EEEB UN3005/GR5005
Lab - Week 11 - 08 and 10 April 2019

USE YOUR NAME HERE

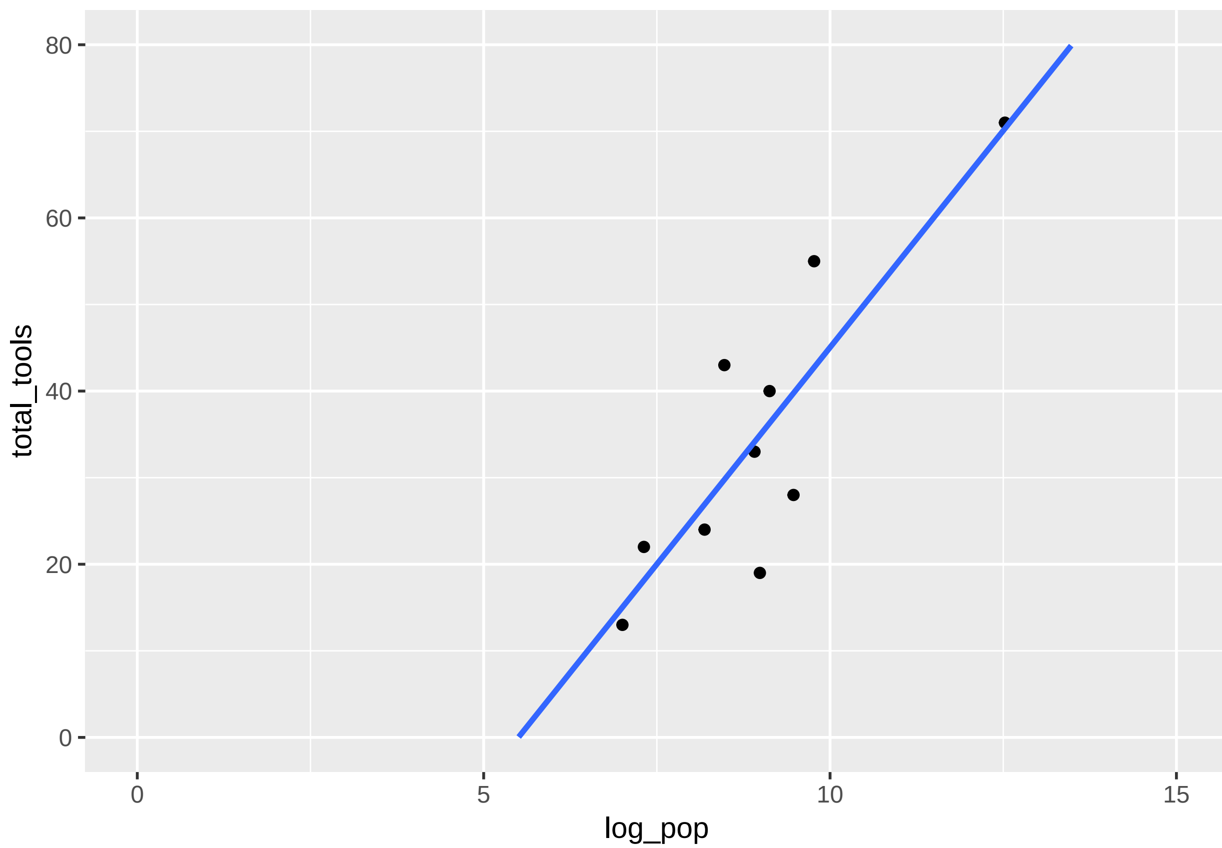
Link Functions and Poisson Regression

Exercise 1: Importing and Visualizing the Kline Dataset

Import the Kline dataset that was shown in the *Statistical Rethinking* text and lecture. Add a log population size variable (`log_pop`) to the data frame for use as a predictor variable. Now, visualize the relationship between `log_pop` and `total_tools` using a scatter plot in `ggplot()`. Make the x-axis limits of your plot span from 0 to 15 and the y-axis limits of your plot span from 0 to 80. In addition, add the layer `geom_smooth(method = "lm", se = FALSE, fullrange = TRUE)` to your plot in order to display a linear trend line on top of the raw data.

```
data(Kline)
d = Kline
d$log_pop = log(d$population)
graph = ggplot(d, aes(x = log_pop, y = total_tools)) +
    geom_point() +
    geom_smooth(method = 'lm', se = FALSE, fullrange = TRUE) +
    xlim(c(0, 15)) +
    ylim(c(0, 80))
plot(graph)
```

```
## Warning: Removed 37 rows containing missing values (geom_smooth).
```



Exercise 2: Fitting a Poisson GLM and a Standard Linear Model

First, fit a Poisson GLM to the Kline data (with `map()`), using `log_pop` as a predictor of total tool count. This model should replicate m10.12 from the *Statistical Rethinking* book, so reference the text if needed. After fitting the model, use `precis()` to display the 97% PIs for all model parameters.

Now, fit a standard linear model (with a Gaussian outcome distribution) to the Kline data, again using `log_pop` as a predictor of total tool count. You'll likely encounter trouble getting the model to fit unless you use the following priors (or something very similar to them): intercept parameter with a prior of `dnorm(-50, 10)`, the beta coefficient for `log_pop` with a prior of `dnorm(0, 10)`, and the standard deviation parameter with a prior of `dunif(0, 10)`. Again, use `precis()` to display the 97% PIs for all model parameters after you've fit the model.

```
# model.pois = map(
#   alist(
#       total_tools ~ dpois(lambda),
#       log(lambda) <- a + b * log_pop,
#       a ~ dnorm(0, 100),
#       b ~ dnorm(0, 1),
#       ),
```

```
#   data = d)
model.gaus = map(
    alist(
        total_tools ~ dnorm(mu, sigma),
        mu <- a + b * log_pop,
        a ~ dnorm(-50, 10),
        b ~ dnorm(0, 10),
        sigma ~ dunif(0, 10)),
    data = d)
# precis(model.pois, prob = 0.97)
precis(model.gaus, prob = 0.97)
```

```
##          Mean StdDev   1.5%  98.5%
## a      -50.53   8.64 -69.28 -31.77
## b        9.51   0.99   7.37  11.65
## sigma    8.75   1.96   4.49  13.01
```

Exercise 3: Comparing Model-based Predictions

Imagine we discover a new Oceanic island with a population of 150 people (log population size of 5.01). Using `sim()`, generate predictions for total tool count on this island for both the Poisson GLM and the standard linear model. Report the mean value of the predictions generated from both models. Which model suggests a higher total tool count for this hypothetical island?

```
predictor = data.frame(log_pop = 5.01)
# pred.sim.pois = sim(model.pois, data = predictor)
pred.sim.gaus = sim(model.gaus, data = predictor)
```

```
## [ 100 / 1000 ]
[ 200 / 1000 ]
[ 300 / 1000 ]
[ 400 / 1000 ]
[ 500 / 1000 ]
[ 600 / 1000 ]
[ 700 / 1000 ]
[ 800 / 1000 ]
[ 900 / 1000 ]
[ 1000 / 1000 ]
```

```
# print(mean(pred.sim.pois))
print(mean(pred.sim.gaus))
```

```
## [1] -2.488587
```

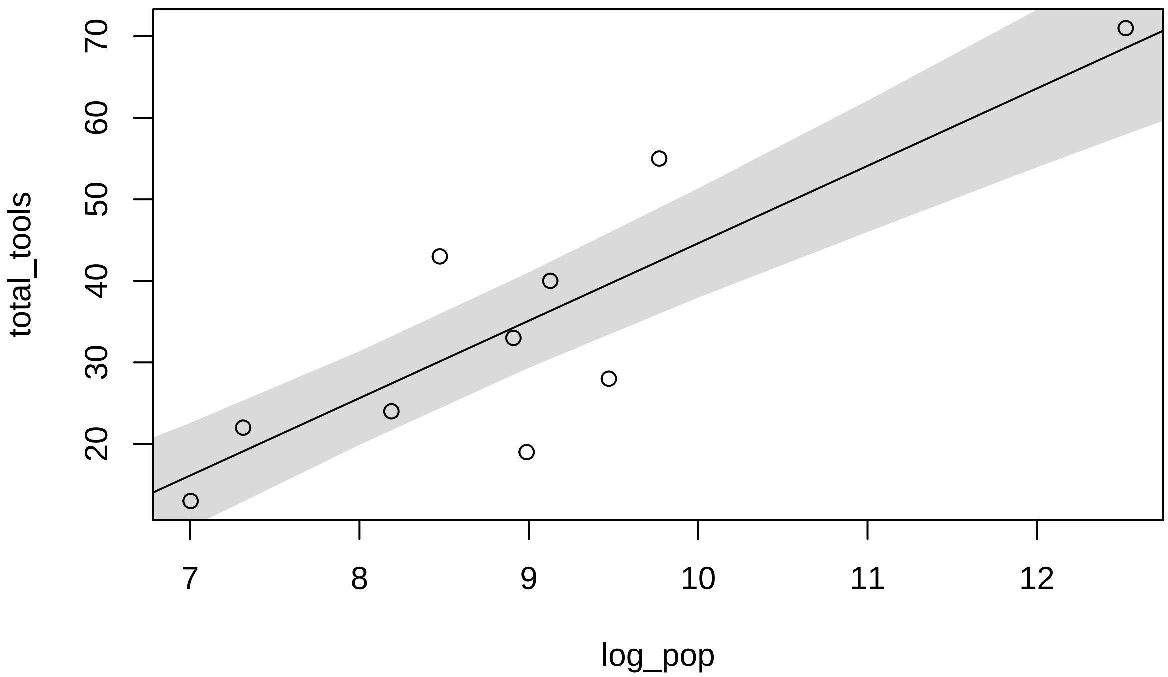
Exercise 4: Visualizing Predictions

Visualize the total tool count predictions from both models using a method of your choice. Using your visualization and the previous exercises as a guide, do you think both of these models generate sensible predictions for total tool count for a hypothetical island with a log population size of 5.01? Why or why not?

```
predictor = data.frame(log_pop = 1:15)
pred.sim.gaus.seq = link(model.gaus, data = predictor, n = 10000)
```

```
## [ 1000 / 10000 ]
[ 2000 / 10000 ]
[ 3000 / 10000 ]
[ 4000 / 10000 ]
[ 5000 / 10000 ]
[ 6000 / 10000 ]
[ 7000 / 10000 ]
[ 8000 / 10000 ]
[ 9000 / 10000 ]
[ 10000 / 10000 ]
```

```
mu.gaus = apply(pred.sim.gaus.seq, 2, mean)
PI.gaus = apply(pred.sim.gaus.seq, 2, PI, prob = 0.97)
plot(total_tools ~ log_pop, data = d)
lines(1:15, mu.gaus)
shade(PI.gaus, 1:15)
```


total_tools
70
60
50
40
30
20
7
8
9
10
11
12
log_pop