STA 602 Lab 7

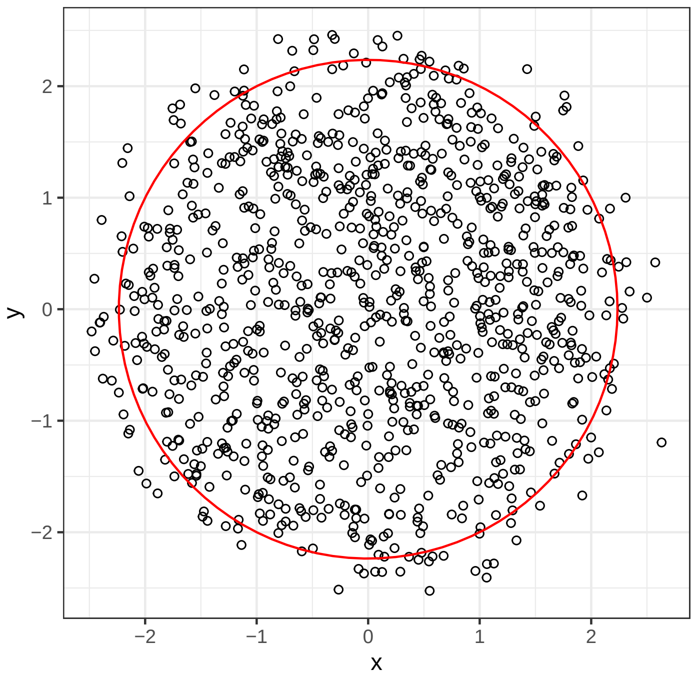
Yicheng Shen

31 October, 2022

---

0. Visualize the data generating process

```
n <- 1000
true_Rsquared <- 5
true_sigma <- 1.25
#
u <- runif(n, 0, true_Rsquared)  # de-noised squared distance of each point to (0,0)
r <- u + rnorm(n, sd = true_sigma)  # add normal noise
theta <- runif(n, 0, 2*pi)

#
ggplot2::ggplot() +
  geom_point(data = data.frame(x = sign(r)*sqrt(abs(r))*cos(theta), y = sign(r)*sqrt(abs(r))*sin(theta)
             aes(x = x, y = y), shape = 1) + geom_path(data = data.frame(R = true_Rsquared) %>%
                              plyr::ddply(~R, function(d){
                                data.frame(x = sqrt(d$R)*cos(seq(0, 2*pi, length.out = 100)),
                                           y = sqrt(d$R)*sin(seq(0, 2*pi, length.out = 100)))
                              }),
                        aes(x = x, y = y), alpha = 1, colour = "red") + coord_fixed()
```

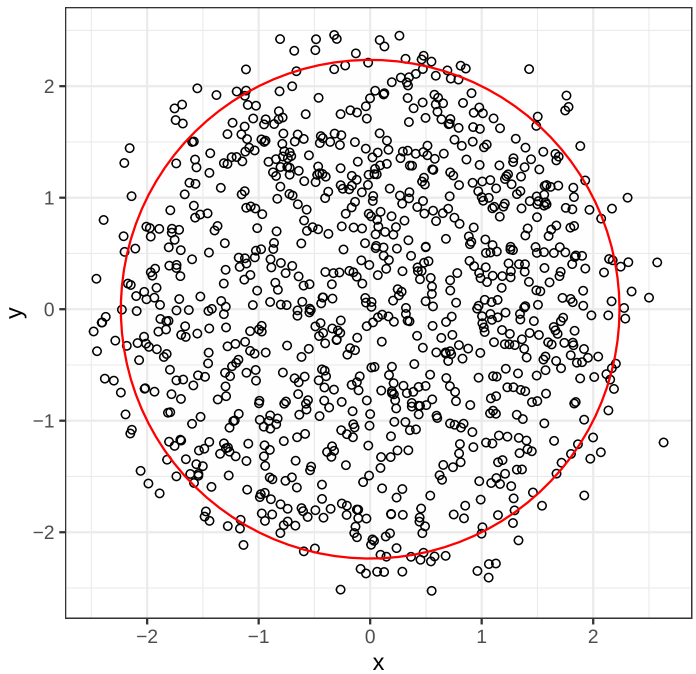


Exercise 1

```
# hyper-parameters
m <- 3
k <- 1
alpha <- 5/2
beta <- 5/2

# generate random sample from pareto(m,k)
rpareto <- function(m, k, trunc = NULL){
  p <- m*(1 - runif(1))^(-1/k)
  if(!is.null(trunc)){
    while(p > trunc){
      p <- m*(1 - runif(1))^(-1/k)
    }
  }
  return(p)
}

#
uni_pareto_gibbs <- function(S, r, m, k, alpha, beta, burn_in = min(1000, S / 2), thin = 5){
  # Reparametrize X matrix to squared radius values
  Rsq <- r  # squared distance
  n <- length(Rsq)  # sample size
  R <- rep(1, S)  # to save the S gibbs samples for R^2, also intialized it
  U <- matrix(0, nrow = S, ncol = n)  # to save the S gibbs samples for u_1, ..., u_n
  U[1, ] <- runif(n, 0, R)  # initialize the first sample for u_1, ..., u_n
  sigma <- rep(1, S)  # to save the S gibbs samples for sigma (not sigma2!)
  #
```

```
  U_curr <- U[1, ]  # save the updated sample for u_1, ..., u_n
  R_curr <- R[1]  # save the updated sample for R^2
  sigma_curr <- sigma[1]  # save the updated sample for sigma (not sigma2!)

  for(s in 1:S){
    # Sample from full conditional of the inner radius
    R_curr <- rpareto(max(c(U_curr, m)), k + n)
    R[s] <- R_curr
    # Sample from full conditional of U values
    U_curr <- truncnorm::rtruncnorm(n, a = 0, b = R_curr, mean = Rsq, sd = sigma_curr)
    U[s, ] <- U_curr
    # Sample from full conditional of sigma
    sigma_curr <- sqrt(1 / rgamma(1, n / 2 + alpha, 1/2 * sum((Rsq-U_curr)^2) + beta ))
    sigma[s] <- sigma_curr
  }
  return(list(R = R[seq(burn_in, S, by = thin)],
              U = U[seq(burn_in, S, by = thin),],
              sigma = sigma[seq(burn_in, S, by = thin)]))
}
#
gibbs_samps <- uni_pareto_gibbs(S = 100000, r, m, k, alpha, beta, burn_in=2000)
```

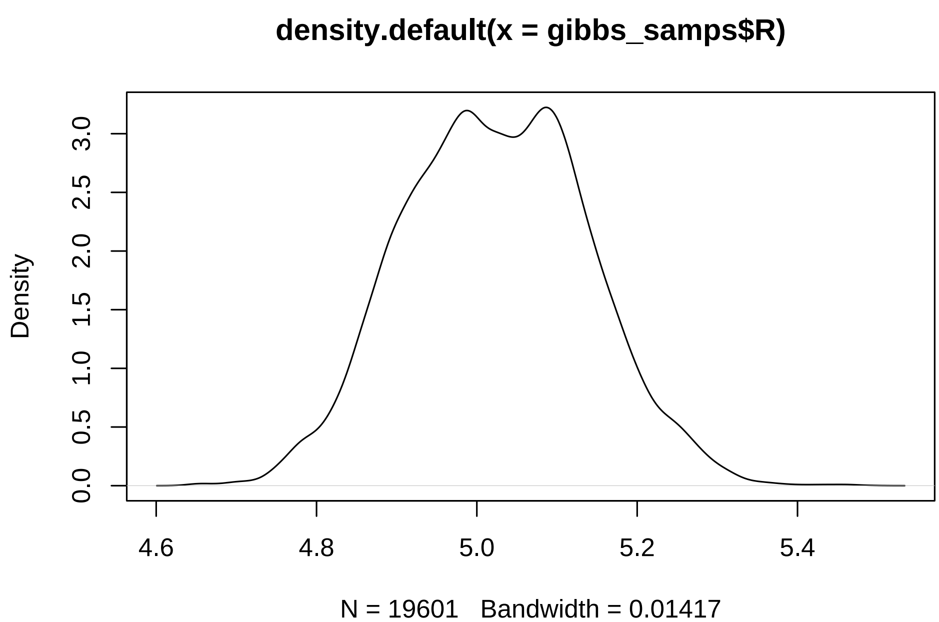
```
ggplot2::ggplot() +
  geom_point(data = data.frame(x = sign(r)*sqrt(abs(r))*cos(theta), y = sign(r)*sqrt(abs(r))*sin(theta)
             aes(x = x, y = y), shape = 1) +
  geom_path(data = data.frame(R = gibbs_samps$R) %>%
                              plyr::ddply(~R, function(d){
                                data.frame(x = sqrt(d$R)*cos(seq(0, 2*pi, length.out = 100)),
                                           y = sqrt(d$R)*sin(seq(0, 2*pi, length.out = 100)))
                              }),
                       aes(x = x, y = y), alpha = 0.005, colour = "blue") +
  geom_path(data = data.frame(R = true_Rsquared) %>%
                              plyr::ddply(~(R), function(d){
                                data.frame(x = sqrt(d$R)*cos(seq(0, 2*pi, length.out = 100)),
                                           y = sqrt(d$R)*sin(seq(0, 2*pi, length.out = 100)))
                              }),
                       aes(x = x, y = y), alpha = 1, colour = "red") +
  coord_fixed()
```



Exercise 2 & 3

Considering the true R^2 is 5 and true σ^2 is 1.5625, both marginal posterior densities are pretty reasonable.

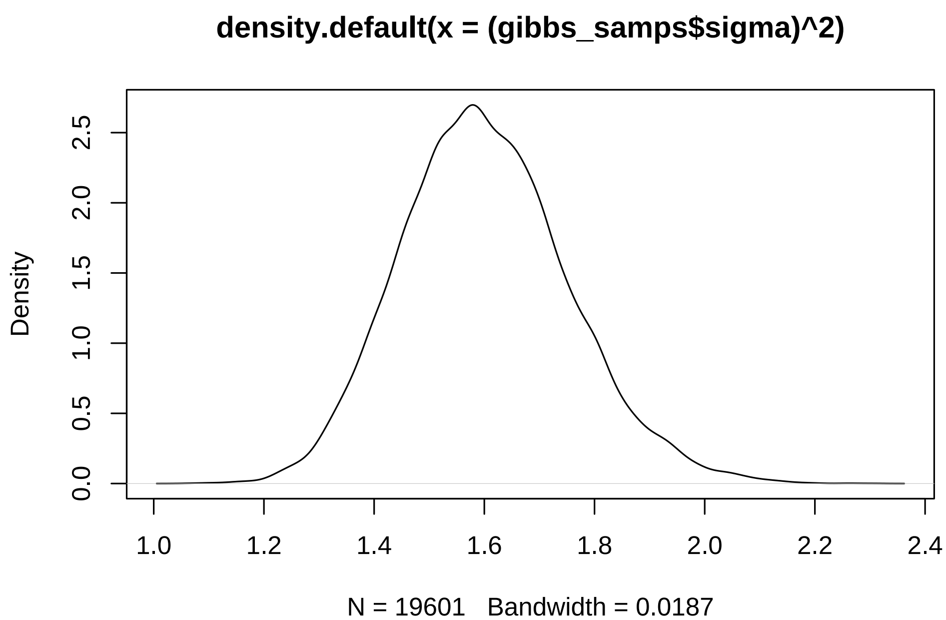
```
plot(density(gibbs_samps$R))
```



```r
true_Rsquared
```

```
## [1] 5
```

```r
plot(density((gibbs_samps$sigma)^2))
```



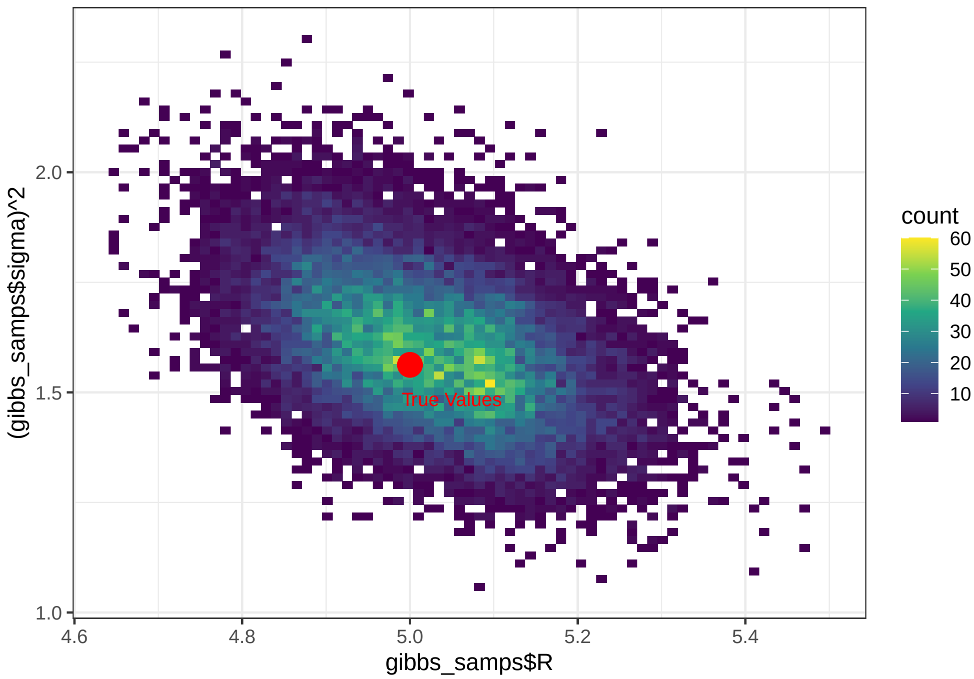

```r
true_sigma^2
```

```
## [1] 1.5625
```

Exercise 4

The true values of both parameters are at the center of the contour plot.

```r
ggplot() +
  geom_bin2d(aes(x=gibbs_samps$R, y=(gibbs_samps$sigma)^2), bins = 70) +
  geom_point(aes(x=true_Rsquared, y=true_sigma^2), size = 5, color = "red")  +
    annotate(
    "text", label = "True Values",
    x = true_Rsquared*1.01, y = true_sigma^2*0.95, size = 3, colour = "red"
  ) +
  scale_fill_continuous(type = "viridis")
```


True Values
count
60
50
40
30
20
10
2.0
1.5
1.0
(gibbs_samps$sigma)^2
4.6
4.8
5.0
5.2
5.4
gibbs_samps$R