Precept Three: Numerical Optimization and Simulation to Derive QOI from Model Estimates

Rebecca Johnson

February 23rd, 2017

Outline

- Logistics/replication
- Extending steps in maximum likelihood estimation from Precept/Problem set 2:
  1. Write out the model.
  2. Calculate the likelihood ($\mathrm{L}(\theta|y)$) for all observations.
     - Using exponential distribution, move from $\theta = \lambda$ the same for all observations to λ_i
  3. Take the log of the likelihood ($\ell(\theta|\mathbf{Y})$).
  4. Plug in the systematic component for θ_i.
  5. Bring in observed data.
  6. Maximize $\ell(\theta|y)$ with respect to θ and confirm that this is a maximum.
     - Previously: analytic optimization or optimize; this week: numerical optimization using optim
  7. Find the variance of your estimate.
  8. Using simulation to go from estimates to QOI

Logistics/replication

- Pset 1: graded copies over Blackboard or email by Friday night
- Pset 3: due next Wednesday
- Upcoming replication deadlines:
  - Wednesday March 1st: provide proof of data to us; we give you 10 extra points on pset 3
  - Wednesday March 8th (no pset due): Memo 1, main goal is to replicate main tables and figures
- Some advice on contacting authors for data/code for those who haven't yet started

Some (preemptive) Pset 3 clarifications

- In general: personal grading pref to use .rmd we provide so easy to link questions with your response
- Problem 1B: Aesthetic choice to use separate Greek latter for coefficient of interest- *does not* affect estimation:

$$dge_{ct} = seg'_{ct}\beta + x'_{ct}\gamma + u_{ct}$$

- Problem 2: "we're going to use `optim` to optimize the likelihood w.r.t β and σ^2"; w.r.t = with respect to

Notecard question: how are things connected under the same framework?

To help illustrate, will walk through an example that:

1. Begins with the distribution from Pset 2 Recessions problem: exponential with λ parameter
2. Uses the story of Jack (Leonardo DiCaprio) and Rose (Kate Winslet) from Titanic to motivate *linking* λ to covariates like passenger social class and gender
3. After adding the covariates, we need to make two shifts:
   - 3.1 Shift from analytic optimization to numerical optimization
   - 3.2 Within numerical optimization, minor programming shift from `optimize` to `optim` in R
4. After we use optimization to find values for β for predictors of interest like Titanic passenger race and social class, how to use simulation to generate and visualize meaningful QOI from those estimates

Example Jack and Rose from Titanic: same iceberg, different time-to-death outcomes

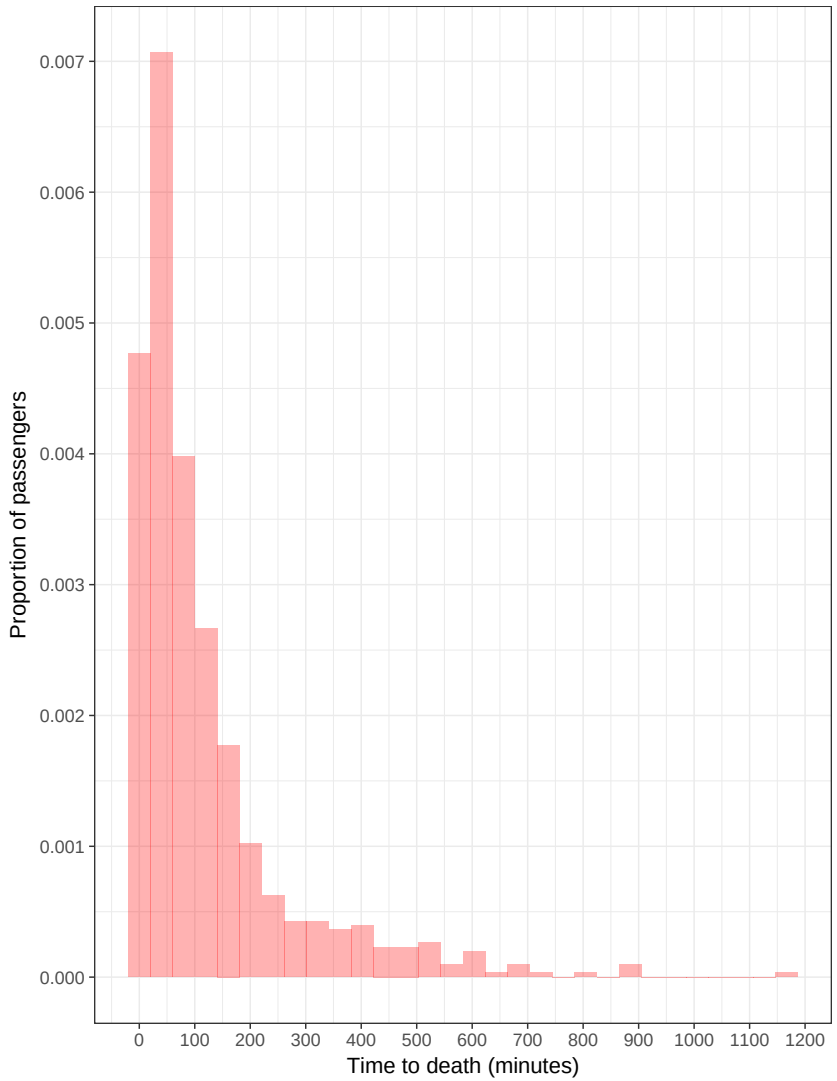
- Jack: age 20 male passenger in steerage (3rd class in data)
- In movie/our example: died from hypothermia roughly 60 minutes after impact

- Rose: age 17 female passenger in 1st class
- In movie: survived Titanic, died at age 101
- In our example: ignoring right censoring for now (will discuss more in Lecture 4 slides), so assume she died about 200 minutes after impact

In precept one, we practiced telling a story about the data generating process (DGP) behind a particular outcome

- Outcome in this case: we're interested in the *time until something happens* (the passenger dies)
- What's the story?:
  - Many passengers die really quickly from the collision or drowning, with a tail end of those who die from hypothermia
  - As discussed on the homework, the process is memoryless- in this example, the expected time until the next passenger dies does not depend on the time that's elapsed since the previous passenger died (might be violated by things like a certain passenger falling off a lifeboat and dying, making it a longer time until the next passenger dies because she took his spot)
- After telling this story, can look through our binder full of distributions (promise this is the last election reference of this precept...) and choose one that characterizes this DGP well

Before poring through this binder, can take a look at the observed time to death



Distribution we settle on: exponential distribution

- Probability density function is the following:

$$f(y) = \lambda e^{-\lambda y}$$

- Then, if we assume that this is independent across passengers (e.g., Rose's time to death is independent from Jack's time to death), we can index the time to death outcome (y) by passenger i, multiply across passengers, and write the likelihood as:

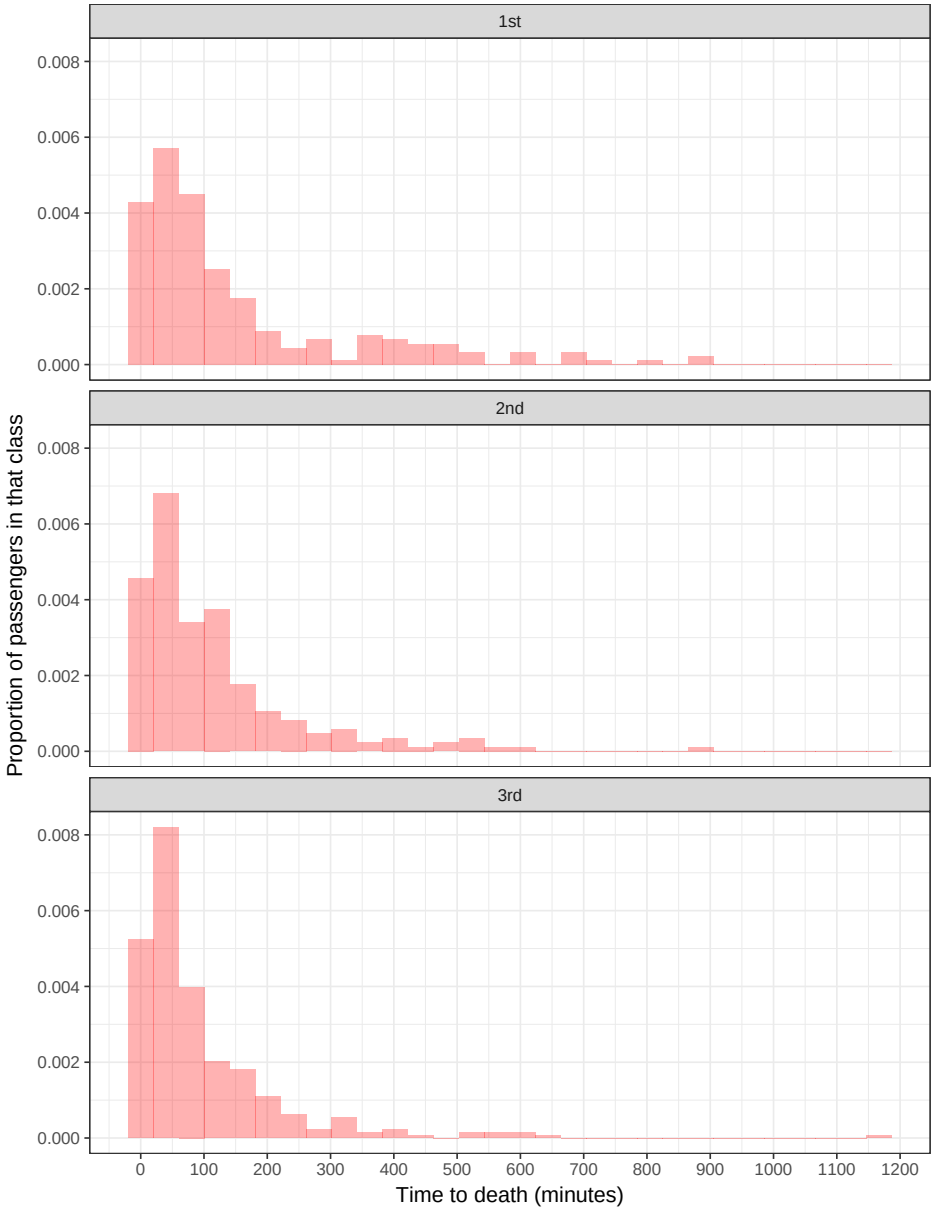
$$L(\lambda | y_{Jack}, y_{Rose}, \ldots y_n) = \prod_{i=1}^{n} \lambda e^{-\lambda y_i}$$

What we can and can't investigate about time to death after the Titanic with this likelihood

$$L(\lambda|y_{Jack}, y_{Rose}, \dots y_n) = \prod_{i=1}^{n} \lambda e^{-\lambda y_i}$$

1. What we *can* investigate:
   - Value of λ (maximum likelihood estimate: $\hat{\lambda}$) that maximizes the probability of observing the pattern of time to death we showed a few slides ago
   - Once have $\hat{\lambda}$ can use to generate *some* QOI like Pset 2, Problem 1M– in this case, could investigate probability of another death in the next 2 minutes for instance
2. What we *can't* but might want to investigate: how does the distribution of time to death vary by gender, social class, and other covariates.
   - *Why we can't investigate with above density*: λ is not indexed by i, which $\implies$ same shape of time to death for all passengers

Linking λ to covariates: motivation



For purposes of this problem: Jack is a male from steerage who has a short time to death; Rose is a female from first class who has a long time to death. Are there more systematic patterns in the distribution of time to death by gender and social class?

Linking λ to covariates: mechanics

1. Begin with the probability density function that does not link λ to covariates:

$$f(y_i) = \lambda e^{-\lambda y_i}$$

2. Create a link between λ and covariates
   - ~~$\mu_i = X_i\beta$~~.
     - Why not? λ must be positive, so we need to choose a functional form for relating covariates to λ that regardless of what data we feed in and what β we estimate, will constrain $\lambda \geq 0$
     - Exponents to the rescue...
   - $\mu_i = e^{X_i\beta}$
     - Why? Still straightforward relationship between λ and passenger characteristics like gender and social class, but exponent constraints $\lambda \geq 0$
3. Can plug back into expression for λ_i: $\lambda_i = \frac{1}{\mu_i} = \frac{1}{e^{X_i\beta}} = e^{-X_i\beta}$

Linking λ to covariates: mechanics continued...

4. Once we've found that function linking λ to our covariates of interest (gender; social class), can plug back into the Likelihood[1]:
   - Start with likelihood:

$$L(\lambda|y_{Jack}, y_{Rose}, \ldots y_n) = \prod_{i=1}^{n} \lambda e^{-\lambda y_i}$$

   - Plug in $\lambda = \frac{1}{e^{X_i\beta}}$ to wherever λ appears (step should be familiar from normal density replacing μ with $X_i\beta$

$$L(\beta|y_{Jack}, y_{Rose}, \ldots y_n) = \prod_{i=1}^{n} e^{-X_i\beta} e^{-e^{-X_i\beta} y_i}$$

[1] Could also plug into log likelihood but that gives us too many Pset 2 answers

Linking λ to covariates: mechanics continued...

$$L(\beta|y) \propto \prod_{i=1}^{n} e^{-x_i\beta} e^{-e^{-x_i\beta} y_i}$$

5. $ln(\prod x) = \sum ln(x)$

$$\ell(\beta|y) \doteq \sum_{i}^{n} ln[e^{-x_i\beta} e^{-e^{-x_i\beta} y_i}]$$

6. $ln(e) = 1$ and $ln(ab) = ln(a) + ln(b)$

$$\ell(\beta|y) \doteq \sum_{i}^{n} -x_i\beta - e^{-x_i\beta} y_i \doteq \sum_{i}^{n} -x_i\beta - \frac{1}{e^{x_i\beta}} y_i$$

Contrast: λ with and without covariates

Without covariates (Pset 2)

1. *Outcome (time to death...y_i)*: distributed exponentially and varies between observations
2. λ: same across observations
3. *What we're solving for when we do optimization*: one $\hat{\lambda}$

With covariates (now)

1. *Outcome (time to death...y_i)*: distributed exponentially and varies between observations
2. λ: may differ between individuals based on X_i, with β indicating how each covariate plays a role in that difference
3. *What we're solving for when we do optimization*: a vector equal to length of covariates (including intercept) of $\hat{\beta}$

Focusing on with covariates, inputs and outputs to this new log-likelihood

$$\ell(\beta|y) \doteq \sum_{i}^{n} -X_i\beta - \frac{1}{e^{X_i\beta}} y_i$$

- Inputs to the machine:
  - y_i: the outcome, or every passenger's time to death in minutes
  - X_i: the predictors, or that passenger's social class, age, gender, etc...
- Output(s) of the machine:
  - $\hat{\beta}$: these maximize the (log) likelihood that a particular vector of β (conditional on the model) generated the time to death we observe. For instance, if we have (class, female, age)
    1. Possibility one: (-0.2, -0.5, 1.5)
    2. Possibility two: (-0.8, 0, 0)
    3. Etc...
  - With other models,the output of the machine might be more than one parameter: for instance, Pset 3 normal linear regression, output of machine is $\hat{\beta}$ and $\hat{\sigma^2}$

Now that we know we want to end up with a vector of $\hat{\beta}$ that relate passenger characteristics to mean time to death, how do we go about doing the optimization?

Two general methods (Lecture 3, Slide 38):

1. Analytic optimization (*example: Pset 2, Problem 1, Part 3*):
   - Start with the log likelihood
   - Take the derivative w.r.t. parameter of interest
   - Set derivative equal to zero
   - Solve for critical point(s)
   - Check that critical point is a maximum by finding second derivative, plugging in the critical point, and showing that it's negative
2. Numerical optimization: for when #1 is difficult to impossible, we'll review two aspects:
   - 2.1 Practice general procedure with Titanic data and R's `optim`
   - 2.2 Throughout, learn more about what R's `optim` is doing under the hood when we do optimization via different methods

General steps in using R for numerical optimization

1. Write a function for the log likelihood that takes in X, y, and a vector of trial or starting parameters for which you're trying to find the maximum likelihood estimate (e.g., vector of β, vector containing starting/trial values for both β and σ^2)
2. Plug that function into R's `optim` along with a vector of starting/trial values
3. The values you're most interested in `optim` returning are, where K refers to the number of parameters:
   - 3.1 K-length vector of estimates:

     `optim$par`
   - 3.2 $K \times K$ matrix of all combinations of second derivatives of estimates (the Hessian) that we use to derive standard error of estimates:

     `optim$Hessian`

Step one (programming log likelihood): practice in R

In an R or .rmd file, load the *titanic.csv* file and then:

1. Program a function for the log likelihood we derived for time to death:
   - Expression for likelihood:

$$\ell(\beta|y) \doteq \sum_{i}^{n} -X_i\beta - \frac{1}{e^{X_i\beta}} y_i \tag{1}$$

   - Arguments (can choose your own names):
     - 1.1 y: a vector of the outcome variable (*deathtime* in your data)
     - 1.2 X: a matrix of the explanatory variables in the following order: intercept, social class, gender, age
     - 1.3 *par*: a vector of trial or starting values for the parameters where length = number of parameters seeking to return (in this case 4: $\hat{\beta}$ for intercept, age, gender, class
   - Returns: a single scalar value for the log likelihood (equation 1) evaluated at the parameters you fed it in step #1.3
2. Then, practice plugging in the following sets of trial parameters + other inputs into the function:
   - Trial 1: (0 0 0 0); Trial 2: (0.5 -0.5 0.5 -0.5); Trial 3: (0.5 -0.2 0.5 -0.2)

Step one (programming log likelihood): solution

```
loglik.exp <- function(param, X, y){
  #check if X has intercept and is matrix and if  not,
  #convert to that form
  if(!all(X[, 1] == 1)){
    X <- cbind(1, X)
  }
  if(!is.matrix(X)) {
    X <- as.matrix(X)
  }
  #plug into expression for loglik
  return(sum(-X%*%param - 1/exp(X%*%param) * y))
}
```

Step one (programming log likelihood): solution continued

```
#create X, y vector,
#and trial values
X <-  titanic_deathfull %>%
      select(classCode, Age, female)
y <-  titanic_deathfull %>%
      select(deathtime)
trial1 <- rep(0, 4)
trial2 <- c(0.5, -0.5, 0.5, -0.5)
trial3 <- c(0.5, -0.2, 0.5, -0.2)

#first application (see .rmd solution for rest)
loglik.exp(param = trial1,
           X = X,
           y = y)
```

Motivation for step two (use optim to find MLE)

- Going from trial1 to trial2 to trial3, the log likelihood should be getting smaller across trials (less negative), indicating we're getting closer to the maximum
- But it's highly inefficient to keep on randomly sampling trial values, plugging those in, and then keeping track of which gets us closer to the maximum
- Instead, we need a way to *begin* with a vector of random trial values, but then use some information about what happens to the function when we plug those trial values in to improve our next set of values
- R's `optim` has a variety of built-in ways of above step, so before we use it to optimize the `loglik.exp` function we wrote, let's review broadly what it's doing under the hood

General idea behind numerical optimization

- Instead of hiking in the Adirondacks armed with eyes + tree blazes to find the valley or peak, imagine being blindfolded and exploring a mountain with two tools:
  1. Tool that tells you what direction to go in
  2. Tool that tells you how far a step to take in that direction (and your steps are large enough that you might overstep)
- Most optimization algorithms use some version of those two tools, but operationalize them in different ways

One general set of tools: gradient descent/ascent

Gradient descent (or ascent for max.) methods: use slope of curve (first derivative) at trial value in iteration k as tool

1. Find general expression for function's gradient (can think of as vector of slopes)- e.g., if have two parameters x and y:

$$\nabla f = \langle \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \rangle \tag{2}$$

2. Plugging the trial value into equation 2 tells us how steep the slope is in x and y directions
3. If we want to descend the slope towards a minimum, can subtract this gradient multiplied by a step size (usually denoted as α)[2] to get our next set of trial values since we want to get further away from value where function is highest; for max, add because we want to get closer:

$$\langle x_k, y_k \rangle = \langle x_{k-1}, y_{k-1} \rangle \pm \alpha \nabla f \langle x_{k-1}, y_{k-1} \rangle$$

4. Repeat process with $\langle x_k, y_k \rangle$ until gradient at trial value gets close to zero (max or min)

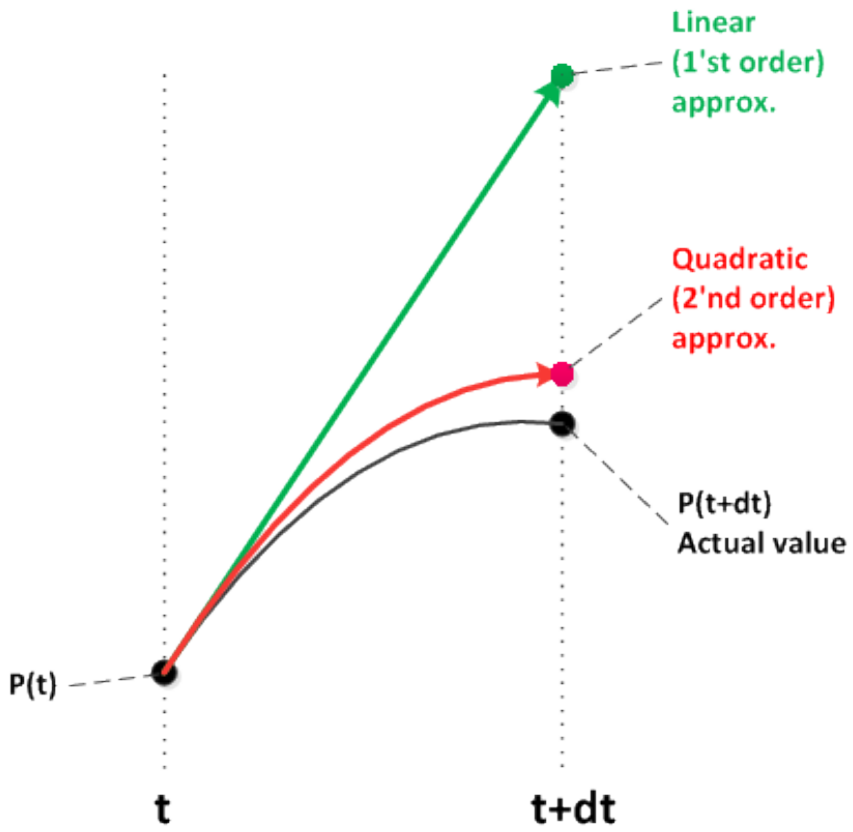
[2] We're not covering choices about fixed α versus α that updates with each iteration

Another general set of tools: Newton-Raphson and "quasi-Newton" methods

- Gradient ascent/descent used first derivative of function we seek to maximize/minimize
- Another family of approaches uses first derivative and second derivative–in multivariate case, gradient (vector) and Hessian (matrix) respectively–to update the trial values with each k iteration
  - Newton-Raphson: uses *actual* gradient and Hessian of function evaluated at trial values for each iteration
  - Quasi-Newton: for cases where the Hessian at each iteration is too computationally intensive, approximate Hessian
- Why can it be more useful to use first and second derivatives to update trial values rather than first derivatives alone? Taylor expansion as background:

$$f(x_k) \approx f(x_{k-1}) + f'(x_{k-1})(x_k - x_{k-1}) + \frac{f''(x_{k-1})}{2!}(x_k - x_{k-1})^2$$

With each higher-order derivative, get closer to true curve (illustration with univariate case)



Source: Rice University Comp 130

Using that curve approx. to get trial value closer to min/max

$$f(x_k) \approx f(x_{k-1}) + f'(x_{k-1})(x_k - x_{k-1}) + \frac{f''(x_{k-1})}{2!}(x_k - x_{k-1})^2$$

- Can then maximize this function w.r.t x_k to get next trial value:
  1. Term to max highlighted in red, take deriv and set equal to zero:

$$f(x_k) \approx f(x_{k-1}) + f'(x_{k-1})(\color{red}{x_k} - x_{k-1}) + \frac{f''(x_{k-1})}{2!}(\color{red}{x_k} - x_{k-1})^2$$

$$0 = f'(x_{k-1}) + f''(x_{k-1})(x_k - x_{k-1})$$

  2. Solve for x_k to get next trial value

$$-f'(x_{k-1}) = f''(x_{k-1})(x_k - x_{k-1})$$

$$\frac{-f'(x_{k-1})}{f''(x_{k-1})} = x_k - x_{k-1}$$

$$x_k = x_{k-1} - \frac{f'(x_{k-1})}{f''(x_{k-1})}$$

Now that we've reviewed a couple general approaches to moving blindfolded around the curve, what built-in methods does R's `optim` have?

- `Nelder-Mead`: this is the default; it is slow but somewhat robust to non-differentiable functions.
- `BFGS`: a quasi-Newton Method; it is fast but needs a well behaved objective function.
- `L-BFGS-B`: similar to BFGS but allows box-constraints (i.e. upper and lower bounds on variables).
- `CG`: conjugate gradient method, may work for really large problems (we won't really use this).
- `SANN`: uses simulated annealing – a stochastic global optimization method; it is very robust but *very* slow.

For pset 3, using BFGS for all of the problems involving `optim`

Refresher of where we are in the steps that find us $\hat{\beta}$ that maximize the time to death we observed in the Titanic data

1. Write a function for the log likelihood that takes in X, y, and a vector of trial or starting parameters for which you're trying to find the maximum likelihood estimate (e.g., vector of β, vector containing starting/trial values for both β and σ^2)
2. Plug that function into R's `optim` along with a vector of starting/trial values
3. The values you're most interested in `optim` returning are, where K refers to the number of parameters:
   - 3.1 K-length vector of estimates:
     `optim$par`
   - 3.2 $K \times K$ matrix of all combinations of second derivatives of estimates (the Hessian) that we use to derive standard error of estimates:
     `optim$Hessian`

Step two (use optim to find MLE): useful arguments

1. `par`: vector of trial/starting parameters; should be equal in length to number of parameters we're trying to estimate (e.g., 10 covars + σ^2 = vector of length 11)
2. `fn`: function we're trying to optimize– here, it's `loglik.exp`
3. Additional arguments: after the function, can specify X, y, etc. and whatever other arguments the function fed in `fn` needs
4. `method` is the algorithm used to find the maximum.
5. `fnscale` multiplies the function by the given constant. As a default `optim()` finds the minimum so multiplying our function by -1 fools `optim()` into finding the maximum.
6. `hessian = TRUE` requests that `optim` return a $K \times K$ matrix of second derivatives, where K = number of parameters

Step two (use optim to find MLE): practice in R

- Use optim to find four $\hat{\beta}$ for the exponential regression function whose log likelihood you programmed as `loglik.exp`:
  1. $\hat{\beta}$ for intercept
  2. $\hat{\beta}$ for social class (classCode)
  3. $\hat{\beta}$ for female
  4. $\hat{\beta}$ for Age
- You can use `method = "BFGS"` and set trial parameters to zero
- After running optim, extract the "par" from the output to get $\hat{\beta}$ and to find standard error of each $\hat{\beta}$, extract Hessian and use following formula:

  `sqrt(diag(solve(-optoutput$hessian)))`
- Briefly interpret the coefficients (for social class variable, higher = in worse class (e.g., 3rd class steerage; 2nd class, 1st class)); if you'd like, you can compare them to `survreg` in the survival library and the esimates should be similar

Step two (use optim to find MLE): solution

```
#run optim with trial parameters
#of 0 and BFGS
opt.time_death <- optim(par = rep(0, 4),
                        fn = loglik.exp,
                        y = y,
                        X = X,
                        control = list(fnscale = -1),
                        method = "BFGS",
                        hessian = TRUE)

#extract estimates and SE
beta_td <- opt.time_death$par
beta_td_se <- sqrt(diag(solve(-opt.time_death$hessian)))
```

Results and comparison with survreg

variables	beta_optim	se_optim	beta_survreg	se_survreg
Intercept	5.1838	0.1612	5.1858	0.1612
Class (1 = 1st; 3 = steerage)	-0.2134	0.0467	-0.2137	0.0467
Age	-0.0059	0.0028	-0.0060	0.0028
female	0.4613	0.0757	0.4614	0.0757

But remember how we feel about usefulness of raw coefficients...

- See social class is highly significant, with passengers in a worse social class (higher value) having a significantly higher expected time to death

$$t = \frac{\beta}{se} \approx -\frac{0.2134}{0.0467} \approx -4.57$$

- Gender and age also seem significant, with older individuals having a shorter time to death and females with a longer time to death
- You're interested in the difference in time to death the model predicts between two passengers who share the same observed characteristics with Jack and Rose (who seems to have a lot going for her longer survival):
  - Jack: 20 year old male with social class $= 3$
  - Rose: 17 year old female with social class $= 1$

Steps for using simulation to generate QOI: adapted from Lecture 3, slide 74

1. Choose values of explanatory variables- in this case, it'll be one vector representing Jack's characteristics and another vector representing Rose's
2. Simulate estimation uncertainty (we use θ as a placeholder to refer to any set of parameters– in this case, only β, but in other cases, might also include σ^2):
   2.1 Draw θ from their sampling distribution: $N(\hat{\beta}, \hat{Var}(\hat{\beta}))$. Label the random draw $\tilde{\theta}$ and in this case, $\tilde{\theta} = \{\tilde{\beta}\}$
   - Why can we use multivariate normal to draw betas when our outcome follows an exponential distribution? CLT! Even if outcome follows exponential, sampling distribution of β is normal as$n \to \infty$
3. Use the $\tilde{\beta}$ to compute the simulated systematic component – in this case, $\tilde{\mu_i} = e^{X_i\tilde{\beta}} = \frac{1}{\tilde{\lambda_i}}$
4. Add fundamental uncertainty: draw $\tilde{y_i} \sim exp(\tilde{\lambda_i})$

Step one: choose (or fix) values of explanatory variables

```
jack <- data.frame(Int = 1,
                   classCode = 3,
                   Age = 20,
                   female = 0)
rose <- data.frame(Int = 1,
                   classCode = 1,
                   Age = 17,
                   female = 1)
```

Step two: draw the parameters (just beta here) from multivariate normal sampling distribution

- When defining the multivariate normal, we use the $\hat{\beta}$ and $\hat{Var}$ we obtained using `optim`- here simulating 5000 values

```
beta_tild_td <- rmvnorm(n = 5000,
                        mean = opt.time_death$par,
                      sigma = solve(-opt.time_death$hessian))
```

- Result is a 5000×4 matrix that for each intercept/covariate, has 5000 different realizations of the $\hat{\beta}$ parameter

Step three: use the $\tilde{\beta}$ to compute the simulated systematic component

1. Remember that for this particular distribution, and that because Jack and Rose have different covariates, we'll want both a $\tilde{\lambda_{Jack}}$ and $\tilde{\lambda_{Rose}}$:

$$\lambda_i = \frac{1}{e^{X_i \beta}}$$

2. Keeping that in mind, we can do the following to generate the λ_i to feed into our exponential distribution for the next step

```
jack_mat <- as.matrix(jack)
lambda_tild_jack <- 1/exp(jack_mat %*% t(beta_tild_td))
rose_mat <- as.matrix(rose)
lambda_tild_rose <- 1/exp(rose_mat %*% t(beta_tild_td))
```

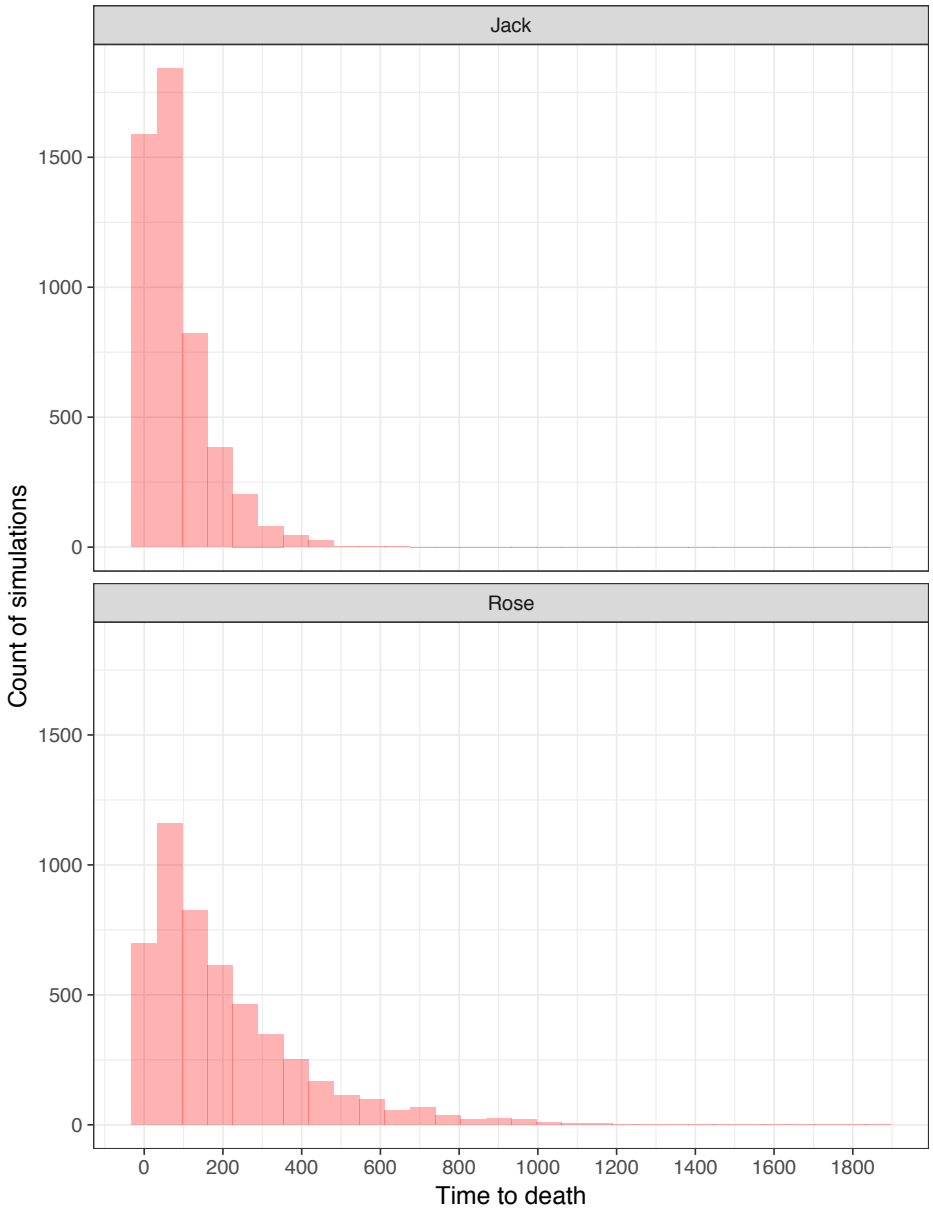
Step four: Add fundamental uncertainty: draw $\tilde{y}_i \sim exp(\tilde{\lambda}_i)$

- Since we have both $\tilde{\lambda_{Jack}}$ and $\tilde{\lambda_{Rose}}$ from previous step, we'll get two different length 5000 time to death vectors, which was the whole point of the exercise!
- In R, setting seed to 1484:

```
jack_simout <- rexp(5000, rate = lambda_tild_jack)
rose_simout <- rexp(5000, rate = lambda_tild_rose)
```

- Could then take difference and plot a histogram of that, plot two distributions of time to death side by side, etc.!

One way of comparing



What we reviewed

1. Write out the model.
2. Calculate the likelihood ($\mathrm{L}(\theta|y)$) for all observations.
   - Using exponential distribution, move from $\theta = \lambda$ the same for all observations to λ_i
3. Take the log of the likelihood ($\ell(\theta|\mathbf{Y})$).
4. Plug in the systematic component for θ_i.
5. Bring in observed data.
6. Maximize $\ell(\theta|y)$ with respect to θ and confirm that this is a maximum.
   - Previously: analytic optimization or optimize; this week: numerical optimization using optim
7. Find the variance of your estimate.
8. Using simulation to go from estimates to QOI