# Bayes Analysis Assignment 2025

*by*

Myeni L (MYNLUN004), Mthetho K (MTHKWE006)

September 5, 2025

## Plaigarism declaration

Our group, **Lungani Myeni, Kwetsi Mthetho** Hereby declare that the work presented in this assignment for the course Experimental Design is entirely our own. We confirm that:

1. We recognise that plaigarism is a serious form of academic dishonesty.
2. Our work has not been previously submitted in whole or in part, for any other course or assessment.
3. We have acknowledged and referenced all the sources of information used in this assignment.
4. We have abided by all ethical guidelines for academic integrity as outlined by the university.

Myeni L.

Mthetho K.

# Contents



# List of Figures



# List of Tables

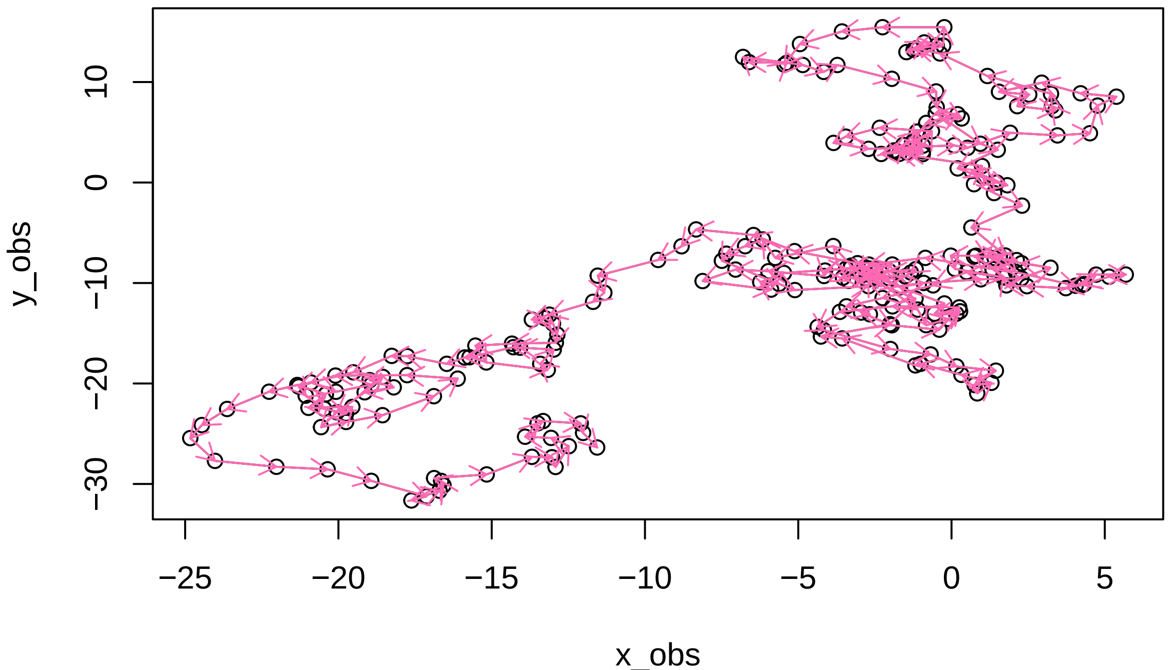



Figure 1: Observed Movement Path

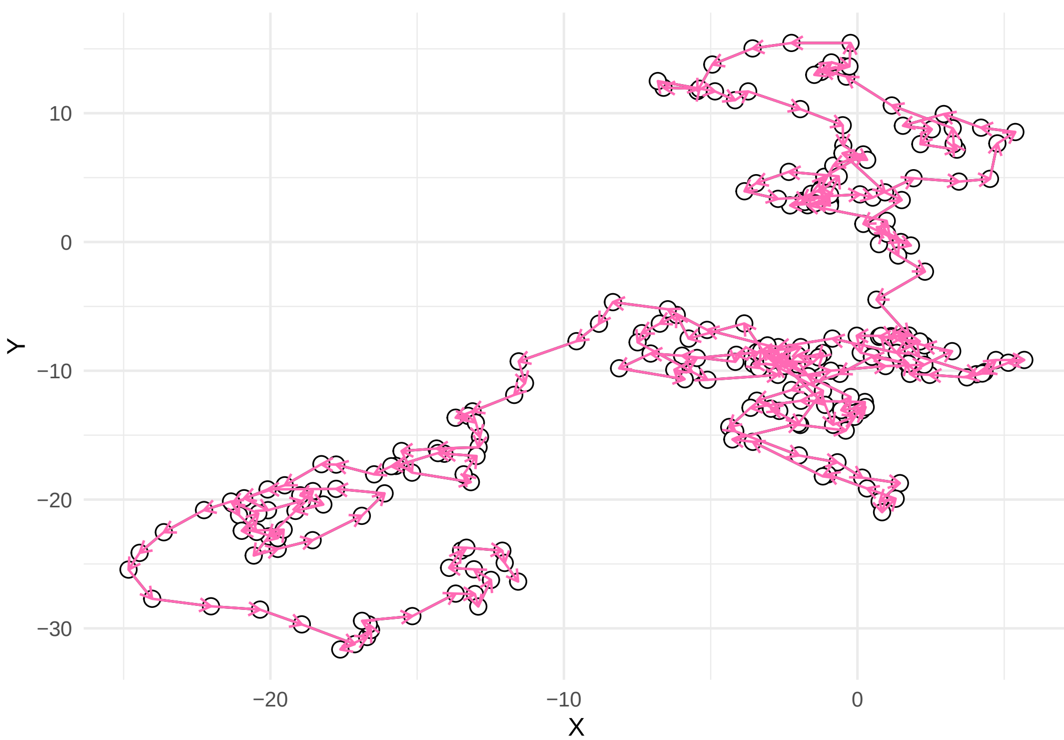
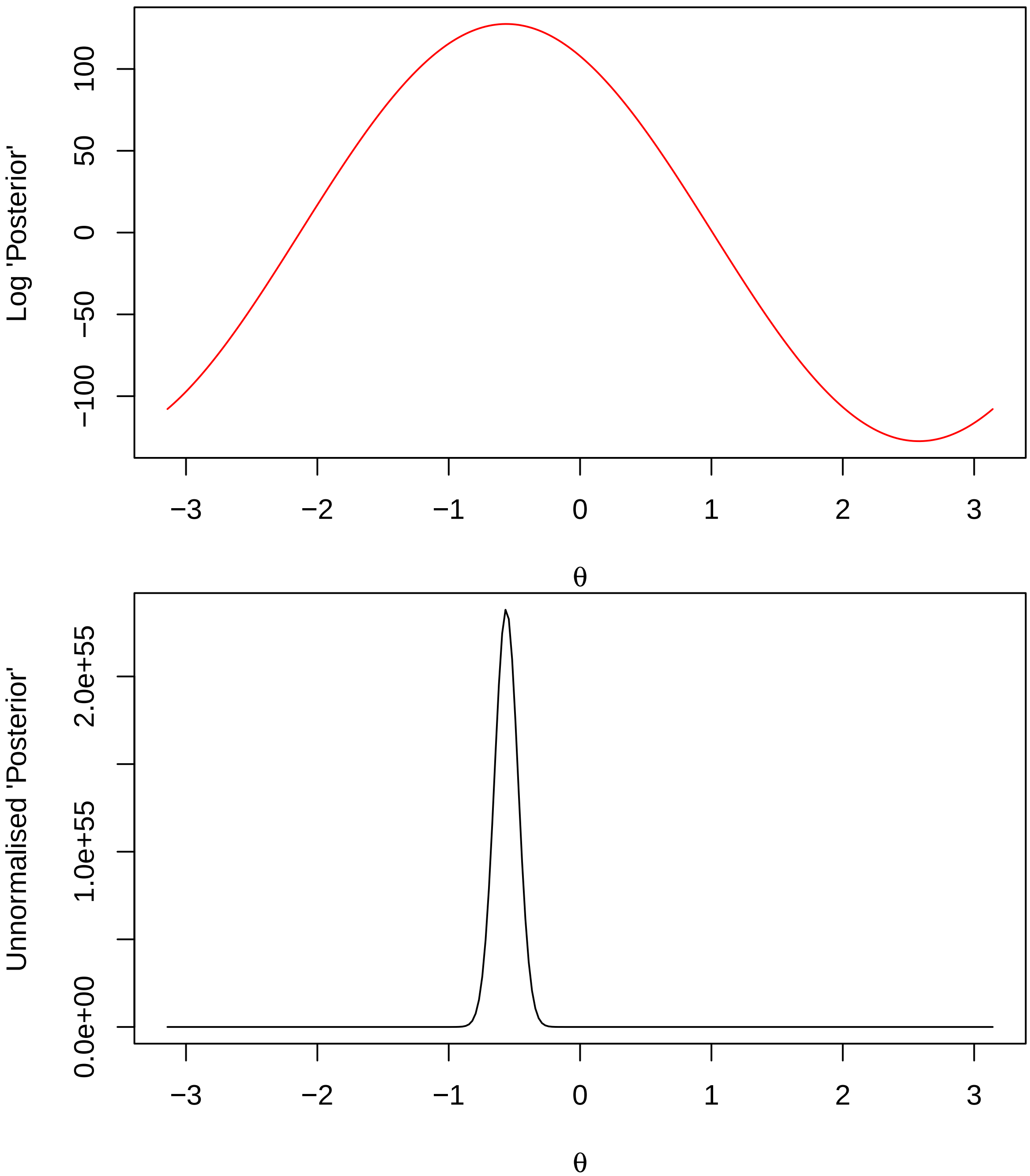
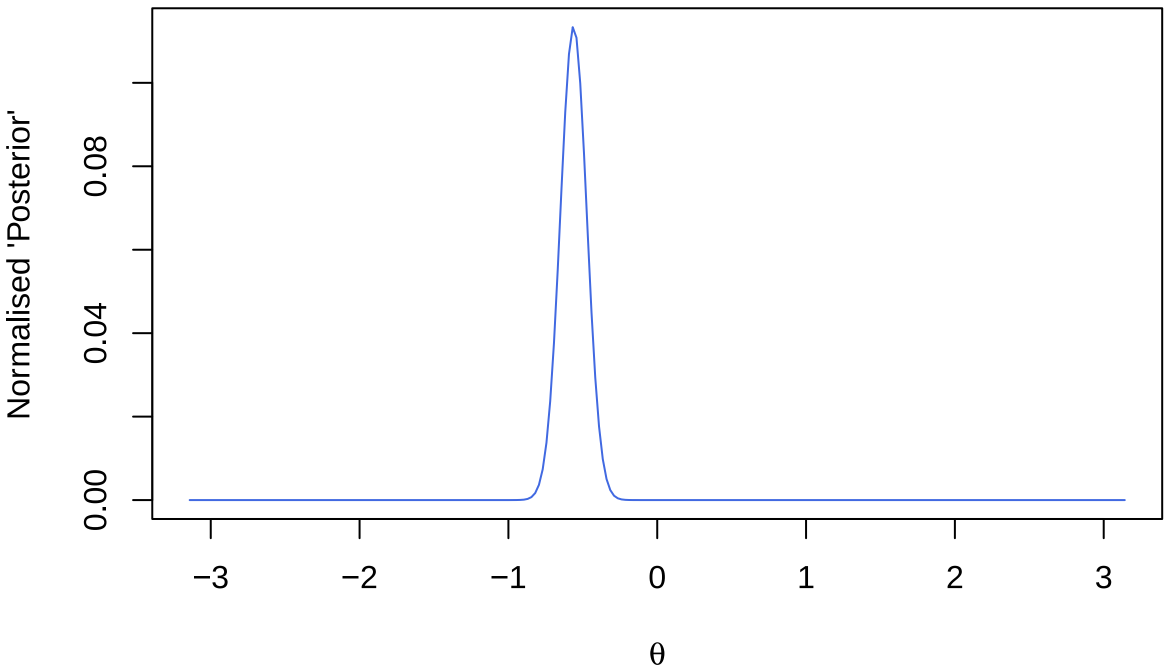
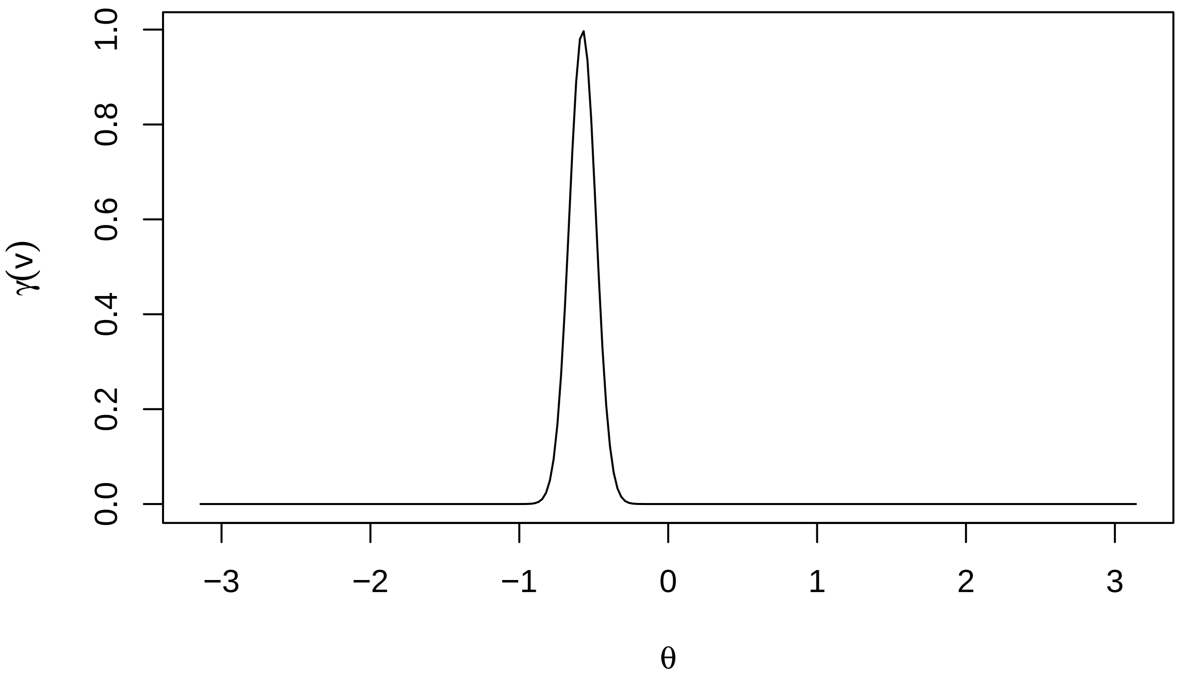


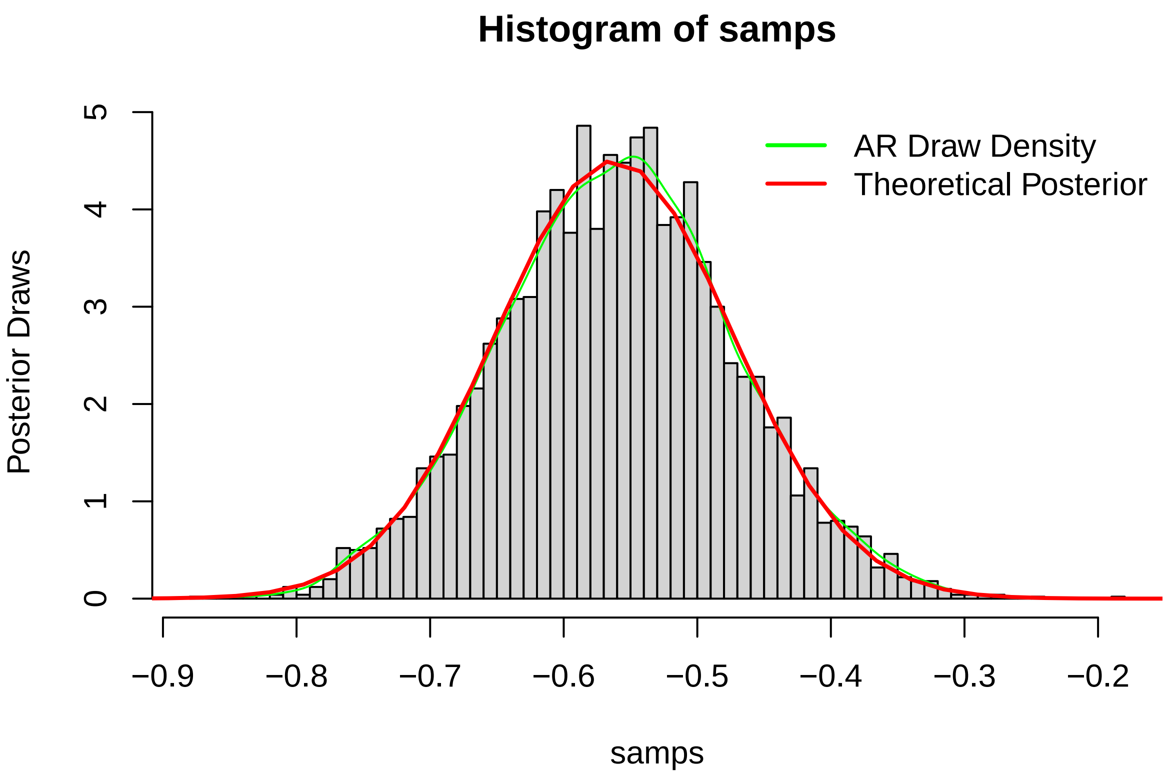

Figure 2: Observed Movement Path


Log 'Posterior'
100
50
0
−50
−100
−3
−2
−1
0
1
2
3
θ
Unnormalised 'Posterior'
2.0e+55
1.0e+55
0.0e+00
−3
−2
−1
0
1
2
3
θ


Normalised 'Posterior'
0.00
0.04
0.08
−3
−2
−1
0
1
2
3
θ

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



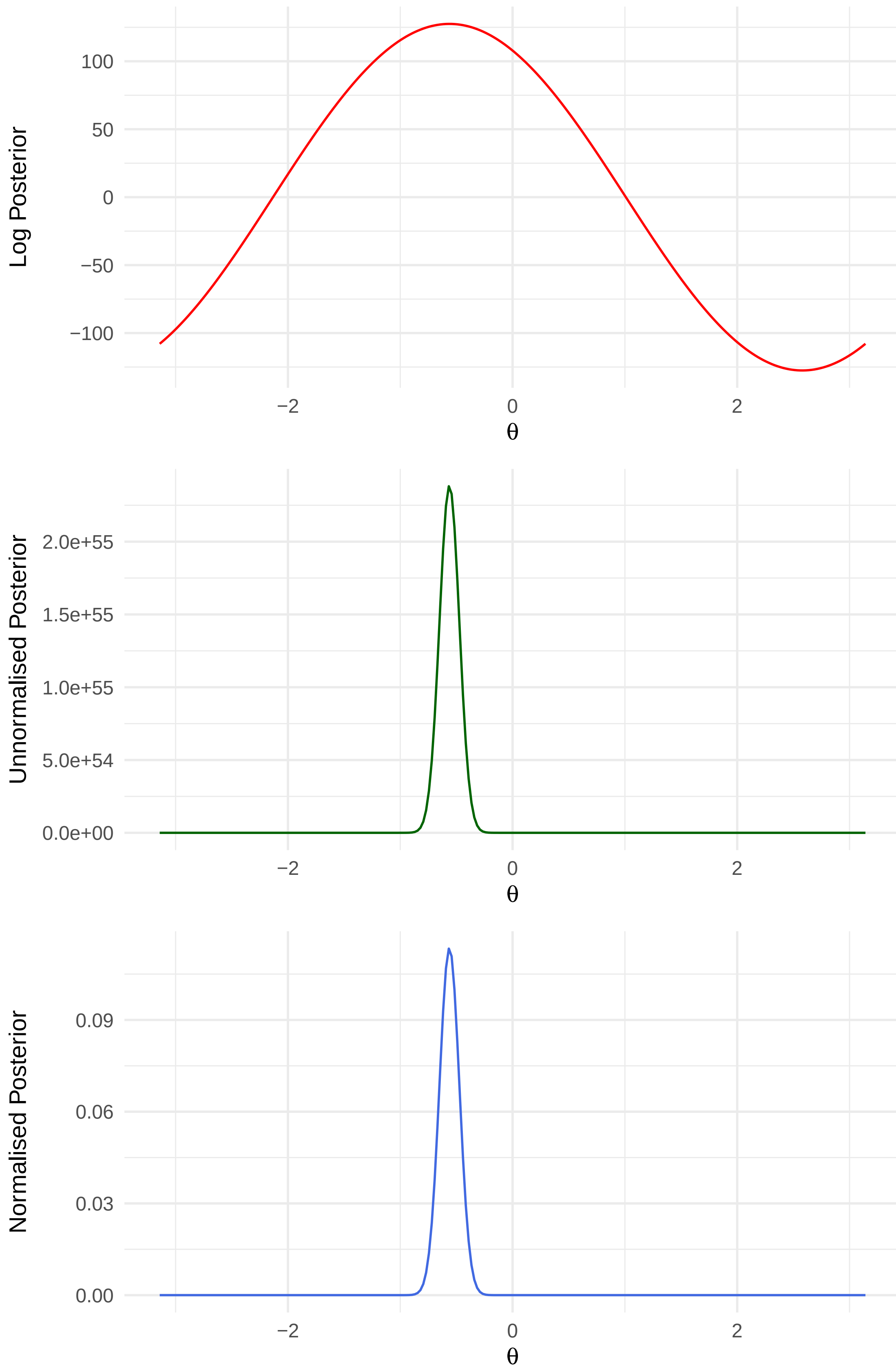



Figure 3: Posterior Distribution Versions

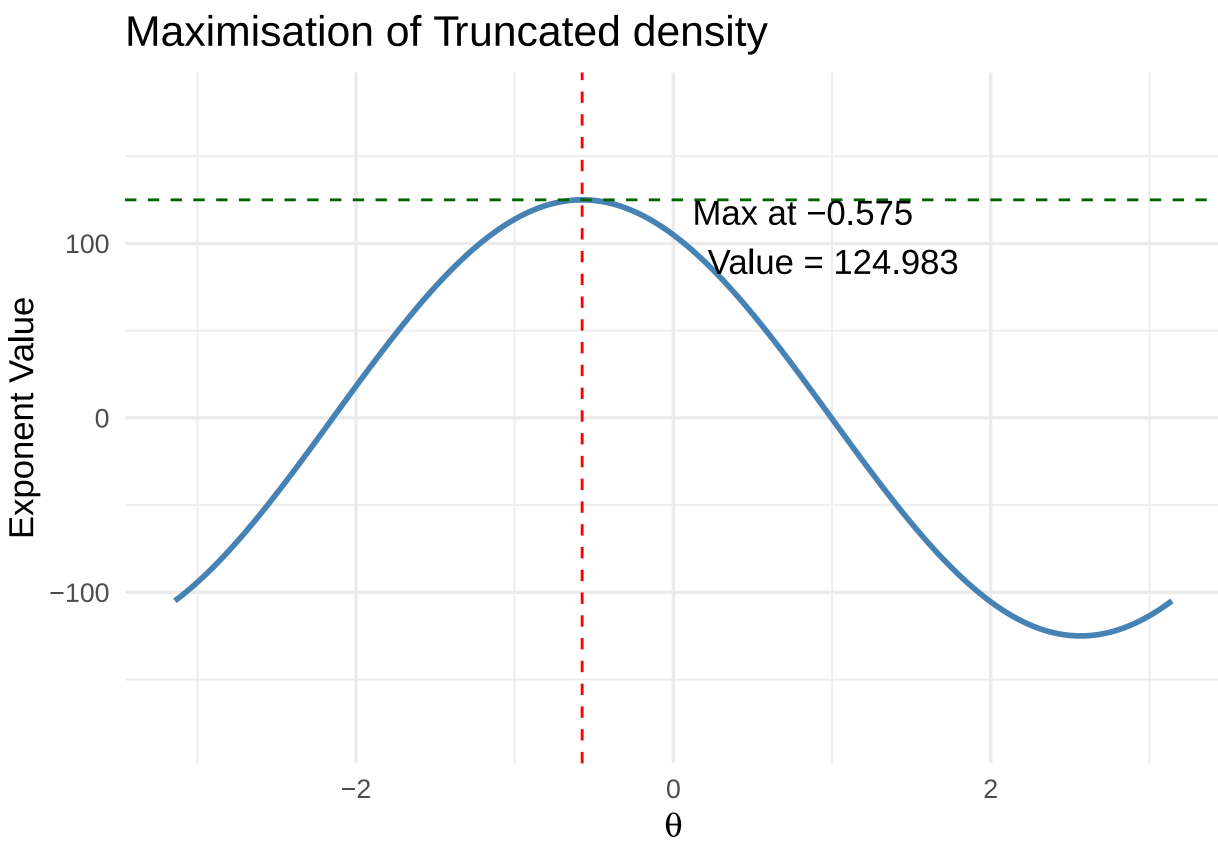



Figure 4: Natural Log Maximisation

Table 1: Summary of Posterior Samples for Draws

| Var1 | Var2 | Freq |
|---|---|---|
| A | Min. | -0.8723 |
| A | 1st Qu. | -0.6193 |
| A | Median | -0.5594 |
| A | Mean | -0.5601 |
| A | 3rd Qu. | -0.5020 |
| A | Max. | -0.1849 |

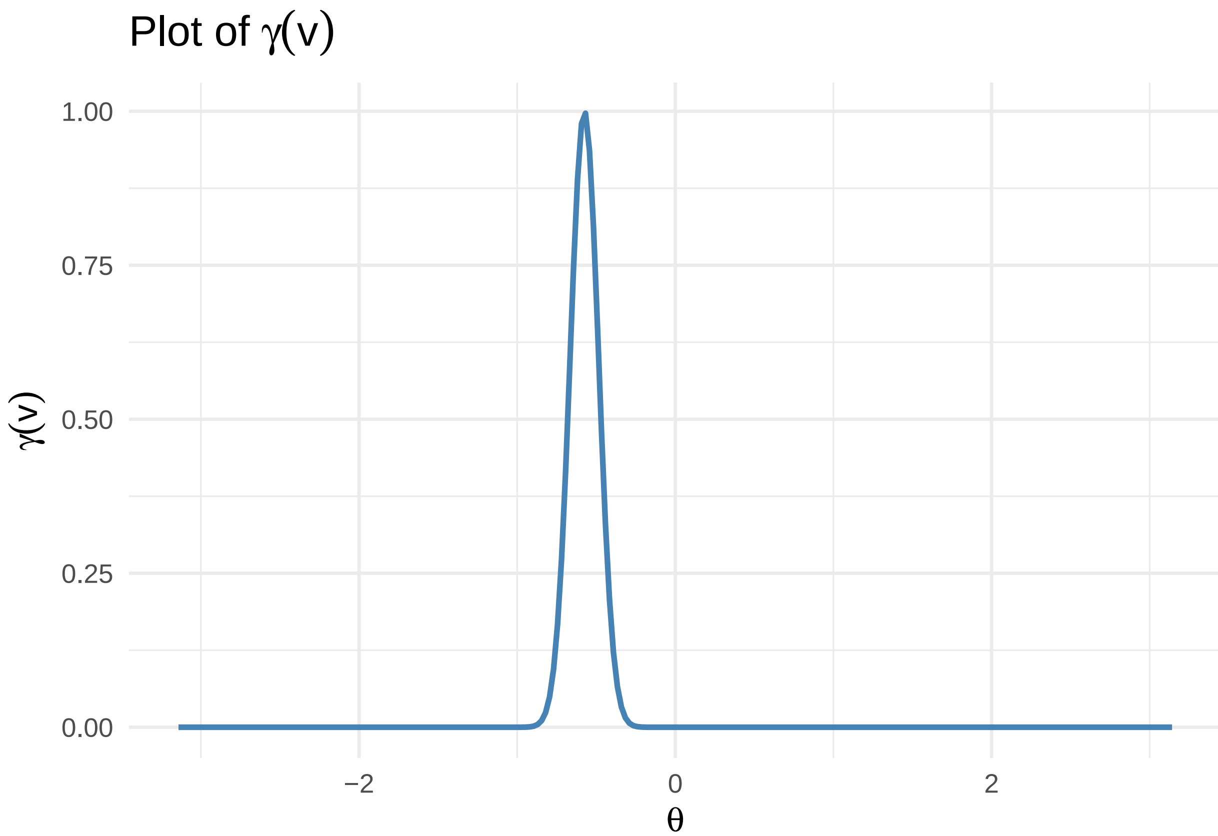



Figure 5: Gamma Function

# Question 3

```
##            [,1]
## [1,] -10.98223
## [2,] -26.85970
```

```
##              [,1]        [,2]
## [1,] 0.001903448 0.002211224
## [2,] 0.002211224 0.002610372
```

We want to find $\mathbb{E}(\mu_{251}|V)$ and $\text{Var}(\mu_{251}|V)$, using our posterior distribution draws of theta.

Note:

$$\mu_{n+1}|V$$

has unknown distribution but we know that the posterior distribution is such that

$$\mu_{n+1}|V,\theta \sim \text{MVN}(\mu_n + aM(\mu_n - \mu_{n-1}), I_2)$$

hence, by the law of total expectation, we have that:

$$\begin{aligned}\mathbb{E}(\mu_{251}|V) &= \mathbb{E}(\mathbb{E}(\mu_{251}|V,\theta)) \\ &= \mathbb{E}(\mu_{250} + aM(\mu_{250} - \mu_{249}))\end{aligned}$$

Now to find the expectation above, we make use of *Monte Carlo Integration*, i.e. we use the posterior draws to estimate the expectation as follows:

$$\mathbb{E}(\mu_{251}|V) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\mu_{250} + aM_{(i)}(\mu_{250} - \mu_{249})\right)$$

where $N$ is the number of posterior draws of $\theta$. In our case, $N = 5000$.

This is easily implemented in R to find that the $251^{th}$ true location estimate is:

$$\mathbb{E}(\mu_{251}|V) \approx \begin{pmatrix}0.7123 \\ 0.2925\end{pmatrix}$$

Now to find the variance, we make use of the law of total variance:

$$\begin{aligned}\text{Var}(\mu_{251}|V) &= \mathbb{E}(\text{Var}(\mu_{251}|V,\theta)) + \text{Var}(\mathbb{E}(\mu_{251}|V,\theta)) \\ &= \mathbb{E}(0) + \text{Var}(\mu_{250} + aM(\mu_{250} - \mu_{249}))\end{aligned}$$

where we used the fact that $\text{Var}(\mu_{251}|V,\theta) = 0$ since $\mu_{251}|V,\theta$ is MVN with identity covariance matrix.

So the job of finding the variance reduces to finding the variance of $\mu_{250} + aM(\mu_{250} - \mu_{249})$ which we can also estimate using Monte Carlo integration as follows:

| |
|---|
| Note: for ease of notation,<br>Let $X(\theta) = \mu_{250} + aM(\mu_{250} - \mu_{249})$ and<br>Let $\bar{X} = \frac{1}{M}\sum_{i=1}^{M} X(\theta_{(i)})$ |

$$\begin{aligned}\text{Var}(\mu_{251}|V) &= \text{Var}(X(\theta)) \\ &\approx \frac{1}{M-1}\sum_{i=1}^{M}(X(\theta_{(i)}) - \bar{X})(X(\theta_{(i)}) - \bar{X})^\top\end{aligned}$$

or equivalently:

$$\begin{aligned}\mathrm{Var}(\mu_{251}|V) &= \mathbb{E}\left((X(\theta))(X(\theta))^\top\right) - \mathbb{E}\left(X(\theta)\right)^2 \\ &\approx \frac{1}{M}\sum_{i=1}^{M}\left((X(\theta_{(i)}))(X(\theta_{(i)}))^\top\right) - \left((\bar{X})(\bar{X})^\top\right)\end{aligned}$$

which can also be easily implemented in R to find that the variance of the $251^{th}$ true location estimate is:

$$\mathrm{Var}(\mu_{251}|V) \approx \begin{pmatrix} 0.0009 & 0.0001 \\ 0.0001 & 0.0007 \end{pmatrix}$$

# Appendix

## Source Code

```
knitr::opts_chunk$set(echo = TRUE)
# Install ggplot2 if you don't have it installed already
if (!requireNamespace("ggplot2", quietly = TRUE)) {
  install.packages("ggplot2")
}
# Load ggplot2
library(ggplot2)
knitr::opts_chunk$set(echo=TRUE)
if (!requireNamespace("dplyr", quietly = TRUE)) {
    install.packages("dplyr")
}
library(dplyr)
knitr::opts_chunk$set(echo=TRUE)
if (!requireNamespace("patchwork", quietly = TRUE)) {
    install.packages("patchwork")
}
library(patchwork) # for nice plot combining
knitr::opts_chunk$set(echo=TRUE)
if (!requireNamespace("broom", quietly = TRUE)) {
  install.packages("broom")
}
library(broom)
knitr::opts_chunk$set(echo=TRUE)
if(!requireNamespace("formatR", quietly = TRUE)) {
  install.packages("formatR")
}
library(formatR)
knitr::opts_chunk$set(echo = TRUE)
if (!requireNamespace("knitr", quietly = TRUE)) {
  install.packages("knitr")
}
library(knitr)
if (!requireNamespace("kableExtra", quietly = TRUE)) {
  install.packages("kableExtra")
}
library(kableExtra)
knitr::opts_chunk$set(echo=TRUE)
if (!requireNamespace("car", quietly = TRUE)) {
  install.packages("car")
}
library(car)
knitr::opts_chunk$set(echo=TRUE)
if (!requireNamespace("latex2exp", quietly = TRUE)) {
  install.packages("latex2exp")
}
library(latex2exp)
knitr::opts_chunk$set(echo=TRUE)
if (!requireNamespace("pracma", quietly = TRUE)) {
  install.packages("pracma")
}
```

```r
library(pracma)

options(show.signif.stars = FALSE)
rm (list=ls())

# loading the assignment data
load ("STA3043_Assignment1_2025.RData")

# retrieval of student observations
kwe_obs = Class.List$KMNMIC001
lun_obs = Class.List$MYNLUN004
x = kwe_obs
# plotting the observation onto a x-y plot
x_obs = x[,2]
y_obs = x[,3]
plot (x_obs, y_obs, type="l")
points(x_obs, y_obs, pch= 21)
arrows (head(x_obs, -1), head(y_obs, -1),
        tail(x_obs, -1), tail(y_obs, -1),
        length= 0.1, col= "hotpink")

x_obs = x[,2]
y_obs = x[,3]
df = data.frame(x = x_obs, y = y_obs)

ggplot(df, aes(x = x, y = y)) +
  geom_path(color = "black") +                           # line for the path
  geom_point(shape = 21, fill = "white", size = 3) +     # points at each location
  geom_segment(aes(xend = lead(x), yend = lead(y)),      # arrows for direction
               arrow = arrow(length = unit(0.2, "cm")),
               color = "hotpink",
               na.rm = TRUE) +
  theme_minimal() +
  labs(x = TeX("X"), y = TeX("Y"))
# computing s1 and s2
# getting the differences of points and isolating a vector for both change in
# x and change in y
diff_x = diff (x)
dx = diff_x[,2]
dy = diff_x[,3]
# need lagged difference of points
dx_LAG = c(0, head(dx, -1))
dy_LAG = c(0, head(dy, -1))
# so that
s1 = sum (dx*dx_LAG + dy*dy_LAG)
s2 = sum (dx*dy_LAG - dy*dx_LAG)

# exponent of function for posterior
expo_postv = function(theta){
  # combining the s1,s2 with trig functions
  x1 = s1*cos(theta) + s2*sin(theta)
  0.5*x1
}
```

```r
# vectorise function above to CORRECTLY handle the theta vector
vec_expo_postv = Vectorize (FUN = expo_postv, vectorize.args = "theta")

# theta's range
theta_range = seq (from = -pi, to = pi, length = 250)

# plotting log posterior
plot(theta_range, vec_expo_postv (theta_range), type = "l",
     xlab = expression (theta), ylab = "Log 'Posterior'", col = "red")

#plotting unnormalised posterior
plot(theta_range, exp(vec_expo_postv (theta_range)), type = "l",
     xlab = expression(theta), ylab = "Unnormalised 'Posterior'")

#plotting posterior
vpost_vals = exp(vec_expo_postv (theta_range))
vpost = vpost_vals / sum (vpost_vals)
plot(theta_range, vpost, type = "l",
     xlab = expression(theta), ylab = "Normalised 'Posterior'", col = "royalblue")


# data prep

log_post_vals <- vec_expo_postv(theta_range)
unnorm_post_vals <- exp(log_post_vals)

norm_post_vals <- unnorm_post_vals / sum(unnorm_post_vals)

# put into one dataframe for plotting
df <- data.frame(
  theta = theta_range,
  log_post = log_post_vals,
  unnorm_post = unnorm_post_vals,
  norm_post = norm_post_vals
)

# make three ggplots
p1 <- ggplot(df, aes(x = theta, y = log_post)) +
  geom_line(color = "red") +
  labs(x = TeX("$\\theta$"), y = TeX("Log Posterior")) +
  theme_minimal()

p2 <- ggplot(df, aes(x = theta, y = unnorm_post)) +
  geom_line(color = "darkgreen") +
  labs(x = TeX("$\\theta$"), y = TeX("Unnormalised Posterior")) +
  theme_minimal()

p3 <- ggplot(df, aes(x = theta, y = norm_post)) +
  geom_line(color = "royalblue") +
  labs(x = TeX("$\\theta$"), y = TeX("Normalised Posterior")) +
  theme_minimal()

# combine with patchwork (1 row, 3 cols)
```

```
#(p1 | p2 | p3)
# OR (3 rows, 1 col)
p1 / p2 / p3 # + plot_layout(heights = c(10, 10, 10)) # 1.5× the default height

# want value of C
# function (ln component, exponent) to maximise
expo = function (theta){
  # the previous work:
  x1 = s1*cos(theta) + s2*sin(theta)
  # candidate sampling distr
  x2 = cos(theta)
  0.5*x1 -3*x2
}

# vectorise the function to CORRECTLY deal with vector val of theta
vec_expo = Vectorize (FUN = expo, vectorize.args = "theta")

# tool to find maximum of exponent component
opt = optimise(vec_expo, interval = c(-pi,pi), maximum = TRUE)

# hence value of C the normalising constant
C = exp(opt$objective)

# Data for plotting
theta_range <- seq(-pi, pi, length.out = 250)
df_max <- data.frame(
  theta = theta_range,
  val = vec_expo(theta_range)
)

# Extract maximum location & value
theta_max <- opt$maximum
val_max <- opt$objective

# Plot
ggplot(df_max, aes(x = theta, y = val)) +
  geom_line(color = "steelblue", size = 1) +
  geom_vline(xintercept = theta_max, linetype = "dashed", color = "red") +
  geom_hline(yintercept = val_max, linetype = "dashed", color = "darkgreen") +
  annotate("text", x = theta_max, y = val_max,
           label = paste0("Max at ", round(theta_max,3),
                          "\nValue = ", round(val_max,3)),
           hjust = -0.5, vjust = 1, size = 4.5) +
  labs(
    title = "Maximisation of Truncated density",
    x = TeX("$\\theta$"),
    y = "Exponent Value"
  ) +
  ylim (-180,180) +
  theme_minimal(base_size = 13)
gamma = function(theta, normconstant=C){
  # the previous work:
  x1 = s1*cos(theta) + s2*sin(theta)
```

```r
  x2 = cos(theta)
  C = normconstant

  (exp(0.5*x1 -3*x2))/(C)
}

# vectorise the function to CORRECTLY deal with vector val of theta
vec_gamma = Vectorize (gamma, "theta")

# plot of gamma
plot (theta_range, vec_gamma (theta_range), type = "l",
      xlab = expression(theta), ylab = expression(gamma(v)),)
gamma_df <- data.frame(
  theta = theta_range,
  gamma_val = vec_gamma(theta_range)
)

ggplot(gamma_df, aes(x = theta, y = gamma_val)) +
  geom_line(color = "steelblue", size = 1) +
  labs(
    title = expression(paste("Plot of ", gamma(v))),
    x = expression(theta),
    y = expression(gamma(v))
  ) +
  theme_minimal(base_size = 13)

# since h(theta) isnt a known distribution we sample by AR too where
# our candidate is a uniform

theta_samps = function (n){
  out = numeric (0)
  while (length(out) < n){
    # want to sample the difference between length of samples and desired n
    m = n - length(out)
    # getting the proposed values using the candidate of uniform -pi,pi
    prop1 = runif (m, -pi, pi)
    # getting the random uniform 0,1 values
    unif1 = runif (m, 0, 1)
    # accept if unif1<gamma(prop1), note this is indeed gamma because
    # C = 2pi*e^3
    accept1 = unif1 <= exp(3*(cos(prop1)-1))
    # updating out
    out = c(out, prop1[accept1])
  }

  #return the samples from h(theta)
  out[1:n]
}

# function named ar to spit out draws BUT first
# need log of gamma to allow for log-scale acceptance since C is LARGE
ln_C = log(C)
expo_gamma = function(theta, normconstant=ln_C){
```

```r
  # the previous work:
  x1 = s1*cos(theta) + s2*sin(theta)
  x2 = cos(theta)
  A = normconstant

  0.5*x1 -3*x2 - A
}

# # vectorise the function to CORRECTLY deal with vector val of theta
# vec_expo_gamma = Vectorize (expo_gamma, "theta")

ar = function (m){
  draw = numeric(0)
  while (length(draw) < m){
    k = m - length(draw)
    # proposed thetas from candidate h(theta)
    prop2 = theta_samps (k)
    # random uniform generation from 0,1
    unif2 = runif (k)
    # accepted proposals condition
    logaccept = expo_gamma (prop2)
    accept2 = log (unif2) <= logaccept # vec_expo_gamma (prop2)
    # updating draw
    draw = c(draw, prop2[accept2])
  }

  # returning final posterior draws
  draw
}

# picking our number of samples to get back from ar()
n1 = 5000

samps = ar (n1)

# posterior draws
hist(samps, breaks= 50, freq= FALSE, ylab = "Posterior Draws")
lines(density(samps), col="green")
# theoretical
df$norm_post <- unnorm_post_vals / trapz(df$theta, unnorm_post_vals)
lines(df$theta, df$norm_post, col="red", lwd=2)
legend("topright", legend=c("AR Draw Density", "Theoretical Posterior"),
       col=c("green", "red"), lwd=2, bty="n")

# to get summary of posterior samples
post_summary <- summary(samps)

# convert to data frame for nicer formatting
post_summary_df <- as.data.frame(t(post_summary))  # transpose to get variables in rows

# display as a kable
kable(post_summary_df,
      caption = "Summary of Posterior Samples for Draws",
```

```
      digits = 4,
      format = "latex",
      row.names = FALSE,
      booktabs = TRUE)
# We do this by using Monte Carlo
# 1st find f(theta) which will be the conditional of estimate mu given V and
# theta

# fucntion to take in theta sample and data to set monte carllo for est of mu_251
f_theta = function (theta_samp, dataobs=x){
  # getting v_250 and mu_250
  mu_249 = dataobs[250,2:3]
  mu_250 = dataobs[251,2:3]

  # defining matrix M
  M = function (theta_samp){
    rbind ( cbind (cos(theta_samp), sin(theta_samp)),
          cbind (-sin(theta_samp), cos(theta_samp)) )
  }

  mu_250 + 0.5 * M(theta_samp) %*% (mu_250-mu_249)
}

# get expectation and variance by for loops
est_Exp = 0
sum1 = 0
for (i in 1:n1) {
  sum1 = sum1 + f_theta(samps[i])
  est_Exp = (1/n1) * (sum1)
}
print (est_Exp)

est_Var = 0
sum2 = 0
for (i in 1:n1) {
  sum2 = sum2 + (f_theta(samps[i])) %*% t(f_theta(samps[i]))
  est_Var = (1/n1) * (sum2) - (est_Exp) %*% t(est_Exp)
}
print (est_Var)
```