CSSS/POLS 510 MLE Lab

Lab 4.Quantities of Interest (QoI) and Binary Model

Ramses Llobet

October 20, 2023

Agenda

1. Review HW1 and Pre-view HW2
2. Last lab review
3. Quantities of Interest
4. Binary model *(if time allows)*

1. Review HW1: Sanity checks!

```r
dbinom(x=16, size=30, prob=0.49)
```

```
## [1] 0.1293457
```

```r
set.seed(12345)

sims <- 100       # 100 simulations

nmen <- rep(NA,sims)
for (i in 1:sims) {
  nmen[i] <- sum(sample(c(0,1),
                        30,
                        replace = TRUE,
                        prob = c(0.51, 0.49) ))
}
sum(nmen==16)/length(nmen) # sum of trials with 16 males
```

```
## [1] 0.16
```

1. Review HW1: Sanity checks!

```r
dbinom(x=16, size=30, prob=0.49)
```

```
## [1] 0.1293457
```

```r
set.seed(12345)

sims <- 1000       # 1000 simulations

nmen <- rep(NA,sims)
for (i in 1:sims) {
  nmen[i] <- sum(sample(c(0,1),
                        30,
                        replace = TRUE,
                        prob = c(0.51, 0.49) ))
}
sum(nmen==16)/length(nmen) # sum of trials with 16 males
```

```
## [1] 0.131
```

1. Review HW1: Sanity checks!

```r
dbinom(x=16, size=30, prob=0.49)
```

```
## [1] 0.1293457
```

```r
set.seed(12345)

sims <- 10000       # 10000 simulations

nmen <- rep(NA,sims)
for (i in 1:sims) {
  nmen[i] <- sum(sample(c(0,1),
                        30,
                        replace = TRUE,
                        prob = c(0.51, 0.49) ))
}
sum(nmen==16)/length(nmen) # sum of trials with 16 males
```

```
## [1] 0.1285
```

1. Review HW1: Sanity checks!

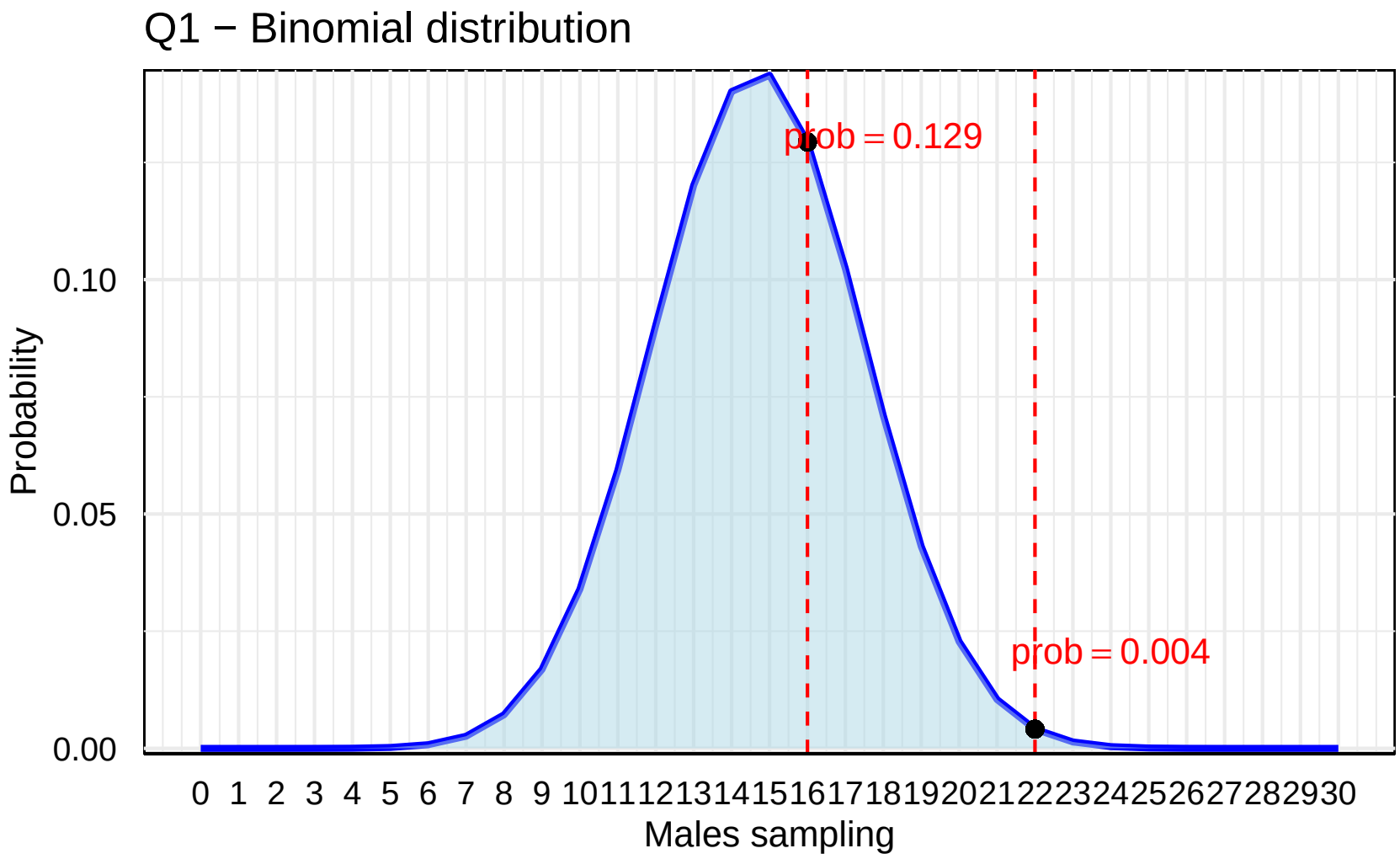
```r
dbinom(x=16, size=30, prob=0.49)
```

```
## [1] 0.1293457
```

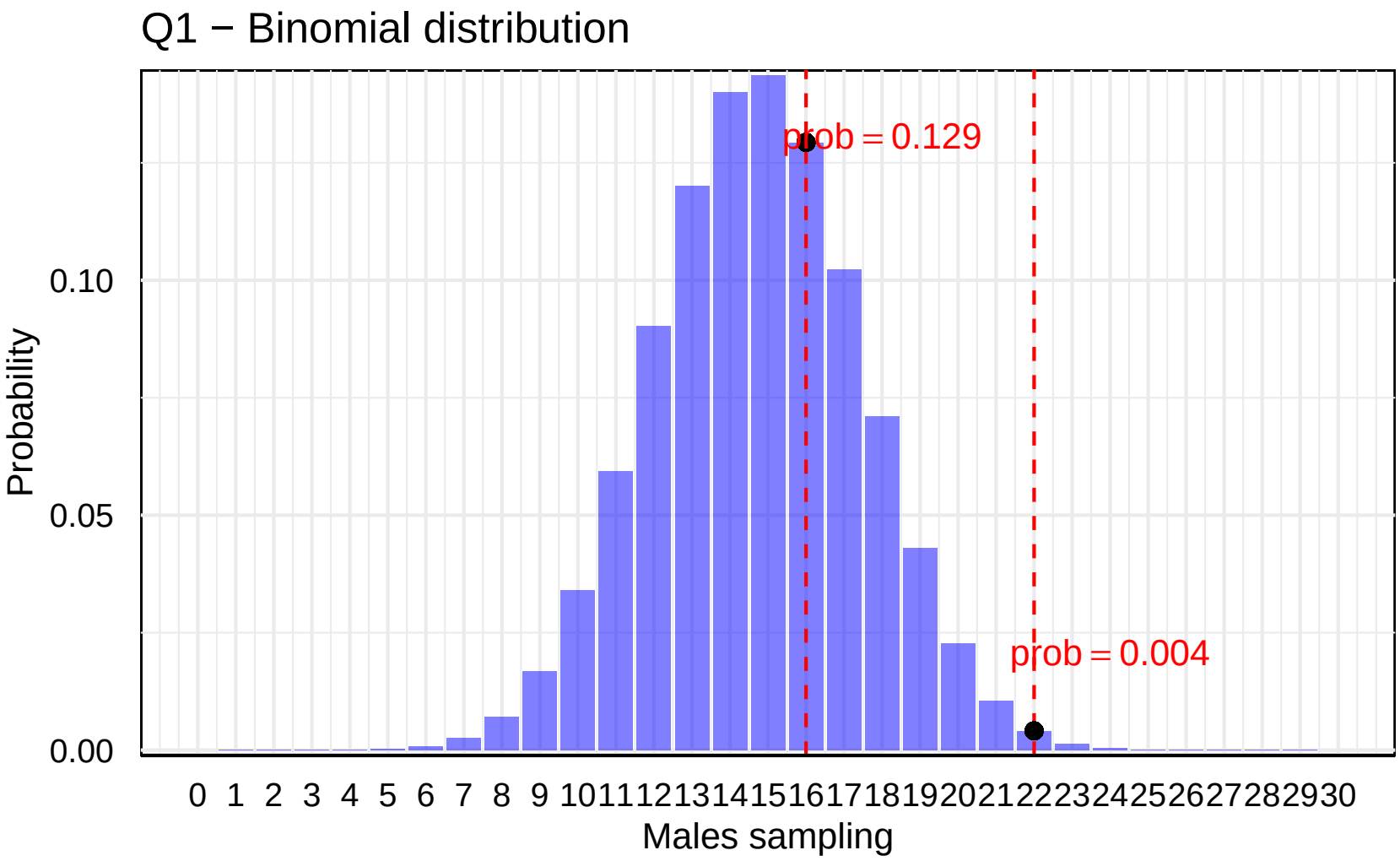
```r
set.seed(12345)

sims <- 100000      # 100000 simulations

nmen <- rep(NA,sims)
for (i in 1:sims) {
  nmen[i] <- sum(sample(c(0,1),
                        30,
                        replace = TRUE,
                        prob = c(0.51, 0.49) ))
}
sum(nmen==16)/length(nmen) # sum of trials with 16 males
```

```
## [1] 0.1294
```

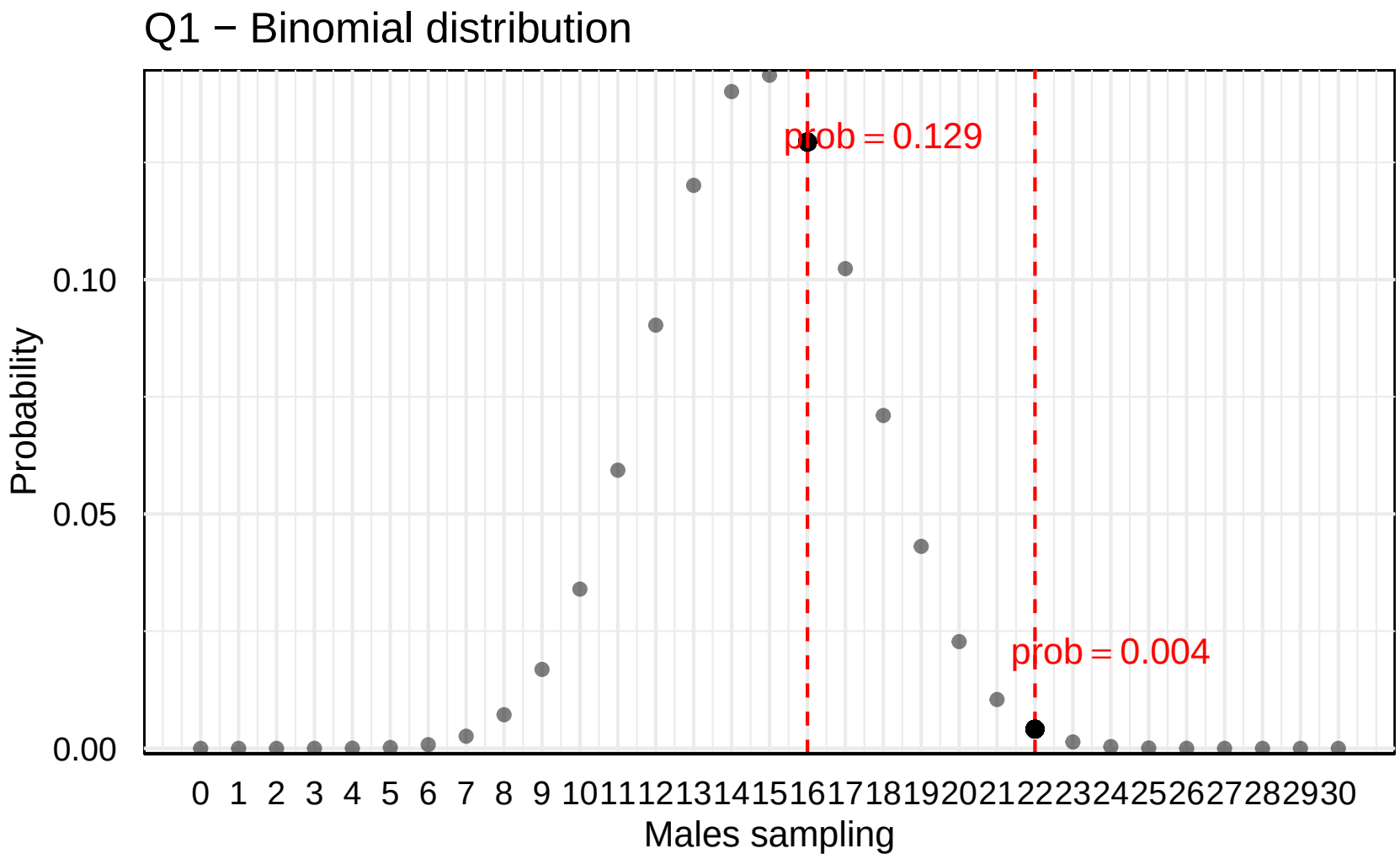
1. Review HW1: Probability Mass Functions



1. Review HW1: Probability Mass Functions



1. Review HW1: Probability Mass Functions



1. Review HW1: summary

- When displaying P**M**Fs, provide visuals with a discrete sample space.
- **Sanity checks**:
  - Use R built-in or packages programs/functions to **double-check**.
  - Increase simulations/sample **size**.
    - Convergence in probability ($N \to \infty$)

1. HW2 Question

Problem 1:

$$
\begin{aligned}
f_{\text{Bern}}(y \mid \pi) &= \pi^{y}(1-\pi)^{1-y} \\
\Pr(y \mid \pi) &= \prod_{i=1}^{n} \pi_i^{y_i}(1-\pi_i)^{1-y_i} \\
L(\pi \mid y) &= k(\mathrm{y})\Pr(y \mid \pi) \\
&= k(\mathrm{y}) \prod_{i=1}^{n} \pi_i^{y_i}(1-\pi_i)^{1-y_i}
\end{aligned}
$$

1. HW2 Question

Problem 2:

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

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

2. Last lab review: Least Squares

- Linear homoskedastic:

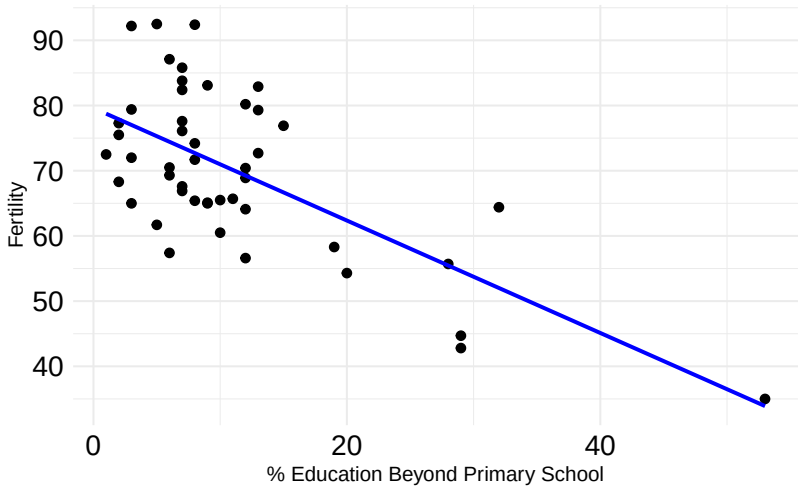
$$Y_i = x_i\beta + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2)$$

- Estimating the slope:

$$\hat{\beta}_j = \frac{\text{Cov}(X_j, Y)}{\text{Var}(X_j)}$$

- Matrix Algebra:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$



2. Last lab review: Least Squares

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \tag{1}$$

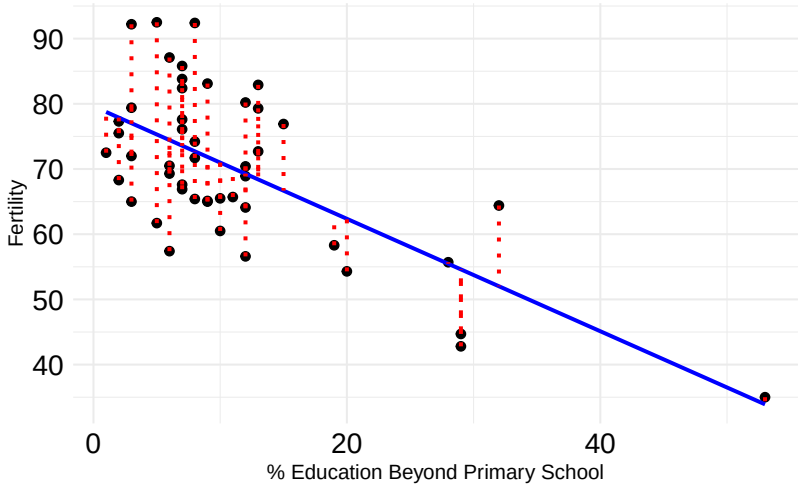
$$\hat{e}_i = y_i - \hat{y}_i \tag{2}$$

$$\sum \hat{e}_i^{\,2} = \sum (y_i - \hat{y}_i)^2 \tag{3}$$

$$\sum \hat{e}_i^{\,2} = \sum (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \tag{4}$$

Note: (3) and (4) are equivalent, and provide the *Residual Sum of Squares* (RSS) or *Sum of Squared Residuals* (SSR).

Choosing the best combination of $\hat{\beta}_j$ that minimizes the SSR provide the best fit for the model.



2. Last lab review: MLE

- How do we estimate the MLE?
  1. Define a probability model (PDF): $Y_i \sim N(\mu_i, \sigma^2)$.
  2. Derive the log-likelihood function.
  3. Reduce to sufficient statistics and substitute systematic component.
  4. Use `optim()` or any other function to find the maxima.

2. Normal homoskedastic

Two **different** notations for the **same** model.

LS notation:

$$\varepsilon \sim N(0, \sigma^2) \qquad (\textit{stochastic})$$
$$Y_i = x_i\beta \qquad (\textit{systematic})$$
$$Y_i = x_i\beta + \varepsilon \qquad (\textit{stochastic} + \textit{systematic})$$

MLE notation:

$$Y_i \sim N(\mu_i, \sigma^2) \qquad (\textit{stochastic})$$
$$\mu_i = x_i\beta \qquad (\textit{systematic})$$
$$Y_i \sim N(x_i\beta, \sigma^2) \qquad (\textit{stochastic} + \textit{systematic})$$

3. MLE general notation

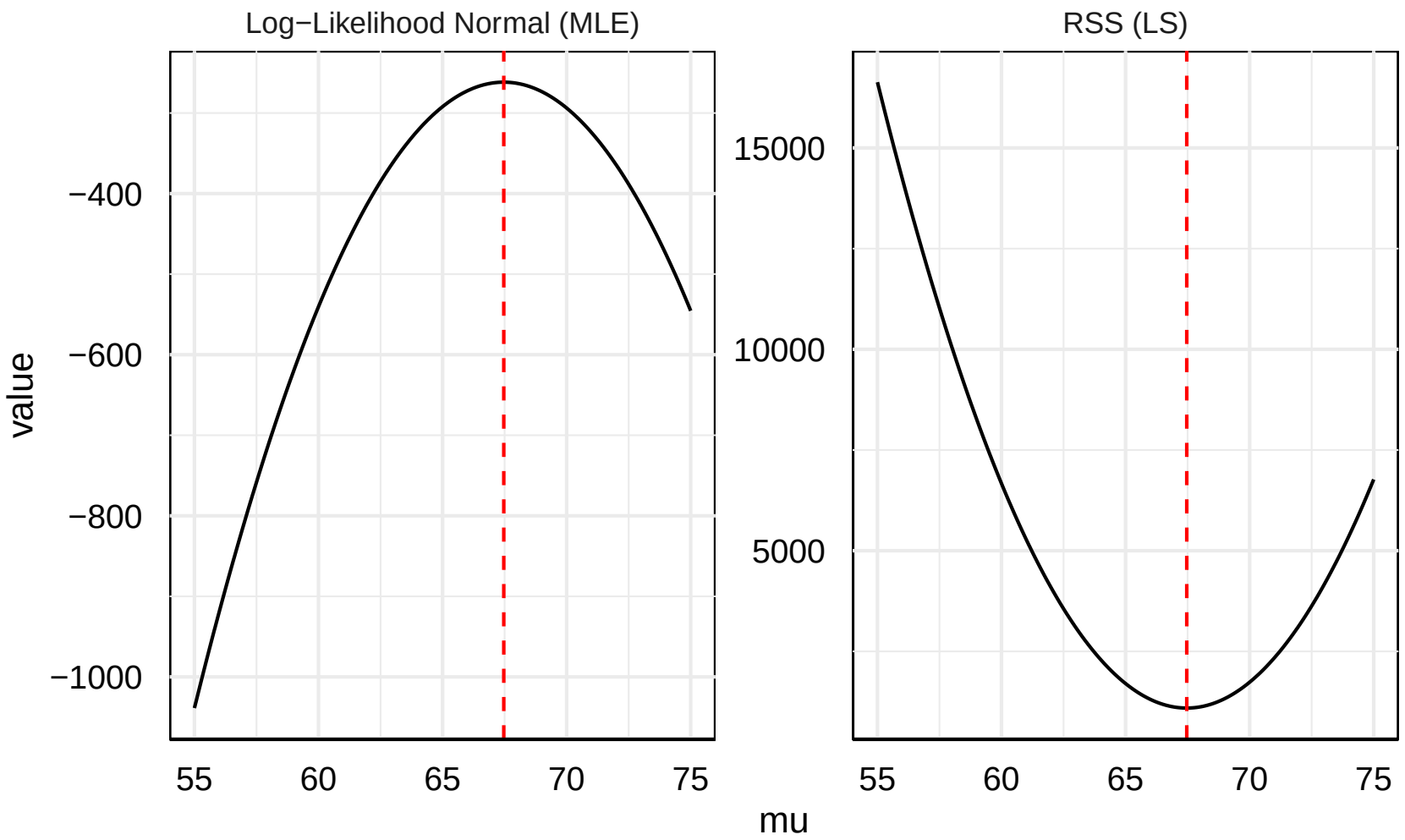
Stochastic component:

$$Y_i \sim f(\theta_i, \alpha) \quad \text{(stochastic)}$$
$$\theta_i = g(\boldsymbol{x}_i \boldsymbol{\beta}) \quad \text{(systematic)}$$

where

- Y_i is a random outcome variable.
- $f(.)$ is a probability density function.
- θ_i is a systematic feature of the PDF that varies over i.
- α is an ancillary parameter (feature of f that we treat as constant).
- $g(.)$ functional form for reparametrization of the data model.
- x_i explanatory variables vector.
- β vector of effect parameters.

2. Last lab review: MLE - Homoskedastic normal



2. Heteroskedastic normal

- 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

2. Heteroskedastic normal

Stochastic component:

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

Systematic components:

$$\begin{aligned}\mu_i &= \boldsymbol{x}_i \boldsymbol{\beta} \\ \sigma_i^2 &= \exp(\boldsymbol{z}_i \boldsymbol{\gamma})\end{aligned}$$

In our lab example, we simulate those, where $\boldsymbol{x}_i = \boldsymbol{z}_i = \boldsymbol{w}_i$.

We estimate **6 parameters**:

$$\begin{aligned}\mu_i &= \beta_0 + \beta_1 w_1 + \beta_2 w_2 \\ &= 0 + 5 w_1 + 15 w_2 \\ \sigma_i^2 &= \exp(\gamma_0 + \gamma_1 w_1 + \gamma_2 w_2) \\ &= \exp(1 + 0 w_1 + 3 w_2)\end{aligned}$$

2.1 Generating heteroskedastic normal data

```
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

- 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:

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

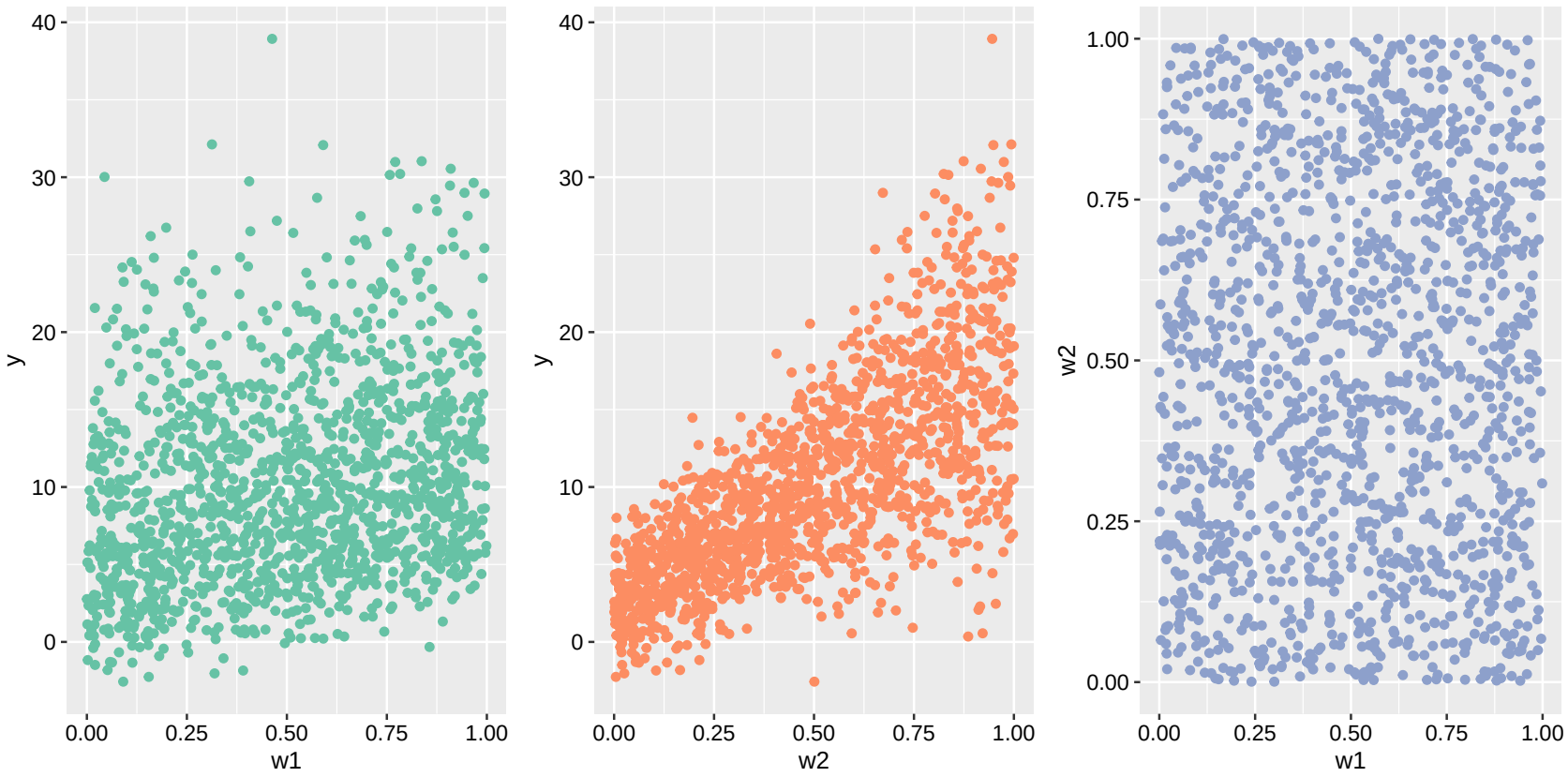
- Fast rule:a $\mathbf{4} \times 3$ matrix multiplied by a $3 \times \mathbf{1}$ one $\rightarrow$ a $\mathbf{4} \times \mathbf{1}$ matrix

2.1 Generating heteroskedastic normal data

```
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")
```

The clearest way is to put the parameters in the correct matrix format. But %*% is smart enough to make a vector conformable when multiplied with a matrix.

2.1 Generating heteroskedastic normal data



2.2 Fit OLS - `lm()`

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

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

2.3 Log-likelihood for Heteroskedastic Normal

$$\mathrm{P}\left(y_1|\mu_1, \boldsymbol{\sigma}_1^2\right) = \left(2\pi\sigma_1^2\right)^{-1/2} \exp\left[\frac{-\left(y_1-\mu_1\right)^2}{2\sigma_1^2}\right] \qquad \text{[Normal PDF for an observation]}$$

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

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2|\mathbf{y}) = k(y) * \mathrm{P}\left(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2\right) \qquad \text{[Likelihood function]}$$

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2|\mathbf{y}) \propto \mathrm{P}\left(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2\right) \qquad \text{[Drop likelihood axiom*]}$$

2.3 Log-likelihood derivation for Heteroskedastic Normal

$$\log \mathcal{L}\left(\boldsymbol{\mu}, \boldsymbol{\sigma}^2 | \mathbf{y}\right) = log\left(\prod_{i=1}^{n} k(y_i) * \left(2\pi\sigma_i^2\right)^{-1/2} * \exp\left[\frac{-\left(y_i - \mu_i\right)^2}{2\sigma_i^2}\right]\right) \quad \text{[Converted to log likelihood]}$$

$$\log \mathcal{L}\left(\boldsymbol{\mu}, \boldsymbol{\sigma}^2 | \mathbf{y}\right) = \sum_{i=1}^{n} log\left(k(y_i) * \left(2\pi\sigma_i^2\right)^{-1/2} * \exp\left[\frac{-\left(y_i - \mu_i\right)^2}{2\sigma_i^2}\right]\right) \quad \text{[Converted to log likelihood]}$$

$$\log \mathcal{L}\left(\boldsymbol{\mu}, \boldsymbol{\sigma}^2 | \mathbf{y}\right) \propto -\frac{1}{2}\sum_{i=1}^{n} \log \sigma_i^2 - \frac{1}{2}\sum_{i=1}^{n} \frac{(y_i - \mu)^2}{\sigma_i^2} \quad \text{[simplify and drop constants]}$$

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

2.3 Log-likelihood function

```r
# A likelihood function for ML heteroskedastic Normal
llk.hetnormlin <- function(param, y, x, z) {
  # data input:
  x <- as.matrix(x)                          # x (some covariates) as a matrix
  z <- as.matrix(z)                          # z (some covariates) as a matrix

  # create intercept:
  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

  # define number of parameters to estimate:
  b <- param[1 : ncol(x)]                          # Parameters for x
  g <- param[(ncol(x) + 1) : (ncol(x) + ncol(z))] # Parameters for z

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

  # define log-likelihood function
  sum(0.5 * (log(s2) + (y - xb)^2 / s2)) # Likelihood we want to maximize
  }
```

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

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

Why the equation does not match?

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

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

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)
                        )
```

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

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

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

```r
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
```

```r
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()

```r
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
```

```r
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
```

```r
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()`

```r
# 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
```

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

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

3. Quantities of Interest

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

3. Predict QoI

- 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 = ... )`
3. Plot the prediction intervals

3. Predict QoI

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

3 Predict QoI: predicted adn expected values

```r
# First set hypothetical values

hyp_w1 <- seq(from = 0, to = 1, by = 0.05)

# Set up - create a new dataset

xhypo <-
  tibble(w1 = hyp_w1,     # Set up hypothetical values for w1
         w2 =  mean(w2)) # Set up mean for w2)

xhypo  # Inspect
```

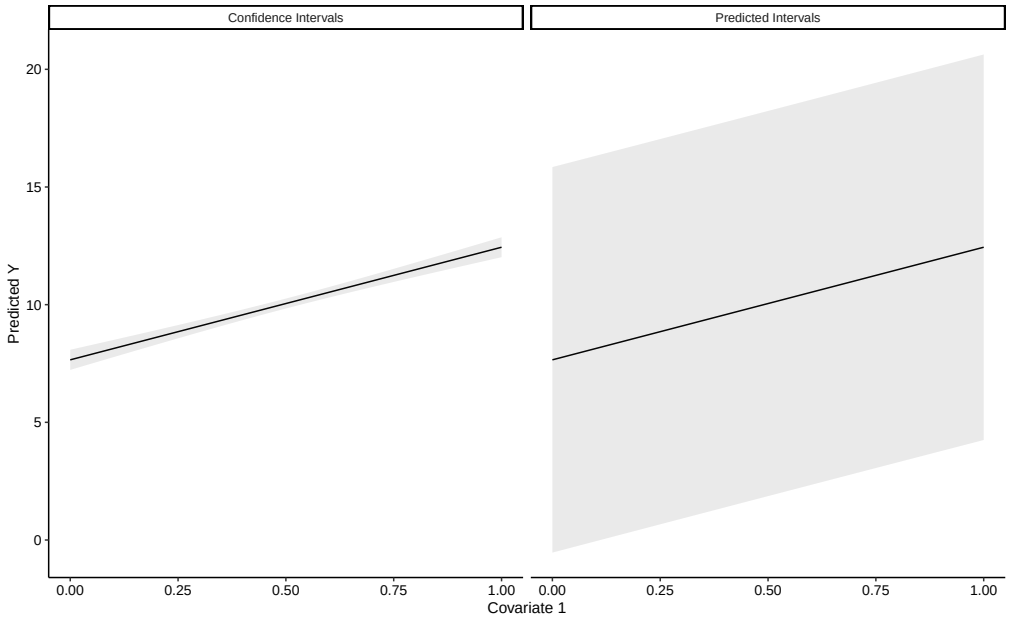
```
## # A tibble: 21 x 2
##       w1    w2
##    <dbl> <dbl>
##  1  0    0.491
##  2  0.05 0.491
##  3  0.1  0.491
##  4  0.15 0.491
##  5  0.2  0.491
##  6  0.25 0.491
##  7  0.3  0.491
```

3 Predict QoI: combine into dataset

```
simPI.w1 <- simPI.w1 %>%
  as_tibble() %>%          # Coerce it into a tibble
  bind_cols(w1 = hyp_w1) # Combine hypo w1 with predicted

simCI.w1 <- simCI.w1 %>%
  as_tibble() %>%
  cbind(w1 = hyp_w1)

# combine the two datasets, make other adjustments
simALL.w1 <-
  rbind(
  simPI.w1 %>% mutate(type = "Predicted Intervals"),
  simCI.w1 %>% mutate(type = "Confidence Intervals")
) %>% as_tibble()
```

3. Predict QoI: visual display

Plot confidence intervals and predictive intervals side by side, here suing `ggplot()`



3.2 Predict QoI

Scenario 2: Vary covariate 2; hold covariate 1 constant

```
w2range <- seq(from = 0, to = 1, by = 0.05)    # Set up hypothetical values for w
w1mean <- mean(w1)                             # Set up mean for w2
xhypo <- crossing(w1 = w1mean,                # Create a new dataset
                  w2 = w2range)                  # crossing() is from tidyr packag
# 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)
```

3.2 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 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

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 valuese of X: X_c

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

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

1.3 Simulating QoI using `simcf`

4. Calculate expected values:
   $\tilde{\mu_c} = g(X_c, \tilde{\beta})$
5. Simulate fundamental uncertainty:
   $\tilde{y_c} \sim f(\tilde{\mu_c}, \tilde{\alpha})$
   $\rightarrow$ `simcf::hetnormsimpv()` ...
   or
6. Compute EVs, First Differences or Relative Risks
   EV: $\mathbb{E}(y|X_{c1})$
   $\rightarrow$ `simcf::logitsimev()` ...
   FD: $\mathbb{E}(y|X_{c2}) - \mathbb{E}(y|X_{c1})$
   $\rightarrow$ `simcf::logitsimfd()` ...
   RR: $\frac{\mathbb{E}(y|X_{c2})}{\mathbb{E}(y|X_{c1})}$
   $\rightarrow$ `simcf::logitsimrr()` ...

3 Simulating QoI: MASS package

```r
# Draw parameters from the model predictive distribution
# Note: mvnorm function is from MASS package

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
```

3 Simulating QoI using `simcf` :S1

```
# Separate into the simulated betas and simulated gammas

# general code; don't be intimidated:
simbetas <- simparam[ , 1:(ncol(xcovariates) + 1)]
simgammas <- simparam[ , (ncol(simbetas) + 1):ncol(simparam)]

# Specify model formulas
formula <- as.formula(y ~ w1 + w2)
```

3 Simulating QoI using `simcf : cfmake`

```r
# Create hypothetical scenarios for w2
hyp_w2 <- seq(from = 0, to = 1, by = 0.05)

xhypo <- cfMake(formula=formula,
                data=data,
                nscen = length(hyp_w2)) # number of scenarios
```

3 Simulating QoI using `simcf` : loop

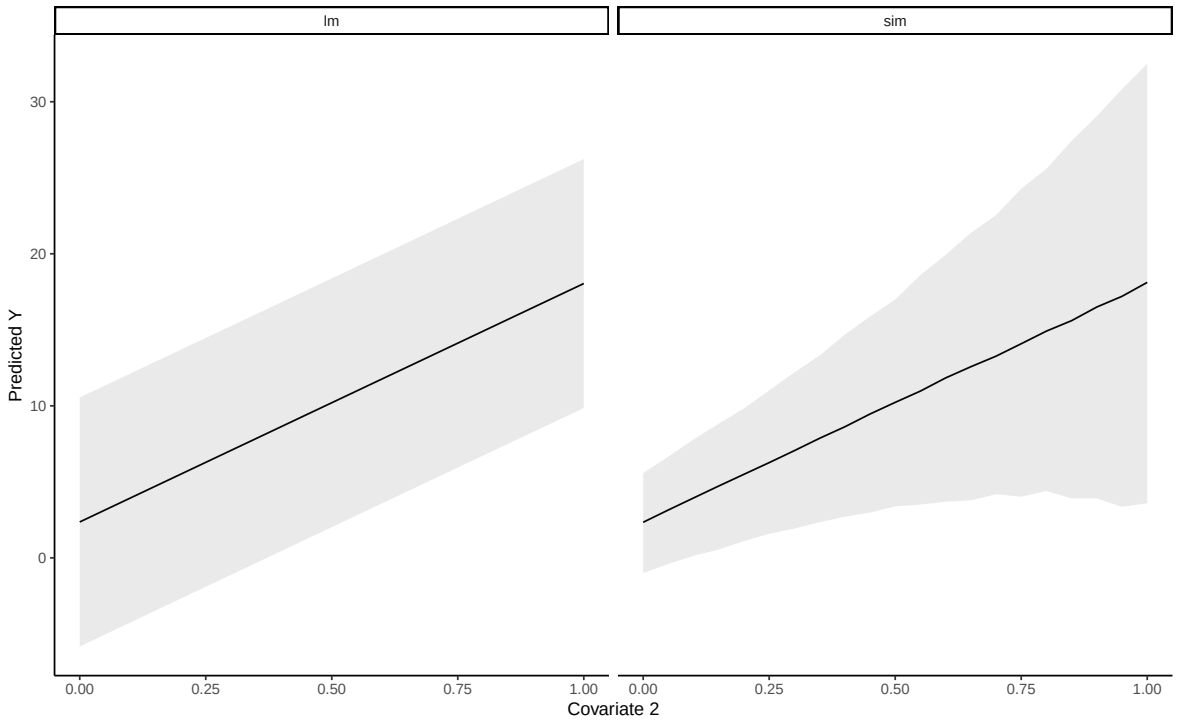
```
# Use `cfChange` and loop function to loop through each hypothetical x values
for (i in 1:length(hyp_w2)) {
  xhypo <- cfChange(xhypo,
                    covname="w2",
                    x = hyp_w2[i],
                    scen = i)
}

## Only for the Heteroskedasti normal, repeat for "z" covaraites
zhypo <- cfMake(formula=formula,
                data=data,
                nscen = length(hyp_w2))

for (i in 1:length(hyp_w2)) {
  zhypo <- cfChange(zhypo, "w2", x = hyp_w2[i], scen = i)
}
```

3 Simulating QoI using `simcf` :S1

```r
# Simulate predictive intervals for heteroskedastic linear models
# `hetnormsimpv()` is from simcf package
simRES.w2 <- hetnormsimpv(xhypo, simbetas,
                          zhypo, simgammas,
                          ci = 0.95,
                          constant = 1,
                          varconstant = 1)
simRES.w2 <- simRES.w2 %>%
  bind_rows() %>%
  bind_cols(w2 = hyp_w2)

# Combine two dataframes
simPI.w2 <- simPI.w2 %>%
  rename(pe = fit, lower = lwr, upper = upr)

simALL.w2 <- bind_rows(
  simRES.w2 %>% mutate(method = "sim"),
  simPI.w2 %>% mutate(method = "lm")
)
```

3 Simulating QoI using `simcf` :S2

```
# Plot the predictive intervals for hypothetical w2
ggplot(simALL.w2, 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)
```



4 Logit model: Voting example

Let's open `RStudio` and [**nesLogitPlot.r**]

F I N