Week 2: Generalized Linear Models

PLSC 40502 - Statistical Models

Review

Previously

- **Likelihood**: $\mathcal{L}(\theta \mid \mathbf{Y}) \propto f(\mathbf{Y} \mid \theta)$
  - Function of θ (not itself a probability density)
  - Comparisons of likelihoods across different parameters capture notions of model "fit"

$$\lambda(\theta_1, \theta_2) = \frac{\mathcal{L}(\theta_1 \mid \mathbf{Y})}{\mathcal{L}(\theta_2 \mid \mathbf{Y})}$$

- **Frequentist** inference: Use the likelihood to find a "good" estimator for θ_0: the MLE $\hat{\theta}$
  - $\hat{\theta}$ is consistent for θ_0
  - It's asymptotically normal
  - It's (asymptotic) variance is the inverse Fisher Information

This week

- **Generalized linear models**
  - What happens when $\mathrm{E}[Y_i \mid X_i] \neq X_i'\beta$?
  - Can we retain the linear form but relate it to a function of the CEF?
- **Types of GLMs**
  - Binary outcome models (e.g. logistic)
  - Ordinal/Multinomial outcome models
  - Count outcome models (e.g. Poisson)
  - Duration models (e.g. exponential)
- **Robust** inference
  - What happens when the GLM outcome distributions are wrong?
  - Can we still do valid inference for the CEF?

Intro to GLMs

Motivation: Propensity Scores

- Researchers wanting to estimate causal effects from observational designs often use a *weighting* estimator to account for non-random treatment assignment.
  - Observe treatment D_i, confounders X_i
  - Need to estimate $Pr(D_i = 1 \mid X_i)$ to construct "inverse propensity of treatment weights"
- **Example**: Keriakes et. al. (2000) *"Abciximab provides cost-effective survival advantage in high-volume interventional practice"*
  - Abciximab, an anti-clotting drug, is often used during certain types of heart surgery to reduce bleeding risk.
  - Keriakes et. al. (2000) look at 1472 surgeries in Ohio Heart Health Center
  - Abciximab was administered *non-randomly* -- some types of patients more likely to receive the drug than others
- **Key problem** - With many continuous covariates, hard to estimate $Pr(D_i = 1 \mid X_i)$ non-parametrically
  - One solution: Assume a parametric *model* for D_i

Generalized Linear Models

- Generalized linear models (GLMs) have three components:

  1. A parametric distribution on $Y_i | X_i$ ("stochastic component")
  2. A linear predictor: $\eta_i = X_i^{'}\beta = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots \beta_k X_{ik}$ ("systematic component")
  3. A link function $g()$ applied to the CEF $E[Y_i | X_i]$ that yields the linear predictor

$$g(E[Y_i | X_i]) = \eta_i$$

Alternatively, we can write the CEF in terms of the "inverse-link" $g^{-1}()$ applied to the linear predictor

$$E[Y_i | X_i] = g^{-1}(\eta_i)$$

Exponential Family

- The types of probability distributions permitted on Y_i are quite general: the **exponential family**
  - This contains the **normal** as well as many other common distributions like the bernoulli, Poisson, exponential, etc...
- Exponential family distributions have density functions of the form

$$P(y \mid \theta) = h(y)\exp\left\{b(\theta) \cdot T(y) - A(\theta)\right\}$$

where $h(y)$, $A(\theta)$, $\eta(\theta)$ and $T(y)$ are known functions

- The key intuition: exponential distributions factorize in a convenient way
  - if $b(\theta) = \theta$, then the distribution is in "canonical" form
  - $T(y)$ is a "sufficient statistic"

Example: Bernoulli

- Consider the bernoulli PMF

$$P(y_i \mid \pi_i) = \pi_i^{y_i}(1 - \pi_i)^{1-y_i}$$

- Take the log, then the exponent

$$P(y_i \mid \pi_i) = \exp\left\{\log\left[\pi_i^{y_i}(1 - \pi_i)^{1-y_i}\right]\right\}$$

- Properties of logs

$$P(y_i \mid \pi_i) = \exp\left\{y_i \log(\pi_i) + (1 - y_i)\log(1 - \pi_i)\right\}$$

Example: Bernoulli

- Rearranging and using properties of logs again

$$P(y_i \mid \pi_i) = \exp\left\{y_i \log\left(\frac{\pi_i}{1-\pi_i}\right) + \log(1-\pi_i)\right\}$$

- So our exponential form is
  - $h(y) = 1$
  - $T(y) = y_i$
  - $A(\theta) = \log(1-\pi_i)$
  - $b(\theta) = \log\left(\frac{\pi_i}{1-\pi_i}\right)$
- Critically, this is where we get a good link function
  - The "canonical parameter" is $\log\left(\frac{\pi}{1-\pi}\right)$.
  - The "canonical link" is the function that equates this parameter with the linear predictor

    $X_i'\beta = \log\left(\frac{\pi_i}{1-\pi_i}\right)$

Logistic regression

- The "logit" or "logistic" GLM models the **log-odds** of a binary outcome as a function of the linear predictor $X_i'\beta$

$$Y_i \sim \text{i.i.d}\text{Bernoulli}(\pi_i)$$

$$E[Y_i \,|\, X_i] = Pr(Y_i = 1 \,|\, X_i) = \pi_i$$

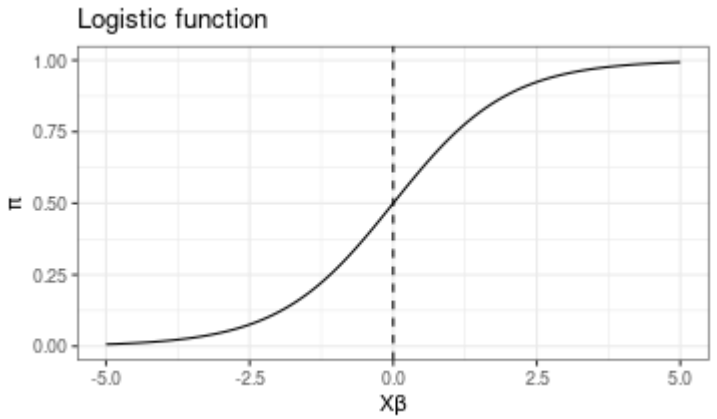
$$\log\left(\frac{\pi}{1-\pi}\right) = X_i'\beta$$

- Alternatively, this is written in terms of the "inverse-link" function (the logistic function) that relates π_i to $g^{-1}(X_i'\beta)$

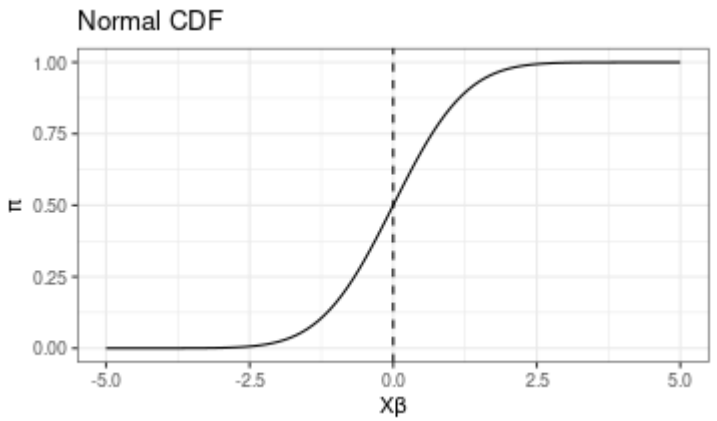
$$\pi_i = \frac{\exp(X_i'\beta)}{1+\exp(X_i'\beta)} = \frac{1}{1+\exp(-X_i'\beta)}$$

Inverse-link functions

- The logistic function maps inputs on $\mathbb{R}$ to $(0, 1)$

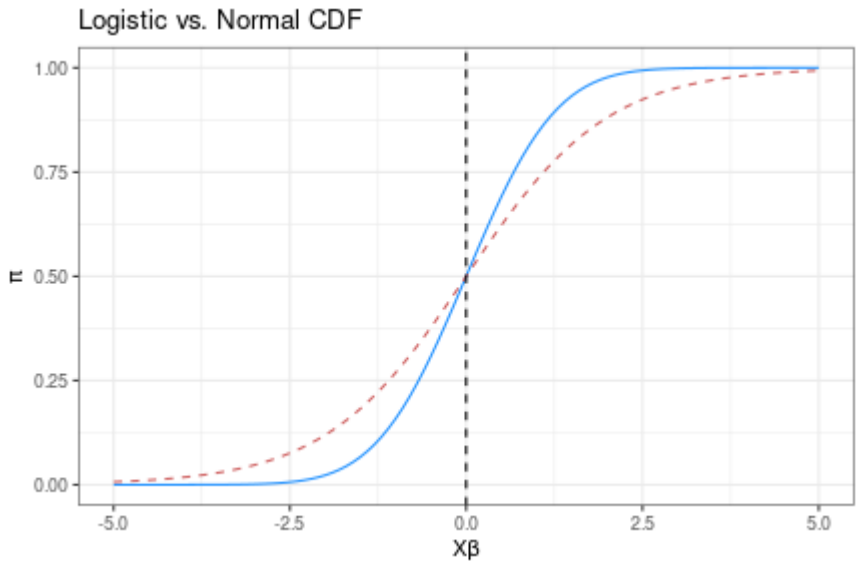


- Another link is the "probit" whose inverse link is the Normal CDF



Inverse-link functions

- When do we use probit vs. logit?
  - Computational convenience!
  - Probit has some good properties for Bayesian inference
- Can you tell the difference?



Estimation

- We obtain an estimate of β, $\hat{\beta}$ using maximum likelihood.
- Our MLE estimator is:

$$\hat{\beta} = \arg\max_{\beta} \log f(\mathbf{y} \mid \beta, \mathbf{X})$$

- Recall our score function is:

$$S(\beta) = \nabla \log f(\mathbf{y} \mid \beta, \mathbf{X}) = \begin{bmatrix} \frac{\partial}{\partial \beta_0} \log f(\mathbf{y} \mid \beta, \mathbf{X}) \\ \frac{\partial}{\partial \beta_1} \log f(\mathbf{y} \mid \beta, \mathbf{X}) \\ \vdots \\ \frac{\partial}{\partial \beta_k} \log f(\mathbf{y} \mid \beta, \mathbf{X}) \end{bmatrix}$$

Newton-Raphson

- An iterative algorithm starts at some initial guess $\hat{\beta}^{(0)}$
- Let $\hat{\beta}^{(t)}$ denote the "current" value of $\hat{\beta}$ and $\hat{\beta}^{(t+1)}$ our update -- we keep iterating until convergence.
- Our goal is to solve for a zero of $S(\beta)$
  - Let's do a first-order Taylor approximation around our current guess $S(\hat{\beta}^{(t)})$

$$S(\beta) \approx S(\hat{\beta}^{(t)}) + \nabla S(\hat{\beta}^{(t)})\left(\beta - \hat{\beta}^{(t)}\right)$$

- What's $\nabla S(\hat{\beta}^{(t)})$?
  - It's the Jacobian of the gradient...or the matrix of second-order partial derivatives of the log-likelihood...or the Hessian!
  - Denote it $\mathbf{H}(\hat{\beta}^{(t)})$

Newton-Raphson

- Our next value of $\hat{\beta}$ is the value of β that sets the score equal to zero

$$0 = S(\hat{\beta}^{(t)}) + \mathbf{H}(\hat{\beta}^{(t)})\left(\hat{\beta}^{(t+1)} - \hat{\beta}^{(t)}\right)$$

- Multiply through by the inverse hessian

$$\hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \mathbf{H}^{-1}(\hat{\beta}^{(t)})S(\hat{\beta}^{(t)})$$

- Recall that the negative inverse hessian is also the **Observed Fisher Information**
  - An alternative algorithm, **Fisher Scoring** substitutes this for the expected Fisher Information
  - Often these update steps can be expressed as solutions to a **weighted least squares** optimization problem
- All of these and more are implemented in the `maxLik` R package, which we will be using this week.
  - Generally more "current" than `optim()` and includes some convenience functions.

Example: Propensity Scores

- Let's estimate a logistic propensity score model for treatment in the Keriakes et. al. (2000) dataset

```
# Read in the dataset
pci <- read_csv("data/pci.csv")
```

- We want to predict treatment: `abcix` using a mix of discrete and continuous covariates...
  - `stent` - Coronary stent deployment; binary indicator
  - `height` - Height in centimeters; numeric integer
  - `female` - Female gender; binary indicator
  - `diabetic` - Diabetes mellitus diagnosis; binary indicator
  - `acutemi` - Acute myocardial infarction within the previous 7 days; binary indicator
  - `ejecfrac` - Left ejection fraction; numeric integer
  - `ves1proc` - Number of vessels involved in the patient's initial PCI procedure; numeric integer

Example: Propensity Scores

- Make the design matrix $\mathbf{X}$

```
X_mat <- model.matrix(abcix ~ stent + height + female + diabetic + acutemi + ejecfrac + ves1prc
head(X_mat) # View the top of the matrix
```

```
##   (Intercept) stent height female diabetic acutemi ejecfrac ves1proc
## 1           1     0    163      1        1       0       56        1
## 2           1     0    168      0        0       0       56        1
## 3           1     0    188      0        0       0       50        1
## 4           1     0    175      0        1       0       50        1
## 5           1     0    168      1        0       0       55        1
## 6           1     0    178      0        0       0       50        1
```

- Sample size

```
n_obs <- nrow(X_mat)
n_obs
```

```
## [1] 996
```

Example: Propensity Scores

- From our logit link function, we have

$$\log\left(\frac{\pi}{1-\pi}\right) = X_i'\beta$$

and

$$1 - \pi_i = 1 - \frac{1}{1+\exp(-X_i'\beta)} = \frac{\exp(-X_i'\beta)}{1+\exp(-X_i'\beta)} = \frac{1}{1+\exp(X_i'\beta)}$$

- Let's write the log-likelihood:

$$\ell(\beta \mid \mathbf{y}, \mathbf{X}) = \sum_{i=1}^{N} y_i X_i'\beta + \log\left(\frac{1}{1+\exp(X_i'\beta)}\right)$$

$$\ell(\beta \mid \mathbf{y}, \mathbf{X}) = \sum_{i=1}^{N} y_i X_i'\beta - \log\left(1+\exp(X_i'\beta)\right)$$

Example: Propensity Scores

- And let's put the log-likelihood into code (this returns a vector of the log-likelihood for each observation)

```
logit_loglik <- function(beta, Y, X){
  eta <- X%*%beta # linear predictor
  lik <- Y*eta - log(1+exp(eta))
  return(lik)
}
```

- Now let's optimize it to get the MLE

```
library(maxLik) # Maximum Likelihood library

logit_mle <- maxLik(logit_loglik,
                    Y=pci$abcix,
                    X=X_mat,
                    start = rep(0, ncol(X_mat)),
                    method = "NR")
```

Example: Propensity Scores

- What did we get?

```
est <- coef(logit_mle) # Our optimization routine
names(est) <- colnames(X_mat)
est
```

```
## (Intercept)       stent      height      female    diabetic     acutemi
##      2.9658      0.5730     -0.0154     -0.3591     -0.4068      1.1995
##    ejecfrac    ves1proc
##     -0.0148      0.7605
```

```
# Compare to built-in R routine?
logit_Rglm <- glm(abcix ~ stent + height + female + diabetic + acutemi + ejecfrac + ves1proc,
                  data=pci, family=binomial(link="logit"))
coef(logit_Rglm)
```

```
## (Intercept)       stent      height      female    diabetic     acutemi
##      2.9657      0.5730     -0.0154     -0.3591     -0.4068      1.1995
##    ejecfrac    ves1proc
##     -0.0148      0.7605
```

Example: Propensity Scores

- Let's obtain our (asymptotic) variance-covariance matrix

```
logit_vcov <- solve(-hessian(logit_mle))
```

- Square root of the diagonal is our SEs

```
logit_SEs <- sqrt(diag(logit_vcov))
```

Example: Propensity Scores

- Let's get our t-statistics and p-values

```
results <- rbind(coef(logit_mle), logit_SEs,
                 coef(logit_mle)/logit_SEs,
                 2*pnorm(-abs(coef(logit_mle)/logit_SEs)))
colnames(results) <- colnames(X_mat)
rownames(results) <- c("Estimate", "Std. Error", "Test statistic", "p-value")
results
```

```
##                (Intercept)    stent  height female diabetic  acutemi ejecfrac
## Estimate             2.966 0.573015 -0.0154 -0.359  -0.4068 1.20e+00 -0.01479
## Std. Error           2.282 0.150510  0.0124  0.237   0.1707 2.72e-01  0.00751
## Test statistic       1.300 3.807156 -1.2378 -1.516  -2.3825 4.42e+00 -1.96844
## p-value              0.194 0.000141  0.2158  0.129   0.0172 1.01e-05  0.04902
##                ves1proc
## Estimate       7.61e-01
## Std. Error     1.39e-01
## Test statistic 5.48e+00
## p-value        4.22e-08
```

Interpreting logit coefficients

- How do we interpret the βs substantively?

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_k X_{ik}$$

- Take the partial derivative w.r.t. X_{i1}

$$\frac{\partial}{\partial X_{i1}} \log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_1$$

- So β_k captures the change in the log-odds for a one-unit change in X_k
  - Descriptively, it's the difference in log-odds between two observations that differ in X_k by one unit.

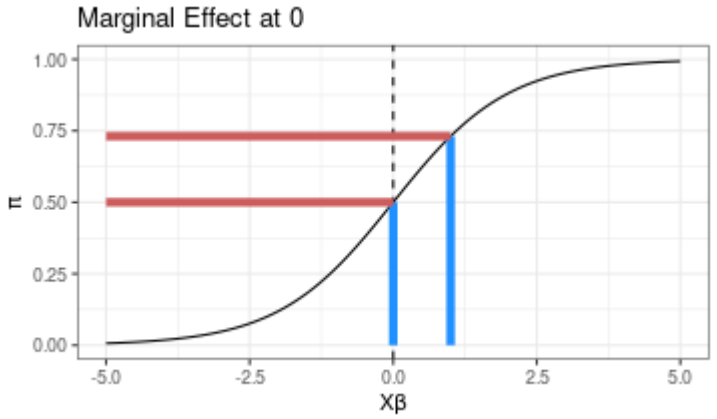
Interpreting logit coefficients

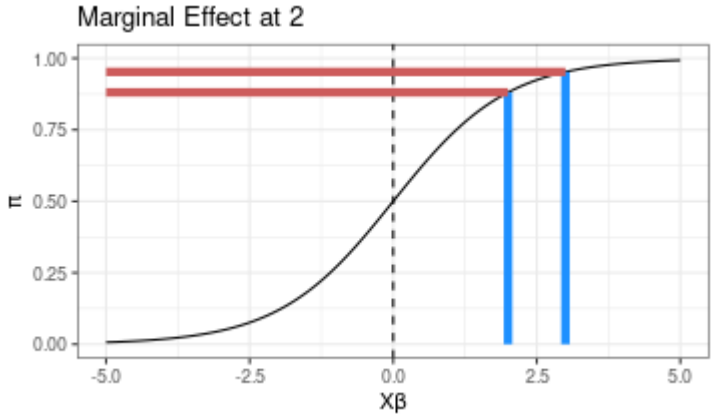
- On the "log-odds" scale, the change due to an increase in X_k does not depend on the values of the other X variables (unless we explicitly specify an interaction).
  - But thinking on the log-odds scale is hard! We think in terms of probabilities.
  - This additivity *does not hold* when we take $\frac{\partial}{\partial X_{i1}} \pi_i$
- Logit models *implicitly* encode interactions with respect to the CEF $\mathbf{E}[Y_i \mid X_i]$

Interpreting logit coefficients

- **Remember**: A one-unit change in the linear predictor corresponds to *different* changes in probability depending on your baseline.

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
```





Transformed quantities

- We have $\hat{\beta}$, but in the propensity score case, we really want $\hat{\pi}_i = Pr(D_i = 1 \mid X_i)$
  - Just apply the inverse link to get the quantity we want

$$\hat{\pi}_i = \frac{1}{1 + \exp(-X_i'\hat{\beta})}$$

- The function of the MLEs is the MLE of the function
  - So $\hat{\pi}_i$ is consistent for the true propensity scores (under our modeling assumptions)
- But what if we want to do inference on $\hat{\pi}_i$ or obtain a confidence interval?
  - How do we obtain $\text{Var}(\hat{\pi}_i) = \text{Var}(g(\hat{\beta}))$?

Delta method

- We know $Var(\hat{\beta})$
  - Asymptotically, it's the inverse Fisher Information or the inverse negative Hessian of the log-likelihood.
- To get $Var(g(\hat{\beta}))$, let's start with a first-order Taylor approximation around the true value β

$$g(\hat{\beta}) \approx g(\beta) + [\nabla g(\beta)'](\hat{\beta} - \beta)$$

- Take the variance

$$Var(g(\hat{\beta})) \approx Var\Big(g(\beta) + [\nabla g(\beta)'](\hat{\beta} - \beta)\Big)$$

- Expand the sum

$$Var(g(\hat{\beta})) \approx Var\Big(g(\beta) + [\nabla g(\beta)']\hat{\beta} - [\nabla g(\beta)']\beta\Big)$$

Delta method

- Variance of a constant is zero

$$Var(g(\hat{\beta})) \approx Var\left([\nabla g(\beta)']\hat{\beta}\right)$$

- Pull out and "square" the constant. We now get an expression in terms of our original variance-covariance matrix

$$Var(g(\hat{\beta})) \approx [\nabla g(\beta)']Var(\hat{\beta})[\nabla g(\beta)]$$

- This approximation is actually exact asymptotically (the higher-order terms of the Taylor polynomial go to zero)
  - Use the usual plug-in estimator for the the gradient at the MLE

Example: Propensity Score

- Let's determine the propensity of receiving treatment for a patient at the median covariate values

```
X_medians <- apply(X_mat, 2, median)
X_medians
```

```
## (Intercept)       stent      height      female    diabetic     acutemi
##           1           1         173           0           0           0
##    ejecfrac    ves1proc
##          55           1
```

- Construct our prediction function

```
pred_prob <- function(beta, X){
  return(1/(1 + exp(-X%*%beta)))
}
pred_median <- pred_prob(coef(logit_mle), X_medians)
pred_median
```

```
##       [,1]
## [1,] 0.696
```

Example: Propensity Score

- We could solve for the gradient in terms of β analytically, but there are plenty of convenient functions that will do this numerically

```
pred_prob_gradient <- numericGradient(pred_prob,
                                      t0 = coef(logit_mle),
                                      X = X_medians)
```

- Applying the delta method

```
pred_prob_var <- pred_prob_gradient%*%logit_vcov%*%t(pred_prob_gradient) # R treats vectors as
```

- Making our 95% asymptotic CI for $\hat{\pi}(X)$

```
c(pred_median - abs(qnorm(.025))*sqrt(pred_prob_var),
  pred_median + abs(qnorm(.025))*sqrt(pred_prob_var))
```

```
## [1] 0.643 0.749
```

"Monte Carlo" Delta Method

- Alternatively, we could approximate the (asymptotic) sampling distribution of $\hat{\pi}$ by:
  1. Sampling from the known asymptotic distribution of $\hat{\beta}$
  2. Passing each sampled β through to our function $\pi = g(\beta)$
  3. Taking the variance of the simulated πs
- With many independent samples, this will get arbitrarily close to the true sampling variance of $\hat{\pi}$
  - King, Tomz and Wittenberg (2000) is essentially this idea
- Note that this is **not** bootstrapping.
  - We're using our existing estimator of $Var(\hat{\beta})$
  - Rather it's doing a "monte carlo" simulation instead of the delta method -- no need to take derivatives!

"Monte Carlo" Delta Method

- Let's try it:

```
set.seed(60637)
sim_betas <- MASS::mvrnorm(n=1e5, mu = coef(logit_mle), Sigma = logit_vcov)
sim_pi <- apply(sim_betas, 1, function(x) pred_prob(x, X=X_medians))
```

- How close are we?

```
c(pred_median - abs(qnorm(.025))*sd(sim_pi),
  pred_median + abs(qnorm(.025))*sd(sim_pi))
```

```
## [1] 0.643 0.749
```

"Latent variables" and logit

Latent variables

- Another common way of formulating discrete outcome regressions is in terms of an unobserved continuous "latent" variable and an observation mechanism.
  - Instead of putting a distribution on Y_i, we put one on an unobserved Y_i^*
  - Y_i is some function of Y_i^* (usually an indicator function of some sort)
- For the logistic regression

$$Y_i = 1(Y_i^* \geq 0)$$

$$Y_i^* = X_i'\beta + \epsilon_i$$

$$\epsilon_i \sim \text{Logistic}(0, 1)$$

- The Logistic distribution is parameterized in terms of a "location" (mean) μ and a "scale" parameter $s > 0$.
  - For $\mu = 0$, $s = 1$ we have the "standard" logistic

$$P(\epsilon_i < x) = \frac{1}{1 + \exp(-x)}$$

Latent variables

- So what's $E[Y_i | X_i] = Pr(Y_i = 1 | X_i)$?

$$Pr(Y_i = 1 | X_i) = Pr(Y_i^* \geq 0) = Pr(\epsilon_i < X_i'\beta) = \frac{1}{1 + \exp(-X_i'\beta)}$$

- We can write the **probit** similarly, defining the error distribution as

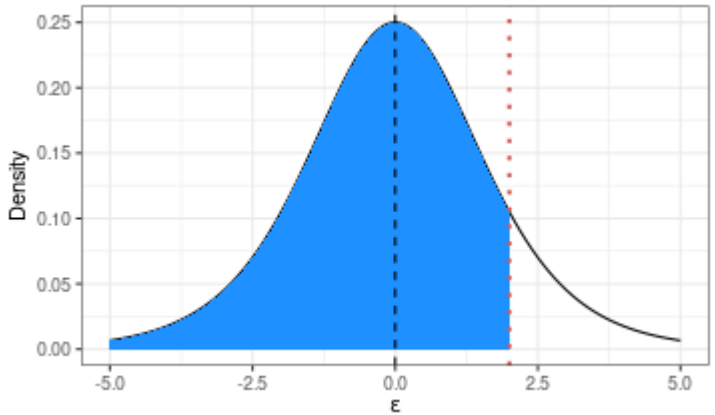
$$\epsilon_i \sim \text{Normal}(0, 1)$$

- And by extension, the probability $Pr(Y_i | X_i)$

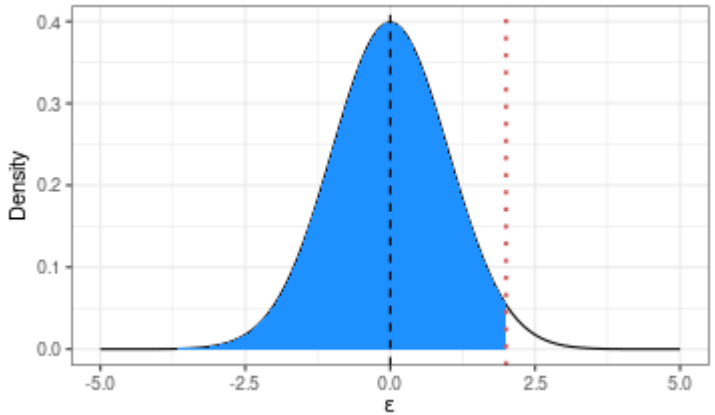
$$Pr(Y_i = 1 | X_i) = Pr(Y_i^* \geq 0) = Pr(\epsilon_i < X_i'\beta) = \Phi(X_i'\beta)$$

Latent variables

- We can visualize this as an integral over the density of the random "error" ϵ_i



- Same thing for a probit



Ordinal logit

- Using the latent variable formulation lets us extend the binary outcomes to logit to outcomes with multiple discrete levels.
- For example, consider an "ordinal" response variable
  - (e.g.) a likert scale (1 = "strongly disagree", 3 = "neutral", 5 = "strongly agree")
- The responses are *ordered* but the intervals between them do not have any meaning.
  - A popular modeling technique with an ordinal Y is to treat it as an expression of a latent variable
- In an *ordered logit* model, an ordinal outcome Y with L unique, ordered values $Y \in \{1, 2, 3, \ldots L\}$ is a function of a latent variable and a set of $L - 1$ cutpoints κ

$$Y_i = \begin{cases} 1 & \text{if } Y_i^* \leq \kappa_1 \\ 2 & \text{if } \kappa_1 < Y_i^* \leq \kappa_2 \\ 3 & \text{if } \kappa_2 < Y_i^* \leq \kappa_3 \\ \vdots & \\ L & \text{if } Y_i^* > \kappa_{L-1} \end{cases}$$

Ordinal logit

- To estimate the model, we now have to estimate both the β parameters and the $L-1$ κ parameters
  - Note we omit the intercept from our X_i -- the κ parameters are essentially the "intercepts" for each boundary between the choices
- How does the model map changes in the latent variable to changes in probability?

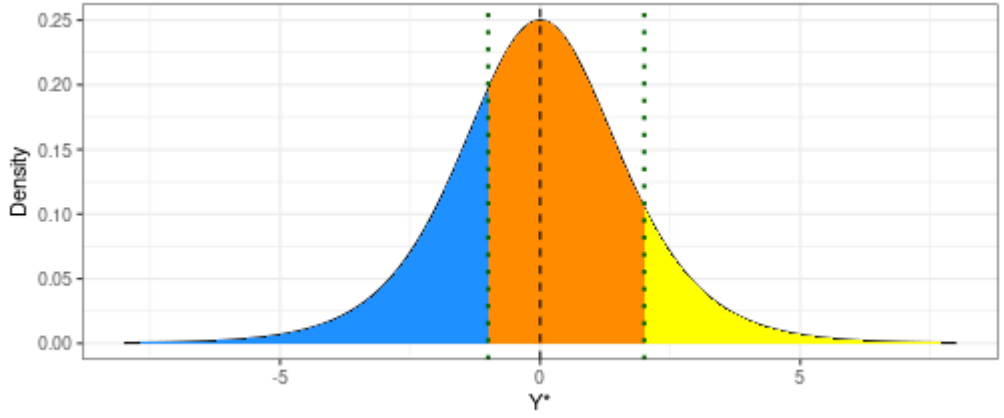
$$Pr(Y_i = 1) = Pr(Y_i^* \leq \kappa_1) = Pr(\epsilon_i \leq \kappa_1 - X_i'\beta)$$

$$Pr(Y_i = 2) = Pr(Y_i = 2 \cup Y_i = 1) - Pr(Y_i = 1) = Pr(Y_i^* \leq \kappa_2) - Pr(Y_i^* \leq \kappa_1)$$

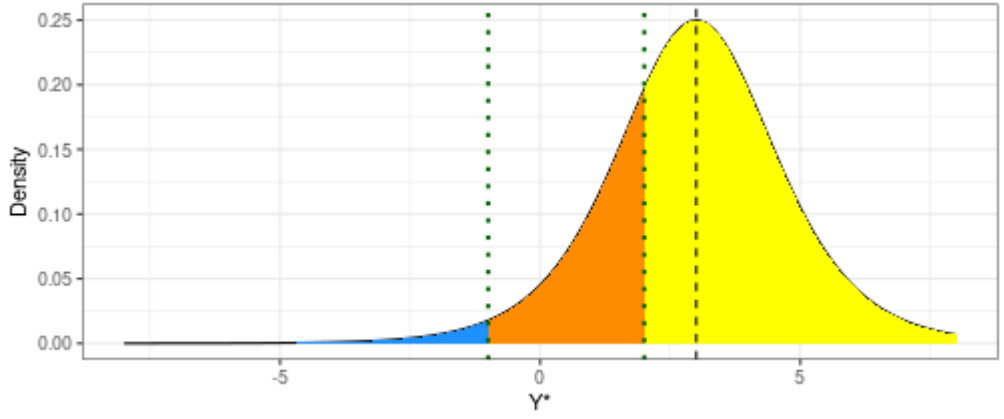
$$Pr(Y_i = 3) = Pr(Y_i = 3 \cup Y_i = 2 \cup Y_i = 1) - Pr(Y_i = 2 \cup Y_i = 1) = Pr(Y_i^* \leq \kappa_3) - Pr(Y_i^* \leq \kappa_2)$$

Ordinal logit

- Let's think in terms of the distribution of Y_i^* and the cut-points for a case with $L = 3$. For $X_i'\beta = 0$:



- Shifting the mean of Y^* up by 3:



Ordinal logit

- This structural mapping from Y^* to $P(Y_i = l)$ is not trivial - it encodes a particular assumption about how these probabilities relate to one another.
- Referred to as the **proportional odds** assumption

$$\log\left(\frac{Pr(Y_i \leq l)}{1 - Pr(Y_i \leq l)}\right) = \kappa_l - X_i'\beta$$

- **Intuitively** -- The β parameters are shared across all the possible outcome levels. Information about whether someone responds "strongly agree" *also* tells us something about how likely they would be to respond specifically to "agree", "neutral", "disagree", and "strongly disagree"
  - Not trivial if there's non-monotonicity or heterogeneous effects.
  - Be careful with these models!

"Robust" inference and misspecification

Misspecification

- Often we'll encounter settings where we only believe *part* of the maximum likelihood model
- Perhaps that we've correctly specified $g(E[Y_i|X_i])$, but we don't believe the complete distributional assumptions we've placed on the outcome
  - We don't believe the observations are "**identically** distributed"
- Or we believe that some outcomes are correlated with one another even after conditioning on the covariates
  - We don't believe they're **independently** distributed.
- What can we do?
  - Suppose our MLE is still **consistent** for something we're interested in (e.g. the CEF).
  - But the outcome model is incorrect (e.g. due to heteroskedasticity) and so our variance estimator is mis-specified.
  - Can we still estimate the variance of our MLE when it's *not* efficient? Yes!

"Robust" Standard Errors

- Suppose we obtain a maximum likelihood estimate based on maximizing the log-likelihood

$$\ell(\theta \mid \mathbf{y}) = \sum_{i=1}^{n} \log f_i(y_i \mid \theta)$$

- Note that the density here is being indexed by i
  - We're relaxing the "identically" distributed assumption -- we're leaving the full distribution of $Y_i \mid \theta$ unspecified
- Denote the score

$$\ell'(\theta \mid \mathbf{y}) = \sum_{i=1}^{n} \frac{\partial}{\partial \theta} \log f_i(Y_i \mid \theta) = \sum_{i=1}^{n} g_i(Y_i \mid \theta)$$

- And the Hessian

$$\ell''(\theta \mid \mathbf{y}) = \sum_{i=1}^{n} \frac{\partial^2}{\partial \theta^2} \log f_i(Y_i \mid \theta) = \sum_{i=1}^{n} h_i(Y_i \mid \theta)$$

"Robust" Standard Errors

- Assume that there is a true θ_0
- Let's do a Taylor approximation of the likelihood around that true value

$$\ell(\theta \,|\, \mathbf{y}) = \ell(\theta_0 \,|\, \mathbf{y}) + \ell'(\theta_0 \,|\, \mathbf{y})(\theta - \theta_0) + \frac{1}{2}(\theta - \theta_0)'\ell''(\theta_0 \,|\, \mathbf{y})(\theta - \theta_0)$$

- The MLE $\hat{\theta}$ is the solution that maximizes that likelihood function.
  - So we know it satisfies the equation that sets the score equal to 0. Let's take the derivative of the above w.r.t θ and solve for 0

$$0 = \ell'(\theta_0 \,|\, \mathbf{y}) + (\hat{\theta} - \theta_0)'\ell''(\theta_0 \,|\, \mathbf{y})$$

- Solve for $\hat{\theta}$

$$(\hat{\theta} - \theta_0) = [-\ell''(\theta_0 \,|\, \mathbf{y})]^{-1}\ell'(\theta_0 \,|\, \mathbf{y})$$

"Robust" Standard Errors

- Plug in our definitions from before

$$(\hat{\theta} - \theta) = \left(- \sum_{i=1}^{n} h_i(Y_i | \theta_0) \right)^{-1} \left(\sum_{i=1}^{n} g_i(Y_i | \theta_0) \right)$$

- Multiplying by 1 and using properties of inverses

$$(\hat{\theta} - \theta) = \left(- \frac{1}{n} \sum_{i=1}^{n} h_i(Y_i | \theta_0) \right)^{-1} \left(\frac{1}{n} \sum_{i=1}^{n} g_i(Y_i | \theta_0) \right)$$

- We make two asymptotic arguments
  - The first term converges in probability to the Fisher information
  - The second term times $\sqrt{n}$ converges in distribution to a normal distribution with mean 0 and variance $Var(g_i(Y_i | \theta_0))$

"Robust" Standard Errors

- So asymptotically, we have

$$\sqrt{n}(\hat{\theta} - \theta) \rightarrow d\left[\mathbf{E}[-\ell''(\theta_0 \mid \mathbf{y})]\right]^{-1}\mathcal{N}\left(0, Var(g_i(Y_i \mid \theta_0))\right)$$

- And therefore, the asymptotic variance of $\hat{\theta}$ is

$$Var(\hat{\theta}) = \left[\mathbf{E}[-\ell''(\theta_0 \mid \mathbf{y})]\right]^{-1}\left[Var(g_i(Y_i \mid \theta_0))\right]\left[\mathbf{E}[-\ell''(\theta_0 \mid \mathbf{y})]\right]^{-1}$$

"Robust" Standard Errors

- We'll use our usual plug-in estimator for the Hessian at the MLE
  - But earlier, we showed that under the model, the middle term *also* equals the expected negative hessian (and therefore cancels with one of the inverses).
  - Instead, we'll use a different plug-in estimator using the outer product of the gradient of each observation at the MLE.

$$\hat{Var}(\hat{\theta}) = \left[-\sum_{i=1}^{n} h_i(Y_i \mid \hat{\theta}) \right]^{-1} \left[\sum_{i=1}^{n} g_i(Y_i \mid \hat{\theta}) g_i(Y_i \mid \hat{\theta})' \right] \left[-\sum_{i=1}^{n} h_i(Y_i \mid \hat{\theta}) \right]^{-1}$$

The "Sandwich" Estimator

- We call this the "sandwich" estimator because it consists of a "meat" between two slices of identical "bread"

$$\hat{Var}(\hat{\theta}) = \underbrace{\left[-\sum_{i=1}^{n} h_i(Y_i \mid \hat{\theta}) \right]^{-1}}_{\text{bread}} \underbrace{\left[\sum_{i=1}^{n} g_i(Y_i \mid \hat{\theta}) g_i(Y_i \mid \hat{\theta})' \right]}_{\text{meat}} \underbrace{\left[-\sum_{i=1}^{n} h_i(Y_i \mid \hat{\theta}) \right]^{-1}}_{\text{bread}}$$

- These estimators are **consistent** for the asymptotic variance under mis-specification of the outcome distribution
  - But finite-sample properties can be not great -- typically we get *undercoverage* in small samples.
  - Typically add a finite-sample correction (e.g. $\frac{N}{N-K}$)

The regression "sandwich"

- A familiar form of the sandwich arises from linear regression.
- Remember our OLS estimator

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

- We wrote this as

$$\hat{\beta} = \beta + (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\epsilon)$$

- And took the variance

$$Var(\hat{\beta}) = Var\left((\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\epsilon)\right)$$

The regression "sandwich"

- We're doing inference conditional on $\mathbf{X}$

$$Var(\hat{\beta}) = Var\left((\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\epsilon)\right)$$

$$Var(\hat{\beta}) = (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'Var(\epsilon)\mathbf{X})(\mathbf{X}'\mathbf{X})^{-1}$$

- $Var(\epsilon)$ is the variance-covariance matrix of the errors.
  - Since the errors are mean zero, it's the expectation of the outer-product of the errors $Var(\epsilon) = E[\epsilon\epsilon']$
  - Under our Gauss-Markov assumptions (specifically, independence and homoskedasticity), $Var(\epsilon) = \sigma^2\mathbf{I}$. And so

The regression "sandwich"

- Instead of making that assumption, we'll just plug in the observed residuals $\hat{\epsilon}$ as estimators of ϵ
  - This yields the classic "robust" variance estimator:

$$\hat{Var}(\hat{\beta}) = (\mathbf{X}'\underbrace{\mathbf{X}})^{-1}(\mathbf{X}'\underbrace{\hat{\epsilon}\hat{\epsilon}'}\mathbf{X})(\mathbf{X}'\underbrace{\mathbf{X}})^{-1}$$

bread meat bread

Implementing sandwich SEs

- Let's go back to our propensity score logistic regression of abciximab treatment on the predictive covariates

```
library(sandwich) # this contains "estfun()"

logit_sandwich <- solve(-hessian(logit_mle))%*%(t(estfun(logit_mle))%*%estfun(logit_mle))%*%sol
```

- Compare the robust SEs to the ones we had earlier

```
# Non-robust
logit_SEs
```

```
## [1] 2.28201 0.15051 0.01242 0.23683 0.17075 0.27166 0.00751 0.13874
```

```
# Robust
logit_sandwich_SEs <- sqrt(diag(logit_sandwich))
logit_sandwich_SEs
```

```
## [1] 3.22525 0.14992 0.01736 0.29404 0.16964 0.26910 0.00773 0.14285
```

Implementing sandwich SEs

- These are also implemented in the `sandwich` package which operates on a lot of common MLE packages in R

```
logit_sandwich_vcov <- sandwich::sandwich(logit_mle)
sqrt(diag(logit_sandwich_vcov))
```

```
## [1] 3.22525 0.14992 0.01736 0.29404 0.16964 0.26910 0.00773 0.14285
```

```
logit_sandwich_SEs
```

```
## [1] 3.22525 0.14992 0.01736 0.29404 0.16964 0.26910 0.00773 0.14285
```

The Bootstrap

- An alternative approach to estimating the sampling variance of any estimator via monte carlo methods is the "bootstrap"
- **Intuitively**: Bootstrap methods use the *empirical* distribution of some random variable to estimate its unknown *theoretical* distribution
  - Use **monte carlo** methods to approximate that empirical distribution.
- For regression, it is common to use the **pairs bootstrap** to obtain a heteroskedasticity-robust variance estimator (Freedman, 1981)
  1. Start with the original sample $(Y_1, X_1), (Y_2, X_2), \ldots, (Y_n, X_n)$
  2. Generate a new "bootstrapped" sample of size n: $(Y_1^*, X_1^*), (Y_2^*, X_2^*), \ldots, (Y_n^*, X_n^*)$ where the probability $Pr[(Y_j^*, X_j^*) = (Y_i, X_i)] = \frac{1}{n}$ for each i and j (sampling *with* replacement)
  3. Compute $\hat{\beta}^*$ using the boostrapped sample $(Y_1^*, X_1^*), (Y_2^*, X_2^*), \ldots, (Y_n^*, X_n^*)$
  4. Repeat 2-3 a large number of times B to obtain a sample of bootstrapped estimates $\hat{\beta}^{(1)}, \hat{\beta}^{(2)}, \ldots, \hat{\beta}^{(B)}$
  5. Use the sample moments of the bootstrapped estimates to estimate the quantities of interest (e.g. standard error; percentile intervals)

Implementing the Bootstrap

- Let's do a pairs/resampling bootstrap

```
set.seed(60637)
nIter <- 1000
beta_boot <- matrix(nrow=nIter, ncol=ncol(X_mat))
for (i in 1:nIter){
  boot_rows <- sample(1:nrow(X_mat), replace=T)#Resample rows
  mle_boot <- maxLik(logit_loglik,
                     Y=pci$abcix[boot_rows],
                     X=X_mat[boot_rows,],
                     start = rep(0, ncol(X_mat)),
                     method = "NR")
  beta_boot[i,] <- coef(mle_boot)
}
```

Implementing the Bootstrap

- Compare our bootstrap variance-covariance matrix to the sandwich estimator

```
# Sandwich
logit_sandwich_SEs
```

```
## [1] 3.22525 0.14992 0.01736 0.29404 0.16964 0.26910 0.00773 0.14285
```

```
# Bootstrap
boot_SEs <- sqrt(diag(var(beta_boot)))
boot_SEs
```

```
## [1] 1.91626 0.15008 0.01039 0.21919 0.16876 0.28782 0.00757 0.14705
```

Survival models

Survival models

- Often we have outcome data that is measured as the "time-to-event"
  - From the medical literature, this is often the number of days before a patient suffers some adverse event (e.g. death).
  - But lots of other applications (e.g. time to treaty ratification)
- Assume we observe N i.i.d. event times $T_1, T_2, \ldots, T_n$
- Define the "survival function"

$$S(t) = Pr(T > t) = 1 - F(t)$$

- We'll also often work with the "hazard" or the "instantaneous" probability of an event happening *given* that it has not yet occurred

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t < T_i \leq t + \Delta t \,|\, T_i > t)}{\Delta t} = \frac{f(t)}{S(t)} = \frac{f(t)}{1 - F(t)}$$

- **Intuitively**: Given that I know that an event has not occurred until t, what is the probability that it will immediately occur next!

- We'll also define the cumulative hazard

$$H(t) = \int_0^t h(u)du = \int_0^t \frac{f(t)}{S(t)}du = -\log(S(t))$$

Non-parametric approach

- The easiest way to estimate $S(t) = 1 - F(t)$ is to just plug in $\hat{F}(t)$ -- our empirical CDF in the sample
  - At any given value t, how many events occur *after* t
- **Problem**: This doesn't address *censoring* of observations -- sometimes we stop observing a unit at time t^* rather than observing an actual event.
  - We know that $T_i > t^*$ but not what T_i actually equals
- Let $C = \{C_1, C_2, \ldots, C_n\}$ denote the **censoring indicator**
  - If $C_i = 1$, then the observation is *censored*, the observed value t only gives us information on $T_i > t$ as opposed to $T_i = t$.
  - Crucially, we assume that censoring is "uninformative"
- One approach to constructing a non-parametric estimator of the survival function *under censoring* is the **Kaplan-Meier** estimator.

Kaplan-Meier Estimator

- Start by arranging the possible event times from $0 < t_{(1)} < t_{(2)} < t_{(3)} < t_{(4)}, \ldots, < t_{(n)}$
  - Suppose t is after $t_{(j)}$ and before $t_{(j+1)}$. We can write the survival function in terms of conditional probabilities:

$$S(t) = Pr(T_i > t \,|\, T_i > t_{(j)}) \times Pr(T_i > t_{(j)} \,|\, T_i > t_{(j-1)}) \times \ldots \times Pr(T_i > t_{(2)} \,|\, T_i > t_{(1)}) \times Pr(T_i > t_{(1)})$$

- The **Kaplan-Meier** estimator $\hat{S}(t)$ can be understood as a "plug-in" estimator for each of these probabilities.

$$\hat{S}(t) = \prod_{j : t_{(j)} \leq t} \left(1 - \frac{d_j}{r_j}\right)$$

- d_j denotes the number of units that experience an event at time $t_{(j)}$
- r_j is the number of units that are still in the risk set up to time j

Kaplan-Meier Estimator

- **Intuition**: Consider one of the terms in the product. Using the definition of conditional probability:

$$Pr(T_i > t_{(j)} \mid T_i > t_{(j-1)}) = 1 - Pr(T_i < t_{(j)} \mid T_i > t_{j-1}) = 1 - \frac{Pr(T_i \leq t_{(j)}, T_i > t_{(j-1)})}{Pr(T_i > t_{(j-1)})}$$

- In the numerator, our "plug-in" estimator is just the share of units that have events occurring at $t_{(j)}$.
- In the denominator we have the share of units that we know for sure are still "active" in the interval after $t_{(j-1)}$
- Hence our plug-in estimator:

$$\hat{Pr}(T_i > t_{(j)} \mid T_i > t_{(j-1)}) = 1 - \frac{d_j}{r_j}$$

- Taking the product across all of the j where $t_{(j)} < t$ yields the Kaplan-Meier estimator.

Illustration: Lung Cancer

- Let's look at a sample dataset from the `survival` R package on survival times for patients with advanced lung cancer.

```
library(survival)
data(cancer, package="survival")
as_tibble(lung)
```

```
## # A tibble: 228 × 10
##     inst  time status   age   sex ph.ecog ph.karno pat.karno meal.cal wt.loss
##    <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>    <dbl>     <dbl>    <dbl>   <dbl>
##  1     3   306      2    74     1       1       90       100     1175      NA
##  2     3   455      2    68     1       0       90        90     1225      15
##  3     3  1010      1    56     1       0       90        90       NA      15
##  4     5   210      2    57     1       1       90        60     1150      11
##  5     1   883      2    60     1       0      100        90       NA       0
##  6    12  1022      1    74     1       1       50        80      513       0
##  7     7   310      2    68     2       2       70        60      384      10
##  8    11   361      2    71     2       2       60        80      538       1
##  9     1   218      2    53     1       1       70        80      825      16
## 10     7   166      2    61     1       2       70        70      271      34
## # … with 218 more rows
```

- We want to estimate $S(t \mid \text{Sex} = \text{Male})$ and $S(t \mid \text{Sex} = \text{Female})$

Illustration: Lung Cancer

- In the `survival` package, we need to define a "survival" outcome in terms of the event time and the censoring indicator

```
surv_model <- survfit(Surv(time, status) ~ 1, data=lung)
```

Illustration: Lung Cancer

- Plot the curve for the sample as a whole

```
plot(surv_model, xlab="Survival Time", conf.int=T, ylab="Survival Rate", mark.time=TRUE)
```

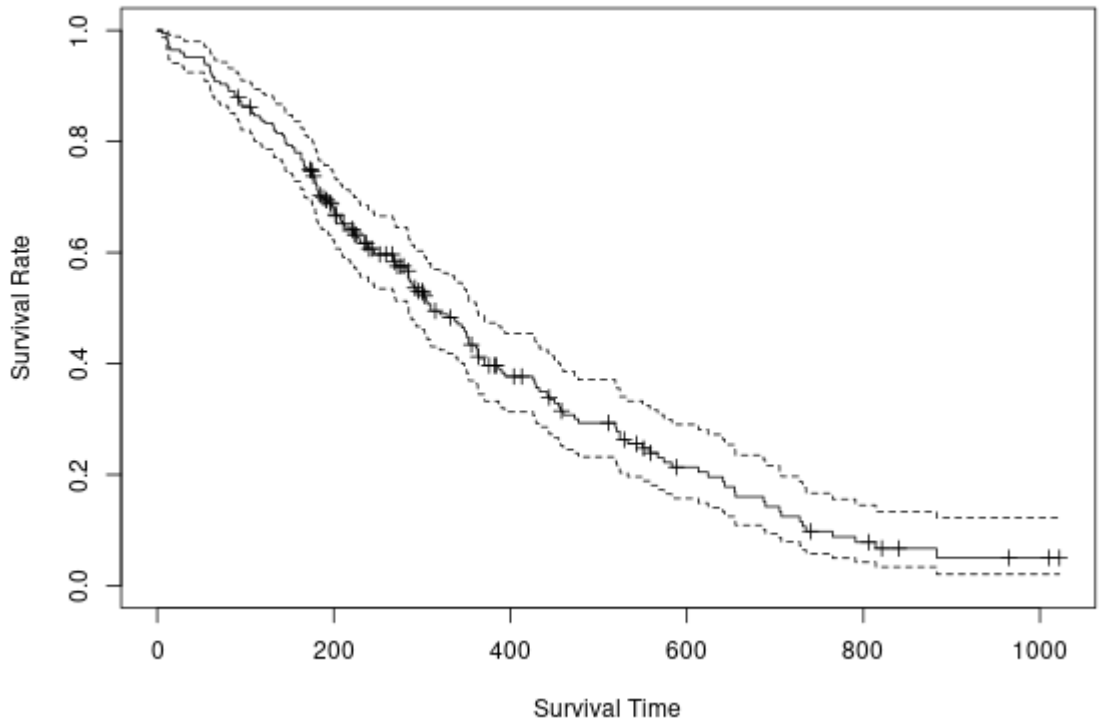


Illustration: Lung Cancer

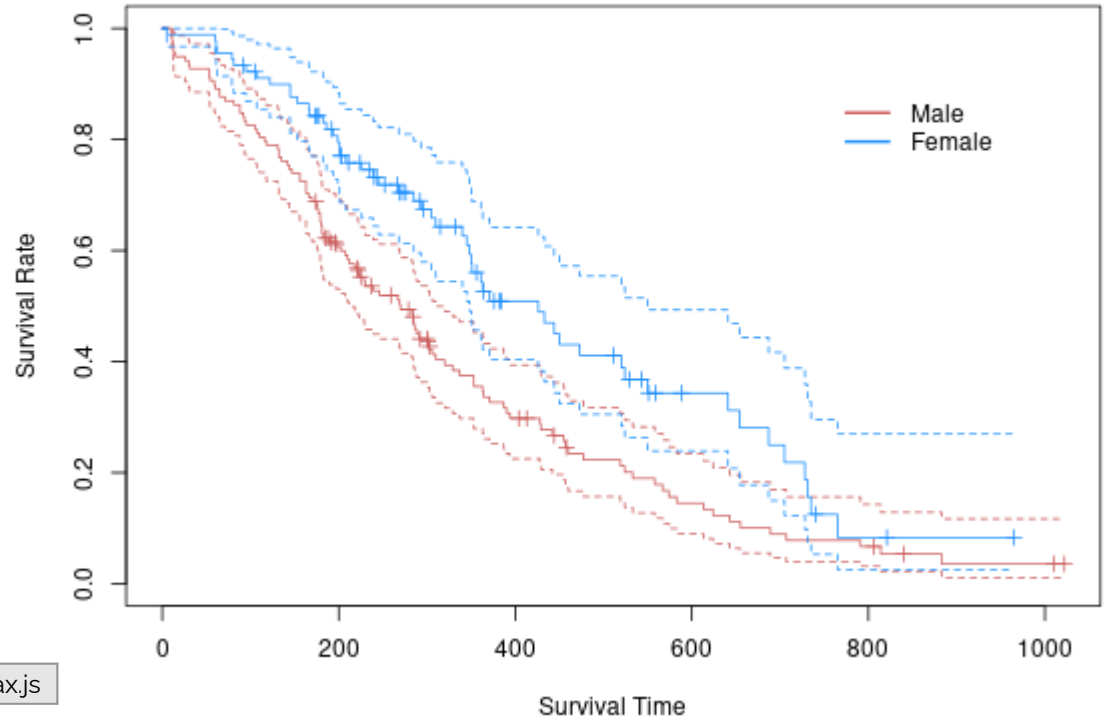
- Now let's do this for gender

```
surv_model_sex <- survfit(Surv(time, status) ~ I(sex==2), data=lung)
```

Illustration: Lung Cancer

- Plot separate curves for men and women

```
plot(surv_model_sex, col=c("indianred", "dodgerblue"), xlab="Survival Time", conf.int=T, ylab='
legend(750, .9, c("Male", "Female"), col=c("indianred", "dodgerblue"), lwd=2, bty='n')
```



Parametric inference

- When we want to model the survival function using covariates, we run into a clear constraint using purely non-parametric methods -- with continuous covariates, we may only have one observation that has a unique covariate value
  - Splitting the sample and constructing separate K-M curves quickly becomes infeasible.
- We may want to assume a **model** for the event time. We can write that density in terms of the *hazard* and the *survival* function

$$f(t) = h(t)S(t)$$

- Note that by the definition of the cumulative hazard $H(t) = -\log(S(t))$ and $S(t) = \exp(-H(t))$, so

$$f(t) = h(t)\exp(-H(t))$$

- Depending on the assumptions we make about the nature of the hazard, we can obtain different distributions for $f(t)$.
  - In the simplest case, lets assume that the hazard does not vary over time: $h(t) = \lambda. \lambda > 0$
  - And the cumulative hazard is $H(t) = \int_0^t \lambda du = \lambda t$
- This yields the **exponential distribution**

$$f(t) = \lambda\exp(-\lambda t)$$

Exponential Model

- The mean of $T_i \sim \text{Exponential}(\lambda)$ is $E[T_i] = \frac{1}{\lambda}$. We'll often reparameterize the mean as θ, yielding a density of

$$f(t) = \frac{1}{\theta}\exp\left(-\frac{1}{\theta}t\right)$$

- θ is strictly greater than 0, so we'll use a log-link to incorporate the covariates:

$$\log(E[Y_i|X_i] = \log(\theta_i) = X_i'\beta$$

- Alternatively, we can write

$$E[Y_i|X_i] = \exp(X_i'\beta)$$

- Note that the constant hazard assumption is a strong one -- the exponential distribution is "memoryless" in that knowing that a unit has survived up until time t tells you nothing about its instantenous probability of failure.
  - More flexible distributions like the Weibull allow for a time-varying hazard.

Exponential Model

- We can interpret changes in the linear predictor in terms of changes in the hazard since

$$h(t) = \frac{1}{\theta_i} = \frac{1}{\exp(X_i'\beta)}$$

- Increases in the linear predictor reduce the hazard.
- This also lets us write down the survival function

$$S(t) = \exp\left(- \frac{1}{\exp(X_i'\beta)} t \right)$$

Application

- Let's compare our parametric fit to the non-parametric Kaplan-Meier estimates

```
exp_model_sex <- survreg(Surv(time, status) ~ I(sex==2), dist="exponential", data=lung) # sex =
```

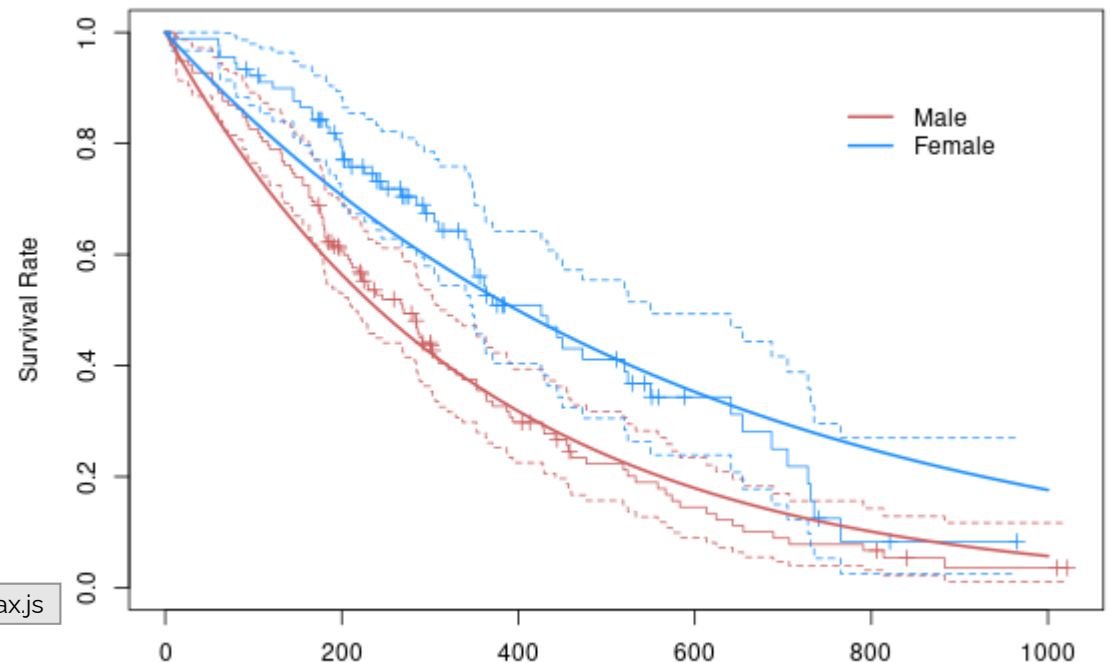
- Get the survival curves for men and women

```
exp_surv_men <- function(x) pexp(x, rate = 1/exp(coef(exp_model_sex)%*%c(1,0)), lower.tail = F)
exp_surv_women <- function(x) pexp(x, rate = 1/exp(coef(exp_model_sex)%*%c(1,1)), lower.tail=F)
```

Application

- Overlay the exponential survival curves on the Kaplan-Meier estimates

```
plot(surv_model_sex, col=c("indianred", "dodgerblue"), xlab="Survival Time", conf.int=T, ylab='
legend(750, .9, c("Male", "Female"), col=c("indianred", "dodgerblue"), lwd=2, bty='n')
curve(exp_surv_men, from = 0, to = 1000, lwd=2, col="indianred", add=T)
curve(exp_surv_women, from = 0, to = 1000, lwd=2, col="dodgerblue", add=T)
```



Semi-parametric

- You'll notice that the fit is okay, but there is some discrepancy
  - This is because the hazard appears to be **time-varying**.
  - You can see a kind of "s-curve" which the exponential by construction cannot capture.
- Other parametric models allow for more flexibility in the hazard rate's evolution over time
  - For example, the **Weibull** model, which adds an additional shape parameter to the hazard to allow it to vary over time actually fits the data quite well!
- But suppose that we want to make very minimal assumptions about the structure of the hazard function -- can we still estimate the relationship between covariates and the hazard?

Cox Proportional Hazards

- The **Cox proportional hazards model** assumes that covariates enter the hazard log-linearly, but the baseline hazard is unspecified

$$h_i(t) = h_0(t)\exp(X_i'\beta)$$

- $h_0(t)$ is the unknown baseline hazard function.
- Note that while the baseline hazard is not explicitly identified, the ratio of the hazards of two observations i and j at time t **does not depend on the baseline**

$$\frac{h_i(t)}{h_j(t)} = \frac{\exp(X_i'\beta)}{\exp(X_j'\beta)}$$

- The model is **semi-parametric** in that some components are left entirely unspecified...
  - ... but others are not -- we assume that the covariates act **proportionally** on the hazard irrespective of time.

Cox Proportional Hazards

- The model is estimated by maximizing a **partial likelihood**
  - We observe event time Y_i - assume no ties
- The individual partial likelihood is

$$L_i(\beta) = \frac{\exp(X_i'\beta)}{\sum_{k;\, Y_k \geq Y_i} \exp(X_k'\beta)}$$

- **Intuitively**: The denominator contains the hazards of all observations in the **risk set** at the event time Y_i

- Then the full likelihood is just the product over all the individual partial likelihoods

$$L(\beta) = \prod_{i=1}^{N} \frac{\exp(X_i'\beta)}{\sum_{k;\, Y_k \geq Y_i} \exp(X_k'\beta)}$$

Application

- What does the Cox fit look like

```
cox_model_sex <- coxph(Surv(time, status) ~ I(sex==2), data=lung) # sex = 2 is a dummy variable
print(cox_model_sex)
```

```
## Call:
## coxph(formula = Surv(time, status) ~ I(sex == 2), data = lung)
##
##                 coef exp(coef) se(coef)  z     p
## I(sex == 2)TRUE -0.5       0.6      0.2 -3 0.001
##
## Likelihood ratio test=11  on 1 df, p=0.001
## n= 228, number of events= 165
```

- Note that we **don't get** an estimate of the hazard because the baseline is left unspecified
  - If we want to plot survival curves, we'll need some estimate of h_0 using another method (like Kaplan-Meier)

Application

- Calculate the baseline hazard via Kaplan-Meier and overlay the Cox predicted hazards for men and women

```
plot(survfit(cox_model_sex, newdata = data.frame(sex=c(1,2))), col=c("indianred", "dodgerblue"))
```