MA 576 Final Project Code

Benjamin Draves

```r
setwd("~/Desktop/GLM_Final_Project/Mouse_Data")

#----------------------------------------------------------------------
#
#                    Bernoulli Simulation
#
#----------------------------------------------------------------------

library(igraph)
library(ggplot2)
library(reshape)
#set seed
set.seed(576)

#Define useful functions------------------------------------------------
format_global = function(x){
  tmp = outer(x, x, "paste")
  tmp = tmp[upper.tri(tmp)]
  split = function(a) as.numeric(unlist(strsplit(a, " ")))
  X = t(sapply(tmp, split))
  return(X)
}
getp = function(x) exp(x)/(1+exp(x))

#set up interesting sample graph----------------------------------------
n = 50
size.gp1 = 20
size.gp2 = n - size.gp1

#make covariates based on groups
x1 = rpois(n, 4)
x2 = c(rep(1, size.gp1),rep(2, size.gp2))

X1 = format_global(x1)
X2 = format_global(x2)

X = cbind(X1, X2)
colnames(X) = rep("", ncol(X))
rownames(X) = rep("", nrow(X))
#create probability matrix
combine = function(x){
  ret = ifelse(x[3] + x[4] == 2, -2, ifelse(x[3] +x[4] == 4 ,-1,0)) +
    +1/10*(x[1] + x[2])
  return(ret)
}
eta = apply(X, 1,combine)
hist(eta)

P = matrix(0, nrow = n, ncol = n)
P[upper.tri(P)] = pnorm(eta)
P[lower.tri(P)] = t(P)[lower.tri(P)]

#make pretty P plot
Pmelt = melt(P)
colnames(Pmelt) = c("Index","Index.","P")
ggplot(data = Pmelt, aes(x=Index, y=Index., fill=P)) +
  geom_tile()+
  scale_fill_gradient2(low = "white",
                       high = "black", mid = "grey",
                       midpoint = 0.5, limit = c(0,1))

#set up sample function
bern_sample = function(p){
  return(rbinom(1,1,p))
}
A_sample = function(P){
  A = apply(P,c(1,2),bern_sample)
  A[lower.tri(A)] = t(A)[lower.tri(A)]
  diag(A) = 0
  return(A)
}

#get sample and make some nice visuals
A = A_sample(P)

Amelt = melt(A)
colnames(Amelt) = c("Index","Index.","A")
ggplot(data = Amelt, aes(x=Index, y=Index., fill=A)) +
  geom_tile()+
  scale_fill_gradient2(low = "white",
                       high = "black", mid = "grey",
                       midpoint = 0.5, limit = c(0,1))

par(mfrow = c(1,1))
net = graph_from_adjacency_matrix(A, mode = "undirected", weighted = TRUE)
V(net)$color = c(rep("red", size.gp1),rep("blue", size.gp2))
V(net)$label = ""
V(net)$size = 5
E(net)$width = .3
E(net)$color = "#55555555"
E(net)$arrow.mode = 0
plot(net)

#set up logistic regression model-----------------------------
#set up response
A = as.matrix(as_adjacency_matrix(net))
Y = A[upper.tri(A, diag = F)]

#set up inter-nodal attributes
#group
group = ifelse(X[,3] == X[,4] & X[,4] == 2, 2, ifelse(X[,3] == X[,4] & X[,4] == 1, 1, 0))

#Construct model---------------------------------------------------

#homogenous group effect
df = data.frame(Response = Y, SameGroup = ifelse(X[,3] == X[,4], 1,0))
m1 = glm(Response~.,data = df, family = binomial)
summary(m1)
par(mfrow = c(2,2)); plot(m1)

#get prob predictions
eta_diff = coef(m1)[1]
eta_same = sum(coef(m1))

getp(eta_same)
getp(eta_diff)

Phat = matrix(0, nrow = n, ncol = n)
Phat[upper.tri(Phat)] = m1$fitted.values
```

```r
Phat[lower.tri(Phat)] = t(Phat)[lower.tri(Phat)]

#make pretty P plot
Pmelt = melt(Phat)
colnames(Pmelt) = c("Index","Index.","P")
ggplot(data = Pmelt, aes(x=Index, y=Index., fill=P)) +
  geom_tile()+
  scale_fill_gradient2(low = "white",
                       high = "black", mid = "grey",
                       midpoint = 0.5, limit = c(0,1))

#construct test for overdispersion
sigma2 = sum(residuals(m1, type = "pearson")^2/m1$df.residual)
teststat = sigma2 * m1$df.residual
cuttoff = qchisq(.95, m1$df.residual)
reject = teststat > cuttoff


#Construct model---------------------------------------------------

#inhomogenous group effect
df = data.frame(Response = Y, SameGroup = as.factor(group))
m2 = glm(Response~.,data = df, family = binomial)
summary(m2)
par(mfrow = c(2,2)); plot(m1)

Phat = matrix(0, nrow = n, ncol = n)
Phat[upper.tri(Phat)] = m1$fitted.values
Phat[lower.tri(Phat)] = t(Phat)[lower.tri(Phat)]

#construct test for overdispersion
sigma2 = sum(residuals(m2, type = "pearson")^2/m2$df.residual)
teststat = sigma2 * m2$df.residual
cuttoff = qchisq(.95, m2$df.residual)
reject = teststat > cuttoff



#make CIs
C = vcov(m2)
p0 = getp(coef(m2)[1])
p0l = getp(coef(m2)[1] - qnorm(0.975)*sqrt(C[1,1]))
p0r = getp(coef(m2)[1] + qnorm(0.975)*sqrt(C[1,1]))

p1 = getp(coef(m2)[1] + coef(m2)[2])
p1l = getp(coef(m2)[1] +coef(m2)[2] - qnorm(0.975)*sqrt(C[1,1] + C[2,2] + 2*C[1,2]))
p1r = getp(coef(m2)[1] +coef(m2)[2] + qnorm(0.975)*sqrt(C[1,1] + C[2,2] + 2*C[1,2]))

p2 = getp(coef(m2)[1] + coef(m2)[3])
p2l = getp(coef(m2)[1] +coef(m2)[3] - qnorm(0.975)*sqrt(C[1,1] + C[3,3] + 2*C[1,3]))
p2r = getp(coef(m2)[1] +coef(m2)[3] + qnorm(0.975)*sqrt(C[1,1] + C[3,3] + 2*C[1,3]))

c(p0l,p0r)
c(p1l,p1r)
c(p2l,p2r)

#include poisson covartiate---------------------------------------------
df = data.frame(Response = Y,
                SameGroup = as.factor(group),
                GlobalPois = as.numeric(X[,1]) +as.numeric(X[,2]))
m3 = glm(Response~.,data = df, family = binomial)
summary(m3)
```

```r
par(mfrow = c(2,2)); plot(m3)

Phat = matrix(0, nrow = n, ncol = n)
Phat[upper.tri(Phat)] = m3$fitted.values
Phat[lower.tri(Phat)] = t(Phat)[lower.tri(Phat)]

Pmelt = melt(Phat)
colnames(Pmelt) = c("Index","Index.","P")
ggplot(data = Pmelt, aes(x=Index, y=Index., fill=P)) +
  geom_tile()+
  scale_fill_gradient2(low = "white",
                       high = "black", mid = "grey",
                       midpoint = 0.5, limit = c(0,1))

#test for which model is best
anova(m1, m2,m3, test = "Chisq")


#-------------------------------------------------------------------------
#
#                    Poisson Simulation
#
#-------------------------------------------------------------------------

library(igraph)
library(ggplot2)
library(reshape)
library(MASS)
#set seed
set.seed(576)

#Define useful functions--------------------------------------------------
format_global = function(x){
  tmp = outer(x, x, "paste")
  tmp = tmp[upper.tri(tmp)]
  split = function(a) as.numeric(unlist(strsplit(a, " ")))
  X = t(sapply(tmp, split))
  return(X)
}

#set up interesting sample graph-----------------------------------------
n = 25
size.gp1 = 10
size.gp2 = n - size.gp1

#make covariates based on groups
x1 = rpois(n, 2)
x2 = c(rep(1, size.gp1),rep(2, size.gp2))

X1 = format_global(x1)
X2 = format_global(x2)

X = cbind(X1, X2)
colnames(X) = rep("", ncol(X))
rownames(X) = rep("", nrow(X))
#create probability matrix
combine = function(x){
  ret = ifelse(x[3] == x[4], 10, 0) + (x[1]+x[2])
  return(ret)
}
eta = apply(X, 1,combine)
hist(eta)
```

```r
L = matrix(0, nrow = n, ncol = n)
L[upper.tri(L)] = eta
L[lower.tri(L)] = t(L)[lower.tri(L)]

#make pretty Lambda plot
Lmelt = melt(L)
colnames(Lmelt) = c("Index","Index.","L")
ggplot(data = Lmelt, aes(x=Index, y=Index., fill=L)) +
  geom_tile()+
  scale_fill_gradient2(low = "white",
                       high = "black", mid = "grey",
                       midpoint = median(L), limit = c(0,max(L)))

#set up sample function
pois_sample = function(l){
  return(rpois(1,l))
}
A_sample = function(L){
  A = apply(L,c(1,2),pois_sample)
  A[lower.tri(A)] = t(A)[lower.tri(A)]
  diag(A) = 0
  return(A)
}

#get sample and make some nice visuals
A = A_sample(L)

Amelt = melt(A)
colnames(Amelt) = c("Index","Index.","W")
ggplot(data = Amelt, aes(x=Index, y=Index., fill=W)) +
  geom_tile()+
  scale_fill_gradient2(low = "white",
                       high = "black", mid = "grey",
                       midpoint = median(A), limit = c(0,max(A)))



net = graph_from_adjacency_matrix(A, mode = "undirected", weighted = TRUE)
V(net)$color = c(rep("red", size.gp1),rep("blue", size.gp2))
V(net)$label = ""
V(net)$size = 5
E(net)$width = E(net)$weight/median(E(net)$weight)
E(net)$color = "#55555555"
E(net)$arrow.mode = 0
dev.off()
plot(net)

#set up poisson regression model-----------------------------
#set up response
A = as.matrix(as_adjacency_matrix(net, attr = 'weight'))
Y = A[upper.tri(A, diag = F)]

#set up inter-nodal attributes
#group
group = ifelse(X[,3] == X[,4], 1, 0)

#Construct model---------------------------------------------------
#group only model
df = data.frame(Response = Y, SameGroup = group)
m4 = glm(Response~.,data = df, family = poisson)
summary(m4)
```

```r
#full model
df = data.frame(Response = Y, SameGroup = group, Cont = X[,1] + X[,2])
m5 = glm(Response~.,data = df, family = poisson)
summary(m5)

#construct test for overdispersion
sigma2 = sum(residuals(m5, type = "pearson")^2/m5$df.residual)
teststat = sigma2 * m5$df.residual
cuttoff = qchisq(.95, m5$df.residual)
reject = teststat > cuttoff
reject

#quassi poisson model
m6 = glm(Response~.,data = df, family = quasipoisson)
summary(m6)

#negative binomial model
m7 = glm.nb(Response~.,data = df, link = log)
summary(m7)

#construct test for overdispersion
sigma2 = sum(residuals(m7, type = "pearson")^2/m7$df.residual)
teststat = sigma2 * m7$df.residual
cuttoff = qchisq(.95, m7$df.residual)
reject = teststat > cuttoff
reject

#compare models
pchisq(-2*(logLik(m6) - logLik(m7)), 1, lower.tail = FALSE)

#get model estimates
Lhat = matrix(0, nrow = n, ncol = n)
Lhat[upper.tri(Lhat)] = m7$fitted.values
Lhat[lower.tri(Lhat)] = t(Lhat)[lower.tri(Lhat)]

Lmelt = melt(Lhat)
colnames(Lmelt) = c("Index","Index.","L")
ggplot(data = Lmelt, aes(x=Index, y=Index., fill=L)) +
  geom_tile()+
  scale_fill_gradient2(low = "white",
                       high = "black", mid = "grey",
                       midpoint = median(L), limit = c(0,max(L)))



#-----------------------------------------------------------------------
#
#                   Mouse Connectome
#
#-----------------------------------------------------------------------

#Define useful functions------------------------------------------------
format_global = function(x){
  n = length(x)
  ind = combn(1:n, 2)
  X = matrix(NA, ncol = 2, nrow = n*(n-1)/2)
  for(i in 1:(n*(n-1)/2)){
    X[i,] = c(x[ind[1,i]], x[ind[2,i]])
  }
  return(X)
}
```

```r
#Read in/Format data-------------------------------------------------

#brain
brain = as.matrix(read.csv("N54859.csv", header = F))
n = nrow(brain)
colnames(brain) = rownames(brain) = 1:n
head(brain)

brain2 = matrix(0, nrow = nrow(brain), ncol = ncol(brain))
for(i in 1:nrow(brain2)){
  ind = as.numeric(which(brain[i,] == max(brain[i,])))
  brain2[i,ind] = brain[i, ind]
}


#covariates
covar = read.csv("first_page.csv")[,-c(1:2,9:11)]
covar = covar[-nrow(covar),]
for(i in 1:ncol(covar)){
  covar[,i] = as.character(droplevels(covar[,i]))
}
covar = as.matrix(covar)

#Make pretty visuals-------------------------------------------------
net = graph_from_adjacency_matrix(brain2,weighted = TRUE)
V(net)$color = c(rep("red", nrow(brain)/2), rep("blue", nrow(brain)/2))
V(net)$label = ""
V(net)$size = 2
E(net)$width = min(E(net)$weight, 100)/50
E(net)$color = "#55555555"
E(net)$arrow.mode = 0
dev.off()
plot(net, layout = layout_on_sphere)


brain2 = log(brain + 1)
brainmelt = melt(brain2)
colnames(brainmelt) = c("Index","Index.","W")
ggplot(data = brainmelt, aes(x=Index, y=Index., fill=W)) +
  geom_tile()+
  scale_fill_gradient2(low = "white",
                       high = "black", mid = "grey",
                       midpoint = median(brain2), limit = c(0,max(brain2)))

#Build homogenous models-------------------------------------------------
#build response
Y = brain[upper.tri(brain)]

#build covariates
hemi = format_global(covar[,1])
level1 = format_global(covar[,2])
level2 = format_global(covar[,3])
level3 = format_global(covar[,4])
level4 = format_global(covar[,5])
subdivision = format_global(covar[,6])

hemi_group = ifelse(hemi[,1] == hemi[,2], 1, 0)
level1_group = ifelse(level1[,1] == level1[,2], 1, 0)
level2_group = ifelse(level2[,1] == level2[,2], 1, 0)
level3_group = ifelse(level3[,1] == level3[,2], 1, 0)
level4_group = ifelse(level4[,1] == level4[,2], 1, 0)
subdivision_group = ifelse(subdivision[,1] == subdivision[,2], 1, 0)
```

```r
X = cbind(hemi, level1 , level2, level3, level4, subdivision)
nested = numeric(nrow(X))
for(i in 1:nrow(X)){
  nested[i] = ifelse(X[i,1] != X[i,2], 0,
                     ifelse(X[i,3] != X[i,4], 1,
                            ifelse(X[i,5] != X[i,6], 2,
                                   ifelse(X[i,7] != X[i,8], 3,
                                          ifelse(X[i,9] != X[i,10], 4,
                                                 ifelse(X[i,11] != X[i,12], 5,6))))))
}

df = data.frame(Response = Y,
                hemisphere = hemi_group,
                level1 = level1_group,
                level2 = level2_group,
                level3 = level3_group,
                level4 = level4_group,
                subdivision = subdivision_group,
                nested = as.factor(nested))

#construct nested varible
m1 = glm(Response~hemisphere + level1 + level2 + level3, data = df, family = poisson)
m2 = glm(Response~hemisphere + level1 + level2 + level3 + level4, data = df, family = poisson)
m3 = glm(Response~hemisphere + level1 + level2 + level3 +level4 + subdivision, data = df, family = poisson)
#
anova(m1, m2, m3, m4, test = "Chisq")
sigma2 = sum(residuals(m3, "pearson")^2)/m3$df.residual
summary(m3, sigma2)

#fix overdispersion by using negative binomial with log link
#test difference between full Poisson and full NegBin
nb_full = glm.nb(Response ~ hemisphere +level1+ level2 + level3 +level4 +subdivision, data = df, link = log)
pchisq(2 * (logLik(nb_full) - logLik(m3)), df = 1, lower.tail = FALSE)
#p value is indistingishable from zero suggesting that the neg bin is a signficantly better fit
#we only consider nb models from here on

nb1 = glm.nb(Response ~ hemisphere, data = df, link = log)
nb2 = glm.nb(Response ~ hemisphere + level3 +level4, data = df, link = log)
nb3 = glm.nb(Response ~ hemisphere +level1+ level2 + level3 +level4, data = df, link = log)
nb4 = glm.nb(Response ~ hemisphere +level1+ level2 + level3 +level4 +subdivision, data = df, link = log)
anova(nb1, nb2, nb3,nb4, test = "Chisq")

x = c(AIC(nb1),AIC(nb1), AIC(nb3), AIC(nb4))
summary(nb4)
#subdivision appears to be too fine getting rid of it
summary(nb3) # best model by AIC and ANOVA standards

est = cbind(Estimates = coef(nb3), confint(nb3))
est
#going to consider adding nonhomogenous for groups 1 and 2
#as their effects seem small

#Build inhomogenous models---------------------------------------------------
#build response
Y = brain[upper.tri(brain)]

#make inhomogenous group labels
make_group_labels = function(x){
  X = format_global(x)
  names = unique(x)

  ret = numeric(nrow(X))
  for(i in 1:nrow(X)){
    if(X[i,1] == X[i,2]){
      ret[i] = names[which(names == X[i,1])]
    }else{
      ret[i] = 0
    }
  }
  return(ret)
}


hemi = make_group_labels(covar[,1])
level1 = make_group_labels(covar[,2])
level2 = make_group_labels(covar[,3])
level3 = make_group_labels(covar[,4])
level4 = make_group_labels(covar[,5])
subdivision = make_group_labels(covar[,6])


#g
df = data.frame(Response = Y,
                hemisphere = as.factor(hemi),
                level1 = as.factor(level1),
                level2 = level2_group,
                level3 = level3_group,
                level4 = level4_group
)

nb1 = glm.nb(Response ~(. - degree), data = df, link = log)
summary(nb1)
nb2 = glm.nb(Response ~., data = df, link = log)
summary(nb2)

anova(nb1, nb2, test = "Chisq")

#get model estimates
Phat = matrix(0, nrow = n, ncol = n)
Phat[upper.tri(Phat)] = m2$fitted.values
Phat[lower.tri(Phat)] = t(Phat)[lower.tri(Phat)]

Pmelt = melt(Phat)
colnames(Pmelt) = c("Index","Index.","Count")
ggplot(data = Pmelt, aes(x=Index, y=Index., fill=Count)) +
  geom_tile()+
  scale_fill_gradient2(low = "white",
                       high = "black", mid = "grey",
                       midpoint = median(Phat), limit = c(0,max(Phat)))



#final model-------------------------------------------------------
df = data.frame(Response = Y,
                hemisphere = as.factor(hemi),
                level1 = as.factor(level1),
                level2 = level2_group,
                level3 = level3_group,
                level4 = level4_group,
                level5 = subdivision_group
)

model = glm.nb(Response ~., data = df, link = log)
model$theta + qnorm(c(0.025, 0.975)) * model$SE.theta
```