Week8 IC1 example

```
load("EPA_mileage.Rdata")
str(epa)

## 'data.frame': 2884 obs. of  50 variables:
##  $ yr         : int  2009 2009 2009 2009 2009 2009 2009 2009 2009 2009 ...
##  $ mfr        : int  20 20 20 20 20 20 20 20 20 20 ...
##  $ mfr.name   : Factor w/ 32 levels "ASTON MARTIN",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ bidx       : int  1 1 3 3 4 4 201 201 202 202 ...
##  $ vid        : Factor w/ 686 levels "04-NHW2","05-GRN1",..: 529 529 529 529 524 524 510
##  $ cfg        : int  0 0 1 1 2 2 0 0 0 0 ...
##  $ carline    : Factor w/ 443 levels "128I","128I CONVERTIBLE",..: 234 234 235 235 234 23
##  $ car.truck  : Factor w/ 2 levels "C","T": 2 2 2 2 2 2 2 2 2 2 ...
##  $ cid        : int  215 215 215 215 144 144 148 148 148 148 ...
##  $ police     : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ rhp        : int  235 235 235 235 173 173 220 220 220 220 ...
##  $ ec1        : logi  NA NA NA NA NA NA ...
##  $ ec2        : logi  NA NA NA NA NA NA ...
##  $ ec3        : logi  NA NA NA NA NA NA ...
##  $ ec4        : logi  NA NA NA NA NA NA ...
##  $ ec5        : logi  NA NA NA NA NA NA ...
##  $ evc        : int  102 102 102 102 102 102 102 102 102 102 ...
##  $ trns       : Factor w/ 17 levels "A4","A6","AU",..: 7 7 7 7 5 5 5 5 5 5 ...
##  $ drv        : Factor w/ 3 levels "4","F","R": 2 2 1 1 2 2 2 2 2 2 ...
##  $ od         : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ etw        : int  4500 4500 4500 4500 4000 4000 3625 3625 3625 3625 ...
##  $ cmp        : num  10 10 10 10 10.5 10.5 9.5 9.5 9.5 9.5 ...
##  $ axle       : num  2.24 2.24 2.24 2.24 2.95 2.95 2.69 2.69 2.69 2.69 ...
##  $ n.v        : num  28.7 28.7 28.7 28.7 36 36 37.3 37.3 37.7 37.7 ...
##  $ a.c        : Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...
##  $ dhp        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ sil        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ prc        : int  3 21 3 21 3 21 3 21 3 21 ...
##  $ prp        : int  31 31 32 32 32 32 31 31 31 31 ...
##  $ tnum       : int  1083480 1083479 1086540 1086539 1083587 1083586 1051401 1051400 1051
##  $ fuel       : int  61 61 61 61 61 61 61 61 61 61 ...
##  $ C.H        : Factor w/ 2 levels "C","H": 2 1 2 1 2 1 2 1 2 1 ...
##  $ avcd       : Factor w/ 3 levels "","1","A": 1 1 1 1 1 1 1 1 1 1 ...
##  $ wt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ hc         : num  0.023 0.064 NA NA NA NA 0.002 0.049 0.001 0.037 ...
##  $ co         : num  0.4 1.07 NA NA NA NA 0.03 0.5 0.07 0.27 ...
##  $ co2        : int  275 459 NA NA NA NA 260 384 260 374 ...
##  $ nox        : num  NA 0 NA NA NA NA NA 0.03 NA 0.02 ...
##  $ pm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ mpg        : num  32.2 19.3 29.9 18.4 35 23.8 34.1 23 34.1 23.7 ...
```

```
##  $ target.a   : num  37.7 37.7 37.7 37.7 28 ...
##  $ target.b   : num  0.634 0.634 0.634 0.634 0.558 ...
##  $ target.c   : num  0.024 0.024 0.024 0.024 0.021 ...
##  $ set.a      : num  13.5 13.5 13.5 13.5 10.8 ...
##  $ set.b      : num  0.104 0.104 0.104 0.104 0.129 ...
##  $ set.c      : num  0.0259 0.0259 0.0259 0.0259 0.0181 ...
##  $ engine.code: Factor w/ 441 levels "07 L537","1",..: 352 352 354 354 236 236 357 357 35
##  $ eng.family : Factor w/ 305 levels "9ADXT04.23UD",..: 33 33 33 33 48 48 45 45 47 47 ...
##  $ vpc        : int  6 6 6 6 4 4 4 4 4 4 ...
##  $ cstdwn     : num  16.1 16.1 16.1 16.1 17.4 ...
```

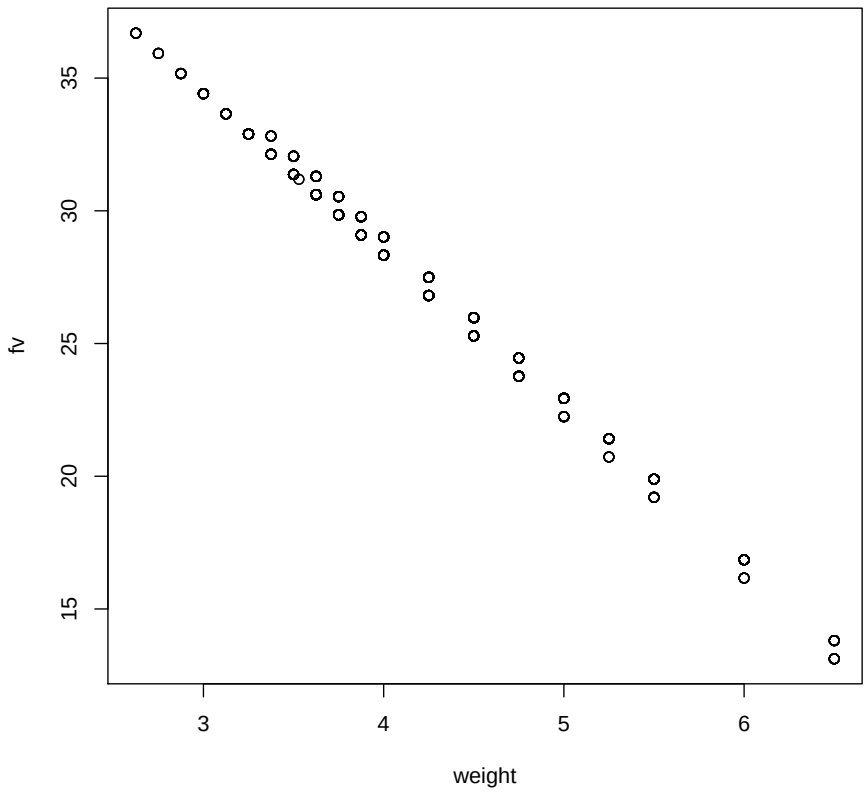
```
cartruck<-as.numeric(epa$car.truck)  #car or truck
weight<-epa$etw/1000                 #etw has vehicle weight
N<-length(weight)                    #number of observations
mpg<-epa$mpg                         #mpg
```

Ordinary least squares model

```
lm2<-lm(mpg~epa$car.truck+weight)
summary(lm2)

##
## Call:
## lm(formula = mpg ~ epa$car.truck + weight)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.749  -5.706  -1.219   5.815  33.710
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     52.6569     0.7566  69.599   <2e-16 ***
## epa$car.truckT   0.6867     0.3281   2.093   0.0365 *
## weight          -6.0821     0.1903 -31.959   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.994 on 2881 degrees of freedom
## Multiple R-squared:  0.3405,Adjusted R-squared:   0.34
## F-statistic: 743.7 on 2 and 2881 DF,  p-value: < 2.2e-16

fv<-lm2$fitted.values
plot(fv~weight)
```




Call STAN for Bayesian model

```
library(rstan)                                  #make sure rstan is available

## Loading required package:  ggplot2
##
## Attaching package:  'ggplot2'
## The following object is masked _by_ '.GlobalEnv':
##
##    mpg
## Loading required package:  StanHeaders
## rstan (Version 2.14.1, packaged:  2016-12-28 14:55:41 UTC, GitRev:
5fa1e80eb817)
## For execution on a local, multicore CPU with excess RAM we recommend
calling
## rstan_options(auto_write = TRUE)
## options(mc.cores = parallel::detectCores())
```

```
rstan_options(auto_write = TRUE)              #use multiple cores
options(mc.cores = parallel::detectCores())   #if we have them
stanfit<-stan("week8_IC1_covariance_example.stan")     #call STAN using defaults
print(stanfit)
```

```
## Inference for Stan model: week8_IC1_covariance_example.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##           mean se_mean   sd     2.5%      25%      50%      75%    97.5%
## car       2.64    0.02 0.78     1.08     2.14     2.65     3.15     4.12
## truck     3.32    0.03 0.99     1.30     2.66     3.32     3.98     5.24
## beta     -6.08    0.01 0.20    -6.46    -6.21    -6.08    -5.95    -5.69
## sigma     7.00    0.00 0.09     6.82     6.93     6.99     7.06     7.17
## lp__  -7050.29    0.04 1.42 -7053.78 -7050.99 -7049.96 -7049.25 -7048.54
##       n_eff Rhat
## car    1152    1
## truck  1130    1
## beta   1133    1
## sigma  1611    1
## lp__   1142    1
##
## Samples were drawn using NUTS(diag_e) at Wed Mar 15 08:48:19 2017.
## 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).
```

```
print(get_stanmodel(stanfit))
```

```
## S4 class stanmodel 'week8_IC1_covariance_example' coded as follows:
## //model file for week8: Single factor ANOVA with a covariate
## data {
##   int N;                  //sample1 size
##   int cartruck[N];        //car or truck
##   real weight[N];         //number of levels
##   real mpg[N];            //y values - nanoseconds
## }
## parameters {
##   real car;               //car
##   real truck;             //truck
##   real beta;              //beta is the slope of the parallel regression lines
##   real<lower=0> sigma;             //residual standard error
## }
## model {
##   car   ~ normal(0,100);        //normal priors for slope includes all reasonable mileage
##   truck ~ normal(0,100);      //slope will be very small with small standard deviation if
```

```
##   beta  ~ normal(0,100);      //slope will be very small with small standard deviation if
##   sigma ~ cauchy(0,10);       //half-Cauchy prior for residual standard error
##
##   for (i in 1:N)                   //loop through y values
##     if (cartruck[i]==1)
##       mpg[i] ~ normal(50.0+car+beta*weight[i],sigma);
##     else
##       mpg[i] ~ normal(50.0+truck+beta*weight[i],sigma);
## }
```

Launch shinystan

```
library(shinystan)                                    #launch shinystan

## Loading required package:  shiny
##
## This is shinystan version 2.3.0

launch_shinystan(stanfit)

##
## Creating shinystan object...
##
## Launching ShinyStan interface...  for large models this  may take
some time.
##
## Listening on http://127.0.0.1:7014
```