Exam 2

Nate Hawkins

3/14/2021

```r
library(R2jags)

library(dplyr)

library(brms)

library(loo)

library(rstan)
```

For the first set of questions we will be using the data file **influent.dat**. You should have received that data file with the exam. Water flows into the Mississippi river from a number of streams and rivers. These sources carry nitrogen into the river. In the data set, there are six sources of the nitrogen, which are a random sample of the many hundreds of streams and rivers that flow into the Mississippi. These sources have been classified by type. The three types are: (1) no farm land in watershed, (2) less than 50% farm land in watershed, and (3) more than 50% farmland in watershed. These three types are the only types that we seek to analyze. In the data file, the three columns are river source, nitrogen, and type.

1. First read in the data and print out the first six rows of the data.

```r
influent <- read.table("influent.dat")
colnames(influent) <- c("source", "nitrogen", "type")
```

2. Ignoring the source, use brm to write a model to find the differences in type. Use priors of normal(0,100) for the type effects, and gamma(2,.1) for σ_{error}. Remember that the kind of variable you are working with will make a difference. What is the level of nitrogen estimated for type 3.

Level of nitrogen estimated for type 3 is 36.36

```r
source = influent$source
nitrogen = influent$nitrogen
type = influent$type
influent$vtype = as.factor(influent$type)


fit1 <- brm(formula = nitrogen ~ -1 + vtype, data = influent,
            family = "gaussian",
            prior = c(set_prior("normal(0,100)", class = "b"),
                      set_prior("gamma(2,.1)", class = "sigma")),
            warmup = 1000, iter = 5000, chains = 4,
```

```
            #control = list(adapt_delta = 0.98),
            save_pars = save_pars(all = TRUE))



summary(fit1)

  Family: gaussian
   Links: mu = identity; sigma = identity
 Formula: nitrogen ~ -1 + vtype
    Data: influent (Number of observations: 37)
 Samples: 4 chains, each with iter = 5000; warmup = 1000; thin = 1;
          total post-warmup samples = 16000

 Population-Level Effects:
        Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
 vtype1    15.61      2.39    10.93    20.43 1.00    16978    11131
 vtype2    19.92      1.59    16.82    23.08 1.00    16060    11019
 vtype3    36.32      3.33    29.83    42.94 1.00    16383    11428

 Family Specific Parameters:
       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
 sigma     7.46      0.95     5.86     9.57 1.00    15101    12282

 Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
 and Tail_ESS are effective sample size measures, and Rhat is the potential
 scale reduction factor on split chains (at convergence, Rhat = 1).

chains <- as.matrix(fit1)
dim(chains)
head(chains)
sims <- as.mcmc(chains)
raftery.diag(sims)
effectiveSize(sims)
```

3. What is the looic for this model?

The looic is 255.6

```
loo1 <- loo(fit1)
print(loo1)


 Computed from 16000 by 37 log-likelihood matrix

          Estimate  SE
 elpd_loo   -127.7 4.0
 p_loo         3.3 0.8
 looic       255.5 8.0
 ------
 Monte Carlo SE of elpd_loo is 0.0.
```

```
All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

4. Now run the same model using JAGS. Use the same priors for the type parameters (remember, normal priors in JAGS use precision), but use a gamma(5,.1) for σ^2_{error}. What is the level of nitrogen estimated for type 3?

```
mdl <- "

model {

  for(i in 1:37){
    nitrogen[i] ~ dnorm(mu[i], 1/s2error)
    mu[i] <- beta[type[i]]
  }

  # Priors
  for(i in 1:3){
    beta[i] ~ dnorm(0, 0.0001)
  }

  s2error ~ dgamma(5, 0.1)

}
"

writeLines(mdl, 'fit2.txt')

source = influent$source
nitrogen = influent$nitrogen
type = influent$type

data.jags <- c('nitrogen', 'type')
parms <- c('beta' ,'s2error')

fit2 <- jags(data= data.jags, parameters.to.save = parms,
                  model.file = 'fit2.txt', inits = NULL,
                  n.iter = 20000, n.thin = 5, n.burnin = 2000,
                  n.chains = 5)
```

```
module glm loaded

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 37
   Unobserved stochastic nodes: 4
   Total graph size: 84
```

```
 Initializing model

fit2

 Inference for Bugs model at "fit2.txt", fit using jags,
  5 chains, each with 20000 iterations (first 2000 discarded), n.thin = 5
  n.sims = 18000 iterations saved
          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat n.eff
 beta[1]   15.570   2.305  11.022  14.053  15.560  17.118  20.080 1.001 18000
 beta[2]   19.883   1.538  16.861  18.879  19.866  20.891  22.955 1.001 18000
 beta[3]   36.352   3.217  29.982  34.208  36.346  38.468  42.730 1.001 18000
 s2error   52.802  11.465  34.535  44.578  51.389  59.398  78.966 1.001 18000
 deviance 251.400   2.736 247.987 249.403 250.754 252.749 258.306 1.001 14000

 For each parameter, n.eff is a crude measure of effective sample size,
 and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

 DIC info (using the rule, pD = var(deviance)/2)
 pD = 3.7 and DIC = 255.1
 DIC is an estimate of expected predictive error (lower deviance is better).

sims <- as.mcmc(fit2)
chains <- as.matrix(sims)
sims <- as.mcmc(chains)
raftery.diag(sims)
effectiveSize(sims)
```

5. What is the DIC of the above model?

DIC is 255.3

6. Now we want to account for the variance in sources to make inference. Redo the model in brm, but now put source in the model appropriately. Use a gamma(2,.1) prior for σ_{source}. The other priors can stay the same. Now what is the estimate for the type 3 mean?

Estimate for type 3 mean is 12.24. This plus whatever the effect for whatever source it is.

```
fit3 <- brm(formula = nitrogen ~ -1 + vtype + (1|source), data = influent,
            family = "gaussian",
            prior = c(set_prior("normal(0,10)", class = "b"),
                      set_prior("gamma(2,.1)", class = "sigma"),
                      set_prior("gamma(2,.1)", class = "sd")),
            warmup = 1000, iter = 20000, chains = 4, thin = 10,
            #control = list(adapt_delta = 0.98),
            save_pars = save_pars(all = TRUE))
```

```
summary(fit3)

  Family: gaussian
   Links: mu = identity; sigma = identity
 Formula: nitrogen ~ -1 + vtype + (1 | source)
    Data: influent (Number of observations: 37)
 Samples: 4 chains, each with iter = 20000; warmup = 1000; thin = 10;
          total post-warmup samples = 7600

 Group-Level Effects:
 ~source (Number of levels: 6)
               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
 sd(Intercept)    16.13      7.57     4.57    34.13 1.00     6961     6295

 Population-Level Effects:
        Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
 vtype1     7.34      7.91    -9.51    21.51 1.00     7534     7448
 vtype2    11.38      7.71    -5.39    24.20 1.00     6803     6943
 vtype3    12.01     10.71    -9.27    32.01 1.00     7091     6080

 Family Specific Parameters:
       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
 sigma     6.85      0.92     5.32     8.90 1.00     6895     6900

 Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
 and Tail_ESS are effective sample size measures, and Rhat is the potential
 scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
fit3$fit

 Inference for Stan model: 68305c8400f0c8b7efc70543152eb5c9.
 4 chains, each with iter=20000; warmup=1000; thin=10;
 post-warmup draws per chain=1900, total post-warmup draws=7600.

                          mean se_mean    sd    2.5%     25%     50%     75%
 b_vtype1                 7.34    0.09  7.91   -9.51    2.34    7.90   12.89
 b_vtype2                11.38    0.09  7.71   -5.39    6.57   12.13   16.99
 b_vtype3                12.01    0.13 10.71   -9.27    4.70   11.95   19.57
 sd_source__Intercept    16.13    0.09  7.57    4.57   10.86   15.03   20.22
 sigma                    6.85    0.01  0.92    5.32    6.20    6.75    7.40
 r_source[1,Intercept]    9.93    0.10  7.97   -3.52    4.23    9.14   14.85
 r_source[2,Intercept]    2.52    0.09  7.77  -10.97   -2.95    1.73    7.36
 r_source[3,Intercept]    9.00    0.10  8.29   -5.58    3.10    8.51   14.21
 r_source[4,Intercept]   12.63    0.10  8.20   -1.17    6.70   11.97   17.73
 r_source[5,Intercept]    6.77    0.09  8.20   -7.98    0.92    6.13   12.11
 r_source[6,Intercept]   23.18    0.13 11.28    1.95   15.33   23.08   31.09
 lp__                  -145.89    0.04  3.43 -153.63 -147.89 -145.43 -143.40
 z_1[1,1]                 0.63    0.01  0.48   -0.32    0.33    0.62    0.93
 z_1[1,2]                 0.06    0.01  0.55   -1.26   -0.24    0.12    0.42
 z_1[1,3]                 0.57    0.01  0.54   -0.49    0.23    0.57    0.91
```

```
z_1[1,4]                0.83    0.01  0.50   -0.10    0.50    0.80    1.14
z_1[1,5]                0.41    0.01  0.54   -0.72    0.07    0.41    0.75
z_1[1,6]                1.51    0.01  0.66    0.25    1.08    1.47    1.91
                       97.5% n_eff Rhat
b_vtype1               21.51  7481    1
b_vtype2               24.20  6609    1
b_vtype3               32.01  7035    1
sd_source__Intercept   34.13  6854    1
sigma                   8.90  6962    1
r_source[1,Intercept]  27.34  6688    1
r_source[2,Intercept]  19.38  6703    1
r_source[3,Intercept]  26.63  7372    1
r_source[4,Intercept]  30.16  6859    1
r_source[5,Intercept]  24.21  7533    1
r_source[6,Intercept]  45.46  7070    1
lp__                 -140.49  6972    1
z_1[1,1]                1.58  6914    1
z_1[1,2]                0.98  7031    1
z_1[1,3]                1.64  7635    1
z_1[1,4]                1.87  7469    1
z_1[1,5]                1.45  7832    1
z_1[1,6]                2.91  7097    1

Samples were drawn using NUTS(diag_e) at Mon Mar 22 15:21:30 2021.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

```
# Diagnostics look good
chains <- as.matrix(fit3)
dim(chains)
```

```
[1] 7600   18
```

```
sims <- as.mcmc(chains)
raftery.diag(sims)
```

```


Quantile (q) = 0.025
Accuracy (r) = +/- 0.005
Probability (s) = 0.95

                       Burn-in  Total Lower bound  Dependence
                       (M)      (N)   (Nmin)       factor (I)
 b_vtype1              2        3924  3746         1.050
 b_vtype2              4        8206  3746         2.190
 b_vtype3              3        4188  3746         1.120
 sd_source__Intercept  2        3759  3746         1.000
 sigma                 6        8690  3746         2.320
 r_source[1,Intercept] 2        3759  3746         1.000
```

```
  r_source[2,Intercept] 2         3840  3746         1.030
  r_source[3,Intercept] 2         3600  3746         0.961
  r_source[4,Intercept] 2         3718  3746         0.993
  r_source[5,Intercept] 2         3718  3746         0.993
  r_source[6,Intercept] 2         3718  3746         0.993
  lp__                  2         3639  3746         0.971
  z_1[1,1]              2         3882  3746         1.040
  z_1[1,2]              2         3924  3746         1.050
  z_1[1,3]              2         3639  3746         0.971
  z_1[1,4]              2         3718  3746         0.993
  z_1[1,5]              2         3600  3746         0.961
  z_1[1,6]              2         3718  3746         0.993
```

```r
effectiveSize(sims)
```

```
             b_vtype1              b_vtype2              b_vtype3
             7600.000              6858.278              7175.409
  sd_source__Intercept                 sigma r_source[1,Intercept]
             6811.934              7600.000              7093.651
 r_source[2,Intercept] r_source[3,Intercept] r_source[4,Intercept]
             6905.563              7600.000              6970.358
 r_source[5,Intercept] r_source[6,Intercept]                  lp__
             7600.000              7200.955              6966.401
             z_1[1,1]              z_1[1,2]              z_1[1,3]
             6668.253              7143.009              7600.000
             z_1[1,4]              z_1[1,5]              z_1[1,6]
             7600.000              7600.000              7220.784
```

7. What is the looic for this model?

looic is 252.7

```r
loo3 <- loo(fit3)
```

8. Which model (problem 2 or problem 6 is better? Why?

The model for problem 6 is better. It has a looic of 252.7 compared to 255.3. Clearly accounting for source is important.

9. Now do the model with the variance for sources accounted for in JAGS. Use a gamma(5,.1) for the variance component σ^2_{source}. What is the estimate for the type 3 mean?

Estimate for type 3 mean is 36.24.

```r
mdl <- "

model {

  for(i in 1:37){
    nitrogen[i] ~ dnorm(mu[i], 1/s2error)
```

```
    mu[i] <- beta[type[i]] + u[source[i]]
  }

  # Priors
  for(i in 1:3){
    beta[i] ~ dnorm(0, 0.0001)
  }
  for(i in 1:6){
    u[i] ~ dnorm(0,1/s_source)
  }

  s2error ~ dgamma(5, 0.1)
  s_source ~ dgamma(5,0.1)

}
"

writeLines(mdl, 'fit4.txt')

source = influent$source
nitrogen = influent$nitrogen
type = influent$type

data.jags <- c('nitrogen', 'type', 'source')
parms <- c('beta' , 'u', 's2error')

fit4 <- jags(data= data.jags, parameters.to.save = parms,
                  model.file = 'fit4.txt', inits = NULL,
                  n.iter = 20000, n.thin = 5, n.burnin = 2000,
                  n.chains = 5)

 Compiling model graph
    Resolving undeclared variables
    Allocating nodes
 Graph information:
    Observed stochastic nodes: 37
    Unobserved stochastic nodes: 11
    Total graph size: 135

 Initializing model

(fit4)

 Inference for Bugs model at "fit4.txt", fit using jags,
  5 chains, each with 20000 iterations (first 2000 discarded), n.thin = 5
  n.sims = 18000 iterations saved
          mu.vect sd.vect   2.5%    25%    50%    75%  97.5%  Rhat n.eff
 beta[1]   15.593   5.122  5.419 12.300 15.621 18.843 25.856 1.001 18000
 beta[2]   19.917   4.056 11.835 17.324 19.906 22.505 27.959 1.001 18000
 beta[3]   36.243   7.261 21.861 31.563 36.218 40.889 50.659 1.001 18000
```

```
 s2error   45.742  10.521  29.092  38.265  44.339  51.735  70.205 1.001 18000
 u[1]       1.460   4.176  -6.915  -1.212   1.456   4.164   9.708 1.001 18000
 u[2]      -5.191   4.260 -13.857  -7.930  -5.108  -2.388   3.021 1.001 18000
 u[3]       0.961   5.040  -8.933  -2.290   0.941   4.204  11.108 1.001 18000
 u[4]       3.790   4.291  -4.691   1.000   3.767   6.552  12.443 1.001 13000
 u[5]      -0.971   5.042 -11.102  -4.198  -0.978   2.284   8.979 1.001 18000
 u[6]       0.106   6.582 -13.033  -3.980   0.121   4.253  13.152 1.001 18000
 deviance 244.616   3.832 239.069 241.797 244.001 246.727 253.897 1.001 18000

 For each parameter, n.eff is a crude measure of effective sample size,
 and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

 DIC info (using the rule, pD = var(deviance)/2)
 pD = 7.3 and DIC = 252.0
 DIC is an estimate of expected predictive error (lower deviance is better).
```

```
sims <- as.mcmc(fit4)
chains <- as.matrix(sims)
sims <- as.mcmc(chains)
raftery.diag(sims)
effectiveSize(sims)
```

10. What is the DIC for this model?

DIC is 252.0

11. Which model (problem 4 or problem 9) is better? Why?

DIC for model 9 is better, it is lower. Therefore there is value in accounting for the which source it comes from. We can now make inference for sources.

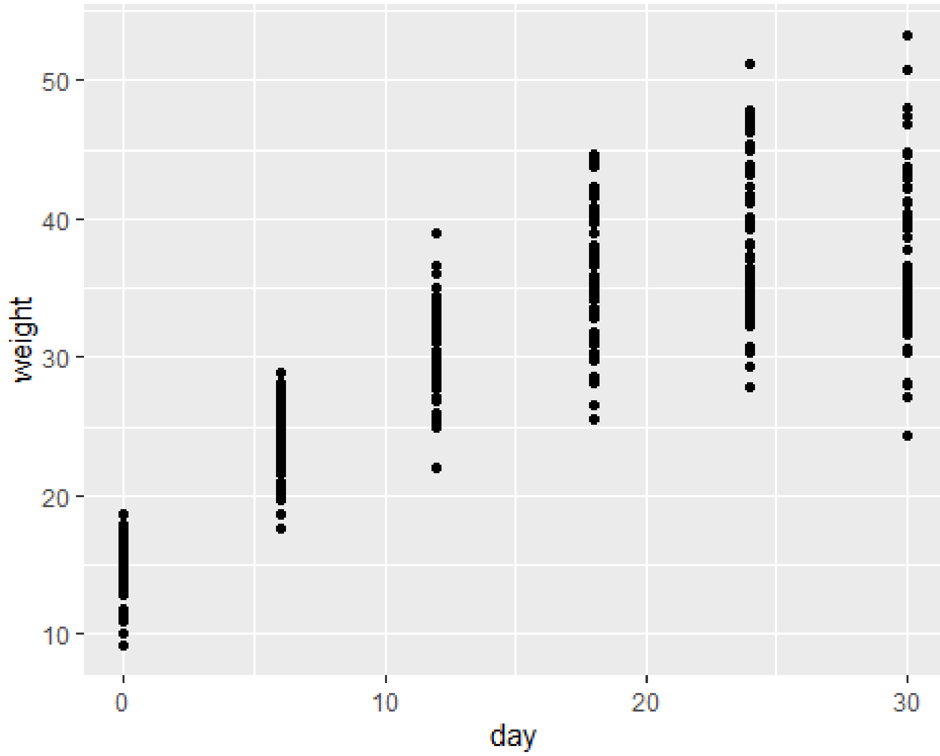
We will be using the data file **pig.dat** for the the next set of questions. You should have received that data file. The data set is from a feeding experiment comparing 3 treatments (basically how much of a particular additive was included in the food ration) on young hogs over the first 30 days of life. The measurements were taken every 6 days. The pig number identifies the pig within a treatment. That is, pig 1 in trt 1 is not the same pig as pig 1 in trt 2.

12. Plot the weight on the y-axis and the day on the x-axis, ignoring the trt and the pig.

```
pig <- read.table("pig.dat", header = TRUE)
head(pig)
```

```
   Obs trt day pig weight
 1   1   1   0   1   14.0
 2   2   1   6   1   22.1
 3   3   1  12   1   27.7
 4   4   1  18   1   31.8
 5   5   1  24   1   35.3
 6   6   1  30   1   32.6
```

```
ggplot(data = pig, mapping  = aes(x = day, y = weight)) +
  geom_point()
```



13. You will note from your plot that weight gain is fairly linear over the first 15 days or so, and then tails off. To account for this, we are going to add a quadratic term for day. That is, include a column for day^2 in the data set. Print the first 6 rows of the new data set.

```
pig$day2 = (pig$day)^2
head(pig)

  Obs trt day pig weight day2
1   1   1   0   1   14.0    0
2   2   1   6   1   22.1   36
3   3   1  12   1   27.7  144
4   4   1  18   1   31.8  324
5   5   1  24   1   35.3  576
6   6   1  30   1   32.6  900
```

14. Using JAGS, run a model with an intercept, linear term, and quadratic term for the overall data. You are essentially fitting the data you plotted, with no concern about treatments or the multiple observations on each pig. Use dnorm(0,0.0001) priors for the coefficients of the model (the β's), and a dgamma(1.1,.1) as the prior for σ^2_{error}. Print a summary of your model. What is the DIC of the model?

DIC is 2073

```r
mdl <- "
  model {

  for (i in 1:360){
    weight[i] ~ dnorm(mu[i], 1/vv)
    mu[i] <- b0 + b1*day[i] + b2*day2[i]
  }

  b0 ~ dnorm(0, 0.0001)
  b1 ~ dnorm(0, 0.0001)
  b2 ~ dnorm(0, 0.0001)
  vv ~ dgamma(1.1, 0.1)
  }

"
day = pig$day
day2 = pig$day2
weight = pig$weight

writeLines(mdl, 'fit5.txt')


data.jags <- c('weight', 'day', 'day2')
parms <- c('b0' , 'b1', 'b2', 'vv')

fit5 <- jags(data= data.jags, parameters.to.save = parms,
                    model.file = 'fit5.txt', inits = NULL,
                    n.iter = 12000, n.thin = 5, n.burnin = 2000,
                    n.chains = 5)
```

```
 Compiling model graph
    Resolving undeclared variables
    Allocating nodes
 Graph information:
    Observed stochastic nodes: 360
    Unobserved stochastic nodes: 4
    Total graph size: 1108

 Initializing model
```

```r
(fit5)
```

```
 Inference for Bugs model at "fit5.txt", fit using jags,
  5 chains, each with 12000 iterations (first 2000 discarded), n.thin = 5
  n.sims = 10000 iterations saved
          mu.vect sd.vect   2.5%    25%    50%    75%  97.5%  Rhat
 b0        14.738   0.503 13.771 14.393 14.740 15.080 15.719 1.002
 b1         1.719   0.078  1.564  1.665  1.718  1.771  1.872 1.001
 b2        -0.031   0.002 -0.036 -0.033 -0.031 -0.030 -0.027 1.001
 vv        18.331   1.385 15.860 17.366 18.254 19.222 21.276 1.001
```

```
 deviance 2068.906   2.874 2065.356 2066.790 2068.267 2070.282 2076.240 1.001
          n.eff
 b0        2600
 b1        6000
 b2       10000
 vv       10000
 deviance 10000

 For each parameter, n.eff is a crude measure of effective sample size,
 and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

 DIC info (using the rule, pD = var(deviance)/2)
 pD = 4.1 and DIC = 2073.0
 DIC is an estimate of expected predictive error (lower deviance is better).

sims <- as.mcmc(fit5)
chains <- as.matrix(sims)
sims <- as.mcmc(chains)
raftery.diag(sims)
effectiveSize(sims)
```

15. Run the same model using brm. Use the same priors for the β's (which will now be normal(0,100) since brm works in standard deviations), and a gamma(1.1,.5) prior for the standard deviation ($sigma_{error}$). Print a summary of your model. What is the looic of the model?

looic is 2073.6

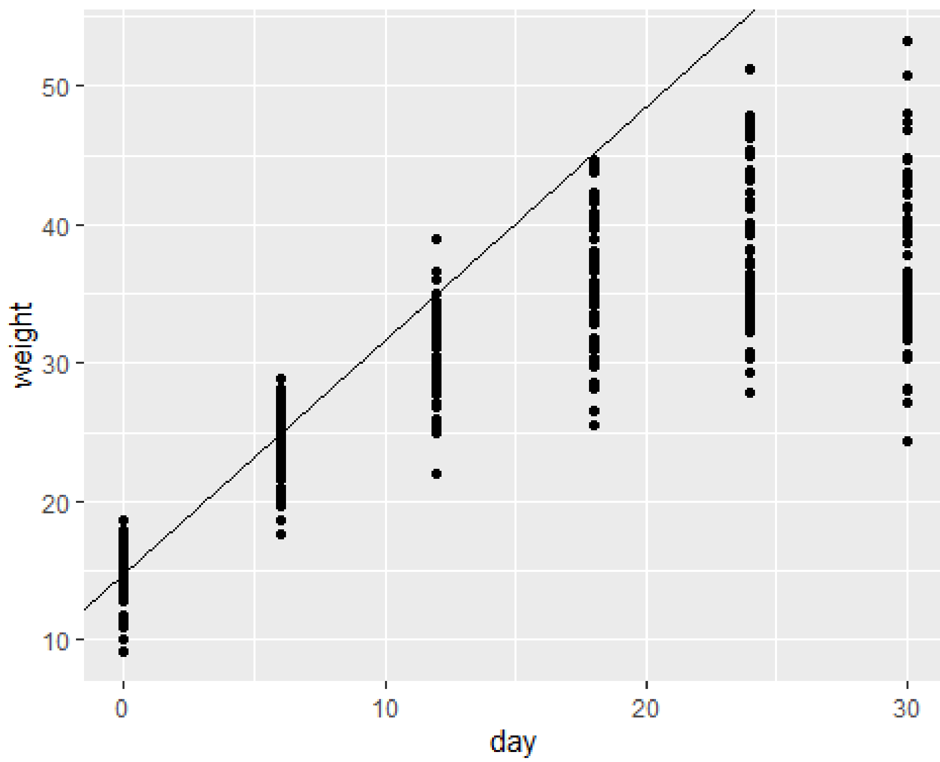
```
fit6 <- brm(formula = weight ~  day + day2, data = pig,
            family = "gaussian",
            prior = c(set_prior("normal(0,100)", class = "b"),
                      #set_prior("normal(0,100)", class = "a"),
                      set_prior("gamma(1.1,0.5)", class = "sigma")),
            warmup = 1000, iter = 5000, chains = 4,
            #control = list(adapt_delta = 0.98),
            save_pars = save_pars(all = TRUE), silent = TRUE)





(fit6)

  Family: gaussian
   Links: mu = identity; sigma = identity
 Formula: weight ~ day + day2
    Data: pig (Number of observations: 360)
 Samples: 4 chains, each with iter = 5000; warmup = 1000; thin = 1;
          total post-warmup samples = 16000

 Population-Level Effects:
```

```
           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept     14.74      0.50    13.76    15.71 1.00    10405     9927
day            1.72      0.08     1.57     1.87 1.00     8871     8980
day2          -0.03      0.00    -0.04    -0.03 1.00     9148     9265

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     4.28      0.16     3.98     4.61 1.00    10447     9294

Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
# Convergence and ESS are good
chains <- as.matrix(fit6)
dim(chains)
sims <- as.mcmc(chains)
raftery.diag(sims)
effectiveSize(sims)

# looic
loo(fit6)
```

16. Using the estimated coefficients from either the JAGS or the brm model, add a best fit line to the plot you made in problem 12.

```
ggplot(data = pig, mapping  = aes(x = day, y = weight)) +
  geom_point() +
  geom_abline(intercept = 14.73, slope = 1.69)
```



17. Using JAGS, run a model with an intercept, linear slope, and quadratic slope for each of the trt's. That is, you will be computing 9 β's, 3 for each trt. Assume all the data are independent (that is, we are not worried about the different pigs at this point). Use dnorm(0,0.0001) priors for the coefficients of the model (the β's), and a dgamma(1.1,.1) as the prior for σ^2_{error}. While you have now accounted for the different treatments, you are still ignoring that there are multiple measurements on each animal.
    What is the DIC for this model?

DIC is 1977.4

```
mdl <- "
  model {

  for (i in 1:360){
    weight[i] ~ dnorm(mu[i], 1/vv)
    mu[i] <- b0[trt[i]] + b1[trt[i]]*day[i] + b2[trt[i]]*day2[i]
  }

  for(i in 1:3){
    b0[i] ~ dnorm(0, 0.0001)
    b1[i] ~ dnorm(0, 0.0001)
    b2[i] ~ dnorm(0, 0.0001)
  }

  vv ~ dgamma(1.1, 0.1)
```

```
  }

"
trt = pig$trt
day = pig$day
day2 = pig$day2
weight = pig$weight

writeLines(mdl, 'fit7.txt')


data.jags <- c('weight', 'day', 'day2', 'trt')
parms <- c('b0' , 'b1', 'b2', 'vv')

fit7 <- jags(data= data.jags, parameters.to.save = parms,
                  model.file = 'fit7.txt', inits = NULL,
                  n.iter = 12000, n.thin = 5, n.burnin = 2000,
                  n.chains = 5)

 Compiling model graph
    Resolving undeclared variables
    Allocating nodes
 Graph information:
    Observed stochastic nodes: 360
    Unobserved stochastic nodes: 10
    Total graph size: 1510

 Initializing model

fit7

 Inference for Bugs model at "fit7.txt", fit using jags,
  5 chains, each with 12000 iterations (first 2000 discarded), n.thin = 5
  n.sims = 10000 iterations saved
          mu.vect sd.vect     2.5%      25%      50%      75%    97.5%  Rhat
 b0[1]      14.902   0.763   13.410   14.384   14.899   15.413   16.410 1.001
 b0[2]      14.954   0.750   13.495   14.448   14.953   15.454   16.436 1.001
 b0[3]      14.360   0.759   12.903   13.841   14.362   14.868   15.840 1.001
 b1[1]       1.723   0.119    1.487    1.643    1.725    1.804    1.958 1.001
 b1[2]       2.011   0.118    1.783    1.930    2.012    2.092    2.240 1.001
 b1[3]       1.424   0.119    1.192    1.343    1.423    1.506    1.653 1.001
 b2[1]      -0.029   0.004   -0.036   -0.031   -0.029   -0.026   -0.021 1.001
 b2[2]      -0.040   0.004   -0.048   -0.043   -0.040   -0.038   -0.033 1.001
 b2[3]      -0.025   0.004   -0.033   -0.028   -0.025   -0.023   -0.018 1.001
 vv         13.847   1.037   11.948   13.130   13.800   14.527   15.990 1.001
 deviance 1967.114   4.509 1960.282 1963.829 1966.427 1969.665 1977.698 1.001
         n.eff
 b0[1]   10000
 b0[2]    5200
 b0[3]   10000
```

```
 b1[1]    10000
 b1[2]     5400
 b1[3]    10000
 b2[1]    10000
 b2[2]     8800
 b2[3]    10000
 vv        9300
 deviance  6700

 For each parameter, n.eff is a crude measure of effective sample size,
 and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

 DIC info (using the rule, pD = var(deviance)/2)
 pD = 10.2 and DIC = 1977.3
 DIC is an estimate of expected predictive error (lower deviance is better).
```

```
# Diagnostics are good
sims <- as.mcmc(fit7)
chains <- as.matrix(sims)
sims <- as.mcmc(chains)
raftery.diag(sims)
effectiveSize(sims)
```

18. Repeat this analysis using brm. You will need to create a factor variable from trt (that is, let a variable be as.factor(pig$trt)). Use the same priors for the β's and a gamma(1.1,.5) as the prior for σ^2_{error}. Print a summary of the model. What is the looic?

looic is 1978.6

```
pig$trtf = as.factor(pig$trt)
fit8 <- brm(formula = weight ~  -1 + trtf + trtf:day + trtf:day2 , data =
pig,
            family = "gaussian",
            prior = c(set_prior("normal(0,100)", class = "b"),
                      set_prior("gamma(1.1,0.5)", class = "sigma")),
            warmup = 1000, iter = 5000, chains = 4,
            #control = list(adapt_delta = 0.98),
            save_pars = save_pars(all = TRUE))




summary(fit8)

  Family: gaussian
   Links: mu = identity; sigma = identity
 Formula: weight ~ -1 + trtf + trtf:day + trtf:day2
    Data: pig (Number of observations: 360)
 Samples: 4 chains, each with iter = 5000; warmup = 1000; thin = 1;
          total post-warmup samples = 16000
```

```
Population-Level Effects:
           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
trtf1         14.91      0.75    13.46    16.38 1.00     9090    10205
trtf2         14.97      0.76    13.48    16.45 1.00     9245    10474
trtf3         14.36      0.75    12.88    15.84 1.00     8936    10113
trtf1:day      1.72      0.12     1.49     1.95 1.00     8112     9099
trtf2:day      2.01      0.12     1.78     2.24 1.00     7920     9231
trtf3:day      1.42      0.12     1.19     1.66 1.00     7623     8821
trtf1:day2    -0.03      0.00    -0.04    -0.02 1.00     8999     9997
trtf2:day2    -0.04      0.00    -0.05    -0.03 1.00     8704    10182
trtf3:day2    -0.03      0.00    -0.03    -0.02 1.00     8326     9337

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     3.72      0.14     3.45     4.00 1.00    13182    10252

Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
# Convergence and ESS are good
chains <- as.matrix(fit8)
dim(chains)
sims <- as.mcmc(chains)
raftery.diag(sims)
effectiveSize(sims)

# looic
loo(fit8)
```

```
Computed from 16000 by 360 log-likelihood matrix

         Estimate   SE
elpd_loo   -989.4 15.5
p_loo        11.2  1.3
looic      1978.7 30.9
------
Monte Carlo SE of elpd_loo is 0.0.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

19. Now we are going to take into account that there are multiple measures on each animal to properly account for this information when we make inference. Using JAGS, create a term for a random deviation from the intercept for the intercepts only (this is sometimes referred to as a random coefficients approach). That is, we are assuming each pig is a random draw from the population of all pigs. You will need to account for the numbering of the pigs in the data set, since pig 1 in trt1 is not the same as pig 1 in trt 2. In this model you will be estimating 60 intercepts (20 pigs in each of the three

treatments). Use a gamma(1.1,.1) prior for $\sigma^2_{intercepts}$. You will also need a prior for all the terms that are deviations from the overall intercept. Use a normal with a mean of 0, and a precision that is $1/\sigma^2_{intercepts}$. What is the DIC?

DIC is 1696.2

```
pig$newpig <- rep(1:60, each = 6)


mdl <- "
  model {

  for (i in 1:360){
    weight[i] ~ dnorm(mu[i], 1/vv)
    mu[i] <- b0[trt[i]] + b1[trt[i]]*day[i] + b2[trt[i]]*day2[i] +
u0[newpig[i]]
  }

  for(i in 1:3){
    b0[i] ~ dnorm(0, 0.0001)
    b1[i] ~ dnorm(0, 0.0001)
    b2[i] ~ dnorm(0, 0.0001)
  }

  for(i in 1:60){
    u0[i] ~ dnorm(0,1/vvint)
  }

  vvint ~ dgamma(1.1, 0.1)
  vv ~ dgamma(1.1, 0.1)
}

"
newpig <- pig$newpig
trt = pig$trt
day = pig$day
day2 = pig$day2
weight = pig$weight

writeLines(mdl, 'fit9.txt')


data.jags <- c('weight', 'day', 'day2', 'trt', 'newpig')
parms <- c('b0' , 'b1', 'b2','u0', 'vv', 'vvint')

fit9 <- jags(data= data.jags, parameters.to.save = parms,
                  model.file = 'fit9.txt', inits = NULL,
                  n.iter = 8000, n.thin = 5, n.burnin = 2000,
                  n.chains = 5)
```

```
 Compiling model graph
    Resolving undeclared variables
    Allocating nodes
 Graph information:
    Observed stochastic nodes: 360
    Unobserved stochastic nodes: 71
    Total graph size: 2274

 Initializing model

fit9

 Inference for Bugs model at "fit9.txt", fit using jags,
  5 chains, each with 8000 iterations (first 2000 discarded), n.thin = 5
  n.sims = 6000 iterations saved
          mu.vect sd.vect     2.5%      25%      50%      75%    97.5%  Rhat
 b0[1]     14.898   0.821   13.278   14.353   14.894   15.461   16.519 1.001
 b0[2]     14.968   0.820   13.363   14.414   14.981   15.518   16.566 1.001
 b0[3]     14.365   0.801   12.787   13.815   14.386   14.914   15.910 1.001
 b1[1]      1.724   0.073    1.583    1.674    1.725    1.774    1.868 1.001
 b1[2]      2.011   0.073    1.871    1.960    2.011    2.061    2.152 1.002
 b1[3]      1.423   0.073    1.283    1.374    1.425    1.472    1.566 1.001
 b2[1]     -0.029   0.002   -0.034   -0.031   -0.029   -0.027   -0.024 1.001
 b2[2]     -0.040   0.002   -0.045   -0.042   -0.040   -0.039   -0.036 1.002
 b2[3]     -0.025   0.002   -0.030   -0.027   -0.025   -0.024   -0.021 1.001
 u0[1]     -3.576   1.094   -5.708   -4.315   -3.566   -2.838   -1.444 1.001
 u0[2]      1.733   1.117   -0.469    0.995    1.723    2.480    3.952 1.001
 u0[3]     -4.090   1.099   -6.218   -4.817   -4.104   -3.375   -1.949 1.001
 u0[4]      1.692   1.099   -0.488    0.953    1.692    2.421    3.790 1.001
 u0[5]      5.016   1.114    2.817    4.260    5.027    5.788    7.197 1.002
 u0[6]      0.967   1.105   -1.159    0.217    0.963    1.722    3.106 1.002
 u0[7]      0.298   1.110   -1.901   -0.435    0.302    1.038    2.510 1.001
 u0[8]     -1.753   1.108   -3.952   -2.504   -1.748   -1.012    0.420 1.001
 u0[9]     -2.993   1.088   -5.200   -3.713   -2.989   -2.245   -0.930 1.001
 u0[10]    -1.328   1.095   -3.429   -2.069   -1.333   -0.586    0.817 1.001
 u0[11]    -1.979   1.103   -4.109   -2.729   -1.981   -1.241    0.209 1.001
 u0[12]     1.527   1.110   -0.632    0.786    1.516    2.254    3.743 1.001
 u0[13]     1.276   1.110   -0.869    0.526    1.284    2.014    3.463 1.001
 u0[14]    -2.243   1.108   -4.467   -2.962   -2.243   -1.502   -0.110 1.001
 u0[15]     4.099   1.095    1.977    3.362    4.091    4.828    6.264 1.001
 u0[16]     6.148   1.126    4.001    5.367    6.127    6.898    8.371 1.001
 u0[17]    -0.767   1.117   -2.986   -1.525   -0.748   -0.006    1.332 1.001
 u0[18]    -2.547   1.107   -4.738   -3.285   -2.562   -1.801   -0.338 1.001
 u0[19]     0.237   1.109   -1.922   -0.502    0.224    0.962    2.432 1.001
 u0[20]    -1.726   1.095   -3.841   -2.458   -1.747   -0.980    0.390 1.001
 u0[21]     1.593   1.082   -0.560    0.867    1.592    2.312    3.711 1.001
 u0[22]    -2.463   1.099   -4.634   -3.184   -2.460   -1.739   -0.300 1.001
 u0[23]    -3.352   1.108   -5.514   -4.092   -3.354   -2.590   -1.177 1.001
 u0[24]     0.643   1.094   -1.507   -0.096    0.653    1.364    2.779 1.001
 u0[25]     6.219   1.108    4.049    5.461    6.208    6.982    8.377 1.001
```

```
u0[26]      -0.426   1.095   -2.610   -1.149   -0.433    0.317    1.713 1.001
u0[27]       4.053   1.109    1.890    3.303    4.048    4.784    6.236 1.001
u0[28]       1.416   1.091   -0.717    0.681    1.406    2.156    3.570 1.001
u0[29]      -3.591   1.107   -5.729   -4.337   -3.586   -2.845   -1.388 1.001
u0[30]      -2.642   1.107   -4.757   -3.404   -2.646   -1.883   -0.482 1.001
u0[31]       1.797   1.101   -0.317    1.042    1.782    2.539    3.971 1.001
u0[32]       3.324   1.121    1.166    2.563    3.327    4.076    5.525 1.001
u0[33]       1.965   1.093   -0.117    1.208    1.957    2.705    4.130 1.001
u0[34]      -2.076   1.105   -4.229   -2.826   -2.058   -1.321    0.034 1.002
u0[35]       2.674   1.106    0.512    1.946    2.662    3.429    4.853 1.001
u0[36]      -3.705   1.100   -5.841   -4.456   -3.700   -2.968   -1.527 1.001
u0[37]      -1.880   1.099   -3.996   -2.623   -1.882   -1.150    0.276 1.001
u0[38]      -3.518   1.102   -5.682   -4.275   -3.512   -2.767   -1.397 1.001
u0[39]       2.672   1.093    0.508    1.948    2.660    3.397    4.800 1.001
u0[40]      -2.796   1.107   -4.960   -3.545   -2.812   -2.050   -0.596 1.001
u0[41]       0.647   1.092   -1.473   -0.082    0.654    1.376    2.834 1.001
u0[42]      -1.767   1.096   -3.932   -2.495   -1.758   -1.016    0.369 1.001
u0[43]      -4.827   1.096   -6.974   -5.587   -4.825   -4.064   -2.713 1.001
u0[44]      -3.760   1.093   -5.862   -4.511   -3.788   -3.029   -1.576 1.001
u0[45]       0.597   1.097   -1.587   -0.132    0.594    1.337    2.715 1.001
u0[46]       5.018   1.090    2.897    4.276    4.999    5.737    7.147 1.001
u0[47]       0.688   1.090   -1.455   -0.041    0.698    1.445    2.791 1.001
u0[48]      -2.065   1.073   -4.181   -2.774   -2.056   -1.353    0.013 1.002
u0[49]       0.957   1.089   -1.181    0.232    0.945    1.691    3.093 1.001
u0[50]       1.305   1.096   -0.859    0.566    1.313    2.052    3.454 1.001
u0[51]      -3.020   1.109   -5.163   -3.784   -3.017   -2.255   -0.813 1.001
u0[52]       1.603   1.096   -0.546    0.869    1.584    2.347    3.787 1.001
u0[53]      -0.017   1.089   -2.155   -0.764   -0.014    0.723    2.063 1.001
u0[54]       3.511   1.078    1.354    2.777    3.526    4.227    5.649 1.001
u0[55]       2.233   1.085    0.147    1.496    2.227    2.963    4.388 1.001
u0[56]      -2.155   1.101   -4.302   -2.895   -2.143   -1.403   -0.011 1.001
u0[57]      -2.569   1.097   -4.739   -3.285   -2.581   -1.836   -0.401 1.001
u0[58]      -0.649   1.104   -2.804   -1.406   -0.637    0.099    1.515 1.001
u0[59]       4.303   1.096    2.187    3.553    4.284    5.059    6.481 1.001
u0[60]      -0.296   1.105   -2.398   -1.066   -0.301    0.442    1.887 1.001
vv           5.243   0.437    4.444    4.935    5.221    5.527    6.147 1.001
vvint        9.174   1.907    6.139    7.827    8.947   10.220   13.607 1.001
deviance  1615.463  12.810 1592.802 1606.477 1614.755 1623.356 1642.846 1.001
         n.eff
b0[1]     6000
b0[2]     5700
b0[3]     6000
b1[1]     6000
b1[2]     2500
b1[3]     6000
b2[1]     6000
b2[2]     2800
b2[3]     6000
u0[1]     3300
u0[2]     4300
```

```
u0[3]     6000
u0[4]     6000
u0[5]     1700
u0[6]     2900
u0[7]     6000
u0[8]     5800
u0[9]     5500
u0[10]    3500
u0[11]    4500
u0[12]    6000
u0[13]    3300
u0[14]    6000
u0[15]    6000
u0[16]    6000
u0[17]    6000
u0[18]    6000
u0[19]    6000
u0[20]    6000
u0[21]    4800
u0[22]    6000
u0[23]    6000
u0[24]    4900
u0[25]    5500
u0[26]    6000
u0[27]    5900
u0[28]    6000
u0[29]    6000
u0[30]    6000
u0[31]    6000
u0[32]    4200
u0[33]    6000
u0[34]    2800
u0[35]    6000
u0[36]    3800
u0[37]    6000
u0[38]    6000
u0[39]    6000
u0[40]    6000
u0[41]    4900
u0[42]    6000
u0[43]    6000
u0[44]    4400
u0[45]    6000
u0[46]    6000
u0[47]    6000
u0[48]    2200
u0[49]    6000
u0[50]    6000
u0[51]    3300
u0[52]    6000
```

```
u0[53]    5300
u0[54]    6000
u0[55]    4100
u0[56]    6000
u0[57]    4200
u0[58]    6000
u0[59]    6000
u0[60]    5800
vv        6000
vvint     6000
deviance  5500

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 82.0 and DIC = 1697.5
DIC is an estimate of expected predictive error (lower deviance is better).
```

```
# Diagnostics are good
sims <- as.mcmc(fit9)
chains <- as.matrix(sims)
sims <- as.mcmc(chains)
raftery.diag(sims)
effectiveSize(sims)
```

20. Continuing in JAGS, add a random coefficient term for the deviations from the linear term in the model. You will now have random coefficients for both the intercepts and the linear term (slopes). You can use a gamma(1.1,.1) prior for σ^2_{linear}. You will also need a prior for all the terms that are deviations from the overall linear term (slopes) . Use a normal with a mean of 0, and a precision that is $1/\sigma^2_{linear}$. What is the DIC?

DIC is 1451.0

```
mdl <- "
  model {

  for (i in 1:360){
    weight[i] ~ dnorm(mu[i], 1/vv)
    mu[i] <- b0[trt[i]] + b1[trt[i]]*day[i] + b2[trt[i]]*day2[i] +
u0[newpig[i]] + u1[newpig[i]]*day[i]
  }

  for(i in 1:3){
    b0[i] ~ dnorm(0, 0.0001)
    b1[i] ~ dnorm(0, 0.0001)
    b2[i] ~ dnorm(0, 0.0001)
  }

  for(i in 1:60){
```

```
    u0[i] ~ dnorm(0,1/vvint)
    u1[i] ~ dnorm(0, 1/vvslp)
  }

  vvslp ~ dgamma(1.1, 0.1)
  vvint ~ dgamma(1.1, 0.1)
  vv ~ dgamma(1.1, 0.1)
}

"
newpig <- pig$newpig
trt = pig$trt
day = pig$day
day2 = pig$day2
weight = pig$weight

writeLines(mdl, 'fit10.txt')


data.jags <- c('weight', 'day', 'day2', 'trt', 'newpig')
parms <- c('b0' , 'b1', 'b2','u0','u1', 'vv', 'vvint', 'vvslp')

fit10 <- jags(data= data.jags, parameters.to.save = parms,
                  model.file = 'fit10.txt', inits = NULL,
                  n.iter = 20000, n.thin = 5, n.burnin = 2000,
                  n.chains = 5)

 Compiling model graph
    Resolving undeclared variables
    Allocating nodes
 Graph information:
    Observed stochastic nodes: 360
    Unobserved stochastic nodes: 132
    Total graph size: 2696

 Initializing model

fit10

 Inference for Bugs model at "fit10.txt", fit using jags,
  5 chains, each with 20000 iterations (first 2000 discarded), n.thin = 5
  n.sims = 18000 iterations saved
          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat
 b0[1]     14.899   0.386  14.148  14.639  14.898  15.156  15.677 1.001
 b0[2]     14.954   0.385  14.205  14.698  14.952  15.211  15.711 1.001
 b0[3]     14.356   0.387  13.602  14.098  14.355  14.616  15.123 1.001
 b1[1]      1.725   0.061   1.603   1.684   1.724   1.766   1.844 1.001
 b1[2]      2.010   0.061   1.889   1.969   2.011   2.051   2.131 1.001
 b1[3]      1.424   0.062   1.303   1.383   1.424   1.465   1.546 1.001
 b2[1]     -0.029   0.001  -0.032  -0.030  -0.029  -0.028  -0.026 1.001
```

b2[2]	-0.040	0.002	-0.043	-0.041	-0.040	-0.039	-0.037 1.001
b2[3]	-0.025	0.002	-0.028	-0.026	-0.025	-0.024	-0.022 1.001
u0[1]	-0.624	0.773	-2.200	-1.134	-0.610	-0.100	0.851 1.001
u0[2]	-0.037	0.763	-1.529	-0.544	-0.033	0.474	1.471 1.001
u0[3]	-0.712	0.772	-2.245	-1.223	-0.707	-0.193	0.788 1.001
u0[4]	0.967	0.790	-0.545	0.422	0.954	1.494	2.568 1.001
u0[5]	0.948	0.781	-0.538	0.413	0.936	1.462	2.516 1.001
u0[6]	0.398	0.768	-1.108	-0.118	0.396	0.912	1.929 1.001
u0[7]	-1.181	0.799	-2.798	-1.713	-1.166	-0.629	0.325 1.001
u0[8]	0.168	0.757	-1.294	-0.337	0.167	0.668	1.654 1.001
u0[9]	0.066	0.766	-1.443	-0.441	0.071	0.568	1.584 1.001
u0[10]	-0.773	0.777	-2.334	-1.285	-0.764	-0.249	0.717 1.001
u0[11]	0.094	0.764	-1.421	-0.417	0.092	0.601	1.596 1.001
u0[12]	-0.175	0.761	-1.693	-0.682	-0.160	0.333	1.305 1.001
u0[13]	0.287	0.764	-1.185	-0.237	0.283	0.794	1.805 1.001
u0[14]	-0.183	0.763	-1.702	-0.687	-0.177	0.324	1.325 1.001
u0[15]	1.270	0.799	-0.227	0.717	1.248	1.798	2.899 1.001
u0[16]	1.245	0.800	-0.273	0.698	1.225	1.773	2.870 1.001
u0[17]	-0.279	0.761	-1.775	-0.782	-0.275	0.225	1.213 1.001
u0[18]	-0.357	0.763	-1.893	-0.860	-0.350	0.149	1.137 1.001
u0[19]	0.073	0.766	-1.432	-0.434	0.074	0.580	1.599 1.001
u0[20]	-1.206	0.801	-2.822	-1.733	-1.190	-0.660	0.345 1.001
u0[21]	0.364	0.762	-1.127	-0.148	0.363	0.868	1.859 1.001
u0[22]	0.400	0.765	-1.082	-0.106	0.385	0.904	1.932 1.001
u0[23]	-0.592	0.770	-2.131	-1.101	-0.586	-0.065	0.877 1.001
u0[24]	0.029	0.761	-1.481	-0.475	0.027	0.531	1.534 1.001
u0[25]	1.336	0.807	-0.187	0.783	1.319	1.862	2.981 1.001
u0[26]	-0.716	0.773	-2.278	-1.227	-0.707	-0.196	0.767 1.001
u0[27]	0.511	0.771	-0.996	-0.007	0.503	1.021	2.042 1.001
u0[28]	0.340	0.763	-1.140	-0.171	0.329	0.838	1.870 1.001
u0[29]	-0.880	0.774	-2.452	-1.395	-0.867	-0.351	0.598 1.001
u0[30]	-0.271	0.763	-1.787	-0.777	-0.271	0.235	1.219 1.001
u0[31]	-0.358	0.767	-1.888	-0.863	-0.357	0.151	1.159 1.001
u0[32]	0.238	0.760	-1.245	-0.273	0.232	0.744	1.734 1.001
u0[33]	0.671	0.772	-0.807	0.152	0.663	1.178	2.212 1.001
u0[34]	-0.377	0.759	-1.894	-0.883	-0.365	0.128	1.104 1.001
u0[35]	0.584	0.767	-0.891	0.069	0.567	1.092	2.126 1.001
u0[36]	-0.902	0.780	-2.493	-1.402	-0.887	-0.374	0.589 1.001
u0[37]	0.092	0.768	-1.395	-0.422	0.087	0.605	1.620 1.001
u0[38]	-0.820	0.779	-2.397	-1.333	-0.801	-0.294	0.671 1.001
u0[39]	0.729	0.777	-0.751	0.197	0.712	1.248	2.291 1.001
u0[40]	-0.355	0.762	-1.865	-0.856	-0.341	0.157	1.116 1.001
u0[41]	1.141	0.789	-0.362	0.601	1.120	1.667	2.738 1.001
u0[42]	-0.702	0.775	-2.275	-1.207	-0.689	-0.175	0.781 1.001
u0[43]	-1.075	0.794	-2.696	-1.597	-1.055	-0.532	0.424 1.001
u0[44]	-1.759	0.836	-3.466	-2.307	-1.741	-1.182	-0.180 1.001
u0[45]	1.211	0.795	-0.309	0.672	1.190	1.730	2.841 1.001
u0[46]	0.922	0.781	-0.559	0.388	0.903	1.436	2.511 1.001
u0[47]	0.569	0.773	-0.924	0.052	0.555	1.076	2.118 1.001
u0[48]	-0.067	0.764	-1.573	-0.571	-0.063	0.441	1.427 1.001

u0[49]	1.041	0.795	-0.477	0.492	1.025	1.564	2.649	1.001
u0[50]	0.364	0.757	-1.109	-0.141	0.362	0.859	1.878	1.001
u0[51]	-1.124	0.791	-2.717	-1.647	-1.095	-0.587	0.376	1.001
u0[52]	0.226	0.768	-1.276	-0.290	0.221	0.734	1.758	1.001
u0[53]	-0.128	0.761	-1.611	-0.629	-0.132	0.376	1.349	1.001
u0[54]	0.490	0.774	-0.994	-0.030	0.478	0.993	2.041	1.001
u0[55]	-0.421	0.767	-1.964	-0.926	-0.416	0.091	1.091	1.001
u0[56]	-1.617	0.822	-3.289	-2.165	-1.591	-1.050	-0.084	1.001
u0[57]	0.045	0.765	-1.463	-0.455	0.045	0.555	1.545	1.001
u0[58]	-0.551	0.769	-2.078	-1.062	-0.547	-0.035	0.936	1.001
u0[59]	0.971	0.784	-0.525	0.437	0.956	1.487	2.558	1.001
u0[60]	0.426	0.768	-1.064	-0.091	0.414	0.927	1.982	1.001
u1[1]	-0.210	0.060	-0.327	-0.250	-0.209	-0.170	-0.093	1.001
u1[2]	0.129	0.060	0.012	0.088	0.129	0.170	0.246	1.001
u1[3]	-0.237	0.060	-0.354	-0.278	-0.237	-0.198	-0.118	1.001
u1[4]	0.036	0.060	-0.082	-0.004	0.037	0.077	0.153	1.001
u1[5]	0.284	0.060	0.164	0.243	0.284	0.324	0.402	1.001
u1[6]	0.037	0.060	-0.082	-0.004	0.036	0.078	0.154	1.001
u1[7]	0.125	0.060	0.006	0.084	0.124	0.165	0.243	1.001
u1[8]	-0.144	0.059	-0.261	-0.184	-0.143	-0.104	-0.027	1.001
u1[9]	-0.225	0.060	-0.342	-0.265	-0.224	-0.184	-0.108	1.001
u1[10]	-0.030	0.060	-0.147	-0.071	-0.030	0.010	0.086	1.001
u1[11]	-0.153	0.060	-0.269	-0.193	-0.153	-0.112	-0.036	1.001
u1[12]	0.126	0.059	0.012	0.086	0.126	0.166	0.242	1.001
u1[13]	0.068	0.060	-0.047	0.028	0.068	0.108	0.186	1.001
u1[14]	-0.147	0.060	-0.264	-0.187	-0.147	-0.107	-0.030	1.001
u1[15]	0.190	0.060	0.070	0.149	0.190	0.230	0.307	1.001
u1[16]	0.341	0.061	0.222	0.300	0.341	0.381	0.462	1.001
u1[17]	-0.031	0.059	-0.147	-0.071	-0.032	0.009	0.085	1.001
u1[18]	-0.154	0.060	-0.271	-0.194	-0.154	-0.114	-0.037	1.001
u1[19]	0.009	0.060	-0.108	-0.031	0.009	0.049	0.126	1.001
u1[20]	-0.020	0.061	-0.139	-0.061	-0.020	0.022	0.100	1.001
u1[21]	0.084	0.060	-0.032	0.045	0.084	0.124	0.201	1.001
u1[22]	-0.214	0.060	-0.331	-0.254	-0.214	-0.173	-0.097	1.001
u1[23]	-0.194	0.059	-0.311	-0.234	-0.195	-0.155	-0.076	1.001
u1[24]	0.046	0.059	-0.070	0.006	0.046	0.086	0.162	1.001
u1[25]	0.337	0.061	0.218	0.295	0.337	0.377	0.458	1.001
u1[26]	0.031	0.060	-0.084	-0.009	0.031	0.072	0.149	1.001
u1[27]	0.253	0.060	0.137	0.212	0.252	0.293	0.370	1.001
u1[28]	0.077	0.060	-0.039	0.036	0.077	0.117	0.194	1.001
u1[29]	-0.187	0.060	-0.303	-0.227	-0.187	-0.147	-0.069	1.001
u1[30]	-0.171	0.060	-0.288	-0.211	-0.171	-0.130	-0.053	1.001
u1[31]	0.164	0.060	0.046	0.124	0.164	0.203	0.281	1.001
u1[32]	0.223	0.059	0.107	0.184	0.223	0.263	0.339	1.001
u1[33]	0.086	0.060	-0.031	0.046	0.086	0.126	0.205	1.001
u1[34]	-0.119	0.059	-0.237	-0.159	-0.119	-0.079	-0.004	1.001
u1[35]	0.146	0.060	0.030	0.106	0.146	0.186	0.263	1.001
u1[36]	-0.193	0.060	-0.309	-0.233	-0.193	-0.152	-0.076	1.001
u1[37]	-0.145	0.060	-0.261	-0.185	-0.145	-0.105	-0.029	1.001
u1[38]	-0.185	0.060	-0.302	-0.226	-0.185	-0.144	-0.068	1.001

```
 u1[39]       0.131   0.059    0.016    0.091    0.131    0.171    0.248 1.001
 u1[40]      -0.175   0.059   -0.291   -0.215   -0.175   -0.135   -0.059 1.001
 u1[41]      -0.051   0.060   -0.171   -0.093   -0.051   -0.011    0.066 1.001
 u1[42]      -0.066   0.060   -0.186   -0.106   -0.066   -0.026    0.052 1.001
 u1[43]      -0.259   0.061   -0.378   -0.300   -0.259   -0.218   -0.140 1.001
 u1[44]      -0.121   0.062   -0.242   -0.162   -0.121   -0.079    0.000 1.001
 u1[45]      -0.061   0.061   -0.179   -0.103   -0.061   -0.021    0.057 1.001
 u1[46]       0.289   0.060    0.170    0.248    0.289    0.329    0.408 1.001
 u1[47]       0.000   0.060   -0.120   -0.039    0.000    0.041    0.120 1.001
 u1[48]      -0.145   0.060   -0.262   -0.185   -0.145   -0.105   -0.027 1.001
 u1[49]      -0.021   0.061   -0.140   -0.062   -0.022    0.019    0.098 1.001
 u1[50]       0.064   0.060   -0.054    0.024    0.064    0.104    0.182 1.001
 u1[51]      -0.122   0.060   -0.238   -0.162   -0.122   -0.082   -0.004 1.001
 u1[52]       0.098   0.060   -0.018    0.058    0.098    0.138    0.216 1.001
 u1[53]       0.010   0.060   -0.107   -0.030    0.009    0.050    0.128 1.001
 u1[54]       0.215   0.060    0.101    0.175    0.215    0.255    0.334 1.001
 u1[55]       0.202   0.060    0.085    0.161    0.202    0.242    0.320 1.001
 u1[56]      -0.016   0.061   -0.134   -0.058   -0.016    0.025    0.108 1.001
 u1[57]      -0.190   0.060   -0.307   -0.230   -0.191   -0.150   -0.073 1.001
 u1[58]       0.002   0.060   -0.115   -0.038    0.001    0.042    0.121 1.001
 u1[59]       0.230   0.061    0.113    0.188    0.229    0.271    0.349 1.001
 u1[60]      -0.059   0.060   -0.177   -0.099   -0.059   -0.020    0.058 1.001
 vv           2.154   0.193    1.808    2.016    2.144    2.276    2.563 1.001
 vvint        1.233   0.460    0.491    0.907    1.180    1.498    2.278 1.001
 vvslp        0.031   0.007    0.021    0.027    0.031    0.035    0.047 1.001
 deviance 1294.889  17.733 1262.084 1282.569 1294.252 1306.464 1331.381 1.001



 DIC info (using the rule, pD = var(deviance)/2)
 pD = 157.2 and DIC = 1452.1
 DIC is an estimate of expected predictive error (lower deviance is better).
```

```
# Diagnostics are good
sims <- as.mcmc(fit10)
chains <- as.matrix(sims)
sims <- as.mcmc(chains)
raftery.diag(sims)
effectiveSize(sims)
```

21. Continuing in JAGS, add a random coefficient term for the deviations from the quadratic term in the model. You will now have random coefficients for the intercepts, the linear term (slopes), and the quadratic term. You can use a gamma(1.1,.1) prior for $\sigma^2_{quadratic}$. You will also need a prior for all the terms that are deviations from the overall quadratic term. Use a normal with a mean of 0, and a precision that is $1/\sigma^2_{quadratic}$. What is the DIC?

DIC is 1476.5

```r
mdl <- "
  model {

  for (i in 1:360){
    weight[i] ~ dnorm(mu[i], 1/vv)
    mu[i] <- b0[trt[i]] + b1[trt[i]]*day[i] + b2[trt[i]]*day2[i] +
u0[newpig[i]] + u1[newpig[i]]*day[i] + u2[newpig[i]]*day2[i]
  }

  for(i in 1:3){
    b0[i] ~ dnorm(0, 0.0001)
    b1[i] ~ dnorm(0, 0.0001)
    b2[i] ~ dnorm(0, 0.0001)
  }

  for(i in 1:60){
    u0[i] ~ dnorm(0,1/vvint)
    u1[i] ~ dnorm(0, 1/vvslp)
    u2[i] ~ dnorm(0, 1/vvslp2)
  }

  vvslp2 ~ dgamma(1.1, 0.1)
  vvslp ~ dgamma(1.1, 0.1)
  vvint ~ dgamma(1.1, 0.1)
  vv ~ dgamma(1.1, 0.1)
}

"
newpig <- pig$newpig
trt = pig$trt
day = pig$day
day2 = pig$day2
weight = pig$weight

writeLines(mdl, 'fit11.txt')


data.jags <- c('weight', 'day', 'day2', 'trt', 'newpig')
parms <- c('b0' , 'b1', 'b2','u0','u1', 'vv', 'vvint', 'vvslp')

fit11 <- jags(data= data.jags, parameters.to.save = parms,
                  model.file = 'fit11.txt', inits = NULL,
                  n.iter = 30000, n.thin = 5, n.burnin = 2000,
                  n.chains = 5)

 Compiling model graph
    Resolving undeclared variables
    Allocating nodes
 Graph information:
```

```
   Observed stochastic nodes: 360
   Unobserved stochastic nodes: 193
   Total graph size: 3118

 Initializing model
```

fit11

```
 Inference for Bugs model at "fit11.txt", fit using jags,
  5 chains, each with 30000 iterations (first 2000 discarded), n.thin = 5
  n.sims = 28000 iterations saved
          mu.vect sd.vect   2.5%     25%     50%     75%   97.5%  Rhat
 b0[1]     14.896   0.398 14.115  14.632  14.893  15.162  15.673 1.001
 b0[2]     14.959   0.400 14.172  14.695  14.956  15.225  15.747 1.001
 b0[3]     14.351   0.397 13.582  14.084  14.353  14.617  15.129 1.001
 b1[1]      1.724   0.058  1.611   1.685   1.724   1.763   1.838 1.001
 b1[2]      2.010   0.058  1.898   1.971   2.010   2.049   2.122 1.001
 b1[3]      1.424   0.058  1.311   1.386   1.424   1.463   1.537 1.001
 b2[1]     -0.029   0.002 -0.032  -0.030  -0.029  -0.028  -0.026 1.001
 b2[2]     -0.040   0.002 -0.043  -0.041  -0.040  -0.039  -0.037 1.001
 b2[3]     -0.025   0.002 -0.028  -0.026  -0.025  -0.024  -0.022 1.001
 u0[1]     -0.794   0.809 -2.411  -1.336  -0.785  -0.252   0.783 1.001
 u0[2]      0.025   0.791 -1.523  -0.506   0.027   0.555   1.583 1.001
 u0[3]     -0.876   0.816 -2.503  -1.414  -0.860  -0.322   0.674 1.001
 u0[4]      1.103   0.815 -0.448   0.545   1.089   1.641   2.739 1.001
 u0[5]      1.195   0.831 -0.377   0.628   1.177   1.737   2.892 1.001
 u0[6]      0.426   0.797 -1.104  -0.115   0.417   0.954   2.034 1.001
 u0[7]     -1.233   0.814 -2.863  -1.778  -1.218  -0.676   0.329 1.001
 u0[8]      0.100   0.793 -1.458  -0.434   0.097   0.629   1.668 1.001
 u0[9]     -0.003   0.799 -1.583  -0.537  -0.001   0.530   1.561 1.001
 u0[10]    -0.878   0.797 -2.462  -1.407  -0.868  -0.336   0.653 1.001
 u0[11]     0.041   0.798 -1.540  -0.488   0.043   0.573   1.610 1.001
 u0[12]    -0.108   0.797 -1.677  -0.637  -0.099   0.426   1.466 1.001
 u0[13]     0.381   0.792 -1.163  -0.155   0.378   0.906   1.954 1.001
 u0[14]    -0.327   0.797 -1.926  -0.855  -0.319   0.210   1.222 1.001
 u0[15]     1.505   0.840 -0.099   0.920   1.492   2.068   3.171 1.001
 u0[16]     1.534   0.856 -0.074   0.944   1.510   2.101   3.255 1.001
 u0[17]    -0.307   0.795 -1.882  -0.833  -0.298   0.230   1.239 1.001
 u0[18]    -0.473   0.802 -2.062  -1.006  -0.469   0.067   1.080 1.001
 u0[19]     0.089   0.798 -1.459  -0.443   0.079   0.620   1.656 1.001
 u0[20]    -1.365   0.822 -3.034  -1.906  -1.349  -0.806   0.198 1.001
 u0[21]     0.439   0.798 -1.104  -0.094   0.429   0.967   2.037 1.001
 u0[22]     0.292   0.799 -1.292  -0.240   0.289   0.821   1.870 1.001
 u0[23]    -0.731   0.816 -2.367  -1.276  -0.715  -0.174   0.834 1.001
 u0[24]     0.041   0.793 -1.507  -0.488   0.038   0.564   1.612 1.001
 u0[25]     1.672   0.863  0.034   1.087   1.649   2.246   3.411 1.001
 u0[26]    -0.771   0.803 -2.373  -1.303  -0.753  -0.233   0.766 1.001
 u0[27]     0.707   0.808 -0.832   0.152   0.703   1.241   2.343 1.001
 u0[28]     0.384   0.800 -1.176  -0.152   0.385   0.925   1.954 1.001
 u0[29]    -1.078   0.821 -2.726  -1.628  -1.060  -0.515   0.484 1.001
```

u0[30]	-0.360	0.798	-1.950	-0.893	-0.347	0.172	1.200	1.001
u0[31]	-0.260	0.795	-1.832	-0.793	-0.259	0.275	1.298	1.001
u0[32]	0.405	0.803	-1.147	-0.140	0.395	0.935	1.995	1.001
u0[33]	0.750	0.809	-0.819	0.202	0.742	1.282	2.353	1.001
u0[34]	-0.482	0.791	-2.040	-1.014	-0.478	0.054	1.064	1.001
u0[35]	0.712	0.811	-0.857	0.158	0.700	1.256	2.359	1.001
u0[36]	-1.096	0.818	-2.722	-1.640	-1.085	-0.532	0.465	1.001
u0[37]	0.034	0.796	-1.533	-0.504	0.034	0.564	1.609	1.001
u0[38]	-1.007	0.819	-2.632	-1.560	-0.995	-0.447	0.568	1.001
u0[39]	0.879	0.811	-0.687	0.329	0.867	1.416	2.495	1.001
u0[40]	-0.486	0.798	-2.055	-1.016	-0.475	0.054	1.066	1.001
u0[41]	1.236	0.819	-0.334	0.676	1.218	1.770	2.886	1.001
u0[42]	-0.792	0.803	-2.393	-1.323	-0.782	-0.255	0.751	1.001
u0[43]	-1.276	0.830	-2.962	-1.821	-1.261	-0.710	0.313	1.001
u0[44]	-2.022	0.864	-3.760	-2.596	-2.003	-1.432	-0.385	1.001
u0[45]	1.322	0.817	-0.245	0.764	1.318	1.865	2.952	1.001
u0[46]	1.176	0.830	-0.399	0.613	1.156	1.720	2.852	1.001
u0[47]	0.563	0.807	-0.997	0.010	0.554	1.100	2.179	1.001
u0[48]	-0.174	0.798	-1.735	-0.707	-0.177	0.374	1.391	1.001
u0[49]	1.167	0.809	-0.370	0.615	1.156	1.697	2.812	1.001
u0[50]	0.428	0.794	-1.105	-0.113	0.424	0.962	2.001	1.001
u0[51]	-1.309	0.827	-2.970	-1.857	-1.296	-0.743	0.272	1.001
u0[52]	0.390	0.799	-1.142	-0.153	0.381	0.919	1.985	1.001
u0[53]	-0.089	0.788	-1.636	-0.615	-0.089	0.443	1.463	1.001
u0[54]	0.635	0.812	-0.931	0.082	0.628	1.167	2.267	1.001
u0[55]	-0.380	0.791	-1.929	-0.906	-0.377	0.153	1.173	1.001
u0[56]	-1.812	0.842	-3.505	-2.372	-1.791	-1.229	-0.220	1.001
u0[57]	-0.070	0.800	-1.653	-0.603	-0.064	0.465	1.497	1.001
u0[58]	-0.588	0.805	-2.190	-1.120	-0.576	-0.049	0.973	1.001
u0[59]	1.187	0.824	-0.377	0.624	1.166	1.734	2.851	1.001
u0[60]	0.437	0.796	-1.124	-0.103	0.437	0.969	1.997	1.001
u1[1]	-0.171	0.081	-0.325	-0.226	-0.173	-0.118	-0.005	1.001
u1[2]	0.105	0.078	-0.051	0.053	0.106	0.158	0.254	1.001
u1[3]	-0.206	0.081	-0.361	-0.261	-0.207	-0.153	-0.043	1.001
u1[4]	0.022	0.078	-0.132	-0.030	0.023	0.075	0.173	1.001
u1[5]	0.225	0.086	0.049	0.169	0.227	0.285	0.388	1.001
u1[6]	0.038	0.077	-0.114	-0.014	0.038	0.089	0.189	1.001
u1[7]	0.097	0.080	-0.062	0.043	0.097	0.151	0.252	1.001
u1[8]	-0.108	0.079	-0.260	-0.161	-0.109	-0.056	0.051	1.001
u1[9]	-0.198	0.080	-0.351	-0.251	-0.199	-0.145	-0.035	1.001
u1[10]	-0.019	0.078	-0.171	-0.071	-0.019	0.032	0.135	1.001
u1[11]	-0.124	0.079	-0.276	-0.178	-0.126	-0.072	0.033	1.001
u1[12]	0.089	0.080	-0.071	0.037	0.092	0.143	0.244	1.001
u1[13]	0.046	0.078	-0.112	-0.005	0.047	0.098	0.198	1.001
u1[14]	-0.083	0.084	-0.241	-0.141	-0.085	-0.028	0.089	1.001
u1[15]	0.150	0.082	-0.017	0.096	0.151	0.205	0.305	1.001
u1[16]	0.277	0.090	0.092	0.218	0.281	0.339	0.442	1.002
u1[17]	-0.036	0.077	-0.190	-0.087	-0.036	0.015	0.114	1.001
u1[18]	-0.117	0.081	-0.271	-0.171	-0.118	-0.064	0.045	1.001
u1[19]	0.014	0.077	-0.139	-0.038	0.014	0.065	0.165	1.001

```
u1[20]     -0.002   0.078   -0.153   -0.055   -0.002    0.050    0.155 1.001
u1[21]      0.070   0.078   -0.082    0.018    0.070    0.123    0.222 1.001
u1[22]     -0.145   0.085   -0.306   -0.204   -0.148   -0.089    0.028 1.002
u1[23]     -0.173   0.080   -0.328   -0.226   -0.174   -0.119   -0.012 1.001
u1[24]      0.045   0.077   -0.108   -0.006    0.044    0.095    0.198 1.001
u1[25]      0.261   0.091    0.073    0.201    0.266    0.324    0.430 1.001
u1[26]      0.026   0.077   -0.126   -0.025    0.026    0.078    0.178 1.001
u1[27]      0.192   0.085    0.019    0.137    0.196    0.250    0.353 1.001
u1[28]      0.083   0.077   -0.066    0.031    0.082    0.135    0.237 1.001
u1[29]     -0.131   0.084   -0.290   -0.188   -0.133   -0.075    0.039 1.001
u1[30]     -0.160   0.079   -0.312   -0.212   -0.161   -0.107   -0.002 1.001
u1[31]      0.100   0.084   -0.072    0.046    0.103    0.157    0.257 1.001
u1[32]      0.157   0.086   -0.020    0.101    0.161    0.217    0.320 1.001
u1[33]      0.088   0.078   -0.063    0.036    0.088    0.140    0.242 1.001
u1[34]     -0.088   0.078   -0.239   -0.141   -0.089   -0.037    0.069 1.001
u1[35]      0.122   0.079   -0.037    0.070    0.122    0.175    0.275 1.001
u1[36]     -0.148   0.082   -0.304   -0.203   -0.150   -0.094    0.019 1.001
u1[37]     -0.120   0.079   -0.271   -0.172   -0.120   -0.067    0.040 1.001
u1[38]     -0.145   0.082   -0.301   -0.200   -0.146   -0.091    0.017 1.001
u1[39]      0.105   0.079   -0.053    0.052    0.106    0.158    0.257 1.001
u1[40]     -0.142   0.080   -0.297   -0.197   -0.143   -0.089    0.018 1.001
u1[41]     -0.043   0.078   -0.195   -0.095   -0.044    0.009    0.109 1.001
u1[42]     -0.058   0.078   -0.209   -0.109   -0.058   -0.006    0.097 1.001
u1[43]     -0.227   0.083   -0.385   -0.283   -0.229   -0.173   -0.060 1.001
u1[44]     -0.078   0.082   -0.235   -0.134   -0.080   -0.023    0.090 1.001
u1[45]     -0.054   0.078   -0.207   -0.105   -0.054   -0.002    0.100 1.001
u1[46]      0.222   0.088    0.041    0.163    0.226    0.284    0.387 1.001
u1[47]      0.043   0.079   -0.108   -0.011    0.040    0.095    0.205 1.001
u1[48]     -0.105   0.080   -0.259   -0.159   -0.106   -0.053    0.058 1.001
u1[49]     -0.030   0.078   -0.184   -0.080   -0.030    0.022    0.125 1.001
u1[50]      0.057   0.078   -0.097    0.005    0.057    0.109    0.210 1.001
u1[51]     -0.095   0.079   -0.248   -0.148   -0.096   -0.042    0.064 1.001
u1[52]      0.023   0.085   -0.152   -0.033    0.026    0.082    0.182 1.001
u1[53]     -0.019   0.078   -0.178   -0.071   -0.018    0.034    0.132 1.001
u1[54]      0.188   0.081    0.027    0.135    0.188    0.241    0.344 1.001
u1[55]      0.180   0.079    0.022    0.128    0.180    0.232    0.336 1.001
u1[56]      0.000   0.079   -0.152   -0.054   -0.001    0.052    0.157 1.001
u1[57]     -0.135   0.084   -0.293   -0.192   -0.137   -0.080    0.038 1.001
u1[58]     -0.013   0.078   -0.169   -0.065   -0.013    0.039    0.138 1.001
u1[59]      0.188   0.083    0.020    0.133    0.189    0.244    0.346 1.001
u1[60]     -0.044   0.077   -0.194   -0.096   -0.045    0.007    0.108 1.001
vv          2.075   0.195    1.731    1.939    2.063    2.198    2.490 1.001
vvint       1.503   0.540    0.633    1.125    1.438    1.808    2.738 1.001
vvslp       0.024   0.007    0.011    0.019    0.024    0.029    0.041 1.002
deviance 1281.125  19.963 1243.238 1267.563 1280.745 1294.475 1321.159 1.001


For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
```

```
 DIC info (using the rule, pD = var(deviance)/2)
 pD = 199.2 and DIC = 1480.3
 DIC is an estimate of expected predictive error (lower deviance is better).
```

```
sims <- as.mcmc(fit10)
chains <- as.matrix(sims)
sims <- as.mcmc(chains)
raftery.diag(sims)
effectiveSize(sims)
```

22. Using DIC, which of the models that allow inference over all pigs would you choose?

Using DIC I would choose the model from question 20. This model just has random coefficients for the intercept and the linear day term. Adding a random effect for pig on the quadratic day term does not improve the model.

Problems 23 and 24 are not required. If completed they will be worth bonus points. 10 points for problem 23, and 5 points for problem 24.

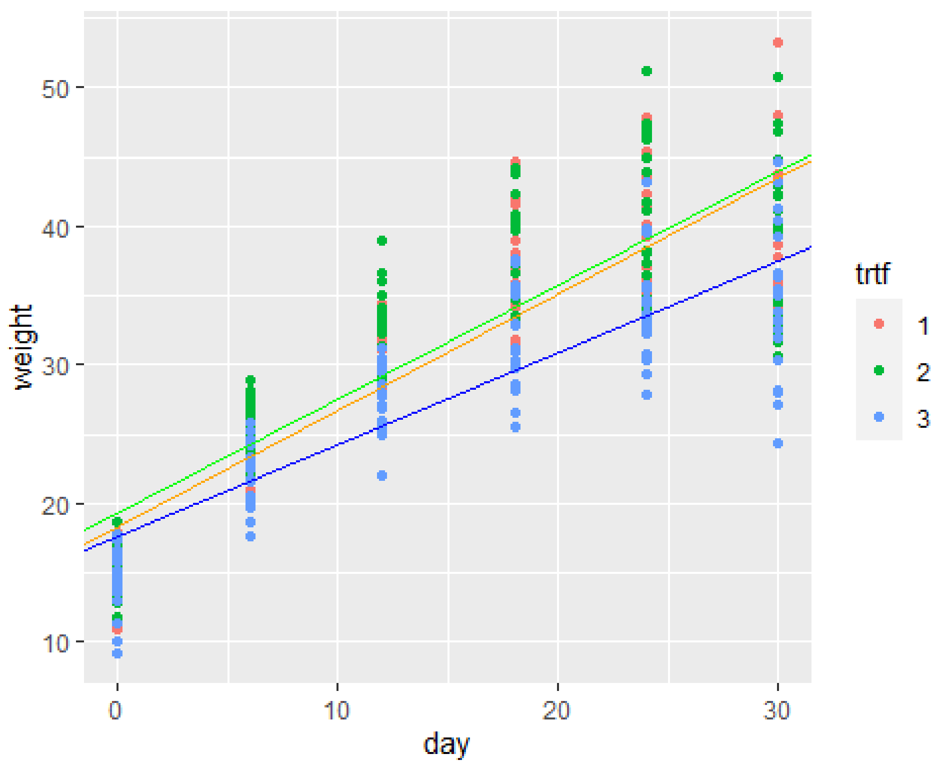
23. Since brm does not do well with these types of models, now you will do the work in Stan. Create a hierarchical model with hierarchical terms for both the intercepts and linear terms (slopes), but not for the quadratic term. This is a hierarchical model that is much like the random coefficients model in number 20. Use normal priors with means of 0 and standard deviations of 100 where appropriate, and gamma(1.1,.5) priors for the variance terms. What are means of the chains for the overall intercept in treatment 1, the overall linear term coefficient (slope) in treatment 1, and the overall quadratic term coefficient in treatment 1?

Intercept for treatment 1 is 18.5, Linear term for treatment 1 is 0.85

```
model <- "

data {
  int <lower = 1> N;
  int q;
  real weight[N];
  int day[N];
  int trt[N];
}

parameters {
  real alpha[q];
  real beta[q];
  real mu_alpha;
  real mu_beta;
  real <lower = 0> serr;
  real <lower = 0> salpha;
  real <lower = 0> sbeta;
}
```

```
model {
  mu_alpha ~ normal(0, 100);
  mu_beta ~ normal(0, 100);
  alpha ~ normal(mu_alpha, salpha);
  beta ~ normal(mu_beta, sbeta);
  serr ~ gamma(1.1, 0.5);
  salpha ~ gamma(1.1, 0.5);
  sbeta ~ gamma(1.1, .5);
  for(i in 1:N){
    weight[i] ~ normal(alpha[trt[i]] + beta[trt[i]] * day[i], serr);
  }
}


generated quantities {
  vector[N] log_lik;
  real s2error;
  real s2int;
  real s2slope;
  s2error = serr*serr;
  s2int = salpha*salpha;
  s2slope = sbeta*sbeta;
  for (i in 1:N) log_lik[i] = normal_lpdf(weight[i] | alpha[trt[i]] +
beta[trt[i]]*day[i], serr);
}

"


writeLines(model, 'hier.stan')
trt = pig$trt
day = pig$day
weight = pig$weight
N <- 360
q <- 3
hier_dat <- list(N=N, weight = weight, trt = trt, day = day, q=q)
fit12 <- stan(file = "hier.stan", data = hier_dat, iter = 11000,
            warmup = 1000, chains = 4, thin = 2)




chains <- as.matrix(fit12)
dim(chains)

 [1] 20000   375

sims <- as.mcmc(chains)
```

```r
ggplot(data = pig, mapping  = aes(x = day, y = weight, col = trtf)) +
  geom_point() +
  geom_abline(intercept = 18.4, slope = 0.84, col = "orange") +
  geom_abline(intercept = 19.38, slope = 0.82, col = "green") +
  geom_abline(intercept = 17.664, slope = 0.663, col = "blue")
```



24. Compute the waic for the model in problem 23.

waic is 2145.3

```r
llfit <- extract_log_lik(fit12, parameter_name = "log_lik",
                         merge_chains = TRUE)
waic(llfit)


 Computed from 20000 by 360 log-likelihood matrix

           Estimate   SE
 elpd_waic  -1072.7 14.0
 p_waic         7.7  0.8
 waic        2145.4 28.0
```