CSSS/POLS 510 Maximum Likelihood Estimation: Lab 3

Heteroskedastic Normal

Kenya Amano

2020-10-3

Ad: Severyns Ravenholt Seminar in Comparative Politics

October 30, 2020 1:30pm -3pm

Speaker: Brian Leung, Former TA of this class

Bricks, Molotov Cocktails and Engaged Bystanders: Why Violence Becomes Sustained in Popular Protests

Agenda

1. Housekeeping: `simcf`, `tile`, LaTeX, R Markdown
2. Heteroskedastic Normal Example
3. Homework: Recap HW1, Question HW2

1. Housekeeping

1. Please make sure that you have Chris's `simcf` and `tile` packages installed
   - You can find these packages on the course website and follow the instructions to install (do not unzip the `.tgz` files)
   - Also `MASS` package
2. Any question related to LaTeX or `R Markdown`?
   - Make sure you have TeX installed, which you can use `tinytex` package
   - One common problem when knitting: the math mode environment doesn't like white space or empty line
     - Try `\begin{alinged}` instead of `\begin{split}`
     - `R Markdown` guide is here

1.Housekeeping

3. Homework Submission

- Try to produce a computational narrative PDF in `R markdown`.
- Email subject: **MLE510HW2**
- File name: **MLE510HW2KenyaAmano**

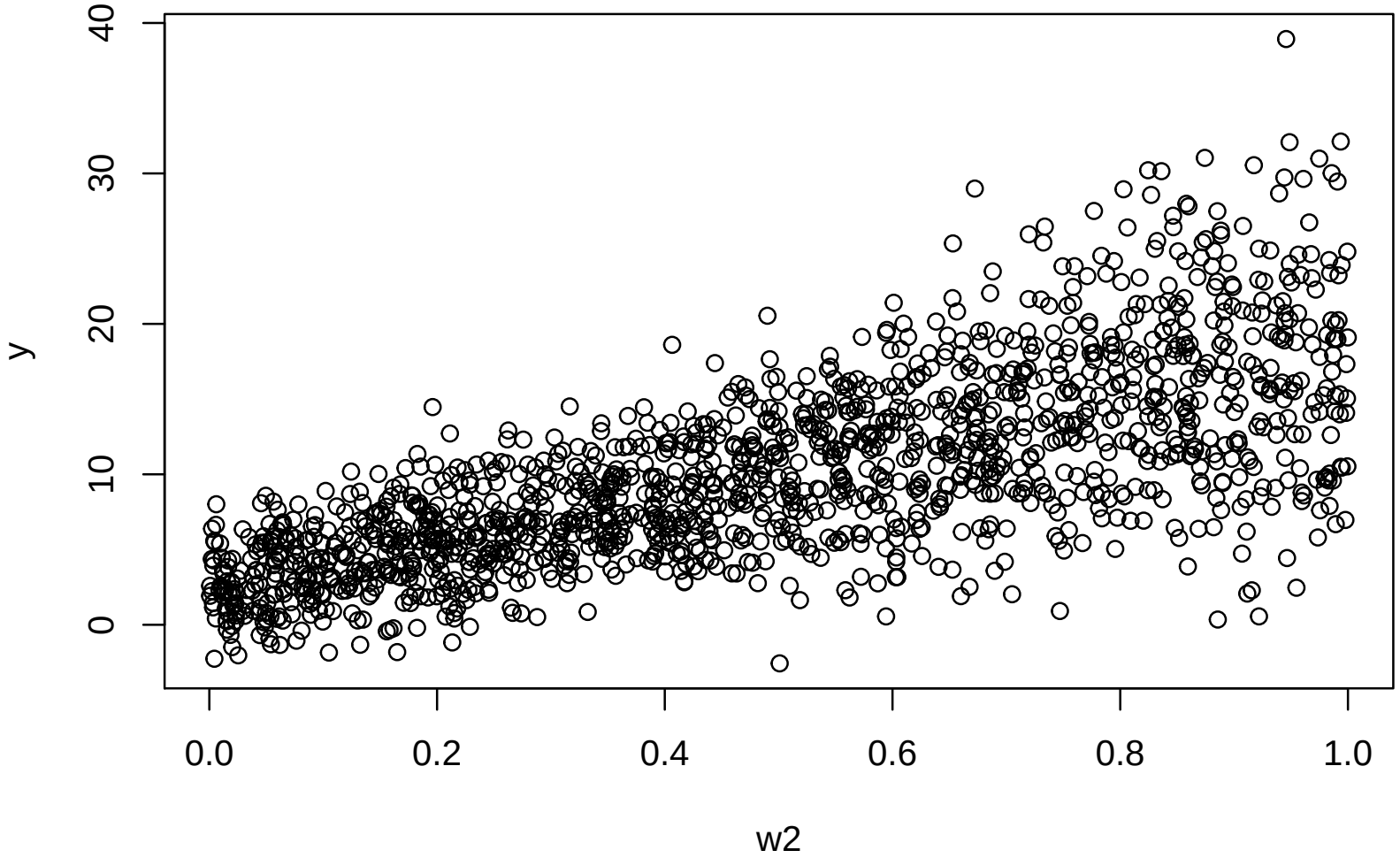
2. Heteroskedastic Normal Example

- 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
- Why do this exercise?

2. Heteroskedastic Normal Example

- Why do this exercise?
  - Learn the concept of MLE
  - Learn how to compute MLE
  - Learn differences between OLS and MLE
  - Learn Data Generating Process (DGP)

2.1 Generating heteroskedastic normal data



2.1 Generating heteroskedastic normal data

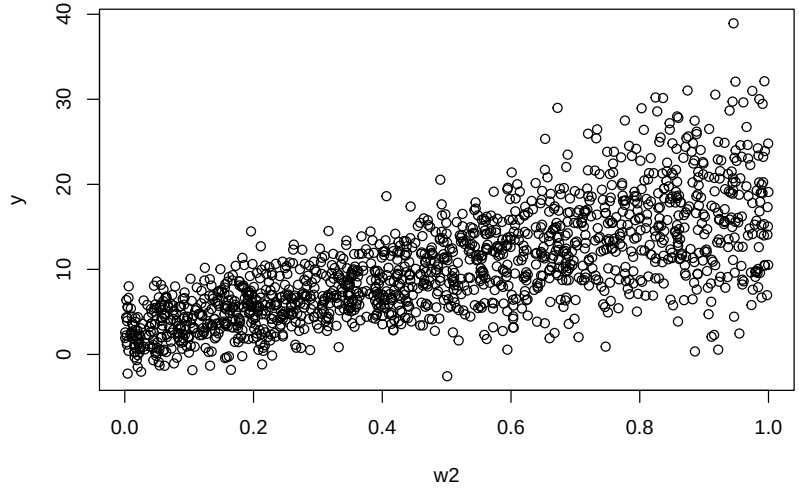
Stochastic component:

$$y_i \sim f_{\mathcal{N}}(\mu_i, \sigma_i^2)$$

Systematic components:

$$\mu_i = \boldsymbol{x}_i\beta$$

$$\sigma_i^2 = \exp(\boldsymbol{z}_i\gamma)$$



2.1 Generating heteroskedastic normal data

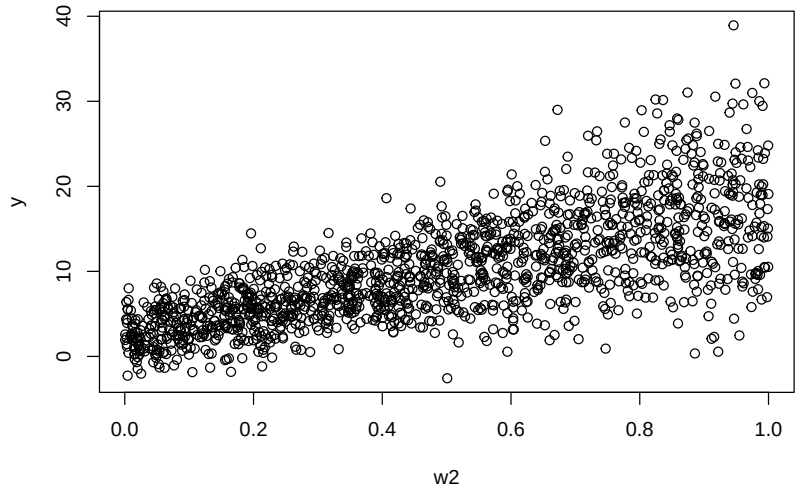
Stochastic component:

$$y_i \sim f_{\mathcal{N}}(\mu_i, \sigma_i^2)$$

Systematic components:

$$\mu_i = \boldsymbol{x}_i\beta$$

$$\sigma_i^2 = \exp(\boldsymbol{z}_i\gamma)$$



2.1 Generating heteroskedastic normal data

- Think back in linear regression where you model the outcome$_i$ on:$\beta_0 + x_{1,i}\beta_1 + x_{2,i}\beta_2 ... x_{k,i}\beta_k$, for person i and a total number of k covariates
- An efficient way is to store in *matrices* all covariates as $\mathbf{X}$ (think about your excel spreadsheet or .csv), and all coefficients as $\boldsymbol{\beta}$.

2.1 Generating heteroskedastic normal data

- Let's say we have one constant and two covariates, i.e. three β:

$$\mathbf{X} = \begin{bmatrix} x_0 & x_{1,1} & x_{2,1} \\ x_0 & x_{1,2} & x_{2,2} \\ x_0 & x_{1,3} & x_{2,3} \\ \vdots & \vdots & \vdots \\ x_0 & x_{1,i} & x_{2,i} \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

2.1 Generating heteroskedastic normal data

- Matrix multiplication generates another matrix of systematic components easily:

$$\mathbf{X}\beta = \begin{bmatrix} x_0 \cdot \beta_0 + x_{1,1} \cdot \beta_1 + x_{2,1} \cdot \beta_2 \\ x_0 \cdot \beta_0 + x_{1,2} \cdot \beta_1 + x_{2,2} \cdot \beta_2 \\ x_0 \cdot \beta_0 + x_{1,3} \cdot \beta_1 + x_{2,3} \cdot \beta_2 \\ \vdots \\ x_0 \cdot \beta_0 + x_{1,i} \cdot \beta_1 + x_{2,i} \cdot \beta_2 \end{bmatrix}$$

- Fast rule:a $\mathbf{4} \times 3$ matrix multiplied by a $3 \times \mathbf{1}$ one $\rightarrow$ a $\mathbf{4} \times \mathbf{1}$ matrix

2.1 Generating heteroskedastic normal data

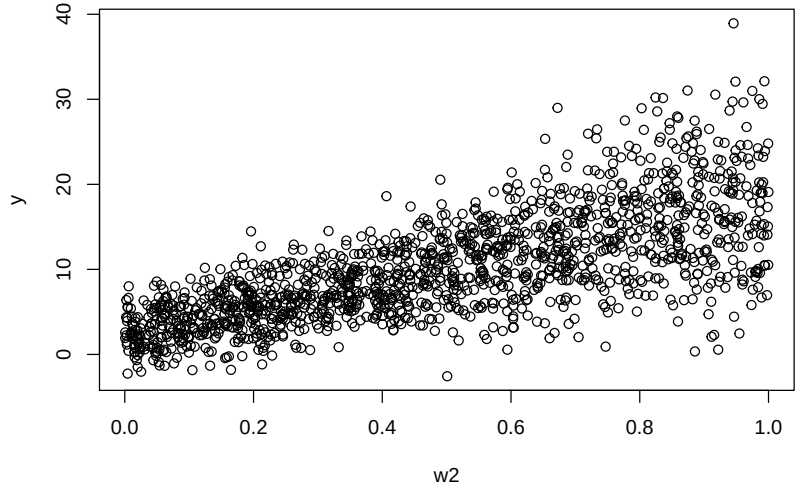
Stochastic component:

$$y_i \sim f_{\mathcal{N}}(\mu_i, \sigma_i^2)$$

Systematic components:

$$\mu_i = \boldsymbol{x}_i\beta$$

$$\sigma_i^2 = \exp(\boldsymbol{z}_i\gamma)$$



2.1 Generating heteroskedastic normal data

Steps

1. Set the number of observations to 1500 (n)
2. Generate two equivalent datasets (matrices **x** and **z**), both with one constant and two covariates
   - The constant takes the value of `1`; two covariates are drawn from a uniform distribution with `min = 0`, `max = 1`
3. Set a parameter vector (β) for the mean, μ_i
   - By "divine revelation", we know the true values are `0, 5, 15`

2.1 Generating heteroskedastic normal data

4. Set a parameter vector (γ) for the variance, σ_i, with heteroskedasticity
   - Again, we know the true values are `1, 0, 3`
   - Pause and think: why is there heteroskedasticity?
5. Create the systematic component for the mean ($\boldsymbol{x}_i\beta$)
6. Create the systematic component for the variance ($\exp(\boldsymbol{z}_i\gamma)$),
   - Assume the same covariates affect both μ_i and σ_i, that is, **x** and **z** are the same
7. Generate the outcome variable (y_i)

2.1 Generating heteroskedastic normal data

8. Save the data to a data frame

9. Plot the data

2.1 Generating heteroskedastic normal data

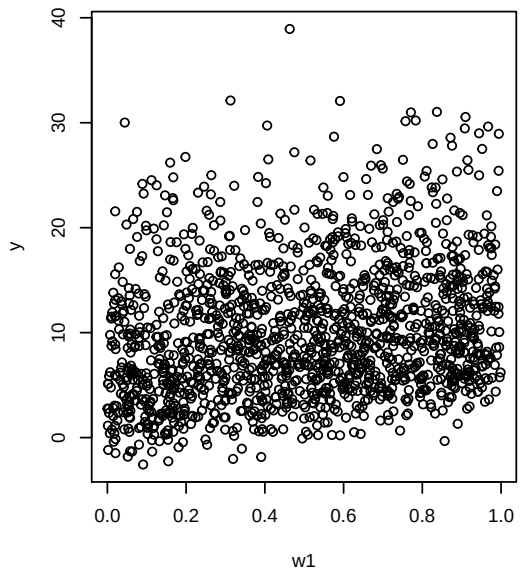
```r
rm(list=ls())          # Clear memory
set.seed(123456)       # For reproducible random numbers
library(MASS)          # Load packages
library(simcf)
library(tidyverse)

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
```

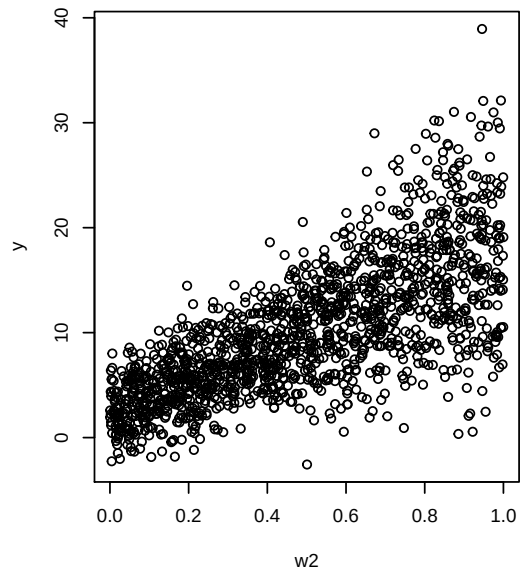
2.1 Generating heteroskedastic normal data

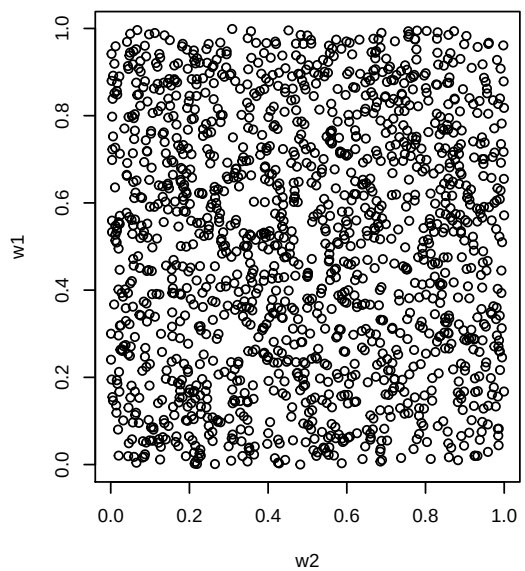
```r
mu <- x %*% beta                # Create systemtic component for the mean

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")
```

2.1 Generating heteroskedastic normal data

```
par(mfrow = c(1, 3))          #Plot the data
plot(y = y, x = w1)
plot(y = y, x = w2)
plot(y = w1, x = w2)
```







2.2. Fitting a model using OLS - `lm()`

Assume, one day, the true values of the parameters become unknown to mankind, and we as scientists try to recover the "truth" by gathering and studying the data

Motivation: If our statistical model is valid, it should be able to recover the true parameter values from the data we synthesize

1. Fit a model using least squares (function `lm()`) and regress the outcome variable on the two covariates
2. Calculate and print the AIC (More in lecture; I'll show you this time)

2.2. Fitting a model using OLS - lm()

```r
ls.result <- lm(y ~ w1 + w2, data = data) # Fit a linear model
                                          # using simulated data
summary(ls.result)
```

```
##
## Call:
## lm(formula = y ~ w1 + w2, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.5710  -2.3157  -0.0219   2.2124  21.9345
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.04867    0.28221  -0.172    0.863
## w1           4.78421    0.37699  12.690   <2e-16 ***
## w2          15.68283    0.37112  42.258   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.17 on 1497 degrees of freedom
## Multiple R-squared:  0.5678, Adjusted R-squared:  0.5672
```

2.2. Fitting a model using OLS - `lm()`

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

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

Taking Stock

- What do the above exercises tell us about statistical model and inference?
  - What do we mean when we say "fitting a model to the data"?

Taking Stock

- Data generating process (DGP): the "true, underlying" process that generates the data we observe
- Model: an attempt to capture important aspects of, and approximate crudely, that DGP
  - We just stipulate the parameters and synthesize the data
  - It'll be very worrying if our model can't recover the truth...
- Specifying a "correct" model to real world phenomena is incredibly hard, but also humbling, and hopefully exciting

2.3 Fitting the model using ML - `optim()`

- Opening the black box: build our own regression machine using maximum likelihood

2.3 Fitting the model using ML - `optim()`

- Lecture recap: How to derive maximum likelihood estimators in four easy steps:
  1. Express the joint probability of the data, using the chosen probability distribution
  2. Convert the joint probability to the likelihood (trivial, as they are proportional)
  3. Simplify the likelihood for easy maximization (take logs and reduce to "sufficient statistics")
  4. Substitute in the systematic component
- Then we find the set of parameters that maximize the likelihood
  - Using gradient descent; `optim()`

2.3 Fitting the model using ML - `optim()`

- Recap: our heteroskedastic normal model
- Stochastic component:

$$y_i \sim f_{\mathcal{N}}(\mu_i, \sigma_i^2)$$

- Systematic components:

$$\mu_i = \boldsymbol{x}_i \boldsymbol{\beta}$$

$$\sigma_i^2 = \exp(\boldsymbol{z}_i \boldsymbol{\gamma})$$

2.3 Fitting the model using ML - `optim()`

- MLE for a heteroskedastic normal data

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

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

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

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

2.3 Fitting the model using ML - optim()

```r
# A likelihood function for ML heteroskedastic Normal
llk.hetnormlin <- 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
}
```

2.3 Fitting the model using ML - `optim()`

```
sum(0.5 * (log(s2) + (y - xb)^2 / s2))
```

Think back:

$$\log \mathcal{L}(\beta, \gamma|\mathbf{y}) \propto -\frac{1}{2}\sum_{i=1}^{n} \log \mathbf{z}_i\gamma - \frac{1}{2}\sum_{i=1}^{n} \frac{(y_i - \mathbf{x}_i\beta)^2}{\exp(\mathbf{z}_i\gamma)}$$

$$\propto -1 \times \sum_{i=1}^{n} 0.5 \times (\log \mathbf{z}_i\gamma - \frac{(y_i - \mathbf{x}_i\beta)^2}{\exp(\mathbf{z}_i\gamma)})$$

Important: `optim()` is a minimizer by default, so you need to reverse the +/- signs to find the maximum point

2.3 Fitting the model using ML - optim()

```r
# 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 estiamtes for constants
stval <- c(0, 0, 0, 0, 0, 0)

# Run ML, and get the output we need
hetnorm.result <- optim(stval,              # Initial guesses
                        llk.hetnormlin,     # 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)`

2.3 Fitting the model using ML - optim()

```
pe <- hetnorm.result$par              # Point estimates
round(pe, 3)
```

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

```
vc <- solve(hetnorm.result$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
```

2.3 Fitting the model using ML - optim()

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

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

```
round(coef(summary(ls.result))[, 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 <- -hetnorm.result$value                    # Report likelihood at maximum
                                               # No need to have negative sign
                                               # if optim is set as maximizer
ll
```

2.3 Fitting the model using ML - optim()

```
# AIC is 2 * number of parameters - 2 * ll (i.e. likelihood at maximum)
hetnorm.aic <- 2 * length(stval) - 2 * ll

# Compare AIC from ML and from lm(); lower is better
print(hetnorm.aic)
```

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

```
print(ls.aic)
```

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

2.4 Calculate quantities of interest

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

2.4 Calculate quantities of interest - S1

- 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
   - What is the sensible range of some hypothetical scenarios for covariate 1? Consider the original range of `w1`.
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

2.4 Calculate quantities of interest - S1

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

2.4 Calculate quantities of interest - S1

```
# 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()
head(xhypo)                                   # Inspect
```

```
## # A tibble: 6 x 2
##      w1    w2
##   <dbl> <dbl>
## 1  0    0.491
## 2  0.05 0.491
## 3  0.1  0.491
## 4  0.15 0.491
## 5  0.2  0.491
## 6  0.25 0.491
```

2.4 Calculate quantities of interest - S1

```
# Calculate predicted values
simPI.w1 <- predict(ls.result,                  # A model object
                    newdata = xhypo,            # New dataset
                    interval = "prediction",    # What kind of intervals?
                    level = 0.95)               # What levels of confidence?
head(simPI.w1)                                  # Inspect
```

```
##        fit         lwr      upr
## 1 7.655838 -0.53595517 15.84763
## 2 7.895048 -0.29514691 16.08524
## 3 8.134259 -0.05450529 16.32302
## 4 8.373470  0.18596961 16.56097
## 5 8.612680  0.42627769 16.79908
## 6 8.851891  0.66641891 17.03736
```

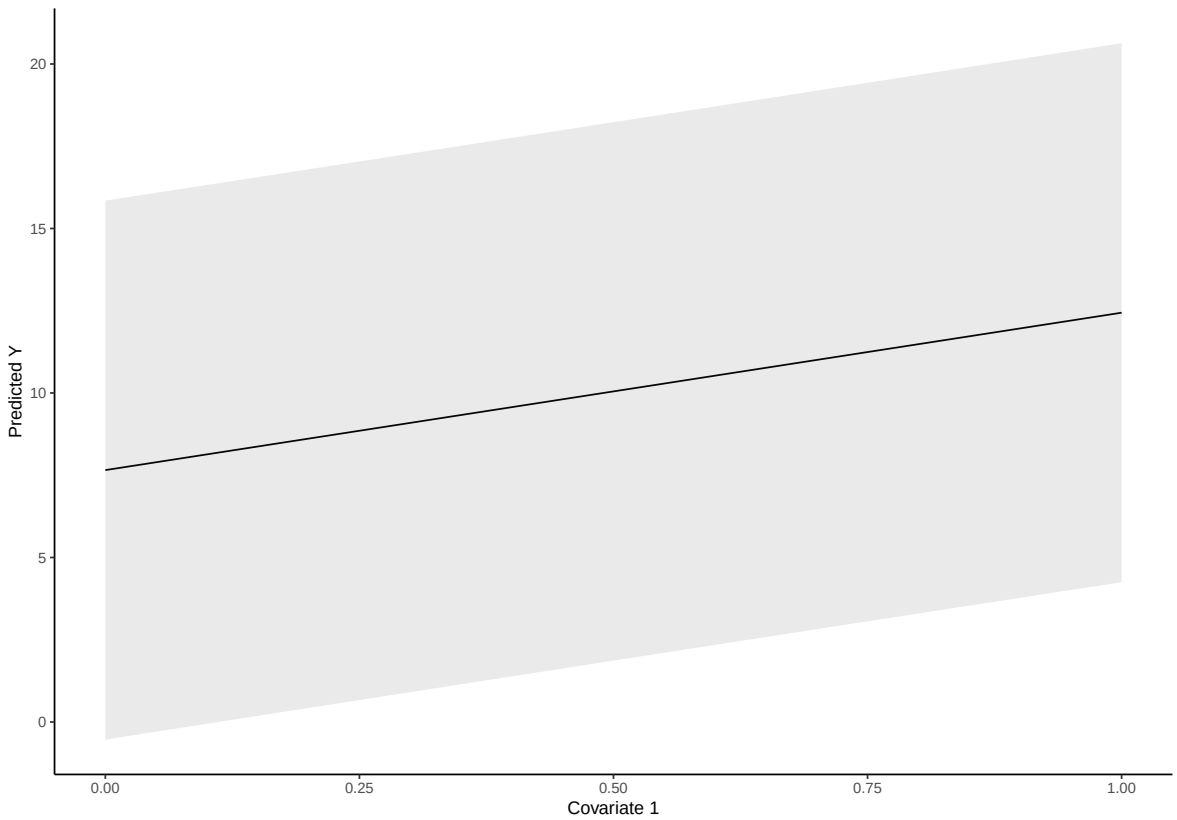
2.4 Calculate quantities of interest - S1

```
simPI.w1 <- simPI.w1 %>%
  as_tibble() %>%          # Coerce it into a tibble
  bind_cols(w1 = w1range) # Combine hypo w1 with predicted y

head(simPI.w1)             # Inspect
```

```
## # A tibble: 6 x 4
##     fit     lwr   upr    w1
##   <dbl>   <dbl> <dbl> <dbl>
## 1  7.66 -0.536   15.8  0
## 2  7.90 -0.295   16.1  0.05
## 3  8.13 -0.0545  16.3  0.1
## 4  8.37  0.186   16.6  0.15
## 5  8.61  0.426   16.8  0.2
## 6  8.85  0.666   17.0  0.25
```

2.4 Calculate quantities of interest - S1

```r
# ggplot2
theme_set(theme_classic())

ggplot(simPI.w1, aes(x = w1, y = fit, ymax = upr, ymin = lwr)) +
  geom_line() +
  geom_ribbon(alpha = 0.1) +
  labs(y = "Predicted Y", x = "Covariate 1")
```



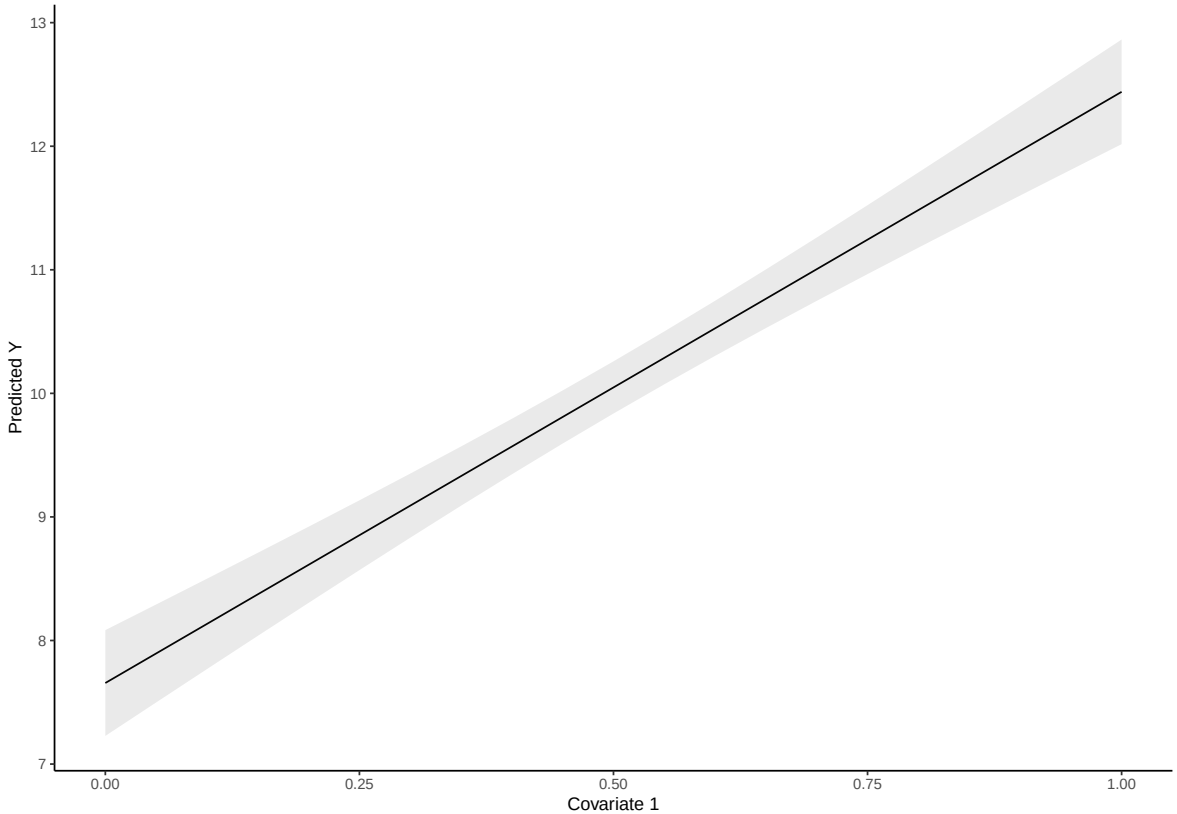
2.4 Calculate quantities of interest - S1

```
# Calculate confidence intervals using predict()
simCI.w1 <- predict(ls.result,
                    newdata = xhypo,
                    interval = "confidence",
                    level = 0.95)

simCI.w1 <- simCI.w1 %>%
  as_tibble() %>%
  bind_cols(w1 = w1range)

head(simCI.w1)
```

```
## # A tibble: 6 x 4
##     fit   lwr   upr    w1
##   <dbl> <dbl> <dbl> <dbl>
## 1  7.66  7.23  8.08  0
## 2  7.90  7.50  8.29  0.05
## 3  8.13  7.77  8.50  0.1
## 4  8.37  8.04  8.71  0.15
## 5  8.61  8.30  8.92  0.2
## 6  8.85  8.57  9.13  0.25
```

2.4 Calculate quantities of interest - S1

```r
# Plot confidence intervals
ggplot(simCI.w1, aes(x = w1, y = fit, ymax = upr, ymin = lwr)) +
  geom_line() +
  geom_ribbon(alpha = 0.1) +
  labs(y = "Predicted Y", x = "Covariate 1")
```



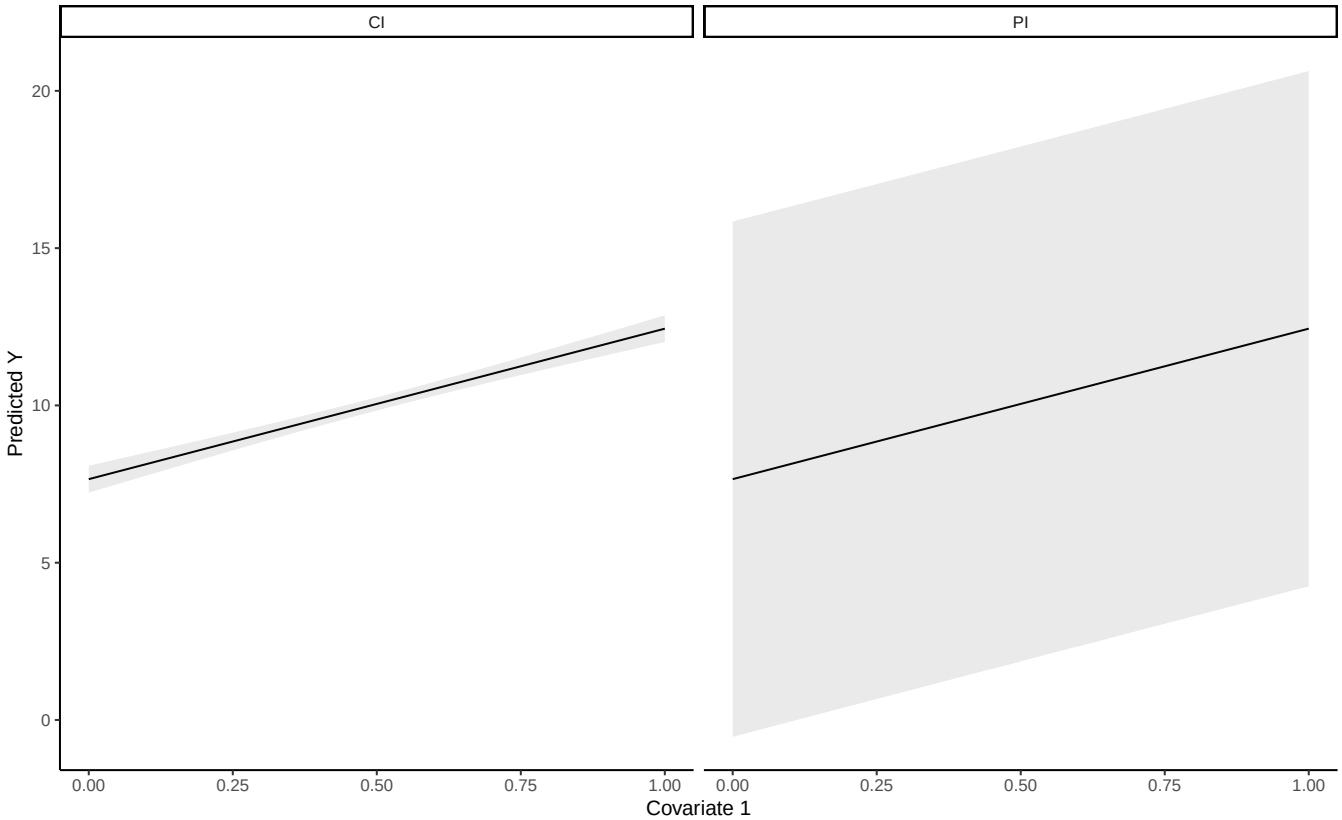
2.4 Calculate quantities of interest - S1

```
# How to plot two plots side by side
# In ggplot2, we need to combine two datasets but also create a new variable
# to identify whether the data are from prediction or confidence intervals
simALL.w1 <- bind_rows(
  simPI.w1 %>% mutate(type = "PI"),
  simCI.w1 %>% mutate(type = "CI")
  )

head(simALL.w1)
```

```
## # A tibble: 6 x 5
##     fit     lwr   upr    w1 type
##   <dbl>   <dbl> <dbl> <dbl> <chr>
## 1  7.66 -0.536   15.8  0    PI
## 2  7.90 -0.295   16.1  0.05 PI
## 3  8.13 -0.0545  16.3  0.1  PI
## 4  8.37  0.186   16.6  0.15 PI
## 5  8.61  0.426   16.8  0.2  PI
## 6  8.85  0.666   17.0  0.25 PI
```

2.4 Calculate quantities of interest - S1

```r
# Plot confidence intervals and predictive intervals side by side
ggplot(simALL.w1, 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)
```



2.4 Calculate quantities of interest - S2

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
head(xhypo)                                    # Inspect
```

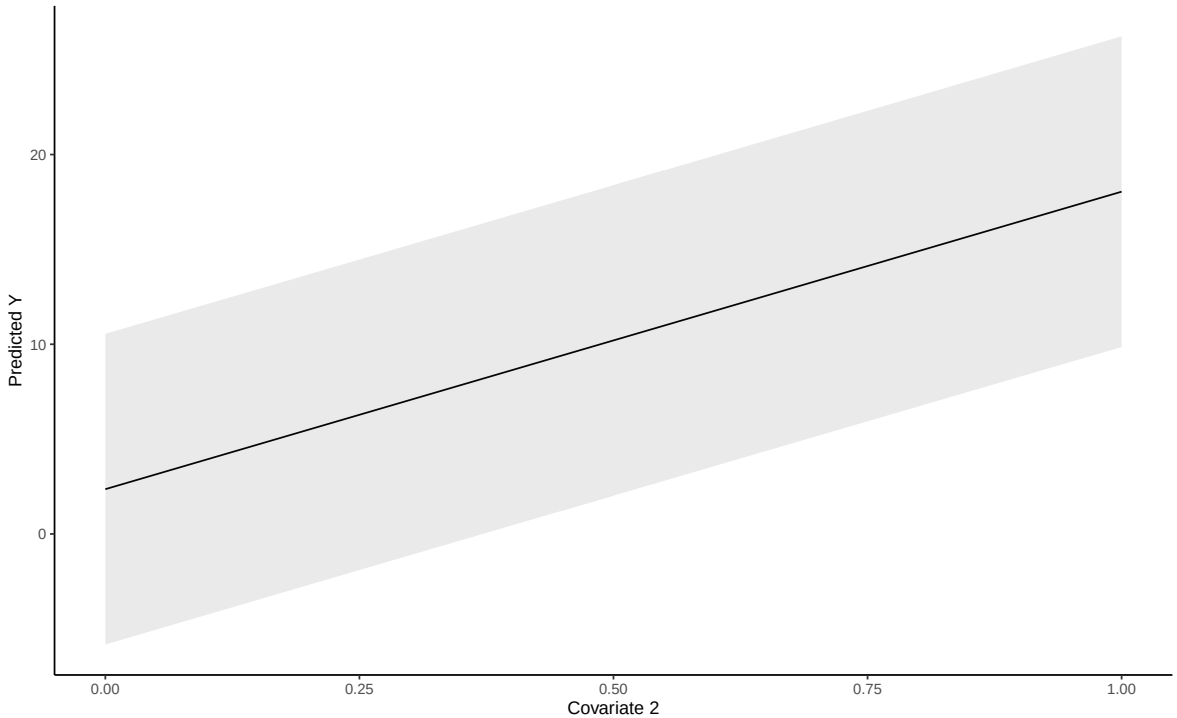
```
## # A tibble: 6 x 2
##      w1    w2
##   <dbl> <dbl>
## 1 0.504  0
## 2 0.504  0.05
## 3 0.504  0.1
## 4 0.504  0.15
## 5 0.504  0.2
## 6 0.504  0.25
```

2.4 Calculate quantities of interest - S2

```r
# Calculate predicted values
simPI.w2 <- predict(ls.result,                    # A model object
                    newdata = xhypo,              # New dataset
                    interval = "prediction",      # What kind of intervals?
                    level = 0.95)                 # What levels of confidence?

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

2.4 Calculate quantities of interest - S2

```r
# Plot the predictive intervals for hypothetical w2
ggplot(simPI.w2, aes(x = w2, y = fit, ymax = upr, ymin = lwr)) +
  geom_line() +
  geom_ribbon(alpha = 0.1) +
  labs(y = "Predicted Y", x = "Covariate 2")
```



The prediction intervals do not capture the heteroskedastic nature of the outcome variable with respect to covariate 2. Any better solution?

3. Homework: Recap HW1

1. Read carefully and answer **ALL** questions
2. Reproducibility

- Don't impute numbers by hand!
- R is object-oriented: Use objects to do calculation, especially when you run simulations.
- It is important for you, **co-authors**, and **reviewers**.

3. Homework: Recap HW1

3. Recycle/Repeatability

- Make the computer do the work (Be a lazy coder)
  - Assign objects properly
  - If you have to repeat task X four times, write code that repeats it for you, don't copy and paste the code four times.
  - Loops & functions

3. Homework: Recap HW1

```r
# HW1 Problem1
N <- 30
pi <- 0.49

HandFunc <- function(x){
  factorial(N)/(factorial(x)*factorial(N-x)) *
  (pi^x) * (1-pi)^(N-x)
}

cat("The probabilities of sampling 22 and 16 males are",
    round(HandFunc(22),4), round(HandFunc(16),4), "respectively.")
```

```
## The probabilities of sampling 22 and 16 males are 0.0041 0.1293 respectively.
```

3. Homework: Recap HW1

4. Readability

- Try to produce a computational narrative PDF in `R markdown`.

3. Homework: Question HW2

- One common problem when knitting: the math mode environment doesn't like white space or empty line + Try `\begin{alinged}` instead of `\begin{split}` + `R Markdown` guide is here
- Email subject: **MLE510HW2**
- File name: **MLE510HW2KenyaAmano**