Beta_reg_stan

First install all the library packages.

```
library(MCMCpack)
library(rstan)
library(reshape2)
library(ggplot2)
library(tidyr)
```

Code to generate a simulated dataset

```
source("Data_gen.R")
```

Number of iterations

```
iter = 10000
```

Stan implementation

```
fit <- stan(file='fels_beta_reg.stan',data=fels_Data,iter=iter,chains=1,seed=6,
            control=list(adapt_delta=0.9))
```

```
##
## SAMPLING FOR MODEL 'fels_beta_reg' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.015373 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 153.73 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 1: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 1: Iteration: 3000 / 10000 [ 30%]  (Warmup)
## Chain 1: Iteration: 4000 / 10000 [ 40%]  (Warmup)
## Chain 1: Iteration: 5000 / 10000 [ 50%]  (Warmup)
## Chain 1: Iteration: 5001 / 10000 [ 50%]  (Sampling)
## Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 2153.12 seconds (Warm-up)
## Chain 1:                2346.58 seconds (Sampling)
## Chain 1:                4499.69 seconds (Total)
## Chain 1:
```

```
fit_ss <- rstan::extract(fit)
```

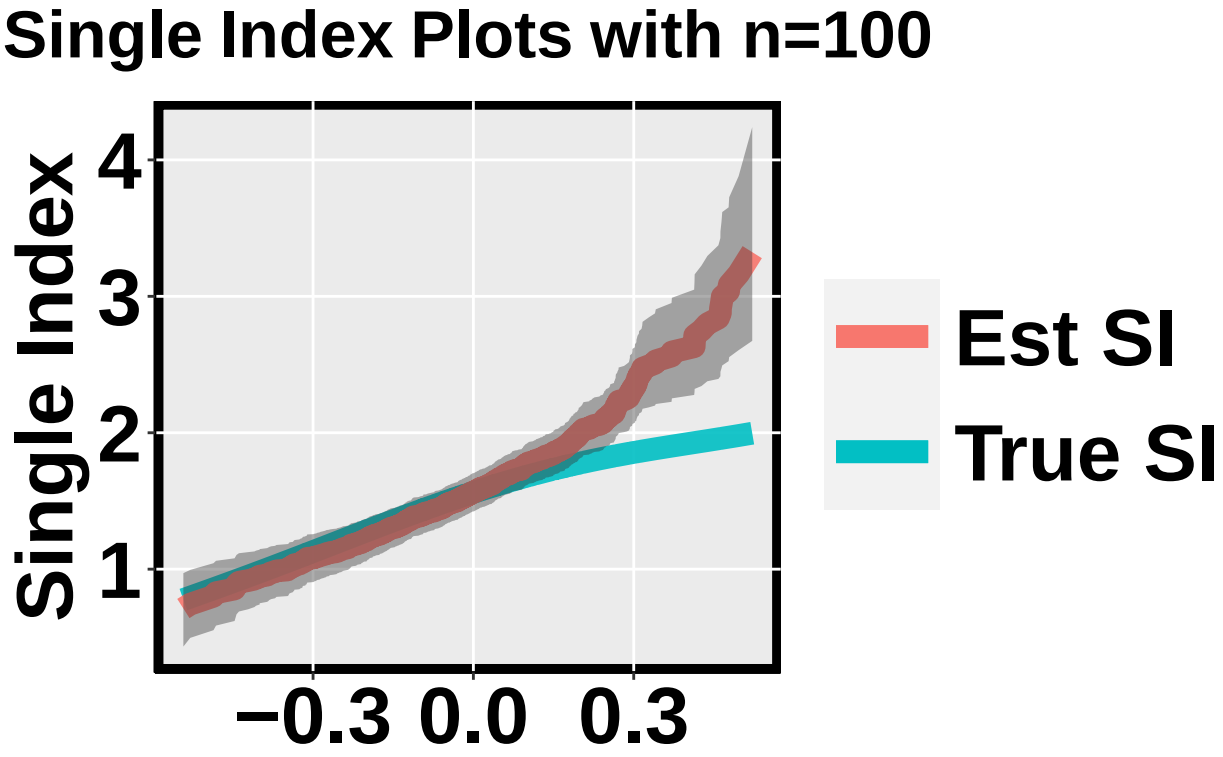
Single Index Plot

```r
si_p_mat <- matrix(0,T,(iter/2))
 for (i in 1:(iter/2)) {
   t_p <- x%*%(abs(fit_ss$beta[i,])/sqrt(sum(fit_ss$beta[i,]^2)))
   t_ps <- sort(t_p)
   si_p_mat[,i] <- D_mat_fun(M,t_ps)%*%as.matrix(fit_ss$phi[i,])
 }
si_p <- apply(si_p_mat, 1, mean)

## Quantiles
si_p_0.025 <- apply(si_p_mat, 1, function(x) quantile(x, 0.025))
si_p_0.975 <- apply(si_p_mat, 1, function(x) quantile(x, 0.975))

#############################################################################
#################          Single Index plot         ##################
#############################################################################
dat_t <- melt(as.matrix(t_n))
dat1 <- melt(as.matrix(si_n))
dat2 <- melt(as.matrix(si_p))
dat3 <- melt(as.matrix(si_p_0.025))
dat4 <- melt(as.matrix(si_p_0.975))

p2<-ggplot(dat1,aes(x=dat_t[,3],y=dat1[,3],color="True SI"))+geom_line(lwd=4,alpha=0.9)+
  geom_line(dat2,mapping=aes(x=dat_t[,3],y=dat2[,3],color="Est SI"),lwd=4,alpha=0.9)+
  geom_ribbon(aes(ymin=dat3[,3], ymax = dat4[,3]),alpha=0.4,colour = NA)
p2 <- p2+theme(panel.grid.minor=element_blank(),axis.line=element_line(colour="black"),
         panel.background = element_rect(colour = "black", size = 3))
p2 <- p2 + labs( x="", y="Single Index",
                 title = expression(bold(paste(Single~Index~Plots~with," n=100" ) )) )
p2 <- p2 + theme(plot.title = element_text(hjust = 0.5,size = 25),
                axis.text = element_text(size=30, face = "bold", colour = "black"),
                axis.title = element_text(size=30, face = "bold") )
p2 <- p2 + theme(legend.title = element_blank(),
                              legend.text = element_text(face = "bold",size = 30),
                              legend.key.size = unit(3, 'lines'))
p2
```


Single Index Plots with n=100
Single Index
4
3
2
1
−0.3
0.0
0.3
Est SI
True SI