MATH 392 Problem Set 9

Sam D. Olson

1. **GLM with Gaussian Response**

Consider the special case of the generalized (simple) linear model where we assume independent Gaussian errors and are linked to the linear predictor using the identify function (vanilla simple linear regression). We can summarize that model as follows (note that in the notation used in this problem, everything is a scalar):

i. If $X = x$, then $Y = \beta_0 + \beta_1 x + \epsilon$ where the β_j are (unknown) parameters and the ϵ is a random variable.
ii. $\epsilon \overset{iid}{\sim} N(0, \sigma^2)$ for some (unknown) parameter σ^2.

When this model was first derived, and only (i) was assumed, the parameters were estimated by minimizing the residual sum of squares (these are the least-squares estimates, $\hat{\beta}^{LS}$). This yielded

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$
$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$
$$\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=1}^n (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x))^2$$

Now that we have added (ii), we have specified a full density function for the random variable Y, $f(y|X = x, \beta_0, \beta_1, \sigma^2)$, which enables a familiar route to estimation: maximum likelihood.

Find the maximum likelihood estimates of β_0, β_1, and σ^2.

a) Provide the derivation of closed-form solutions, if they exist. For reference, if $X \sim N(\mu, \sigma^2)$, $f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}}$.

(*): Formulating the likelihood function

Note: For X_i, $f(y_i|x_i, \beta_0, \beta_1, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - (\beta_0 + \beta_1 x_i))^2}{2\sigma^2}}$

Thus, for $X_1, ..., X_n$, we may write:

$$f(y_1|x_1, \beta_0, \beta_1, \sigma^2)...f(y_n|x_n, \beta_0, \beta_1, \sigma^2) = \prod_{i=1}^n f(y_i|x_i, \beta_0, \beta_1, \sigma^2) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - (\beta_0 + \beta_1 x_i))^2}{2\sigma^2}}$$

To ease our future calculations, we take the log-likelihood, turning the above relation into a summation instead of a product. To that end, let L denote the log, giving us a log-likelihood of:

$$L(\beta_0, \beta_1, \sigma^2) = L(f(y_1|x_1, \beta_0, \beta_1, \sigma^2)...f(y_n|x_n, \beta_0, \beta_1, \sigma^2)) = L(\prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - (\beta_0 + \beta_1 x_i))^2}{2\sigma^2}}) = \sum_{i=1}^n log(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - (\beta_0 + \beta_1 x_i))^2}{2\sigma^2}})$$

We isolate each of the logs into the following relation:

(**):

$$L(\beta_0, \beta_1, \sigma^2) = nlog(\frac{1}{\sqrt{2\pi}}) + nlog(\frac{1}{\sqrt{\sigma^2}}) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2$$

For the following derivations, we maximize the above relation with respect to our parameter of interest, which yields the maximum likelihood estimate. This involves deriving the equation (w.r.t. our parameter) and setting equal to zero, then solving for that parameter.

(1): β_0^{MLE}

$\frac{\partial(L(\beta_0,\beta_1,\sigma^2))}{\partial\beta_0} = \frac{\partial}{\partial\beta_0}(-\frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2)$

Note, the expansion of the summation as follows:

(***): $(y_i - (\beta_0 + \beta_1 x_i))^2) = y_i^2 - 2y_i(\beta_0 + \beta_1 x_i) + (\beta_0 + \beta_1 x_i)^2 = y_i^2 - 2y_i\beta_0 - 2y_i\beta_1 x_i + \beta_0^2 + 2\beta_0\beta_1 x_i + \beta_1^2 x_i^2$

Using relation (***), we may simplify the derivation as:

$\frac{\partial}{\partial\beta_0}(-\frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2) = -\frac{1}{\sigma^2}\sum_{i=1}^{n} -2y_i - 2\beta_0 + 2\beta_1 x_i$

Setting the above expression equal to zero, we may derive the MLE as:

$\beta_0^{MLE} = \bar{y}_n - \beta_1\bar{x}_n$

(2): β_1^{MLE}

$\frac{\partial(L(\beta_0,\beta_1,\sigma^2))}{\partial\beta_1} = \frac{\partial}{\partial\beta_1}(-\frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2)$

Using relation (***), we may simplify the derivation as:

$\frac{\partial}{\partial\beta_1}(-\frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2) = -\frac{1}{\sigma^2}\sum_{i=1}^{n} -2y_i x_i + 2\beta_0 x_i + 2\beta_1 x_i^2$

Setting the above expression equal to zero, we may derive the MLE as:

$\beta_1^{MLE} = \frac{\sum_{i=1}^{n}(x_i - \bar{x}_n)(y_i - \bar{y}_n)}{\sum_{i=1}^{n}(x_i - \bar{x}_n)^2} = \frac{ss_{xy}}{ss_x}$

(3): σ^2_{MLE}

$\frac{\partial(L(\beta_0,\beta_1,\sigma^2))}{\partial\sigma^2} = \frac{\partial}{\partial\sigma^2}(-nlog(\sigma^2) - \frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2)$

$\frac{\partial(L(\beta_0,\beta_1,\sigma^2))}{\partial\sigma^2} = -\frac{n}{\sigma^2} + \frac{1}{(\sigma^2)^2}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2)$

Setting the above equation equal to zero, we have:

$\sigma^2_{MLE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2)$

Commentary

The MLEs of β_0, β_1, and σ^2 are the same as the least squares estimates of β_0, β_1, and σ^2. We will take advantage of this when finding the MLEs using numerical optimization in part b).

b) Describe in pseudocode (or actual R code) how to find them using numerical optimization.

MLE

There are a number of options available for estimating the MLE, ranging from general optimzation using 'optim', generating a generalized linear model, to dedicated maximum likelihood package such as 'maxLik'.

The process is as follows:

(1): Specify likelihood function (2): Specify parameters (3): Generate data (X's and Y's) (4): Generate Model/Estimates (optim, maxLik, glm, etc.)

Code Example - MLE

```
# specify likelihood function, using gaussian assumptions
set.seed(1999)
l_gauss <- function(n, sigma, B, X, Y) {
  (-(n / 2) * log(2 * pi)) - (n * log( sigma)) - ((1 / (2 * (sigma^2))) * (sum((Y - (X %*% B))^2)))
}
# Set Parameters
n <- 500
p <- 1
sigma <- 2
# Generate X
X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2))
# Set Beta values
B <- c(2, 4)
# Generate Y
Y <- rnorm(n, mean = (X %*% B), sd = sqrt(sigma))
# Optim example
optim(par = c(0, 0), fn = l_gauss, X = X, Y = Y, n = n, sigma = sigma)
```

```
## $par
## [1] -3.183805e+55  4.996335e+54
##
## $value
## [1] -6.351068e+112
##
## $counts
## function gradient
##      501       NA
##
## $convergence
## [1] 1
##
## $message
## NULL
```

```
# maxLik example
ml <- maxLik(l_gauss, start = c(0, 0), X = X, Y = Y, n = n, sigma = sigma)
ml$estimate
```

```
## [1] 2.090525 4.006276
```

```r
# glm example
df <- data.frame(Y = Y, x1 = X[, 2])
coef(glm(Y ~ x1, data = df, family = "gaussian"))
```

```
## (Intercept)          x1
##    2.090525    4.006276
```

Commentary

In the above code, the estimates of the coefficients are quite off. However, the maxLik estimates and the glm estimates are similar.

---

2. **Logistic Regression MLEs: Bias, Variance, and Shrinkage**:

For this problem you'll be working in a setting when the design matrix including the intercept is an $n \times 2$ matrix X and the response is Bernoulli with an inverse logit link function to the linear predictor (logistic regression). Since there is no closed form of the MLE, you'll be using simulation, meaning you'll need to specify values for all of the parameters needed to generate data from the Logistic Regression model.

a) *How does a single estimate compare with the true mean function?* Simulate one data set and fit one model using the MLEs. Construct a scatterplot with the simulated data, the estimated mean function ($\hat{E}(Y|X = x)$) and the true mean function ($\hat{E}(Y|X = x)$).

```r
l_bern <- function(B, X, Y) {
  p <- 1/(1 + exp(- X %*% B))
  sum(Y * log(p) + (1 - Y) * log(1 - p))
}
# Set data
p <- 1
n <- 500
# Generate X
X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2))
# Set B
B <- c(-1, 2)
# Simulate Y
Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% B)))
# Log Likelihood
l_bern(B, X, Y)
```

```
## [1] -250.0697
```

```r
l_bern(c(0, 0), X, Y)
```

```
## [1] -346.5736
```

```r
# Using optim
optim(par = c(0, 0), fn = l_bern, X = X, Y = Y)
```

```
## $par
## [1] 36.163917 -0.157243
##
## $value
## [1] -12326.93
##
## $counts
## function gradient
##       87       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
# Dedicated optimizer
maxLik(l_bern, start = c(0, 0), X = X, Y = Y)
```

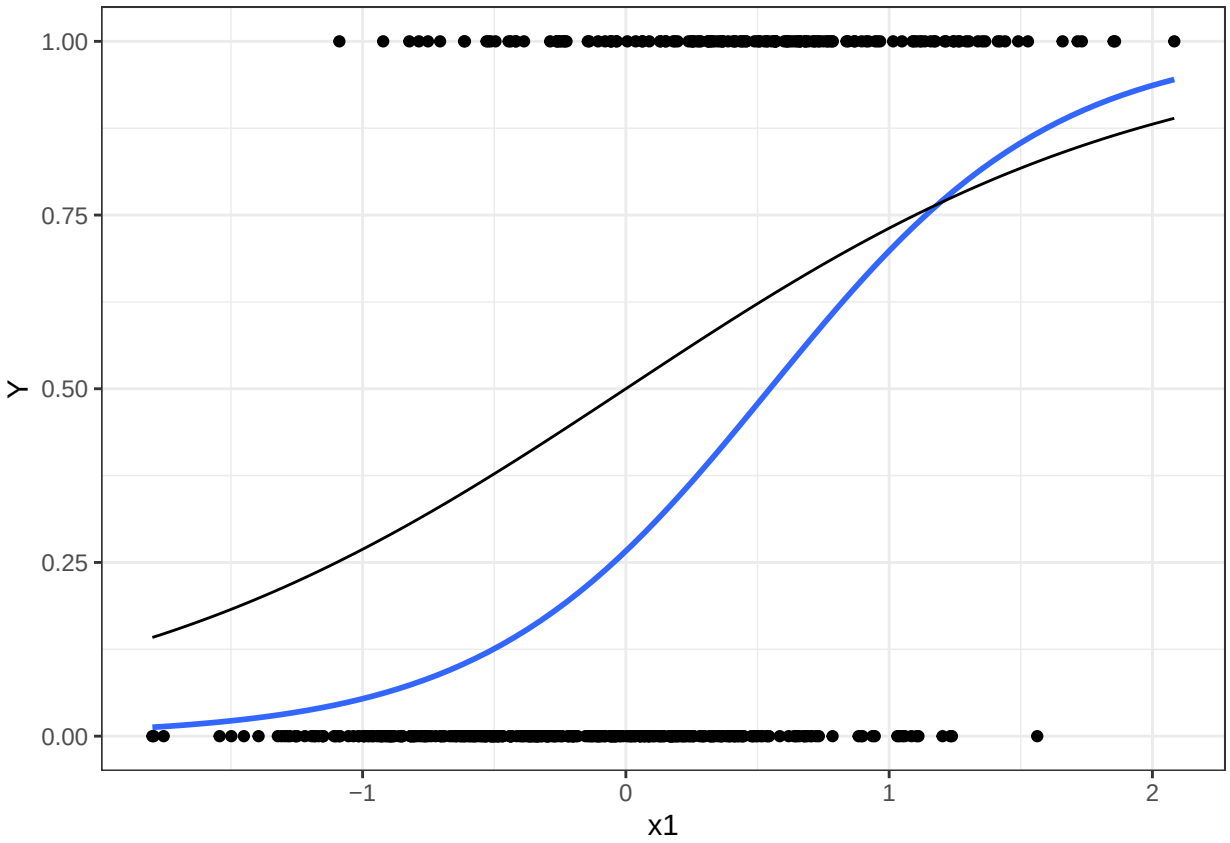
```
## Maximum Likelihood estimation
## Newton-Raphson maximisation, 5 iterations
## Return code 1: gradient close to zero
## Log-Likelihood: -249.7181 (2 free parameter(s))
## Estimate(s): -1.012918 1.852514
```

```
# Glm method
df <- data.frame(Y = Y, x1 = X[, 2])
coef(glm(Y ~ x1, data = df, family = "binomial"))
```

```
## (Intercept)          x1
##   -1.012918    1.852514
```

```
fit <- (glm(Y ~ x1, data = df, family = "binomial"))
```

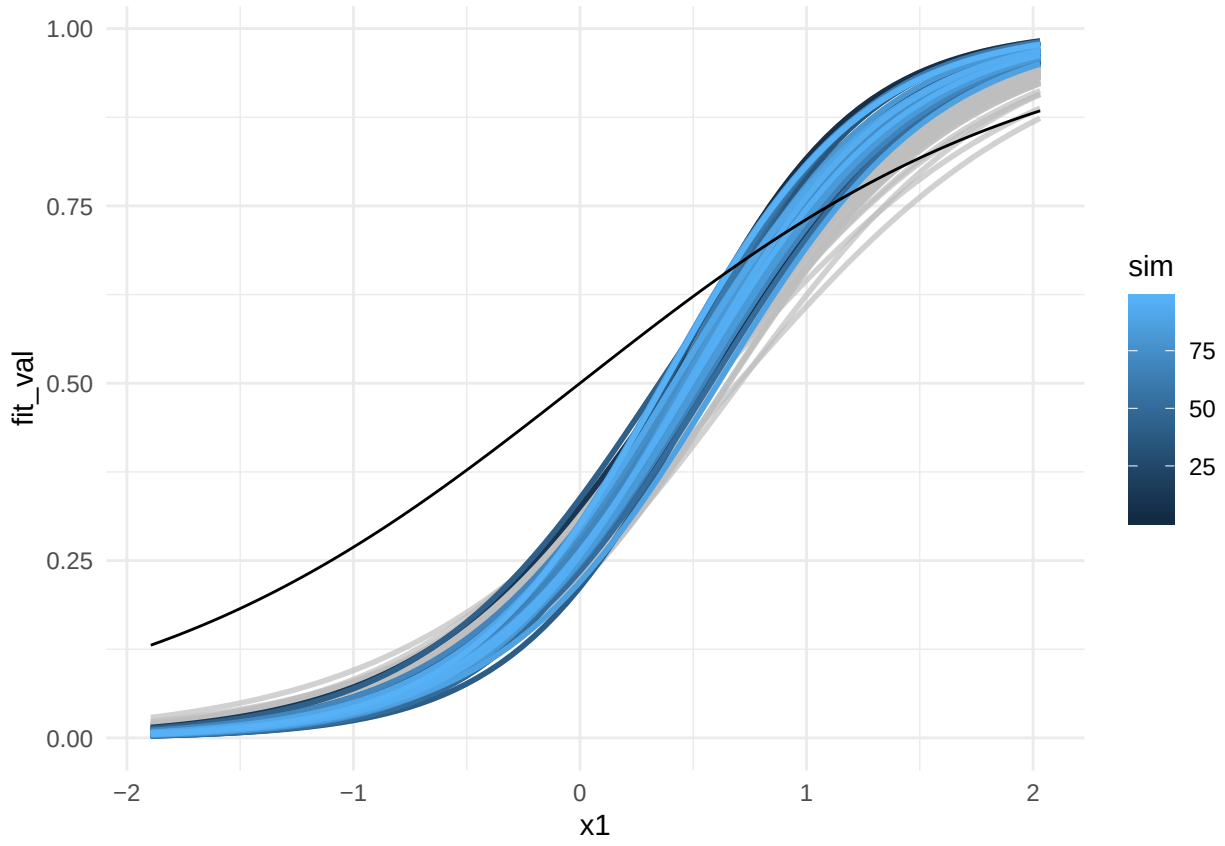
```
ggplot(df, aes(x = x1, y = Y)) +
  geom_point() +
  geom_smooth(method = "glm",
    method.args = list(family = "binomial"),
    se = FALSE) +
  stat_function(fun = function(x) (1 / (1 + exp(-x)))) +
  theme_bw()
```



b) *Is the MLE Biased?* Simulate many data sets and fit many models using MLE. Create a plot similar to the previous, but with *all* of the fitted models' mean functions plotted. To make this more complex plot intelligible, I recommend the gghighlight package.

```
# Set data
p <- 1
n <- 500
# Generate X
X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2))
# Set B
B <- c(-1, 2)
it <- 100
data <- data.frame()
# Simulate Y
for(i in 1:it) {
  Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% B)))
  df <- data.frame(Y = Y, x1 = X[, 2])
  fit <- (glm(Y ~ x1, data = df, family = "binomial"))
  fit_val <- fit$fitted.values
  sim <- i
  df <- cbind( sim , df, fit_val)
  data <- rbind(data, df)
}
```

```
ggplot(data, aes(x = x1, y = fit_val, color=sim)) +
  geom_smooth(aes(group=sim), method = "glm",
   method.args = list(family = "binomial"),
   se = FALSE) +
  gghighlight(max(fit_val) > 0.95) +
  stat_function(fun = function(x) 1 / (1 + exp(-x)))  +
  theme_minimal()
```



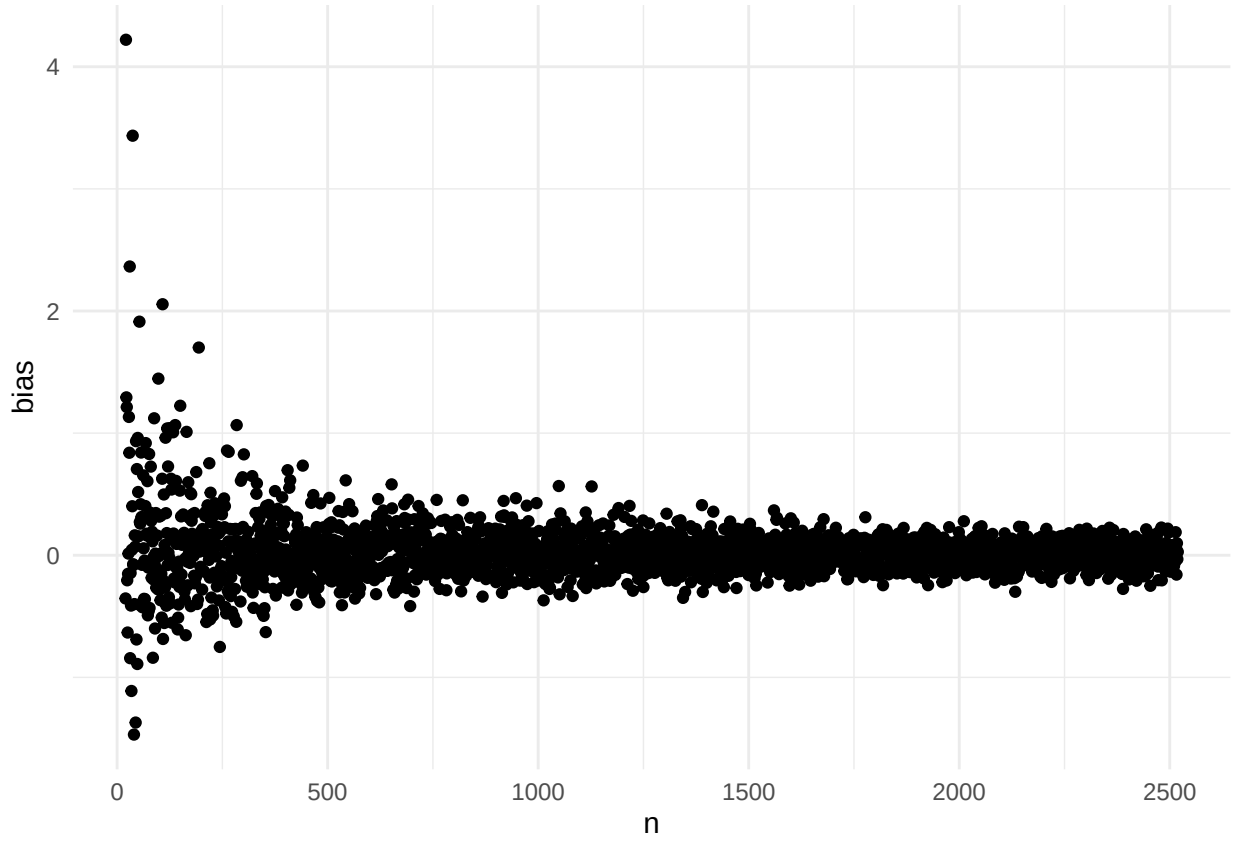

```
ggplot(data, aes(x = x1, y = fit_val, color=sim)) +
  geom_smooth(aes(group=sim), method = "glm",
   method.args = list(family = "binomial"),
   se = FALSE) +
  gghighlight(max(fit_val) > 0.95) +
  stat_function(fun = function(x) 1 / (1 + exp(-x)))  +
  theme_minimal() +
  facet_wrap(~ sim)
```

c) *How does the bias of an estimate change with sample size for a particular value of the parameter?* For a single fixed value of β_1, construct a plot that shows the relationship between n and the bias of the corresponding MLE.

```
# Set B
B <- c(-1, 2)
# Calc
```

```r
it <- 2500
beta <- rep(NA, it)
bias <- rep(NA, it)
df <- data.frame()
for (j in 1:it) {
  p <- 1
  n <- j + 20
  # Generate X
  X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2))
  Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% B)))
  df <- data.frame(Y = Y, x1 = X[, 2])
  fit <- glm(Y ~ x1, data = df, family = "binomial")
  beta[j] <- coef(fit)[2] # beta_1
  bias[j] <- beta[j] - 2
}
# Generate dataset
n <- seq(20, 2519, by=1)
df <- cbind.data.frame(n, bias)
```

```r
ggplot(df, aes(x = n, y = bias)) +
  geom_point() +
  theme_minimal()
```



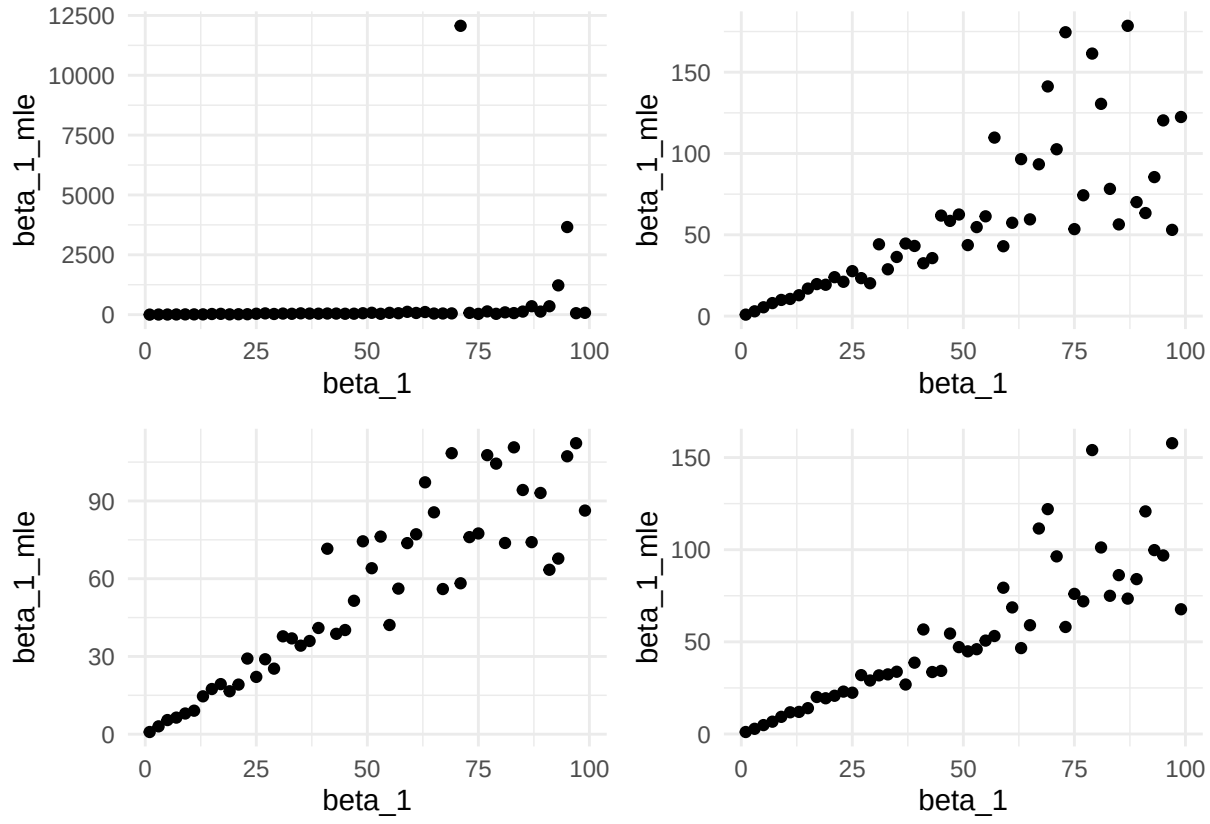
d) *How does the bias of an estimate change with sample size for multiple values of the parameter?* Extend the idea of the previous plot by expressing the relationship between the value of β_1 and the correspond-

ing element of $\hat{\beta}^{MLE}$ for various fixed values of β_1. Examine this relationship at a handful of sample sizes n.

```
# Set B
beta_1 <- seq(1, 100, by=2)
beta <- matrix(nrow=50, ncol=2)
for(i in 1:length(beta_1)){
  beta[i,] <- c(-1, beta_1[i])
}
# simulation
p <- 1
n <- 200
it <- length(beta_1)
X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2))
df <- data.frame()
beta_1_mle <- rep(NA, it)
for (j in 1:length(beta_1)) {
  Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% beta[j,])))
  df <- data.frame(Y = Y, x1 = X[, 2])
  fit <- glm(Y ~ x1, data = df, family = "binomial")
  beta_1_mle[j] <- coef(fit)[2] # beta_1
}
df_1 <- cbind.data.frame(beta_1, beta_1_mle)

# simulation
n <- 500
X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2))
df <- data.frame()
beta_1_mle <- rep(NA, it)
for (j in 1:length(beta_1)) {
  Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% beta[j,])))
  df <- data.frame(Y = Y, x1 = X[, 2])
  fit <- glm(Y ~ x1, data = df, family = "binomial")
  beta_1_mle[j] <- coef(fit)[2] # beta_1
}
df_2 <- cbind.data.frame(beta_1, beta_1_mle)

# simulation
n <- 1000
X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2))
df <- data.frame()
beta_1_mle <- rep(NA, it)
for (j in 1:length(beta_1)) {
  Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% beta[j,])))
  df <- data.frame(Y = Y, x1 = X[, 2])
  fit <- glm(Y ~ x1, data = df, family = "binomial")
  beta_1_mle[j] <- coef(fit)[2] # beta_1
}
df_3 <- cbind.data.frame(beta_1, beta_1_mle)

# simulation
it <- length(beta_1)
X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2))
df <- data.frame()
```

```r
beta_1_mle <- rep(NA, it)
for (j in 1:length(beta_1)) {
  Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% beta[j,])))
  df <- data.frame(Y = Y, x1 = X[, 2])
  fit <- glm(Y ~ x1, data = df, family = "binomial")
  beta_1_mle[j] <- coef(fit)[2] # beta_1
}
df_4 <- cbind.data.frame(beta_1, beta_1_mle)
```

```r
# Generate dataset
library(patchwork)
a1 <- ggplot(df_1, aes(x = beta_1, y = beta_1_mle)) +
  geom_point() +
  theme_minimal()
a2 <- ggplot(df_2, aes(x = beta_1, y = beta_1_mle)) +
  geom_point() +
  theme_minimal()
a3 <- ggplot(df_3, aes(x = beta_1, y = beta_1_mle)) +
  geom_point() +
  theme_minimal()
a4 <- ggplot(df_4, aes(x = beta_1, y = beta_1_mle)) +
  geom_point() +
  theme_minimal()
a1 + a2 + a3 + a4
```



e) *Can I perform shrinkage on logistic coefficients?* The original motivation for ridge regression was to make the $X'X$ matrix in OLS invertible. Statisticians have since realized the practical value of its variance-reducing characteristics when shrinking $\hat{\beta}$ towards zero. Traditional ridge regression is performed by adding a penalty term, $\lambda \sum_{j=1}^{p} \beta_j^2$, to the RSS. In logistic regression, we instead of finding our estimates by minimizing RSS, we choose to maximize the likehood. Although the original motivation of matrix inversion is lost, it can still be perfectly valid and valuable to shink the logistic regression estimates by penalizing the likelihood.

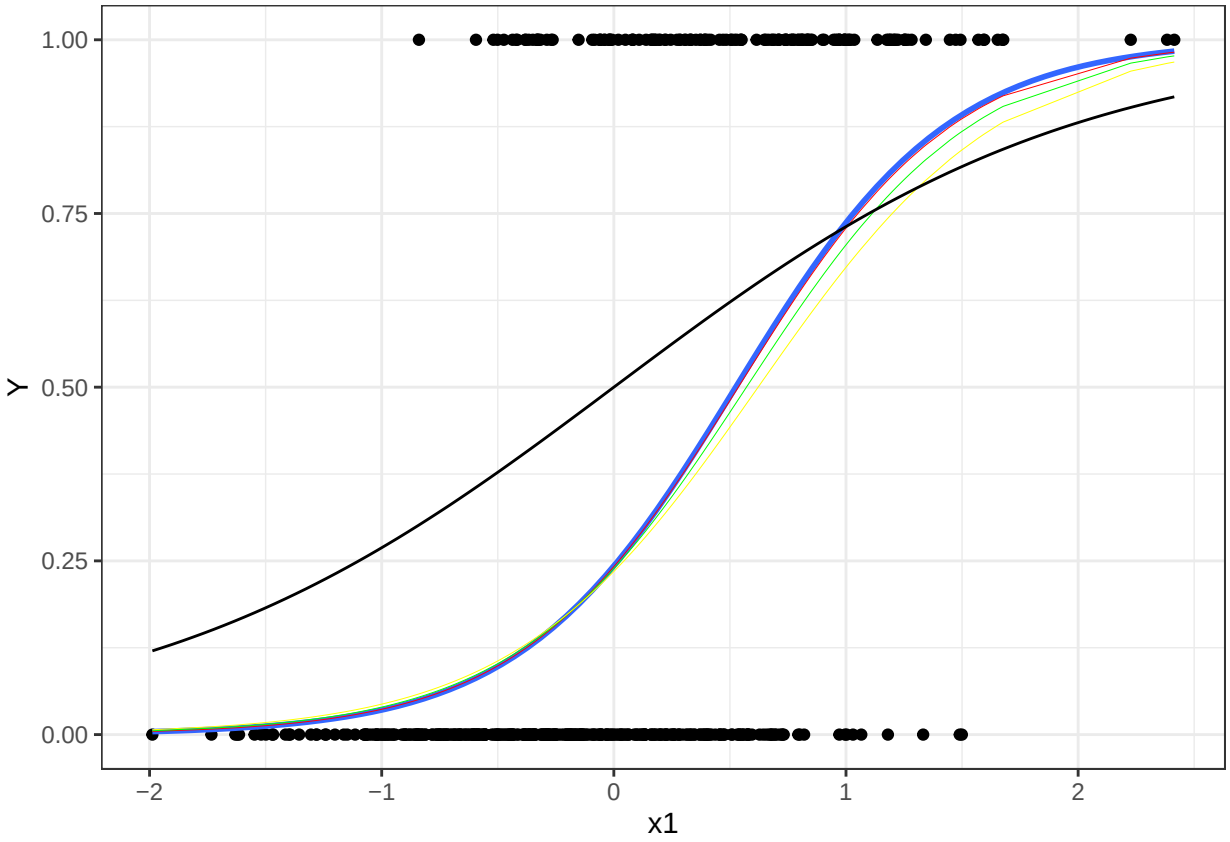
For the same data set that you used to create the plot in part a, find three more estimated mean functions, each one corresponding to a different value of λ, and add them to the plot. Admittedly, this will demonstrate the *downside* of penalized regression: that we have actually *increased* the bias. The plot should have five overlaid sigmoid curves. Play around with the values of λ so that you can the shape of all five on the same plot.

```
penalized_l_bern <- function(B, X, Y, lamby) {
  p <- 1/(1 + exp(- X %*% B))
  sum(Y * log(p) + (1 - Y) * log(1 - p)) - lamby*sum(B)
}
l_bern <- function(B, X, Y) {
  p <- 1/(1 + exp(- X %*% B))
  sum(Y * log(p) + (1 - Y) * log(1 - p))
}
```

```
# Set data
p <- 1
n <- 500
# Generate X
X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2))
# Set B
B <- c(-1, 2)
it <- 500
Y <- rbinom(n, size = 1, prob = 1/(1+exp(- X%*%B)))
lambda <- c(1, 4, 8)
estimates <- maxLik(penalized_l_bern , start = c(0, 0), X = X, Y = Y, lamby = lambda[1])$estimate
B0_ests_1 <- estimates[1]
B1_ests_1 <- estimates[2]
estimates <- maxLik(penalized_l_bern , start = c(0, 0), X = X, Y = Y, lamby = lambda[2])$estimate
B0_ests_2 <- estimates[1]
B1_ests_2 <- estimates[2]
estimates <- maxLik(penalized_l_bern , start = c(0, 0), X = X, Y = Y, lamby = lambda[3])$estimate
B0_ests_3 <- estimates[1]
B1_ests_3 <- estimates[2]

fit_values_1 <- rep(NA, n)
fit_values_2 <- rep(NA, n)
fit_values_3 <- rep(NA, n)

for(i in 1:n){
  fit_values_1 <- 1/(1+exp(- (B0_ests_1 + B1_ests_1 * X[, 2])))
  fit_values_2 <- 1/(1+exp(- (B0_ests_2 + B1_ests_2 * X[, 2])))
  fit_values_3 <- 1/(1+exp(- (B0_ests_3 + B1_ests_3 * X[, 2])))
}

x1 <- X[, 2]
```

```
fit_data_1 <- cbind.data.frame(fit_values_1, x1)
fit_data_2 <- cbind.data.frame(fit_values_2, x1)
fit_data_3 <- cbind.data.frame(fit_values_3, x1)

df <- data.frame(Y = Y, x1 = X[, 2])
fit <- (glm(Y ~ x1, data = df, family = "binomial"))
```

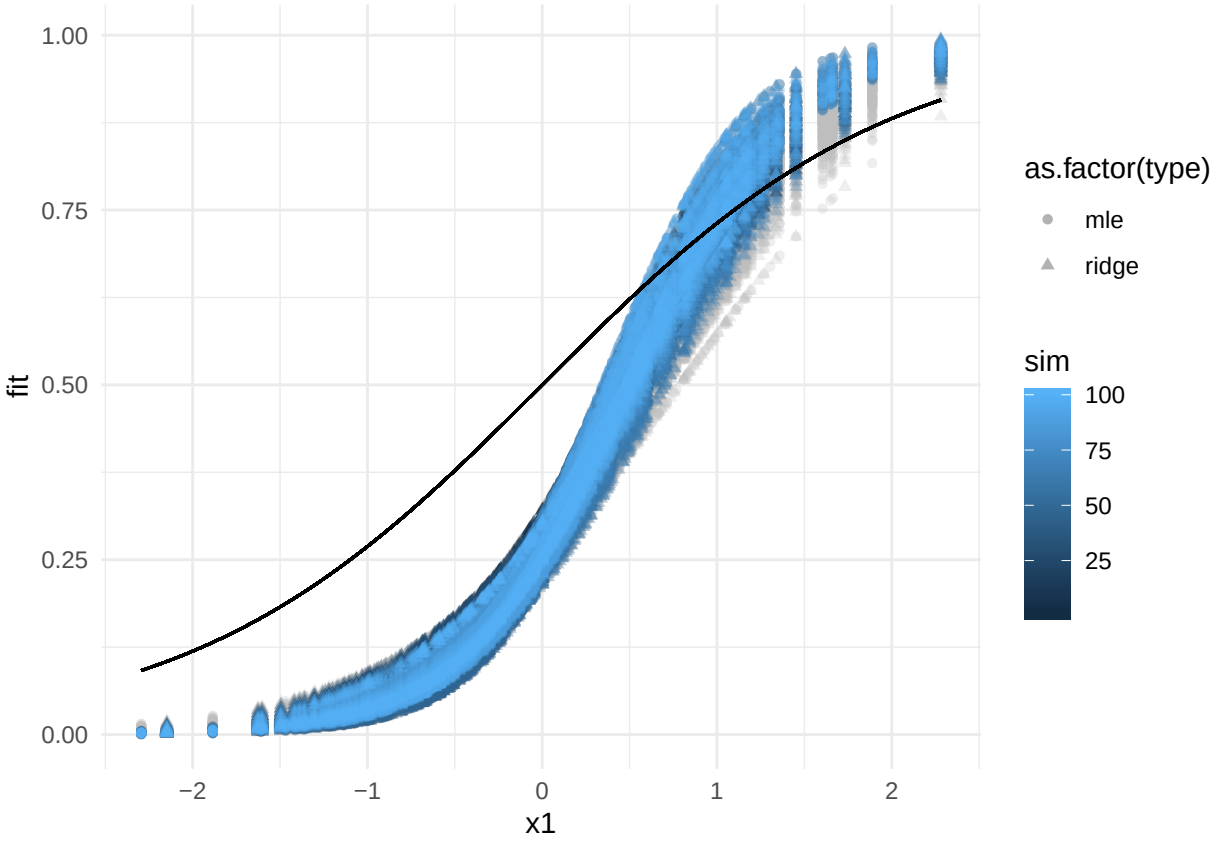
```
ggplot(df, aes(x = x1, y = Y)) +
  geom_point() +
   geom_smooth(method = "glm",
    method.args = list(family = "binomial"),
    se = FALSE) +
  geom_line(data = fit_data_1, mapping = aes(x = x1, y = fit_values_1),
            color = "red", size = 0.05) +
  geom_line(data = fit_data_2, mapping = aes(x = x1, y = fit_values_2),
            color = "green", size = 0.05) +
  geom_line(data = fit_data_3, mapping = aes(x = x1, y = fit_values_3),
            color = "yellow", size = 0.05) +
  stat_function(fun = function(x) (1 / (1 + exp(-x)))) +
  theme_bw()
```



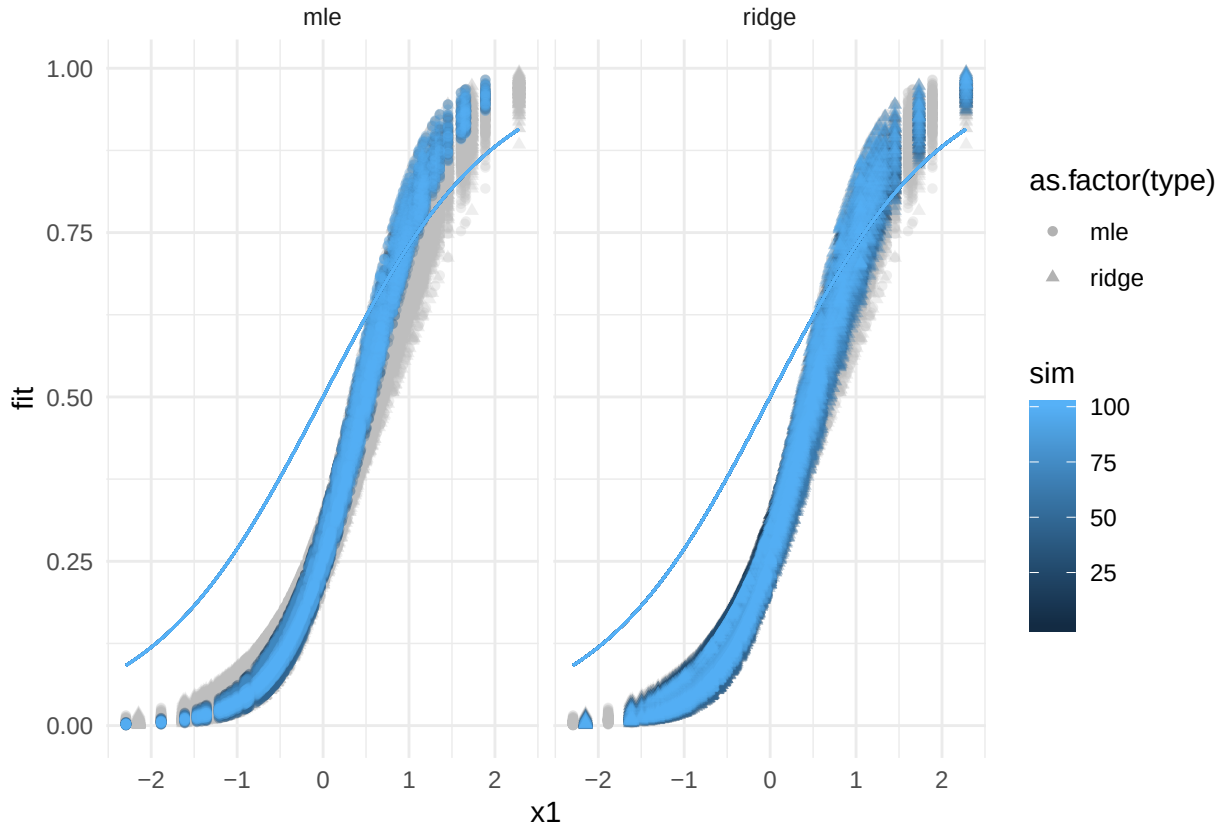
f) To bring in a sense of both bias and variance, select one of your values of λ and use it to replicate the plot from part b, but now with two clumps of sigmoids: one corresponding to the MLE, the other to the ridge estimates. Use color to differentiate between the two clups. Describe what the plot demonstrates about the bias and variance for the MLE and the ridge estimates in this setting.

```r
# Actual Work
# Set data
p <- 1
n <- 500
# Generate X
X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2))
# Set B
B <- c(-1, 2)
lambda <- 4
it <- 100
data <- data.frame()
# Simulate Y
for(i in 1:it) {
  Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% B)))
  estimates <- maxLik(penalized_l_bern , start = c(0, 0), X = X, Y = Y, lamby = lambda)$estimate
  B0_ests_1 <- estimates[1]
  B1_ests_1 <- estimates[2]
  df <- data.frame(Y = Y, x1 = X[, 2])
  fit <- (glm(Y ~ x1, data = df, family = "binomial"))
  fit_mle <- fit$fitted.values
  fit_ridge <- 1/(1+exp(- (B0_ests_1 + B1_ests_1 * X[, 2])))
  sim <- c(i,i)
  type <- c("mle", "ridge")
  fit <- c(fit_mle, fit_ridge)
  df <- cbind(sim , df, fit, type)
  data <- rbind(data, df)
}
```

```r
ggplot(data, aes(x = x1, y = fit, color=sim, shape=as.factor(type), group=interaction(sim, type))) +
  geom_point(alpha=.25) +
  gghighlight(max(fit) > 0.95) +
  stat_function(fun = function(x) 1 / (1 + exp(-x)), color="black")  +
  theme_minimal()
```




```
ggplot(data, aes(x = x1, y = fit, color=sim, shape=as.factor(type), group=interaction(sim, type))) +
  geom_point(alpha=.25) +
  gghighlight(max(fit) > 0.95) +
  stat_function(fun = function(x) 1 / (1 + exp(-x)))  +
  theme_minimal() +
  facet_wrap(~ type)
```



An Extra Portion for Your Consideration

```
# Set B, Lambda values
B <- c(-1, 2)
lambda <- c(1, 4, 8)
# Set Parameters
it <- 500
beta_mle <- rep(NA, it)
beta_ridge1 <- rep(NA, it)
beta_ridge2 <- rep(NA, it)
beta_ridge3 <- rep(NA, it)
bias_mle <- rep(NA, it)
bias_ridge1 <- rep(NA, it)
bias_ridge2 <- rep(NA, it)
bias_ridge3 <- rep(NA, it)
df <- data.frame()
# For loop to generate data
for (j in 1:it) {
  p <- 1
  n <- j + 20
  # Generate X
  X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2))
  Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% B)))
  df <- data.frame(Y = Y, x1 = X[, 2])
```

```r
  fit <- glm(Y ~ x1, data = df, family = "binomial")
  beta_mle[j] <- coef(fit)[2] # beta_1
  bias_mle[j] <- beta_mle[j] - 2
  estimates <- maxLik(penalized_l_bern , start = c(0, 0), X = X, Y = Y, lamby = lambda[1])$estimate
  beta_ridge1[j] <- estimates[2]
  bias_ridge1[j] <- beta_ridge1[j] - 2
  estimates <- maxLik(penalized_l_bern , start = c(0, 0), X = X, Y = Y, lamby = lambda[2])$estimate
  beta_ridge2[j] <- estimates[2]
  bias_ridge2[j] <- beta_ridge2[j] - 2
  estimates <- maxLik(penalized_l_bern , start = c(0, 0), X = X, Y = Y, lamby = lambda[3])$estimate
  beta_ridge3[j] <- estimates[2]
  bias_ridge3[j] <- beta_ridge3[j] - 2
}
# Generate dataset
n <- seq(20, 519, by=1)
type <- rep(c("mle", "ridge_1", "ridge_2", "ridge_3"), each=length(n))
bias <- c(bias_mle, bias_ridge1, bias_ridge2, bias_ridge3)
df <- cbind.data.frame(n, bias, type)
```

```r
ggplot(df, aes(x = n, y = bias, color=type)) +
  geom_point() +
  ylim(-10,10) +
  theme_minimal()
```