BDA - Assignment 4

Anonymous

Exercise 1

a)

Prior distribution $\alpha \sim N(\mu_\alpha, \sigma_\alpha^2) = N(0, 2^2)$

Prior distribution $\beta \sim N(\mu_\beta, \sigma_\beta^2) = N(10, 10^2)$

The correlation $corr(\alpha, \beta) = \rho = 0.6$

Mean of the bivariate normal distribution: $\begin{pmatrix} \mu_\alpha \\ \mu_\beta \end{pmatrix} = \begin{pmatrix} 0 \\ 10 \end{pmatrix}$

Covariance of the bivariate normal distribution: $\begin{pmatrix} \sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta \\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{pmatrix} = \begin{pmatrix} 2^2 & 0.6 * 2 * 10 \\ 0.6 * 2 * 10 & 10^2 \end{pmatrix}$

b)

```
# import the datasets
data("bioassay_posterior")
data("bioassay")
#head(bioassay_posterior)

# implement function for calculating MCSE of mean
mcse_mean <- function(data) {
  sqrt(var(data)/length(data))
}

draws = length(bioassay_posterior$alpha)
draws
```

```
## [1] 4000
```

```
#means
mean(bioassay_posterior$alpha)
```

```
## [1] 0.9852263
```

```
mean(bioassay_posterior$beta)
```

```
## [1] 10.59648
```

```
mcse_mean(bioassay_posterior$alpha)
```

```
## [1] 0.01482435
```

```
mcse_mean(bioassay_posterior$beta)
```

```
## [1] 0.07560016
```

```
#quantiles for alpha
quantile(bioassay_posterior$alpha, c(0.05, 0.95))
```

```
##         5%        95%
## -0.4675914  2.6102028
```

```
mcse_quantile(bioassay_posterior$alpha, 0.05)[1,1]
```

```
## [1] 0.02600412
```

```
mcse_quantile(bioassay_posterior$alpha, 0.95)[1,1]
```

```
## [1] 0.04206342
```

```
#quantiles for beta
quantile(bioassay_posterior$beta, c(0.05, 0.95))
```

```
##        5%       95%
##  3.991403 19.340365
```

```
mcse_quantile(bioassay_posterior$beta, 0.05)[1,1]
```

```
## [1] 0.07043125
```

```
mcse_quantile(bioassay_posterior$beta, 0.95)[1,1]
```

```
## [1] 0.2412129
```

The Monte Carlo Standard Error (MCSE) is an approximation of how inaccurate the results are. Intuitively I would say, that for example a MCSE of 0.15 means, that the result could be off by 0.15. In this case, the first number after the decimal point could be off by 1 or even 2, so in most applications it would not make sense to report that decimal in the report.

The posterior distribution alpha has mean of 1.0, with 5% and 95% quantiles -0.5 and 2.6. The MCSEs are 0.015, 0.026, and 0.042.

The posterior distribution beta has mean of 10.6, with 5% and 95% quantiles -4.0 and 19. The MCSEs are 0.076, 0.070, and 0.24.

c)

Log ratios are sometimes better than regular ratios, because they allow expressing a lot wider range of numbers. Computers only have limited accuracy, so for example if a distribution has a density that is closer to 0 than what the computer is able to represent, that is effectively reduced to 0. With logarithmic scale the expressive range can be vastly improved, at a cost of (absolute) accuracy especially on larger numbers, and the inability to express negative numbers. On statistics, typically the number of correct digits is typically more important than absolute error.

```
logit <- function(x) {log(x/(1-x))}

log_importance_weights <- function(alpha, beta) {
  bioassaylp(alpha, beta, bioassay$x, bioassay$y, bioassay$n)
}
```

d)

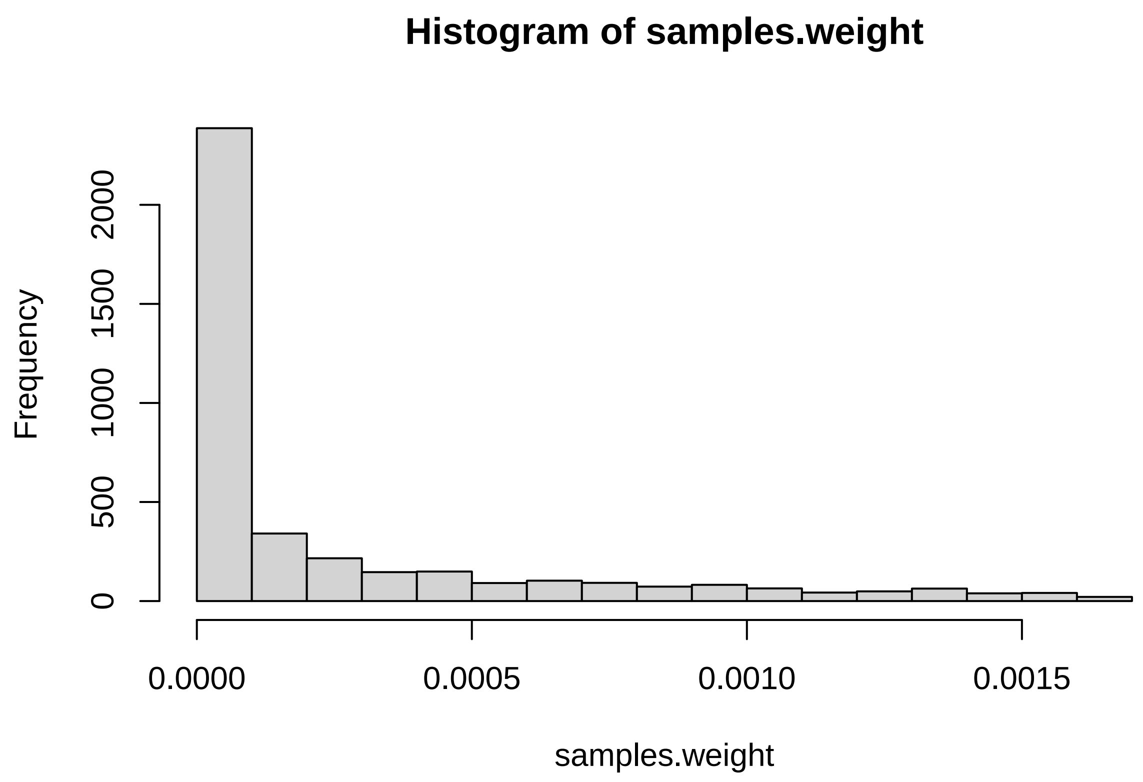
Now we implement a function, that gets rid of the log scale and also normalizes the weights. Getting rid of log scale is necessary for normalization, and also allows us humans to read the data more easily. Normalizing the weights allows us to apply them to the data without the need to divide by the sum of weights, as is done in h).

```
normalized_importance_weights <- function(alpha, beta) {
  log_values = bioassaylp(alpha, beta, bioassay$x, bioassay$y, bioassay$n)
  values = exp(log_values)
  values/sum(values)
}
```

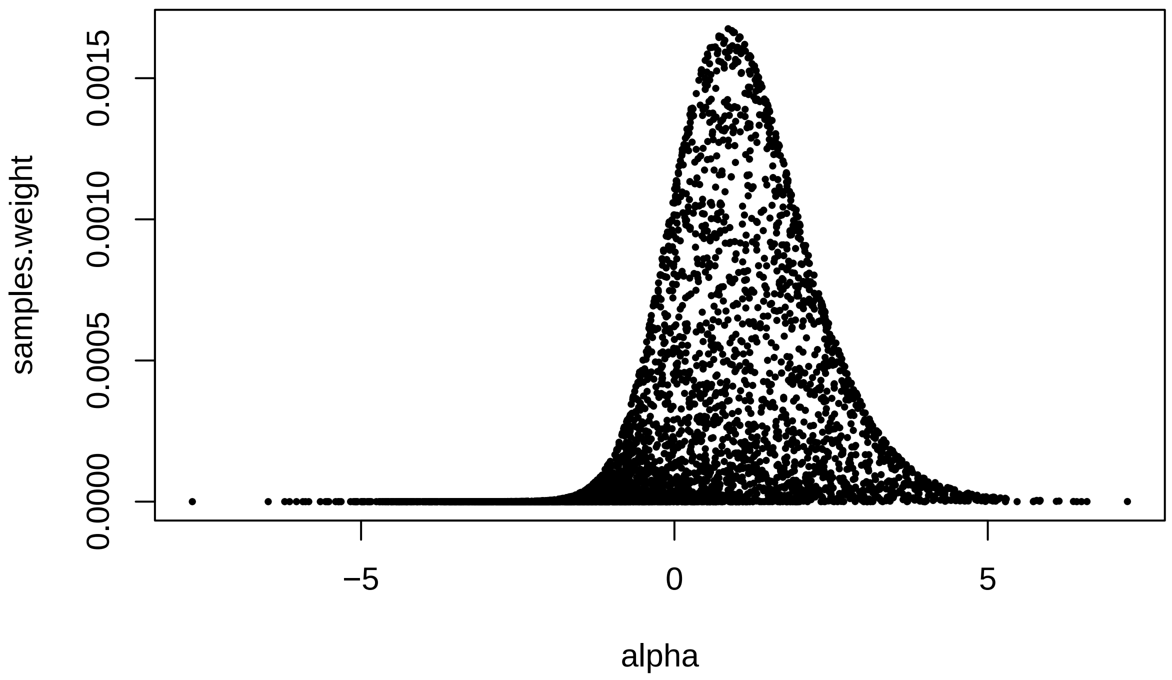
e)

Now we plot a histogram of the normalized weights, using a sample of 4000 observed data points. We can see, that most of the values are small, or in other words, there are few large weights. The largest weights have the greatest effect on the distribution's mean and variance, but are also the rarest to obtain using our sampling method. I also plotted alpha against the weights.

```
# add data from a)
mu_alpha_prior = 0
sigma_alpha_prior = 2
mu_beta_prior = 10
sigma_beta_prior = 10
correlation_prior = 0.6

samples = rmvnorm(4000, c(mu_alpha_prior, mu_beta_prior), matrix(data=c(sigma_alpha_prior^2, sigma_alpha

samples.weight = normalized_importance_weights(samples[,1],samples[,2])

hist(samples.weight)
```



```r
df <- data.frame(samples[,1],samples.weight)
plot(df, pch=16, col='black', cex=0.5, xlab="alpha")
```



f)

```r
S_eff <- function(alpha, beta){
  weights = normalized_importance_weights(alpha, beta)
  1/sum(weights^2)
}

round(S_eff(samples[,1],samples[,2]),3)
```

```
## [1] 1142.525
```

g)

The effective sample size of importance sampling is an estimate of how many samples we would have to get from the "real" distribution to get roughly as good results. As I mentioned before, because it is less likely to hit the more significant values of the distribution, we need more samples to obtain reliable data about the distribution if we use importance sampling. For example, in this case we would need only roughly 25% of the amount of data if we could sample the real distribution directly.

Having no prior reference, the histogram on e) seems pretty bad, but I guess it is not too bad. There is still a notable volume of samples on the higher end of the distribution, which I suppose give us enough information, that we only need 4x the amount of samples. From just the histogram, I would have guessed something more along the lines of 10x. This is a good example of why you should always calculate what sample size you need, even if you think you can just wing it.

As the second graph from e) shows, most of the large values are near the mean, so the effect of us getting less high-weight samples is somewhat mitigated.

h)

```r
posterior_mean <- function(alpha, beta){
  weights = normalized_importance_weights(alpha, beta)
  c(sum(weights*alpha), sum(weights*beta))
}

mcse_mean_cvar <- function(sample_size, variance) {
  sqrt(variance/sample_size)
}

s_eff = S_eff(samples[,1],samples[,2])
weights = normalized_importance_weights(samples[,1],samples[,2])
variance_alpha = sum(weights*samples[,1]^2) - sum(weights*samples[,1])^2
mcse_mean_cvar(s_eff, variance_alpha)
```

```
## [1] 0.0273056
```

```r
variance_beta = sum(weights*samples[,2]^2) - sum(weights*samples[,2])^2
mcse_mean_cvar(s_eff, variance_beta)
```

```
## [1] 0.1351135
```

```r
round(posterior_mean(samples[,1],samples[,2]),3)
```

```
## [1]  0.975 10.575
```

The posterior mean of alpha is 0.9 and posterior mean of beta is 10. The MCSEs are 0.026 and 0.13.