EEEB UN3005/GR5005
Homework - Week 11 - Due 16 Apr 2019

Xun Zhao, xz2827

Homework Instructions: Complete this assignment by writing code in the code chunks provided. If required, provide written explanations below the relevant code chunks. Replace "USE YOUR NAME HERE" with your name in the document header. When complete, knit this document within RStudio to generate a pdf. Please review the resulting pdf to ensure that all content relevant for grading (i.e., code, code output, and written explanations) appears in the document. Rename your pdf document according to the following format: hw_week_11_firstname_lastname.pdf. Upload this final homework document to CourseWorks by 5 pm on the due date.

After loading the `rethinking` package, if you run `data(salamanders)` you'll find a dataset of salamander counts (the `SALAMAN` variable) recorded at 47 forest plots. Given a common exposure across forest plots (i.e., if data was collected at a regular interval for all plots), then this count data is ideal for modeling as a Poisson variable.

Problem 1 (4 points)

Model the relationship between salamander count (`SALAMAN`) and percentage of vegetation cover on the forest floor (`PCTCOVER`) using a Poisson generalized linear model (GLM). Use priors of your choice. If you're having trouble getting a model that consistently fits without error messages, trying using explicit start values of 0 for all model parameters.

Use `precis()` to report the 97% PI of fit model parameters. What does your model suggest about the effect of vegetation cover on salamander counts?

```
data(salamanders)
d = salamanders
model.pois = map(
                  alist(
                        SALAMAN ~ dpois(lambda),
                        log(lambda) <- a + b * PCTCOVER,
                        a ~ dnorm(0, 1),
                        b ~ dnorm(0, 1)),
                  data = d)
precis(model.pois, 0.97)
```

```
##    Mean StdDev  5.5% 94.5%
## a -1.25   0.38 -1.86 -0.63
## b  0.03   0.00  0.02  0.04
```

Answer:

According to the result, intercept `b` is positive.

So the expectation of variable `SALAMAN` is positive related to `PCTCOVER`.

Problem 2 (3 points)

Refit the same model as in Problem 1, this time using `map2stan()`, specifying four MCMC chains. Don't worry about the large amount of R console output that will turn up in your knit pdf document.

After you've fit the model, report the 97% HPDIs of model parameters using `precis()`, and use a method of your choice (two were shown in lecture) to display parameter trace plots from the fit model.

```
model.stan = map2stan(
                      alist(
                            SALAMAN ~ dpois(lambda),
                            log(lambda) <- a + b * PCTCOVER,
                            a ~ dnorm(0, 1),
                            b ~ dnorm(0, 1)),
                      data = d,
                      chains = 4)
```

```
##
## SAMPLING FOR MODEL 'SALAMAN ~ dpois(lambda)' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.54 seconds (Warm-up)
## Chain 1:                0.077 seconds (Sampling)
## Chain 1:                0.617 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'SALAMAN ~ dpois(lambda)' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.186 seconds (Warm-up)
## Chain 2:                0.073 seconds (Sampling)
## Chain 2:                0.259 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'SALAMAN ~ dpois(lambda)' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
```

```
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.192 seconds (Warm-up)
## Chain 3:                0.08 seconds (Sampling)
## Chain 3:                0.272 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'SALAMAN ~ dpois(lambda)' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.167 seconds (Warm-up)
## Chain 4:                0.073 seconds (Sampling)
## Chain 4:                0.24 seconds (Total)
## Chain 4:
##
## SAMPLING FOR MODEL 'SALAMAN ~ dpois(lambda)' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
```

```
## Chain 1: WARNING: No variance estimation is
## Chain 1:          performed for num_warmup < 20
## Chain 1:
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0 seconds (Warm-up)
## Chain 1:                0.001 seconds (Sampling)
## Chain 1:                0.001 seconds (Total)
## Chain 1:
```

```
## Warning: There were 1 divergent transitions after warmup. Increasing adapt_delta abov
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
## Computing WAIC
```

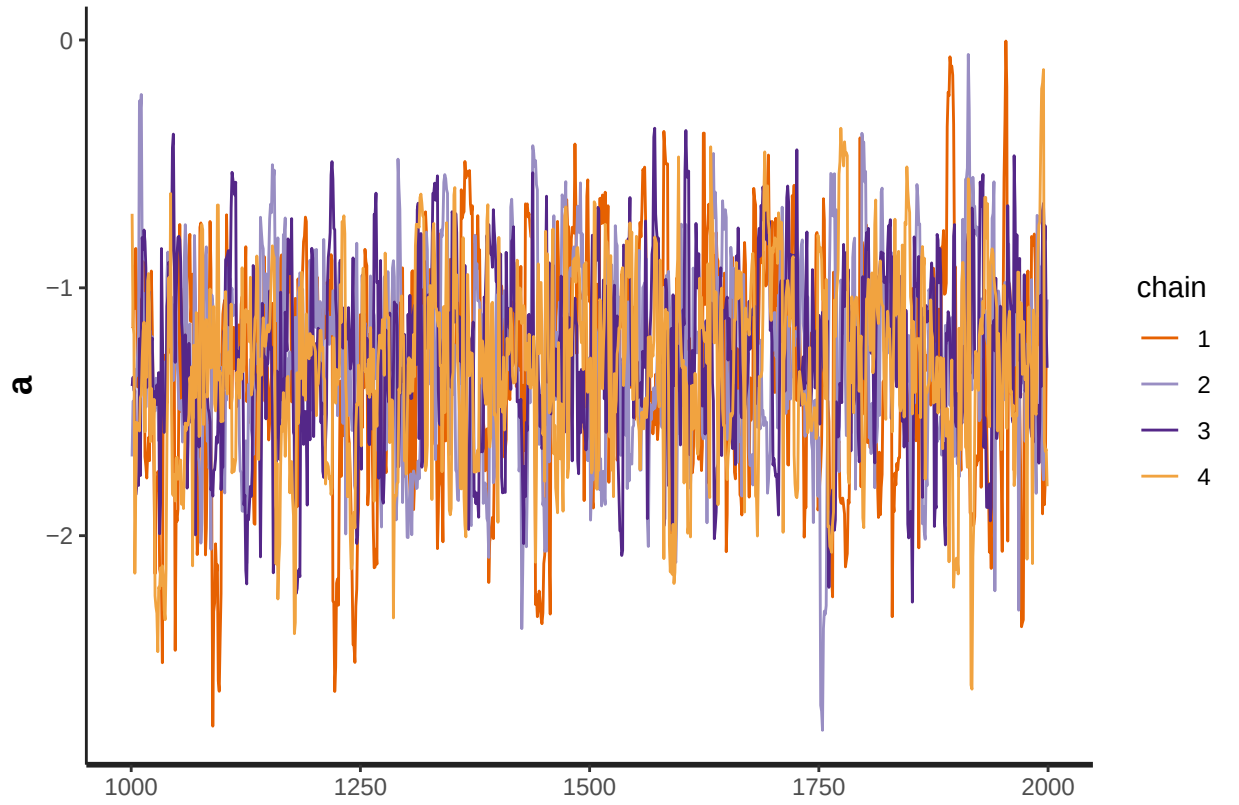
```
## Constructing posterior predictions
```

```
## [ 400 / 4000 ]
[ 800 / 4000 ]
[ 1200 / 4000 ]
[ 1600 / 4000 ]
[ 2000 / 4000 ]
[ 2400 / 4000 ]
[ 2800 / 4000 ]
[ 3200 / 4000 ]
[ 3600 / 4000 ]
[ 4000 / 4000 ]
```

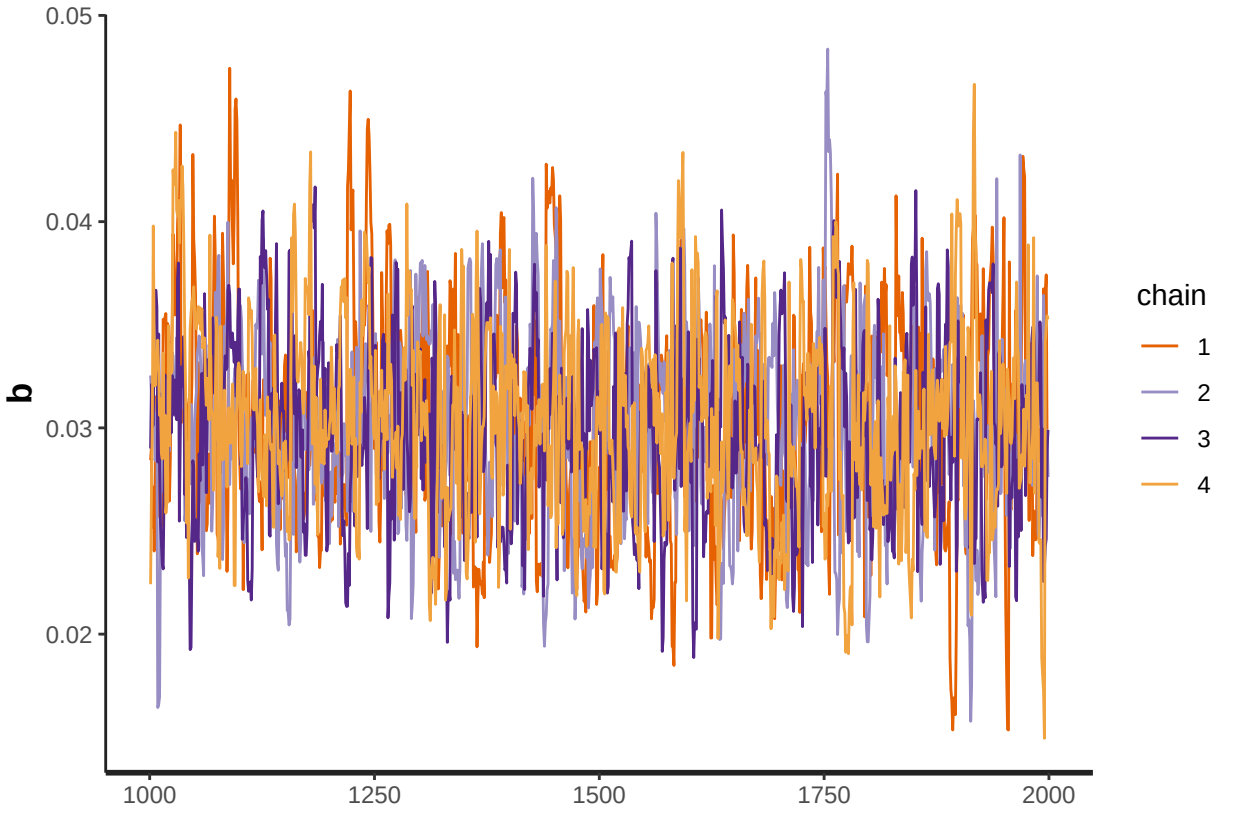
```
precis(model.stan, 0.97)
```

```
##    Mean StdDev lower 0.89 upper 0.89 n_eff Rhat
## a -1.30   0.39      -1.89      -0.66   733    1
## b  0.03   0.00       0.02       0.04   740    1
```

```
rstan::traceplot(model.stan@stanfit, par = c('a'))
```




```
rstan::traceplot(model.stan@stanfit, par = c('b'))
```



Problem 3 (3 points)

Generate a plot showing the raw salamander count data (against vegetation cover) along with model-based predictions from the Poisson GLM you fit in Problem 1. More specifically, plot a line showing the mean predicted count and a shaded 97% HPDI interval for the predictions.

As a hint, use `sim()` to generate your predictions. Note that this will require counterfactual data, so your prediction generation process will start by defining a sequence of predictor values to generate predictions for. From there, everything should be very similar to examples you've encountered previously in class.

Using your plot to help with interpretation, how does the model perform well and how does it perform poorly?

```
pred.seq = seq(min(d$PCTCOVER), max(d$PCTCOVER), out.length = 50)
```

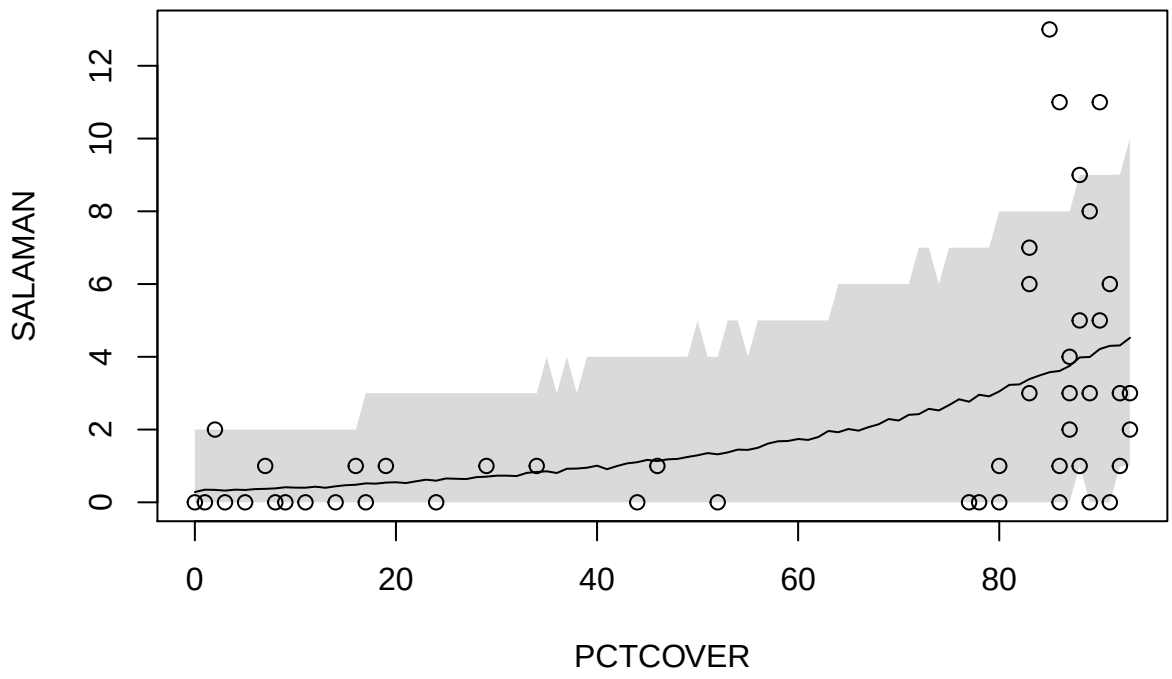
```
## Warning: In seq.default(min(d$PCTCOVER), max(d$PCTCOVER), out.length = 50) :
##  extra argument 'out.length' will be disregarded
```

```
pred = data.frame(PCTCOVER = pred.seq)
preds.sim.pois = sim(model.pois, data = pred)
```

```
## [ 100 / 1000 ]
[ 200 / 1000 ]
```

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

```r
mu.pois = apply(preds.sim.pois, 2, mean)
PI.pois = apply(preds.sim.pois, 2, PI, prob = 0.97)
plot(SALAMAN ~ PCTCOVER, data = d)
lines(pred.seq, mu.pois)
shade(PI.pois, pred.seq)
```



Answer :

Considering the model, the line and shade can cover most data points, and shows that `SALAMAN` is positively correlated with `PCTCOVER`.

However, when it comes to the data itself, we can see that the data can be devided into two

parts, the low PCTCOVER part and high PCTCOVER part. Especially for high PCTCOVER one, the SALAMAN is distributed in a large range, from low SALAMAN to high SALAMAN. Conversely, for the low PCTCOVER part, the SALAMAN is only distributed in a small range.

So I think the GLM model is wrong, and we should fit two differents parts separately.
