lecture17

Rest of semester plan

- no more new labs
- Nov 20: databases
- Nov 23: work on projects in class
- Nov 25: work on projects in class
- Nov 30: ggplot()
- Dec 2,4,7,9,11: group project presentations

Final Projects: All

1. Write a script that gets builds a dataset to summarize yelp.com reviews of restaurants in a town. Your code should be a function that takes two town name (i.e. Amherst) and state (i.e. MA) as an input, and it should return a dataframe were where each row is one restaurant, and the variables should include restaurant name, type of food, price range, average rating, and number of raters. (You can do more with this for extra credit!) Due at end of finals.

Final Projects: 5 Groups of 5 people

- Shiny (Dec 2) - builds an interative website with R data analysis
- Package to use more than one processor in parallel (Dec 4)
- Using Amazon Cloud to run R (Dec 7) - how to use Amazon Cloud to run R (HPC)
- **How to make an R package (Dec 9)**
- Web scraping package (Dec 11)

All should include an example (dont have to make it up but can get it from somewhere)

(JWS: run random assignment program!)

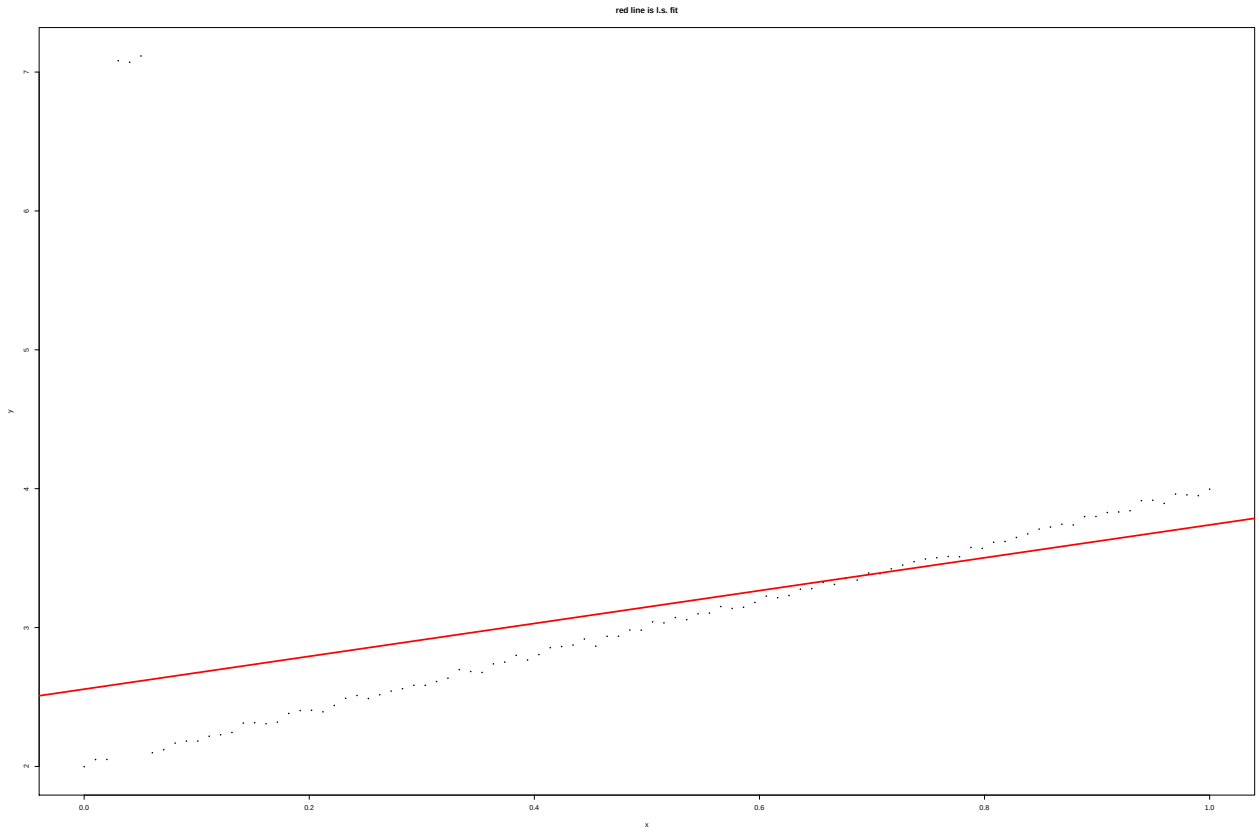
Optimization and simulation example

Least squares can be biased by outliers. Let's simulate some data to show this:

```
n <- 100
a <- 2
b <- 2
sig <- 0.02
x <- seq(from = 0, to = 1, length = n)
y <- a + b * x + rnorm(n, sd = sig)
y[4:6] <- y[4:6] + 5
```

======

```
## Warning: package 'numDeriv' was built under R version 3.1.3
```



least squares, normality, and t

- Points that are far away from the mean will be very influential because they will have lots of weight
- Least squares is maximum likelihood normal model for errors.
- Normal model for errors says that observations $>$ a few sds from mean function are unlikely.
- t model (with small df) for errors says that observation far from the mean are not unlikely because the tails are heavier.
- – So then if we assume t errors then the points far away from the mean will not have that much weight.
- As a result, let's try maximum likelihood assuming t errors. It should not be influenced by outliers.

Steps

1. Build negative log likelihood function (and gradient)
2. Use optimization to find maximum likelihood estimate
3. Simulate more data to see how model does on average

Log-likelihood

```
# build a likelihood that uses t errors
t.lhood <- function(params, x.y.data, df = 4) {
    x <- x.y.data$x
    y <- x.y.data$y
    a <- params[1]  # intercept
    b <- params[2]  # slope
    sig <- exp(params[3])  # input=log(sd)
    mean.func <- a + b * x
    e <- (y - mean.func)/sig

    neg.log.l.hood <- -(-n * log(sig) + sum(dt(e, df = df, log = T)))
    return(neg.log.l.hood)
}
```

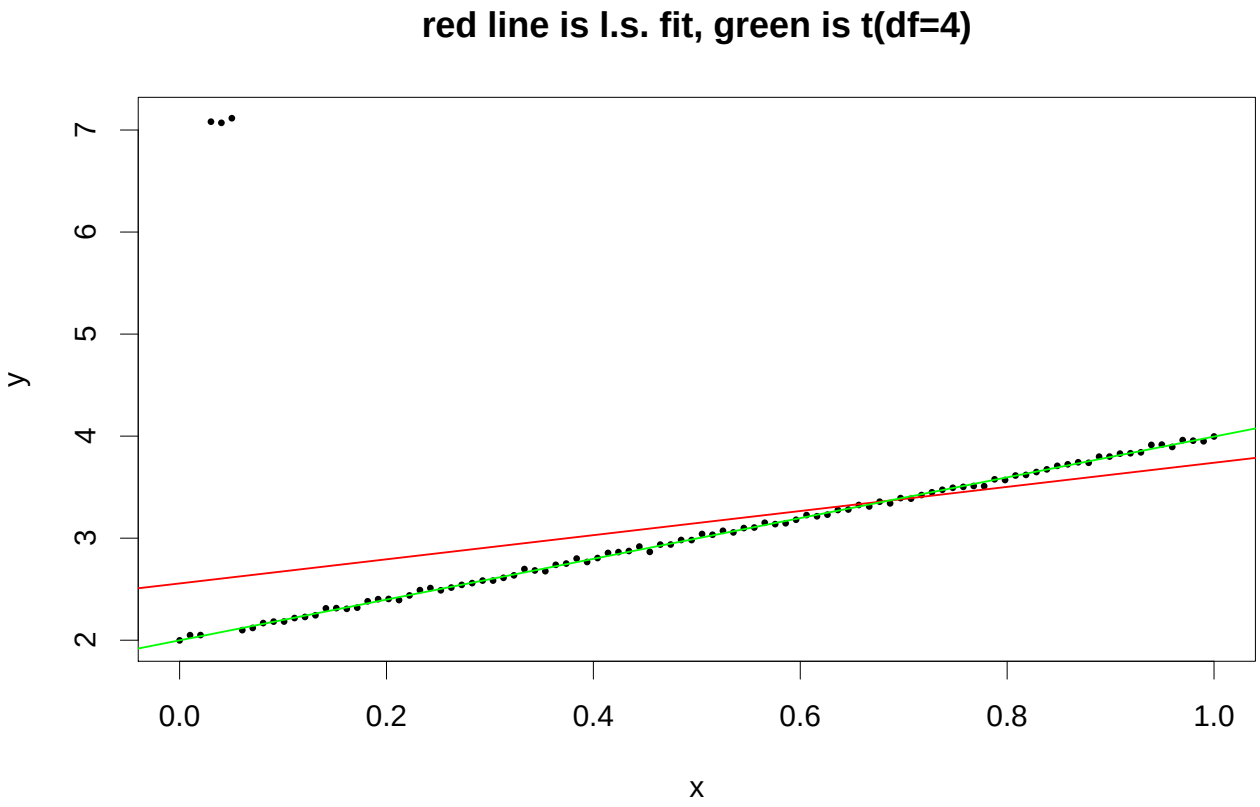
Numerical estimate of gradient

```
# build a gradient function
t.grad.neg.log.l <- function(params, x.y.data, df = 4) {
    grad.t <- grad(t.lhood, x = params, x.y.data = x.y.data, df = df)
    return(grad.t)
}
```

Optimize!

```
start <- c(1, 1, 0)
est <- optim(start, t.lhood, t.grad.neg.log.l, method = "BFGS", x.y.data = data,
    hessian = T)
```

Plot

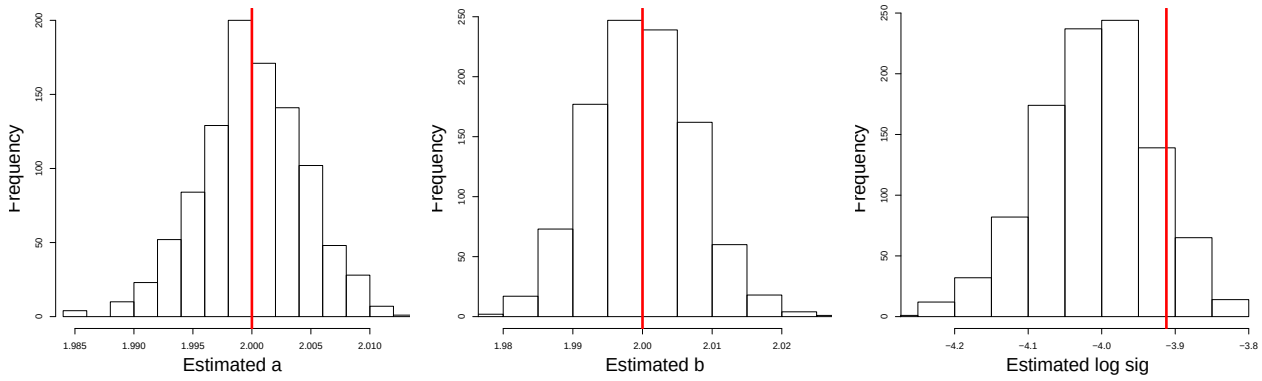


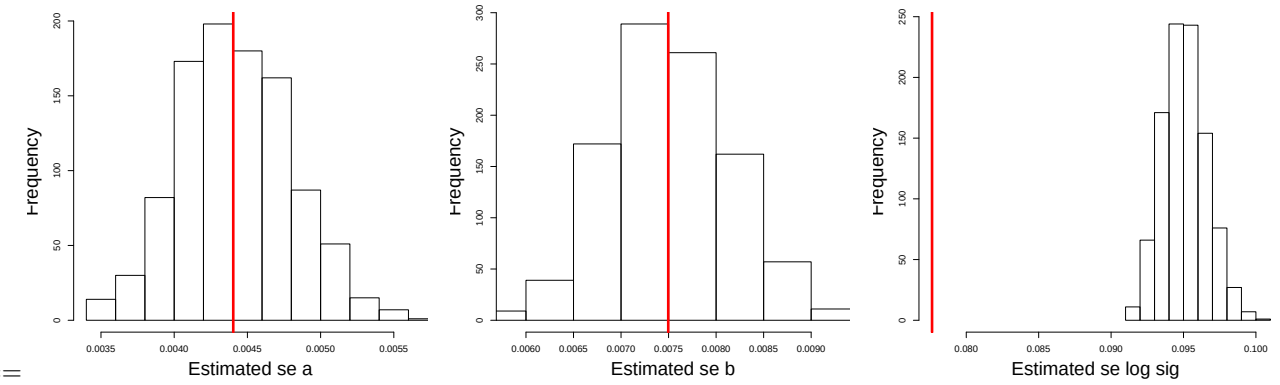
Simulation

1. Create lots of simulated datasets and apply method to each one.
2. Make histogram of estimated values and compare to truth.

====

```
MC.N <- 1000
keep.est <- data.frame(a.hat = rep(NA, MC.N), b.hat = rep(NA, MC.N), log.s.hat = rep(NA,
    MC.N))
keep.se <- data.frame(se.a.hat = rep(NA, MC.N), se.b.hat = rep(NA, MC.N), se.log.s.hat = rep(NA,
    MC.N))
for (mc.i in 1:1000) {
    y <- a + b * x + rnorm(n, sd = sig)
    y[4:6] <- y[4:6] + 5
    data <- data.frame(x = x, y = y)
    est <- optim(start, t.lhood, t.grad.neg.log.l, method = "BFGS", x.y.data = data,
        hessian = T)
    keep.est[mc.i, ] <- est$par
    keep.se[mc.i, ] <- sqrt(diag(solve(est$hessian)))
}
```





====

Summary

1. Method estimates intercept and slope well (little bias)
2. Method understimates error log sd on average.
3. sd errors from inv hessian are a little too big on average.
4. sd errors for log sd are much too large