Multiple Linear Regression

Logistic Regression

Math 392

Logistic Regression

$$[Y|X = x] \sim \mathrm{Bern}(p = g(x))$$

$$f(y) = p^y (1-p)^{1-y}; \quad y \in \{0, 1\}$$

We form predictions with $E(Y|X = x) = p = g(x)$.

Estimating β

RSS is out, but we have $f(y)$, so: *MLE*.

$$
\begin{aligned}
f(y_1, y_2, \ldots, y_n) &= \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} \\
log(f(y_1, y_2, \ldots, y_n)) &= \sum_{i=1}^{n} y_i \log p_i + (1-y_i)\log(1-p_i) \\
l(\beta) &= \sum_{i=1}^{n} y_i \log(\frac{1}{1+e^{-X\beta}}) + (1-y_i)\log(1-(\frac{1}{1+e^{-X\beta}}))
\end{aligned}
$$

then take derivatives w.r.t. β, set to 0, and ... not solve.

Since there is no closed-form solution, use numerical optimization.

Generalized Linear Models

Describes a relationship between the mean of a response variable Y and an independent set of variables X. Each GLM requires that you specify the

Distribution of the Y

Generally independent draws from specified member of the exponential family, e.g. Normal, Poisson, Binomial, etc.

Linear Predictor

A function of the predictor variables that is linear in the parameters, e.g. $X\beta$.

Link Function g

Links the linear predictor to $E(Y)$.

We can write a function to calculate the Bernoulli log-likelihood.

```r
l_bern <- function(B, X, Y) {
  p <- 1/(1 + exp(- X %*% B))
  sum(Y * log(p) + (1 - Y) * log(1 - p))
}
# can also use dbinom()
```

Simulating Bernoulli data

First set the size of the data.

```
p <- 1
n <- 35
```

Then generate the *X*.

```
library(mvtnorm)
X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2
X
```

```
##       [,1]        [,2]
##  [1,]    1  0.55631727
##  [2,]    1  0.33418991
##  [3,]    1 -0.48288345
##  [4,]    1  0.35334951
##  [5,]    1  0.46369987
##  [6,]    1  0.39317711
##  [7,]    1  0.52765154
##  [8,]    1  1.25041415
##  [9,]    1 -2.54123163
```

Simulating Bernoulli data, cont.

Then set B.

```
B <- c(-1, 2)
```

Finally, simulate Y.

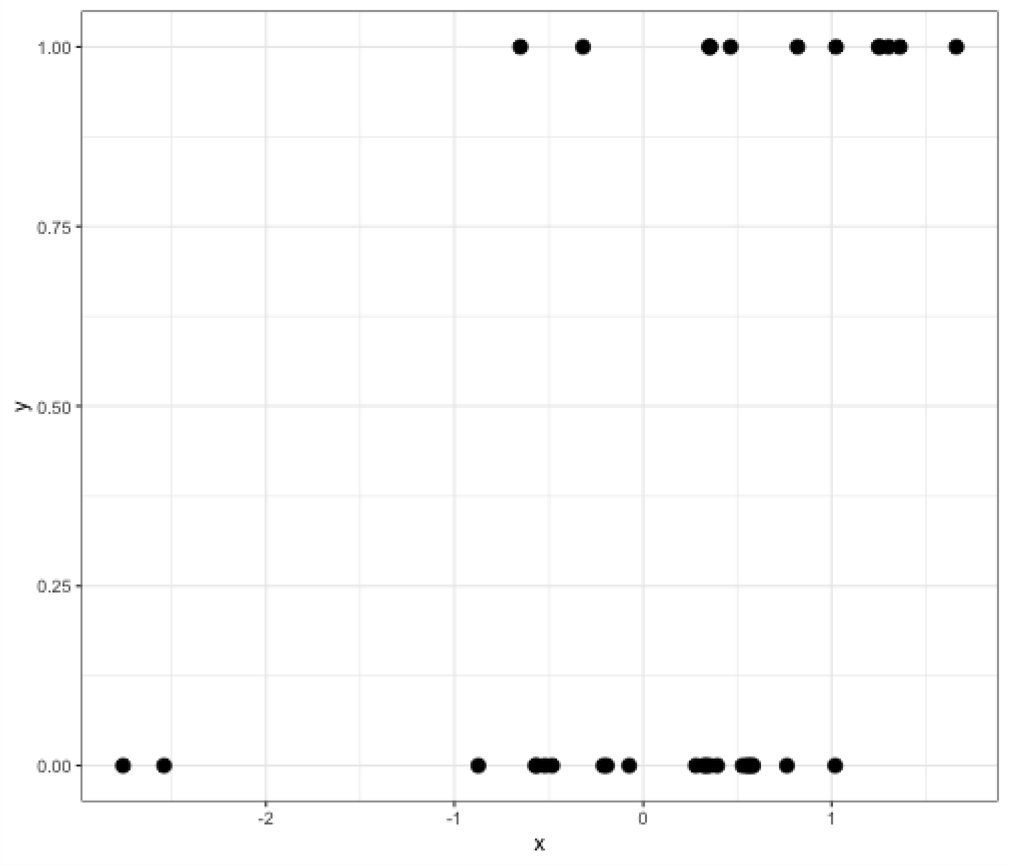
```
Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% B)))
Y
```

```
##  [1] 1 0 0 0 0 1 1 1 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0
```

```
Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% B)))
Y
```

```
##  [1] 0 0 0 1 1 0 0 1 0 0 1 0 0 1 1 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0
```

Visualizing simulated data



Compute log-likelihood

```
l_bern(B, X, Y)
```

```
## [1] -19.14079
```

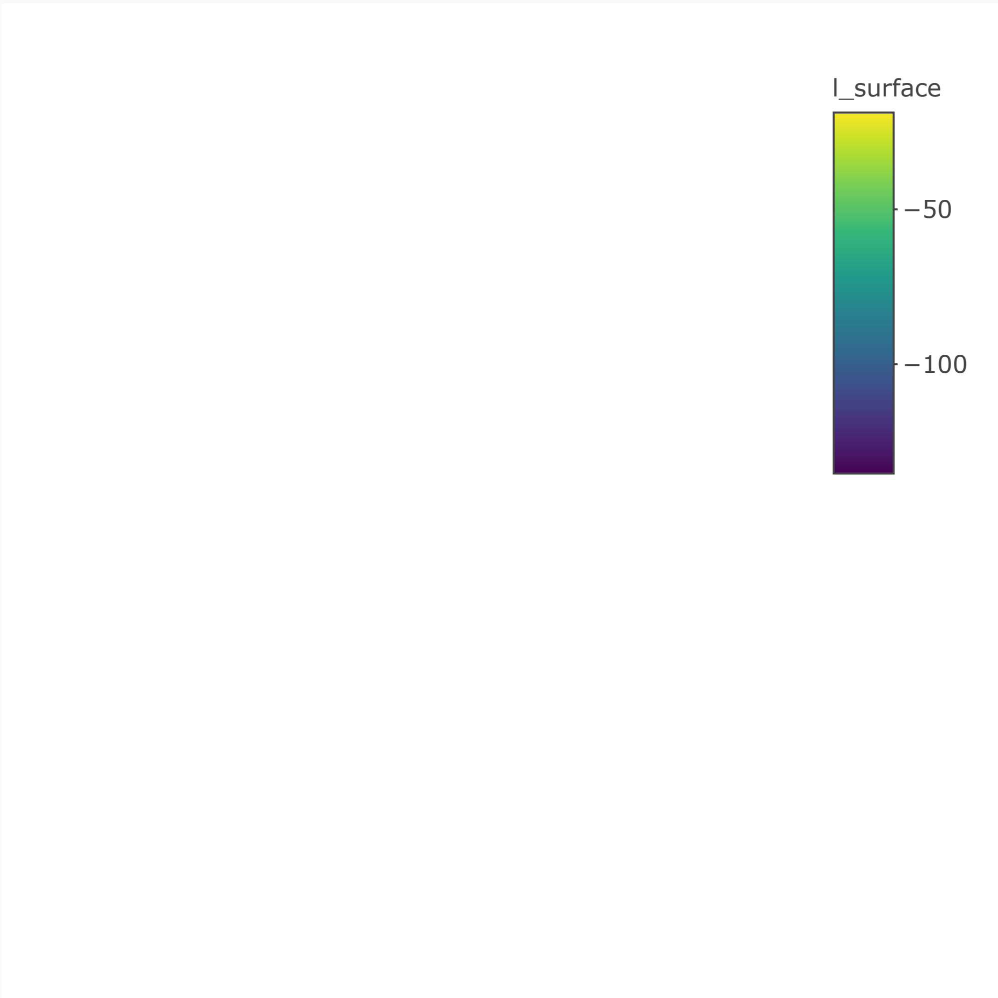
```
l_bern(c(0, 0), X, Y)
```

```
## [1] -24.26015
```

For a whole range of values...

```
B0 <- seq(-7, 5, length.out = 300)
B1 <- seq(-4, 8, length.out = 300)
l_surface <- matrix(0, nrow = length(B0), ncol = length(B1))
for(i in 1:nrow(l_surface)) {
  for(j in 1:ncol(l_surface)) {
    l_surface[i, j] <- l_bern(B = c(B0[i], B1[j]), X, Y)
  }
}
```

```r
library(plotly)
plot_ly(z = ~l_surface) %>%
  add_surface(x = B0, y = B1)
```



Numerical Optimization

You could try your luck with all-purpose `optim()`...

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

```
## $par
## [1] 30.994838  3.453131
##
## $value
## [1] -673.1047
##
## $counts
## function gradient
##      203       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Numerical Optimization

Or look for a dedicated optimizer.

```r
library(maxLik)
maxLik(l_bern, start = c(0, 0), X = X, Y = Y)
```

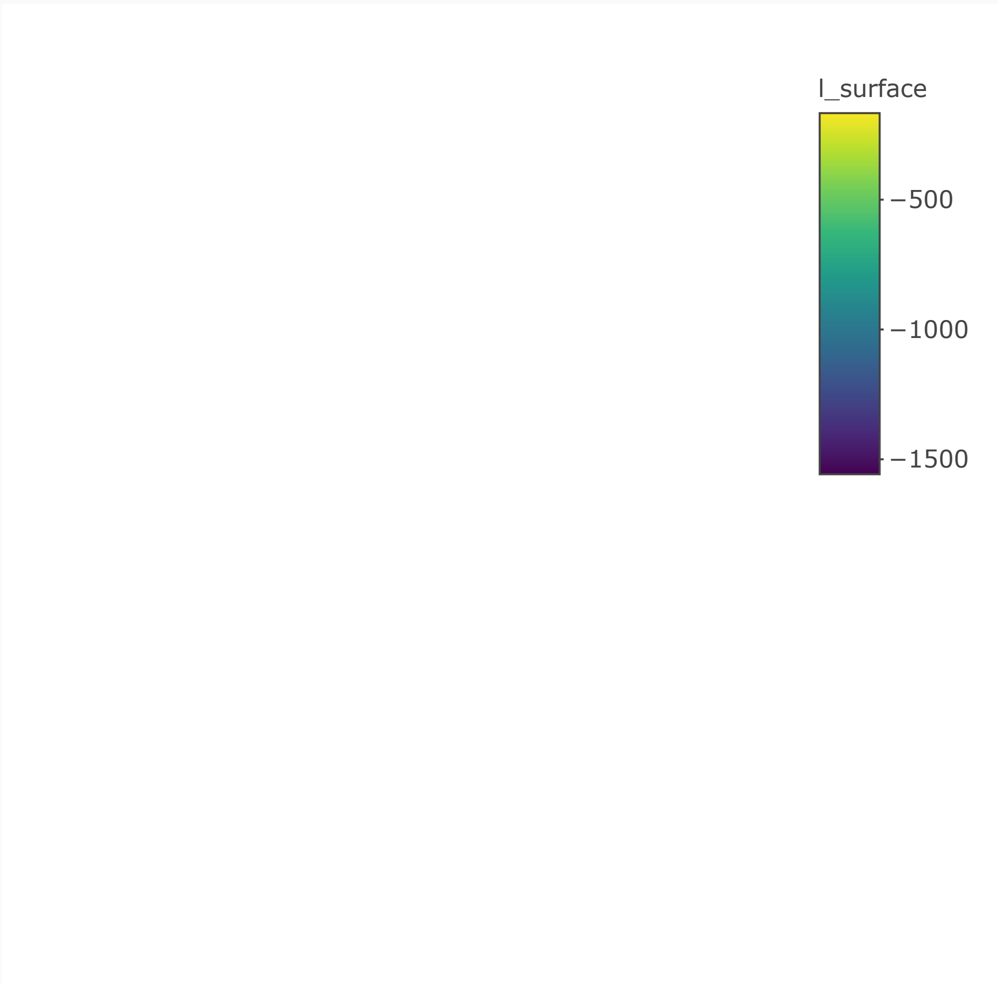
```
## Maximum Likelihood estimation
## Newton-Raphson maximisation, 5 iterations
## Return code 1: gradient close to zero
## Log-Likelihood: -18.74537 (2 free parameter(s))
## Estimate(s): -1.151947 1.669761
```

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

```
## (Intercept)          x1
##   -1.151947    1.669761
```

Sample size and likelihood

What happens when we draw $n = 350$ instead of $n = 35$?



Inferece: MLE as a RV

Any $\hat{\theta}_{MLE}$ is a function of (random) data, therefore it's a random variable with a distribution. If $\hat{\theta}_{MLE}$ is a *vector* then the random vector has a *joint* distribution.

```
n <- 35
X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2
Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% B)))
ml <- maxLik(l_bern, start = c(0, 0), X = X, Y = Y)
ml$estimate
```

```
## [1] -0.9274079  1.7005533
```

```
Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% B)))
ml <- maxLik(l_bern, start = c(0, 0), X = X, Y = Y)
ml$estimate
```

```
## [1] -1.425193  4.523493
```

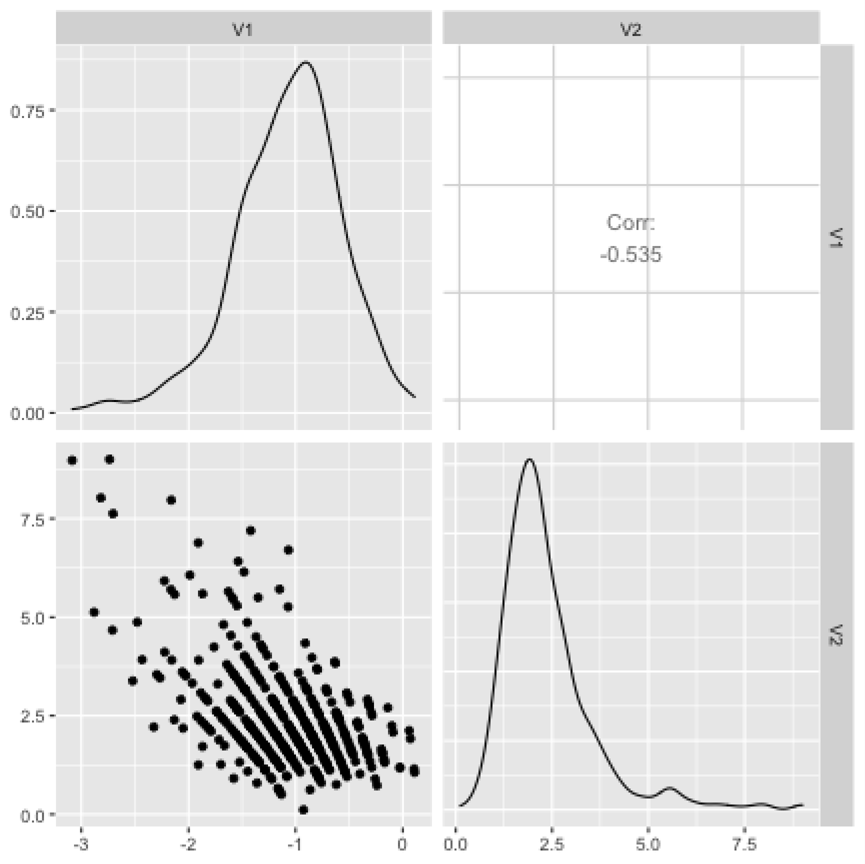
Simulation

We can fully simulate the joint distribution of $\hat{\theta}_{MLE}$.

```
it <- 500
MLE <- matrix(rep(NA, it * (p + 1)), ncol = p + 1)
for (i in 1:it) {
  Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% B)))
  ml <- maxLik(l_bern, start = c(0, 0), X = X, Y = Y)
  MLE[i, ] <- ml$estimate
}
MLE_35 <- as.data.frame(MLE)
sapply(MLE_35, mean)
```

```
##         V1        V2
## -1.072705  2.373614
```

```
ggpairs(MLE_35)
```



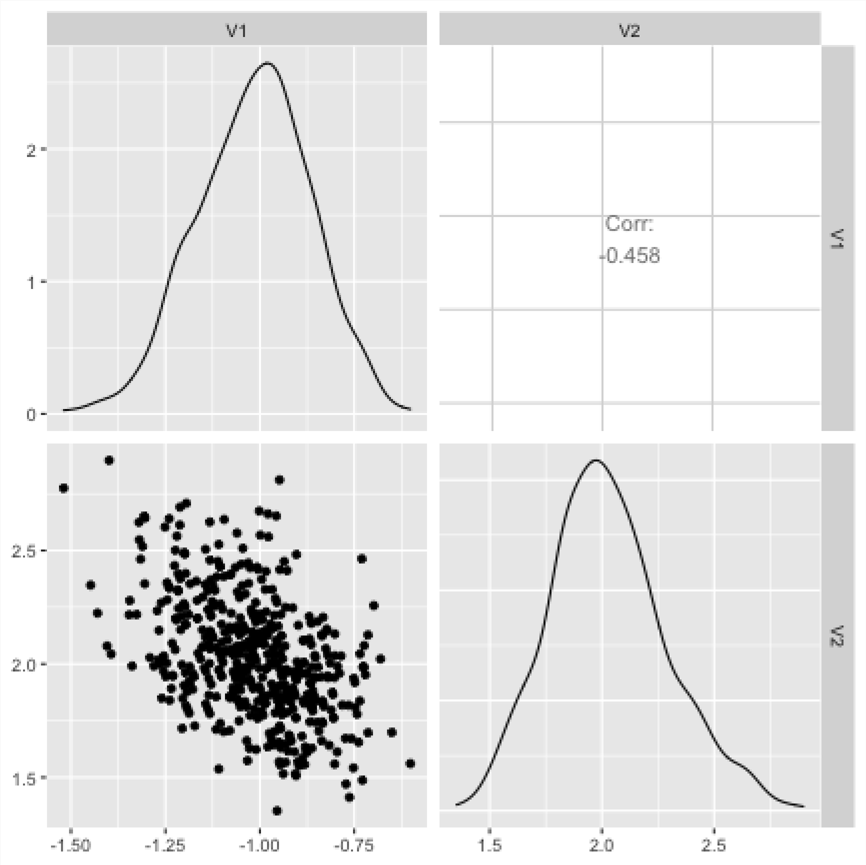
Looks a bit skewed and a bit biased. How does this change with sample size?

Larger sample simulation

```
n <- 350
X <- cbind(1, rmvnorm(n, mean = rep(0, p), sigma = diag(p)/2
MLE <- matrix(rep(NA, it * (p + 1)), ncol = p + 1)
for (i in 1:it) {
  Y <- rbinom(n, size = 1, prob = 1/(1 + exp(- X %*% B)))
  ml <- maxLik(l_bern, start = c(0, 0), X = X, Y = Y)
  MLE[i, ] <- ml$estimate
}
MLE_350 <- as.data.frame(MLE)
sapply(MLE_350, mean)
```

```
##        V1        V2
## -1.015276  2.029536
```

```
ggpairs(MLE_350)
```



Bias seems to be going away, variance is shrinking (consistent?) and it's starting to look MVN...

Bias in Logistic MLEs

Goal: empirically demonstrate the bias of the MLE as a function of sample size.

Investigating the MLE

We're thinking of

$$\hat{\theta}_{MLE}$$

as a generic MLE estimator of a parameter θ.

What can we say about its distribution as $n \to \infty$?

Expected Value

For an iid sample of size n, the log-likelihood is

$$l(\theta) = \sum_{i=1}^{n} \log(f(x_i; \theta))$$

Claim: If f is sufficiently smooth, as $n \to \infty$, $E(\hat{\theta}_{MLE}) = \theta_0$.

Proof

Consider maximizing

$$\frac{1}{n}l(\theta) = \frac{1}{n}\sum_{i=1}^{n}\log(f(x_i;\theta)).$$

By the LLN, as $n \to \infty$

$$\frac{1}{n}l(\theta) \to E(\log(f(x_i;\theta))).$$

By applying the law of the unconscious statistician, this can be re-expressed as $\int \log(f(x;\theta))f(x|\theta_0)\mathrm{d}x$, where θ_0 is the true parameter value.

Consider the derivative of this integral w.r.t. θ.

$$\frac{\partial}{\partial\theta}\int \log(f(x;\theta))f(x|\theta_0)\mathrm{d}x = \int \frac{\frac{\partial}{\partial\theta}f(x;\theta)}{f(x|\theta)}f(x|\theta_0)\mathrm{d}x$$