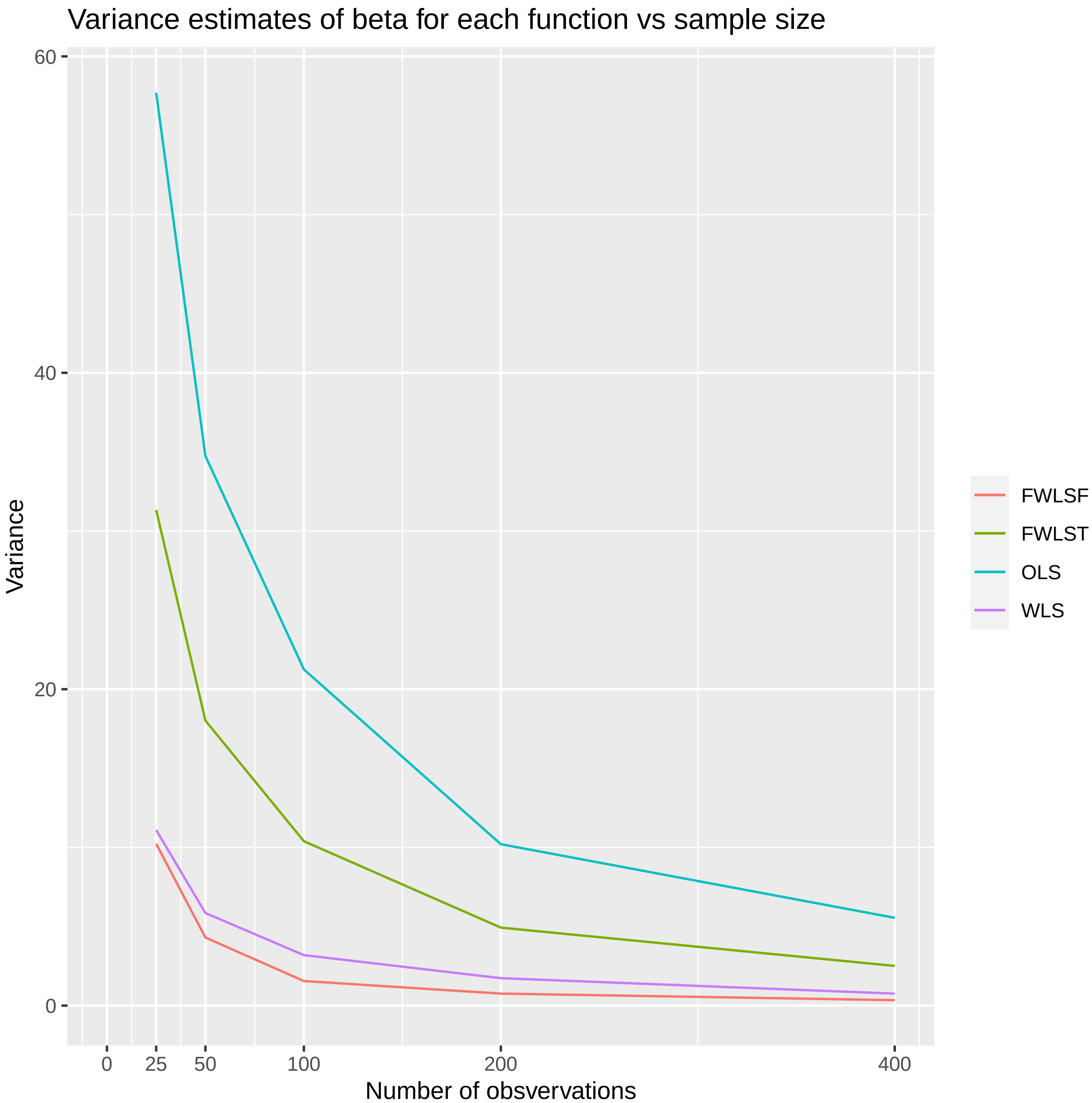
```
#### Data simulation
DataFun <- function(n, lambda) {

    # independent variable
    x <- runif(n, min = 0, max = 2)

    # standard deviation in epsilons normal distribution
    s <- NULL
    for (i in seq_len(n)) {
        s[i] <- exp(x[i]*lambda)
    }

    # error term
    epsilon <- rnorm(n, mean = rep(0, n), sd = s)

    beta <- 2

    # dependent variable
    y <- NULL
    for (i in seq_len(n)) {
        y[i] <- beta*x[i] + epsilon[i]
    }

    # container matrix
    mat <- matrix(data = 0, ncol = 2, nrow = n)

    # creating matrix of indepedent and dependent variables
    for (i in seq_len(n)) {
        mat[i, 1] <- y[i]
        mat[i, 2] <- x[i]
    }
    return(mat)
}
```

Data generation...

```
SimFun <- function(n, sim_reps, seed, lambda) {
    set.seed(seed)
    R <- sim_reps

    # saving betas
    mat <- matrix(0, nrow = R, ncol = 4)

    for (i in seq_len(R)) {
    # data sim
    dat <- DataFun(n, lambda)
    # estimate sim
    mat[i,1] <- olsFun(dat)
    mat[i,2] <- wlsFun(dat, lambda)
    mat[i,4]<- fwlsFun(dat, trueVar = TRUE)
    mat[i,3]<- fwlsFun(dat, trueVar = FALSE)
    }
```

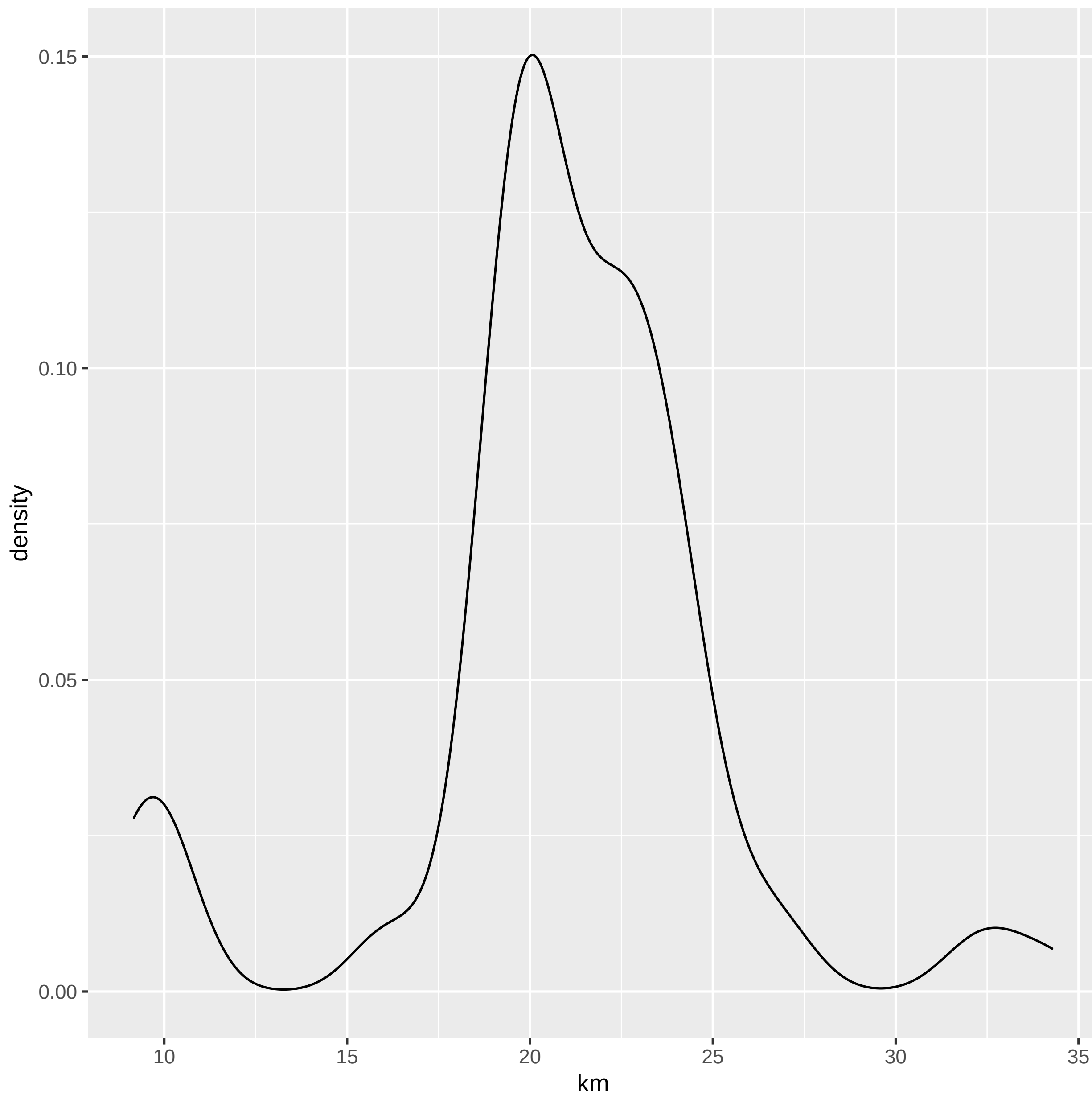
```r
    betas <- apply(mat, 2, var)

    return(betas)
}
```

```r
##### 4. Plot variance estimates
x <- c(25, 50, 100, 200, 400)

var_obs <- matrix(0, ncol = 4, nrow = 5)

for (i in seq_along(x)) {
    var_obs[i,] <- (SimFun(x[i], 100, 2020, 2))
}

var_obs = as.data.frame(var_obs)
rownames(var_obs) = c(25, 50, 100, 200, 400)


colnames(var_obs) = c("OLS","WLS","FWLST","FWLSF")
rownames(var_obs) = c("25","50","100","200","400")

ggplot(as.data.frame(var_obs),aes(x=as.numeric(rownames(var_obs)))) +
  geom_line(aes(y = OLS, colour = "OLS")) +
  geom_line(aes(y = WLS, colour = "WLS")) +
  geom_line(aes(y = FWLST, colour = "FWLST")) +
  geom_line(aes(y = FWLSF, colour = "FWLSF")) +
  labs(x="Number of obsvervations",y="Variance",
       title="Variance estimates of beta for each function vs sample size") +
  theme(legend.title = element_blank()) +
  scale_x_continuous(breaks=c(0,25,50,100,200,400),limits=c(0, 400))
```




```
#############
```

```
##################################################
# start of part2
##################################################
galaxies <- as.data.frame(galaxies)
names(galaxies) <- "km"

ggplot(galaxies, aes(x = km)) +
  geom_density()
```

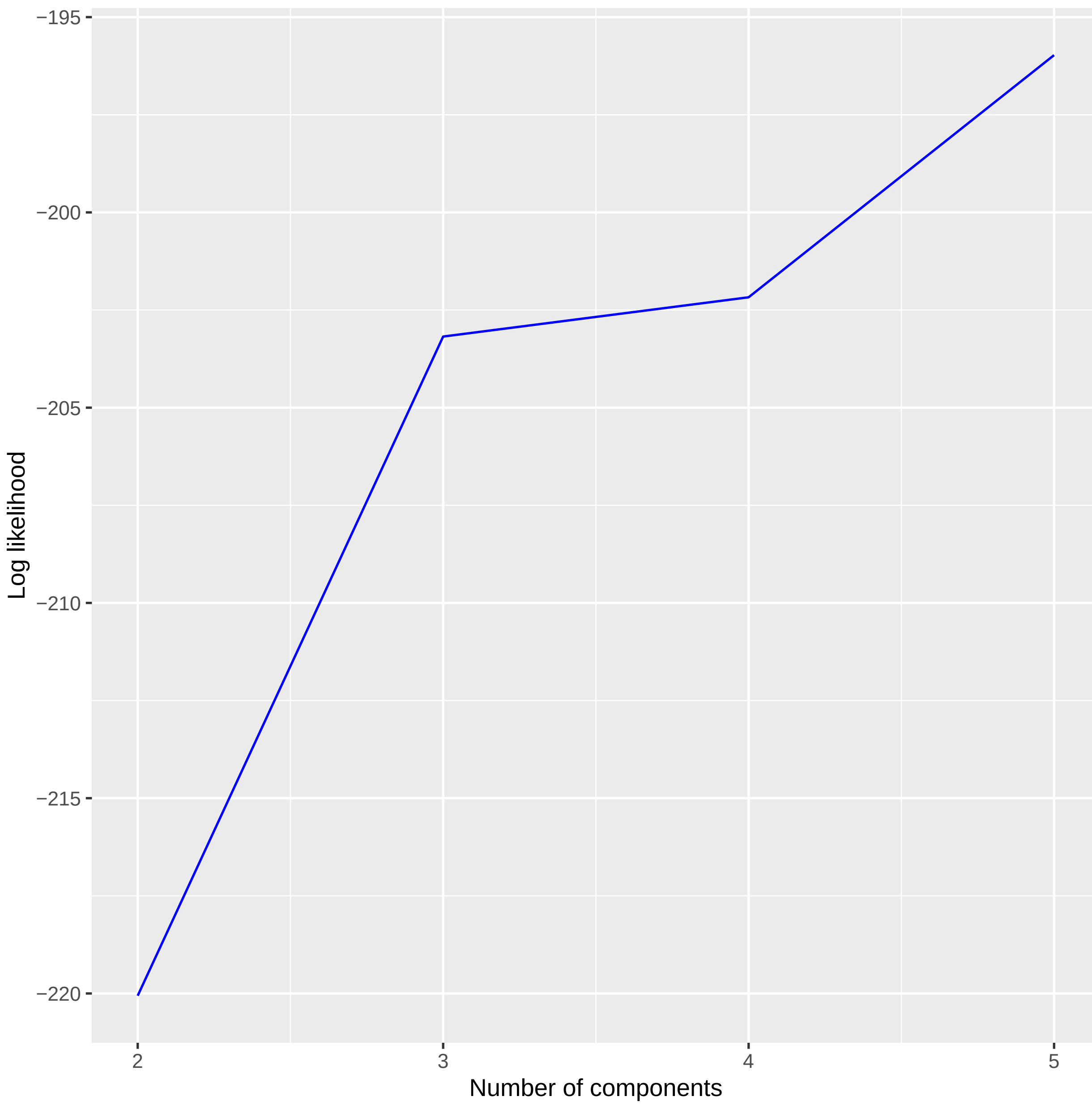



```
galaxies = galaxies$km

gammaUpdate = function(x, mu, sigma, pi){
  pdf = t(sapply(x, dnorm, mean = mu, sd = sigma))
  for(n in 1:length(x)){
    for(k in 1:length(mu)){
      pdf[n, k] = pi[k]*pdf[n, k]
    }
  }
  gamma = as.data.frame(matrix(NA, ncol = length(pi), nrow = length(x)))
  for(n in 1:length(x)){
    for(k in 1:length(mu)){
      gamma[n, k] = pdf[n, k]/sum(pdf[n,])
    }
```

```
  }
  return(as.data.frame(gamma))
}

# mu
muUpdate = function(x, gamma){
  K <- ncol(gamma)
  mu <- NULL
  for (i in seq_len(K)) {
    mu[i] <- sum(gamma[,i]*x)/sum(gamma[,i])
  }
  return(mu)
}

### Sigma
sigmaUpdate = function(x, gamma, mu){
  N = matrix(0, ncol= ncol(gamma))
  sigma = matrix(0, ncol = ncol(gamma))
  for(k in 1:ncol(gamma)){
    for(n in 1:length(x)){
      sigma[k] = sigma[k] + gamma[n,k]*(x[n]-mu[k])^2
      N[k] = N[k] + gamma[n,k]
    }
    sigma[k] = sqrt(sigma[k]/N[k])
  }
  return(sigma)
}

piUpdate = function(gamma){
  pi <- NULL
  for (i in 1:ncol(gamma)) {
    pi[i] <- sum(gamma[,i])/sum(gamma)
  }
  return(pi)
}

### Log-likelihood
loglik = function(x, pi, mu, sigma){
  sum_pdf = matrix(0, nrow = length(x))
  loglike = 0
  for(n in 1:length(x)){

    for(k in 1:length(pi)){
      sum_pdf[n] = sum_pdf[n] + (pi[k] * dnorm(x[n], mu[k], sigma[k]))
    }
    loglike = loglike + log(sum_pdf[n])
  }
  return(loglike)
}

initialValues = function(x, K, reps = 100){
  mu = rnorm(K, mean(x), 5)
  sigma = sqrt(rgamma(K, 5))
```

```r
  p = runif(K)
  p = p/sum(p)
  currentLogLik = loglik(x, p, mu, sigma)
  for(i in 1:reps){
    mu_temp = rnorm(K, mean(x), 10)
    sigma_temp = sqrt(rgamma(10, 5))
    p_temp = runif(K)
    p_temp = p_temp/sum(p_temp)
    tempLogLik = loglik(x, p_temp, mu_temp, sigma_temp)
    if(tempLogLik > currentLogLik){
      mu = mu_temp
      sigma = sigma_temp
      p = p_temp
      currentLogLik = tempLogLik
    }
  }
  return(list("mu" = mu, "sigma" = sigma, "p" = p))
}

##################################################
# EM algo
##################################################
EM = function(x, K, tol = 0.001){
  inits = initialValues(x, K, 1000)
  mu = inits$mu
  sigma = inits$sigma
  prob = inits$p

  prevLoglik <- 0
  loglikDiff<- 1

  # while loop
  while(loglikDiff > tol){

    gamma <- gammaUpdate(x, mu, sigma, prob)
    mu <- muUpdate(x, gamma)
    sigma <- sigmaUpdate(x, gamma, mu)
    prob <- piUpdate(gamma)

    currentLogLik <- loglik(x, prob, mu, sigma)

    loglikDiff <- abs(prevLoglik - currentLogLik)

    prevLoglik <- currentLogLik

  }

  return(list('loglik' = currentLogLik, 'mu' = mu, 'sigma' = sigma, 'prob' = prob))
}

set.seed(1996)
final_plot = matrix(0, ncol = 4, nrow= length(galaxies))
loglik_values = NULL
```

```r
for(k in 2:5){
  z = EM(galaxies, k)
  loglik_values[(k-1)] = z$loglik
  for(i in 1:k){
    final_plot[,(k-1)] = final_plot[,(k-1)] + z$prob[i] * dnorm(galaxies, z$mu[i], z$sigma[i])
  }
}

loglik_values <- as.data.frame(loglik_values) %>%
  mutate("K" = (2:5))

loglik_plot = ggplot(data = loglik_values) +
  geom_line(aes(x=K, y=loglik_values), color = "blue") +
  labs(x="Number of components",y= "Log likelihood")
loglik_plot
```




```
final_plot = as.data.frame(final_plot)
colnames(final_plot) = c("K = 2", "K = 3", "K = 4", "K = 5")
final_plot = cbind(final_plot, galaxies)

ggplot(data = final_plot, aes(x = galaxies)) +
  geom_line(aes(y = `K = 2`, color = "K = 2"), size = 1) +
  geom_line(aes(y = `K = 3`, color = "K = 3"), size = 1) +
  geom_line(aes(y = `K = 4`, color = "K = 4"), size = 1) +
  geom_line(aes(y = `K = 5`, color = "K = 5"), size = 1) +
  geom_density(aes(fill = "Density plot"), color = "pink", alpha = 0.2, size = 0) +
  labs(x = "km", y = "Density",title="Plot of different values of K and the density plot of galaxies") +
  theme(legend.title = element_blank(),legend.position = c(.95, .95),
        legend.justification = c("right", "top"),
        legend.box.just = "right",
```

```
    legend.margin = margin(6, 6, 6, 6))
```