Technical Report for Robust model-based estimation for binary outcomes in genomics studies

Suyoung Park, Alexander E. Lipka, Daniel J. Eck

August 31, 2021

- 1. Introduction
  - Library
- 2. Visualizing Complete Separation
  - Data Configuration (Figure 1)
  - Log Likelihood Values of Logistic Model (Figure 2)
  - One-sided Confindence Interval
- 3. Results
  - 3.1. Agresti Complete Separation Example
  - 3.2. Agresti Quasi-Complete Separation Example
  - 3.3. Quadratic Example
  - 3.4. Endometrial Example
  - 3.5. Maize Example

1. Introduction

This technical report demonstrates how to reproduce all figures and results we used for "Robust model-based estimation for binary outcomes in genomics studies." In Section 2, we visualize the complete separation issue with Agresti's example. In Section 3, we compare the performance of `glmdr`, `bayesglm`, `brglm2` / `logistf` and `OLS` models for five examples we tested.

Library

We used R version 3.6.1 and the required R packages for `glmdr` is `nloptr` version 1.2.2.2. To compare its performance, we considered `arm` version 1.11-1, `brglm2` 0.7.0, `logistf` version 1.23 and `stats` version 3.6.1. To determine the optimal cut-off for the logistic regression, we used `PresenceAbsence` version 1.1.9, and to compute the Wilson intervals, we used `binom` version 1.1-1. For visualization, data wrangling and experiments, we used `ggplot2` version 3.3.3, `gridExtra` version 2.3, `latex2exp` version 0.4.0, `foreach` version 1.4.7, `doParallel` version 1.0.15, `readxl` version 1.3.1, and `tidyverse` version 1.2.1.

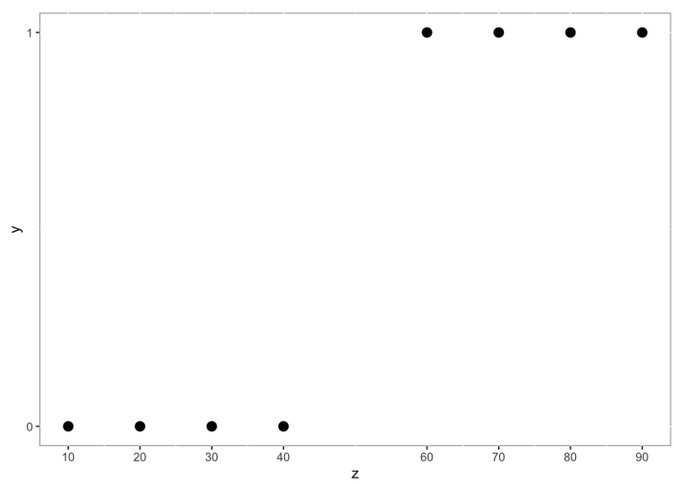
```
library(glmdr)
library(arm) #package for bayesglm
library(brglm2)
library(logistf)
library(stats)
library(binom)
library(PresenceAbsence)
library(readxl)
library(tidyverse)
library(ggplot2)
library(gridExtra)
library(foreach)
library(doParallel)
library(latex2exp)
```

2. Visualizing Complete Separation

In this section, we show the code to reproduce the Figure 1, 2, and 3 in our paper. We take the example from the Section 6.5.1 of Agresti[2013].

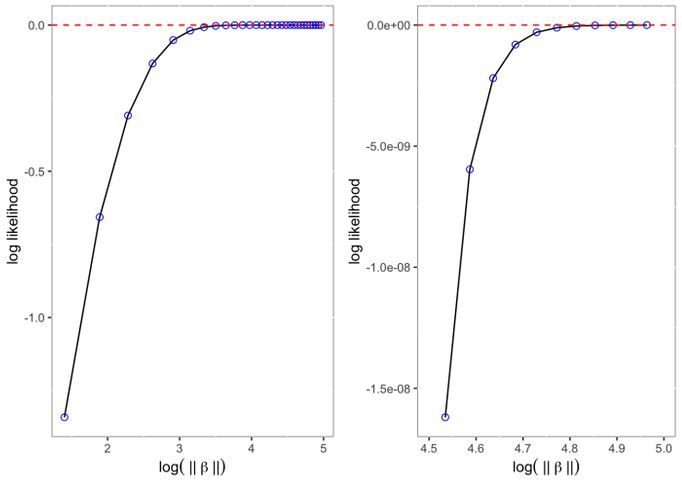
Data Configuration (Figure 1)

```
x <- c(10,20,30,40,60,70,80,90)
y <- c(0,0,0,0,1,1,1,1)
ggplot(data=as.data.frame(cbind(x,y)),aes(x=x,y=y)) +
  geom_point(size=3) +
  theme(panel.background = element_rect(fill = "white", colour = "grey50"))+
  scale_x_continuous(breaks = x)+
  labs(x="z",y="y")+
  scale_y_continuous(breaks = c(0.0,1.0))
```



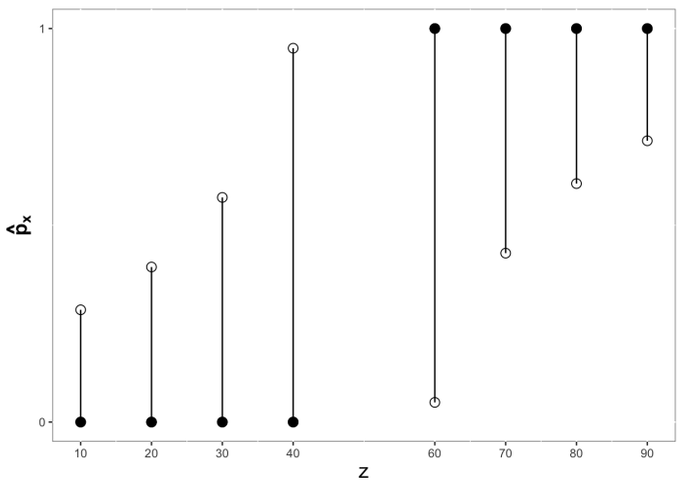
Log Likelihood Values of Logistic Model (Figure 2)

```
asymptote <- t(sapply(1:30, function(iter){
  m1 <- suppressWarnings(glm(y ~ x, family = "binomial", control = list(maxit = iter, epsilon = 1e-50)))
  c(log(sqrt(crossprod(coef(m1)))),logLik(m1))
  }))
asymptote <- as.data.frame(asymptote)
asymptote$V3 <- as.factor(ifelse(asymptote$V1 >=4.5,1,0))
plot1 <- ggplot(asymptote,aes(x = V1, y = V2)) +
  labs(x= expression(log(~"||"~beta~"||")), y= "log likelihood") +
  geom_line(col = "black") +
  geom_point(pch=1,col="blue",size=2) +
  geom_abline(intercept = 0, slope = 0, lty = 2, col = "red") +
  theme(panel.background = element_rect(fill = "white", colour = "grey50"))

plot2 <- ggplot(asymptote[asymptote$V3 == 1,],aes(x = V1, y = V2)) +
  labs(x= expression(log(~"||"~beta~"||")), y= "log likelihood") +
  geom_line(col = "black") +
  xlim(4.5,5.0)+
  geom_point(pch=1,col="blue",size=2) +
  geom_abline(intercept = 0, slope = 0, lty = 2, col = "red") +
  theme(panel.background = element_rect(fill = "white", colour = "grey50"))

grid.arrange(plot1,plot2, ncol=2)
```



One-sided Confindence Interval

```
glmdr_mod <- glmdr(y~x,family="binomial")
mod_CI <- inference(glmdr_mod)
ggplot(data=as.data.frame(cbind(x,y,mod_CI)),aes(x=x,y=y)) +
  geom_point(size=3) +
  geom_point(aes(x=x, y=lower),size=3,pch=1) +
  geom_point(aes(x=x, y=upper),size=3,pch=1) +
  geom_point(size=3) +
  geom_segment(aes(x = x, y = y, xend = x, yend = lower)) +
  geom_segment(aes(x = x, y = y, xend = x, yend = upper)) +
  theme(panel.background = element_rect(fill = "white", colour = "grey50"), axis.title=element_text(size=15))+
  labs(x="z", y= TeX('$\\textbf{\\hat{p}_x}$')) +
  scale_x_continuous(breaks = x)+
  scale_y_continuous(breaks = c(0.0,1.0))
```



3. Results

In this section, we display the code to reproduce the results section in the paper. The order follows 1) wrangling data, 2) inference and 3) prediction.

3.1. Agresti Complete Separation Example

Data Wrangling

```
x <- c(10,20,30,40,60,70,80,90)
y <- c(0,0,0,0,1,1,1,1)
```

Inference

```
glmdr_mod <- glmdr(y~x,family="binomial")
glmdr_pred <- predict(glmdr_mod$om,type="response",se.fit=TRUE)
glmdr_CI <- inference(glmdr_mod)
glmdr_length <- sum(as.matrix(glmdr_CI) %*% c(-1,1))

bayes_mod <- bayesglm(y~x,family="binomial")
bayes_pred <- predict(bayes_mod,type="response",se.fit=TRUE)
bayes_CI <- binom.confint(bayes_pred$fit,n=1,methods="wilson")[,c(5,6)]
bayes_length <- sum(as.matrix(bayes_CI) %*% c(-1,1))

brglm_mod <- glm(y~x, family = "binomial", method = "brglmFit", type = "MPL_Jeffreys")
brglm_pred <- predict(brglm_mod,type="response",se.fit=TRUE)
brglm_CI <- binom.confint(predict(brglm_mod,type="response"),n=1,methods="wilson")[,c(5,6)]
brglm_length <- sum(as.matrix(brglm_CI) %*% c(-1,1))

lm_mod <- lm(y~x)
lm_pred <- (predict(lm_mod,se.fit = TRUE))
lm_pred$fit <- ifelse(lm_pred$fit<=0, 0,lm_pred$fit)
lm_pred$fit <- ifelse(lm_pred$fit>=1, 1,lm_pred$fit)
lm_CI <- binom.confint(lm_pred$fit,n=1,methods="wilson")[,c(5,6)]
lm_length <- sum(as.matrix(lm_CI) %*% c(-1,1))
```

```
#In-sample Accuracy
cbind("glmdr"=mean(as.numeric(glmdr_pred$fit >= 0.5) == y),
"bayesglm"=mean(as.numeric(bayes_pred$fit >= 0.5) == y),
"brglm2"=mean(as.numeric(brglm_pred$fit >= 0.5) == y),
"OLS"=mean(as.numeric(lm_pred$fit >= 0.5) == y))
```

```
##      glmdr bayesglm brglm2 OLS
## [1,]     1        1      1   1
```

```
#Average Length of Confidence Intervals
cbind(glmdr_length,bayes_length,brglm_length,lm_length) / length(x)
```

```
##      glmdr_length bayes_length brglm_length lm_length
## [1,]    0.5500288    0.8284682    0.8350299  0.829067
```

Prediction

```
numCores <- detectCores()
registerDoParallel(numCores)
n_sample <- length(x)
y_idx <- which(names(glmdr_mod$om$model) == paste(formula(glmdr_mod$om))[2])
dat <- glmdr_mod$om$model
#glmdr
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  glmdr_mod <- glmdr(y~x, data=training,family="binomial")
  ret2 <- predict(glmdr_mod,newdata = testing_X, alpha=0.05,crit="AICc")
  ret2
}
total_mat_glmdr_test <- do.call(c,ret)
```

```
#bayesglm
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  bayesglm_mod <- bayesglm(y~x, data=training, family="binomial")
  ret2 <- predict(bayesglm_mod, newdata= testing_X, type="response")
  ret2
}
total_mat_bayes_test <- do.call(c,ret)
```

```
#brglm2
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  brglm_mod <- glm(y~x, family = binomial(logit), data=training,  method = "brglmFit", type = "MPL_Jeffreys")
  ret2 <- predict(brglm_mod,newdata=testing_X,type="response")
  ret2
}
total_mat_brglm_test <- do.call(c,ret)
```

```
#OLS
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  lm_mod <- lm(y~x,data=training, family="binomial")
  ret2 <- predict(lm_mod, newdata= testing_X, type="response")
  ret2
}
total_mat_linear_test <- do.call(c,ret)
total_mat_linear_test <- ifelse(total_mat_linear_test>1,1,total_mat_linear_test)
total_mat_linear_test <- ifelse(total_mat_linear_test<0,0,total_mat_linear_test)
```

```
thresholds <- optimal.thresholds(data.frame("ID"=length(x),"observed"=y,"glmdr"=total_mat_glmdr_test,"bayesglm"=t
otal_mat_bayes_test,"brglm2"=total_mat_brglm_test,"OLS"=total_mat_linear_test),opt.methods="MaxPCC")
thresholds
```

Method <fct>	glmdr <dbl>	bayesglm <dbl>	brglm2 <dbl>	OLS <dbl>
MaxPCC	0.5	0.5	0.5	0.5

1 row

```
#Out-of-sample accuracy
cbind("glmdr"=mean(as.numeric(total_mat_glmdr_test >= thresholds[2]$glmdr) == y),
"bayesglm"=mean(as.numeric(total_mat_bayes_test >= thresholds[3]$bayesglm) == y),
"brglm2"=mean(as.numeric(total_mat_brglm_test >= thresholds[4]$brglm2) == y),
"OLS"=mean(as.numeric(total_mat_linear_test >= thresholds[5]$OLS) == y))
```

```
##      glmdr bayesglm brglm2 OLS
## [1,]     1        1      1   1
```

```
#Prediction Interval
glmdr_PI <- sum(apply(binom.confint(total_mat_glmdr_test,1,methods="wilson")[,c(5,6)],1,diff))
bayes_PI <- sum(apply(binom.confint(total_mat_bayes_test,1,methods="wilson")[,c(5,6)],1,diff))
brglm_PI <- sum(apply(binom.confint(total_mat_brglm_test,1,methods="wilson")[,c(5,6)],1,diff))
linear_PI <- sum(apply(binom.confint(total_mat_linear_test,1,methods="wilson")[,c(5,6)],1,diff))
cbind("glmdr"=glmdr_PI,"bayesglm"=bayes_PI,"brglm2"=brglm_PI,"OLS"=linear_PI) / length(x)
```

```
##          glmdr  bayesglm    brglm2       OLS
## [1,] 0.8219481 0.8388592 0.8429006 0.8328665
```

3.2. Agresti Quasi-Complete Separation Example

Data Wrangling

```
x <- c(10,20,30,40,60,70,80,90,50,50)
y <- c(0,0,0,0,1,1,1,1,1,0)
```

Inference

```
glmdr_mod <- glmdr(y~x,family="binomial")
glmdr_pred <- c(predict(glmdr_mod$om,type="response",se.fit=TRUE)$fit[!glmdr_mod$linearity],
                predict(glmdr_mod$lcm,type="response"))
glmdr_CI <- inference(glmdr_mod)
glmdr_length <- sum(as.matrix(glmdr_CI) %*% c(-1,1))

bayes_mod <- bayesglm(y~x,family="binomial")
bayes_pred <- predict(bayes_mod,type="response",se.fit=TRUE)
bayes_CI <- binom.confint(bayes_pred$fit[!glmdr_mod$linearity],n=1,methods="wilson")[,c(5,6)]
bayes_length <- sum(as.matrix(bayes_CI) %*% c(-1,1))

brglm_mod <- glm(y~x, family = "binomial", method = "brglmFit", type = "MPL_Jeffreys")
brglm_pred <- predict(brglm_mod,type="response",se.fit=TRUE)
brglm_CI <- binom.confint(predict(brglm_mod,type="response")[!glmdr_mod$linearity],n=1,methods="wilson")[,c(5,6)]
brglm_length <- sum(as.matrix(brglm_CI) %*% c(-1,1))

lm_mod <- lm(y~x)
lm_pred <- (predict(lm_mod,se.fit = TRUE))
lm_pred$fit <- ifelse(lm_pred$fit<=0, 0,lm_pred$fit)
lm_pred$fit <- ifelse(lm_pred$fit>=1, 1,lm_pred$fit)
lm_CI <- binom.confint(lm_pred$fit[!glmdr_mod$linearity],n=1,methods="wilson")[,c(5,6)]
lm_length <- sum(as.matrix(lm_CI) %*% c(-1,1))
```

```
#In-sample Accuracy
cbind("glmdr"=mean(as.numeric(glmdr_pred >= 0.5) == y),
"bayesglm"=mean(as.numeric(bayes_pred$fit >= 0.5) == y),
"brglm2"=mean(as.numeric(brglm_pred$fit >= 0.5) == y),
"OLS"=mean(as.numeric(lm_pred$fit >= 0.5) == y))
```

```
##      glmdr bayesglm brglm2 OLS
## [1,]   0.9      0.9    0.9 0.9
```

```
#Average length of Confidence Interval
cbind(glmdr_length,bayes_length,brglm_length,lm_length) / sum(!glmdr_mod$linearity)
```

```
##      glmdr_length bayes_length brglm_length lm_length
## [1,]    0.3078401    0.8271118    0.8305578  0.829067
```

Prediction

```
n_sample <- length(x)
y_idx <- which(names(glmdr_mod$om$model) == paste(formula(glmdr_mod$om))[2])
dat <- glmdr_mod$om$model
#glmdr
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  glmdr_mod <- glmdr(y~x, data=training,family="binomial")
  ret2 <- predict(glmdr_mod,newdata = testing_X, alpha=0.05,crit="AICc")
  ret2
}
total_mat_glmdr_test <- do.call(c,ret)
```

```
#bayesglm
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  bayesglm_mod <- bayesglm(y~x, data=training, family="binomial")
  ret2 <- predict(bayesglm_mod, newdata= testing_X, type="response")
  ret2
}
total_mat_bayes_test <- do.call(c,ret)
```

```
#logistf
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  brglm_mod <- glm(y~x, family = binomial(logit), data=training,  method = "brglmFit", type = "MPL_Jeffreys")
  ret2 <- predict(brglm_mod,newdata=testing_X,type="response")
  ret2
}
total_mat_brglm_test <- do.call(c,ret)
```

```
#OLS
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  lm_mod <- lm(y~x,data=training, family="binomial")
  ret2 <- predict(lm_mod, newdata= testing_X, type="response")
  ret2
}
total_mat_linear_test <- do.call(c,ret)
total_mat_linear_test <- ifelse(total_mat_linear_test <0,0,total_mat_linear_test)
total_mat_linear_test <- ifelse(total_mat_linear_test >1,1,total_mat_linear_test)
```

```
thresholds = optimal.thresholds(data.frame("ID"=1:length(x),"observed"=y,"glmdr"=total_mat_glmdr_test,"bayesglm"=
total_mat_bayes_test,"brglm2"=total_mat_brglm_test,"OLS"=total_mat_linear_test),opt.methods="MaxPCC")
thresholds
```

Method <fct>	glmdr <dbl>	bayesglm <dbl>	brglm2 <dbl>	OLS <dbl>
MaxPCC	0.5	0.5	0.5	0.5

1 row

```
#Out-of-sample accuracy
cbind("glmdr"=mean(as.numeric(total_mat_glmdr_test >= thresholds[2]$glmdr) == y),
"bayesglm"=mean(as.numeric(total_mat_bayes_test >= thresholds[3]$bayesglm) == y),
"brglm2"=mean(as.numeric(total_mat_brglm_test >= thresholds[4]$brglm2) == y),
"OLS"=mean(as.numeric(total_mat_linear_test >= thresholds[5]$OLS) == y))
```

```
##      glmdr bayesglm brglm2 OLS
## [1,]   0.8      0.8    0.8 0.8
```

```
#Prediction Interval
glmdr_PI <- sum(apply(binom.confint(total_mat_glmdr_test,1,methods="wilson")[,c(5,6)],1,diff))
bayes_PI <- sum(apply(binom.confint(total_mat_bayes_test,1,methods="wilson")[,c(5,6)],1,diff))
brglm_PI <- sum(apply(binom.confint(total_mat_brglm_test,1,methods="wilson")[,c(5,6)],1,diff))
linear_PI <- sum(apply(binom.confint(total_mat_linear_test,1,methods="wilson")[,c(5,6)],1,diff))
cbind("glmdr"=glmdr_PI,"bayesglm"=bayes_PI,"brglm2"=brglm_PI,"OLS"=linear_PI) / length(x)
```

```
##          glmdr  bayesglm    brglm2       OLS
## [1,] 0.8592767 0.8446978 0.8467347 0.8437463
```

3.3. Quadratic Example

Data Wrangling

```
data(quadratic)
```

Inference

```
glmdr_mod <- glmdr(y~x+I(x^2),data=quadratic,family="binomial")
glmdr_pred <- predict(glmdr_mod$om,type="response",se.fit=TRUE)
glmdr_CI <- inference(glmdr_mod)
glmdr_length <- sum(as.matrix(glmdr_CI) %*% c(-1,1))

bayes_mod <- bayesglm(y~x+I(x^2),data=quadratic,family="binomial")
bayes_pred <- predict(bayes_mod,type="response",se.fit=TRUE)
bayes_CI <- binom.confint(bayes_pred$fit,n=1,methods="wilson")[,c(5,6)]
bayes_length <- sum(as.matrix(bayes_CI) %*% c(-1,1))

brglm_mod <- glm(y~x+I(x^2),data=quadratic,family = "binomial", method = "brglmFit", type = "MPL_Jeffreys")
brglm_pred <- predict(brglm_mod,type="response",se.fit=TRUE)
brglm_CI <- binom.confint(predict(brglm_mod,type="response"),n=1,methods="wilson")[,c(5,6)]
brglm_length <- sum(as.matrix(brglm_CI) %*% c(-1,1))

lm_mod <- lm(y~x+I(x^2),data=quadratic)
lm_pred <- (predict(lm_mod,se.fit = TRUE))
lm_pred$fit <- ifelse(lm_pred$fit<=0, 0,lm_pred$fit)
lm_pred$fit <- ifelse(lm_pred$fit>=1, 1,lm_pred$fit)
lm_CI <- binom.confint(lm_pred$fit,n=1,methods="wilson")[,c(5,6)]
lm_length <- sum(as.matrix(lm_CI) %*% c(-1,1))
```

```
#In-sample Accuracy
cbind("glmdr"=mean(as.numeric(glmdr_pred$fit >= 0.5) == quadratic$y),
"bayesglm"=mean(as.numeric(bayes_pred$fit >= 0.5) == quadratic$y),
"brglm2"=mean(as.numeric(brglm_pred$fit >= 0.5) == quadratic$y),
"OLS"=mean(as.numeric(lm_pred$fit >= 0.5) == quadratic$y))
```

```
##      glmdr bayesglm brglm2 OLS
## [1,]     1        1      1 0.9
```

```
#Average length of Confidence Interval
cbind(glmdr_length,bayes_length,brglm_length,lm_length) / sum(!glmdr_mod$linearity)
```

```
##      glmdr_length bayes_length brglm_length lm_length
## [1,]    0.1994554    0.8234644    0.8109742 0.8587379
```

Prediction

```
n_sample <- nrow(quadratic)
y_idx <- which(names(glmdr_mod$om$model) == paste(formula(glmdr_mod$om))[2])
dat <- glmdr_mod$om$model
#glmdr
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  glmdr_mod <- glmdr(y~x+I(x^2),data=training, family="binomial")
  ret2 <- predict(glmdr_mod,newdata = testing_X, alpha=0.05,crit="AICc")
  ret2
}
total_mat_glmdr_test <- do.call(c,ret)
```

```
#bayesglm
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  bayesglm_mod <- bayesglm(y~x+I(x^2),data=training, family="binomial")
  ret2 <- predict(bayesglm_mod, newdata= testing_X, type="response")
  ret2
}
total_mat_bayes_test <- do.call(c,ret)
```

```
#logistf
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  brglm_mod <- glm(y~x+I(x^2), family = binomial(logit), data=training,  method = "brglmFit", type = "MPL_Jeffrey
s")
  ret2 <- predict(brglm_mod,newdata=testing_X,type="response")
  ret2
}
total_mat_brglm_test <- do.call(c,ret)
```

```
#OLS
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  lm_mod <- lm(y~x+I(x^2),data=training)
  ret2 <- predict(lm_mod, newdata= testing_X, type="response")
  ret2
}
total_mat_linear_test <- do.call(c,ret)
total_mat_linear_test <- ifelse(total_mat_linear_test <0,0,total_mat_linear_test)
total_mat_linear_test <- ifelse(total_mat_linear_test >1,1,total_mat_linear_test)
```

```
thresholds = optimal.thresholds(data.frame("ID"=1:nrow(quadratic),"observed"=quadratic$y,"glmdr"=total_mat_glmdr_
test,"bayesglm"=total_mat_bayes_test,"brglm2"=total_mat_brglm_test,"OLS"=total_mat_linear_test),opt.methods="MaxP
CC")
thresholds
```

Method	glmdr	bayesglm	brglm2	OLS
<fct>	<dbl>	<dbl>	<dbl>	<dbl>
MaxPCC	0.5	0.57	0.62	0.575

1 row

```
#Out-of-sample accuracy
cbind("glmdr"=mean(as.numeric(total_mat_glmdr_test >= thresholds[2]$glmdr) == quadratic$y),
"bayesglm"=mean(as.numeric(total_mat_bayes_test >= thresholds[3]$bayesglm) == quadratic$y),
"brglm2"=mean(as.numeric(total_mat_brglm_test >= thresholds[4]$brglm2) == quadratic$y),
"OLS"=mean(as.numeric(total_mat_linear_test >= thresholds[5]$OLS) == quadratic$y))
```

```
##          glmdr  bayesglm brglm2 OLS
## [1,] 0.9333333 0.9333333      1 0.9
```

```
#Prediction Interval
glmdr_PI <- sum(apply(binom.confint(total_mat_glmdr_test,1,methods="wilson")[,c(5,6)],1,diff))
bayes_PI <- sum(apply(binom.confint(total_mat_bayes_test,1,methods="wilson")[,c(5,6)],1,diff))
brglm_PI <- sum(apply(binom.confint(total_mat_brglm_test,1,methods="wilson")[,c(5,6)],1,diff))
linear_PI <- sum(apply(binom.confint(total_mat_linear_test,1,methods="wilson")[,c(5,6)],1,diff))
cbind("glmdr"=glmdr_PI,"bayesglm"=bayes_PI,"brglm2"=brglm_PI,"OLS"=linear_PI) / n_sample
```

```
##          glmdr  bayesglm    brglm2       OLS
## [1,] 0.8069115 0.8278469 0.8127081 0.8605427
```

3.4. Endometrial Example

Data Wrangling

```
data(endometrial)
```

Inference

```
glmdr_mod <- glmdr(HG~.,data=endometrial,family="binomial")
glmdr_pred <- c(predict(glmdr_mod$om,type="response",se.fit=TRUE)$fit[!glmdr_mod$linearity],
                predict(glmdr_mod$lcm,type="response"))
glmdr_pred <- glmdr_pred[order(as.numeric(names(glmdr_pred)))]
glmdr_CI <- inference(glmdr_mod)
glmdr_length <- sum(as.matrix(glmdr_CI) %*% c(-1,1))

bayes_mod <- bayesglm(HG~.,data=endometrial,family="binomial")
bayes_pred <- predict(bayes_mod,type="response",se.fit=TRUE)
bayes_CI <- binom.confint(bayes_pred$fit[!glmdr_mod$linearity],n=1,methods="wilson")[,c(5,6)]
bayes_length <- sum(as.matrix(bayes_CI) %*% c(-1,1))

brglm_mod <- glm(HG~.,data=endometrial,family = "binomial", method = "brglmFit", type = "MPL_Jeffreys")
brglm_pred <- predict(brglm_mod,type="response",se.fit=TRUE)
brglm_CI <- binom.confint(predict(brglm_mod,type="response")[!glmdr_mod$linearity],n=1,methods="wilson")[,c(5,6)]
brglm_length <- sum(as.matrix(brglm_CI) %*% c(-1,1))

lm_mod <- lm(HG~.,data=endometrial)
lm_pred <- (predict(lm_mod,se.fit = TRUE))
lm_pred$fit <- ifelse(lm_pred$fit<=0, 0,lm_pred$fit)
lm_pred$fit <- ifelse(lm_pred$fit>=1, 1,lm_pred$fit)
lm_CI <- binom.confint(lm_pred$fit[!glmdr_mod$linearity],n=1,methods="wilson")[,c(5,6)]
lm_length <- sum(as.matrix(lm_CI) %*% c(-1,1))
```

```
#In-sample Accuracy
cbind("glmdr"=mean(as.numeric(glmdr_pred >= 0.5) == endometrial$HG),
"bayesglm"=mean(as.numeric(bayes_pred$fit >= 0.5) == endometrial$HG),
"brglm2"=mean(as.numeric(brglm_pred$fit >= 0.5) == endometrial$HG),
"OLS"=mean(as.numeric(lm_pred$fit >= 0.5) == endometrial$HG))
```

```
##          glmdr  bayesglm    brglm2       OLS
## [1,] 0.8860759 0.8860759 0.8860759 0.8607595
```

```
#Average length of Confidence Interval
cbind(glmdr_length,bayes_length,brglm_length,lm_length) / sum(!glmdr_mod$linearity)
```

```
##      glmdr_length bayes_length brglm_length lm_length
## [1,]    0.1939755     0.803561    0.8081352 0.8064995
```

Prediction

```
n_sample <- nrow(endometrial)
y_idx <- which(names(glmdr_mod$om$model) == paste(formula(glmdr_mod$om))[2])
dat <- glmdr_mod$om$model
#glmdr
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  glmdr_mod <- glmdr(HG~.,data=training, family="binomial")
  ret2 <- predict(glmdr_mod,newdata = testing_X, alpha=0.05,crit="AICc")
  ret2
}
total_mat_glmdr_test <- do.call(c,ret)
```

```
#bayesglm
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  bayesglm_mod <- bayesglm(HG~.,data=training, family="binomial")
  ret2 <- predict(bayesglm_mod, newdata= testing_X, type="response")
  ret2
}
total_mat_bayes_test <- do.call(c,ret)
```

```
#brglm
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  brglm_mod <- glm(HG~.,data=training,family = binomial(logit), method = "brglmFit", type = "MPL_Jeffreys")
  ret2 <- predict(brglm_mod, newdata= testing_X, type="response")
  ret2
}
total_mat_brglm_test <- do.call(c,ret)
```

```
#OLS
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  lm_mod <- lm(HG~.,data=training)
  ret2 <- predict(lm_mod, newdata= testing_X, type="response")
  ret2
}
total_mat_linear_test <- do.call(c,ret)
total_mat_linear_test <- ifelse(total_mat_linear_test <0,0,total_mat_linear_test)
total_mat_linear_test <- ifelse(total_mat_linear_test >1,1,total_mat_linear_test)
```

```
thresholds = optimal.thresholds(data.frame("ID"=1:nrow(endometrial),"observed"=endometrial$HG,"glmdr"=total_mat_g
lmdr_test,"bayesglm"=total_mat_bayes_test,"brglm2"=total_mat_brglm_test,"OLS"=total_mat_linear_test),opt.methods=
"MaxPCC")
thresholds
```

Method	glmdr	bayesglm	brglm2	OLS
<fct>	<dbl>	<dbl>	<dbl>	<dbl>
MaxPCC	0.4833333	0.47	0.47	0.44

1 row

```
#Out-of-sample accuracy
cbind("glmdr"=mean(as.numeric(total_mat_glmdr_test >= thresholds[2]$glmdr) == endometrial$HG),
"bayesglm"=mean(as.numeric(total_mat_bayes_test >= thresholds[3]$bayesglm) == endometrial$HG),
"brglm2"=mean(as.numeric(total_mat_brglm_test >= thresholds[4]$brglm2) == endometrial$HG),
"OLS"=mean(as.numeric(total_mat_linear_test >= thresholds[5]$OLS) == endometrial$HG))
```

```
##          glmdr  bayesglm    brglm2       OLS
## [1,] 0.8734177 0.8607595 0.8607595 0.8607595
```

```
#Prediction Interval
glmdr_PI <- sum(apply(binom.confint(total_mat_glmdr_test,1,methods="wilson")[,c(5,6)],1,diff))
bayes_PI <- sum(apply(binom.confint(total_mat_bayes_test,1,methods="wilson")[,c(5,6)],1,diff))
brglm_PI <- sum(apply(binom.confint(total_mat_brglm_test,1,methods="wilson")[,c(5,6)],1,diff))
linear_PI <- sum(apply(binom.confint(total_mat_linear_test,1,methods="wilson")[,c(5,6)],1,diff))
cbind("glmdr"=glmdr_PI,"bayesglm"=bayes_PI,"brglm2"=brglm_PI,"OLS"=linear_PI) / n_sample
```

```
##          glmdr  bayesglm    brglm2       OLS
## [1,] 0.8475621 0.8426893 0.8436958 0.8506793
```

3.5. Maize Example

Data Wrangling

```
corn_dat <- readxl::read_xlsx("./Combined_Final_Product.xlsx")
names(corn_dat)[c(10,8)] <- c("Kernel.color","Pop.structure")
Xind <- 11:ncol(corn_dat)
foo <- corn_dat[, c(10,8,Xind[-c(15,19,22,23,24)])]
foo$Pop.structure <- factor(foo$Pop.structure)
dat <- data.frame(foo)
dat[,-c(1,2)] <- scale(dat[,-c(1,2)]) #scale the data
```

Inference

```
glmdr_mod <- glmdr(Kernel.color~.,data=dat,family="binomial")
glmdr_pred <- c(predict(glmdr_mod$om,type="response",se.fit=TRUE)$fit[!glmdr_mod$linearity],
                predict(glmdr_mod$lcm,type="response"))
glmdr_pred <- glmdr_pred[order(as.numeric(names(glmdr_pred)))]
glmdr_CI <- inference(glmdr_mod)
glmdr_length <- sum(as.matrix(glmdr_CI) %*% c(-1,1))

bayes_mod <- bayesglm(Kernel.color~.,data=dat,family="binomial")
bayes_pred <- predict(bayes_mod,type="response",se.fit=TRUE)
bayes_CI <- binom.confint(bayes_pred$fit[!glmdr_mod$linearity],n=1,methods="wilson")[,c(5,6)]
bayes_length <- sum(as.matrix(bayes_CI) %*% c(-1,1))
```

```
#brglm2 does not conver. Alternatively, we will use the logistf.
brglm2_mod <- glm(Kernel.color~.,data=dat,family = "binomial", method = "brglmFit", type = "MPL_Jeffreys")
```

```
## Warning: brglmFit: algorithm did not converge
```

```
## Warning: brglmFit: fitted probabilities numerically 0 or 1 occurred
```

```
logistif_mod <- logistf(Kernel.color~.,data=dat, family = "binomial")
logistif_pred <- c(invlogit(model.matrix(glmdr_mod$om) %*% logistif_mod$coefficients))
logistf_CI <- binom.confint(logistif_pred[!glmdr_mod$linearity],n=1,methods="wilson")[,c(5,6)]
logistf_length <- sum(as.matrix(logistf_CI) %*% c(-1,1))

lm_mod <- lm(Kernel.color~.,data=dat)
lm_pred <- (predict(lm_mod,se.fit = TRUE))
lm_pred$fit <- ifelse(lm_pred$fit<=0, 0,lm_pred$fit)
lm_pred$fit <- ifelse(lm_pred$fit>=1, 1,lm_pred$fit)
lm_CI <- binom.confint(lm_pred$fit[!glmdr_mod$linearity],n=1,methods="wilson")[,c(5,6)]
lm_length <- sum(as.matrix(lm_CI) %*% c(-1,1))
```

```
#In-sample Accuracy
cbind("glmdr"=mean(as.numeric(glmdr_pred >= 0.5) == dat$Kernel.color),
"bayesglm"=mean(as.numeric(bayes_pred$fit >= 0.5) == dat$Kernel.color),
"brglm2(logistf)"=mean(as.numeric(logistif_pred >= 0.5) == dat$Kernel.color),
"OLS"=mean(as.numeric(lm_pred$fit >= 0.5) == dat$Kernel.color))
```

```
##          glmdr  bayesglm brglm2(logistf)       OLS
## [1,] 0.8713639 0.8707175       0.8700711 0.8681319
```

```
#Average length of Confidence Interval
cbind(glmdr_length,bayes_length,logistf_length,lm_length) / sum(!glmdr_mod$linearity)
```

```
##      glmdr_length bayes_length logistf_length lm_length
## [1,]    0.5631796    0.8137985      0.8260355 0.8375342
```

Prediction

```
n_sample <- nrow(dat)
y_idx <- which(names(glmdr_mod$om$model) == paste(formula(glmdr_mod$om))[2])
#glmdr
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  glmdr_mod <- glmdr(Kernel.color~.,data=training, family="binomial")
  ret2 <- predict(glmdr_mod,newdata = testing_X, alpha=0.05,crit="AICc")
  ret2
}
total_mat_glmdr_test <- do.call(c,ret)
```

```
#bayesglm
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  bayesglm_mod <- bayesglm(Kernel.color~.,data=training, family="binomial")
  ret2 <- predict(bayesglm_mod, newdata= testing_X, type="response")
  ret2
}
total_mat_bayes_test <- do.call(c,ret)
```

```
#logistf
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  logistf_mod <- logistf(Kernel.color~.,data=training, family="binomial")
  new_dat <- rbind(logistf_mod$data,as.data.frame(cbind("Kernel.color"=testing_Y,testing_X),col.names=colnames(lo
gistf_mod$data)))
  ret2 <- c(invlogit(tail(model.matrix(Kernel.color~.,data=new_dat),1) %*% logistf_mod$coefficients))
  ret2
}
total_mat_logistf_test <- do.call(c,ret)
```

```
#OLS
ret <- foreach(i=1:n_sample) %dopar% {
  training <- dat[-i,]
  testing_X <- dat[i,-y_idx,drop=FALSE]
  testing_Y <- dat[i,y_idx]
  lm_mod <- lm(Kernel.color~.,data=training, family="binomial")
  ret2 <- predict(lm_mod, newdata= testing_X, type="response")
  ret2
}
total_mat_linear_test <- do.call(c,ret)
total_mat_linear_test <- ifelse(total_mat_linear_test <0,0,total_mat_linear_test)
total_mat_linear_test <- ifelse(total_mat_linear_test >1,1,total_mat_linear_test)
```

```
thresholds = optimal.thresholds(data.frame("ID"=1:nrow(dat),"observed"=dat$Kernel.color,"glmdr"=total_mat_glmdr_t
est,"bayesglm"=total_mat_bayes_test,"logistf"=total_mat_logistf_test,"OLS"=total_mat_linear_test),opt.methods="Ma
xPCC")
thresholds
```

Method <fct>	glmdr <dbl>	bayesglm <dbl>	logistf <dbl>	OLS <dbl>
MaxPCC	0.5	0.62	0.85	0.605

1 row

```
#Out-of-sample accuracy
cbind("glmdr"=mean(as.numeric(total_mat_glmdr_test >= thresholds[2]$glmdr) == dat$Kernel.color),
"bayesglm"=mean(as.numeric(total_mat_bayes_test >= thresholds[3]$bayesglm) == dat$Kernel.color),
"logistf"=mean(as.numeric(total_mat_logistf_test >= thresholds[4]$logistf) == dat$Kernel.color),
"OLS"=mean(as.numeric(total_mat_linear_test >= thresholds[5]$OLS) == dat$Kernel.color))
```

```
##          glmdr bayesglm   logistf       OLS
## [1,] 0.8623142 0.863607 0.8629606 0.8655462
```

```
#Prediction Interval
glmdr_PI <- sum(apply(binom.confint(total_mat_glmdr_test,1,methods="wilson")[,c(5,6)],1,diff))
bayes_PI <- sum(apply(binom.confint(total_mat_bayes_test,1,methods="wilson")[,c(5,6)],1,diff))
brglm_PI <- sum(apply(binom.confint(total_mat_brglm_test,1,methods="wilson")[,c(5,6)],1,diff))
linear_PI <- sum(apply(binom.confint(total_mat_linear_test,1,methods="wilson")[,c(5,6)],1,diff))
cbind("glmdr"=glmdr_PI,"bayesglm"=bayes_PI,"brglm2"=brglm_PI,"OLS"=linear_PI) / n_sample
```

```
##          glmdr  bayesglm     brglm2       OLS
## [1,] 0.8371185 0.8370744 0.04308466 0.8391364
```