This report is automatically generated with the R package **knitr** (version `1.24`) .

````
---
title: STAT 457 Homework 05
author: Martha Eichlersmith
date: 2019-12-03
output:
  pdf_document:
    fig_caption: yes
header-includes:
  - \usepackage{xcolor}
  - \usepackage{mathtools}
  - \usepackage{amsbsy} #bold in mathmode
  - \usepackage{nicefrac} # for nice fracs
  - \usepackage{booktabs}
  - \usepackage{geometry}
  - \usepackage{caption} #to remove automatic table name and number - \captionsetup[table]{labelformat=empty}, put code under ---
geometry: "left=1.75cm,right=1.75cm,top=1.5cm,bottom=2cm"

---

\captionsetup[table]{labelformat=empty}
```{r setup, echo=FALSE, results="hide", warning=FALSE, message=FALSE}
rm(list=ls()) ### To clear namespace
library(ggplot2) #ggplot
library(readr) #import CSV
library(gridExtra) #organize plots
library(grid) #organize plots
library(latex2exp) #latex in ggplot titles
library(matlib) #A = matrix, inv(A) = A^{-1}
library(numDeriv) #calculate numerical first and second order derivatives
library(gtable) #for tablegrob functions
library(dplyr) #for piping
library(MCMCpack) #for dirichelt
knitr::opts_chunk$set(echo=FALSE, fig.width = 10, fig.height = 4)
#knitr::opts_chunk$set(eval=FALSE)
decimal <- function(x, k) trimws(format(round(x, k), nsmall=k))
dec <- 5
#knitr::opts_chunk$set(echo=FALSE) #using knitr for this option but don't have to load
```


\newpage
### Problem 3a
For the genetic linkage model: use importance sampling to obtain the posterior mean for data $Y = (125, 18, 20, 34)$.  Use the matching normal distribut

```{r p3ab_function}
func_Like.Y <- function(x, yvec){
  (x + 2)^(yvec[1]) * (1 - x)^(yvec[2]+yvec[3]) *  (x)^( yvec[4])
}


func_ifoutside <- function(x){
  y <- 0
  if (x>1 | x<0) {y=0}
  else {y=x}
  return(y)
}


func_g <- function(w){
  y <- 0
  if (w==0) {y =0}
  else {y =func_Like.Y(w, Y.vec)/w}
  return(y)
}


func_w.star <- function(it, Y.vec, N.mu, N.sig){
w <- rnorm(it, N.mu, N.sig)
#randomly draw w_i's
w <- sapply(w, func_ifoutside)
#for if w not in [0, 1]
g_w <- sapply(w, func_g)
#function g(w_i)'s where g(x) =  likelihood / w_i
w.star <- g_w / sum(g_w)
#w.star = weights = g(x)/sum(g(x))
it.vec <- c(rep(it, it))
df <- data.frame("w"=w, "w.star"=w.star, "it"=it.vec)
return(df)
}

func_compare <- function(w, w.star, N.mu, N.sig){
  it <- length(w)
  post.mean <- sum(w.star*w)
  post.sd <- sqrt(sum(w.star*(w - post.mean)^2))
  compare <- data.frame("IS"=c(post.mean, post.sd), "Norm Apprx" =c(N.mu, N.sig), "Diff"=c(post.mean-N.mu, post.sd-N.sig))
  rownames(compare) <- c("mean", "sd")
  return(compare)
}

func_plotsAB <- function(it.vec, Y.vec, N.mu, N.sig){

df1 <- func_w.star(it.vec[1], Y.vec, N.mu, N.sig)
df2 <- func_w.star(it.vec[2], Y.vec, N.mu, N.sig)
````
````

```
df3 <- func_w.star(it.vec[3], Y.vec, N.mu, N.sig)

compare1 <- func_compare(df1$w, df1$w.star, N.mu, N.sig)
compare2 <- func_compare(df2$w, df2$w.star, N.mu, N.sig)
compare3 <- func_compare(df3$w, df3$w.star, N.mu, N.sig)

table1 <- tableGrob(round(compare1, dec))
table2 <- tableGrob(round(compare2, dec))
table3 <- tableGrob(round(compare3, dec))

big.df <- rbind(df1, df2, df3)

dat_text <- data.frame(label =c(
                paste("w.star sd=", round(sd(df1$w.star), 10)),
                paste("w.star sd=", round(sd(df2$w.star), 10)),
                paste("w.star sd=", round(sd(df3$w.star), 10))
                ),
                       Iteration = it.vec)

df_w.star <- data.frame("w.star"=big.df$w.star, "Iteration"=big.df$it )
print.Y.vec <- paste(Y.vec, collapse=",")
name <- paste("Important Sampling Weights for Y=(", print.Y.vec, ")")

plot <-  ggplot(df_w.star, aes(w.star))+geom_histogram(aes(y=..density..), color="black", alpha=0.5)+
   facet_wrap(~Iteration, ncol=3)+
  ggtitle(paste(name))+
  theme( axis.text.x=element_blank()
        ,axis.text.y=element_blank()
        )+
  geom_text(data=dat_text, mapping=aes(x=Inf, y = Inf, label=label), hjust=1.5, vjust=2, size=4)+
  xlab(TeX("$w^*$=weights"))

gs <- list(plot, table1, table2, table3)
grid.arrange(grobs=gs,
              widths = c(1, 1, 1),
              heights =2:1,
              layout_matrix = rbind( c(1, 1, 1),
                                     c(2, 3, 4)
 ))
}
```

```{r p3a, fig.height=5, warning=FALSE, message=FALSE}
set.seed(060301)
Y.vec <- c(125, 18, 20, 34)
N.mu  <- 0.62682
N.sig <- 0.05382
it.vec <- c(1e04, 1e05, 1e06)

func_plotsAB(it.vec, Y.vec, N.mu, N.sig)
```

Problem 3b
Repeat (a) for the data $Y = (14, 0, 1, 5)$. Normal Approximation for $Y = (125, 18, 20, 34) \sim \mathcal{N}(\mu = 0.90344, \ \sigma=0.09348)$

```{r p3b, fig.height=5, warning=FALSE, message=FALSE}
set.seed(060302)
Y.vec <- c(14,0,1,5)
N.mu  <- 0.90344
N.sig <- 0.09348
it.vec <- c(1e04, 1e05, 1e06)

func_plotsAB(it.vec, Y.vec, N.mu, N.sig)
```
Using a normal important sampling function to estiamte the posterior mean is closer for $Y=(125, 18, 20, 34)$ normal approximation than $Y=(14, 0, 1, 5)

\newpage
Problem 3c
Repeat (a) and (b) with a Uniform[0, 1] importance function.

```{r p3C_function}
func_Like.Y <- function(x, yvec){
  (x + 2)^(yvec[1]) * (1 - x)^(yvec[2]+yvec[3]) *  (x)^( yvec[4])
}


func_ifoutside <- function(x){
  y <- 0
  if (x>1 | x<0) {y=0}
  else {y=x}
  return(y)
}


func_g <- function(w){
  y <- 0
  if (w==0) {y =0}
  else {y =func_Like.Y(w, Y.vec)/w}
  return(y)
}


func_w.star <- function(it, Y.vec){
w <- runif(it, 0, 1)
#randomly draw w_i's
w <- sapply(w, func_ifoutside)
#for if w not in [0, 1]
g_w <- sapply(w, func_g)
```

```
#function g(w_i)'s where g(x) =  likelihood / w_i
w.star <- g_w / sum(g_w)
#w.star = weights = g(x)/sum(g(x))
it.vec <- c(rep(it, it))
df <- data.frame("w"=w, "w.star"=w.star, "it"=it.vec)
return(df)
}

func_compare <- function(w, w.star, N.mu, N.sig){
  it <- length(w)
  post.mean<- sum(w.star*w)
  post.sd <- sqrt(sum(w.star*(w - post.mean)^2))
  compare <- data.frame("IS"=c(post.mean, post.sd), "Norm Apprx" =c(N.mu, N.sig), "Diff"=c(post.mean-N.mu, post.sd-N.sig))
  rownames(compare) <- c("mean", "sd")
  return(compare)
}

func_plotsC <- function(it.vec, Y.vec){

df1 <- func_w.star(it.vec[1], Y.vec)
df2 <- func_w.star(it.vec[2], Y.vec)
df3 <- func_w.star(it.vec[3], Y.vec)

compare1 <- func_compare(df1$w, df1$w.star, N.mu, N.sig)
compare2 <- func_compare(df2$w, df2$w.star, N.mu, N.sig)
compare3 <- func_compare(df3$w, df3$w.star, N.mu, N.sig)

table1 <- tableGrob(round(compare1, dec))
table2 <- tableGrob(round(compare2, dec))
table3 <- tableGrob(round(compare3, dec))

big.df <- rbind(df1, df2, df3)

dat_text <- data.frame(label =c(
                paste("w.star sd=", round(sd(df1$w.star), 10)),
                paste("w.star sd=", round(sd(df2$w.star), 10)),
                paste("w.star sd=", round(sd(df3$w.star), 10))
                ),
                       Iteration = it.vec)

df_w.star <- data.frame("w.star"=big.df$w.star, "Iteration"=big.df$it )
print.Y.vec <- paste(Y.vec, collapse=",")
name <- paste("Important Sampling Weights for Y=(", print.Y.vec, ")")

plot <-  ggplot(df_w.star, aes(w.star))+geom_histogram(aes(y=..density..), color="black", alpha=0.5)+
   facet_wrap(~Iteration, ncol=3)+
  ggtitle(paste(name))+
  theme( axis.text.x=element_blank()
        ,axis.text.y=element_blank()
        )+
  geom_text(data=dat_text, mapping=aes(x=Inf, y = Inf, label=label), hjust=1.5, vjust=2, size=4)+
  xlab(TeX("$w^*$=weights"))

gs <- list(plot, table1, table2, table3)
grid.arrange(grobs=gs,
              widths = c(1, 1, 1),
              heights =2:1,
              layout_matrix = rbind( c(1, 1, 1),
                                     c(2, 3, 4)
 ))
}
```

```{r p3c, fig.height=5, warning=FALSE, message=FALSE}
it.vec <- c(1e04, 1e05, 1e06)

set.seed(0603031)
Y.vec <- c(125, 18, 20, 34)
func_plotsC(it.vec, Y.vec)

set.seed(0603032)
Y.vec <- c(14, 0, 1,5)
func_plotsC(it.vec, Y.vec)
```
Note that the histograms of the weights, w^*, are very similar for both sets of data. This is because the importance function is not dependent on the

\newpage
Problem 4

Problem 4a
Solve the following problem posted by the Reverend Thomas Bayes in his essay "Essay Towards Solving a Problem in the Doctrine of Chances," which was pub
Given the number of times in which an unknown event has happened and failed: *Required* the chance that the probability of its happening in a single t
In other words, if the number of the successful happenings of the event is p and the failures q, and if the named "degrees" of the probability are $


```{r p4a-functions}
func_Like.Y <- function(x){ x^1 * (1 - x)^4 }

func_plots <- function(big.df, text.df, name){
df_w.star <- data.frame("w.star"=big.df$w.star, "Iteration"=big.df$it )
name <- paste(name, "Important Sampling")
plot <-  ggplot(df_w.star, aes(w.star))+geom_histogram(aes(y=..density..), color="black", alpha=0.5)+
   facet_wrap(~Iteration, ncol=3)+
  ggtitle(paste(name))+
  theme( axis.text.x=element_blank()
        ,axis.text.y=element_blank()
        )+
  geom_text(data=text.df, mapping=aes(x=Inf, y = Inf, label=label), hjust=1.5, vjust=2, size=4)+
```

```
  xlab(TeX("$w^*$=weights"))
plot
}
```
```

```{r p4a-Uniform, warning=FALSE, message=FALSE, fig.height=3}
func_J.Uniform <- function(it, b, f){
x <- runif(it, b, f)
w <- x / dunif(x, b, f)
J <- sum((w * func_Like.Y(x))/sum(w))
return(c(J))
}

func_w.star.Uniform <- function(it, b, f){
w <- runif(it, b, f)
g_w <- func_Like.Y(w) / w
w.star <- g_w / sum(g_w)
it.vec <- c(rep(it, it))
df <- data.frame("w"=w, "w.star"=w.star, "it"=it.vec)
return(df)
}

func_big.df.Uniform <- function(it.vec, b, f){
df1 <- func_w.star.Uniform(it.vec[1], b, f)
df2 <- func_w.star.Uniform(it.vec[2], b, f)
df3 <- func_w.star.Uniform(it.vec[3], b, f)
big.df <- rbind(df1, df2, df3)
}

b <- 0.7
f <- 0.9
it.vec <- c(1e04, 1e05, 1e06)
set.seed(0604011)
IS.Uniform.J <- mapply(func_J.Uniform, it.vec, b, f)

Uniform.df <- func_big.df.Uniform(it.vec, b, f)
Uniform.text.df <- data.frame(label =c(
                paste("J=", round(IS.Uniform.J[1], 10)),
                paste("J=", round(IS.Uniform.J[2], 10)),
                paste("J=", round(IS.Uniform.J[3], 10))
                ),
                       Iteration = it.vec)
func_plots(Uniform.df, Uniform.text.df, "Uniform")

```

```{r p4a-Beta, warning=FALSE, message=FALSE, fig.height=3}
func_J.Beta <- function(it, b, f){
x <- rbeta(it, b+1, f+1)
w <- x / dbeta(x, b+1, f+1)
J <- sum((w * func_Like.Y(x))/sum(w))
return(c(J))
}

func_w.star.Beta <- function(it, b, f){
w <- rbeta(it, b+1, f+1)
g_w <- func_Like.Y(w) / w
w.star <- g_w / sum(g_w)
it.vec <- c(rep(it, it))
df <- data.frame("w"=w, "w.star"=w.star, "it"=it.vec)
return(df)
}

func_big.df.Beta <- function(it.vec, b, f){
df1 <- func_w.star.Beta(it.vec[1], b, f)
df2 <- func_w.star.Beta(it.vec[2], b, f)
df3 <- func_w.star.Beta(it.vec[3], b, f)
big.df <- rbind(df1, df2, df3)
}

b <- 0.7
f <- 0.9
it.vec <- c(1e04, 1e05, 1e06)
set.seed(0604012)
IS.Beta.J <- mapply(func_J.Beta, it.vec, b, f)

Beta.df <- func_big.df.Beta(it.vec, b, f)
Beta.text.df <- data.frame(label =c(
                paste("J1=", round(IS.Beta.J[1], 10)),
                paste("J1=", round(IS.Beta.J[2], 10)),
                paste("J1=", round(IS.Beta.J[3], 10))
                ),
                       Iteration = it.vec)
func_plots(Beta.df, Beta.text.df, "Beta")
```

```{r p4a-Normal, warning=FALSE, message=FALSE, fig.height=3}
func_J.Normal <- function(it, b, f){
alpha <- b + 1
beta <- f + 1
mean <- alpha / (alpha + beta)
var <- (alpha*beta) / ( (alpha + beta)^2 * (alpha + beta + 1) )
x <- rnorm(it, mean, sqrt(var))
w <- x / dnorm(x, mean, sqrt(var))
J <- sum((w * func_Like.Y(x))/sum(w))
return(c(J))
}
```
```

```
func_w.star.Normal <- function(it, b, f){
alpha <- b + 1
beta <- f + 1
mean <- alpha / (alpha + beta)
var <- (alpha*beta) / ( (alpha + beta)^2 * (alpha + beta + 1) )
w <- rnorm(it, mean, sqrt(var))
g_w <- func_Like.Y(w) / w
w.star <- g_w / sum(g_w)
it.vec <- c(rep(it, it))
df <- data.frame("w"=w, "w.star"=w.star, "it"=it.vec)
return(df)
}

func_big.df.Normal <- function(it.vec, b, f){
df1 <- func_w.star.Normal(it.vec[1], b, f)
df2 <- func_w.star.Normal(it.vec[2], b, f)
df3 <- func_w.star.Normal(it.vec[3], b, f)
big.df <- rbind(df1, df2, df3)
}

b <- 0.7
f <- 0.9
it.vec <- c(1e04, 1e05, 1e06)
set.seed(0604012)
IS.Normal.J <- mapply(func_J.Normal, it.vec, b, f)

Normal.df <- func_big.df.Normal(it.vec, b, f)
Normal.text.df <- data.frame(label =c(
                paste("J1=", round(IS.Normal.J[1], 10)),
                paste("J1=", round(IS.Normal.J[2], 10)),
                paste("J1=", round(IS.Normal.J[3], 10))
                ),
                       Iteration = it.vec)
func_plots(Normal.df, Normal.text.df, "Normal")
```

```
\newpage
### Problem 4b
Repeat the calculation using numerical integration.  Compare the results of (a) and (b).

```{r p4b}
Integrate.J <- integrate(func_Like.Y, lower=b, upper=f)$value / integrate(func_Like.Y, lower=0, upper=1)$value

iteration <- c("N/A", it.vec, it.vec, it.vec)
J.vec <- c(Integrate.J, IS.Uniform.J, IS.Beta.J, IS.Normal.J)
Integrate.J.vec <- rep(Integrate.J, length(J.vec))
diff.J <- J.vec - Integrate.J.vec

result.4b <- rbind(iteration, round(J.vec, 5), round(diff.J, 5))
rownames(result.4b) <- c("It", "J", "J1-Intg")
```
```{r p4b-table}
knitr::kable(result.4b, booktabs=T, 'latex') %>%
  kableExtra::kable_styling(latex_options="hold_position" ) %>% #hold table in place
  kableExtra::add_header_above(c(" "=1, "Integration"=1, "IS - Uniform"=3, "IS - Beta"=3, "IS- Normal"=3)) #need to have a space in empty columns
```

Problem 6a
Under the likelihood $\theta^k (1 - \theta)^{n-x}$ and the Beta(a, b) prior (a and b known) compute the exact posterior mean. Repeat the calculat

$$
\begin{aligned}
        &\underline{\textbf{Exact}} \\
        p_E(\theta \mid Y) & \propto \text{Beta}(x + a, \ n - x + b) \quad \implies \quad
        \mu_{E} = \frac{ x + a}{ a + b + n}
        \\[2ex]
        &\underline{\textbf{2nd Laplace Approx}}
        \\
        \mu_{L} & = \frac{ \sigma^*}{\sigma^\dagger} \cdot \frac{ \exp \left\{ - nh^*(\theta^*) \right\} }
        { \exp \left\{ - nh^\dagger(\theta^\dagger) \right\} }
        \\[2ex]
        -nh^\dagger(\theta) & = \ell(\theta \mid Y) + \ln(p(\theta))
        \\
        & \quad \quad \arraycolsep=1pt \def\arraystretch{1.4}   \begin{array}{r c l}
                \ell(\theta \mid Y) & =& x \ln(\theta) + (n - x) \ln(\theta) \\
                \ln(p(\theta)) & =& (a - 1) \ln(\theta) + (b-1)\ln(1 - \theta)
        \end{array}
        \\
        &= \alpha_\dagger \ln(\theta) + \beta_\dagger \ln(1 - \theta)
        \quad \quad \arraycolsep=1pt \def\arraystretch{1.4} \begin{array}{r c l}
        \alpha_\dagger &=& x + a -1 \\
        \beta_\dagger &=& n - x + b - 1
        \end{array}
        \\[2ex]
                -nh^*(\theta) & = \ell(\theta \mid Y) + \ln(p(\theta)) + ln(g(\theta))
        \\
        & \quad \quad \begin{array}{r c l}
        \ell(\theta \mid Y) & =& x \ln(\theta) + (n - x) \ln(\theta) \\
        \ln(p(\theta)) & =& (a - 1) \ln(\theta) + (b-1)\ln(1 - \theta) \\
        ln(g(\theta)) &=& \ln(\theta)
        \end{array}
        \\
        &= \alpha_* \ln(\theta) + \beta_* \ln(1 - \theta)
        \quad \quad \arraycolsep=1pt \def\arraystretch{1.4} \begin{array}{r c l}
        \alpha_* &=& x + a \\
        \beta_* &=& n - x + b - 1
        \end{array}
```

```
        \\[2ex]
        \theta^{(\cdot)} &  =
        \arg\max_\theta (- nh^{(\cdot)}(\theta)) =
                \frac{ \partial - n h^{(\cdot)}(\theta) }{\partial \theta} =  \frac{\alpha_{(\cdot)}}{\theta} - \frac{\beta_{(\cdot)}}{1 - \theta} \stac
                \implies
                \theta^{(\cdot)} =
         \frac{ \alpha_{(\cdot)} }{ \alpha_{(\cdot)} + \beta_{(\cdot)} }
        \\[1ex]
        \sigma^{(\cdot)} & = \left[ \frac{ \partial^2 h^{(\cdot)}(\theta)}{\partial \theta^2} \Big|_{\theta^{(\cdot)}} \right]^{-1/2}
        = \left[ \frac{1}{n} \left(  \frac{ \alpha_{(\cdot)}}{ (\theta^{(\cdot)})^2 } + \frac{\beta_{(\cdot)}}{(1 - \theta^{(\cdot)})^2}\right) \right]^
\end{aligned}
$$

```{r p6a-function}
func_theta.dot <- function(alpha, beta){
  alpha / (alpha + beta)}

func_sigma.dot <- function(alpha, beta, theta, n){
  sqrt( ( (1/n) *(
               (alpha / theta^2) + (beta / ((1 - theta)^2))
              )
        )
  )
}

func_nh.dot<- function(alpha, beta, theta){
  theta^alpha * (1 - theta)^beta
}


func_mu.Laplace <- function(n, x, a, b){
alpha.dagger <- x + a - 1
beta.dagger <- n - x + b - 1
alpha.star <- x + a
beta.star <- n - x + b - 1

theta.dagger <- func_theta.dot(alpha.dagger, beta.dagger)
theta.star   <- func_theta.dot(alpha.star,   beta.star)
sigma.dagger <- func_sigma.dot(alpha.dagger, beta.dagger, theta.dagger, n)
sigma.star   <- func_sigma.dot(alpha.star,   beta.star,   theta.star,   n)

mu.Laplace <- (sigma.star/sigma.dagger)*(func_nh.dot(alpha.star, beta.star, theta.star)/func_nh.dot(alpha.dagger, beta.dagger, theta.dagger))
return(mu.Laplace)
}


func_mu.Exact <- function(n, x, a, b){
  mu.Exact <- (x + a) / (a + b +n)
  return(mu.Exact)
}

func_rel.error <- function(n, x, a, b){
  mu.Exact <- func_mu.Exact(n, x, a, b)
  mu.Laplace <- func_mu.Laplace(n, x, a, b)
  error <- mu.Laplace - mu.Exact
  relative.error <- error / mu.Exact
  return(relative.error)
}
```


```{r 6a-data1}
n1 <- 5
x1 <- 3
a1 <- 1/2
b1 <- 1/2

mu.Laplace1 <- func_mu.Laplace(n1, x1, a1, b1)
mu.Exact1 <- func_mu.Exact(n1, x1, a1, b1)
rel.error1 <- func_rel.error(n1, x1, a1, b1)
```

```{r 6a-data2}
n2 <- 25
x2 <- 15
a2 <- 1/2
b2 <- 1/2

mu.Laplace2 <- func_mu.Laplace(n2, x2, a2, b2)
mu.Exact2 <- func_mu.Exact(n2, x2, a2, b2)
rel.error2 <- func_rel.error(n2, x2, a2, b2)
```

```{r 6a-table}
one <- c( decimal(n1,0)
         ,round(x1,0)
         ,round(a1,1)
         ,round(b1,1)
         ,round(mu.Exact1,dec)
         ,round(mu.Laplace1,dec)
         ,round(rel.error1, dec)
        )
two <- c( decimal(n2,0)
         ,round(x2,0)
         ,round(a2,1)
         ,round(b2,1)
         ,round(mu.Exact2,dec)
         ,round(mu.Laplace2,dec)
```
```

```
        ,round(rel.error2, dec)
       )
table6a <- cbind(one, two)
rownames(table6a) <- c("Data.n", "Data.x", "Prior.a", "Prior.b", "Exact.mean", "Laplace.Mean", "Relative.Error")
colnames(table6a) <- c("Part.1", "Part.2")
table6a <- as.data.frame(table6a)
knitr::kable(table6a, 'markdown', align='rrr')
```
```
\newpage
## Problem 1
Recall the genetic linkage model of Section 4.1.

### Problem 1a
For the data $Y = (125, 18, 20, 34)$ implement the *EM* algorithm.  Use a flat prior on $\theta$.  Try starting your algorithm at $\theta = .1, .2, .3,

$$
\begin{aligned}
\text{Given } Y & = (y_1, y_2, y_3, y_4) \text{ with probabilities } \left( \frac{ \theta + 2}{4}, \frac{ 1 - \theta}{4},  \frac{ 1 - \theta}{4}, \frac{
\text{Say } Y & = (x, z, y_2, y_3, y_4) \text{ with probabilities } \left( \frac{1}{2}, \frac{ \theta}{4}, \frac{ 1 - \theta}{4},  \frac{ 1 - \theta}{4}
\\[2ex]
p(Z \mid Y, \theta) & \sim \text{Binomial}\left(x + z, \  \frac{p(z)}{p(x) + p(z)} \right) = \text{Binomial} \left(y_1, \ \frac{ \theta / 4}{(\theta + 2
\\[2ex]
p(\theta \mid Y, Z) &\sim
\left( \frac{1}{2} \right) ^{x}
\left( \frac{\theta}{4} \right)^{z}
\left( \frac{1 - \theta}{4} \right)^{y_2 + y_3}
\left( \frac{\theta}{4} \right)^{y_4}
\propto \theta^{z + x_5} (1 - \theta)^{y_2 + y_3}
\\[2ex]
&\underline{\textbf{E Step}: \text{to get Q function}}
\\
Q(\theta, \theta^i) & = \mathbb{E}_{Z \mid \theta^i} \left[ \log \left( p(\theta \mid Y, Z) \right) \right]
= \mathbb{E}_{Z \mid \theta^i} \Big[ (z + y_4)\log(\theta) + (y_2 + y_3) \log(1 - \theta) \mid \theta^i, Y \Big]
\\
& = (y_2 + y_3) \log(1 - \theta) + \left( \mathbb{E}_{Z \mid \theta^i} \left[ Z \mid \theta^i, Y \right] + y_4 \right) \log(\theta)
\\
& \quad \quad \quad \quad p(Z \mid Y, \theta^i) \sim \text{Binomial}(y_1, \theta/(\theta+2)) \implies
\mathbb{E}_{Z \mid \theta^i} \left[ Z \mid \theta^i, Y \right] = \frac{y_1 \theta^i}{\theta^i + 2}
\\
& = (y_2 + y_3) \log(1 - \theta) + \left( \frac{y_1 \theta^i}{\theta^i + 2}+ y_4 \right) \log(\theta)
\\[1ex]
&\underline{\textbf{M Step}: \textstyle\arg\max_\theta Q(\theta, \theta^i)}
\\
\frac{ \partial Q(\theta, \theta^i)}{\partial \theta} & = - \frac{y_2 + y_3}{1 - \theta} + \frac{ \mathbb{E}Z}{\theta} \stackrel{\text{set}}{=} 0
\quad \quad \implies \theta^{i+1} = \frac{ \mathbb{E}Z + y_4}{\mathbb{E}Z + y_2 + y_3 + y_4}
\\[0.5ex]
\theta^{i+1} & = \frac{ \frac{y_1 \theta^i}{\theta^i + 2} + y_4}{\frac{y_1 \theta^i}{\theta^i + 2} + y_2 + y_3 + y_4}
\end{aligned}
$$
```

```{r p1EM-function, eval=TRUE}
func_EM <- function(start, max.iteration, Y){
  theta_i <- start
  chain <- rep(NA, max.iteration)
  chain[1] <- theta_i
  EZ_i <- (Y[1]*theta_i)/(theta_i+2)
  theta_i <- (EZ_i + Y[4]) / (EZ_i + Y[2] + Y[3] + Y[4])
  chain[2] <- theta_i
  for (j in 3:max.iteration){
    EZ_i <- (Y[1]*theta_i)/(theta_i+2)
    theta_i <- (EZ_i + Y[4]) / (EZ_i + Y[2] + Y[3] + Y[4])
    chain[j] <- theta_i
    if (abs(chain[j]- chain[j-1]) <= 1e-07){
      estimate <- decimal(chain[j], 10)
      break  }
    else (estimate <- "DID NOT CONVERGE")
    }
  it.used <- length(chain[!is.na(chain)])
  result <- c(start, estimate, it.used)
  return(result)
}
```

```{r p1a, eval=TRUE}
Y1 <- c(125, 18, 20, 34)
mle1 <- 0.62682
se1 <- 0.05382
print.Y1 <- paste(Y1, collapse=",")

start.1 <- func_EM(0.1, 100, Y1)
start.2 <- func_EM(0.2, 100, Y1)
start.3 <- func_EM(0.3, 100, Y1)
start.4 <- func_EM(0.4, 100, Y1)
start.6 <- func_EM(0.6, 100, Y1)
start.8 <- func_EM(0.8, 100, Y1)

table1a <- cbind(start.1, start.2, start.3, start.4, start.6, start.8)
rownames(table1a) <- c("Start Value", "Estimation", "Iterations Used")
colnames(table1a) <- c(rep("", 6))
knitr::kable(table1a, align='rrrrrr', caption=paste("EM for Y=(", print.Y1, ")"), booktabs=T, 'latex') %>%
  kableExtra::kable_styling(latex_options="hold_position" )
```

```
Convergence is determined if the the values within the chain have an absolute difference less than 1e-07.
```

```
\newpage

### Problem 1c
Plot the normal approximation along with the normalized likelihood.  Is the normal approximation appropriate in this case?


```{r p1c-graph}
func_scalelike<-function(x,y1,y2,y3,y4){
  like <-(2+x)^y1*(1-x)^(y2+y3)*(x)^y4
  like.max<-max(like)
  like/like.max #normalized likelihood (on scale from 0 to 1)
}

func_scalenormal<-function(x, mean, sd){
  scales::rescale(dnorm(x, mean, sd), to=c(0, 1)) #normal (on scale from 0 to 1)
}

func_plots <- function(yval, mle, se){
colors <- c("navy", "maroon")
norm.like <-    stat_function(fun = func_scalelike, args = list(y1=yval[1], y2=yval[2], y3=yval[3], y4=yval[4]), lwd = 1.5, linetype="solid", aes(col="N
normal.approx <-stat_function(fun = func_scalenormal, args = list(mean=mle, sd=se), lwd = 2.5, linetype="dotted", aes(col="Normal Approximation"))

print.yval <- paste(yval, collapse=", ")
name <- paste("Normal Likelihood and Normal Approximation for Y=(", print.yval, ")")


x <- seq(0, 1, 0.001)
df <- data.frame("X"=x)
ggplot(data=df, aes(x=X))+
  norm.like+normal.approx+
  ggtitle(paste(name))+
  theme(axis.title.x = element_blank())+
  scale_colour_manual("", values = c(colors[1], colors[2]))
  #theme(legend.position = "bottom")+

}
```


```{r p1c, warning=FALSE}
func_plots(Y1, mle1, se1)
```
The Normal Approximation is appropirate in this case.

Problem 1d
Repeat (a) and (c) for the data $Y = (14, 0, 1, 5)$. did the algorithm coverage for all of the above starting values?


```{r p1d_a}
Y2 <- c(14, 0, 1, 5)
mle2 <- 0.903344
se2 <- 0.09348
print.Y2 <- paste(Y2, collapse=",")

start.1 <- func_EM(0.1, 100, Y2)
start.2 <- func_EM(0.2, 100, Y2)
start.3 <- func_EM(0.3, 100, Y2)
start.4 <- func_EM(0.4, 100, Y2)
start.6 <- func_EM(0.6, 100, Y2)
start.8 <- func_EM(0.8, 100, Y2)

table1d_a <- cbind(start.1, start.2, start.3, start.4, start.6, start.8)
rownames(table1d_a) <- c("Start Value", "Estimation", "Iterations Used")
colnames(table1d_a) <- c(rep("", 6))

knitr::kable(table1d_a, align='rrrrrr', caption=paste("EM for Y=(", print.Y2, ")"), booktabs=T, 'latex') %>%
  kableExtra::kable_styling(latex_options="hold_position" )
```


```{r p1d_c, warning=FALSE}
func_plots(Y2, mle2, se2)
```
The Normal Approximation is not appropriate in this case.

Problem 2
Repeat Problem 1 (a) and (d) using the Monte Carlo *EM*. How did you assess convergence.
```{r p2a-MCEM}
func_MCEM <- function(start, max.iteration, Y, m){
  theta_i <- start
  chain <- rep(NA, max.iteration)
  chain[1] <- theta_i
  p.z <- theta_i / (theta_i +2)
  EZ_i <- mean(rbinom(m, Y[1], p.z))
  theta_i <- (EZ_i + Y[4]) / (EZ_i + Y[2] + Y[3] + Y[4])
  chain[2] <- theta_i
  for (j in 3:max.iteration){
    p.z <- theta_i / (theta_i +2)
    EZ_i <- mean(rbinom(m, Y[1], p.z))
    theta_i <- (EZ_i + Y[4]) / (EZ_i + Y[2] + Y[3] + Y[4])
    chain[j] <- theta_i
    if (abs(chain[j]- chain[j-1]) <= 1e-07){
      estimate <- decimal(chain[j], 10)
      break  }
    else (estimate <- "DID NOT CONVERGE")
    }
  it.used <- length(chain[!is.na(chain)])
```
```

````
  result <- c(start, estimate, it.used)
  return(result)
}

func_MCMC.m <- function(start.vec, max.iteration, Y, m){

start.1 <- func_MCEM(start.vec[1], max.iteration, Y, m)
start.2 <- func_MCEM(start.vec[2], max.iteration, Y, m)
start.3 <- func_MCEM(start.vec[3], max.iteration, Y, m)
start.4 <- func_MCEM(start.vec[4], max.iteration, Y, m)
start.6 <- func_MCEM(start.vec[5], max.iteration, Y, m)
start.8 <- func_MCEM(start.vec[6], max.iteration, Y, m)

table<- cbind(start.1, start.2, start.3, start.4, start.6, start.8)
rownames(table) <- c("Start Value", "Estimation", "Iterations Used")
colnames(table) <- c(rep("", 6))

print.Y <- paste(Y, collapse=",")

knitr::kable(table, align='rrrrrr', caption=paste("MCEM for Y=(", print.Y, "), m=",m), booktabs=T, 'latex') %>%
   kableExtra::kable_styling(latex_options="hold_position" )
}
```

```{r p2_1}
Y1 <- c(125, 18, 20, 34)
start.vec <- c(0.1, 0.2, 0.3, 0.4, 0.6, 0.8)

func_MCMC.m(start.vec, 10000, Y1, 1e02)
func_MCMC.m(start.vec, 10000, Y1, 1e03)
func_MCMC.m(start.vec, 10000, Y1, 1e04)
```

```{r p2_2}
Y2 <- c(14, 0, 1, 5)
start.vec <- c(0.1, 0.2, 0.3, 0.4, 0.6, 0.8)

func_MCMC.m(start.vec, 10000, Y2, 1e02)
func_MCMC.m(start.vec, 10000, Y2, 1e03)
func_MCMC.m(start.vec, 10000, Y2, 1e04)
```
````

```
## Error: <text>:2:13: unexpected numeric constant
## 1: ---
## 2: title: STAT 457
##                ^
```

The R session information (including the OS info, R version and all packages used):

```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17763)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252  LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252 LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] MCMCpack_1.4-4      MASS_7.3-51.4       coda_0.19-3         dplyr_0.8.3
##  [5] gtable_0.3.0        numDeriv_2016.8-1.1 matlib_0.9.2        latex2exp_0.4.0
##  [9] gridExtra_2.3       readr_1.3.1         ggplot2_3.2.1
##
## loaded via a namespace (and not attached):
##  [1] rgl_0.100.30            Rcpp_1.0.2              lattice_0.20-38
##  [4] assertthat_0.2.1        zeallot_0.1.0           digest_0.6.20
##  [7] mime_0.7                R6_2.4.0                cellranger_1.1.0
## [10] MatrixModels_0.4-1      backports_1.1.4         evaluate_0.14
## [13] highr_0.8               httr_1.4.1              pillar_1.4.2
## [16] rlang_0.4.0             lazyeval_0.2.2          curl_4.2
## [19] readxl_1.3.1            SparseM_1.77            rstudioapi_0.10
## [22] data.table_1.12.4       miniUI_0.1.1.1          car_3.0-3
## [25] Matrix_1.2-17           rmarkdown_1.14          labeling_0.3
## [28] webshot_0.5.1           stringr_1.4.0           foreign_0.8-71
## [31] htmlwidgets_1.5.1       tinytex_0.15            munsell_0.5.0
## [34] shiny_1.3.2             compiler_3.6.1          httpuv_1.5.1
## [37] xfun_0.8                pkgconfig_2.0.2         mcmc_0.9-6
## [40] htmltools_0.3.6         tidyselect_0.2.5        tibble_2.1.3
## [43] rio_0.5.16              viridisLite_0.3.0       crayon_1.3.4
## [46] withr_2.1.2             later_0.8.0             jsonlite_1.6
## [49] xtable_1.8-4            magrittr_1.5            scales_1.0.0
## [52] zip_2.0.4               stringi_1.4.3           carData_3.0-2
## [55] promises_1.0.1          xml2_1.2.2              vctrs_0.2.0
## [58] openxlsx_4.1.0.1        kableExtra_1.1.0        tools_3.6.1
## [61] forcats_0.4.0           manipulateWidget_0.10.0 glue_1.3.1
## [64] purrr_0.3.2             hms_0.5.0               crosstalk_1.0.0
```
```

```
## [67] abind_1.4-5           yaml_2.2.0            colorspace_1.4-1
## [70] rvest_0.3.4           knitr_1.24            haven_2.1.1
## [73] quantreg_5.51
```

```
Sys.time()
```

```
## [1] "2019-11-29 21:23:55 CST"
```