PHP 2530: BAYESIAN STATISTICAL METHODS
HOMEWORK III R CODE APPENDIX

NICK LEWIS

Packages

R Code

```
library(ggplot2) #this makes better looking plots in R
theme_set(theme_minimal())
library(patchwork) #for plot manipulations (i.e. subplots)
library(Cairo) #Windows is bad at makinf good ggplots so this helps with resolution
library(tidyr)
'
Use this pieces of code to get smooth ggplots for windows. I dont like this
method since it uses too much memory in declaring, but do this to save your plots
'


#ggsave(county.plot,path="~/Bayesian Statistical Methods/HW3/Figures",
#       filename = 'County plots.png', dpi = 300, type = 'cairo',
#       width = 11, height = 8, units = 'in',bg="white")
```

Problem 1

R Code

```
### Problem 1 (BDA 3rd Ed. Exercise 5.3)

#Part a
#data for school
school.data <- data.frame(y = c(28,8,-3,7,-1,1,18,12),
                          sd = c(15,10,16,11,9,11,10,18) )

#Number of iterations
N <- 500
#range of tau as described on page 121
tau <- seq(from = 0.0001, to = 40, length.out = N)
#best guess of mu
mu <- seq(from = -30, to = 30, length.out = N)


#posterior density for (mu,tau)|y (bottom of page 116)
school <- function(a,b,data,sd){
  '
    PARAMETERS:
    ----------
        data - estimated means
        sd - accompanying standard deviations
        a - gridspace for mu
        b - grid space for tau
```

```r
    Returns:
    -------
    logpost:natural logarithm of unnormalized posterior density
    '
  loglik  <- function(x,y,m,s){
   -((m - x)^2 / (2*(y^2 + s^2))) - 0.5*log(y^2 + s^2)
  }
  prior <- 1
  logpost <- log(prior) #initialize logposterior with prior

  for(j in 1:length(data)) {logpost <- logpost + loglik(a,b,data[j],sd[j])}

return( logpost )
}


#calculates the logposterior
school.post <- outer(mu,tau,school,
                     data=school.data[,"y"],
                     sd=school.data[,"sd"])
#calculates the posterior
school.post <- exp(school.post)

#posterior used to select random numbers that correspond to indices of our grid
samples <- sample(length(school.post), size = 2000,replace = T,
                  prob = c(school.post) )

#Random jitter
#plug sample values in to give us result + random jitter
d.mu <- diff(mu)[1]/2
d.tau <- diff(tau)[1]/2

#I don't declare these anymore to save on memory
mu.post <- rep(mu, times = length(tau))[samples]+runif(2000,-d.mu,d.mu)
tau.post <- rep(tau, each = length(mu))[samples]+runif(2000,0,d.tau)


theta.post <- function(x,y,data,sd){
  '
    PARAMETERS:
    ----------
        x - posterior draws for mu
        y - posterior draws for tau
        data - estimated means
        sd - accompanying standard deviations

    Returns:
    -------
    Posterior draws theta_j | tau, mu, y for each school
  '

  #lengths of posterior draws and data
  N <- length(x); n <- length(data)
  #variance
```

```r
  V <- outer(y,sd, function(x,y) 1 / sqrt((1/x)^2 + (1/y)^2) )
  #mean
  mu <- outer(x/y^2,data/sd^2, function(x,y) x+y)*V^2

  #use updates to now make informed draws of theta_j
  thetas <- sapply(1:n,function(j) rnorm(n = N,mean = mu[,j], sd = V[,j]))
  return(thetas)
}

thetas <- theta.post(mu.post,tau.post,
                     data=school.data[,"y"],
                     sd=school.data[,"sd"])


'smoother way to calculate best probability. This gives us the col number of the
best, then organizes them (table function), and then we average over our draws.'

Best <- table( apply(thetas,1, which.max) ) / nrow(thetas)

#Just some organizational work
Best <- 100*round(matrix(Best,ncol=nrow(school.data)),3)
colnames(Best) <- LETTERS[1:8]; rownames(Best) <-"Pr"

print(Best)
#nicer way to write our probability matrix

#I don't like the way it looks so I transpose it
Probabilities <- apply(thetas, 2, function(x) 100*round(colMeans(x > thetas),3))

#turns array into matrix and turns it into dataframe
Probabilities <- as.data.frame(t(Probabilities))
rownames(Probabilities) <- colnames(Probabilities) <- colnames(Best)

print(Probabilities)


### PART B
theta.inf <- sapply(1:nrow(school.data), function(x) rnorm(n=2000,
                                mean=school.data[x,"y"],sd=school.data[x,"sd"]))

Best.inf <- table( apply(theta.inf,1, which.max) ) / nrow(theta.inf)

#Just some organizational work
Best.inf <- 100*round(matrix(Best.inf,ncol=nrow(school.data)),3)
colnames(Best.inf) <- LETTERS[1:8]; rownames(Best) <-"Pr"

print(Best.inf)
#nicer way to write our probability matrix

#I don't like the way it looks so I transpose it
Prob.inf <- apply(theta.inf, 2, function(x) 100*round(colMeans(x > theta.inf),3))

#turns array into matrix and turns it into dataframe
Prob.inf <- as.data.frame(t(Prob.inf))
```

```r
rownames(Prob.inf) <- colnames(Prob.inf) <- colnames(Best.inf)

print(Prob.inf)

#### REPRODUCING THE FIGURES AND COMPUTATIONS IN SECTION 5.5

#Plots p(tau |y)
tau.plot <- ggplot()+
  geom_line(aes(x=tau,y  = colSums(school.post)),size=1.60,col="hotpink") +
  coord_cartesian(xlim = c(0,30)) + #chooses limits for x,y axis
  scale_x_continuous(breaks=seq(0,30,by=5))+ #breaks the x-axis into pieces
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=20),
        axis.text = element_text(colour = "black",size = 20,face="bold"),
        axis.title = element_text(size = 24,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5),
        legend.background = element_blank())+
  ylab(~ paste("p(",tau,"|y)"))+
  xlab(~paste(tau))

#Graphs of E(theta_j | tau,y)
expected.tau <- function(x,data,sd){
  '
    PARAMETERS:
    ----------
        x - grid for tau
        data - estimated means
        sd - accompanying standard deviations

    Returns:
    -------
    E(theta_j | tau,y) for every school
  '

  mu.hat <- sapply(x, function(t) sum(data/(sd^2 + t^2)) / sum(1/(sd^2+t^2)) )
  V.tau <- outer(x,sd, function(x,y) 1 / ((1/x)^2 + (1/y)^2 ) )
  theta.taus <- outer(mu.hat/x^2,data/sd^2,function(x,y) x+y)*V.tau
  colnames(theta.taus) <- sapply(LETTERS[1:8],function(x) paste0("School ", x))
  theta.taus <- as.data.frame(cbind(x,theta.taus)) %>% gather(school,p,-x)
  return(theta.taus)
}

school.mean <- expected.tau(x=tau,
                     data=school.data[,"y"],
                     sd=school.data[,"sd"])
ggplot(data=school.mean)+
  geom_line(aes(x=x,y = p,color=school,group=school),size=1.7) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=20),
```

```
        axis.text = element_text(colour = "black",size = 20,face="bold"),
        axis.title = element_text(size = 24,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5),
        legend.background = element_blank())+
  scale_color_manual(name="School", values=c("green",
                                             "steelblue",
                                             "red","purple","blue","yellow",
                                             "pink","orange"),
                     labels=unique(school.mean[,"school"]) )+
  ylab(~ paste("E(",theta[j],"|",tau,", y)"))+
  xlab(~paste(tau))

#Graphs of sd(theta_j | tau,y)

sd.tau <- function(x,data,sd){
  '
    PARAMETERS:
    ----------
        x - grid for tau
        data - estimated means
        sd - accompanying standard deviations

    Returns:
    -------
    sd(theta_j | tau,y) for every school
  '

  mu.hat.V <- sapply(x, function(t) 1 / sum(1/(sd^2+t^2)) )
  V <- outer(x,sd, function(x,y) 1 / ((1/x)^2 + (1/y)^2 ) )
  V.taus <- outer(x,sd, function(x,y) (1/x)^2 / ((1/x)^2 + (1/y)^2 ) )
  return(sqrt( V+mu.hat.V*V.taus^2 ))
}

school.sd <- sd.tau(x=tau,
                    data=school.data[,"y"],
                    sd=school.data[,"sd"])

#add little jitter so these are visible
school.sd[,7] <- school.sd[,7]+75*diff(school.sd[,7])[1]
school.sd[,6] <- school.sd[,6]+75*diff(school.sd[,6])[1]


colnames(school.sd) <- sapply(LETTERS[1:8],function(x) paste0("School ", x))
school.sd <- as.data.frame(cbind(tau,school.sd)) %>% gather(school,p,-tau)

ggplot(data=school.sd)+
  geom_line(aes(x=tau,y = p,color=school,group=school),size=1.60) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=20),
        axis.text = element_text(colour = "black",size = 20,face="bold"),
        axis.title = element_text(size = 24,face="bold"),
```

```
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5),
        legend.background = element_blank())+
  scale_color_manual(name="   School", values=c("green",
                                              "steelblue",
                                              "red","purple","blue","yellow",
                                              "pink","orange"),
                     labels=unique(school.sd[,"school"]) )+
  ylab(~ paste("sd(",theta[j],"|",tau,", y)"))+
  xlab(~paste(tau))
```

Problem 4

R Code

```
### PROBLEM 4 (BDA 3rd Ed, Exercise 5.14)

#Data for this problem
#y-bikes for streets w/ bike lanes;v- vehicles for streets w/ bike lanes
res.data <- data.frame(bikes = c(16, 9, 10, 13,19, 20, 18, 17,35, 55) ,
                       vehicles = c(58,90, 48, 57, 103, 57, 86,112, 273, 64))

res.data["total"] <- res.data["bikes"]+res.data["vehicles"]

vehicle.post <- function(a,b,v){
  '
  PARAMETERS:
  ----------
    v - vehicle data
    a - alpha
    b - beta

  Returns:
  -------
    logpost: natural logarithm of the unnormalized posterior density
  '
  N <- length(v) #length of the data

  prior <- (a+b)^(-5/2) #prior

  loglik <- function(a,b,v){
    lgamma(a+v) + a*log(b/(b+1)) - (v)*log(b+1) - lgamma(a) - lgamma(v+1)
  }

  logpost <- log(prior)
  for (j in 1:N){logpost <- logpost + loglik(a,b,v = v[j])}

  #subract maximum to reduce overflow
  return( logpost )
}

alpha <- seq(from=0.0001, to = 15, length.out=500)
beta <- seq(from=0.0001, to = 0.3, length.out=500)

#Calculation of the log Posterior Distributions
```

```
vehicles <- outer(alpha,beta,vehicle.post,v=res.data[,"total"])

#calculates posterior
vehicles <- exp(vehicles - max(vehicles))

#Make vectors that contain all pairwise combinations of A and B
alpha.grid <- rep(alpha, times = length(beta))
beta.grid <- rep(beta, each = length(alpha))

v.samps <- 2000
samples <- sample(length(vehicles), size = v.samps,replace = T,
                  prob = c(vehicles) )

#plug sample values in to give us result + random jitter
d.alpha <- diff(alpha)[1]/2
d.beta <- diff(beta)[2]/2

alphas.post <-  alpha.grid[samples]+runif(v.samps,min = -d.alpha, max = d.alpha)
beta.post <- beta.grid[samples]+runif(v.samps,min = -d.beta, max = d.beta)

# This is another Posterior plot, but this has the simulated points built on top of it
vehicle.data <- data.frame(alpha = alpha.grid,
                           beta = beta.grid,
                           post = c(vehicles))
point.data <- data.frame(x=alphas.post,y=beta.post,z=c(vehicles)[samples])
#contour levels for flight posterior
levels <- c(0.001,0.01,0.05,0.25,0.50,0.75,0.95)
vehicle.cont <- quantile(seq(min(vehicles),max(vehicles),length.out=1e5),levels)

# Contour Plot of Observed Vehicles Posterior
vehicle.plot <- ggplot(vehicle.data, aes(x=alpha, y= beta, z=post))+
  stat_contour(breaks= vehicle.cont,color="black",size = 1.4)+ #contour levels
  scale_fill_gradient(low = 'yellow', high = 'red', guide = "none") +
  scale_alpha(range = c(0, 1), guide = "none")+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=20),
        axis.text = element_text(colour = "black",size = 20,face="bold"),
        axis.title = element_text(size = 24,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab(~ paste(beta))+
  xlab(~ paste(alpha))

vehicles.samples.plot <- ggplot(vehicle.data, aes(x=alpha, y= beta, z=post))+
  stat_contour(breaks= vehicle.cont,color="black",size = 1.4)+ #contour levels
  coord_cartesian(xlim = c(0,15), ylim = c(0,0.15)) +
  scale_fill_gradient(low = 'yellow', high = 'red', guide = "none") +
  scale_alpha(range = c(0, 1), guide = "none")+
  geom_point(aes(x=x,y=y,z=z),point.data,colour="red",size=3)+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
```

```r
        text = element_text(size=20),
        axis.text = element_text(colour = "black",size = 20,face="bold"),
        axis.title = element_text(size = 24,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab(~ paste(beta))+
  xlab(~ paste(alpha))

(vehicle.plot | vehicles.samples.plot) +
  plot_annotation(tag_levels = 'A')

#Here we get our posterior draws for the parameters and hyperparameters
theta.params <- sapply(1:nrow(res.data), function(x)
                  rgamma(n=v.samps,
                  shape = alphas.post+res.data[x,"total"],
                    rate = beta.post+1))

vehicle.params <- cbind(alphas.post,beta.post,theta.params)
colnames(vehicle.params) <- c("alpha","beta",
                              sapply(1:nrow(res.data),function(j) paste0("theta ",j)) )

vehicle.stats <- data.frame(mean = apply(vehicle.params,2,mean),
                         std = apply(vehicle.params,2,sd),
                         p_0.025 = apply(vehicle.params,2,quantile,probs=0.025),
                         p_0.25 = apply(vehicle.params,2,quantile,probs=0.25),
                         p_0.50 = apply(vehicle.params,2,quantile,probs=0.50),
                         p_0.75 = apply(vehicle.params,2,quantile,probs=0.75),
                         p_0.975 = apply(vehicle.params,2,quantile,probs=0.975))

vehicle.stats <- round(vehicle.stats,3)
rownames(vehicle.stats) <- colnames(vehicle.params)
colnames(vehicle.stats) <- c("Mean","Standard Deviation","2.5%",
                          "25%","50%","75%","97.5%")
vehicle.stats
```

Problem 5

R Code

```r
### PROBLEM 5 (BDA 3rd Ed., Exercise 6.2)

#data for the problem
df <- data.frame(year =1:10 ,
                 accidents=c(24, 25, 31, 31, 22, 21, 26, 20, 16, 22),
                 deaths=c(734, 516, 754, 877, 814, 362, 764, 809, 223, 1066),
                 rate =c(0.19, 0.12, 0.15, 0.16, 0.14, 0.06, 0.13, 0.13, 0.03, 0.15)
)
df["miles"] <- df["deaths"]*(1e8)/df["rate"]

#Prior distribution parameters (here so you can adjust for different priors)
prior.shape <- 0; prior.rate <- 0

R <- 1000
## APPROACH: FIND POSTERIOR PREDICTIVE DISTRIBUTION
```

```r
sizes <- c(sum(df["accidents"]),sum(df["deaths"])) + prior.shape

#corresponding probability parameters
probs1 <- rep(nrow(df)/(nrow(df)+1+prior.rate),10)
probs2 <-sum(df["miles"])/(sum(df["miles"])+df[,"miles"]+prior.rate)

# Replication Draws for each model
M1.draws <- replicate(1000, rnbinom(n=10,size = sizes[1],prob= probs1) )
M2.draws <- replicate(1000, rnbinom(n=10,size = sizes[1],prob= probs2) )
M3.draws <- replicate(1000, rnbinom(n=10,size = sizes[2],prob= probs1) )
M4.draws <- replicate(1000, rnbinom(n=10,size = sizes[2],prob= probs2) )


#First Test Statistic

M1.T1 <- apply(M1.draws,2,function(x) acf(x,plot=F,lag.max=1)$acf[2])

M2.T1 <- apply(M2.draws,2,function(x) acf(x,plot=F,lag.max=1)$acf[2])

M3.T1 <- apply(M3.draws,2,function(x) acf(x,plot=F,lag.max=1)$acf[2])

M4.T1 <- apply(M4.draws,2,function(x) acf(x,plot=F,lag.max=1)$acf[2])


Test1.hist <- ggplot() +aes(x=M1.T1) +
  geom_histogram(color="black", fill="blue")+
  geom_vline(xintercept = acf(df["accidents"],plot=F,lag.max=1)$acf[2],size=2)+
  annotate("text",x = -0.74, y=70, size = 6,label =
             paste0("p = ",mean(M1.T1 >= acf(df["accidents"],plot=F,lag.max=1)$acf[2])))+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

Test2.hist <- ggplot() +aes(x=M2.T1) +
  geom_histogram(color="black", fill="red")+
  geom_vline(xintercept = acf(df["accidents"],plot=F,lag.max=1)$acf[2],size=2)+
  annotate("text",x = -0.20, y=70, size = 6,label =
             paste0("p = ",mean(M2.T1 >= acf(df["accidents"],plot=F,lag.max=1)$acf[2])))+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
```

```
  xlab(~ paste(T(y^{rep})))

Test3.hist <- ggplot() +aes(x=M3.T1) +
  geom_histogram(color="black", fill="yellow")+
  geom_vline(xintercept = acf(df["deaths"],plot=F,lag.max=1)$acf[2],size=2)+
  annotate("text",x = 0.3, y=80, size = 6,label =
             paste0("p = ",mean(M3.T1 >= acf(df["deaths"],plot=F,lag.max=1)$acf[2])))+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))


Test4.hist <- ggplot() +aes(x=M4.T1) +
  geom_histogram(color="black", fill="green")+
  geom_vline(xintercept = acf(df["deaths"],plot=F,lag.max=1)$acf[2],size=2)+
  annotate("text",x = 0.2, y=300, size = 6,face="bold",label =
             paste0("p = ",mean(M4.T1 >= acf(df["deaths"],plot=F,lag.max=1)$acf[2])))+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

(Test1.hist | Test2.hist) / (Test3.hist | Test4.hist) +
  plot_annotation(tag_levels = 'A')

#Second Test Statistic

#Replications

M1.T2 <- apply(M1.draws,2,function(x) cor(x,df[,"year"],method = "spearman"))

M2.T2 <- apply(M2.draws,2,function(x) cor(x,df[,"year"],method = "spearman"))

M3.T2 <- apply(M3.draws,2,function(x) cor(x,df[,"year"],method = "spearman"))

M4.T2 <- apply(M4.draws,2,function(x) cor(x,df[,"year"],method = "spearman"))


Test1.hist2 <- ggplot() +aes(x=M1.T2) +
  geom_histogram(color="black", fill="blue")+
  geom_vline(xintercept = cor(df[,"accidents"],df[,"year"],method = "spearman"),size=2)+
```

```
  annotate("text",x = 0.6, y=75, size = 5,label =
              paste0("p = ",
          mean(M1.T2 >= cor(df[,"accidents"],df[,"year"],method = "spearman"))))+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

Test2.hist2 <- ggplot() +aes(x=M2.T2) +
  geom_histogram(color="black", fill="red")+
  geom_vline(xintercept = cor(df[,"accidents"],df[,"year"],method = "spearman"),size=2)+
  annotate("text",x = -0.20, y=70, size = 5,label =
              paste0("p = ",
          mean(M2.T2 >= cor(df[,"accidents"],df[,"year"],method = "spearman"))))+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

Test3.hist2 <- ggplot() +aes(x=M3.T2) +
  geom_histogram(color="black", fill="yellow")+
  geom_vline(xintercept = cor(df[,"deaths"],df[,"year"],method = "spearman"),size=2)+
  annotate("text",x = 0.6, y=70, size = 5,label =
              paste0("p = ",
          mean(M3.T2 >= cor(df[,"deaths"],df[,"year"],method = "spearman"))))+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))


Test4.hist2 <- ggplot() +aes(x=M4.T2) +
  geom_histogram(color="black", fill="green")+
  geom_vline(xintercept = cor(df[,"deaths"],df[,"year"],method = "spearman"),size=2)+
  annotate("text",x = 0.4, y=70, size = 5,label =
              paste0("p = ",
```

```
            mean(M4.T2 >= cor(df[,"deaths"],df[,"year"],method = "spearman"))))+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

(Test1.hist2 | Test2.hist2) / (Test3.hist2 | Test4.hist2) +
  plot_annotation(tag_levels = 'A')
```

Problem 6

R Code

```
### PROBLEM 6 (BDA 3rd Ed. Exercise 6.7)

post.pred <- function(y,N){
  '
  PARAMETERS:
  ----------
    y - mean of the data
    N - number of samples in the data

  Returns:
  -------
      test statistic : the absolute value of the maximum from 100 samples
  '
  p <- rnorm(n=N,mean=y,sd=sqrt(1+1/N))
  return( abs(max(p)) )
}

sample.max <- replicate(1000,post.pred(y=5.1,N=100))

ggplot() +aes(x=sample.max) +
  geom_histogram(color="black", fill="gold")+
  geom_vline(xintercept = 8.1,size=2)+
  annotate("text",x = 7.5, y=150, size = 7,label =
             paste0("p = ",mean(sample.max > 8.1)))+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

prior.pred <- function(n,A){
```

```r
  '
  PARAMETERS:
  ----------
    n - number of draws
    A - bounds of the uniform prior on theta (i.e. theta ~Unif(-A,A))

  Returns:
  -------
    test statistic : the absolute value of the maximum from 100 samples
  '
  theta <- runif(n,-A,A)
  draws <- sapply(1:n, function(x)  rnorm(100,theta[x],1))
  reps <- apply(draws, 2, function(x) abs(max(x)) )
  return( reps )
}

#y.prior <- seq(from=-1e5,to=1e5,length.out=2e5)
prior.max <- prior.pred(n=100000,A=1e5)

ggplot() +aes(x=prior.max) +
  geom_histogram(color="black", fill="azure")+
  geom_vline(xintercept = 8.1,size=2)+
  annotate("text",x = 20000, y=75, size = 7,label =
             paste0("p = ",mean(prior.max> 8.1)))+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))
```

Problem 7

R Code

```r
### PROBLEM 7 (BDA 3rd Ed. Exercise 6.9)

#Rat Tumor Data for all 71 Experiments

rat.data <- data.frame(
  tumors = c( 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,
  1,  1,  1,  1,  1,  2,  2,  2,  2,  2,  2,  2,  2,  2,  1,  5,  2,
  5,  3,  2,  7,  7,  3,  3,  2,  9, 10,  4,  4,  4,  4,  4,  4,  4,
  10,  4,  4,  4,  5, 11, 12,  5,  5,  6,  5,  6,  6,  6,  6, 16, 15,
  15,  9,  4),

  rats = c(20, 20, 20, 20, 20, 20, 20, 19, 19, 19, 19, 18, 18, 17, 20, 20, 20,
  20, 19, 19, 18, 18, 25, 24, 23, 20, 20, 20, 20, 20, 20, 10, 49, 19,
  46, 27, 17, 49, 47, 20, 20, 13, 48, 50, 20, 20, 20, 20, 20, 20, 20,
  48, 19, 19, 19, 22, 46, 49, 20, 20, 23, 19, 22, 20, 20, 20, 52, 46,
  47, 24, 14)
```

```r
  )

R2 <- 200

ralpha <- seq(from = 0.5, to = 12, length = R2)
rbetas <- seq(from = 3, to = 53, length = R2)

#of form seen in derivation
rat.post <- function(y,n,a,b){
  '
  PARAMETERS:
  ----------
    y - data on bicyle proportions on residential street
    n - number of vehicles seen in total
    a - alpha parameter
    b - beta parameter

  Returns:
  -------
    logpost: natural logarithm of the unnormalized posterior density
  '

#length of the data
N <- length(y)

prior <- (a+b)**(-5/2) #prior distribution used for this problem

# for brevity, split the likelihood into a numerator term and denominator
loglik <- function(a,b,y,n){
  upper <- lgamma(a+b)+lgamma(a + y)+lgamma(b + (n - y))
  lower <- lgamma(a)+lgamma(b)+lgamma(a + b + n)
  return(upper - lower)
    }
logpost <- log(prior) #initialize value for the for loop with log prior

for (j in 1:N) { logpost <- logpost + loglik(a,b,y=y[j],n=n[j]) }

return(logpost)
}

#calculates log posterior
rats <- outer(ralpha,rbetas,rat.post,y=rat.data[,"tumors"],n=rat.data[,"rats"])

rats <- exp(rats-max(rats)) #calculates the posterior

#Make vectors that contain all pairwise combinations of A and B
ralpha.grid <- rep(ralpha, times = length(rbetas))
rbeta.grid <- rep(rbetas, each = length(ralpha))

'
unravel matrix going row to row instead of column to column. This way we sample
(mu, sigma) instead of having to sample mu, then sigma.
'
ratsamps <- 1000
```

```r
samples <- sample(length(rats), size = ratsamps,replace = T,
                  prob = c(rats) )

#plug sample values in to give us result + random jitter
d.alpha <- diff(ralpha)[1]/2
d.beta <- diff(rbetas)[1]/2

ralpha.post <-  ralpha.grid[samples]+runif(ratsamps,-d.alpha, d.alpha)
rbeta.post <- rbeta.grid[samples]+runif(ratsamps,-d.beta,d.beta)

# This is another Posterior plot, but this has the simulated points built on top of it
rat.stuff <- data.frame(alpha = ralpha.grid,
                           beta = rbeta.grid,
                           post = c(rats))
point.data <- data.frame(x=ralpha.post,y=rbeta.post,z=c(rats)[samples])

#contour levels for flight posterior
levels <- c(0.001,0.01,0.05,0.25,0.50,0.75,0.95)
rat.cont <- quantile(seq(min(rats),max(rats),length.out=1e5),levels)

rat.plot <- ggplot(rat.stuff, aes(x=alpha, y= beta, z=post))+
  stat_contour(breaks= rat.cont,color="black",size = 1.4)+ #contour levels
  scale_fill_gradient(low = 'yellow', high = 'red', guide = "none") +
  scale_alpha(range = c(0, 1), guide = "none")+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=20),
        axis.text = element_text(colour = "black",size = 20,face="bold"),
        axis.title = element_text(size = 24,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab(~ paste(beta))+
  xlab(~ paste(alpha))

rat.samples.plot <- ggplot(rat.stuff, aes(x=alpha, y= beta, z=post))+
  stat_contour(breaks= rat.cont,color="black",size = 1.4)+ #contour levels
  scale_fill_gradient(low = 'yellow', high = 'red', guide = "none") +
  scale_alpha(range = c(0, 1), guide = "none")+
  geom_point(aes(x=x,y=y,z=z),point.data,colour="red",size=3)+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=20),
        axis.text = element_text(colour = "black",size = 20,face="bold"),
        axis.title = element_text(size = 24,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab(~ paste(beta))+
  xlab(~ paste(alpha))

 (rat.plot | rat.samples.plot) +
  plot_annotation(tag_levels = 'A')
```

```r
#theta parameters
rat.params <- sapply(1:nrow(rat.data), function(x)
  rbeta(n=ratsamps,
         shape1 = ralpha.post+rat.data[x,"tumors"],
         shape2 = rbeta.post+(rat.data[x,"rats"]-rat.data[x,"tumors"]) ))

rat.draws <- sapply(1:nrow(rat.data), function(x)
                    rbinom(n=ratsamps,
                           size = rat.data[x,"rats"],
                           prob = rat.params[,x]))


#test statistics

#max number of tumors
rat.max <- apply(rat.draws,1,max)

#Mean proportion
ratprop.mean <- apply(rat.draws,1,function(x) mean(x/rat.data[,"rats"]) )

#Number of Zeroes
rat.zeros <- apply(rat.draws,1,function(x) sum(x==0) )

# Histograms for Max test statistic

Test1.hist <- ggplot() +aes(x=rat.max) +
  geom_histogram(color="black", fill="blue",bins=23)+
  geom_vline(xintercept = max(rat.data[,"tumors"]),size=2)+
  annotate("text",x = 25, y=200, size = 7,label =
             paste0("p = ",mean(rat.max >= max(rat.data[,"tumors"]) )) )+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

Test2.hist <- ggplot() +aes(x=ratprop.mean) +
  geom_histogram(color="black", fill="red",bins=23)+
  geom_vline(xintercept = mean(rat.data[,"tumors"]/rat.data[,"rats"]),size=2)+
  annotate("text",x = 0.15, y=300, size = 7,label =
             paste0("p = ",
          mean(ratprop.mean >= mean(rat.data[,"tumors"]/rat.data[,"rats"]) )) )+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
```

```
         axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

Test3.hist <- ggplot() +aes(x=rat.zeros) +
  geom_histogram(color="black", fill="yellow",bins=22)+
  geom_vline(xintercept = sum(rat.data[,"tumors"]==0),size=2)+
  annotate("text",x = 7, y=150, size = 7,label =
          paste0("p = ",mean(rat.zeros >= sum(rat.data[,"tumors"]==0) )) )+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

( Test1.hist | Test2.hist) /
  (plot_spacer() + Test3.hist + plot_spacer() + plot_layout(widths = c(1,2,1))) +
  plot_annotation(tag_levels = 'A')
```

Problem 9

R Code

```
### Problem 9 (BDA 3rd Ed, Exercise 7.6)

#data put into this form since I have to use pooled model
radon <- list(
  "Blue Earth" =  c(5,13,7.2,6.8,12.8,9.5,6,3.8,1.8,6.9,4.7,9.5),
  "Clay" = c(12.9,2.6,26.6,1.5,13,8.8,19.5,9.0,13.1,3.6),
  "Goodhue" =  c(14.3,7.6,2.6,43.5,4.9,3.5,4.8,5.6,3.5,3.9,6.7)
)
boxcox <- function(data,phi) {
  data <- unlist(data) #just in case a list is inputted
  if (min(data) <= 0){
    print("All data must be positive")
  }
  else {
    sapply(data, function(x) ifelse(phi !=0 , (x^(phi) - 1) / (phi), log(x) ) )
  }
}

#PART A

N9 <- 1000
phi.grid <- seq(from = -5, to = 5 , length.out = N9)

#with our draw of blue earth, we will update our values of phi
phi.post <- function(y,phi){
  '
    PARAMETERS:
```

```
    ----------
        y-data
        phi - phi parameter

    Returns:
    -------
      post: the unnormalized posterior density
    '
 y <- unlist(y); n <- length(y) #number of observations

geo.mean <- (phi-1)*(1-(1/n))*sum(log(y)) #geometric mean in log form

boxcox.var <- var(boxcox(data=y,phi=phi)) #sample variance of boxcox

phi.prior <- 1  #p(phi)

logpost <- log(phi.prior) + geo.mean - ((n-1)/2)*log(boxcox.var) #log posterior

return( exp(logpost))
}

phi.post.pooled <- function(data,phi){
  '
    PARAMETERS:
    ----------
        data - list of county data (NOTE: in list form)
        phi - phi parameter
    Returns:
    -------
      post: the unnormalized posterior density
    '
  y <- unlist(data); n <- sapply(data,length) #pooled data, number of observations
  N <- sum(n) #length of pooled data

  geo.mean <- (phi-1)*(1-(1/N))*sum(log(y)) #geometric mean in log form

  boxcox.vars <- sapply(1:length(n),
                function(x) ((n[x]-1)/2)*log(var(boxcox(data=data[x],phi=phi))))
  phi.prior <- 1  #p(phi)
  logpost <- log(phi.prior) + geo.mean - sum(boxcox.vars) #log posterior
  return( exp(logpost))
}

#posteriors for blue earth, clay and goodhue + pooled data
counties.df <- data.frame(x = phi.grid,
      blueearth = sapply(phi.grid, function(x) phi.post(y = radon["Blue Earth"], phi = x)),
      clay = sapply(phi.grid, function(x) phi.post(y = radon["Clay"], phi = x)),
      goodhue = sapply(phi.grid, function(x) phi.post(y = radon["Goodhue"], phi = x)),
      pooled = sapply(phi.grid, function(x) phi.post.pooled(data=radon, phi = x)))

#normalize densities
counties.df[,2:5] <- apply(counties.df[,2:5],2,function(x) x/ sum(x))

county.df <- counties.df %>% gather(counties,p,-x)
```

```
ggplot(data=county.df) + aes(x=phi.grid)+
  geom_line(aes(x=x,y = p,color=counties,group=counties),size=1.60) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=20),
        axis.text = element_text(colour = "black",size = 20,face="bold"),
        axis.title = element_text(size = 24,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5),
        legend.background = element_blank())+
  scale_color_manual(name="County", values=c("blue",
                                              "brown",
                                              "yellow","purple"),
                     labels=c("Blue Earth","Clay","Goodhue","Pooled"))+
  ylab(~ paste("p(",paste(phi),"|y)"))+
  xlab(~paste(phi))

#Statistics and values to use for parts a and b

#jitter
d.phi <- diff(phi.grid)[1]/2

#Draws
BE.draws <- ( sample(phi.grid,1000,replace=T,prob=counties.df[,"blueearth"])+
               runif(1000,-d.phi,d.phi) )

pooled.draws <- ( sample(phi.grid,1000,replace=T,prob=counties.df[,"pooled"])+
               runif(1000,-d.phi,d.phi) )

#PART A: Draws for mu, sigma for Blue Earth County

norm.param.draws <- function(y,phi){
  '
  PARAMETERS:
  ----------
    y - data
    phi - posterior draws of phi

  Returns:
  -------
    tuple of (mu,sigma) draws
  '

  #First, create your boxcox data
  #NOTE: this uses our external phi function so be wary
  boxcox.data <- boxcox(y,phi)

  #Second,now get sufficient statistics for the data
  y.mean <- apply(boxcox.data,1,mean)
  y.var <- apply(boxcox.data,1,var)
  n <- length(y)
  #Finally, get our samples of mu, sigma
```

```
  sigma <- sqrt( (n-1)*y.var/(rchisq(length(phi),n-1)) )
  mu <- rnorm(length(phi),y.mean, sigma/sqrt(n))

  return(cbind(mu,sigma))
}

# WE CALCULATE THE MEAN AND VARIANCE FOR mu and sigma.
df.blueearth  <- cbind(norm.param.draws(blue.earth,BE.draws),BE.draws)
colnames(df.blueearth) <- c("Mu","Sigma","Phi")

stats.blueearth  <- data.frame(mean = apply(df.blueearth,2,mean),
                        std = apply(df.blueearth,2,sd),
                        p_0.025 = apply(df.blueearth,2,quantile,probs=0.025),
                        p_0.25 = apply(df.blueearth,2,quantile,probs=0.25),
                        p_0.50 = apply(df.blueearth,2,quantile,probs=0.50),
                        p_0.75 = apply(df.blueearth,2,quantile,probs=0.75),
                        p_0.975 = apply(df.blueearth,2,quantile,probs=0.975))
stats.blueearth <- round(stats.blueearth,4)
rownames(stats.blueearth) <- c("Mu","Sigma","Phi")
colnames(stats.blueearth) <- c("Mean","Standard Deviation","2.5%",
                           "25%","50%","75%","97.5%")

print(stats.blueearth)

#print into latex
#xtable(stats.blueearth)

#PART B

df.countydata <- cbind(pooled.draws,norm.param.draws(blue.earth,pooled.draws),
                       norm.param.draws(clay,pooled.draws),
                       norm.param.draws(goodhue,pooled.draws))
colnames(df.countydata) <- c("Phi","Blue Earth Mu","Blue Earth Sigma",
                              "Clay Mu", "Clay Sigma","Goodhue Mu","Goodhue Sigma")

countydata <- data.frame(mean = apply(df.countydata,2,mean),
                           std = apply(df.countydata,2,sd),
                           p_0.025 = apply(df.countydata,2,quantile,probs=0.025),
                           p_0.25 = apply(df.countydata,2,quantile,probs=0.25),
                           p_0.50 = apply(df.countydata,2,quantile,probs=0.50),
                           p_0.75 = apply(df.countydata,2,quantile,probs=0.75),
                           p_0.975 = apply(df.countydata,2,quantile,probs=0.975))
countydata <- round(countydata,3)
rownames(countydata) <- c("Phi","Blue Earth Mu","Blue Earth Sigma",
                             "Clay Mu", "Clay Sigma","Goodhue Mu","Goodhue Sigma")
colnames(countydata) <- c("Mean","Standard Deviation","2.5%",
                           "25%","50%","75%","97.5%")
countydata


#PART C. Getting Samples

#phi to use
```

```r
phi.mode <- phi.grid[which.max(counties.df[,"pooled"])]
#Inverts the boxcox transform
invboxcox <- function(data,phi) {
    data <- unlist(data)
    sapply(data, function(x) ifelse(phi !=0 , (phi*x+1)^(1/phi),  exp(x) ) )
}


boxcox.samples <- function(n,y,phi){
  '
  PARAMETERS:
  ----------
    n - number of samples
    y - data
    phi - posterior draw of phi

  Returns:
  -------
    data replications for a county
  '

  #draw samples
  #First, create your boxcox data, and unlist your data

  y <- unlist(y)
  #NOTE: this uses our external phi function so be wary
  boxcox.data <- boxcox(y,phi)

  #Second,now get sufficient statistics for the data
  y.mean <- mean(boxcox.data)
  y.var <- var(boxcox.data)
  N <- length(y)
  #get our samples of mu, sigma
  sigma <- sqrt( (N-1)*y.var/(rchisq(n,N-1)) )
  mu <- rnorm(n,y.mean, sigma/sqrt(N))

  #use samples of mu, sigma to get boxcox draws

  data <- replicate(n,rnorm(N,mu,sigma))
  #reverse samples
  invdata <- sapply(1:n, function(x) invboxcox(data[,x],phi))

  return(invdata)
}

BE.samples <- boxcox.samples(n = 1000,
                             y = radon["Blue Earth"],
                             phi = phi.mode)

#use this to make sure no missingness
sum(is.na(c(BE.samples))    )

clay.samples <- boxcox.samples(n=1000,
                               y=radon["Clay"],
```

```r
                              phi = phi.mode)

sum(is.na(c(clay.samples))   )

goodhue.samples <- boxcox.samples(n=1000,
                                 y = radon["Goodhue"],
                              phi = phi.mode)

sum(is.na(c(goodhue.samples))   )

#Maximum Test Statistic
BlueEarth.max <- apply(BE.samples,2,max)
Clay.max <- apply(clay.samples,2,max)
Goodhue.max <- apply(goodhue.samples,2,max)

# Histograms for Max test statistic

Test1.hist <- ggplot() +aes(x=BlueEarth.max) +
  geom_histogram(color="black", fill="blue")+
  geom_vline(xintercept = max(blue.earth),size=2)+
  annotate("text",x = 50, y=200, size = 7,label =
             paste0("p-value = ",mean(BlueEarth.max >= max(blue.earth))) )+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

Test2.hist <- ggplot() +aes(x=Clay.max) +
  geom_histogram(color="black", fill="red")+
  geom_vline(xintercept = max(clay),size=2)+
  annotate("text",x = 100, y=150, size = 7,label =
             paste0("p-value = ",mean(Clay.max >= max(clay))) )+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

Test3.hist <- ggplot() +aes(x=Goodhue.max) +
  geom_histogram(color="black", fill="yellow")+
  geom_vline(xintercept = max(goodhue),size=2)+
  annotate("text",x = 100, y=200, size = 7,label =
             paste0("p-value = ",mean(Goodhue.max >= max(goodhue))) )+
```

```r
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

( Test1.hist | Test2.hist) /
  (plot_spacer() + Test3.hist + plot_spacer() + plot_layout(widths = c(1,2,1))) +
  plot_annotation(tag_levels = 'A')

#Standard Deviation Test Statistic

BlueEarth.sd <- apply(BE.samples,2,sd)
Clay.sd <- apply(clay.samples,2,sd)
Goodhue.sd <- apply(goodhue.samples,2,sd)

# Histograms for Max test statistic

Test1.hist <- ggplot() +aes(x=BlueEarth.sd) +
  geom_histogram(color="black", fill="blue")+
  geom_vline(xintercept = sd(blue.earth),size=2)+
  annotate("text",x = 15, y=200, size = 7,label =
             paste0("p-value = ",mean(BlueEarth.sd >= sd(blue.earth))) )+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

Test2.hist <- ggplot() +aes(x=Clay.sd) +
  geom_histogram(color="black", fill="red")+
  geom_vline(xintercept = sd(clay),size=2)+
  annotate("text",x = 40, y=100, size = 7,label =
             paste0("p-value = ",mean(Clay.sd >= sd(clay))) )+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))
```

```
Test3.hist <- ggplot() +aes(x=Goodhue.sd) +
  geom_histogram(color="black", fill="yellow")+
  geom_vline(xintercept = sd(goodhue),size=2)+
  annotate("text",x = 30, y=200, size = 7,label =
             paste0("p-value = ",mean(Goodhue.sd >= sd(goodhue))) )+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",size=2),
        text = element_text(size=24),
        axis.text = element_text(colour = "black",size = 24,face="bold"),
        axis.title = element_text(size = 30,face="bold"),
        axis.ticks.length=unit(.25, "cm"),
        axis.ticks = element_line(colour = "black", size = 1.5))+
  ylab("Frequency")+
  xlab(~ paste(T(y^{rep})))

( Test1.hist | Test2.hist) /
  (plot_spacer() + Test3.hist + plot_spacer() + plot_layout(widths = c(1,2,1))) +
  plot_annotation(tag_levels = 'A')
```