---

title: "Final Project"
output: pdf_document

---

## Name: Mitchell Wang

## I worked with:

**Click the "Knit" button in RStudio to knit this file to a pdf.**

---

## Data Wrangling

```
#load data
library(readxl)
Covid_Carleton<- read.csv("Covid Data - Carleton.csv")
```

```
#data wrangling -- separate terms
Covid_Carleton <- Covid_Carleton %>%
  filter(Term == "FA20"| Term == "WI21" | Term == "SP21" | Term == "FA21")  %>%
  mutate(term=recode(Term, "FA20" = 1,"WI21" = 2,"SP21" = 3,"FA21" = 4))
```

## EDA

```
#mutate variable, rename variable
Covid_Summary <- Covid_Carleton %>%
  mutate(Term = recode(term, "1"="FA20","2"="WI21","3"="SP21","4"="FA21")) %>%
  group_by(term) %>%
  summarise(Mean_Positive_Case = mean(Positive_Results),Mean_Positive_Rate_ = mean(Positivity_Rate))

# at most 4 decimal places
knitr::kable(Covid_Summary, digits = 3)
```

```
#standardize variable rice_standard
Covid_Carleton$Rice_Standard <- as.vector(scale(Covid_Carleton$Rice_County))
#EDA
ggplot(Covid_Carleton, aes(Rice_Standard, Positive_Results, group = 1)) +
  geom_point() + theme_bw() +
  geom_point() + geom_smooth(method = "lm", se = F) +
  xaxis_text(angle = 90, size = 6) +
  ggtitle("Relationship between Carleton Cases and Rice County Cases")

Covid_Carleton$Temp_Standard <- as.vector(scale(Covid_Carleton$Ave_temp))
ggplot(Covid_Carleton, aes(Temp_Standard, Positive_Results, group = 1)) +
  geom_point() + geom_smooth(method = "lm", se = F) +
  theme_bw() +
  xaxis_text(angle = 90, size = 6)+
  ggtitle("Relationship between Carleton Cases and Temperature")
```

```
#time series plot of positivity rate in different period
ggplot(Covid_Carleton, aes(Period, Positive_Results, group = 1)) +
  geom_point() +  geom_line() +
  theme_light() +
  xaxis_text(angle = 90, size = 6)+ggtitle("Covid Cases by Week")
```

```r
#boxplot of positive rate by term
Covid_Carleton  %>%
  group_by(Term) %>%
  ggplot(aes(Term, Positive_Results)) + geom_boxplot() +
  theme_light() +
  coord_flip()+
  ggtitle("Positive Case by Term")
```

## JAGS

```r
# JAGS code
modelString <-"
model{
## sampling
for (i in 1:N){
 y[i] ~ dbin(theta[term[i]], n[i])}

## priors
for (j in 1:M){
 theta[j] ~ dbeta(alpha, beta)}

alpha <- mu / pow(eta,2)
beta <- (1-mu) / pow(eta,2)
mu ~ dbeta(1,1)
eta <- exp(logeta)
logeta ~ dlogis(log(39), 1)
}"
```

```r
#data details
y <- Covid_Carleton$Positive_Results
n <- Covid_Carleton$Total_Tests
N <- length(y)
M <- length(unique(Covid_Carleton$Term))
term <- Covid_Carleton$term
the_data <- list(y = y, n=n, N=N,M=M, term = term)

# seed for JAGS
init = list(
            list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 987654),
            list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 987653),
            list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 987652)
            )

# MCMC draws
carleton_post <- run.jags(
  model = modelString,
  n.chains = 3,
  data = the_data,
  monitor = c("theta"),
  adapt = 1000,
  burnin = 2000,
  sample = 10000,
  silent.jags = TRUE  # Eliminates progress bar
)
```

## Diagnostics

```
#convert to mcmc object
post_mcmc <- as.data.frame(as.mcmc(carleton_post))
#trace plot
mcmc_trace(post_mcmc)
#acf plot
mcmc_acf(post_mcmc)
#overlayed density
mcmc_dens_overlay(carleton_post$mcmc)
#summary mcmc
summary(post_mcmc)
```

## Posterior Predictive

```
# JAGS code

modelString <-"
model{
## sampling
for (i in 1:N){
 y[i] ~ dbin(theta[term[i]], n[i])}

## priors
for (j in 1:M){
 theta[j] ~ dbeta(alpha, beta)}

alpha <- mu / pow(eta,2)
beta <- (1-mu) / pow(eta,2)
mu ~ dbeta(1,1)
eta <- exp(logeta)
logeta ~ dlogis(log(100), 1)
for (i in 1:N) {
   y_pred[i] ~ dbin(theta[term[i]], n[i])
}

}
"
```

```
# data details
y <- Covid_Carleton$Positive_Results
n <- Covid_Carleton$Total_Tests
N <- length(y)
M <- length(unique(Covid_Carleton$Term))
term <- Covid_Carleton$term
the_data <- list(y = y, n=n, N=N,M=M, term = term)

#seed
init = list(
            list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 987654),
            list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 987653),
            list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 987652)
            )
#MCMC draws
carleton_pred <- run.jags(
  model = modelString,
  n.chains = 3,
  data = the_data,
  monitor = c("y_pred"),
  adapt = 1000,
  burnin = 2000,
  sample = 10000,
  thin=5,
  silent.jags = TRUE  # Eliminates progress bar
)
```

```
#posterior predictive check
ppc_dens_overlay(y = Covid_Carleton$Positive_Results, yrep = carleton_pred$mcmc[[1]][1:100,])
```

## Inference

```
#make posterior inference plot
post_intervals <- mcmc_intervals_data(carleton_post$mcmc) %>%
  mutate(para = c("FA20","WI21","SP21","FA21"))
head(post_intervals)

slice(post_intervals, 1:4) %>%
  ggplot(
    aes(x = reorder(para, (m)), y = (m), ymin = (ll), ymax = (hh))) +
  geom_pointrange() +
  theme_light() +
  xaxis_text(angle = 0, size = 6) +
  theme(
    panel.grid.major.x = element_blank(),
    panel.grid.minor.x = element_blank()
)+
xlab("Term") + ylab("Positive Test Rate")
```

## Logistical Regression

```
#JAGS
modelString <-"
model {
## likelihood
for (i in 1:n){
   y[i] ~ dbin(p[i], N[i])
}
## priors and regression
for (i in 1:n){
   logit(p[i]) <- beta0 +beta1*rice[i]+beta2*temp[i]
   }
## hyperpriors
beta0 ~ dnorm(0, 0.0001)
beta1 ~ dnorm(0, 0.0001)
beta2 ~ dnorm(0, 0.0001)

}
"
# data details
Covid_Carleton <- Covid_Carleton %>%
  mutate(Rice_Standard = as.vector(scale(Rice_County)),
         temp_standard = as.vector(scale(Ave_temp)))


y <- Covid_Carleton$Positive_Results
n <- length(y)
N <- Covid_Carleton$Total_Tests
rice <- Covid_Carleton$Rice_Standard
temp <- Covid_Carleton$temp_standard

the_data <- list(y=y,n=n,N=N,rice=rice,temp=temp)

# seed
init = list(
            list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 987654),
            list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 987653),
            list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 987652)
            )

# MCMC draws
carleton_post <- run.jags(
  model = modelString,
  n.chains = 3,
  data = the_data,
  monitor = c("beta0","beta1","beta2","y_pred"),
  adapt = 1000,
  burnin = 2000,
  sample = 10000,
  thin=5,
  silent.jags = TRUE  # Eliminates progress bar
)
```

```
# convert to mcmc
post_mcmc <- as.data.frame(as.mcmc(carleton_post))
quantile(post_mcmc$beta0,c(0.05,0.95))
quantile(post_mcmc$beta1,c(0.05,0.95))
quantile(post_mcmc$beta2,c(0.05,0.95))
```

```
# diagnostics
#trace plot
mcmc_trace(post_mcmc)
#acf
mcmc_acf(post_mcmc)
#density plot
mcmc_dens_overlay(carleton_post$mcmc)
#summary
summary(post_mcmc)
```

## Posterior Predictive

```
#JAGS
modelString <-"
model {
## likelihood
for (i in 1:n){
   y[i] ~ dbin(p[i], N[i])
}
## priors and regression
for (i in 1:n){
   logit(p[i]) <- beta0 +beta1*rice[i]+beta2*temp[i]
   }
## hyperpriors
beta0 ~ dnorm(0, 0.0001)
beta1 ~ dnorm(0, 0.0001)
beta2 ~ dnorm(0, 0.0001)

for (i in 1:n) {
   y_pred[i] ~ dbin(p[i], N[i])
}

}
"
# MCMC draws
carleton_pred <- run.jags(
  model = modelString,
  n.chains = 3,
  data = the_data,
  monitor = c("y_pred"),
  adapt = 1000,
  burnin = 2000,
  sample = 10000,
  thin=5,
  silent.jags = TRUE  # Eliminates progress bar
)
```

```
#posterior preditive check
ppc_dens_overlay(y = Covid_Carleton$Positive_Results, yrep = carleton_pred$mcmc[[1]][1:100,])
```

## Residual Diagnostics

```
#residual of SLR
resids <- Covid_Carleton %>%
mutate(
beta0 = mean(post_mcmc$beta0),
beta1 = mean(post_mcmc$beta1),
beta2 = mean(post_mcmc$beta2),
phat_logit = beta0 + beta1 * Rice_Standard + beta2 * temp_standard,
phat = exp(phat_logit)/(1+exp(phat_logit)),
yhat = rbinom(39,Total_Tests, phat),
resid = Positive_Results - yhat
)
```

```
# residuals against fitted values
ggplot(data = resids, aes(x = yhat, y = resid)) + # fitted values
  geom_hline(yintercept = 0, linetype = 2, color = "blue") +
  geom_point() + theme_bw()
qqnorm(resids$resid)
qqline(resids$resid)

#against explanatory variable
ggplot(data = resids, aes(x = Rice_Standard, y = resid)) + # fitted values
  geom_hline(yintercept = 0, linetype = 2, color = "blue") +
  geom_point() + theme_bw()
```

```
#quantile of parameter
quantile(post_mcmc$beta2,c(0.05,0.95))
```

## Between instituiton comparison

### EDA

### model mu

```
#JAGS
modelString_mu <-"
model {
## likelihood
for (i in 1:N){
   y[i] ~ dbin(p[i], n[i])
}

## priors and regression
for (i in 1:N){
   p[i] ~ dbeta(a[i], b[i])
   a[i] <- phi[i] * mu[i]
   b[i] <- phi[i] * (1 - mu[i])
   logit(mu[i]) <- beta0 +beta1*type[i] +beta3*test_rate[i]
   phi[i] <- exp(logeta[i])
   logeta[i] ~ dlogis(logn, 1)

}
## hyperpriors

beta0 ~ dnorm(0, 0.0001)
beta1 ~ dnorm(0, 0.0001)
beta3 ~ dnorm(0, 0.0001)

## predictive
for (i in 1:N) {
   y_pred[i] ~ dbin(p[i], n[i])
}

}

"
```

```r
# seed
init = list(
            list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 987654),
            list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 987653),
            list(.RNG.name = "base::Wichmann-Hill", .RNG.seed = 987652)
            )



# data details
y <- cross$Positive_Results
n <- cross$Total_Tests
type <- cross$type_num
test_rate<- cross$Test_Rate
N <- length(y)
the_data2 <- list("y" = y, "n" = n, "N" = N, "type" = type, "test_rate" = test_rate,
                 "logn" = log(100))


#mcmc draws
posterior_mu <- run.jags(modelString_mu,
                 data = the_data2,
                 n.chains = 3,
                 monitor = c("beta0","beta1","beta3", "p", "y_pred"),
                 adapt = 1000,
                 burnin = 5000,
                 sample = 10000,
                 thin = 5,
                 inits = init, silent.jags = TRUE)
```

## model p standardized

## model diagnostics

```r
# diagnostics for model p standradized
mcmc_trace(posterior_p_stand$mcmc)
mcmc_acf(posterior_p_stand$mcmc)
mcmc_dens_overlay(posterior_p_stand$mcmc)
```

```r
# diagnostics for model mu
mcmc_trace(posterior_mu$mcmc)
mcmc_acf(posterior_mu$mcmc)
mcmc_dens_overlay(posterior_mu$mcmc)
```

```r
# gelman -- good
gelman.diag(posterior_mu$mcmc)

# geweke -- good
geweke.diag(posterior_mu$mcmc)

# effective sample size -- good
effectiveSize(posterior_mu$mcmc)

# pretty good
```

```
summary(posterior_mu$mcmc)
```

## inference plot

### observed rate vs posterior intervals

```
# intervals
post_intervals_mu <- mcmc_intervals_data(posterior_mu$mcmc, regex_pars = "p", prob_outer = 0.9)

# plot observed rate vs poterior intervals
ggplot(cross, mapping =aes(x = (cross$Positive_Results)/(cross$Total_Tests))) +
  geom_point(data.frame(post_intervals_mu[1:30, ]), mapping = aes(y = post_intervals_mu[1:30,  7] %>% un
  geom_linerange(data.frame(post_intervals_mu[1:30, ]), mapping = aes(ymin = post_intervals_mu[1:30,  5]
  geom_abline(slope = 1, intercept = 0, color = "dodgerblue")
```

## residual diagnostics

```
# residual diagnostics for model p standardized
post_p_stand<-as.mcmc(posterior_p_stand)
# post_p_stand[, 5:34]
resids_p_stand <- cross %>%
mutate(
   phat = colMeans(post_p_stand[, 5:34]),
   yhat = rbinom(30,cross$Total_Tests, phat),
   resid = Positive_Results - yhat
)

# vs fitted values
ggplot(data = resids_p_stand, aes(x = yhat, y = resid)) + # fitted values
  geom_hline(yintercept = 0, linetype = 2, color = "blue") +
  geom_point() + theme_bw()
qqnorm(resids_p_stand$resid)
qqline(resids_p_stand$resid)

# vs z_test rate
ggplot(data = resids_p_stand, aes(x = z_test_rate, y = resid)) + # fitted values
  geom_hline(yintercept = 0, linetype = 2, color = "blue") +
  geom_point() + theme_bw()


# vs z_enrol
ggplot(data = resids_p_stand, aes(x = z_enrol, y = resid)) + # fitted values
  geom_hline(yintercept = 0, linetype = 2, color = "blue") +
  geom_point() + theme_bw()

## qqplot of model p stand
qqnorm(resids_p_stand$resid)
qqline(resids_p_stand$resid)
```

```
# resiudal diagnostics for model mu
post_mu<-as.mcmc(posterior_mu)
# post_mu[, 4:33]
resids_mu <- cross %>%
```

```r
mutate(
   phat = colMeans(post_mu[, 4:33]),
   yhat = rbinom(30,cross$Total_Tests, phat),
   resid = Positive_Results - yhat
)

# vs fitted values
ggplot(data = resids_mu, aes(x = yhat, y = resid)) + # fitted values
  geom_hline(yintercept = 0, linetype = 2, color = "blue") +
  geom_point() + theme_bw()
qqnorm(resids_mu$resid)
qqline(resids_mu$resid)

# vs test rate
ggplot(data = resids_mu, aes(x = cross$Test_Rate, y = resid)) + # fitted values
  geom_hline(yintercept = 0, linetype = 2, color = "blue") +
  geom_point() + theme_bw()
```