Moffat1D

2024-03-21

library(ggplot2)  
library(plotly)

##   
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':  
##   
## last\_plot

## The following object is masked from 'package:stats':  
##   
## filter

## The following object is masked from 'package:graphics':  
##   
## layout

library(readr)  
library(ggpointdensity)  
library(viridis)

## Loading required package: viridisLite

library(tidyr)  
library(fitdistrplus)

## Loading required package: MASS

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:plotly':  
##   
## select

## Loading required package: survival

## Color pallete  
cbPalette <- c("#56B4E9", "#E69F00", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

Importing the data.

library(here)

## here() starts at C:/Users/Blake/Dropbox/My PC (LAPTOP-R53ILDBG)/Documents/GitHub/Terzan-5

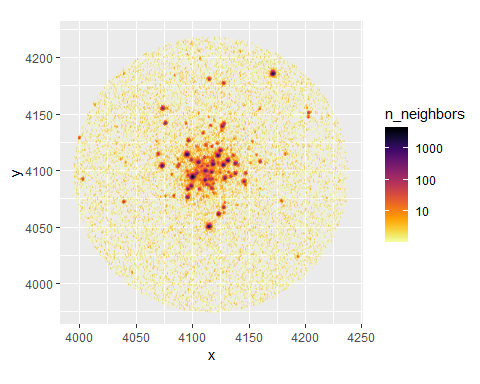
i\_am("MoffatPDF.Rmd")

## here() starts at C:/Users/Blake/Dropbox/My PC (LAPTOP-R53ILDBG)/Documents/GitHub/Terzan-5

# laptop data from csv  
Terzan5 <- readr::read\_csv(here("Data", "Terzan 5 X-ray events.csv"), col\_types = list(.default = readr::col\_guess()), )  
# head(Terzan5)  
  
#fishbowl data from ODS  
# library(readODS)  
# Terzan5 <- read\_ods(here("Raw Data.ods"), col\_types = list(.default = readr::col\_guess()), )  
#removing extraneous columns  
T5 <- data.frame(Terzan5$x, Terzan5$y)  
colnames(T5) <- c('x','y')

Viewing the dataset via a density heatmap.

ggplot(T5,aes(x,y)) + geom\_pointdensity(adjust = 0.05, size = 0.1, shape = "1") + scale\_color\_viridis(direction = -1, option = "B", trans = "log", breaks = c(10, 100, 1000, 10000)) + coord\_fixed()

 Darker areas indicate areas containing a high number of observations. These indicates the existance of a celestial source. The first aim is to try to fit a Moffat Model to each individual celestial source to predict. By then removing all of the celestial sources and their associated observations, if there is a higher level remaining this is an indication of interstellar gas remaining within the cluster.

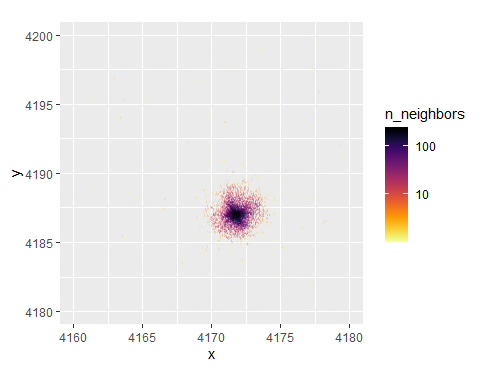
Before getting to that stage, I want to focus on two individual celestial sources that are isolated (unlikely to contain observations that may come from another nearby source).

Choosing two Points of Interest (POI)

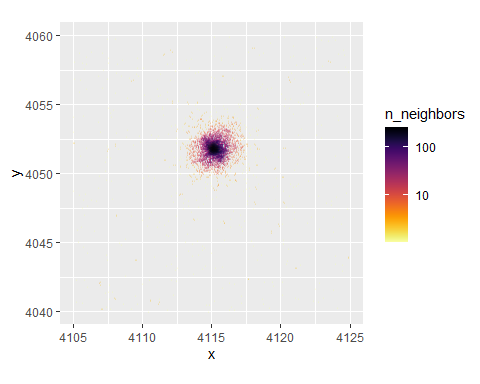
POI1 <- T5 %>%  
 filter(x>4160 & x<4180 & y>4180 & y<4200)  
POI2 <- T5 %>%  
 filter(x>4105 & x< 4125 & y>4040 & y<4060)

Viewing one of those Points of Interest

ggplot(POI1, aes(x,y)) +  
 geom\_pointdensity(adjust = 0.05, size = 0.1, shape = "1") +   
 scale\_color\_viridis(direction = -1, option = "B", trans = "log", breaks = c(10, 100, 1000)) +   
 coord\_fixed()



ggplot(POI2, aes(x,y)) +  
 geom\_pointdensity(adjust = 0.05, size = 0.1, shape = "1") +   
 scale\_color\_viridis(direction = -1, option = "B", trans = "log", breaks = c(10, 100, 1000)) +   
 coord\_fixed()

 It’s important to note that while a lot of the surrounding area appears to be empty, it is full of background observations. It appears blank as observations are darker in colour and more prominent when there is a higher number of other observations close by.

Before fitting a 2D Moffat Model, I want to simplify and fit a 1D model to see if it is the most appropriate fit to the data. To do that, I’m going to slice my celestial sources (POI’s) into lines, with fewer observations and treat them as being one-dimensional.

# line0 <- POI2 %>%  
# filter(y>4048.5 & y< 4049.5)  
# line1 <- POI2 %>%  
# filter(y>4049.5 & y< 4050.5)  
line2 <- POI2 %>%  
 filter(y>4050.5 & y< 4051.5)  
# line3 <- POI2 %>%  
# filter(y>4051.5 & y< 4052.5)  
# line4 <- POI2 %>%  
# filter(y>4052.5 & y< 4053.5)  
# line5 <- POI2 %>%  
# filter(y>4053.5 & y< 4054.5)  
# line6 <- POI2 %>%  
# filter(y>4054.5 & y< 4055.5)  
# lineback <- POI2 %>%  
# filter(y>4055 & y< 4060)  
  
line2.5 <- POI2 %>%  
 filter(y>4050.5 & y< 4051)

Choosing one of my slices, I am to fit a distribution to it to be able to predict if the observations belong to the celestial source or not. Fitting to a Normal Distribution

xl2 <- line2$x  
  
# Define the Gaussian density function  
gaussian\_density <- function(parameters, x) {  
 mean <- parameters[[1]]  
 sd <- parameters[[2]]  
   
 return(dnorm(x, mean = mean, sd = sd))  
}  
  
# Define the negative log-likelihood function for Gaussian distribution  
negative\_log\_likelihood <- function(parameters, x) {  
 predicted\_density <- gaussian\_density(parameters, x)  
   
 return(-sum(log(predicted\_density)))  
}  
  
initial\_guess <- c(mean = 4115, sd = 1)  
# Optimize the negative log-likelihood function  
result <- optim(par = initial\_guess, fn = negative\_log\_likelihood, x = xl2, method = "Nelder-Mead")

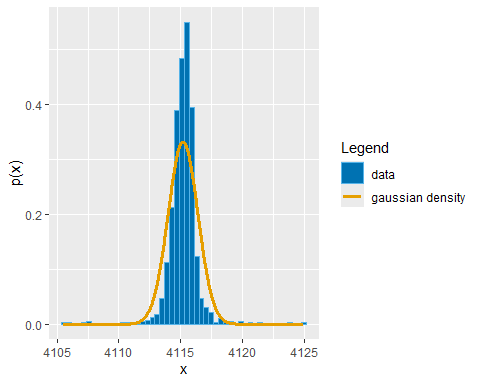
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced  
  
## Warning in dnorm(x, mean = mean, sd = sd): NaNs produced

# Extract the optimized parameters  
optimized\_parameters <- result$par  
optimized\_parameters

## mean sd   
## 4115.19535 1.20138

optimParameters2N <- gaussian\_density(optimized\_parameters, xl2)  
## plotting layers in ggplot  
ggplot(line2) +  
 geom\_histogram(aes(x, y = after\_stat(density), colour = "data"), bins = 50, fill=cbPalette[5]) +  
 stat\_function(fun = gaussian\_density, args = list(parameters = optimized\_parameters), n=1000, aes(colour="gaussian density"), linewidth=1.2) +  
 ylab("p(x)") +   
 scale\_colour\_manual("Legend", values = cbPalette)

## Warning in stat\_function(fun = gaussian\_density, args = list(parameters = optimized\_parameters), : All aesthetics have length 1, but the data has 963 rows.  
## ℹ Did you mean to use `annotate()`?

 It is clear that the Normal distribution doesn’t provide an accurate model of the data.

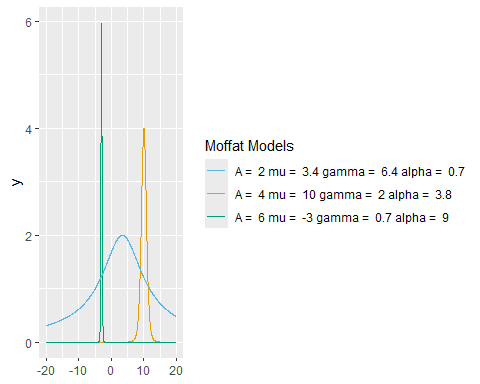
I want to have a look at the 1D Moffat distribution. The Moffat Model was specifically designed to handle point spread functions that occur in the field of astronomy. So I expect to to yield a better fit to data than a normal distribution.

The 1D Moffat Model:

Where A = amplitude, = position of the maximum, = core width, = power index.

Visualising the Moffat Function

Moffat <- function(x, amplitude, mu, gamma, alpha){  
 amplitude\*(1+((x-mu)^2/gamma^2))^(-alpha)  
}  
  
#Function 1  
A1 <- 4  
mu1 <- 10  
gamma1 <- 2  
alpha1 <- 3.8  
String1 = paste("A = ", A1, "mu = ", mu1, "gamma = ", gamma1, "alpha = ", alpha1)  
  
#Function 2  
A2 <- 6  
mu2 <- -3  
gamma2 <- 0.7  
alpha2 <- 9  
String2 = paste("A = ", A2, "mu = ", mu2, "gamma = ", gamma2, "alpha = ", alpha2)  
  
#Function 3  
A3 <- 2  
mu3 <- 3.4  
gamma3 <- 6.4  
alpha3 <- 0.7  
String3 = paste("A = ", A3, "mu = ", mu3, "gamma = ", gamma3, "alpha = ", alpha3)  
  
  
ggplot() +  
 stat\_function(fun = Moffat, n = 1000, args = list(amplitude = A1, mu = mu1, gamma = gamma1, alpha = alpha1), aes(colour = String1)) +  
 stat\_function(fun = Moffat, n = 1000, args = list(amplitude = A2, mu = mu2, gamma = gamma2, alpha = alpha2), aes(colour = String2)) +  
 stat\_function(fun = Moffat, n = 1000, args = list(amplitude = A3, mu = mu3, gamma = gamma3, alpha = alpha3), aes(colour = String3)) +  
 xlim(-20, 20) + ylim(0, NA) +  
 scale\_colour\_manual("Moffat Models", values = cbPalette)

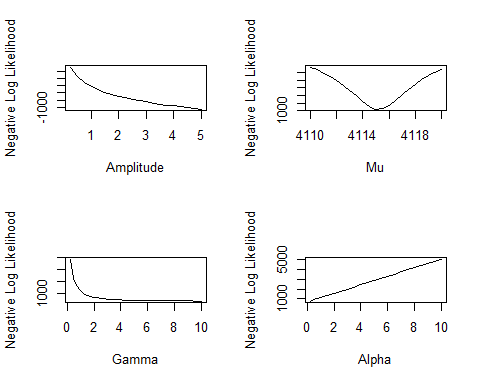
 Trying to use the optim function to estimate the parameters of the Moffat Model when fitted to the data.

Moffat <- function(parameters, x){  
 amplitude <- parameters[[1]]  
 mu <- parameters[[2]]  
 gamma <- parameters[[3]]  
 alpha <- parameters[[4]]  
  
 amplitude <- pmax(amplitude, 1e-6)  
 gamma <- pmax(gamma, 1e-6)  
 alpha <- pmax(alpha, 1e-6)  
 #limiting values, amplitude must be greater than 0. As I am creating a probability density function, the area under the curve sums to 1.   
 #gamma can't equal zero, otherwise there is a math error.  
 #alpha can't equal 0 otherwise the density will be equal to 1 for everything. It can't be less than zero as that'll cause the density to be greater than 1.  
  
 predictedDensity <- amplitude\*(1+((x-mu)^2/gamma^2))^(-alpha)  
}  
  
negLogLike <- function(parameters, x){  
 predictedDens <- Moffat(parameters, x)  
 return(-sum(log(predictedDens)))  
}

When putting the above model into optim, the returned results for the list of parameters were unrealistic. In order to work out why I plotted the Negative Log Likelihood Function to varying parameters as the objective of the optim function is to minimize the Negative Log Likelihood Function.

The baseline parameters are (0.5, 0, 1, 1) and x is 1 in all estimates of the negative log likelihood.

#changing the value of the amplitude  
amplitude <- seq(0.25, 5, by = 0.25)  
  
i <- 1  
changingA <- vector("numeric", length = 20)  
  
while(i<=20){  
 parameters <- c(amplitude[[i]], 4115, 1, 1)  
 changingA[i] <- (negLogLike(parameters, xl2))  
 i <- i+1  
}  
  
#changing the value of mu  
mu <- seq(4110, 4120, by = 0.5)  
  
i <- 1  
changingMu <- vector("numeric", length = 21)  
  
while(i<=21){  
 parameters <- c(0.5, mu[[i]], 1, 1)  
 changingMu[i] <- (negLogLike(parameters, xl2))  
 i <- i+1  
}  
  
#changing the value of gamma  
gamma <- seq(0.25, 10, by = 0.25)  
  
i <- 1  
changingGamma <- vector("numeric", length = 40)  
  
while(i<=40){  
 parameters <- c(0.5, 4115, gamma[[i]], 1)  
 changingGamma[i] <- (negLogLike(parameters, xl2))  
 i <- i+1  
}  
  
#changing the value of alpha  
alpha <- seq(0.25, 10, by = 0.25)  
  
i <- 1  
changingAlpha <- vector("numeric", length = 40)  
  
while(i<=40){  
 parameters <- c(0.5, 4115, 1, alpha[[i]])  
 changingAlpha[i] <- (negLogLike(parameters, xl2))  
 i <- i+1  
}  
  
par(mfrow = c(2,2))  
plot(amplitude, changingA, type = "l", xlab = "Amplitude", ylab = "Negative Log Likelihood")  
plot(mu, changingMu, type = "l", xlab = "Mu", ylab = "Negative Log Likelihood")  
plot(gamma, changingGamma, type = "l", xlab = "Gamma", ylab = "Negative Log Likelihood")  
plot(alpha, changingAlpha, type = "l", xlab = "Alpha", ylab = "Negative Log Likelihood")

 By the plots, it can be interpreted that optim likes to blow up the value of A, the amplitude. The larger the value of A, the smaller the Negative Log Likelihood. A similar thing happens for the value of gamma and the inverse happens for the value of alpha. This is not ideal.

As the goal is to try to fit to a probability density function, the amplitude can be expressed in terms of the other parameters gamma and alpha. From this point on the core width gamma is referred to as alpha and the power index alpha is referred to as beta

The equation of the probability density function is as follows:

In order for the function to remain positive, alpha must be greater than 0 and beta must be greater than or equal to 1.5.

MOFFATPDF <- function(parameters, x){  
 mu <- parameters[[1]]  
 alpha <- parameters[[2]]  
 beta <- parameters[[3]]  
   
 alpha <- pmax(alpha, 1e-6)  
 beta <- pmax(beta, 1.5)  
   
 predictedDensity <- (gamma(beta)/(alpha\*sqrt(pi)\*gamma(beta-0.5)))\*(1+((x-mu)/alpha)^2)^-beta  
 return(predictedDensity)  
}

The optim function has two requirements. The first is a function that it sets out to minimize. For this I will use the Negative Log Likelihood of the function. The second requirement is the gradient of the function that is being minimized. For this I will take the partial derivative of the Negative Log Likelihood with respect to each parameter.

To obtain the Negative Log Likelihood. First I take the Likelihood of the parameters of f(x). Next I take the log of the likelihood and apply the log law : . Taking the negative of this yields the Negative Log Likelihood.

NLL <- function(parameters, x){  
 predictedDensity <- MOFFATPDF(parameters, x)  
 return(-sum(log(predictedDensity)))  
}

Now I need the partial derivatives of the negative log likelihood with respect to each parameter. Before doing so I applied some log laws to transform the equation.

Log Laws:

By applying these log laws the negative log likelihood transforms into an equation that is simpler to obtain the partial derivatives to.

In order to take the partial derivative with respect to I utilised the chain rule. Let $ u = $. Let $ v = x - $ $ = -(-) $

Resulting in

A similar approach was used to take the partial derivative with respect to and . Resulting in

To obtain the partial derivative with respect to beta I will be using the Rcode of digamma

derivMOFFATPDF <- function(parameters, x){  
 mu <- parameters[[1]]  
 alpha <- parameters[[2]]  
 beta <- parameters[[3]]  
   
   
 dMu <- -sum((2\*beta\*(x-mu))/(alpha^2 + (x-mu)^2))  
 dAlpha <- -sum(-1/alpha + (2\*beta\*(x-mu)^2)/(alpha^3+alpha\*(x-mu)^2))  
 dBeta <- -sum(digamma(beta)-digamma(beta-0.5)-log(1+(x-mu)^2/alpha^2))  
   
 return(c(dMu, dAlpha, dBeta))  
}

Data to optimise.

xl2 <- line2$x  
xl2.5 <- line2.5$x  
xVall2 <- seq(min(xl2), max(xl2), length = 963)  
xVall2.5 <- seq(min(xl2.5), max(xl2.5), length = 291)

Optimising the parameters.

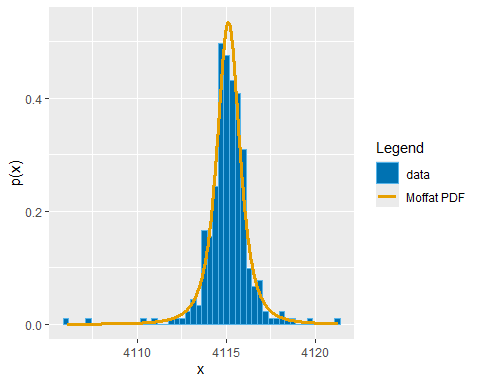
initialGUESS <- c(mu = 4115, alpha = 1, beta = 4)  
lowerBOUNDS <- c(0, 1e-6, 1.5)  
  
RESULT <- optim(par = initialGUESS, fn = NLL, gr = derivMOFFATPDF, x = xl2.5, method = "L-BFGS-B", lower = lowerBOUNDS)  
optimPAR2.5 <- RESULT$par  
optimPAR2.5

## mu alpha beta   
## 4115.125763 1.082681 1.771935

Visualising the results

optimDENS2.5 <- MOFFATPDF(optimPAR2.5, xVall2.5)  
## plotting layers in ggplot  
ggplot(line2.5) +   
 geom\_histogram(aes(x, y=after\_stat(density), colour="data"),bins=50,fill=cbPalette[5]) +   
 stat\_function(fun=MOFFATPDF,args = list(parameters = optimPAR2.5), n=1000, aes(colour="Moffat PDF"), linewidth=1.2) +  
 ylab("p(x)") +   
 scale\_colour\_manual("Legend", values = cbPalette)

## Warning in stat\_function(fun = MOFFATPDF, args = list(parameters = optimPAR2.5), : All aesthetics have length 1, but the data has 291 rows.  
## ℹ Did you mean to use `annotate()`?



Optimising the parameters on a larger dataset.

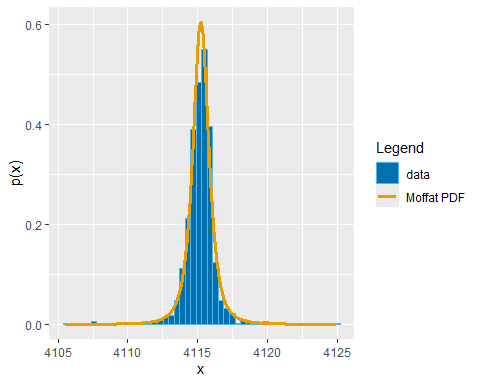
RESULT <- optim(par = initialGUESS, fn = NLL, gr = derivMOFFATPDF, x = xl2, method = "L-BFGS-B", lower = lowerBOUNDS)  
optimPAR2 <- RESULT$par  
optimPAR2

## mu alpha beta   
## 4115.2307593 0.9860251 1.8368043

Visualising the results

optimDENS2 <- MOFFATPDF(optimPAR2, xVall2)  
## plotting layers in ggplot  
ggplot(line2) +   
 geom\_histogram(aes(x, y=after\_stat(density), colour="data"),bins=50,fill=cbPalette[5]) +   
 stat\_function(fun=MOFFATPDF,args = list(parameters = optimPAR2), n=1000, aes(colour="Moffat PDF"), linewidth=1.2) +  
 ylab("p(x)") +   
 scale\_colour\_manual("Legend", values = cbPalette)

## Warning in stat\_function(fun = MOFFATPDF, args = list(parameters = optimPAR2), : All aesthetics have length 1, but the data has 963 rows.  
## ℹ Did you mean to use `annotate()`?

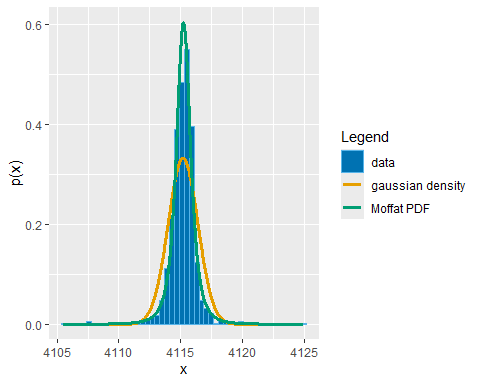


Visually comparing the Moffat PDF and the Normal PDF when the parameters are estimated.

ggplot(line2) +  
 geom\_histogram(aes(x, y = after\_stat(density), colour = "data"), bins = 50, fill=cbPalette[5]) +  
 stat\_function(fun = gaussian\_density, args = list(parameters = optimized\_parameters), n=1000, aes(colour="gaussian density"), linewidth=1.2) +  
 stat\_function(fun=MOFFATPDF,args = list(parameters = optimPAR2), n=1000, aes(colour="Moffat PDF"), linewidth=1.2) +  
 ylab("p(x)") +   
 scale\_colour\_manual("Legend", values = cbPalette)

## Warning in stat\_function(fun = gaussian\_density, args = list(parameters = optimized\_parameters), : All aesthetics have length 1, but the data has 963 rows.  
## ℹ Did you mean to use `annotate()`?

## Warning in stat\_function(fun = MOFFATPDF, args = list(parameters = optimPAR2), : All aesthetics have length 1, but the data has 963 rows.  
## ℹ Did you mean to use `annotate()`?



It is clear that the 1D Moffat Distribution provides is a much better fit to the data.

Now I will attempt to fit a 2D Moffat Distribution to the data.

GOODNESS OF FIT Kolmogorov-Smirnov Tests

For unknown distribution F. is a pre-specified, not data dependent distribution model. < Problematic statement in this instance. As the parameters of the Moffat 1D distribution model were derived from the data. This will lead to highly conservative tests.

Test purpose: Given that . Test against all alternatives in Statistic Definition: Supremum Distance () between and . Where If the null hypothesis holds then the value of tends to be small.

The maximum difference betwen and happens at a certain value of Where is the j-th sorted

has an asymptotic cdf given by the Kolmogorov-Smirnov K function:

Let F be the data in line 2 Let be the modelled normal distribution

ks.test(x = line2$x, y = "pnorm")

## Warning in ks.test.default(x = line2$x, y = "pnorm"): ties should not be  
## present for the Kolmogorov-Smirnov test

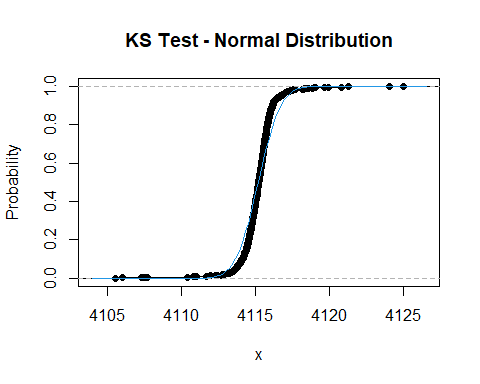
##   
## Asymptotic one-sample Kolmogorov-Smirnov test  
##   
## data: line2$x  
## D = 1, p-value < 2.2e-16  
## alternative hypothesis: two-sided

noTiesL2 <- unique(xl2)  
ks.test(x = noTiesL2, y = "pnorm")

##   
## Asymptotic one-sample Kolmogorov-Smirnov test  
##   
## data: noTiesL2  
## D = 1, p-value < 2.2e-16  
## alternative hypothesis: two-sided

Strongly rejects the null hypothesis

plot(ecdf(noTiesL2), main = "KS Test - Normal Distribution", ylab = "Probability")  
curve(pnorm(x, mean = 4115.19535, sd = 1.20138), add = TRUE, col = 4)



Let be the modelled 1D Moffat Distribution.

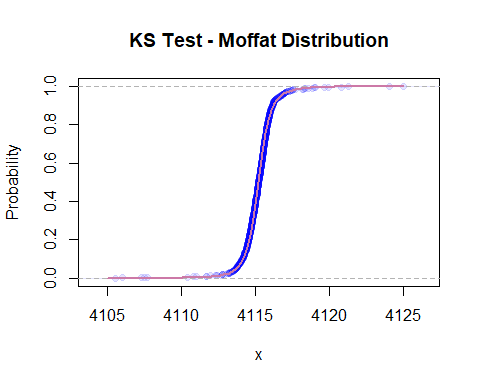
xRange <- seq(4105, 4125, length = 100)  
integrate(MOFFATPDF, lower = min(xRange), upper = max(xRange), parameters = optimPAR2)

## 0.9990977 with absolute error < 3.2e-05

MOFFATCDF <- vector("numeric", length = 100)  
  
for (i in 1:100) {  
 MOFFATCDF[i] <- integrate(MOFFATPDF, lower = xRange[1], upper = xRange[i], parameters = optimPAR2)$value  
}  
MOFFATCDF

## [1] 0.000000e+00 2.302260e-05 4.779631e-05 7.449375e-05 1.033086e-04  
## [6] 1.344586e-04 1.681893e-04 2.047782e-04 2.445398e-04 2.878314e-04  
## [11] 3.350601e-04 3.866915e-04 4.432594e-04 5.053780e-04 5.737561e-04  
## [16] 6.492154e-04 7.327116e-04 8.253607e-04 9.284716e-04 1.043587e-03  
## [21] 1.172531e-03 1.317476e-03 1.481018e-03 1.666281e-03 1.877043e-03  
## [26] 2.117903e-03 2.394498e-03 2.713790e-03 3.084434e-03 3.517290e-03  
## [31] 4.026095e-03 4.628398e-03 5.346838e-03 6.210942e-03 7.259674e-03  
## [36] 8.545100e-03 1.013774e-02 1.213449e-02 1.467056e-02 1.793774e-02  
## [41] 2.221271e-02 2.790149e-02 3.561002e-02 4.625553e-02 6.123955e-02  
## [46] 8.270065e-02 1.138292e-01 1.590838e-01 2.237771e-01 3.119712e-01  
## [51] 4.220707e-01 5.430605e-01 6.577920e-01 7.527567e-01 8.238712e-01  
## [56] 8.741104e-01 9.087498e-01 9.325831e-01 9.491526e-01 9.608632e-01  
## [61] 9.692967e-01 9.754874e-01 9.801162e-01 9.836373e-01 9.863588e-01  
## [66] 9.884932e-01 9.901896e-01 9.915542e-01 9.926643e-01 9.935765e-01  
## [71] 9.943330e-01 9.949658e-01 9.954992e-01 9.959522e-01 9.963393e-01  
## [76] 9.966722e-01 9.969602e-01 9.972106e-01 9.974294e-01 9.976215e-01  
## [81] 9.977909e-01 9.979408e-01 9.980741e-01 9.981929e-01 9.982993e-01  
## [86] 9.983948e-01 9.984808e-01 9.985584e-01 9.986287e-01 9.986925e-01  
## [91] 9.987506e-01 9.988035e-01 9.988520e-01 9.988964e-01 9.989371e-01  
## [96] 9.989746e-01 9.990091e-01 9.990409e-01 9.990704e-01 9.990977e-01

CDF <- cbind(xRange, MOFFATCDF)  
  
plot(ecdf(noTiesL2), main = "KS Test - Moffat Distribution", ylab = "Probability", col = rgb(0,0,1, alpha = 0.1))  
lines(CDF[,1], CDF[,2], type = "l", col = cbPalette[7], lwd = 2)



ks.test(x = noTiesL2, y = MOFFATCDF)

##   
## Asymptotic two-sample Kolmogorov-Smirnov test  
##   
## data: noTiesL2 and MOFFATCDF  
## D = 1, p-value < 2.2e-16  
## alternative hypothesis: two-sided

tweak

xRange <- seq(4105, 4125, length = 100)  
MCDF <- function(x){  
  
 return(integrate(MOFFATPDF, lower = 4100, upper = x, parameters = optimPAR2)$value)  
}  
  
VMCDF <- Vectorize(MCDF, vectorize.args = "x")  
VMCDF(x = 4115:4225)

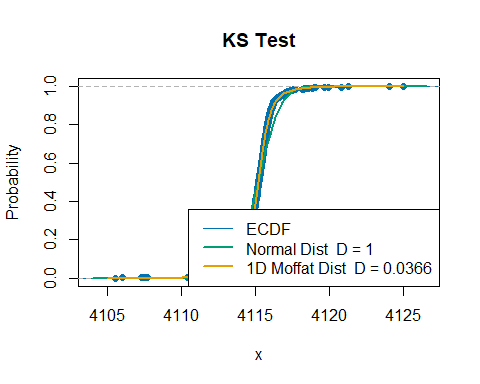
## [1] 0.3650173 0.8490324 0.9648387 0.9874405 0.9940948 0.9967038 0.9979337  
## [8] 0.9985909 0.9989752 0.9992157 0.9993743 0.9994834 0.9995611 0.9996180  
## [15] 0.9996608 0.9996935 0.9997191 0.9997393 0.9997556 0.9997688 0.9997797  
## [22] 0.9997888 0.9997964 0.9998028 0.9998082 0.9998129 0.9998169 0.9998204  
## [29] 0.9998235 0.9998261 0.9998285 0.9998305 0.9998324 0.9998340 0.9998355  
## [36] 0.9998368 0.9998380 0.9998391 0.9998400 0.9998409 0.9998417 0.9998424  
## [43] 0.9998431 0.9998437 0.9998443 0.9998448 0.9998453 0.9998457 0.9998461  
## [50] 0.9998465 0.9998468 0.9998471 0.9998474 0.9998477 0.9998480 0.9998482  
## [57] 0.9998484 0.9998487 0.9998489 0.9998490 0.9998492 0.9998494 0.9998495  
## [64] 0.9998497 0.9998498 0.9998500 0.9998501 0.9998502 0.9998503 0.9998504  
## [71] 0.9998505 0.9998506 0.9998507 0.9998508 0.9998509 0.9998509 0.9998510  
## [78] 0.9998511 0.9998512 0.9998512 0.9998513 0.9998513 0.9998514 0.9998514  
## [85] 0.9998515 0.9998515 0.9998516 0.9998516 0.9998517 0.9998517 0.9998518  
## [92] 0.9998518 0.9998518 0.9998519 0.9998519 0.9998519 0.9998520 0.9998520  
## [99] 0.9998520 0.9998520 0.9998521 0.9998521 0.9998521 0.9998521 0.9998522  
## [106] 0.9998522 0.9998522 0.9998522 0.9998523 0.9998523 0.9998523

#ordering the data  
sortNTL2 <- sort(noTiesL2)  
#sortNTL2  
#MCDF(xRange = sortNTL2, pdf = MOFFATPDF)

ks.test(x = sortNTL2, y = VMCDF)

##   
## Asymptotic one-sample Kolmogorov-Smirnov test  
##   
## data: sortNTL2  
## D = 0.036258, p-value = 0.2055  
## alternative hypothesis: two-sided

plot(ecdf(noTiesL2), main = "KS Test", ylab = "Probability", col = cbPalette[5])  
curve(pnorm(x, mean = 4115.19535, sd = 1.20138), add = TRUE, col = cbPalette[3], lwd = 2)  
lines(CDF[,1], CDF[,2], type = "l", col = cbPalette[2], lwd = 2)  
legend("bottomright", legend = c("ECDF", "Normal Dist D = 1", "1D Moffat Dist D = 0.0366"), col = cbPalette[c(5,3,2)], lty = c(1,1,1), lwd = c(1,2,2))



Visually and numerically, it is evident that the normal distribution is not a good fit for the data. While the null hypothesis of the Kolmogorov-Smirnov test is accepted for the 1D Moffat Distribution.

Accept the null hypothesis !Warning assumption violation - model derived from the dataset. Use bootstrapping KS or Anderson-Darling

library(DescTools)  
AndersonDarlingTest(x = sortNTL2, null = VMCDF)

##   
## Anderson-Darling test of goodness-of-fit  
## Null hypothesis: distribution 'VMCDF'  
##   
## data: sortNTL2  
## An = 1.4766, p-value = 0.1822

library(goftest)  
ad.test(x = sortNTL2, null = "norm")

##   
## Anderson-Darling test of goodness-of-fit  
## Null hypothesis: Normal distribution  
## Parameters assumed to be fixed  
##   
## data: sortNTL2  
## An = Inf, p-value = 6.936e-07

The next goal is to create some pp plots and qq plots for the 1D Moffat Function. in order to do that I require a function that calculates the Quantiles of the moffat function.

The QQ-plot should compare the the quantiles from the data to the quantiles of the 1D Moffat model. Here is the quantiles from the data.

quantile(sortNTL2)

## 0% 25% 50% 75% 100%   
## 4105.540 4114.689 4115.220 4115.714 4124.971

Now i need to calculate the quantiles of the create 1D Moffat model. In order to find the quantiles, I will use interpolation.

# Define the inverse cumulative distribution function (quantile function) for Moffat distribution  
qmoffat <- function(p, params) {  
 # Define a function to find the root (quantile) of CDF - p, this equates to 0.  
 findRoot <- function(x) {  
 integrate(MOFFATPDF, lower = -Inf, upper = x, parameters = params)$value - p  
 }  
   
 # Find the root (quantile) using uniroot. For a probability, this will look to find a value of x that causes the findRoot function to equate to 0. This value of x is the quantile.  
 quantile <- uniroot(findRoot, interval = c(4000,4200))$root  
 return(quantile)  
}  
  
probs <- seq(0.01, 0.99, by = 0.01)  
  
qmoffat(0.5, optimPAR2)

## [1] 4115.231

Vqmoffat <- Vectorize(qmoffat, vectorize.args = "p")  
quantiles <- Vqmoffat(p = probs, params = optimPAR2)  
  
data.frame(probs, quantiles)

## probs quantiles  
## 1 0.01 4112.206  
## 2 0.02 4112.963  
## 3 0.03 4113.332  
## 4 0.04 4113.568  
## 5 0.05 4113.738  
## 6 0.06 4113.870  
## 7 0.07 4113.976  
## 8 0.08 4114.066  
## 9 0.09 4114.143  
## 10 0.10 4114.210  
## 11 0.11 4114.269  
## 12 0.12 4114.323  
## 13 0.13 4114.372  
## 14 0.14 4114.417  
## 15 0.15 4114.458  
## 16 0.16 4114.497  
## 17 0.17 4114.533  
## 18 0.18 4114.567  
## 19 0.19 4114.599  
## 20 0.20 4114.629  
## 21 0.21 4114.658  
## 22 0.22 4114.686  
## 23 0.23 4114.712  
## 24 0.24 4114.738  
## 25 0.25 4114.762  
## 26 0.26 4114.786  
## 27 0.27 4114.809  
## 28 0.28 4114.831  
## 29 0.29 4114.853  
## 30 0.30 4114.874  
## 31 0.31 4114.894  
## 32 0.32 4114.914  
## 33 0.33 4114.934  
## 34 0.34 4114.953  
## 35 0.35 4114.972  
## 36 0.36 4114.991  
## 37 0.37 4115.009  
## 38 0.38 4115.027  
## 39 0.39 4115.045  
## 40 0.40 4115.062  
## 41 0.41 4115.080  
## 42 0.42 4115.097  
## 43 0.43 4115.114  
## 44 0.44 4115.131  
## 45 0.45 4115.148  
## 46 0.46 4115.164  
## 47 0.47 4115.181  
## 48 0.48 4115.198  
## 49 0.49 4115.214  
## 50 0.50 4115.231  
## 51 0.51 4115.247  
## 52 0.52 4115.264  
## 53 0.53 4115.281  
## 54 0.54 4115.297  
## 55 0.55 4115.314  
## 56 0.56 4115.331  
## 57 0.57 4115.348  
## 58 0.58 4115.365  
## 59 0.59 4115.382  
## 60 0.60 4115.399  
## 61 0.61 4115.417  
## 62 0.62 4115.435  
## 63 0.63 4115.453  
## 64 0.64 4115.471  
## 65 0.65 4115.490  
## 66 0.66 4115.508  
## 67 0.67 4115.528  
## 68 0.68 4115.547  
## 69 0.69 4115.567  
## 70 0.70 4115.588  
## 71 0.71 4115.609  
## 72 0.72 4115.631  
## 73 0.73 4115.653  
## 74 0.74 4115.676  
## 75 0.75 4115.699  
## 76 0.76 4115.724  
## 77 0.77 4115.749  
## 78 0.78 4115.776  
## 79 0.79 4115.803  
## 80 0.80 4115.832  
## 81 0.81 4115.863  
## 82 0.82 4115.895  
## 83 0.83 4115.929  
## 84 0.84 4115.965  
## 85 0.85 4116.003  
## 86 0.86 4116.045  
## 87 0.87 4116.090  
## 88 0.88 4116.138  
## 89 0.89 4116.192  
## 90 0.90 4116.252  
## 91 0.91 4116.319  
## 92 0.92 4116.396  
## 93 0.93 4116.485  
## 94 0.94 4116.592  
## 95 0.95 4116.723  
## 96 0.96 4116.893  
## 97 0.97 4117.129  
## 98 0.98 4117.499  
## 99 0.99 4118.255

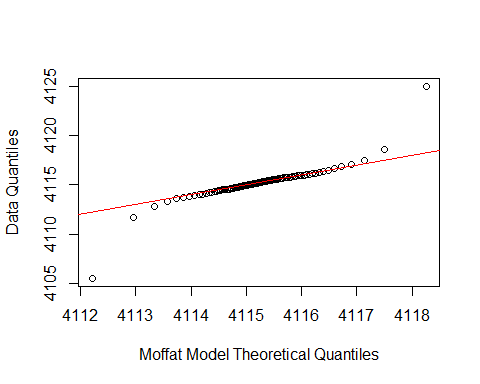
quantile(sortNTL2) #data

## 0% 25% 50% 75% 100%   
## 4105.540 4114.689 4115.220 4115.714 4124.971

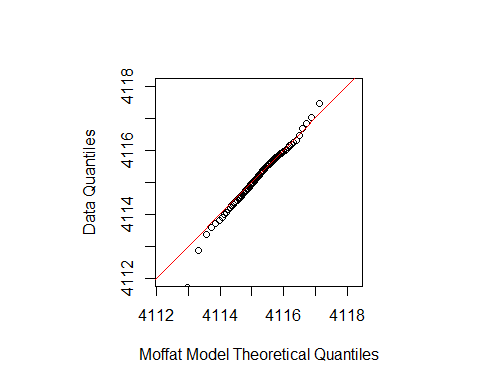
quantile(quantiles) #theoretical values

## 0% 25% 50% 75% 100%   
## 4112.206 4114.774 4115.231 4115.688 4118.255

qqplot(quantiles, sortNTL2, xlab = "Moffat Model Theoretical Quantiles", ylab = "Data Quantiles")  
abline(a = 0, b =1, col = "red")

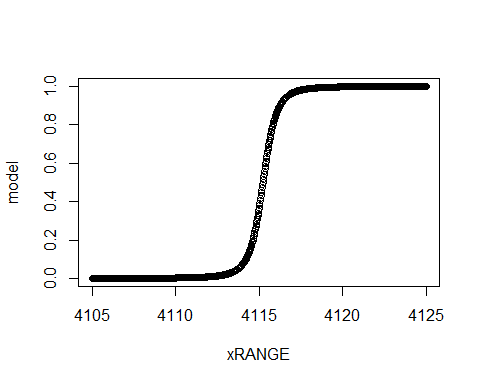


par(pty = "s")  
qqplot(quantiles, sortNTL2, ylim = c(4112, 4118), xlab = "Moffat Model Theoretical Quantiles", ylab = "Data Quantiles")  
abline(a = 0, b = 1, col = "red")

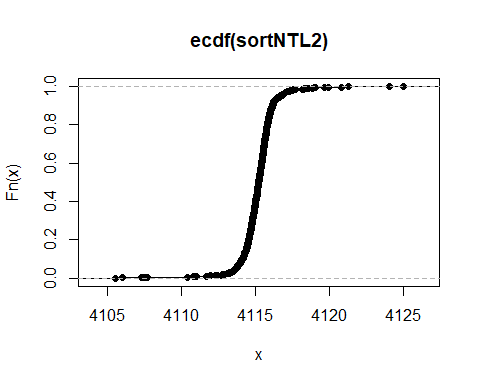


A P-P plot is used for assessing how close a dataset fits to a particular model. In this instance it is used to see how well the dataset (the “1 Dimensional” Slice of the celestial source) fits to the 1D Moffat distribution with the parameters given by the previous optimisation. The two cumulative distribution functions are plotted against each other.

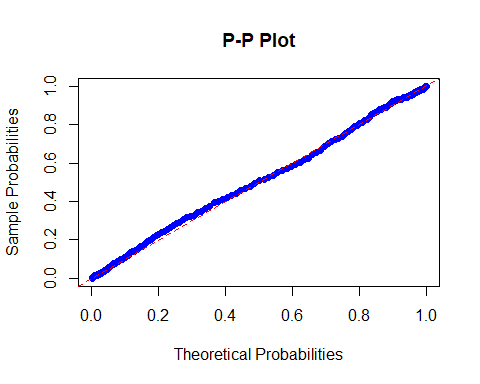
xRANGE <- seq(4105, 4125, length = length(sortNTL2))  
model <- VMCDF(xRANGE) #this represents the cumulative distribution of the 1D moffat distribution   
plot(x = xRANGE, y = model)



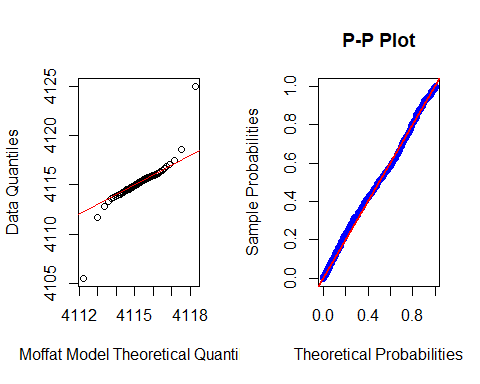
dataset <- ecdf(sortNTL2)#this represents the cumulative distribution of the dataset  
plot(dataset)



thereoticalProbs <- model  
empiricalProbs <- dataset(xRANGE)  
  
plot(thereoticalProbs, empiricalProbs, main = "P-P Plot", xlab = "Theoretical Probabilities", ylab = "Sample Probabilities", col = "blue", pch = 16)  
abline(0, 1, col = "red", lty = 2) # Add a reference line



par(mfrow = c(1,2))  
qqplot(quantiles, sortNTL2, xlab = "Moffat Model Theoretical Quantiles", ylab = "Data Quantiles")  
abline(a = 0, b =1, col = "red")  
plot(thereoticalProbs, empiricalProbs, main = "P-P Plot", xlab = "Theoretical Probabilities", ylab = "Sample Probabilities", col = "blue", pch = 16)  
abline(0, 1, col = "red", lwd = 2) # Add a reference line



plot with quartiles

q <- quantile(quantiles)  
ggplot(line2) +   
 geom\_histogram(aes(x, y=after\_stat(density), colour="data"),bins=50,fill=cbPalette[5]) +   
 stat\_function(fun=MOFFATPDF,args = list(parameters = optimPAR2), n=1000, aes(colour="Moffat PDF"), linewidth=1.2) +  
 ylab("p(x)") +   
 scale\_colour\_manual("Legend", values = cbPalette) +  
 geom\_vline(xintercept = c(q[2], q[3], q[4]), linetype = "dashed", lwd = 0.2, col = "black")

## Warning in stat\_function(fun = MOFFATPDF, args = list(parameters = optimPAR2), : All aesthetics have length 1, but the data has 963 rows.  
## ℹ Did you mean to use `annotate()`?

