Test 2

Jeff Rouder

Winter 2020

My colleagues at Zurich, led by then post-doctoral researcher Alodie Rey-Mermet, are very much concerned with the mental processes underlying attention and inhibition. They collect a lot of data from a lot of people on a lot of tasks. Here, I have provided one of their sets, a flanker tasks. Participants must decide if a center letter is an S or C. Sometimes the surrounding letters are S, sometimes they are C. Half the time, the surround and center match (congruent trials); half the time, they mismatch (incongruent trials). Attention is needed more on incongruent than congruent trials. The difference in RT between incongruent and congruent trials forms the *flanker effect.*

The main questions are whether there is a flanker effect, and if so, how is it distributed among people? Does everyone show the same sized flanker effect? Is there a lot of variability? Do some people have no flanker effects? Do some people have opposite flanker effects? What is going on.

1. Load up the data.

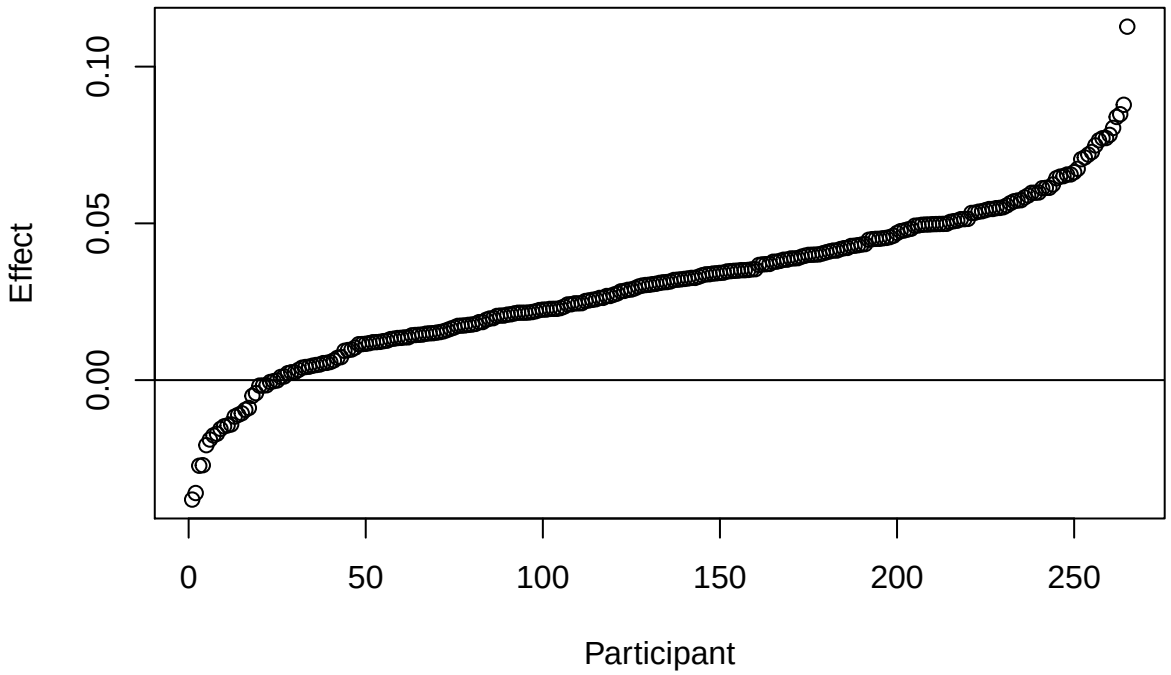
```
indat=read.table('test2.dat',head=T)
rt=indat$rt
sub=indat$sub
cond=indat$cond
I=max(sub)
```

2. Most psychologists would study the *observed individual flanker effect.* Let $\bar{y}_{ij}$ be the sample mean for the ith person in the jth condition. The observed flanker effect, d_i is $d_i = \bar{y}_{i2} - \bar{y}_{i1}$.

- Tabulate these observed flanker effects and plot them.
- How big is the average effect? How variable is it across people?
- Interpret the effects in light of the main questions above.

```
m=tapply(rt,list(sub,cond),mean)
d=m[,2]-m[,1]
```

```
plot(1:I,sort(d),ylab="Effect",xlab="Participant")
abline(h=0)
```




```
mean(d)
```

```
## [1] 0.03037683
```

```
sd(d)
```

```
## [1] 0.02385298
```

```
mean(d<0)
```

```
## [1] 0.09433962
```

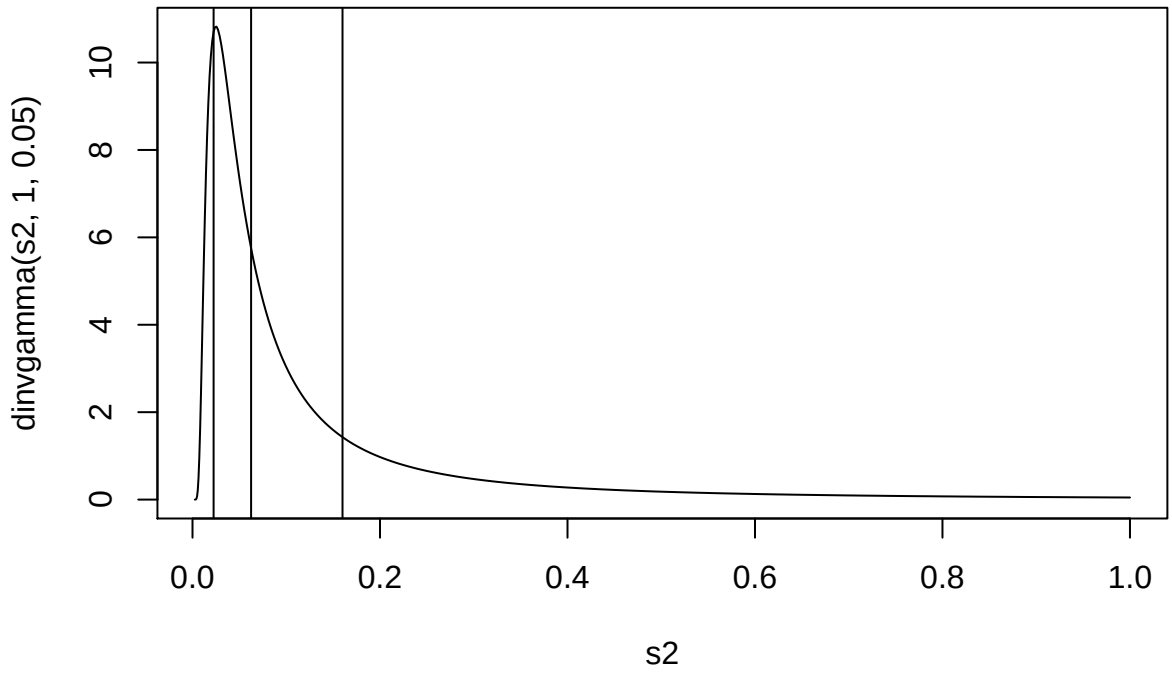
3. The first model is a separate normal model for each individual. For each person:

$$Y_{ijk} \sim \mathrm{N}(\mu_{ij}, \sigma_i^2)$$

Priors: We can assume that any μ parameter is on the subsecond scale ($\mu \sim \mathrm{N}(.5, .5^2)$). Also, standard deviations of replicate RTs are on subsecond scale too, say between 2s and .3s. This means that variances are about hundreths to tenths of squared seconds. A broad prior is $\sigma^2 \sim \mathrm{IG}(1, .05)$.

- Plot this prior. Does it have good mass for standard deviations as small as .15s and and as large as .4s?

```
s=seq(.05,1,.005)
s2=s^2
plot(s2,dinvgamma(s2,1,.05),type='l')
abline(v=c(.15^2,.25^2,.4^2))
```



- Analyze this model. You can compute a posterior density for d_i by making it a transformed parameter in your chains.

Here is some free-bee stan code:

```
model1Code <- "

data {
        int<lower=1> n;
        int<lower=1> I;
    int<lower=0,upper=I> sub[n];
    int<lower=0,upper=2> cond[n];
    vector[n] y;
    }

parameters {
        matrix[I,2] mu;
    vector<lower=0>[I] sigma2;
    }

transformed parameters {
    vector<lower=0>[I] sigma;
    vector[I] theta;
    vector[n] center;
    vector[n] scale;
    sigma=sqrt(sigma2);
    for (i in 1:I){
    theta[i]=mu[i,2]-mu[i,1];}
    for (k in 1:n){
    center[k]=mu[sub[k],cond[k]];
    scale[k]=sigma[sub[k]];}
    }
```

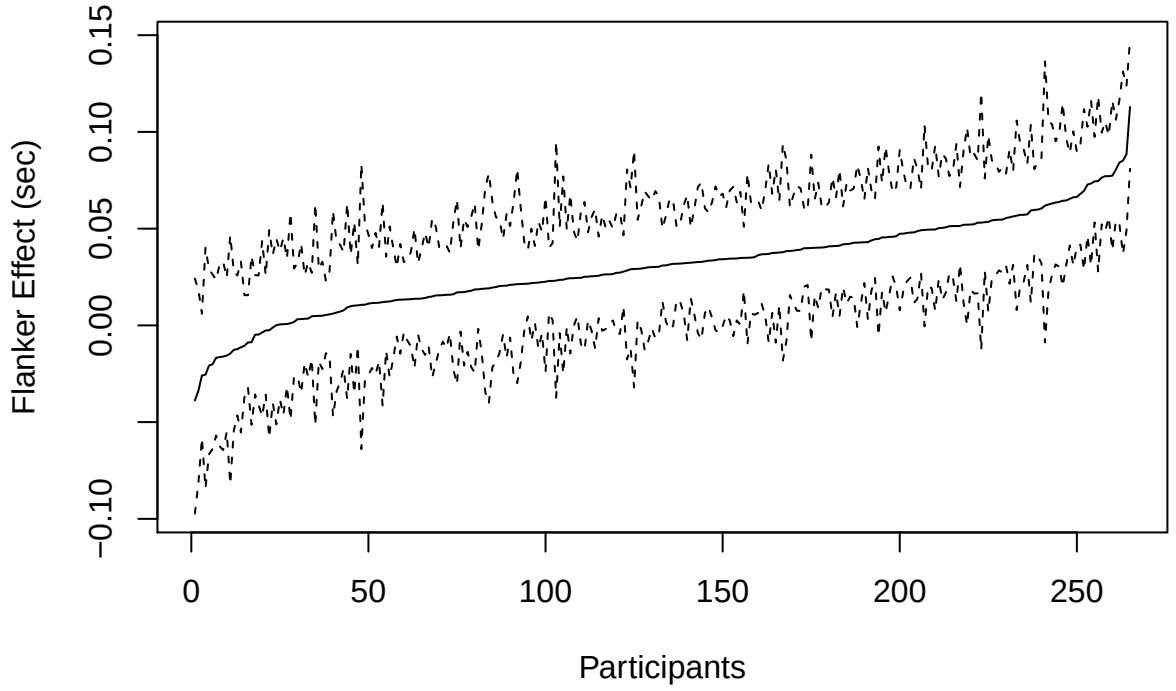
```
model {
    to_vector(mu) ~ normal(.5,.5);
        for (i in 1:I){
          sigma2[i] ~ inv_gamma(1,.05);}
        y ~ normal(center,scale);
    }"

model1 <- stan_model(model_code = model1Code)
dat <- list(
  n=length(rt),
  I=max(sub),
  sub=sub,
  cond=cond,
  y=rt)
samples <- sampling(model1,
                    data=dat,
                    iter=600,
                    chains=1,
                    warmup=200)
```

```
##
## SAMPLING FOR MODEL 'bc84ef34e4e1f6ea1f9406e0d66e3491' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.005032 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 50.32 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 600 [  0%]  (Warmup)
## Chain 1: Iteration:  60 / 600 [ 10%]  (Warmup)
## Chain 1: Iteration: 120 / 600 [ 20%]  (Warmup)
## Chain 1: Iteration: 180 / 600 [ 30%]  (Warmup)
## Chain 1: Iteration: 201 / 600 [ 33%]  (Sampling)
## Chain 1: Iteration: 260 / 600 [ 43%]  (Sampling)
## Chain 1: Iteration: 320 / 600 [ 53%]  (Sampling)
## Chain 1: Iteration: 380 / 600 [ 63%]  (Sampling)
## Chain 1: Iteration: 440 / 600 [ 73%]  (Sampling)
## Chain 1: Iteration: 500 / 600 [ 83%]  (Sampling)
## Chain 1: Iteration: 560 / 600 [ 93%]  (Sampling)
## Chain 1: Iteration: 600 / 600 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 55.1426 seconds (Warm-up)
## Chain 1:                19.9494 seconds (Sampling)
## Chain 1:                75.092 seconds (Total)
## Chain 1:
```

- Plot the estimated flanker effect per person. Add in credible intervals. Here is some free-bee plotting code (note the 90% CIs):

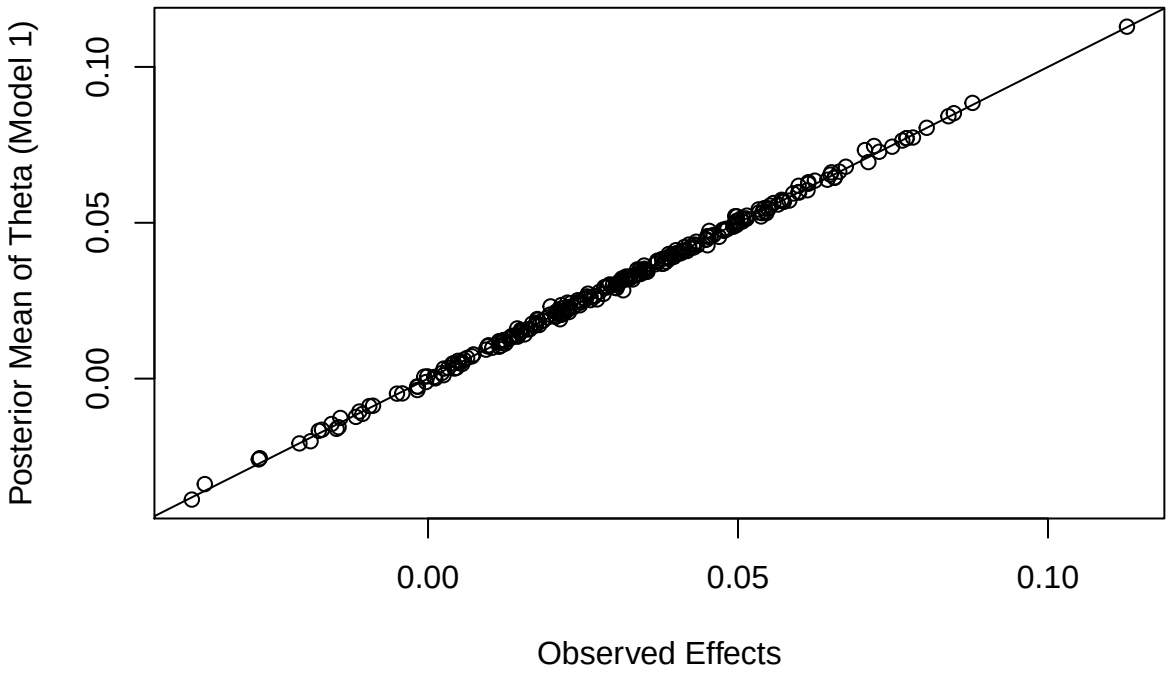
```
I=max(sub)
theta=extract(samples)$theta
thetaMean=apply(theta,2,mean)
thetaLo=apply(theta,2,quantile,p=.05)
thetaHi=apply(theta,2,quantile,p=.95)
o=order(thetaMean)
```

```
range=c(min(thetaLo),max(thetaHi))
plot(1:I,thetaMean[o],typ='l',ylim=range,
     ylab="Flanker Effect (sec)",
     xlab="Participants")
lines(1:I,thetaHi[o],lty=2)
lines(1:I,thetaLo[o],lty=2)
```



- Interpret the graph.
- Compare estimates here to the observed flanker effects in Q2. Use a graph please for this comparison.

```
plot(d,thetaMean,ylab="Posterior Mean of Theta (Model 1)",xlab="Observed Effects")
abline(0,1)
```



4. Notice how sloppy the individual CIs are. That is because we are using a separate variance parameter per person. What a mistake, we don't have the resolution for such fine grain modeling, plus it makes the interpretation of delta difficult.

Fit the model

$$Y_{ijk} \sim \mathrm{N}(\mu_{ij}, \sigma^2)$$

We can call this the *cell-means* model. We have a seperate, independent parameter for each cell.

- How does the posterior of the flanker effect change? You need to make appropriate graphs with (hopefully now smoother) credible intervals.

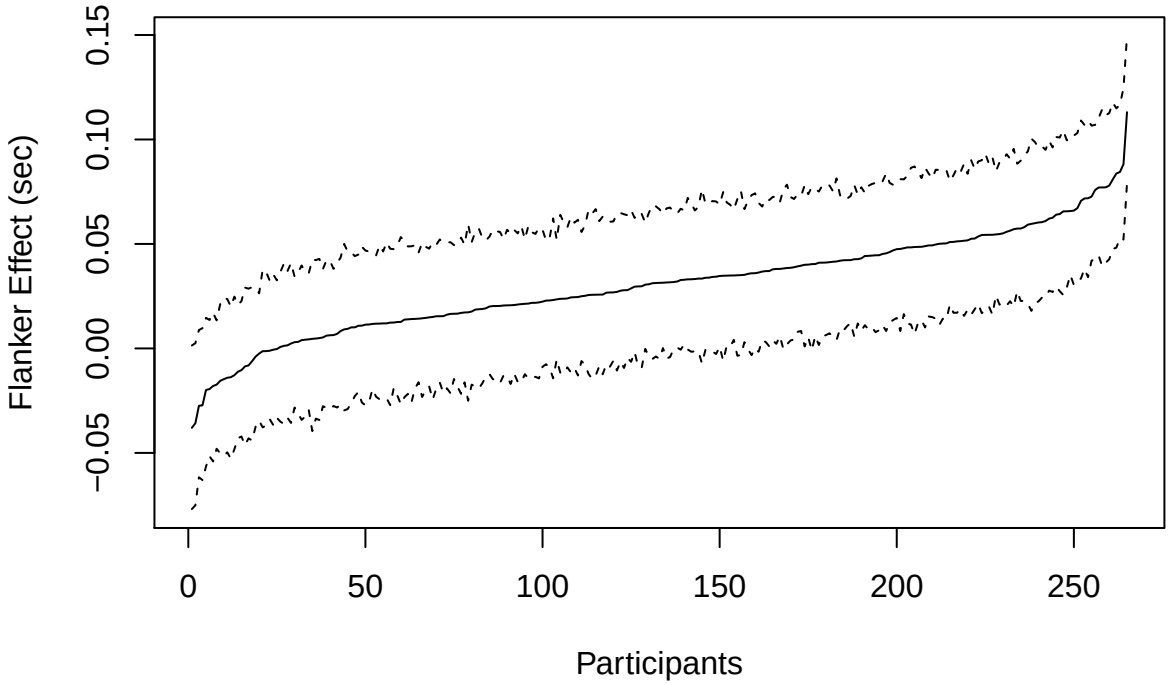
```
model2Code <- "

data {
        int<lower=1> n;
        int<lower=1> I;
    int<lower=0,upper=I> sub[n];
    int<lower=0,upper=2> cond[n];
    vector[n] y;
    }

parameters {
        matrix[I,2] mu;
    real<lower=0> sigma2;
    }

transformed parameters {
    real<lower=0> sigma;
    vector[I] theta;
    vector[n] center;
    sigma=sqrt(sigma2);
    for (i in 1:I){
    theta[i]=mu[i,2]-mu[i,1];}
```

```
    for (k in 1:n){
    center[k]=mu[sub[k],cond[k]];}
    }

model {
    to_vector(mu) ~ normal(.5,.5);
        sigma2 ~ inv_gamma(1,.05);
        y ~ normal(center,sigma);
    }"

model2 <- stan_model(model_code = model2Code)
dat <- list(
  n=length(rt),
  I=max(sub),
  sub=sub,
  cond=cond,
  y=rt)
samples <- sampling(model2,
                    data=dat,
                    iter=600,
                    chains=1,
                    warmup=200)
```

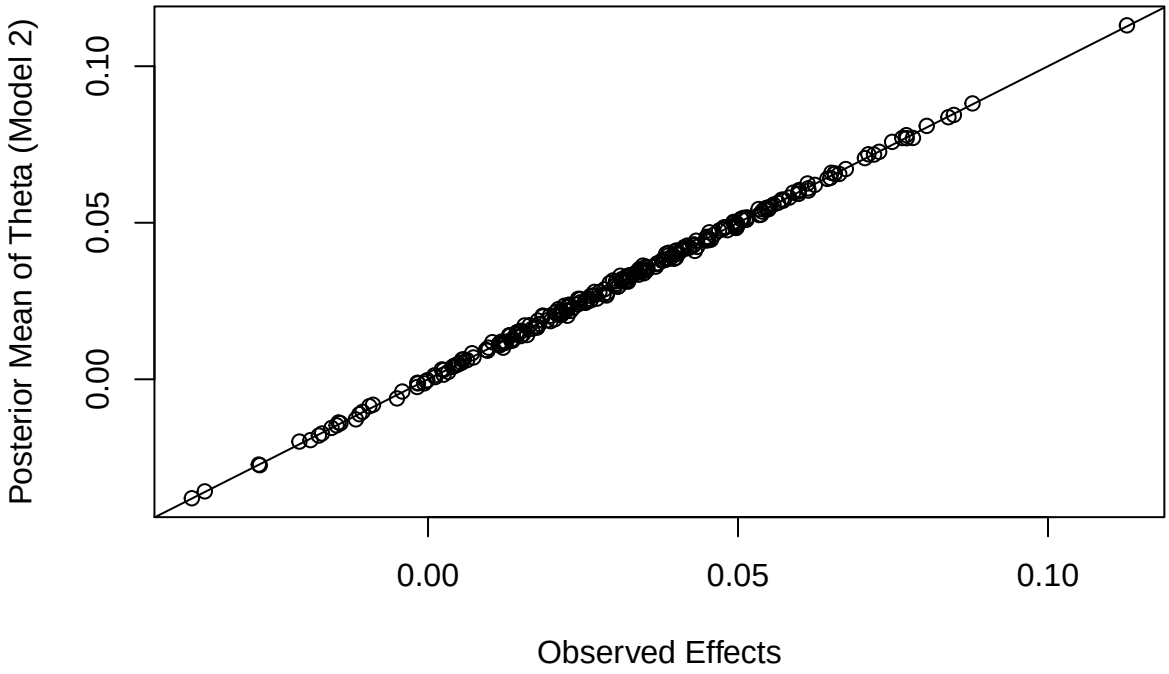
```
##
## SAMPLING FOR MODEL '75698015c3736181a4d9fb3d0599f68b' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.002145 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 21.45 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 600 [  0%]  (Warmup)
## Chain 1: Iteration:  60 / 600 [ 10%]  (Warmup)
## Chain 1: Iteration: 120 / 600 [ 20%]  (Warmup)
## Chain 1: Iteration: 180 / 600 [ 30%]  (Warmup)
## Chain 1: Iteration: 201 / 600 [ 33%]  (Sampling)
## Chain 1: Iteration: 260 / 600 [ 43%]  (Sampling)
## Chain 1: Iteration: 320 / 600 [ 53%]  (Sampling)
## Chain 1: Iteration: 380 / 600 [ 63%]  (Sampling)
## Chain 1: Iteration: 440 / 600 [ 73%]  (Sampling)
## Chain 1: Iteration: 500 / 600 [ 83%]  (Sampling)
## Chain 1: Iteration: 560 / 600 [ 93%]  (Sampling)
## Chain 1: Iteration: 600 / 600 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 15.6305 seconds (Warm-up)
## Chain 1:                9.36641 seconds (Sampling)
## Chain 1:                24.9969 seconds (Total)
## Chain 1:
```

```
I=max(sub)
theta=extract(samples)$theta
thetaMean=apply(theta,2,mean)
thetaLo=apply(theta,2,quantile,p=.05)
thetaHi=apply(theta,2,quantile,p=.95)
```

```
o=order(thetaMean)
range=c(min(thetaLo),max(thetaHi))
plot(1:I,thetaMean[o],typ='l',ylim=range,
     ylab="Flanker Effect (sec)",
     xlab="Participants")
lines(1:I,thetaHi[o],lty=2)
lines(1:I,thetaLo[o],lty=2)
```




```
plot(d,thetaMean,ylab="Posterior Mean of Theta (Model 2)",xlab="Observed Effects")
abline(0,1)
```



5. The above two models are pretty much equivalent to inspecting the observed flanker effects. Let's go hierarchical. Here is the first hierarchical model:

$$Y_{ijk} \sim \mathrm{N}(\mu_{ij}, \sigma^2)$$
$$\mu_{ij} \sim \mathrm{N}(\eta, \delta^2)$$

The priors on η and δ^2 may be the aforementioned $\mathrm{N}(.5,.5^2)$ and $\mathrm{IG}(1,.05)$.

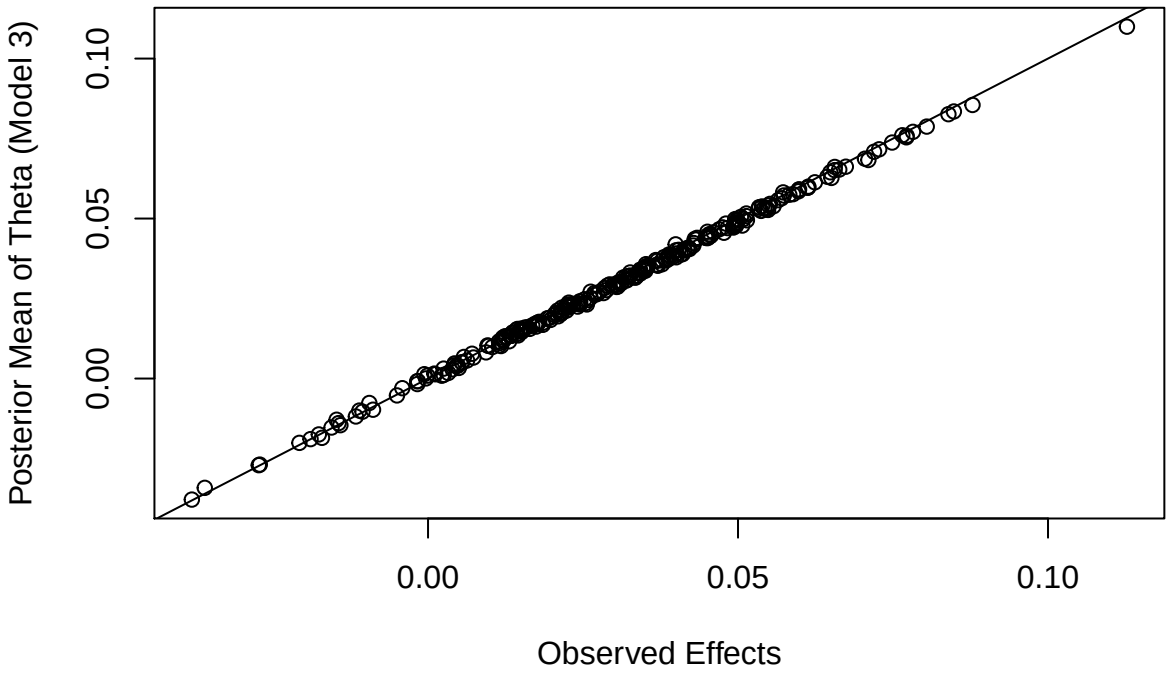
- We may expect some regularization of the μ parameters. Was there any? Plot the posterior means of μ as a function of sample means to find out. Perhaps surprisingly, there is very little regularization, why not?

```
model3Code <- "

data {
        int<lower=1> n;
        int<lower=1> I;
    int<lower=0,upper=I> sub[n];
    int<lower=0,upper=2> cond[n];
    vector[n] y;
    }

parameters {
        matrix[I,2] mu;
    real<lower=0> sigma2;
    real eta;
    real <lower=0> delta2;
    }

transformed parameters {
    real<lower=0> sigma;
    real<lower=0> delta;
    vector[I] theta;
    vector[n] center;
    sigma=sqrt(sigma2);
    delta=sqrt(delta2);
    for (i in 1:I){
    theta[i]=mu[i,2]-mu[i,1];}
    for (k in 1:n){
    center[k]=mu[sub[k],cond[k]];}
    }

model {
    delta2 ~ inv_gamma(1,.05);
    eta ~ normal(.5,.5);
    to_vector(mu) ~ normal(eta,delta);
        sigma2 ~ inv_gamma(1,.05);
        y ~ normal(center,sigma);
    }"

model3 <- stan_model(model_code = model3Code)
dat <- list(
  n=length(rt),
  I=max(sub),
```

```
  sub=sub,
  cond=cond,
  y=rt)
samples <- sampling(model3,
                    data=dat,
                    iter=600,
                    chains=1,
                    warmup=200)
```

```
##
## SAMPLING FOR MODEL '34a81d51d6696532cf14782ceaa521bb' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.001218 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 12.18 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 600 [  0%]  (Warmup)
## Chain 1: Iteration:  60 / 600 [ 10%]  (Warmup)
## Chain 1: Iteration: 120 / 600 [ 20%]  (Warmup)
## Chain 1: Iteration: 180 / 600 [ 30%]  (Warmup)
## Chain 1: Iteration: 201 / 600 [ 33%]  (Sampling)
## Chain 1: Iteration: 260 / 600 [ 43%]  (Sampling)
## Chain 1: Iteration: 320 / 600 [ 53%]  (Sampling)
## Chain 1: Iteration: 380 / 600 [ 63%]  (Sampling)
## Chain 1: Iteration: 440 / 600 [ 73%]  (Sampling)
## Chain 1: Iteration: 500 / 600 [ 83%]  (Sampling)
## Chain 1: Iteration: 560 / 600 [ 93%]  (Sampling)
## Chain 1: Iteration: 600 / 600 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 9.15107 seconds (Warm-up)
## Chain 1:                9.63473 seconds (Sampling)
## Chain 1:                18.7858 seconds (Total)
## Chain 1:
```

```
theta=extract(samples)$theta
thetaMean=apply(theta,2,mean)
plot(d,thetaMean,ylab="Posterior Mean of Theta (Model 3)",xlab="Observed Effects")
abline(0,1)
```



- Perhaps there was some effect on the estimated flanker effect. The previous models yield estimates that are about the same as the sample flanker effect. How about this model? Make the appropriate graphs.

6. Perhaps hierarchical on cell means is not the most useful approach. What we are really concerned about are flanker effects, not condition means. Here is a simple model that isolates the flanker effect as a full-fledged parameter with priors and posteriors.

$$
\begin{aligned}
Y_{ijk} &\sim \mathrm{N}(\alpha_i + x_j\theta, \sigma^2) \\
\alpha_i &\sim \mathrm{N}(.5, .5^2) \\
\theta &\sim \mathrm{N}(0, .05^2)
\end{aligned}
$$

This is an easy model to analyze. There is only one effect parameter, θ. So, plot it's prior and posterior.

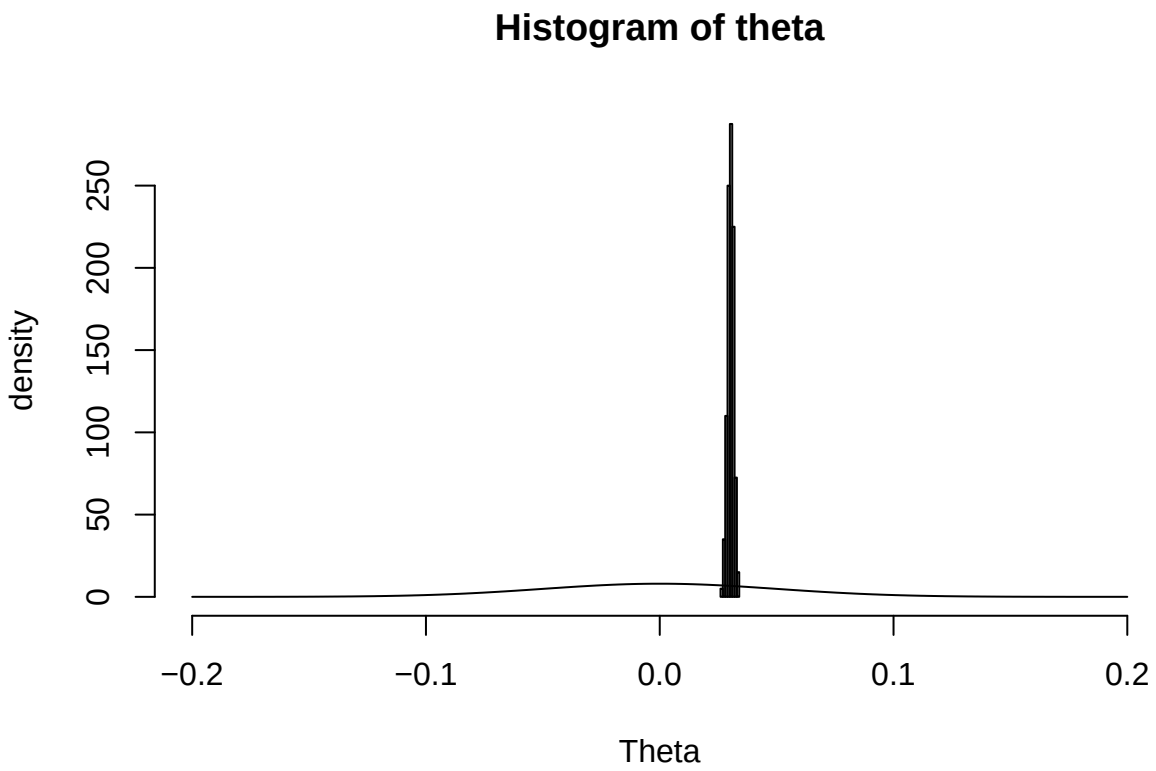
```
model4Code <- "

data {
        int<lower=1> n;
        int<lower=1> I;
    int<lower=0,upper=I> sub[n];
    int<lower=0,upper=2> cond[n];
    vector[n] y;
}

transformed data {
  vector[n] x;
  for (k in 1:n){
  x[k]=cond[k]-1.5;}
}

parameters {
        vector[I] alpha;
        real theta;
```

```
    real<lower=0> sigma2;
    }


transformed parameters {
    real<lower=0> sigma;
    vector[n] center;
    sigma=sqrt(sigma2);
    for (k in 1:n){
    center[k]=alpha[sub[k]]+x[k]*theta;}
    }

model {
    theta ~ normal(0,.05);
    to_vector(alpha) ~ normal(.5,.5);
        sigma2 ~ inv_gamma(1,.05);
        y ~ normal(center,sigma);
    }"

model4 <- stan_model(model_code = model4Code)
dat <- list(
  n=length(rt),
  I=max(sub),
  sub=sub,
  cond=cond,
  y=rt)
samples <- sampling(model4,
                    data=dat,
                    iter=600,
                    chains=1,
                    warmup=200)
```

```
##
## SAMPLING FOR MODEL 'e517c3582ee370f326abcf2634ddd42f' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.004693 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 46.93 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 600 [  0%]  (Warmup)
## Chain 1: Iteration:  60 / 600 [ 10%]  (Warmup)
## Chain 1: Iteration: 120 / 600 [ 20%]  (Warmup)
## Chain 1: Iteration: 180 / 600 [ 30%]  (Warmup)
## Chain 1: Iteration: 201 / 600 [ 33%]  (Sampling)
## Chain 1: Iteration: 260 / 600 [ 43%]  (Sampling)
## Chain 1: Iteration: 320 / 600 [ 53%]  (Sampling)
## Chain 1: Iteration: 380 / 600 [ 63%]  (Sampling)
## Chain 1: Iteration: 440 / 600 [ 73%]  (Sampling)
## Chain 1: Iteration: 500 / 600 [ 83%]  (Sampling)
## Chain 1: Iteration: 560 / 600 [ 93%]  (Sampling)
## Chain 1: Iteration: 600 / 600 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 26.4398 seconds (Warm-up)
```

```
## Chain 1:                43.5679 seconds (Sampling)
## Chain 1:                70.0077 seconds (Total)
## Chain 1:
```

```
vals=seq(-.2,.2,.001)
theta=extract(samples)$theta
hist(theta,prob=T,xlim=c(min(vals),max(vals)),xlab="Theta",ylab="density")
lines(vals,dnorm(vals,0,.05))
```



7. And now we need a model where we can measure individual flanker effects. Let's do it non-hierarchical first.

$$
\begin{aligned}
Y_{ijk} &\sim \mathrm{N}(\alpha_i + x_j\theta_i, \sigma^2) \\
\alpha_i &\sim \mathrm{N}(.5, .5^2) \\
\theta_i &\sim \mathrm{N}(0, .05^2)
\end{aligned}
$$

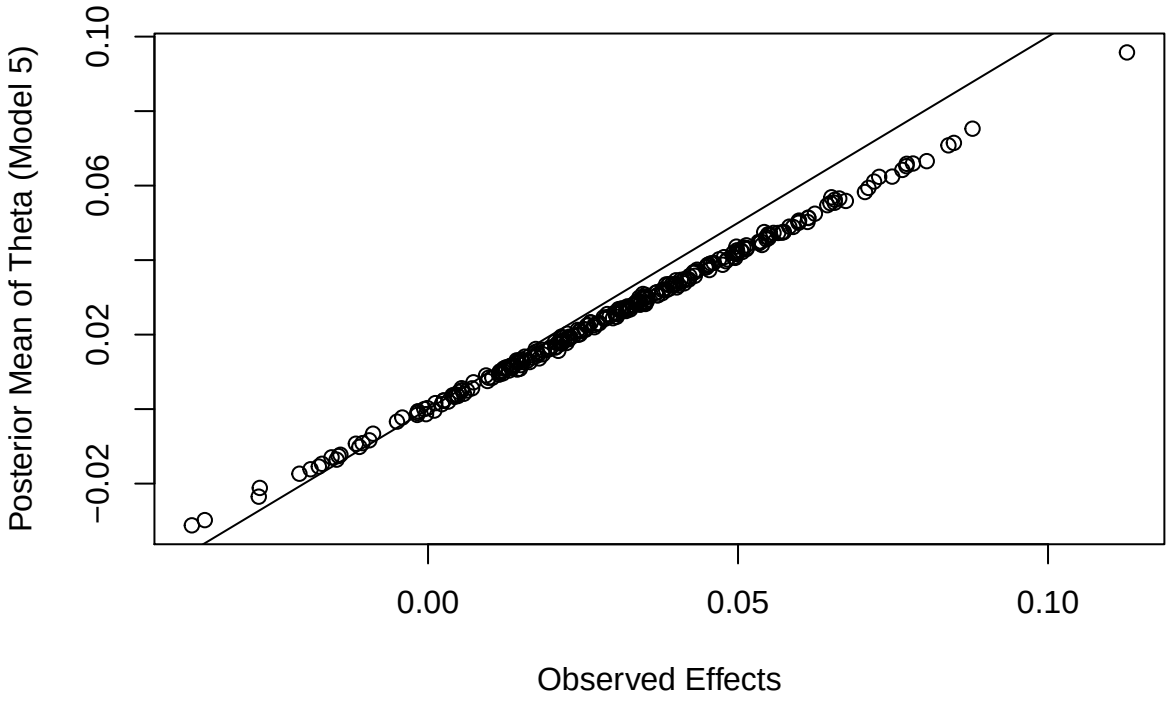
```
model5Code <- "

data {
        int<lower=1> n;
        int<lower=1> I;
    int<lower=0,upper=I> sub[n];
    int<lower=0,upper=2> cond[n];
    vector[n] y;
}

transformed data {
  vector[n] x;
  for (k in 1:n){
  x[k]=cond[k]-1.5;}
```

```
}

parameters {
        vector[I] alpha;
        vector[I] theta;
    real<lower=0> sigma2;
    }


transformed parameters {
    real<lower=0> sigma;
    vector[n] center;
    sigma=sqrt(sigma2);
    for (k in 1:n){
    center[k]=alpha[sub[k]]+x[k]*theta[sub[k]];}
    }



model {
    to_vector(theta) ~ normal(0,.05);
    to_vector(alpha) ~ normal(.5,.5);
        sigma2 ~ inv_gamma(1,.05);
        y ~ normal(center,sigma);
    }"

model5 <- stan_model(model_code = model5Code)
dat <- list(
  n=length(rt),
  I=max(sub),
  sub=sub,
  cond=cond,
  y=rt)
samples <- sampling(model5,
                    data=dat,
                    iter=600,
                    chains=1,
                    warmup=200)
```

```
##
## SAMPLING FOR MODEL '4280dc95b3b5713f90ea9161c37d2b80' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.005582 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 55.82 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 600 [  0%]  (Warmup)
## Chain 1: Iteration:  60 / 600 [ 10%]  (Warmup)
## Chain 1: Iteration: 120 / 600 [ 20%]  (Warmup)
## Chain 1: Iteration: 180 / 600 [ 30%]  (Warmup)
## Chain 1: Iteration: 201 / 600 [ 33%]  (Sampling)
## Chain 1: Iteration: 260 / 600 [ 43%]  (Sampling)
## Chain 1: Iteration: 320 / 600 [ 53%]  (Sampling)
```

```
## Chain 1: Iteration: 380 / 600 [ 63%]  (Sampling)
## Chain 1: Iteration: 440 / 600 [ 73%]  (Sampling)
## Chain 1: Iteration: 500 / 600 [ 83%]  (Sampling)
## Chain 1: Iteration: 560 / 600 [ 93%]  (Sampling)
## Chain 1: Iteration: 600 / 600 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 33.6644 seconds (Warm-up)
## Chain 1:                23.5139 seconds (Sampling)
## Chain 1:                57.1784 seconds (Total)
## Chain 1:
```

```
theta=extract(samples)$theta
thetaMean=apply(theta,2,mean)
plot(d,thetaMean,ylab="Posterior Mean of Theta (Model 5)",xlab="Observed Effects")
abline(0,1)
```



8. Let's go hierarchical on each individual's flanker effect:

$$
\begin{aligned}
Y_{ijk} &\sim \mathrm{N}(\alpha_i + x_j\theta_i, \sigma^2) \\
\alpha_i &\sim \mathrm{N}(.5, .5^2) \\
\theta_i &\sim \mathrm{N}(\eta, \delta^2)
\end{aligned}
$$

with priors $\eta \sim \mathrm{N}(0, .05^2)$ and $\delta \sim \mathrm{IG}(1, .002)$. These priors place mass at the 10 ms scale, which is appropriate. You can plot them if you wish to see.

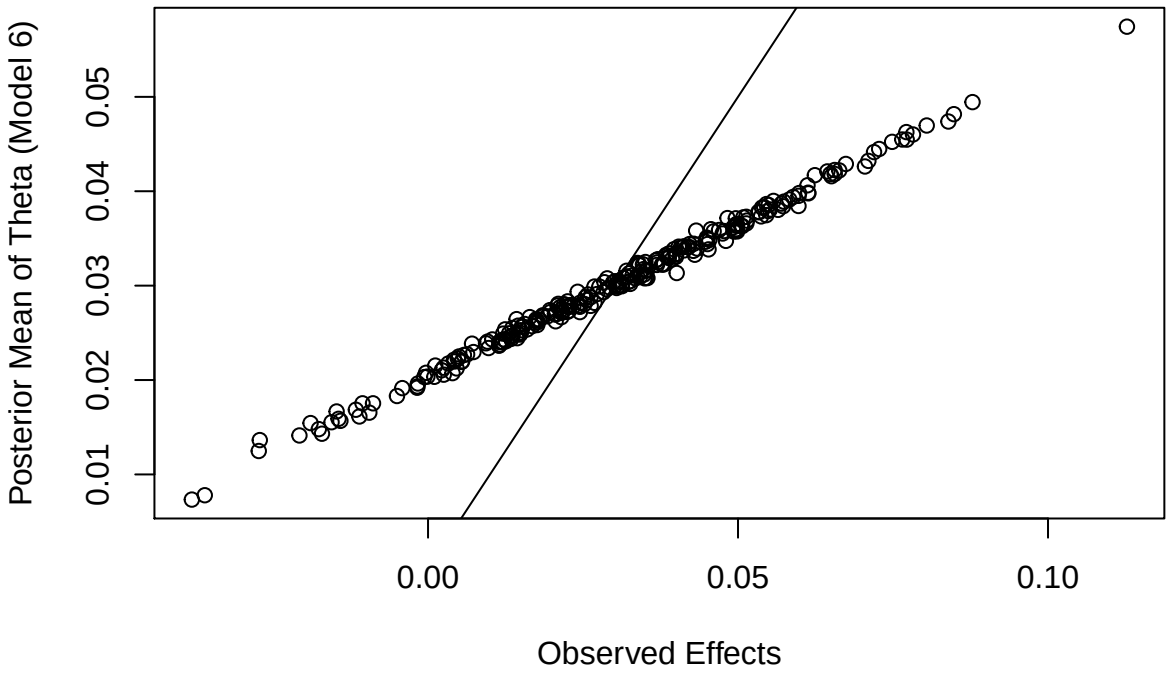
- Do the analysis. Now, plot the posterior mean of θ, perhaps with credible intervals.
- Be sure to plot the posterior mean of θ vs. the sample flanker effect.
- You should see a lot of regularization.
- What is your interpretation of these data?

```
model6Code <- "

data {
        int<lower=1> n;
        int<lower=1> I;
    int<lower=0,upper=I> sub[n];
    int<lower=0,upper=2> cond[n];
    vector[n] y;
}

transformed data {
  vector[n] x;
  for (k in 1:n){
  x[k]=cond[k]-1.5;}
}

parameters {
        vector[I] alpha;
        vector[I] theta;
    real<lower=0> sigma2;
    real eta;
    real <lower=0> delta2;
    }


transformed parameters {
    real<lower=0> sigma;
    real<lower=0> delta;
    vector[n] center;
    sigma=sqrt(sigma2);
    delta=sqrt(delta2);
    for (k in 1:n){
    center[k]=alpha[sub[k]]+x[k]*theta[sub[k]];}
    }



model {
    delta2 ~ inv_gamma(1,.002);
    eta ~ normal(0,.05);
    to_vector(theta) ~ normal(eta,delta);
    to_vector(alpha) ~ normal(.5,.5);
        sigma2 ~ inv_gamma(1,.05);
        y ~ normal(center,sigma);
    }"

model6 <- stan_model(model_code = model6Code)
dat <- list(
  n=length(rt),
  I=max(sub),
  sub=sub,
  cond=cond,
  y=rt)
```

```
samples <- sampling(model6,
                    data=dat,
                    iter=600,
                    chains=1,
                    warmup=200)
```

```
##
## SAMPLING FOR MODEL 'c900e127f86baa2f095cb472b2d7bb7b' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.004569 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 45.69 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 600 [  0%]  (Warmup)
## Chain 1: Iteration:  60 / 600 [ 10%]  (Warmup)
## Chain 1: Iteration: 120 / 600 [ 20%]  (Warmup)
## Chain 1: Iteration: 180 / 600 [ 30%]  (Warmup)
## Chain 1: Iteration: 201 / 600 [ 33%]  (Sampling)
## Chain 1: Iteration: 260 / 600 [ 43%]  (Sampling)
## Chain 1: Iteration: 320 / 600 [ 53%]  (Sampling)
## Chain 1: Iteration: 380 / 600 [ 63%]  (Sampling)
## Chain 1: Iteration: 440 / 600 [ 73%]  (Sampling)
## Chain 1: Iteration: 500 / 600 [ 83%]  (Sampling)
## Chain 1: Iteration: 560 / 600 [ 93%]  (Sampling)
## Chain 1: Iteration: 600 / 600 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 33.1355 seconds (Warm-up)
## Chain 1:                46.2873 seconds (Sampling)
## Chain 1:                79.4228 seconds (Total)
## Chain 1:
```

```
theta=extract(samples)$theta
thetaMean=apply(theta,2,mean)
plot(d,thetaMean,ylab="Posterior Mean of Theta (Model 6)",xlab="Observed Effects")
abline(0,1)
```



9. Extra credit. How does the regularization depend on the the scale of the prior on δ^2. Draw a few priors, and see how the regularizagtion changes.