

CSSS/POLS 510 MLE Lab

4. Binary Model and Quantities of Interest (QoI)

Inhwan Ko

Oct 22, 2021

Agenda

- 0 HW2 Question
- 1 Obtaining QoI from Heteroskedastic Normal
- 2 Binary Model (NES)

0. HW2 Question

How to derive the likelihood function?

- 1 Express the joint probability of the data, using the chosen probability distribution
- 2 Convert 1. to the likelihood
- 3 Simplify 2. for easy maximization (*sufficient statistics*)
- 4 Replace parameter with the systematic component (e.g., $\lambda_i = \exp(\mathbf{x}_i\beta)$)

0. HW2 Question

1 Problem 1:

$$f_{\text{Bern}}(y | \pi) = \pi^y (1 - \pi)^{1-y}$$

$$\Pr(y | \pi) = \prod_{i=1}^n \pi^{x_i} (1 - \pi)^{1-x_i}$$

$$\begin{aligned} L(\pi | y) &= k(y) \Pr(y | \pi) \\ &= k(y) \prod_{i=1}^n \pi^{y_i} (1 - \pi)^{1-y_i} \end{aligned}$$

0. HW2 Question

2 Problem 2:

$$L(\lambda | y) = k(y)\Pr(y | \lambda) = k(y) \prod_{i=1}^n \frac{\exp(-\lambda_i)\lambda_i^{y_i}}{y!}$$

Question: why λ_i on the right hand side rather than just using λ ?

1. Heteroskedastic Normal Example

- Steps

- ▶ The full R code can be found [here from Chris' website](#)

- 2.1 Generate Data

- 2.2 Fit OLS - `lm()`

- 2.3 Fit MLE - `optim()`

- 2.4 Calculate quantities of interest

- ★ Use `predict()`

- ★ Use simulation

1.1 Recap

```
rm(list=ls())           # Clear memory
set.seed(123456)       # For reproducible random numbers
library(MASS)          # Load packages
library(simcf)
library(tidyverse)

n <- 1500               # Generate 1500 observations

w0 <- rep(1, n)        # Create the constant
w1 <- runif(n)         # Create two covariates
w2 <- runif(n)

x <- cbind(w0, w1, w2) # Create a matrix of the covariates
z <- x                 # i.e., same covariates affect mu and sigma

beta <- c(0, 5, 15)   # Set a parameter vector for the mean
                      # One for constant, one for covariate 1,
                      # one for covariate 2.

gamma <- c(1, 0, 3)   # Set a parameter vector for the variance
                      # Gamma estimate for covariate 2 is set to be 3,
                      # whic creates heteroskedasticity
```

1.1 Recap

```
mu <- x %*% beta           # Create systematic component for the mean

sigma2 <- exp(z %*% gamma) # Create systematic component for the variance
                          # Since ith row of sigma2 = exp(1 + 0 + w2_i * 3)
                          # i.e., it is a function of w2 (heteroskedastic)

y <- rnorm(n = n,          # Create the outcome variable
          mean = mu,      # Think about the stochastic component!
          sd = sqrt(sigma2)
          )

data <- cbind(y, w1, w2)   # Save the data to a data frame
data <- as.data.frame(data)
names(data) <- c("y", "w1", "w2")
```


1.1 Recap

```
lsResult <- lm(y ~ w1 + w2, data = data) # Fit a linear model
                                         # using simulated data

lsAIC <- AIC(lsResult) # Calculate and print the AIC
                       # i.e. Akaike Information Criterion
                       # to assess goodness of fit; lower AIC is better
```

1.1 Recap

```
# A likelihood function for ML heteroskedastic Normal
llkHetrnormlin <- function(param, y, x, z) {
  x <- as.matrix(x)           # x (some covariates) as a matrix
  z <- as.matrix(z)           # z (some covariates) as a matrix
  os <- rep(1, nrow(x))       # Set the intercept as 1 (constant)
  x <- cbind(os, x)           # Add intercept to covariates x
  z <- cbind(os, z)           # Add intercept to covariates z

  b <- param[1 : ncol(x)]      # Parameters for x
  g <- param[(ncol(x) + 1) : (ncol(x) + ncol(z))] # Parameters for z

  xb <- x %*% b                # Systematic components for mean
  s2 <- exp(z %*% g)           # Systematic components for variance

  sum(0.5 * (log(s2) + (y - xb)^2 / s2)) # Likelihood we want to maximize
                                       # optim is a minimizer by default
                                       # To maximize lnL is to minimize -lnL
                                       # so the +/- signs are reversed
}
```

1.1 Recap

```
# Create input matrices
xcovariates <- cbind(w1, w2)
zcovariates <- cbind(w1, w2)

# Initial guesses of beta0, beta1, ..., gamma0, gamma1, ...
# We need one entry per parameter, in order!
# Note: also include beta and gamma estimates for constants
stval <- c(0, 0, 0, 0, 0, 0)

# Run ML, and get the output we need
hetnormResult <- optim(stval,           # Initial guesses
                       llkHetnormlin,  # Likelihood function
                       method = "BFGS", # Gradient method
                       hessian = TRUE,  # Return Hessian matrix
                       y = y,           # Outcome variable
                       x = xcovariates, # Covariates x (w/o constant)
                       z = zcovariates  # Covariates z (w/o constant)
                       )
```

Note: By default, `optim()` performs a minimizing procedure. You can make `optim` a maximizer by adding `control = list(fnscale = -1)`

1.1 Recap

```
pe <- hetnormResult$par           # Point estimates
round(pe, 3)
```

```
## [1] -0.167  5.007 15.698  0.910  0.224  2.994
```

```
vc <- solve(hetnormResult$hessian) # Var-cov matrix (for computing s.e.)
round(vc, 5)
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,]  0.02909 -0.03265 -0.02810  0.00059 -0.00087 -0.00032
## [2,] -0.03265  0.06787 -0.00025 -0.00091  0.00099  0.00084
## [3,] -0.02810 -0.00025  0.10331 -0.00056  0.00143 -0.00033
## [4,]  0.00059 -0.00091 -0.00056  0.00920 -0.00832 -0.00749
## [5,] -0.00087  0.00099  0.00143 -0.00832  0.01693 -0.00042
## [6,] -0.00032  0.00084 -0.00033 -0.00749 -0.00042  0.01568
```

```
se <- sqrt(diag(vc))           # To compute standard errors (s.e.)
                                # take the diagonal of the Hessian;
                                # then take square root
round(se, 3)
```

```
## [1] 0.171 0.261 0.321 0.096 0.130 0.125
```

1.1 Recap

```
mleResult <- cbind(pe[1:3], se[1:3])           # Report ML results
colnames(mleResult) <- c("Estimate", "Std.Error")
rownames(mleResult) <- c("(Intercept)", "w1", "w2")
round(mleResult, 3)
```

```
##           Estimate Std. Error
## (Intercept)  -0.167    0.171
## w1           5.007    0.261
## w2          15.698    0.321
```

```
round(coef(summary(lsResult))[, c(1, 2)], 3) # Compare with lm() results
```

```
##           Estimate Std. Error
## (Intercept)  -0.049    0.282
## w1           4.784    0.377
## w2          15.683    0.371
```

```
ll <- -hetnormResult$value           # Report likelihood at maximum
                                     # No need to have negative sign
                                     # if optim is set as maximizer
ll
```

```
## [1] -2620.334
```

1.1 Recap

```
# AIC is 2 * number of parameters - 2 * ll (i.e. likelihood at maximum)
```

```
hetnormAIC <- 2 * length(stval) - 2 * ll
```

```
# Compare AIC from ML and from lm(); lower is better
```

```
print(hetnormAIC)
```

```
## [1] 5252.668
```

```
print(lsAIC)
```

```
## [1] 8545.903
```

1.2 Recap2: Predict QoI

Motivation: We want to study how the change in a particular explanatory variable affects the outcome variable, *all else being equal*

1.2 Recap2: Predict Qol

- Scenario 1: Vary covariate 1 (w_1); hold covariate 2 (w_2) constant
- ① Create a data frame with a set of hypothetical scenarios for covariate 1, while keeping covariate 2 at its mean
 - ▶ What is the sensible range of some hypothetical scenarios for covariate 1? Consider the original range of w_1 .
- ② Calculate the predicted values using the `predict()` function
 - ▶ Hint: you need at least the following arguments: `predict(object = ... , newdata = ... , interval = ... , level = ...)`
- ③ Plot the prediction intervals

1.2 Recap2: Predict Qol

- ④ Similarly, calculate the confidence intervals using the `predict()` function
- ⑤ Plot the confidence intervals; compare them with the predictive intervals

1.2 Recap2: Predict QoI

```
# Set up
w1range <- seq(from = 0, to = 1, by = 0.05)  # Set up hypothetical values for w1
w2mean <- mean(w2)                            # Set up mean for w2
xhypo <- crossing(w1 = w1range, w2 = w2mean)  # Create a new dataset using crossing
# Calculate predicted values
simPIw1 <- predict(lsResult,                 # A model object
                   newdata = xhypo,          # New dataset
                   interval = "prediction",  # What kind of intervals?
                   level = 0.95)            # What levels of confidence?
```

1.2 Recap2: Predict QoI

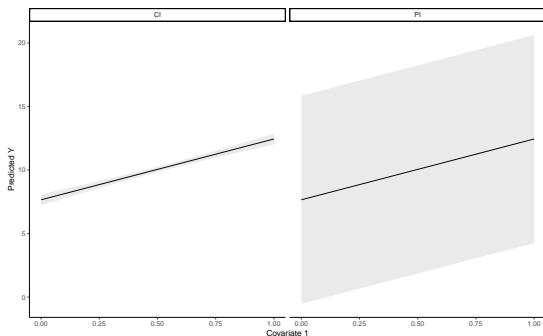
```
simPIw1 <- simPIw1 %>%  
  as_tibble() %>%           # Coerce it into a tibble  
  bind_cols(w1 = w1range) # Combine hypo w1 with predicted y  
  
# Calculate confidence intervals using predict()  
simCIw1 <- predict(lsResult,  
                  newdata = xhypo,  
                  interval = "confidence",  
                  level = 0.95)  
  
simCIw1 <- simCIw1 %>%  
  as_tibble() %>%  
  bind_cols(w1 = w1range)
```

1.2 Recap2: Predict QoI

```
# ggplot2
theme_set(theme_classic())
simALLw1 <- bind_rows(
  simPIw1 %>% mutate(type = "PI"),
  simCIw1 %>% mutate(type = "CI")
)
```

1.2 Recap2: Predict Qol

```
# Plot confidence intervals and predictive intervals side by side  
ggplot(simALLw1, aes(x = w1, y = fit, ymax = upr, ymin = lwr)) +  
  geom_line() +  
  geom_ribbon(alpha = 0.1) +  
  labs(y = "Predicted Y", x = "Covariate 1") +  
  facet_grid(~ type)
```



1.2 Recap2: Predict Qol

Scenario 2: Vary covariate 2; hold covariate 1 constant

```
w2range <- seq(from = 0, to = 1, by = 0.05) # Set up hypothetical values for w1
w1mean <- mean(w1) # Set up mean for w2
xhypo <- crossing(w1 = w1mean, # Create a new dataset
                 w2 = w2range) # crossing() is from tidyr package
# Calculate predicted values
simPIw2 <- predict(lsResult, # A model object
                  newdata = xhypo, # New dataset
                  interval = "prediction", # What kind of intervals?
                  level = 0.95) # What levels of confidence?

simPIw2 <- simPIw2 %>%
  as_tibble() %>%
  bind_cols(w2 = w2range)
```

1.3 Simulating QoI using `simcf`

Motivation: Can we use simulation methods to produce the same prediction and confidence intervals? Recall the lecture: we can draw a bunch of $\tilde{\beta}$ from a multivariate normal distribution

Demonstration:

- Scenario 1: Vary covariate 1; hold covariate 2 constant
- ① Create a data frame with a set of hypothetical scenarios for covariate 1 while keeping covariate 2 at its mean
- ② Simulate the predicted values using `MASS` and `simcf`
- ③ Plot the results

1.3 Simulating QoI using `simcf`

In order to use `simcf` to generate quantities of interest, the following steps are needed:

- 1 Estimate: MLE $\hat{\beta}$ and its variance $\hat{V}(\hat{\beta})$
- 2 Simulate estimation uncertainty from a multivariate normal distribution:
Draw $\tilde{\beta} \sim MVN[\hat{\beta}, \hat{V}(\hat{\beta})]$
- 3 Create hypothetical scenarios of your substantive interest:
Choose values of X: X_c

1.3 Simulating QoI using `simcf`

- 4 Calculate expected values:

$$\tilde{\mu}_c = g(X_c, \tilde{\beta})$$

- 5 Simulate fundamental uncertainty:

$$\tilde{y}_c \sim f(\tilde{\mu}_c, \tilde{\alpha})$$

or

- 6 Compute EVs, First Differences or Relative Risks

$$\text{EV: } \mathbb{E}(y|X_{c1})$$

$$\text{FD: } \mathbb{E}(y|X_{c2}) - \mathbb{E}(y|X_{c1})$$

$$\text{RR: } \frac{\mathbb{E}(y|X_{c2})}{\mathbb{E}(y|X_{c1})}$$

1.3 Simulating QoI using `simcf`

In order to use `simcf` to generate quantities of interest, the following steps are needed:

- 1 Estimate: MLE $\hat{\beta}$ and its variance $\hat{V}(\hat{\beta})$
→ `optim()`, `glm()`
- 2 Simulate estimation uncertainty from a multivariate normal distribution:
Draw $\tilde{\beta} \sim MVN[\hat{\beta}, \hat{V}(\hat{\beta})]$
→ `MASS::mvrnorm()`
- 3 Create hypothetical scenarios of your substantive interest:
Choose values of X: $X_c \rightarrow \text{simcf}::\text{cfmake}(), \text{cfchange}() \dots$

1.3 Simulating QoI using simcf

- 4 Calculate expected values:

$$\tilde{\mu}_c = g(X_c, \tilde{\beta})$$

- 5 Simulate fundamental uncertainty:

$$\tilde{y}_c \sim f(\tilde{\mu}_c, \tilde{\alpha})$$

→ `simcf::hetnormsimpv()` ...

or

- 6 Compute EVs, First Differences or Relative Risks

EV: $\mathbb{E}(y|X_{c1})$

→ `simcf::logitsimev()` ...

FD: $\mathbb{E}(y|X_{c2}) - \mathbb{E}(y|X_{c1})$

→ `simcf::logitsimfd()` ...

RR: $\frac{\mathbb{E}(y|X_{c2})}{\mathbb{E}(y|X_{c1})}$

→ `simcf::logitsimrr()` ...

1.3 Simulating QoI using simcf

Recall our likelihood function is:

$$P(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2) = \prod_{i=1}^n f_{\mathcal{N}}(y_i|\mu_i, \sigma_i^2) \quad \text{[Joint probability]}$$

$$P(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2) = \prod_{i=1}^n (2\pi\sigma_i^2)^{-1/2} \exp\left[-\frac{(y_i - \mu_i)^2}{2\sigma_i^2}\right] \quad \text{[Expressed in Normal distribution]}$$

$$\log \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\sigma}^2|\mathbf{y}) \propto -\frac{1}{2} \sum_{i=1}^n \log \sigma_i^2 - \frac{1}{2} \sum_{i=1}^n \frac{(y_i - \mathbf{x}_i\boldsymbol{\beta})^2}{\sigma_i^2} \quad \text{[Converted to log likelihood; simplify]}$$

$$\log \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\gamma}|\mathbf{y}) \propto -\frac{1}{2} \sum_{i=1}^n \log z_i\gamma - \frac{1}{2} \sum_{i=1}^n \frac{(y_i - \mathbf{x}_i\boldsymbol{\beta})^2}{\exp(z_i\gamma)} \quad \text{[Substitute in systematic components]}$$

1.3 Simulating QoI using simcf

```
# Draw parameters from the model predictive distribution
sims <- 10000
simparam <- mvrnorm(n = sims,
                   mu = pe,      # M.N.D. with population mean = pe (from ML);
                   Sigma = vc) # population var-covar matrix = vc (from ML)
head(simparam)                # Inspect
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,]  0.24211743  4.253787  15.94193  0.8538881  0.1676888  3.187143
## [2,] -0.10028149  4.866367  15.88825  0.8131899  0.2369895  3.038976
## [3,] -0.15729804  4.813402  15.81664  0.9172401  0.3016272  2.864065
## [4,] -0.01521245  4.482252  16.12675  0.9590520  0.2337601  2.975280
## [5,]  0.10941227  4.821406  15.16535  0.9812625  0.2492318  2.763246
## [6,] -0.22838155  4.947090  15.62518  0.9596191  0.1447425  2.870572
```

1.3 Simulating QoI using simcf :S1

```
# Separate into the simulated betas and simulated gammas
simbetas <- simparam[ , 1:(ncol(xcovariates) + 1)]
simgammas <- simparam[ , (ncol(simbetas) + 1):ncol(simparam)]

# Specify model formulas
varmodel <- (y ~ w1 + w2)

# Create hypothetical scenarios for covariate 1
wirange <- seq(from = 0, to = 1, by = 0.2)
```

1.3 Simulating QoI using simcf :S1

```
# Use `cfMake` to create a baseline dataset
```

```
xhypo <- cfMake(varmodel,  
               data,  
               nscen = length(wirange))
```

```
xhypo
```

```
## $x  
##      y      w1      w2  
## 1 10.06525 0.5036167 0.4912698  
## 2 10.06525 0.5036167 0.4912698  
## 3 10.06525 0.5036167 0.4912698  
## 4 10.06525 0.5036167 0.4912698  
## 5 10.06525 0.5036167 0.4912698  
## 6 10.06525 0.5036167 0.4912698  
##  
## $xpre  
##      y      w1      w2  
## 1 10.06525 0.5036167 0.4912698  
## 2 10.06525 0.5036167 0.4912698  
## 3 10.06525 0.5036167 0.4912698  
## 4 10.06525 0.5036167 0.4912698  
## 5 10.06525 0.5036167 0.4912698  
## 6 10.06525 0.5036167 0.4912698  
##  
## $model  
## y ~ w1 + w2  
##  
## attr(,"class")  
## [1] "list"           "counterfactual"
```

1.3 Simulating QoI using simcf :S1

```
# Use `cfChange` and loop function to loop through each hypothetical x values
for (i in 1:length(w1range)) {
  xhypo <- cfChange(xscen = xhypo,
                   covname = "w1",
                   x = w1range[i],
                   scen = i)
}

xhypo
```

```
## $x
##      y w1      w2
## 1 10.06525 0.0 0.4912698
## 2 10.06525 0.2 0.4912698
## 3 10.06525 0.4 0.4912698
## 4 10.06525 0.6 0.4912698
## 5 10.06525 0.8 0.4912698
## 6 10.06525 1.0 0.4912698
##
## $xpre
##      y      w1      w2
## 1 10.06525 0.5036167 0.4912698
## 2 10.06525 0.5036167 0.4912698
## 3 10.06525 0.5036167 0.4912698
## 4 10.06525 0.5036167 0.4912698
## 5 10.06525 0.5036167 0.4912698
## 6 10.06525 0.5036167 0.4912698
##
## $model
## y ~ w1 + w2
##
## attr(,"class")
## [1] "list"          "counterfactual"
```


1.3 Simulating QoI using simcf :S1

```
# Repeat the same procedures for z
zhypo <- cfMake(varmodel, data, nscen = length(w1range))

for (i in 1:length(w1range)) {
  zhypo <- cfChange(zhypo, "w1", x = w1range[i], scen = i)
}

zhypo
```

```
## $x
##      y w1      w2
## 1 10.06525 0.0 0.4912698
## 2 10.06525 0.2 0.4912698
## 3 10.06525 0.4 0.4912698
## 4 10.06525 0.6 0.4912698
## 5 10.06525 0.8 0.4912698
## 6 10.06525 1.0 0.4912698
##
## $xpre
##      y      w1      w2
## 1 10.06525 0.5036167 0.4912698
## 2 10.06525 0.5036167 0.4912698
## 3 10.06525 0.5036167 0.4912698
## 4 10.06525 0.5036167 0.4912698
## 5 10.06525 0.5036167 0.4912698
## 6 10.06525 0.5036167 0.4912698
##
## $model
## y ~ w1 + w2
##
## attr(,"class")
## [1] "list"          "counterfactual"
```

1.3 Simulating QoI using simcf :S1

```
# Simulate predictive intervals for heteroskedastic linear models  
# `hetnormsimpv()` is from simcf package  
simRESw1 <- hetnormsimpv(xhypo, simbetas,  
                        zhypo, singammas,  
                        ci = 0.95,  
                        constant = 1,  
                        varconstant = 1)
```

```
simRESw1
```

```
## $pe  
## [1] 7.515323 8.566149 9.575781 10.579808 11.552741 12.524484  
##  
## $lower  
##      [,1]  
## low 0.8847932  
## low 1.9381952  
## low 2.8502531  
## low 3.5495829  
## low 4.5126707  
## low 5.3290850  
##  
## $upper  
##      [,1]  
## up 13.87476  
## up 15.15291  
## up 16.41457  
## up 17.46719  
## up 18.47127  
## up 19.71323
```

1.3 Simulating QoI using simcf :S1

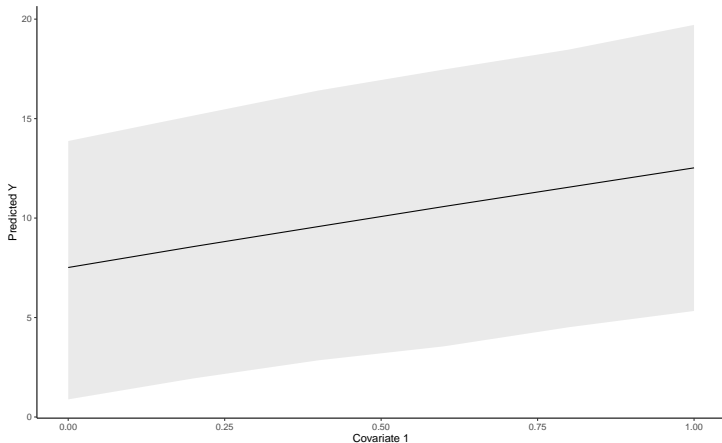
```
simRESw1 <- simRESw1 %>%  
  bind_rows() %>%           # Collaspe the list into d.f.  
  bind_cols(w1 = w1range) # Combine with hypo. w1 values  
  
simRESw1                    # Inspect
```

```
## # A tibble: 6 x 4  
##   pe lower[,1] upper[,1] w1  
##   <dbl>     <dbl>     <dbl> <dbl>  
## 1  7.52      0.885      13.9  0  
## 2  8.57      1.94       15.2  0.2  
## 3  9.58      2.85       16.4  0.4  
## 4 10.6      3.55       17.5  0.6  
## 5 11.6      4.51       18.5  0.8  
## 6 12.5      5.33       19.7  1
```

1.3 Simulating QoI using simcf :S1

```
# ggplot2
```

```
ggplot(simRESw1, aes(x = w1, y = pe, ymax = upper, ymin = lower)) +  
  geom_line() +  
  geom_ribbon(alpha = 0.1) +  
  labs(y = "Predicted Y", x = "Covariate 1")
```



1.3 Simulating QoI using `simcf` :S2

Scenario 2: Vary covariate 2; hold covariate 1 constant

- 1 Create a data frame with a set of hypothetical scenarios for covariate 2 while keeping covariate 1 at its mean
- 2 Simulate the predicted values using MASS and `simcf`
- 3 Plot the results

1.3 Simulating QoI using simcf :S2

```
# Create hypothetical scenarios for w2
w2range <- seq(0, 1, by = 0.05)

xhypo <- cfMake(varmodel, data, nscen = length(w2range))

for (i in 1:length(w2range)) {
  xhypo <- cfChange(xhypo, "w2", x = w2range[i], scen = i)
}

zhypo <- cfMake(varmodel, data, nscen = length(w2range))

for (i in 1:length(w2range)) {
  zhypo <- cfChange(zhypo, "w2", x = w2range[i], scen = i)
}
```

1.3 Simulating QoI using simcf :S2

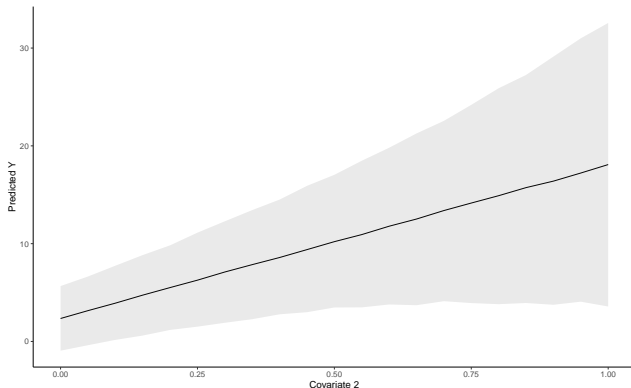
```
# Simulate the predicted Y's and PI's
simRESw2 <- hetnormsimpv(xhypo, simbetas,
                        zhypo, simgammas,
                        ci = 0.95,
                        constant = 1,
                        varconstant = 1)

simRESw2 <- simRESw2 %>%
  bind_rows() %>%
  bind_cols(w2 = w2range)
```

1.3 Simulating QoI using simcf :S2

Plot the predictive intervals for hypothetical w2

```
ggplot(simRESw2, aes(x = w2, y = pe, ymax = upper, ymin = lower)) +  
  geom_line() +  
  geom_ribbon(alpha = 0.1) +  
  labs(y = "Predicted Y", x = "Covariate 2")
```



Can we compare them with the prediction intervals generated by `predict()`?

1.3 Simulating QoI using simcf :S2

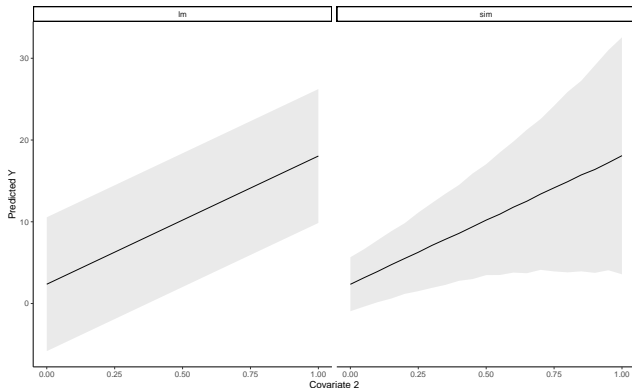
```
# Combine two dataframes
simPIw2 <- simPIw2 %>%
  rename(pe = fit, lower = lwr, upper = upr)

simALLw2 <- bind_rows(
  simRESw2 %>% mutate(method = "sim"),
  simPIw2 %>% mutate(method = "lm")
)
```

1.3 Simulating QoI using simcf :S2

Plot the predictive intervals for hypothetical w2

```
ggplot(simALLw2, aes(x = w2, y = pe, ymax = upper, ymin = lower)) +  
  geom_line() +  
  geom_ribbon(alpha = 0.1) +  
  labs(y = "Predicted Y", x = "Covariate 2") +  
  facet_grid(~ method)
```



2 Logit model: Voting example

Let's open RStudio and [nesLogitPlot.r]