BIOS 7345: Advanced Regression for Independent Data

Andrew J. Spieker, Ph.D.

Associate Professor of Biostatistics
Vanderbilt University

Set 11: Generalized linear models (via likelihood)

Version: 10/29/2025

Table of Contents

Motivating example

Notes:

- We're about to pivot to generalized linear models (GLMs).
- You may already know:
  - Linear, logistic, and Poisson regression are all examples of GLMs.
  - Coefficients for most GLMs (including logistic and Poisson) cannot be expressed in closed form.
  - GLMs can be (but need not be) motivated in a likelihood framework.
- You may also already be aware of the `optim()` function in R for numerical optimization. The first order of business is to give you a couple of reasons why we will *not* be using the `optim()` function in our study of GLMs.
- I want you to believe fairly early on in the notes that studying GLMs in detail is a good use of your time.
  - Knowing what's going on under the hood, coding/algorithms, methodological skill building.

Logistic regression: Generating data

```
## Define expit function
expit <- function(x)
{
  expitx <- exp(x)/(1 + exp(x))
  return(expitx)
}

## Set seed for reproducibility
set.seed(7345)

## Generate data
n <- 50
X <- rbinom(n, 1, 0.5)
Y <- rbinom(n, 1, expit(-0.5 + X))

## Display data from single replicate
> table(X,Y)
   Y
X    0  1
  0 11  9
  1 13 17
```

Logistic regression: Hard-coding the likelihood

```
## Negative log-likelihood for logistic regression (hard-coded)
negll <- function(theta, x, y)
{
  beta0 <- theta[1]
  beta1 <- theta[2]
  lik <- dbinom(y,
                size = 1,
                prob = expit(beta0 + beta1 * x))
  negloglik <- -1 * sum(log(lik))
  return(negloglik)
}
```

Logistic regression: Numerical optimization

```
## Nelder-Mead (initialized at (log(26/24), 0))
zz0 <- optim(c(log(26/24), 0), negll, x = X, y = Y, hessian = TRUE)

## Report estimated coefficients
> zz0$par
[1] -0.2003658  0.4684407

## Report estimated standard errors
> sqrt(diag(solve(zz0$hessian)))
[1] 0.4494598 0.5811689
```

Logistic regression: Using the `glm()` function

```
## GLM function in R
zz1 <- glm(Y ~ X, family = binomial(link = "logit"))

## Report results (fairly close; not exact)
> summary(zz1)

Call:
glm(formula = Y ~ X, family = binomial(link = "logit"))

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.2007     0.4495  -0.446    0.655
X             0.4689     0.5812   0.807    0.420

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 69.235  on 49  degrees of freedom
Residual deviance: 68.579  on 48  degrees of freedom
AIC: 72.579

Number of Fisher Scoring iterations: 3

## You may not yet know what "Fisher Scoring" means
### ...but you will after this set of notes!
```

Logistic regression: Testing out speed

```r
## Estimate coefficients by numerical approximation
do.optim <- function(x,y)
{
  init <- c(log(sum(y)/(length(y) - sum(y))), 0)
  zz <- optim(init, negll, x = x, y = y, hessian = TRUE)
  return(list(COEF = as.numeric(zz$par),
              SE = sqrt(diag(solve(zz$hessian)))))
}

## Estimate coefficients by iterative approach (to discuss)
do.glm <- function(x,y)
{
  init <- c(log(sum(y)/(length(y) - sum(y))), 0)
  zz <- glm(Y ~ X, family = binomial(link = "logit"), start = init)
  return(list(COEF = coef(zz),
              SE = summary(zz)$coef[,2]))
}
```

Logistic regression: Testing out speed ($n = 50$)

```
## Number of simulations
nsim <- 1000

## Sample size
n <- 50

## How long does it take to do optim()?
> system.time(for (j in 1:nsim) {
    X <- rbinom(n, 1, 0.5)
    Y <- rbinom(n, 1, expit(-0.5 + X))
    do.optim(x = X, y = Y)
  })

>
   user  system elapsed
  0.306   0.025   0.336

## How long does it take to do glm()?
> system.time(for (j in 1:nsim) {
    X <- rbinom(n, 1, 0.5)
    Y <- rbinom(n, 1, expit(-0.5 + X))
    do.glm(x = X, y = Y)
  })

   user  system elapsed
  0.445   0.019   0.470
```

Logistic regression: Testing out speed ($n = 300$)

```
## Number of simulations
nsim <- 1000

## Sample size
n <- 300

## How long does it take to do optim()?
> system.time(for (j in 1:nsim) {
    X <- rbinom(n, 1, 0.5)
    Y <- rbinom(n, 1, expit(-0.5 + X))
    do.optim(x = X, y = Y)
  })

>
   user  system elapsed
  1.207   0.028   1.253

## How long does it take to do glm()?
> system.time(for (j in 1:nsim) {
    X <- rbinom(n, 1, 0.5)
    Y <- rbinom(n, 1, expit(-0.5 + X))
    do.glm(x = X, y = Y)
  })

   user  system elapsed
  0.596   0.010   0.611
```

Logistic regression: Testing out speed ($n = 1000$)

```
## Number of simulations
nsim <- 1000

## Sample size
n <- 1000

## How long does it take to do optim()?
> system.time(for (j in 1:nsim) {
    X <- rbinom(n, 1, 0.5)
    Y <- rbinom(n, 1, expit(-0.5 + X))
    do.optim(x = X, y = Y)
  })

>
   user  system elapsed
  3.518   0.144   3.663

## How long does it take to do glm()?
> system.time(for (j in 1:nsim) {
    X <- rbinom(n, 1, 0.5)
    Y <- rbinom(n, 1, expit(-0.5 + X))
    do.glm(x = X, y = Y)
  })

   user  system elapsed
  1.168   0.021   1.189
```

Logistic regression: Testing out speed ($n = 10{,}000$)

```
## Number of simulations
nsim <- 1000

## Sample size
n <- 10000

## How long does it take to do optim()?
> system.time(for (j in 1:nsim) {
    X <- rbinom(n, 1, 0.5)
    Y <- rbinom(n, 1, expit(-0.5 + X))
    do.optim(x = X, y = Y)
  })

>
    user  system  elapsed
  40.459   1.700  42.159

## How long does it take to do glm()?
> system.time(for (j in 1:nsim) {
    X <- rbinom(n, 1, 0.5)
    Y <- rbinom(n, 1, expit(-0.5 + X))
    do.glm(x = X, y = Y, init = c(log(sum(Y)/(n - sum(Y))), 0))
  })

   user  system  elapsed
   9.800   0.368  10.166
```

Logistic regression: Comments

- Nelder-Mead starts to wheeze a bit with large samples.
  - Unless you calculate and supply the gradient, the algorithm will approximate it at every step, which is computationally taxing.
  - May not be a "big deal" when running a single model.
  - Likely a big deal when you're trying to implement bootstrap/imputation methods within a simulation study.
- Even if you supply the gradient, calculation of the inverse Hessian is approximate. The quality of the approximation depends in part on how well behaved the likelihood surface is.
  - Although you may not know this yet, the likelihood surface for logistic regression is quite well behaved for reasons we will discuss.

Logistic regression: Comments

- The `glm()` function utilizes an algorithm that numerically solves the exact estimating equations (though there is no closed-form expression).
- Keep in mind that our illustration is for the simple logistic regression setting; you could imagine how changing the model to be more complicated might compound these issues.
  - Informal homework: try it!
- For what it's worth, you can improve speed even further by using `glm.fit()`, but *for now*, we're going to learn all the unifying theory that underlies GLMs, including how to hard-code one from scratch.

Table of Contents

Recall: Exponential families

- As in the previous set of notes, assume Y has a density function that can be written in the following very nice form:

$$f_Y(y;\theta,\phi) \quad = \quad \exp\left[\frac{y\theta - b(\theta)}{\phi} + c(y,\phi)\right].$$

- θ the "natural parameter" or "canonical parameter."
- ϕ is a nuisance parameter, serving the role of a "scale" parameter.
- $\mathrm{E}[Y] = b'(\theta)$.
- $\mathrm{Var}[Y] = \phi b''(\theta)$.
- Some mean variance relationship, $V(\mu)$, is implied.

Pushing things forward: Regression

- Suppose we observe outcomes $Y_1, \ldots, Y_N$, with associated covariate vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ (which you can assume for now to be fixed for ease of presentation, though we will eventually deal with the random case just as we did for OLS), with density taking the form:

$$f_Y(y|\mathbf{x};\theta,\phi) = \exp\left[\frac{y\theta(\mathbf{x}^\top\boldsymbol{\beta}) - b(\theta(\mathbf{x}^\top\boldsymbol{\beta}))}{\phi} + c(y,\phi)\right],$$

  which is to say that $\theta = \theta(\mathbf{x}^\top\boldsymbol{\beta})$ is a function of $\mathbf{x}^\top\boldsymbol{\beta}$.
- A GLM specifies two components:
  1. The mean model: $g(\mathrm{E}[Y]) = \mathbf{x}^\top\boldsymbol{\beta}$.
  2. The family of distributions to which Y belongs (must be of the nice exponential family form shown above).
- Ultimately (in the next set of notes), we will argue that a GLM is uniquely determined by *less* information; but for now, we are imagining that the likelihood to be completely correct.

Now: Regression

- The log-likelihood based on observations $Y_1, \ldots, Y_N$ is given by:

$$\ell(\boldsymbol{\beta}, \phi; \mathbf{y}, \mathbf{X}) = \sum_{i=1}^{N} \left[\frac{y_i \theta(\mathbf{x}_i^\top \boldsymbol{\beta}) - b(\theta(\mathbf{x}_i^\top \boldsymbol{\beta}))}{\phi} + c(y_i, \phi) \right]$$

- Our first order of business is going to be to find score equations. We'll do this first for $\boldsymbol{\beta}$ (though we'll eventually have to deal with ϕ).

Notation and terminology:

- Linear predictor: $\eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$, or $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$.
- Natural/canonical parameter: $\boldsymbol{\theta} = \boldsymbol{\theta}(\boldsymbol{\eta}) = (\theta(\mathbf{x}_1^\top \boldsymbol{\beta}), \cdots, \theta(\mathbf{x}_N^\top \boldsymbol{\beta}))^\top$.
- Mean: $\mathrm{E}[Y_i] = \mu_i$, or $\mathrm{E}[\mathbf{y}] = \boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{\beta})$.
  - Recall that $\mathrm{E}[Y_i] = \mu_i = b'(\theta_i)$, or $\mathrm{E}[\mathbf{y}] = \boldsymbol{\mu} = \mathrm{vec}(b'(\theta_i))$.
  - I may sometimes use the notation $b'(\boldsymbol{\theta})$ for simplicity, especially on the board.
- Link function: $g(\cdot)$.
  - $g(\mathrm{E}[Y]) = g(\mu) = g(b'(\theta))$.

Score equations:

- GLM mean model: $g(\mathrm{E}[Y]) = g(\mu) = g(b'(\theta)) = \mathbf{x}^\mathsf{T}\boldsymbol{\beta} =: \eta$.
- Note: we've already done some of the heavy lifting by developing score equations for exponential families with respect to a single real-valued parameter, θ.
- We can develop the score equations for $\boldsymbol{\beta}$ by the chain rule:

$$\left[\frac{\partial \ell(\boldsymbol{\beta}, \phi)}{\partial \boldsymbol{\beta}}\right]_{K\times 1} = \left[\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\beta}}\right]_{K\times N} \left[\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\eta}}\right]_{N\times N} \left[\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\mu}}\right]_{N\times N} \left[\frac{\partial \ell}{\partial \boldsymbol{\theta}}\right]_{N\times 1}$$

- Let's take each of these three components one step at a time.

Score equations: The "left two" components

- The derivative of $\boldsymbol{\mu}$ with respect to $\boldsymbol{\beta}$ is a $K \times N$ matrix:

$$\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\beta}} = \frac{\partial}{\partial \boldsymbol{\beta}} \text{vec}(g^{-1}(\mathbf{x}_i^\top \boldsymbol{\beta})) = \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\beta}} \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\eta}}$$

- We typically denote this as $\mathbf{D}^\top = \mathbf{D}^\top(\boldsymbol{\beta})$.
  - Referring to the $K \times N$ derivative matrix as $\mathbf{D}^\top$ instead of simply "$\mathbf{D}$" is standard, and feels natural largely because we're already used to expressions involving $\mathbf{X}^\top$.
- Note that $\partial \boldsymbol{\eta} / \partial \boldsymbol{\beta} = \partial(\mathbf{X}\boldsymbol{\beta}) / \partial \boldsymbol{\beta} = \mathbf{X}^\top$.
- Further note that $\partial \boldsymbol{\mu} / \partial \boldsymbol{\eta} = \text{diag}(\partial \mu_i(\boldsymbol{\beta}) / \partial \eta_i(\boldsymbol{\beta}))$.
- I will use $\mathbf{D}^\top$ notation when it is called for, but ultimately what we've learned is that:

$$\mathbf{D}^\top(\boldsymbol{\beta}) = \mathbf{X}^\top \text{diag}\left(\frac{\partial \mu_i(\boldsymbol{\beta})}{\partial \eta_i(\boldsymbol{\beta})} \right)$$

Score equations: The "middle" component

- The derivative of $\boldsymbol{\theta}$ with respect to $\boldsymbol{\mu}$ is—in this course (independent observations)—an $N \times N$ (diagonal) matrix:

$$\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\mu}} = ???$$

- Already know: $\mu_i = b'(\theta_i)$. Therefore,

$$\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\theta}} = \text{diag}(b''(\theta_i)) = \text{diag}(V(\mu_i)) =: \mathbf{V} = \mathbf{V}(\boldsymbol{\beta}).$$

- Recall that $\text{Var}[Y_i] = \phi b''(\theta_i) \propto b''(\theta_i)$; the notation "$\mathbf{V}$" signifies the nature of the mean-variance relationship; in this set of slides $V(\mu_i)$ will be used as shorthand for $b''(\theta_i)$. Therefore,

$$\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\mu}} = \mathbf{V}^{-1} = \mathbf{V}^{-1}(\boldsymbol{\beta}) = \text{diag}(1/V(\mu_i)).$$

Score equations: The "right" component

- The derivative of ℓ with respect to $\boldsymbol{\theta}$ is an $N \times 1$ vector:

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}}.$$

- We already did the heavy lifting on deriving the score function in this nice exponential family (with respect to θ):

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}} = \frac{\mathbf{y} - \text{vec}(b'(\theta_i))}{\phi} = \frac{\mathbf{y} - \boldsymbol{\mu}(\boldsymbol{\beta})}{\phi} = \frac{\mathbf{y} - \text{vec}(g^{-1}(\mathbf{x}_i^\top \boldsymbol{\beta}))}{\phi}.$$

Score equations:

- Putting this all together,

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}} = \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\beta}} \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\eta}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\mu}} \frac{\partial \ell}{\partial \boldsymbol{\theta}} = \mathbf{D}^\intercal(\boldsymbol{\beta})\mathbf{V}^{-1}(\boldsymbol{\beta})(\mathbf{y} - \boldsymbol{\mu}(\boldsymbol{\beta}))/\phi.$$

- To estimate $\boldsymbol{\beta}$, we solve the following equations for $\boldsymbol{\beta}$:

$$\mathbf{D}^\intercal\mathbf{V}^{-1}(\mathbf{y} - \boldsymbol{\mu})/\phi = \mathbf{0}.$$

- The solution, $\widehat{\boldsymbol{\beta}}$, to this problem, does not depend upon ϕ, and so we can instead solve:

$$\mathbf{D}^\intercal\mathbf{V}^{-1}(\mathbf{y} - \boldsymbol{\mu}) = \mathbf{0}.$$

- This can be written with greater detail as:

$$\mathbf{X}^\intercal \text{diag}((\partial \mu_i(\boldsymbol{\beta})/\partial \eta_i(\boldsymbol{\beta}))/V(\mu_i(\boldsymbol{\beta})))(\mathbf{y} - \text{vec}(g^{-1}(\mathbf{x}_i^\intercal \boldsymbol{\beta}))) = \mathbf{0}.$$

Example 11.1: Normal distribution (identity link)

- Suppose our GLM is based on:
  - $Y \sim \mathcal{N}(\mu, \sigma^2)$.
  - $g(\cdot)$ given by the identity link (i.e., $g(\mu) = \mu$).
- Then,
  1. Since $g(\mu) = \mu = \mathbf{x}^\top \boldsymbol{\beta}$, we have $\mathbf{D}^\top = \partial \boldsymbol{\mu} / \partial \boldsymbol{\beta} = \mathbf{X}^\top$.
  2. $\mathbf{V} = \text{diag}(1) = \mathbf{I}$ (constant variance).
- The likelihood-based equations below can be used to solve for $\boldsymbol{\beta}$:

$$\mathbf{D}^\top \mathbf{V}^{-1}(\mathbf{y} - \boldsymbol{\mu}) = \mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = \mathbf{0}.$$

- We recognize these equations are the normal equations for OLS (we previously motivated the same equations without a likelihood).
- These equations have a closed-form solution (don't get used to it).

Example 11.2: Normal distribution (log link)

- Suppose our GLM is based on:
  - $Y \sim \mathcal{N}(\mu, \sigma^2)$.
  - $g(\cdot)$ given by the log link (i.e., $g(\mu) = \log(\mu)$).
- Then,
  1. To determine $\mathbf{D}^\intercal$, note that $\boldsymbol{\mu} = \text{vec}(\exp(\mathbf{x}_i^\intercal \boldsymbol{\beta}))$. Therefore,

$$\mathbf{D}^\intercal = \left[\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\beta}}\right]_{K \times N} = \left[\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\beta}}\right]_{K \times N} \left[\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\eta}}\right]_{N \times N} = \mathbf{X}^\intercal \text{diag}(\exp(\mathbf{x}_i^\intercal \boldsymbol{\beta})).$$

  2. $\mathbf{V} = \mathbf{I}$.
- The likelihood-based equations below can be used to solve for $\boldsymbol{\beta}$:

$$\mathbf{D}^\intercal \mathbf{V}^{-1}(\mathbf{y} - \boldsymbol{\mu}) = \mathbf{X}^\intercal \text{diag}(\exp(\mathbf{x}_i^\intercal \boldsymbol{\beta}))(\mathbf{y} - \text{vec}(\exp(\mathbf{x}_i^\intercal \boldsymbol{\beta}))) = \mathbf{0}.$$

- As another note, I may sometimes write in the style "$\boldsymbol{\mu} = \exp(\mathbf{X}\boldsymbol{\beta})$," which you should take to mean entries of a vector rather than, say, the matrix exponential.

Example 11.3: Bernoulli distribution (logit link)

- Suppose our GLM is based on:
  - $Y \sim \text{Bernoulli}(p)$.
  - $g(\cdot)$ given by the logit link (i.e., $g(\mu) = \text{logit}(\mu)$).
- Then,
  1. To determine $\mathbf{D}^\top$, note that $\boldsymbol{\mu} = \text{vec}(\text{expit}(\mathbf{x}_i^\top \boldsymbol{\beta}))$. Therefore,

$$\begin{aligned} \mathbf{D}^\top &= \left[\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\beta}}\right]_{K\times N} = \left[\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\beta}}\right]_{K\times N}\left[\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\eta}}\right]_{N\times N} \\ &= \mathbf{X}^\top \text{diag}(\text{expit}(\mathbf{x}_i^\top \boldsymbol{\beta})(1-\text{expit}(\mathbf{x}_i^\top \boldsymbol{\beta}))). \end{aligned}$$

  2. $\mathbf{V} = \text{diag}(\mu_i(1-\mu_i)) = \text{diag}(\text{expit}(\mathbf{x}_i^\top \boldsymbol{\beta})(1-\text{expit}(\mathbf{x}_i^\top \boldsymbol{\beta})))$.
- The likelihood-based equations below can be used to solve for $\boldsymbol{\beta}$:

$$\mathbf{D}^\top \mathbf{V}^{-1}(\mathbf{y}-\boldsymbol{\mu}) = \mathbf{X}^\top(\mathbf{y} - \text{vec}(\text{expit}(\mathbf{x}_i^\top \boldsymbol{\beta}))) = \mathbf{0}.$$

- Notice the nice cancellation in this case (no accident)!

Example 11.4: Bernoulli distribution (log link)

- Suppose our GLM is based on:
  - $Y \sim \text{Bernoulli}(p)$.
  - $g(\cdot)$ given by the log link (i.e., $g(\mu) = \log(\mu)$).
- Then,
  1. Now, note that $\boldsymbol{\mu} = \text{vec}(\exp(\mathbf{x}_i^\top \boldsymbol{\beta}))$, so $\mathbf{D}^\top = \mathbf{X}^\top \text{diag}(\exp(\mathbf{x}_i^\top \boldsymbol{\beta}))$.
  2. $\mathbf{V} = \text{diag}(\mu_i(1-\mu_i)) = \text{diag}(\exp(\mathbf{x}_i^\top \boldsymbol{\beta})(1-\exp(\mathbf{x}_i^\top \boldsymbol{\beta})))$.
- The likelihood-based equations below can be used to solve for $\boldsymbol{\beta}$:

$$\mathbf{D}^\top \mathbf{V}^{-1}(\mathbf{y}-\boldsymbol{\mu}) = \mathbf{X}^\top \text{diag}\left(\frac{1}{1-\exp(\mathbf{x}_i^\top \boldsymbol{\beta})}\right)(\mathbf{y} - \text{vec}(\exp(\mathbf{x}_i^\top \boldsymbol{\beta}))) = \mathbf{0}.$$

Table of Contents

The canonical link function

Choosing a link function:

- GLM mean model:

$$g(\mathrm{E}[Y_i]) = g(\mu_i) = g(b'(\theta_i)) = \mathbf{x}_i^\top \boldsymbol{\beta} =: \eta_i$$

- What seems to be a really "nice" choice of a link function?
- The choice $g^{-1}(\cdot) = b'(\cdot)$ seems to be nice in the sense that under this choice, the canonical parameter is linear in $\boldsymbol{\beta}$. Specifically, $\theta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$.
- In fact, this choice of $g(\cdot)$ produces some very desirable mathematical properties, and is termed the canonical link function.

The canonical link function

The canonical link: $g^{-1}(\cdot) = b'(\cdot)$

- GLM mean model under the canonical link function:

$$g(\mathrm{E}[Y_i]) = g(\mu_i) = g(b'(\theta_i)) = \theta_i = \mathbf{x}_i^\mathsf{T}\boldsymbol{\beta} =: \eta_i$$

- Recall the likelihood-based equations for $\boldsymbol{\beta}$:

$$\left[\frac{\partial \ell}{\partial \boldsymbol{\beta}}\right]_{K\times 1} = \left[\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\beta}}\right]_{K\times N}\left[\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\mu}}\right]_{N\times N}\left[\frac{\partial \ell}{\partial \boldsymbol{\theta}}\right]_{N\times 1}$$

- Latter two terms unaffected by choice of link function beyond the fact that $\mu = g^{-1}(\mathbf{x}^\mathsf{T}\boldsymbol{\beta})$. However, if $g^{-1}(\cdot) = b'(\cdot)$:

$$\mathbf{D}^\mathsf{T} := \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\beta}} = \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\beta}}\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\eta}} = \mathbf{X}^\mathsf{T}\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\eta}} = \mathbf{X}^\mathsf{T}\frac{\partial}{\partial \boldsymbol{\theta}}\mathrm{vec}(b'(\theta_i)) = \mathbf{X}^\mathsf{T}\mathbf{V}.$$

The canonical link function

The canonical link: $g^{-1}(\cdot) = b'(\cdot)$

- GLM mean model under the canonical link function:

$$g(\mathrm{E}[Y_i]) = g(\mu_i) = g(b'(\theta_i)) = \theta_i = \mathbf{x}_i^\top \boldsymbol{\beta} =: \eta_i$$

- Under the canonical link function, the likelihood-based equations take the following form:

$$\begin{aligned} \mathbf{D}^\top \mathbf{V}^{-1}(\mathbf{y} - \boldsymbol{\mu}) &= (\mathbf{X}^\top \mathbf{V})\mathbf{V}^{-1}(\mathbf{y} - \boldsymbol{\mu}) \\ &= \mathbf{X}^\top(\mathbf{y} - \boldsymbol{\mu}) = \mathbf{X}^\top(\mathbf{y} - \mathrm{vec}(g^{-1}(\mathbf{x}_i^\top \boldsymbol{\beta}))) = \mathbf{0}. \end{aligned}$$

- That is, $\mathbf{D}^\top \mathbf{V}^{-1} = \mathbf{X}^\top$ under the canonical link.
- Note that while $\mathbf{D}^\top$ and $\mathbf{V}^{-1}$ may depend upon $\boldsymbol{\beta}$, $\mathbf{X}^\top$ does not.

The canonical link function

The canonical link: $g^{-1}(\cdot) = b'(\cdot)$

- We saw this nice reduction in two of our prior examples!
- For a normal distribution, the identity link produced a score equation of this form. We can see this from factoring this exponential family:
  - $\theta = \mu$ is the natural parameter.
  - $b(\theta) = \theta^2/2$.
  - $b'(\theta) = \theta$.
  - $g(\mu) = \mu$ is the canonical link.
- For a Bernoulli distribution, the logit link produced a score equation of this form. We can see this from factoring this exponential family:
  - $\theta = \text{logit}(p)$ is the natural parameter.
  - $b(\theta) = \log(1 + \exp(\theta))$.
  - $b'(\theta) = \text{expit}(\theta)$.
  - $g(\mu) = \text{logit}(p)$ is the canonical link.
- Notice: the canonical link is determined by how θ relates to the target parameter, or how the inverse of $b'(\cdot)$ relates to θ.

Table of Contents

The Gauss-Newton approach

- Let's return to the general case (i.e., not necessarily the canonical link function). The following equations arise from the likelihood:

$$\mathbf{D}^\intercal \mathbf{V}^{-1}(\mathbf{y} - \boldsymbol{\mu}) = \mathbf{0}.$$

- Again, I could expand the known expression for $\mathbf{D}^\intercal$, but it is convenient to do that on the back end.
- These equations need not possess a closed-form solution. In many (most) cases, we must implement this fit numerically.

The Gauss-Newton approach: Definitions/notation

- Let $\mathbf{G}(\boldsymbol{\beta}; \mathbf{x}, Y) = \mathbf{x}\frac{\partial\mu/\partial\eta}{b''(\theta)}(Y - g^{-1}(\mathbf{x}^\top\boldsymbol{\beta}))$.
  - This is the estimating function (in this set of notes, they happen to come from the score function).
- Let $\mathbb{G}_N(\boldsymbol{\beta}; \mathbf{X}, \mathbf{y}) = \mathbf{D}^\top\mathbf{V}^{-1}(\mathbf{y} - \boldsymbol{\mu}) = \sum_{i=1}^N \mathbf{G}_i(\boldsymbol{\beta}; \mathbf{x}_i, Y_i)$.
  - The estimating equations are given by $\mathbb{G}_N(\boldsymbol{\beta}; \mathbf{X}, \mathbf{y}) = \mathbf{0}$, which—in this set of notes—happen to arise from score equations.

The Gauss-Newton approach: Definitions/notation

- Let $\mathbf{A}(\boldsymbol{\beta}) = \mathrm{E}\Big[-\frac{\partial}{\partial \boldsymbol{\theta}}\mathbf{G}(\boldsymbol{\theta}; \mathbf{x}, Y)\big|_{\boldsymbol{\theta}=\boldsymbol{\beta}}\Big]$.
  - Proportional to the expected Fisher information for one observation.
- Let $\mathbb{A}_N(\boldsymbol{\beta}) = \mathrm{E}\Big[-\frac{\partial}{\partial \boldsymbol{\theta}}\mathbb{G}_N(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y})\big|_{\boldsymbol{\theta}=\boldsymbol{\beta}}\Big]$.
  - Proportional to the expected Fisher information for N observations.

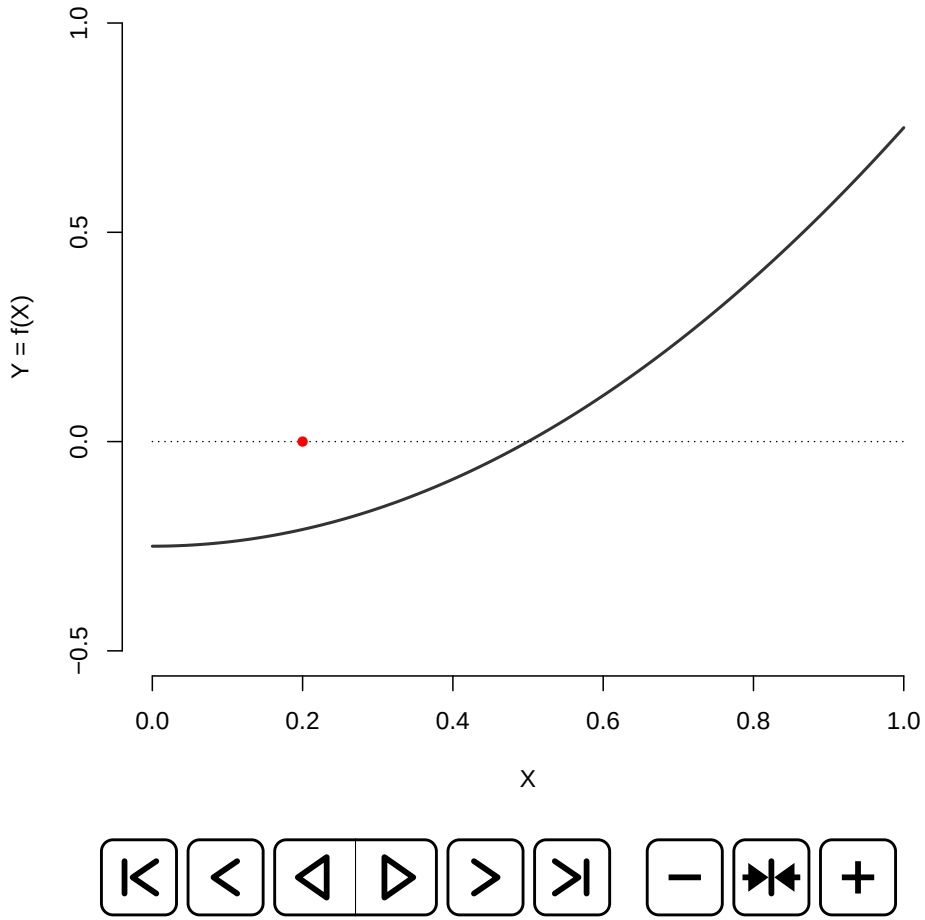
The Gauss-Newton approach: Steps

- Our approach will be similar to Newton's method—which you may recall from Calculus I, but I am including a reminder here.
- When seeking to solve $f(x) = 0$, we begin with an initial guess, x_0, and update as follows:

$$x_{j+1} = x_j - \frac{f(x_j)}{f'(x_j)}$$

- In the next slide, I illustrate a few iterations of Newton's method for a function of a single variable.

The Gauss-Newton approach: Steps



The Gauss-Newton approach: Steps

- We now implement a similar method, though in higher dimensions.
- Like the version you learned in single-variable calculus, we have to propose an initial value (can be an art). Call this $\boldsymbol{\beta}^{(0)}$.
  - Example: Calibrate intercept to the observed data mean (on the scale of the link function) and initialize the remaining coefficients to zero.
- The $(j+1)$-step is given by: $\boldsymbol{\beta}^{(j+1)} = \boldsymbol{\beta}^{(j)} + \left[\mathbb{A}_N(\boldsymbol{\beta}^{(j)})\right]^{-1} \mathbb{G}_N(\boldsymbol{\beta}^{(j)})$.
- Note that $\mathbb{A}_N(\boldsymbol{\beta})$ involves an *expectation* of the (negative) derivative, distinguishing it slightly from the version of Newton's method you're already familiar with. Because $\mathbb{A}_N(\boldsymbol{\beta})$ is proportional to the expected Fisher information, this is also called the Fisher scoring algorithm.
  - See? I promised you'd learn it!
- For the sake of giving the algorithm a bit more specificity, let's determine more explicit expressions for $\mathbb{G}_N(\boldsymbol{\beta})$ and $\mathbb{A}_N(\boldsymbol{\beta})$ for GLMs.

The Gauss-Newton approach: More explicit expressions

- We already know that $\mathbb{G}_N(\boldsymbol{\beta}) = \mathbf{D}^\mathsf{T}(\boldsymbol{\beta})\mathbf{V}^{-1}(\boldsymbol{\beta})(\mathbf{y} - \boldsymbol{\mu}(\boldsymbol{\beta}))$.
- Recall: $\mathbf{D}^\mathsf{T}(\boldsymbol{\beta}) = \mathbf{X}^\mathsf{T}\text{diag}(\partial\mu_i(\boldsymbol{\beta})/\partial\eta_i(\boldsymbol{\beta}))$.
- Recall: $\mathbf{V}(\boldsymbol{\beta}) = \text{diag}(V(\mu_i(\boldsymbol{\beta})))$.
- Therefore,

$$\begin{aligned}\mathbb{G}_N(\boldsymbol{\beta}) &= \mathbf{D}^\mathsf{T}(\boldsymbol{\beta})\mathbf{V}^{-1}(\boldsymbol{\beta})(\mathbf{y} - \boldsymbol{\mu}(\boldsymbol{\beta})) \\ &= \mathbf{X}^\mathsf{T}\text{diag}\left(\left(\frac{\partial\mu_i(\boldsymbol{\beta})}{\partial\eta_i(\boldsymbol{\beta})}\right)\left(\frac{1}{V(\mu_i(\boldsymbol{\beta}))}\right)\right)(\mathbf{y} - \text{vec}(g^{-1}(\mathbf{x}_i^\mathsf{T}\boldsymbol{\beta}))).\end{aligned}$$

- The estimating equations for a GLM look quite a bit like those from weighted least squares, with the weights depending upon the nature of the mean model and the mean-variance relationship!

The Gauss-Newton approach: More explicit expressions

- We can determine the derivative of $\mathbb{G}_N(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$:

$$\begin{aligned}
-\frac{\partial}{\partial \boldsymbol{\beta}}\mathbb{G}_N(\boldsymbol{\beta}) &= -\left[\frac{\partial}{\partial \boldsymbol{\beta}}(\mathbf{y}-\boldsymbol{\mu}(\boldsymbol{\beta}))\right]\text{diag}\left(\frac{\partial \mu_i(\boldsymbol{\beta})}{\partial \eta_i(\boldsymbol{\beta}))}\frac{1}{V(\mu_i(\boldsymbol{\beta}))}\right)\mathbf{X} \\
&\quad -\left[\frac{\partial}{\partial \boldsymbol{\beta}}\text{vec}\left(\frac{\partial \mu_i(\boldsymbol{\beta})}{\partial \eta_i(\boldsymbol{\beta})}\frac{1}{V(\mu_i(\boldsymbol{\beta}))}\right)\right]\text{diag}(y_i-\mu_i(\boldsymbol{\beta}))\mathbf{X} \\
&= \mathbf{X}^\top\mathbf{W}(\boldsymbol{\beta})\mathbf{X} - \left[\frac{\partial}{\partial \boldsymbol{\beta}}\text{vec}\left(\frac{\partial \mu_i(\boldsymbol{\beta})}{\partial \eta_i(\boldsymbol{\beta})}\frac{1}{V(\mu_i(\boldsymbol{\beta}))}\right)\right]\text{diag}(y_i-\mu_i(\boldsymbol{\beta}))\mathbf{X},
\end{aligned}$$

  where $\mathbf{W}(\boldsymbol{\beta}) = \text{diag}\left(\left(\frac{\partial \mu_i(\boldsymbol{\beta})}{\partial \eta_i(\boldsymbol{\beta})}\right)^2\frac{1}{V(\mu_i(\boldsymbol{\beta}))}\right)$.

- Now, $\mathbb{A}_N(\boldsymbol{\beta})$ is the expectation (with respect to $\mathbf{y}$):

$$\mathbb{A}_N(\boldsymbol{\beta}) = \mathbf{X}^\top\mathbf{W}(\boldsymbol{\beta})\mathbf{X}.$$

- Why does $\mathbb{A}_N(\boldsymbol{\beta})$ reduce down in this way?

The Gauss-Newton approach: More explicitly for GLMs

- Note: $\mathbb{A}_N(\boldsymbol{\beta}) = \mathbf{D}^\top(\boldsymbol{\beta})\mathbf{V}^{-1}(\boldsymbol{\beta})\mathbf{D}(\boldsymbol{\beta})$. The expression we derived better tells us how to calculate it:

$$\begin{aligned}
\mathbb{A}_N(\boldsymbol{\beta}) &= \mathbf{D}^\top(\boldsymbol{\beta})\mathbf{V}^{-1}(\boldsymbol{\beta})\mathbf{D}(\boldsymbol{\beta}) \\
&= \mathbf{X}^\top\mathbf{W}(\boldsymbol{\beta})\mathbf{X} \\
&= \mathbf{X}^\top \text{diag}\left(\left(\frac{\partial \mu_i(\boldsymbol{\beta})}{\partial \eta_i(\boldsymbol{\beta})}\right)^2 \frac{1}{V(\mu_i(\boldsymbol{\beta}))}\right)\mathbf{X}.
\end{aligned}$$

The Gauss-Newton approach: More explicitly for GLMs

- Given these forms for $\mathbb{G}_N(\boldsymbol{\beta})$ and $\mathbb{A}_N(\boldsymbol{\beta})$, the Gauss-Newton step can be more expressly described.
- The $(j+1)$-step, $\boldsymbol{\beta}^{(j+1)}$, is given by:

$$\boldsymbol{\beta}^{(j)} + \left[\mathbb{A}_N(\boldsymbol{\beta}^{(j)})\right]^{-1} \mathbb{G}_N(\boldsymbol{\beta}^{(j)})$$

$$\boldsymbol{\beta}^{(j)} + \left[\mathbf{D}^\intercal(\boldsymbol{\beta}^{(j)})\mathbf{V}^{-1}(\boldsymbol{\beta}^{(j)})\mathbf{D}(\boldsymbol{\beta}^{(j)})\right]^{-1} \mathbf{D}^\intercal(\boldsymbol{\beta}^{(j)})\mathbf{V}^{-1}(\boldsymbol{\beta}^{(j)})(\mathbf{y} - \boldsymbol{\mu}(\boldsymbol{\beta}^{(j)}))$$

$$\boldsymbol{\beta}^{(j)} + \left[\mathbf{X}^\intercal\mathbf{W}(\boldsymbol{\beta}^{(j)})\mathbf{X}\right]^{-1} \mathbf{X}^\intercal\mathbf{W}(\boldsymbol{\beta}^{(j)}) \left(\frac{\partial \eta_i(\boldsymbol{\beta}^{(j)})}{\partial \mu_i(\boldsymbol{\beta}^{(j)})}\right) (\mathbf{y} - \text{vec}(g^{-1}(\mathbf{x}_i^\intercal \boldsymbol{\beta}^{(j)}))),$$

  where $\mathbf{W}(\boldsymbol{\beta}^{(j)}) = \text{diag}((\partial \mu_i(\boldsymbol{\beta}^{(j)})/\partial \eta_i(\boldsymbol{\beta}^{(j)}))^2 (1/V(\mu_i(\boldsymbol{\beta}^{(j)}))))$.
- Don't worry—there will be examples of how to code this later in the notes—it's not as bad as it may appear.

The Gauss-Newton approach: GLMs with canonical link

- Under the canonical link, $\mathbf{D}^\intercal(\boldsymbol{\beta})\mathbf{V}^{-1}(\boldsymbol{\beta}) = \mathbf{X}^\intercal$ and $\mathbf{D}(\boldsymbol{\beta}) = \mathbf{V}(\boldsymbol{\beta})\mathbf{X}$.
- Therefore, the estimating equations are given by:

$$\mathbb{G}_N(\boldsymbol{\beta}) = \mathbf{X}^\intercal(\mathbf{y} - \boldsymbol{\mu}(\boldsymbol{\beta})) = \mathbf{0},$$

and $\mathbb{A}_N(\boldsymbol{\beta})$ is given by:

$$\mathbb{A}_N(\boldsymbol{\beta}) = \mathbf{D}^\intercal(\boldsymbol{\beta})\mathbf{V}^{-1}(\boldsymbol{\beta})\mathbf{D}(\boldsymbol{\beta}) = \mathbf{X}^\intercal\mathbf{V}(\boldsymbol{\beta})\mathbf{X}.$$

- With this in mind, the $(j+1)$-step, $\boldsymbol{\beta}^{(j+1)}$, is given by:

$$\boldsymbol{\beta}^{(j)} + \left[\mathbb{A}_N(\boldsymbol{\beta}^{(j)})\right]^{-1}\mathbb{G}_N(\boldsymbol{\beta}^{(j)})$$

$$\boldsymbol{\beta}^{(j)} + \left[\mathbf{D}^\intercal(\boldsymbol{\beta}^{(j)})\mathbf{V}^{-1}(\boldsymbol{\beta}^{(j)})\mathbf{D}(\boldsymbol{\beta}^{(j)})\right]^{-1}\mathbf{D}^\intercal(\boldsymbol{\beta}^{(j)})\mathbf{V}^{-1}(\boldsymbol{\beta}^{(j)})(\mathbf{y} - \boldsymbol{\mu}(\boldsymbol{\beta}^{(j)}))$$

$$\boldsymbol{\beta}^{(j)} + \left[\mathbf{X}^\intercal\mathbf{V}(\boldsymbol{\beta}^{(j)})\mathbf{X}\right]^{-1}\mathbf{X}^\intercal(\mathbf{y} - \text{vec}(g^{-1}(\mathbf{x}_i^\intercal\boldsymbol{\beta}^{(j)}))).$$

The Newton-Raphson approach:

- You can base the algorithm on the *observed* derivative, $\mathbb{A}_N^{\text{obs}}(\boldsymbol{\beta})$, rather than its expectation (referred to as the Newton-Raphson algorithm), though it can be somewhat messier to do so:

$$\begin{aligned}\mathbb{A}_N^{\text{obs}}(\boldsymbol{\beta}) &= -\frac{\partial}{\partial \boldsymbol{\beta}}\mathbb{G}_N(\boldsymbol{\beta}) \\ &= \mathbf{X}^\intercal \mathbf{W}(\boldsymbol{\beta})\mathbf{X} - \left[\frac{\partial}{\partial \boldsymbol{\beta}}\text{vec}\left(\frac{\partial \mu_i(\boldsymbol{\beta})}{\partial \eta_i(\boldsymbol{\beta})}\frac{1}{V(\mu_i(\boldsymbol{\beta}))}\right)\right]\text{diag}(y_i - \mu_i(\boldsymbol{\beta}))\mathbf{X},\end{aligned}$$

  where $\mathbf{W}(\boldsymbol{\beta}^{(j)}) = \text{diag}((\partial\mu_i(\boldsymbol{\beta}^{(j)})/\partial\eta_i(\boldsymbol{\beta}^{(j)}))^2(1/V(\mu_i(\boldsymbol{\beta}))))$.

- Caveat: if the canonical link is used, $\mathbb{A}_N^{\text{obs}}(\boldsymbol{\beta})$ and $\mathbb{A}_N(\boldsymbol{\beta})$ are the same! Why?

Table of Contents

Score equations:

- If forming estimates of the variance based on likelihood theory, we need to revisit the exact score equations. We did this for $\boldsymbol{\beta}$ already:

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}} = \mathbf{D}^{\intercal}\mathbf{V}^{-1}(\mathbf{y} - \boldsymbol{\mu})/\phi.$$

- If $\phi = 1$, there is no need to worry about estimating ϕ, with caveats that we will discuss when we cover quasi-likelihood.
- Now, if there is a nuisance parameter, we do need to give it some attention; the score equations for ϕ take the following form:

$$\begin{aligned}\frac{\partial \ell}{\partial \phi} &= \sum_{i=1}^{N} \frac{\partial}{\partial \phi}\left[\frac{y_i\theta_i - b(\theta_i)}{\phi} + c(y_i, \phi)\right] \\ &= \sum_{i=1}^{N}\left[-\frac{1}{\phi^2}(y_i\theta_i - b(\theta_i)) + c'(y_i, \phi)\right]\end{aligned}$$

Information:

- The (expected) information matrix takes the following form:

$$\mathcal{I}(\boldsymbol{\beta}, \phi) = -\mathrm{E}\begin{bmatrix} \frac{\partial^2 \ell(\boldsymbol{\beta},\phi)}{\partial \boldsymbol{\beta}^2} & \frac{\partial^2 \ell(\boldsymbol{\beta},\phi)}{\partial \boldsymbol{\beta} \partial \phi} \\ \frac{\partial^2 \ell(\boldsymbol{\beta},\phi)}{\partial \phi \partial \boldsymbol{\beta}} & \frac{\partial^2 \ell(\boldsymbol{\beta},\phi)}{\partial \phi^2} \end{bmatrix}$$

- Note the equivalence of the off-diagonal blocks.
- Further, the lower-right block is not of primary interest and we won't compute it explicitly.

Information:

- Tackling the upper-left block:

$$-\mathrm{E}\left[\frac{\partial^2 \ell(\boldsymbol{\beta}, \phi)}{\partial \boldsymbol{\beta}^2}\right] = \mathbf{D}^\intercal \mathbf{V}^{-1} \mathbf{D}/\phi$$

- See presentation of the Gauss-Newton algorithm for this derivation (we're just including the term ϕ this time).

Information:

- Tackling the off-diagonal blocks:

$$
\begin{aligned}
-\mathrm{E}\left[\frac{\partial^2 \ell(\boldsymbol{\beta}, \phi)}{\partial \phi \partial \boldsymbol{\beta}}\right] &= \mathrm{E}\left[\sum_{i=1}^{N} \frac{\partial}{\partial \boldsymbol{\beta}}\left[\frac{1}{\phi^2}(y_i\theta_i - b(\theta_i)) + c'(y_i, \phi)\right]\right] \\
&= \frac{1}{\phi^2}\mathrm{E}\left[\sum_{i=1}^{N} \frac{\partial}{\partial \boldsymbol{\beta}}(y_i\theta_i - b(\theta_i))\right] \\
&= \frac{1}{\phi^2}\mathrm{E}\left[\mathbf{D}^\top \mathbf{V}^{-1}(\mathbf{y} - \boldsymbol{\mu})\right] = \mathbf{0}
\end{aligned}
$$

- See derivation of the score equations for $\boldsymbol{\beta}$ for this result, and recognize the appearance of the unbiased estimating equations.

Information:

- Here's the bottom line:

$$\mathcal{I}(\boldsymbol{\beta}, \phi) = \begin{bmatrix} \mathbf{D}^\top \mathbf{V}^{-1} \mathbf{D}/\phi & \mathbf{0} \\ \mathbf{0}^\top & \dots \end{bmatrix}$$

- Because of the block-diagonal structure of the Fisher information, we need not propagate uncertainty in estimation of ϕ in estimating the variance of $\widehat{\boldsymbol{\beta}}$.
  - Why is this the case? This fact is worth understanding and not one to simply take at face value.

Asymptotic distribution:

- Under standard likelihood theory,

$$\widehat{\boldsymbol{\beta}} \dot{\sim} \mathcal{N}\left(\boldsymbol{\beta}, \phi(\mathbf{D}^\intercal \mathbf{V}^{-1}\mathbf{D})^{-1}\right).$$

- To estimate $\text{Cov}[\widehat{\boldsymbol{\beta}}]$, we could be in one of two cases:
  1. $\phi = 1$, in which case $\widehat{\text{Cov}}[\widehat{\boldsymbol{\beta}}] = (\mathbf{D}^\intercal(\widehat{\boldsymbol{\beta}})\mathbf{V}^{-1}(\widehat{\boldsymbol{\beta}})\mathbf{D}(\widehat{\boldsymbol{\beta}}))^{-1}$.
  2. Otherwise, we require a consistent estimate of ϕ. This one will do:

$$\widehat{\phi} \quad = \quad \frac{1}{N-K}\sum_{i=1}^{N}\frac{(y_i - \widehat{\mu}_i)^2}{V(\widehat{\mu}_i)},$$

and we have that $\widehat{\text{Cov}}[\widehat{\boldsymbol{\beta}}] = \widehat{\phi}(\mathbf{D}^\intercal(\widehat{\boldsymbol{\beta}})\mathbf{V}^{-1}(\widehat{\boldsymbol{\beta}})\mathbf{D}(\widehat{\boldsymbol{\beta}}))^{-1}$, where again, you can use the more detailed formulas for $\mathbf{D}^\intercal$.

Table of Contents

Example: Logistic regression

Example 11.5: Bernoulli distribution (logit link)

- Mean model: $\text{logit}(\text{E}[Y]) = \mathbf{x}^\top\boldsymbol{\beta}$ ($\mu_i = \text{expit}(\mathbf{x}_i^\top\boldsymbol{\beta}) = \text{expit}(\eta_i)$).
- Mean-variance relationship: $\mathbf{V} = \text{diag}(\mu_i(1-\mu_i))$.
- Solving the following equations (previously derived) gives us $\widehat{\boldsymbol{\beta}}$:

$$\mathbf{X}^\top(\mathbf{y} - \text{vec}(\text{expit}(\mathbf{x}_i^\top\boldsymbol{\beta}))) = \mathbf{0}.$$

- Because we are in the setting of the canonical link,

$$\mathbb{A}_N(\boldsymbol{\beta}) = \mathbf{X}^\top\mathbf{V}(\boldsymbol{\beta})\mathbf{X} = \mathbf{X}^\top\text{diag}(\text{expit}(\mathbf{x}_i^\top\boldsymbol{\beta})(1-\text{expit}(\mathbf{x}_i^\top\boldsymbol{\beta})))\mathbf{X}.$$

- Each Gauss-Newton step, $\boldsymbol{\beta}^{(j+1)}$, is given by:

$$\boldsymbol{\beta}^{(j)} + (\mathbf{X}^\top\text{diag}(\text{expit}(\mathbf{x}_i^\top\boldsymbol{\beta}^{(j)})(1-\text{expit}(\mathbf{x}_i^\top\boldsymbol{\beta}^{(j)})))\mathbf{X})^{-1}\mathbf{X}^\top(\mathbf{y} - \text{vec}(\text{expit}(\mathbf{x}_i^\top\boldsymbol{\beta}^{(j)}))).$$

- Further, the variance can be estimated as:

$$\widehat{\text{Cov}}[\widehat{\boldsymbol{\beta}}] \quad = \quad (\mathbf{X}^\top\text{diag}(\text{expit}(\mathbf{x}_i^\top\widehat{\boldsymbol{\beta}})(1-\text{expit}(\mathbf{x}_i^\top\widehat{\boldsymbol{\beta}})))\mathbf{X})^{-1}.$$

Example 11.5: Simulation (Setup)

```
## Define expit function
expit <- function(x)
{
  expitx <- exp(x)/(1 + exp(x))
  return(expitx)
}

## Set seed for reproducibility
set.seed(7345)

## Set sample size
n <- 100

## Set true value of beta
beta <- matrix(c(0.7, 1.2), nrow = 2)
```

Example: Logistic regression

Example 11.5: Simulation (Data generation/set tolerance)

```
## Generate data
X <- matrix(cbind(1, rnorm(n, 0, 1)), ncol = 2)
y <- rbinom(n, 1, expit(X %*% beta))

## Initialize beta
betaj <- c(log(mean(y)/(1 - mean(y))),0)

## Set tolerance and count iterations
tol <- 1
iter <- 1
```

Example 11.5: Simulation (Gauss-Newton iterations)

```
while(tol > 1e-12 & iter < 50)
{
  ## Store prior iteration
  betaj.prior <- betaj

  ## Linear predictor
  etaj <- c(X %*% betaj)

  ## Estimating function
  Gn <- t(X) %*% (y - expit(etaj))

  ## Mean-variance relationship
  Vn <- as.numeric(expit(etaj)*(1 - expit(etaj)))

  ## An
  An <- t(X * Vn) %*% X

  ## Update to next iteration
  betaj <- betaj + solve(An) %*% Gn

  ## Determine if conditions are met
  tol <- sum((betaj - betaj.prior)^2)
  iter <- iter + 1
}
```

Example 11.5: Simulation (Report results)

```
## Report number of iterations
> print(iter)
  [1] 7

## Report tolerance
> print(tol)
  [1] 1.218104e-23
```

Example: Logistic regression

Example 11.5: Simulation (Report estimates/standard errors)

```
## Report estimate
> betaj
             [,1]
  [1,] 0.7115884
  [2,] 1.2642435

## Obtain and report standard errors (re-update)
Vn <- as.numeric(expit(c(X %*% betaj))*(1 - expit(c(X %*% betaj))))
An <- t(X * Vn) %*% X

> sqrt(diag(solve(An)))
  [1] 0.2394554 0.3358070

## Compare to R's glm capabilities
zz <- glm(y ~ X[,2], family = binomial(link = "logit"))

> summary(zz)$coef
              Estimate Std. Error z value    Pr(>|z|)
  (Intercept) 0.711588   0.239450 2.97176 0.002960970
  X[, 2]      1.264244   0.335792 3.76497 0.000166572

## More stringent convergence threshold (perfect agreement)
zz2 <- glm(y ~ X[,2], family = binomial(link = "logit"), control = list(epsilon = 1e-12))

> summary(zz2)$coef
Coefficients:
             Estimate Std. Error z value   Pr(>|z|)
(Intercept) 0.7115884  0.2394554 2.97169 0.00296161 **
X[, 2]      1.2642435  0.3358070 3.76479 0.00016669 ***
```

Example: Logistic regression

Advantages of hard-coding:

- If you create a function to use the `glm()` function to run logistic regression 10,000 times, it takes about four seconds (on my computer).
  - Running the same number of iterations of the hard-coded version takes about one second.
- When we talk about different variance estimation procedures, you'll have more flexibility to modify them as appropriate without being at the mercy of others' pre-written code.
- Know what's going on under the hood!

Aside: Computational efficiency

A note about computation:

- Keep in mind that I employed a computational efficiency trick in hard-coding the GLM.
- We often write various estimating equations, algorithms, and variance estimators using the notation diag(·).
- A large diagonal matrix has a lot of zeros; multiplying diagonal matrices the "literal" way is computationally wasteful.
- In R, the following two commands produce the same answer:

```
out1 <- t(X) %*% diag(z)
out2 <- t(X * z)
```

- . . . but the difference in computation time is *not* subtle when your data set is of moderate or larger size. Let's see a speed test in action.

Aside: Computational efficiency

A note about computation: Speed test (small data)

```
## Set seed for reproducibility (computing time varies)
set.seed(7345)

## Sample size
N <- 100

## Number of parameters
K <- 2

## Matrix
X <- matrix(rnorm(N*K), nrow = N)

## Vector
z <- rnorm(N)

## Using our symbolic representation
> system.time(for (j in 1:50) {t(X) %*% diag(z)})
   user  system elapsed
  0.004   0.001   0.004

## Computational trick
> system.time(for (j in 1:50){t(X * z)})
   user  system elapsed
  0.001   0.000   0.001

## No big deal so far...
```

Aside: Computational efficiency

A note about computation: Speed test (less-small data)

```
## Set seed for reproducibility (computing time varies)
set.seed(7345)

## Sample size
N <- 500

## Number of parameters
K <- 3

## Matrix
X <- matrix(rnorm(N*K), nrow = N)

## Vector
z <- rnorm(N)

## Using our symbolic representation
> system.time(for (j in 1:50) {t(X) %*% diag(z)})
   user  system elapsed
  0.061   0.006   0.069

## Computational trick
> system.time(for (j in 1:50){t(X * z)})
   user  system elapsed
  0.001   0.000   0.001

## Still not convinced...
```

Aside: Computational efficiency

A note about computation: Speed test (healthy-sized data)

```
## Set seed for reproducibility (computing time varies)
set.seed(7345)

## Sample size
N <- 1000

## Number of parameters
K <- 4

## Matrix
X <- matrix(rnorm(N*K), nrow = N)

## Vector
z <- rnorm(N)

## Using our symbolic representation
> system.time(for (j in 1:50) {t(X) %*% diag(z)})
   user  system elapsed
  0.263   0.025   0.289

## Computational trick
> system.time(for (j in 1:50){t(X * z)})
   user  system elapsed
  0.002   0.000   0.002

## I might be getting closer to being convinced...
```

Aside: Computational efficiency

A note about computation: Speed test (moderately-sized data)

```
## Set seed for reproducibility (computing time varies)
set.seed(7345)

## Sample size
N <- 5000

## Number of parameters
K <- 4

## Matrix
X <- matrix(rnorm(N*K), nrow = N)

## Vector
z <- rnorm(N)

## Using our symbolic representation
> system.time(for (j in 1:50) {t(X) %*% diag(z)})
   user  system elapsed
  9.931   0.827  10.772

## Computational trick
> system.time(for (j in 1:50){t(X * z)})
   user  system elapsed
  0.004   0.001   0.005

## I think I'm convinced...
```

Aside: Computational efficiency

A note about computation: Speed test (hefty data)

```
## Set seed for reproducibility (computing time varies)
set.seed(7345)

## Sample size
N <- 10000

## Number of parameters
K <- 4

## Matrix
X <- matrix(rnorm(N*K), nrow = N)

## Vector
z <- rnorm(N)

## Using our symbolic representation
> system.time(for (j in 1:50) {t(X) %*% diag(z)})
   user  system elapsed
 31.672   3.296  35.307

## Computational trick
> system.time(for (j in 1:50){t(X * z)})
   user  system elapsed
  0.012   0.004   0.017

## Getting out of hand!!
```

Aside: Computational efficiency

A note about computation:

- The bottom line is this: we'll continue to use diag($\cdot$) notation as our typical symbolic representation of estimating equations, algorithms, variances, etc.
- For coding, however, you will save yourself time by taking advantage of these simple computational tricks.
- Let's try another example of a GLM we've probably not encountered before.

Example: Normal regression with a log link

Example 11.6: Normal regression (log link)

- Mean model: $\log(\mathrm{E}[Y]) = \mathbf{x}^\top\boldsymbol{\beta}$ ($\mu_i = \exp(\mathbf{x}_i^\top\boldsymbol{\beta}) = \exp(\eta_i)$).
- Mean-variance relationship: $\mathbf{V} = \mathbf{I}$.
- $\theta_i = \mu_i$ is the natural parameter; $b(\theta_i) = \mu_i^2/2$, and $b''(\theta_i) = 1$ (see prior notes in which we factored density into exponential form).
- Solving the following equations (previously derived) gives us $\widehat{\boldsymbol{\beta}}$:

$$\mathbf{D}^\top\mathbf{V}^{-1}(\mathbf{y}-\boldsymbol{\mu}) = \mathbf{X}^\top\mathrm{diag}(\exp(\mathbf{x}_i^\top\boldsymbol{\beta}))(\mathbf{y}-\mathrm{vec}(\exp(\mathbf{x}_i^\top\boldsymbol{\beta}))) = \mathbf{0}.$$

- Because this is a non-canonical link,

$$\begin{aligned}\mathbb{A}_N(\boldsymbol{\beta}) &= \mathbf{D}^\top\mathbf{V}^{-1}\mathbf{D}^{-1} = \mathbf{X}^\top\mathrm{diag}\left(\left(\frac{\partial\mu_i(\boldsymbol{\beta})}{\partial\eta_i(\boldsymbol{\beta})}\right)^2\frac{1}{V(\mu_i(\boldsymbol{\beta}))}\right)\mathbf{X}\\ &= \mathbf{X}^\top\mathrm{diag}(\exp(\mathbf{x}_i^\top\boldsymbol{\beta}))^2\mathbf{X}.\end{aligned}$$

Example: Normal regression with a log link

Example 11.6: Normal regression (log link)

- Each Gauss-Newton step, $\boldsymbol{\beta}^{(j+1)}$, is given by:

$$\boldsymbol{\beta}^{(j)} + (\mathbf{X}^\top \text{diag}(\exp(\mathbf{x}_i^\top \boldsymbol{\beta}^{(j)}))^2 \mathbf{X})^{-1} \mathbf{X}^\top \text{diag}(\exp(\mathbf{x}_i^\top \boldsymbol{\beta}^{(j)}))(\mathbf{y} - \text{vec}(\exp(\mathbf{x}_i^\top \boldsymbol{\beta}^{(j)}))).$$

- Here, we have a nuisance parameter that must be estimated:

$$\widehat{\phi} = \frac{1}{N-K} \sum_{i=1}^{N} \frac{(y_i - \widehat{\mu}_i)^2}{V(\widehat{\mu}_i)} = \frac{1}{N-K} \sum_{i=1}^{N} (y_i - \exp(\mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}))^2.$$

- Therefore, this gives rise to the following estimator for $\text{Cov}[\widehat{\boldsymbol{\beta}}]$:

$$\widehat{\text{Cov}}[\widehat{\boldsymbol{\beta}}] = \left(\frac{1}{N-K} \sum_{i=1}^{N} (y_i - \exp(\mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}))^2 \right) \left(\mathbf{X}^\top \text{diag}(\exp(\mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}))^2 \mathbf{X} \right)^{-1}.$$

Example: Normal regression with a log link

Example 11.6: Simulation (Setup)

```
## Set seed for reproducibility
set.seed(7345)

## Set sample size
n <- 50

## Set true value of beta
beta <- matrix(c(-0.5, 0.6), nrow = 2)
```

Example: Normal regression with a log link

Example 11.6: Simulation (Data generation/set tolerance)

```
## Generate data
X <- matrix(cbind(1, rnorm(n, 0, 1)), ncol = 2)
y <- rnorm(n, exp(X %*% beta))

## Initialize beta
betaj <- c(log(mean(y)), 0)

## Set tolerance and count iterations
tol <- 1
iter <- 1
```

Example: Normal regression with a log link

Example 11.6: Simulation (Gauss-Newton iterations)

```
while(tol > 1e-15 & iter < 50)
{
  ## Store prior iteration
  betaj.prior <- betaj

  ## Linear predictor
  etaj <- c(X %*% betaj)

  ## Estimating function
  Gn <- t(X * exp(etaj)) %*% (y - exp(etaj))

  ## An
  An <- t(X * exp(etaj)^2) %*% X

  ## Update to next iteration
  betaj <- betaj + solve(An) %*% Gn

  ## Nuisance
  phi <- sum((y - exp(etaj))^2)/(n - 2)

  ## Determine if conditions are met
  tol <- sum((betaj - betaj.prior)^2)
  iter <- iter + 1
}
```

Example: Normal regression with a log link

Example 11.6: Simulation (Report results)

```
## Report number of iterations
> print(iter)
  [1] 6

## Report tolerance
> print(tol)
  [1] 4.620906e-17
```

Example: Normal regression with a log link

Example 11.6: Simulation (Report estimates/standard errors)

```
## Report estimate
> betaj
           [,1]
  [1,] -0.2832756
  [2,]  0.4282233

## Obtain and report standard errors (re-update)
An <- t(X * exp(c(X %*% betaj))^2) %*% X

> sqrt(diag(phi * solve(An)))
  [1] 0.1804808 0.1373220

## Compare to R's glm capabilities
zz <- glm(y ~ X[,2], start = c(log(mean(y)), 0), family = gaussian(link = "log"))

> summary(zz)$coef
              Estimate Std. Error z value Pr(>|z|)
  (Intercept)  -0.2833     0.1805  -1.570  0.12309
  X[, 2]        0.4282     0.1373   3.118  0.00307 **
```

Example: Normal regression with a log link

Example 11.6: Normal regression (log link)

- Let's try re-doing this based on the Newton-Raphson algorithm.
- Recall: $(\partial/\partial\boldsymbol{\beta})\text{vec}(f(\mathbf{x}_i^\intercal\boldsymbol{\beta})) = \mathbf{X}^\intercal\text{diag}(f'(\mathbf{x}_i^\intercal\boldsymbol{\beta}))$.
- More detail: $\mu_i(\boldsymbol{\beta}) = \exp(\mathbf{x}_i^\intercal\boldsymbol{\beta})$ and $b''(\theta_i(\boldsymbol{\beta})) = 1$ in this example, so:

$$\begin{aligned}
\frac{\partial}{\partial\boldsymbol{\beta}}\left[\text{vec}\left(\frac{\partial\mu_i(\boldsymbol{\beta})}{\partial\eta_i(\boldsymbol{\beta})}\frac{1}{V(\mu_i(\boldsymbol{\beta}))}\right)\right] &= \frac{\partial}{\partial\boldsymbol{\beta}}\left[\text{vec}\left(\frac{\partial\mu_i(\boldsymbol{\beta})}{\partial\eta_i(\boldsymbol{\beta})}\right)\right] = \frac{\partial}{\partial\boldsymbol{\beta}}\left[\text{vec}(\exp(\mathbf{x}_i^\intercal\boldsymbol{\beta}))\right] \\
&= \begin{bmatrix} \exp(\mathbf{x}_1^\intercal\boldsymbol{\beta}) & \exp(\mathbf{x}_2^\intercal\boldsymbol{\beta}) & \cdots & \exp(\mathbf{x}_N^\intercal\boldsymbol{\beta}) \\ x_{11}\exp(\mathbf{x}_1^\intercal\boldsymbol{\beta}) & x_{21}\exp(\mathbf{x}_2^\intercal\boldsymbol{\beta}) & \cdots & x_{N1}\exp(\mathbf{x}_N^\intercal\boldsymbol{\beta}) \\ \vdots & \vdots & \ddots & \vdots \\ x_{1K}\exp(\mathbf{x}_1^\intercal\boldsymbol{\beta}) & x_{2K}\exp(\mathbf{x}_2^\intercal\boldsymbol{\beta}) & \cdots & x_{NK}\exp(\mathbf{x}_N^\intercal\boldsymbol{\beta}) \end{bmatrix} \\
&= \mathbf{X}^\intercal\text{diag}(\exp(\mathbf{x}_i^\intercal\boldsymbol{\beta})).
\end{aligned}$$

- Therefore,

$$\begin{aligned}
\mathbb{A}_N^{\text{obs}}(\boldsymbol{\beta}) &= \mathbb{A}_N(\boldsymbol{\beta}) - \frac{\partial}{\partial\boldsymbol{\beta}}\left[\text{vec}\left(\frac{\partial\mu_i(\boldsymbol{\beta})}{\partial\eta_i(\boldsymbol{\beta})}\frac{1}{V(\mu_i(\boldsymbol{\beta}))}\right)\right]\text{diag}(y_i - \mu_i(\boldsymbol{\beta}))\mathbf{X} \\
&= \mathbf{X}^\intercal\text{diag}\big(\exp(\mathbf{x}_i^\intercal\boldsymbol{\beta})^2 - \exp(\mathbf{x}_i^\intercal\boldsymbol{\beta})(y_i - \exp(\mathbf{x}_i^\intercal\boldsymbol{\beta}))\big)\mathbf{X}.
\end{aligned}$$

Example: Normal regression with a log link

Example 11.6: Simulation (Newton-Raphson iterations)

```
while(tol > 1e-15 & iter < 50)
{
  ## Store prior iteration
  betaj.prior <- betaj

  ## Linear predictor
  etaj <- c(X %*% betaj)

  ## Estimating function
  Gn <- t(X * exp(etaj)) %*% (y - exp(etaj))

  ## An
  An <- t(X * exp(etaj)^2) %*% X

  ## AnObs
  AnObs <- An - t(X * c(exp(etaj) * (y - exp(etaj)))) %*% X

  ## Update to next iteration
  betaj <- betaj + solve(AnObs) %*% Gn

  ## Nuisance
  phi <- sum((y - exp(etaj))^2)/(n - 2)

  ## Determine if conditions are met
  tol <- sum((betaj - betaj.prior)^2)
  iter <- iter + 1
}
```

Example: Normal regression with a log link

Example 11.6: Simulation (Report results)

```
## Report number of iterations
> print(iter)
  [1] 8

## Report tolerance
> print(tol)
  [1] 2.501604e-23

## Report estimate
>      betaj
            [,1]
[1,] -0.2832756
[2,]  0.4282233
```

Example: Normal regression with a log link

Notes:

- It is typical for the Gauss-Newton algorithm to converge in fewer iterations. The algorithms converge to the **same** quantity.
- Note that the likelihood-based variance estimator is still based on $\mathbb{A}_N(\widehat{\boldsymbol{\beta}})$. We don't change the variance estimator based on the algorithm (although we will discuss in the next set of notes other uses of $\mathbb{A}_N^{\text{obs}}(\widehat{\boldsymbol{\beta}})$ for variance estimation).

Example: Inverse Gaussian distribution

Example 11.7: Inverse-Gaussian

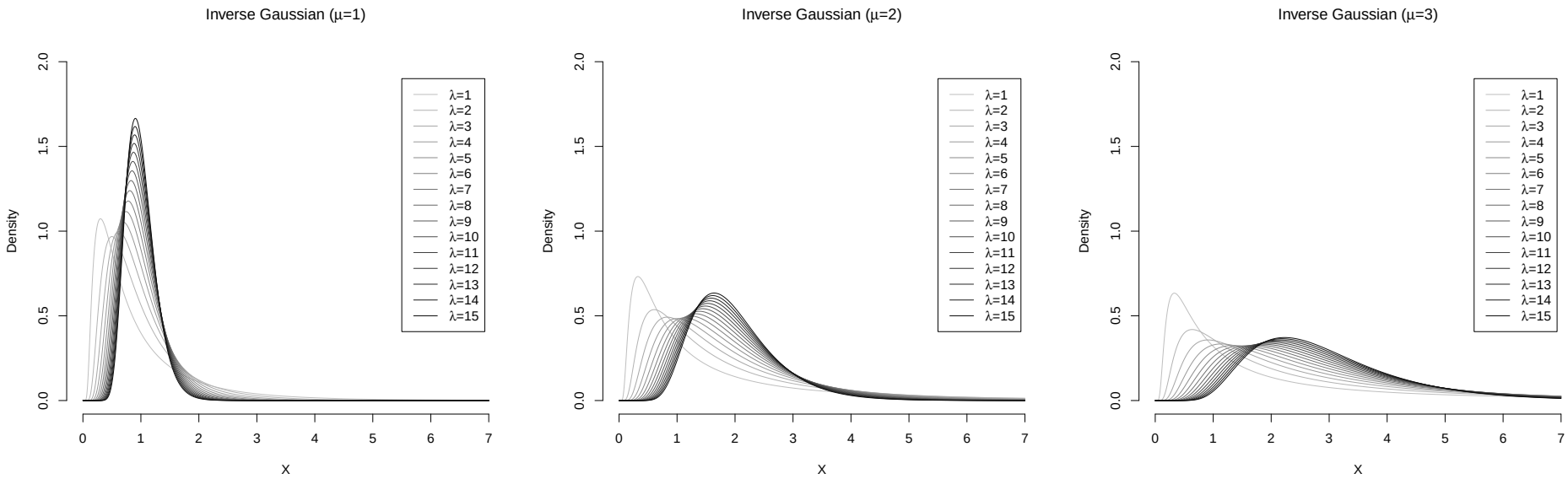
- Let's go through an example that we haven't worked with as closely. Consider modeling the outcome as an inverse Gaussian distribution, which has the following density function:

$$f_X(x;\mu,\lambda) = \sqrt{\frac{\lambda}{2\pi x^3}} \exp\left(-\frac{\lambda(x-\mu)^2}{2\mu^2 x}\right) 1(x>0).$$

- Parameters: $\mu > 0$ (mean), and $\lambda > 0$ (shape) or $1/\lambda$ (dispersion).
- Goals for this example:
  1. Write this exponential family in the canonical form.
  2. Determine E$[Y]$ and Var$[Y]$ based on this canonical form.
  3. Determine the canonical link based on the corresponding GLM.
  4. Determine the score equations for $\boldsymbol{\beta}$ based on the canonical link.
  5. Determine the variance of $\widehat{\boldsymbol{\beta}}$ based on the above model.
  6. Use R to iterate the Gauss-Newton algorithm on a simulated example.

Example: Inverse Gaussian distribution

Example 11.7: Density function



Example: Inverse Gaussian distribution

Example 11.7: Density function in canonical form

- Time for some algebra:

$$\begin{aligned} f_Y(y;\mu,\lambda) &= \sqrt{\frac{\lambda}{2\pi y^3}}\exp\left(-\frac{\lambda(y-\mu)^2}{2\mu^2 y}\right) \\ &= \exp\left(-\frac{\lambda(y^2-2\mu y+\mu^2)}{2\mu^2 y}+\frac{1}{2}\left(\log(\lambda)-\log(2\pi y^3)\right)\right) \\ &= \exp\left(-\frac{y/2\mu^2-1/\mu+1/2y}{1/\lambda}+\frac{1}{2}\left(\log(\lambda)-\log(2\pi y^3)\right)\right) \\ &= \exp\left(\frac{y(-1/(2\mu^2))-(-1/\mu)}{1/\lambda}+\frac{1}{2}\left(\frac{\lambda}{y}+\log(\lambda)-\log(2\pi y^3)\right)\right). \end{aligned}$$

- Natural parameter: $\theta = -1/(2\mu^2)$; nuisance parameter: $\phi = 1/\lambda$.
- $b(\theta) = -(-2\theta)^{1/2}$.
- $\mathrm{E}[Y] = b'(\theta) = (-2\theta)^{-1/2} = \mu$.
- $b''(\theta) = (-2\theta)^{-3/2} \Rightarrow V(\mu) = \mu^3$.
- $\mathrm{Var}[Y] = \phi b''(\theta) = \cdots = \mu^3/\lambda$.

Example: Inverse Gaussian distribution

Example 11.7: Some points of caution

- What's to stop us from choosing $\theta = 1/\mu^2$ and $\phi = -2/\lambda$?
  - Well... nothing. In fact, a prior iteration of these slides had exactly that parameterization and everything worked out fine...
- However, there is some benefit to choosing the nuisance parameter such that $b''(\theta)$ is a non-scaled transformation of $\mu = b'(\theta)$.
- In particular, having a standard—even if arbitrary—procedure helps us avoid getting tripped up by ambiguity.
  - Defaulting to $V(\mu) = \mu^3$ (equality rather than proportionality) helps with what you might think of as "GLM bookkeeping."

Example: Inverse Gaussian distribution

Example 11.7: Canonical link function

- The canonical link function is based on $g^{-1}(\cdot) = b'(\cdot)$.
- More explicitly for an inverse Gaussian GLM:

$$\begin{aligned} g^{-1}(\eta_i) &= b'(\theta) \\ &= 1/\sqrt{-2\theta} \\ \Rightarrow g(\mu_i) &= -1/(2\mu_i^2) \end{aligned}$$

- The above formulation of the canonical link is *technically correct*, but in practice we often choose $g(\mu_i) = 1/\mu_i^2$ (this does not change the mechanics of the GLM procedures).
- Under this "near-canonical" link:

$$\begin{aligned} 1/\mu^2 = 1/(\mathrm{E}[Y])^2 &= \eta = \mathbf{x}^\top \boldsymbol{\beta}, \text{ or} \\ \mu = \mathrm{E}[Y] &= 1/\sqrt{\eta} = 1/\sqrt{\mathbf{x}^\top \boldsymbol{\beta}}. \end{aligned}$$

Example: Inverse Gaussian distribution

Example 11.7: Score equations

- Under this "near-canonical" link, the score function for $\boldsymbol{\beta}$ is given by:

$$\begin{aligned}
\mathbb{G}_N(\boldsymbol{\beta}) &= \mathbf{X}^\top \text{diag}((\partial\mu_i/\partial\eta_i))/V_i)\,(\mathbf{y} - \text{vec}(\mu_i))\,/\phi \\
&= \mathbf{X}^\top \text{diag}\left(-\frac{1}{2}(\mathbf{x}_i^\top\boldsymbol{\beta})^{-3/2}\frac{1}{(\mathbf{x}_i^\top\boldsymbol{\beta})^{-3/2}}\right)\left(\mathbf{y} - \text{vec}((\mathbf{x}_i^\top\boldsymbol{\beta})^{-1/2})\right)/\phi \\
&= (-1/2)\mathbf{X}^\top\left(\mathbf{y} - \text{vec}((\mathbf{x}_i^\top\boldsymbol{\beta})^{-1/2})\right)/\phi.
\end{aligned}$$

- While you can ignore the nuisance when solving the estimating equations, I believe you'll ultimately be happier leaving the "$-1/2$" in so that you never have to worry if you threw off a constant in the $V(\cdot)$ function.
- Any such constant will be absorbed into estimation of ϕ, but...
  - What if $\phi = 1$ is known?
  - Even if ϕ is unknown, you might be wondering why you're not estimating it consistently (say, in a simulation).

Example: Inverse Gaussian distribution

Example 11.7: Gauss-Newton implementation

- We can determine $\mathbb{A}_N(\boldsymbol{\beta})$ as follows:

$$\begin{aligned}\mathbb{A}_N(\boldsymbol{\beta}) &= \mathbf{X}^\mathsf{T}\text{diag}\left(\left(-\frac{1}{2}(\mathbf{x}_i^\mathsf{T}\boldsymbol{\beta})^{-3/2}\right)^2 \frac{1}{(\mathbf{x}_i^\mathsf{T}\boldsymbol{\beta})^{-3/2}}\right)\mathbf{X} \\ &= \frac{1}{4}\mathbf{X}^\mathsf{T}\text{diag}\left((\mathbf{x}_i^\mathsf{T}\boldsymbol{\beta})^{-3/2}\right)\mathbf{X}.\end{aligned}$$

- Having a "near-canonical" link just means that the usual mechanics of a canonical link are off by a constant.
- Each Gauss-Newton step, $\boldsymbol{\beta}^{(j+1)}$, is given by:

$$\boldsymbol{\beta}^{(j+1)} = \boldsymbol{\beta}^{(j)} - 2(\mathbf{X}^\mathsf{T}\text{diag}((\mathbf{x}_i^\mathsf{T}\boldsymbol{\beta}^{(j)})^{-3/2})\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\left(\mathbf{y} - \text{vec}((\mathbf{x}_i^\mathsf{T}\boldsymbol{\beta}^{(j)})^{-1/2})\right).$$

- What do you think the Newton-Raphson algorithm will look like?

Example: Inverse Gaussian distribution

Example 11.7: Variance

- We can also determine the covariance matrix for $\widehat{\boldsymbol{\beta}}$:

$$\text{Cov}[\widehat{\boldsymbol{\beta}}] \quad = \quad 4\phi(\mathbf{X}^{\mathsf{T}}\text{diag}((\mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta})^{-3/2}))\mathbf{X})^{-1}.$$

- We can estimate the nuisance parameter in the usual way:

$$\widehat{\phi} \quad = \quad \frac{1}{N-K}\sum_{i=1}^{N}\frac{(y_i-\widehat{\mu}_i)^2}{V(\widehat{\mu}_i)},$$

where—by construction—we chose the parameterization in which $V(\mu) = \mu^3$, such that $\widehat{\phi}$ will be consistent for ϕ.

Example: Inverse Gaussian distribution

Example 11.7: Simulation (Setup)

```
## Set seed for reproducibility
set.seed(7345)

## Important library for inverse Gaussian
library("statmod")

## Set sample size
n <- 5000

## Set true value of beta
beta <- matrix(c(0.5, 0.2), nrow = 2)
```

Example: Inverse Gaussian distribution

Example 11.7: Simulation (Data generation/set tolerance)

```
## Generate data
X <- matrix(cbind(1, runif(n, 0, 5)), ncol = 2)
y <- rinvgauss(n, mean = 1/sqrt((X %*% beta)), dispersion = 1.2)

## Initialize beta
betaj <- c(1/mean(y)^2, 0)

## Set tolerance and count iterations
tol <- 1
iter <- 1
```

Example: Inverse Gaussian distribution

Example 11.7: Simulation (Gauss-Newton iterations)

```
while(tol > 1e-12 & iter < 50)
{
  ## Store prior iteration
  betaj.prior <- betaj

  ## Linear predictor
  etaj <- c(X %*% betaj)

  ## Estimating function
  Gn <- -(1/2)*t(X) %*% (y - 1/sqrt(etaj))

  ## An
  An <- t(X * c((1/4)*(etaj^(-3/2)))) %*% X

  ## Update to next iteration
  betaj <- betaj + solve(An) %*% Gn

  ## Nuisance
  num <- (y - 1/(X %*% betaj)^(1/2))
  denom <- (X %*% betaj)^(-3/2)
  phi <- (1/(n - 2)) * sum((num^2/denom))

  ## Determine if conditions are met
  tol <- sum((betaj - betaj.prior)^2)
  iter <- iter + 1
}
```

Example: Inverse Gaussian distribution

Example 11.7: Simulation (Report results)

```
## Report number of iterations
> print(iter)
  [1] 6

## Report tolerance
> print(tol)
  [1] 2.407682e-17
```

Example: Inverse Gaussian distribution

Example 11.7: Simulation (Report estimates/standard errors)

```
## Report estimates
> betaj
            [,1]
 [1,] 0.4680579
 [2,] 0.2093309

## Estimate of phi
> phi
[1] 1.118703

## Obtain and report standard errors (re-update)
etaj <- c(X %*% betaj)
An <- t(X * c((1/4)*(etaj^(-3/2)))) %*% X

> sqrt(diag(phi*solve(An)))
  [1] 0.04374994 0.01953502

## Compare to R's glm capabilities
zz <- glm(y ~ X[,2], start = c(1/mean(y)^2, 0), family = inverse.gaussian(),
          control = list(epsilon = 1e-12))

> summary(zz)$coef
              Estimate Std. Error t value Pr(>|t|)
  (Intercept)  0.46806    0.04375   10.70   <2e-16 ***
  X[, 2]       0.20933    0.01954   10.72   <2e-16 ***

(Dispersion parameter for inverse.gaussian family taken to be 1.118703)
```

Table of Contents

Rewriting the Gauss-Newton algorithm:

- We can expand the $(j+1)$ step of the Gauss-Newton algorithm:

$$\begin{aligned}
\boldsymbol{\beta}^{(j+1)} &= \boldsymbol{\beta}^{(j)} + \left[\mathbf{X}^\intercal \mathbf{W}(\boldsymbol{\beta}^{(j)})\mathbf{X}\right]^{-1} \mathbf{X}^\intercal \mathbf{W}(\boldsymbol{\beta}^{(j)}) \left[\text{diag}\left(\frac{\partial \boldsymbol{\eta}(\boldsymbol{\beta})}{\partial \boldsymbol{\mu}(\boldsymbol{\beta})}\bigg|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(j)}}\right)(\mathbf{y} - \boldsymbol{\mu}(\boldsymbol{\beta}^{(j)}))\right] \\
&= \left[\mathbf{X}^\intercal \mathbf{W}(\boldsymbol{\beta}^{(j)})\mathbf{X}\right]^{-1} \left[\mathbf{X}^\intercal \mathbf{W}(\boldsymbol{\beta}^{(j)})\mathbf{X}\boldsymbol{\beta}^{(j)} + \mathbf{X}^\intercal \mathbf{W}(\boldsymbol{\beta}^{(j)}) \left[\text{diag}\left(\frac{\partial \boldsymbol{\eta}(\boldsymbol{\beta})}{\partial \boldsymbol{\mu}(\boldsymbol{\beta})}\bigg|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(j)}}\right)(\mathbf{y} - \boldsymbol{\mu}(\boldsymbol{\beta}^{(j)}))\right]\right] \\
&= \left[\mathbf{X}^\intercal \mathbf{W}(\boldsymbol{\beta}^{(j)})\mathbf{X}\right]^{-1} \mathbf{X}^\intercal \mathbf{W}(\boldsymbol{\beta}^{(j)}) \left[\mathbf{X}\boldsymbol{\beta}^{(j)} + \left[\text{diag}\left(\frac{\partial \boldsymbol{\eta}(\boldsymbol{\beta})}{\partial \boldsymbol{\mu}(\boldsymbol{\beta})}\bigg|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(j)}}\right)(\mathbf{y} - \boldsymbol{\mu}(\boldsymbol{\beta}^{(j)}))\right]\right] \\
&= \left[\mathbf{X}^\intercal \mathbf{W}(\boldsymbol{\beta}^{(j)})\mathbf{X}\right]^{-1} \mathbf{X}^\intercal \mathbf{W}(\boldsymbol{\beta}^{(j)})\mathbf{z}(\boldsymbol{\beta}^{(j)}),
\end{aligned}$$

where $\mathbf{z}(\boldsymbol{\beta}^{(j)}) = \mathbf{X}\boldsymbol{\beta}^{(j)} + \left[\text{diag}\left(\frac{\partial \boldsymbol{\eta}(\boldsymbol{\beta})}{\partial \boldsymbol{\mu}(\boldsymbol{\beta})}\bigg|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(j)}}\right)(\mathbf{y} - \boldsymbol{\mu}(\boldsymbol{\beta}^{(j)}))\right]$

- $r_{ij}^W = \frac{\partial \eta_i(\boldsymbol{\beta})}{\partial \mu_i(\boldsymbol{\beta})}\Big|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(j)}}(y_i - \mu_i(\boldsymbol{\beta}^{(j)}))$ is known as the *working residual*.

The Gauss-Newton algorithm and IRLS

IRLS:

- The Gauss-Newton algorithm and IRLS are equivalent!
- As an informal exercise, try to hard-code one of the previous examples using IRLS and verify that you're getting the same answer.

Table of Contents

Example: Binomial regression

Example 11.8: Binomial experiments in trial/success format

- Consider the following data set emerging from a set of binomial experiments within each of seven levels of X (treated continuously):

x	n_x	y_x
0	50	22
1	25	7
2	60	30
3	70	50
4	20	14
5	50	39
6	30	28

- n_x represents number of trials; y_x represents number of successes.
- $N = \sum_x n_x = 305$. Could represent the data as a 305×2 array with y_x converted to binary format and run logistic regression, but it is faster to use the compact data representation.

Example: Binomial regression

Example 11.8: Binomial experiments in trial/success format

- Note: $\mathrm{E}[y_x|X = x] = n_x p_x$.
- Note: $\mathrm{Var}[y_x|X = x] = n_x p_x(1 - p_x)$.
- Estimating equations (canonical link):

$$\mathbf{X}^\top(\mathbf{y} - \mathrm{diag}(n_X)\mathrm{vec}(\mathrm{expit}(\mathbf{x}_i^\top\boldsymbol{\beta}))) = \mathbf{0}.$$

- Gauss-Newton step:

$$\boldsymbol{\beta}^{(j+1)} = \boldsymbol{\beta}^{(j)} + (\mathbf{X}^\top\mathbf{V}(\boldsymbol{\beta}^{(j)})\mathbf{X})^{-1}\mathbf{X}^\top(\mathbf{y} - \mathrm{diag}(n_X)\mathrm{vec}(\mathrm{expit}(\mathbf{x}_i^\top\boldsymbol{\beta}^{(j)}))),$$

  where $\mathbf{V}(\boldsymbol{\beta}^{(j)}) = \mathrm{diag}(n_x\mathrm{expit}(\mathbf{x}_i^\top\boldsymbol{\beta}^{(j)})(1 - \mathrm{expit}(\mathbf{x}_i^\top\boldsymbol{\beta}^{(j)})))$.

- Using `glm()` in R:

```
glm(cbind(y, n-y) ~ x, family = binomial(link = "logit"))
```

Table of Contents

Example: Poisson regression

Example 11.9: Offsets

- Poisson regression is commonly used to analyze count data ($Y \in \mathbb{N}$).
- Sometimes, counts measured over variable ranges of space-time.
  - Number of asthma exacerbations over a length of time that may vary from patient to patient.
  - Number of SARS-CoV-2 cases emerging in counties that have different population sizes.
- Including an *offset term* standardizes the region of comparison.
- Model (log link): $\log(\mathrm{E}[Y]/W) = \mathbf{x}^\mathsf{T}\boldsymbol{\beta} \Rightarrow \log(\mathrm{E}[Y]) = \log(W) + \mathbf{x}^\mathsf{T}\boldsymbol{\beta}$.
- Estimating equations (convince yourself that this is correct):

$$\mathbf{X}^\mathsf{T}(\mathbf{y} - \mathrm{diag}(w_i)\mathrm{vec}(\exp(\mathbf{x}_i^\mathsf{T}\boldsymbol{\beta}))) = \mathbf{0}$$

- Using `glm()` in R:

```
glm(y ~ x, offset = log(w), family = poisson(link = "log"))
```

Table of Contents

Saturation and super-saturation

Linear models: Discrete strata

- Suppose you have N observations, $(X_1, Y_1), \ldots, (X_N, Y_N)$.
  - Consider X binary (0/1).
- Consider fitting the following regression model:

$$\mathrm{E}[Y|X = x] = \beta_0 + \beta_1 x.$$

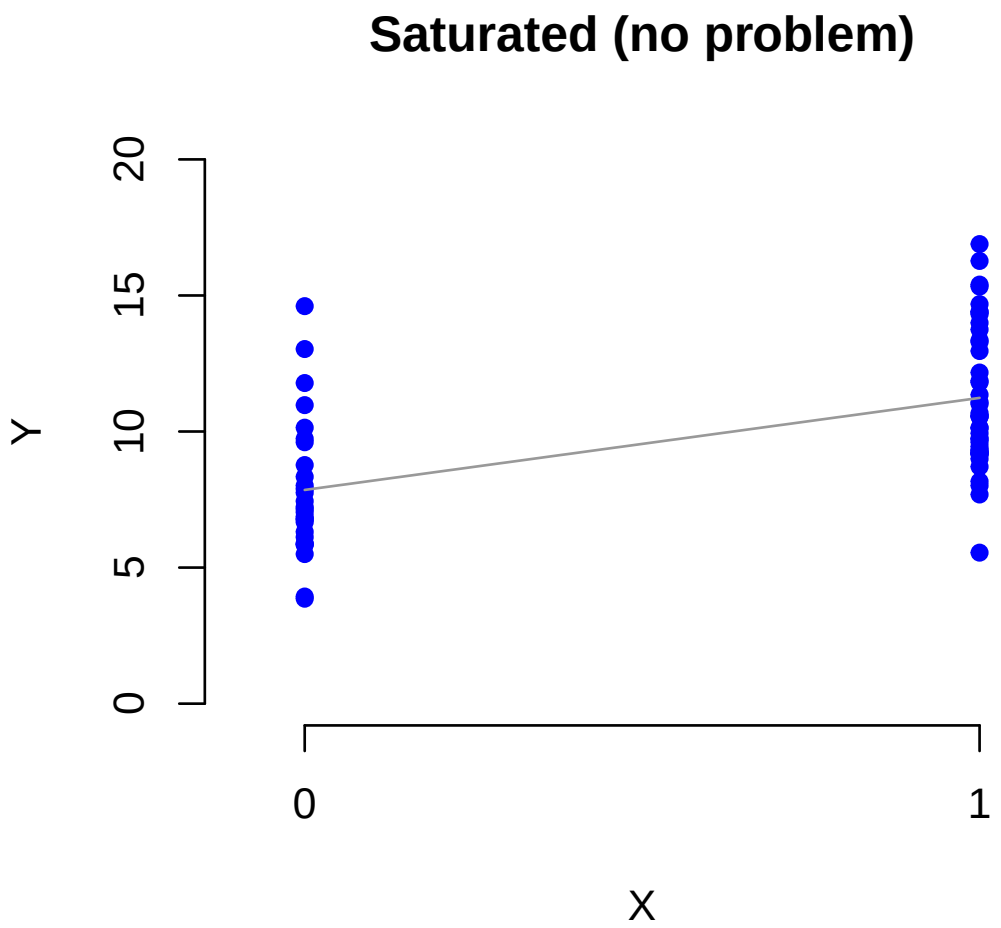
- You may already know:
  - Each of the two group gets its own mean.
  - No borrowing of information.
  - Essentially a two-sample t-test.
- It is possible to build saturated models when the strata defined by **x** are discrete.

Linear models: Discrete strata

```
## Set seed for reproducibility
set.seed(7345)

## Generate data
n <- 70
x <- rbinom(n, 1, 0.5)
y <- 7 + 4*x + rnorm(n, 0, 3)

## Simple linear regression (t-test, effectively)
zz <- lm(y ~ x)

> summary(zz)
Call:
lm(formula = y ~ x)

Residuals:
   Min     1Q Median     3Q    Max
-5.686 -1.877 -0.666  1.851  6.753

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    7.854      0.485    16.2   <2e-16 ***
x              3.382      0.626     5.4    9e-07 ***
---
Residual standard error: 2.57 on 68 degrees of freedom
Multiple R-squared:   0.3,  Adjusted R-squared:  0.29
F-statistic: 29.2 on 1 and 68 DF,  p-value: 9.02e-07

## No concerns!
```

Saturation and super-saturation

Linear models: Discrete strata



Saturation and super-saturation

Linear models: Continuous predictor

- Now, suppose you have N observations, $(X_1, Y_1), \ldots, (X_N, Y_N)$, with the values of X continuous.
- Consider fitting the following regression model:

$$\mathrm{E}[Y|X = x] = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_{N-1} x^{N-1}.$$

- You may already know that this will fail beautifully:
  - Every observation will be predicted perfectly.
  - $\widehat{\sigma}^2 = 0$.
- No inference on $\boldsymbol{\beta}$ possible.

Linear models: Polynomial interpolation

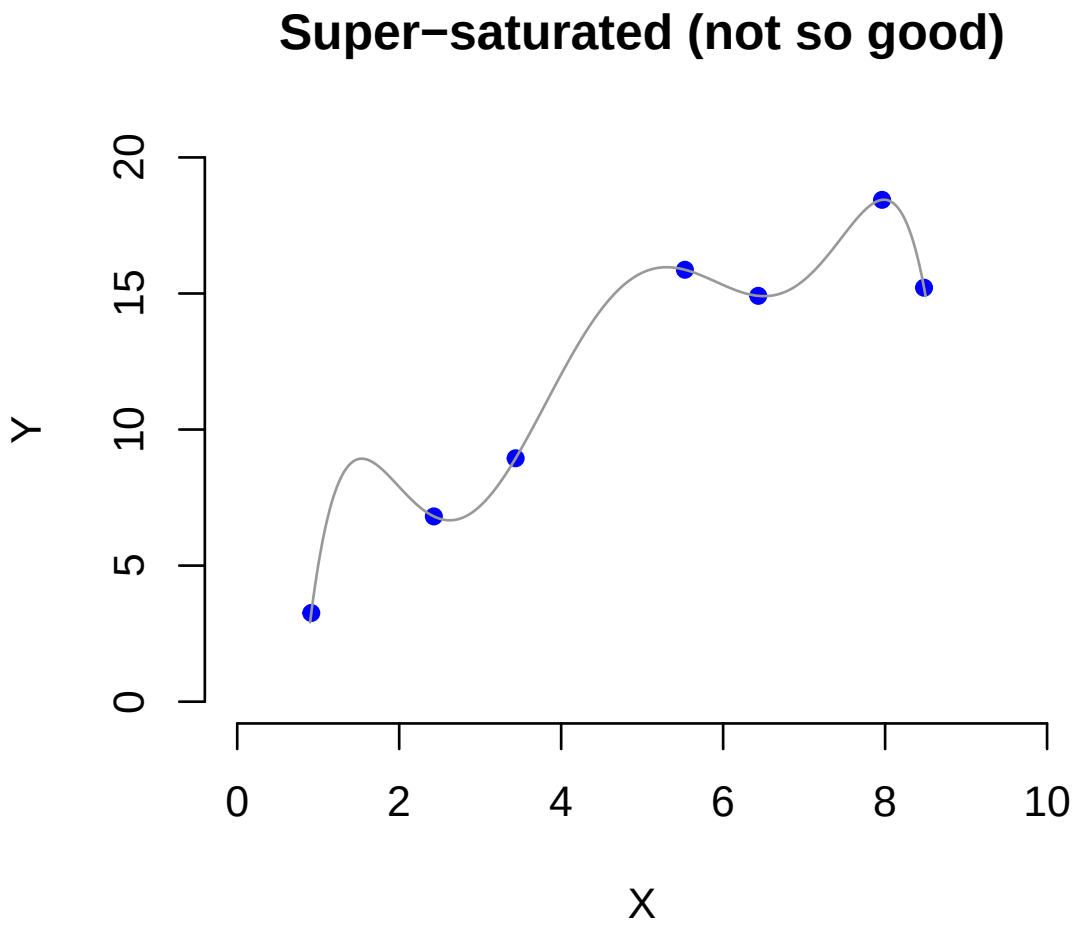
```
## Set seed for reproducibility
set.seed(7345)

## Generate data
n <- 7
x <- rnorm(n, 5, 3)
y <- 6 + 0.5*x + rnorm(n, 0, 7)
```

Linear models: Polynomial interpolation

```
## Polynomial regression
zz <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6))

> summary(zz)

Call:
lm(formula = y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6))

Residuals:
ALL 7 residuals are 0: no residual degrees of freedom!

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -60.5965        NaN     NaN      NaN
x            138.5121        NaN     NaN      NaN
I(x^2)      -103.8012        NaN     NaN      NaN
I(x^3)        37.0178        NaN     NaN      NaN
I(x^4)        -6.7027        NaN     NaN      NaN
I(x^5)         0.5969        NaN     NaN      NaN
I(x^6)        -0.0208        NaN     NaN      NaN

Residual standard error: NaN on 0 degrees of freedom
Multiple R-squared:     1,  Adjusted R-squared:   NaN
F-statistic:  NaN on 6 and 0 DF,  p-value: NA

## Sad trombone noises
```

Saturation and super-saturation

Linear models: Polynomial interpolation



Saturation and super-saturation

Tabular data:

- Suppose I have the following data, represented by a 2×2 table:

	$X_1 = 0$	$X_1 = 1$
$X_2 = 0$	a	b
$X_2 = 1$	c	d

- We could convert the data to a binary-outcome representation and fit, say, a simple logistic regression model.
  - Sample size: $N = a + b + c + d$.
  - Such a model, having $K = 2$ degrees of freedom, will perfectly predict the relative proportions as seen in the table (no concerns).
- I'd like to challenge you to think about what might happen if we were to conceptualize the cell values as originating from Poisson counts.
- Sample size: $N = 4$?
- Such a model, having $K = 4$ degrees of freedom, will perfectly predict the exact *counts* seen in the table. Problem?

Tabular data:

```
## Set seed for reproducibility
set.seed(7345)

## Two-by-two tabular data
n <- 4

## Columns
x1 <- c(0,0, 1,1)

## Rows
x2 <- c(0,1, 0,1)

## Counts
y <- rpois(n, lambda = exp(1 + x1 + x2 + x1*x2))

tab <- data.frame(matrix(y, nrow = 2))
names(tab) <- c("x1=0", "x1=1")
row.names(tab) <- c("x2=0", "x2=1")

> tab
     x1=0 x1=1
x2=0    4   11
x2=1    2   48

> cbind(x1, x2, y)
     x1 x2  y
[1,]  0  0  4
[2,]  0  1  2
[3,]  1  0 11
[4,]  1  1 48
```

Tabular data:

```
## Model
zz <- glm(y ~ x1 + x2 + I(x1*x2), family = poisson(link = "log"))

> ## Should recover [1,1], [1,2], [2,1], and [2,2] entries perfectly
> as.numeric(exp(coef(zz)[1]))
[1] 4

> as.numeric(exp(coef(zz)[1] + coef(zz)[2]))
[1] 11

> as.numeric(exp(coef(zz)[1] + coef(zz)[3]))
[1] 2

> as.numeric(exp(coef(zz)[1] + coef(zz)[2] + coef(zz)[3] + coef(zz)[4])
    )
[1] 48

## Looks like a ``super-saturated'' model.
```

Saturation and super-saturation

Tabular data:

```
> summary(zz)

Call:
glm(formula = y ~ x1 + x2 + I(x1 * x2), family = poisson(link = "log"))

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.386      0.500    2.77   0.0056 **
x1             1.012      0.584    1.73   0.0832 .
x2            -0.693      0.866   -0.80   0.4235
I(x1 * x2)     2.166      0.928    2.33   0.0196 *
---
(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 7.5800e+01  on 3  degrees of freedom
Residual deviance: 4.6629e-15  on 0  degrees of freedom
AIC: 23.84

Number of Fisher Scoring iterations: 3

## No problem...surprisingly?
```

Saturation and super-saturation

Tabular data:

- Super-saturation does not appear to pose a problem in this case.
  - There is no nuisance parameter.
  - Mean-variance relationship: $V(\mu) = \mu$; $\phi = 1$.
- Poisson standard errors rely on mean-variance relationship.
  - Analogue to OLS: if assuming $\sigma^2 = 1$ (bad), super-saturated linear model would have no problem.
- Recall that information about the nuisance comes from the *residuals*. If the residuals are all zero, so too will be the estimated nuisance.
- On the flip side: residuals are also where you assess assumptions...
  - Cannot use the data to assess the validity of the Poisson model in a "super-saturated" analysis of tabular data.

Table of Contents

Fixed vs. random design matrices

Considerations:

- In this set of notes, we are treating GLMs as a likelihood-based approach, and we are assuming that the models put forth are correctly specified.
- It is mathematically convenient to think of $\mathbf{X}$ as fixed and known in advance (that's what we have done this entire slide set).
- However, when the models are correctly specified, this makes no meaningful difference as key quantities described in these notes (such as $\mathbf{D}^\top\mathbf{V}^{-1}\mathbf{D}$) are consistent for their their respective expectations over the marginal distribution of $\mathbf{X}$.
- We will cover various forms of GLM misspecification, at which point we will actually have to pay attention to whether $\mathbf{X}$ is fixed or random and use the correct approach.

Summary

This unit:

- Generalized linear models.
- The canonical link.
- The Newton-Raphson and Gauss-Newton algorithms.
- Asymptotics.
- Offsets.
- Saturation.

Summary: So far

- Random vectors and matrices; multivariate normal theory.
- Ordinary least squares.
- Hypothesis testing and ANOVA.
- Weighted least squares.
- Misspecification.
- Confidence regions and prediction.
- Diagnostics.
- Regularization.
- Bayesian regression.
- Exponential families.
- Generalized linear models.

Summary: Coming up

- Sandwich and bootstrap.
- Quasi-likelihood.
- Hypothesis testing for GLMs.
- Diagnostics for GLMs.
- Further considerations for binary outcomes.
- Nonlinear least squares.