Assignment1

Cyril 2010216

4 March 2020

```r
data(GaltonFamilies)
nrep <- 1000
n <- dim(GaltonFamilies)[1]
ntest <- 200
MSE <- data.frame(matrix(0,nrep,5))
MSE_info <- data.frame(matrix(0,2,5))
names(MSE) <- c("baseline",'JS','GMR','Tweedie','MCMC')
names(MSE_info) <- c("baseline",'JS','GMR','Tweedie','MCMC')
row.names(MSE_info) = c('mean','sd')
```

Baseline method

```r
for (i in 1:nrep){
  # partion training data and testing data
  train = sample(1:n, n-ntest, replace = FALSE)
  traindata <- GaltonFamilies[train,]
  testdata <- GaltonFamilies[-train,]
  # Fit a baseline model
  fit_baseline <- lm(I(childHeight - mean(childHeight))~gender +
                     I(midparentHeight -mean(midparentHeight))-1, traindata)
  # make prediction based on the fitted model
  height_baseline_pred <- predict(fit_baseline,testdata) + mean(traindata$childHeight)
  # Evaluate model performance
  MSE[i,1] <- mean((testdata$childHeight - height_baseline_pred)^2)
}
MSE_info[1,1] = mean(MSE[,1])
MSE_info[2,1] = sd(MSE[,1])^2
```

Second approach with James Stein estimator:

```r
js = function(draw) {
  l2 <- sum(draw^2)
  return((1 - (length(draw) - 2) / l2) *draw)
}

for (i in 1:nrep){
  # partion training data and testing data
  train = sample(1:n, n-ntest, replace = FALSE)
  traindata <- GaltonFamilies[train,]
  testdata <- GaltonFamilies[-train,]
  # Fit a JS model
  fit_js =  lm(js(childHeight)~gender + js(midparentHeight), traindata)
  # make prediction based on the fitted model
  height_pred <- predict(fit_js,testdata)
```

```
  #1/(sd(traindata$childHeight)^2+1))*mean(traindata$childHeight)
  # Evaluate model performance
  MSE[i,2] <- mean((testdata$childHeight - height_pred)^2)
}
MSE_info[1,2] = mean(MSE[,2])
MSE_info[2,2] = sd(MSE[,2])^2
```

Third approach: Exponential GLM following Geometric mean regression

```
for (i in 1:nrep){
  # partion training data and testing data
  train = sample(1:n, n-ntest, replace = FALSE)
  traindata <- GaltonFamilies[train,]
  testdata <- GaltonFamilies[-train,]
  # Fit a model
  fit_gme <- glm(js(childHeight)~gender + js(midparentHeight),
                    family = Gamma(link="log"),data=traindata)
  # make prediction based on the fitted model
  height_pred <- predict(fit_gme,testdata,type='response')
  # Evaluate model performance
  MSE[i,3] <- mean((testdata$childHeight - height_pred)^2)
}
MSE_info[1,3] = mean(MSE[,3])
MSE_info[2,3] = sd(MSE[,3])^2
```

Fourth approach: Tweedie

```
xi.vec <- seq(1, 3, by=0.2)
out <- tweedie.profile(GaltonFamilies$childHeight ~1, xi.vec=xi.vec, do.plot=TRUE, verbose=TRUE)
```

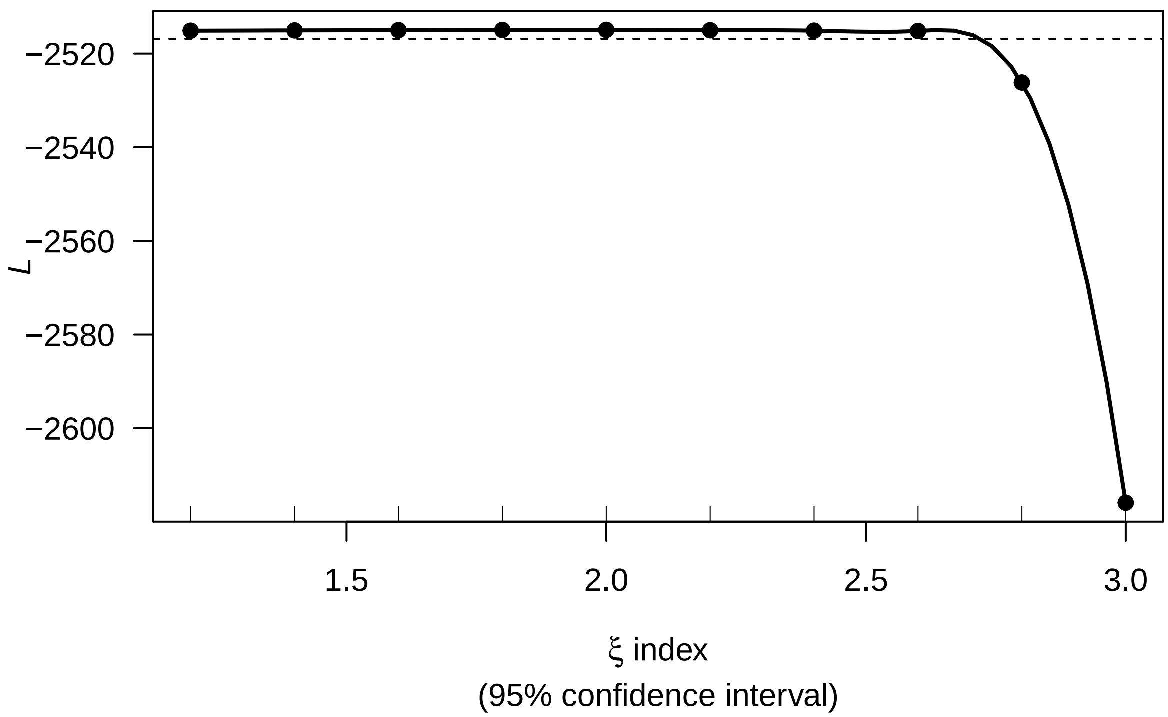
```
## ---
##  This function may take some time to complete;
##  Please be patient.  If it fails, try using  method="series"
##  rather than the default  method="inversion"
##  Another possible reason for failure is the range of p:
##  Try a different input for  p.vec
## ---

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored

## Values of xi between 0 and 1 and less than zero have been removed: such values are not possible.
## 1.2 1.4 1.6 1.8 2 2.2 2.4 2.6 2.8 3
## xi = 1.2
## * Phi estimation, method:  mle (using optimize):  Done (phi = 0.08269415 )
## * Computing the log-likelihood (method = inversion ): Done: L = -2515.106
## xi = 1.4
## * Phi estimation, method:  mle (using optimize):  Done (phi = 0.03567929 )
## * Computing the log-likelihood (method = inversion ): Done: L = -2515.04
## xi = 1.6
## * Phi estimation, method:  mle (using optimize):  Done (phi = 0.01540395 )
## * Computing the log-likelihood (method = inversion ): Done: L = -2514.988
## xi = 1.8
## * Phi estimation, method:  mle (using optimize):  Done (phi = 0.006680226 )
## * Computing the log-likelihood (method = inversion ): Done: L = -2514.952
## xi = 2
```

```
## * Phi estimation, method:  mle (using optimize):  Done (phi = 0.002855578 )
## * Computing the log-likelihood (method = inversion ): Done: L = -2514.931
## xi = 2.2
## * Phi estimation, method:  mle (using optimize):  Done (phi = 0.001215127 )
## * Computing the log-likelihood (method = inversion ): Done: L = -2515.011
## xi = 2.4
## * Phi estimation, method:  mle (using optimize):  Done (phi = 0.0005208739 )
## * Computing the log-likelihood (method = inversion ): Done: L = -2515.097
## xi = 2.6
## * Phi estimation, method:  mle (using optimize):  Done (phi = 0.000224104 )
## * Computing the log-likelihood (method = inversion ): Done: L = -2515.164
## xi = 2.8
## * Phi estimation, method:  mle (using optimize):  Done (phi = 0.0001254246 )
## * Computing the log-likelihood (method = inversion ): Done: L = -2526.174
## xi = 3
## * Phi estimation, method:  mle (using optimize):  Done (phi = 9.026968e-05 )
## * Computing the log-likelihood (method = inversion ): Done: L = -2615.927
## . ---
## * Smoothing:
```

```
## Warning in tweedie.profile(GaltonFamilies$childHeight ~ 1, xi.vec = xi.vec, : Confidence interval ca
```




```r
for (i in 1:nrep){  # partion training data and testing data
  train = sample(1:n, n-ntest, replace = FALSE)
  traindata <- GaltonFamilies[train,]
  testdata <- GaltonFamilies[-train,]
  # Fit a model
```

```
  fit_tweedie <- glm(js(childHeight)~gender + js(midparentHeight),
                     data =traindata,
                     family=tweedie(var.power=out$xi.max,link.power=0))
  # make prediction based on the fitted model
  height_pred <- predict(fit_tweedie,testdata,type='response')
  # Evaluate model performance
  MSE[i,4] <- mean((testdata$childHeight - height_pred)^2)
}
MSE_info[1,4] = mean(MSE[,4])
MSE_info[2,4] = sd(MSE[,4])^2
```

Fifth Approach: Stan

```
glm_mcmc = stan_glm(js(childHeight) ~gender + js(midparentHeight),data=GaltonFamilies, family=Gamma(lin
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.002 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 20 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.661 seconds (Warm-up)
## Chain 1:                2.009 seconds (Sampling)
## Chain 1:                3.67 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1.77 seconds (Warm-up)
## Chain 2:                1.434 seconds (Sampling)
## Chain 2:                3.204 seconds (Total)
## Chain 2:
```

```
summary(glm_mcmc)
```

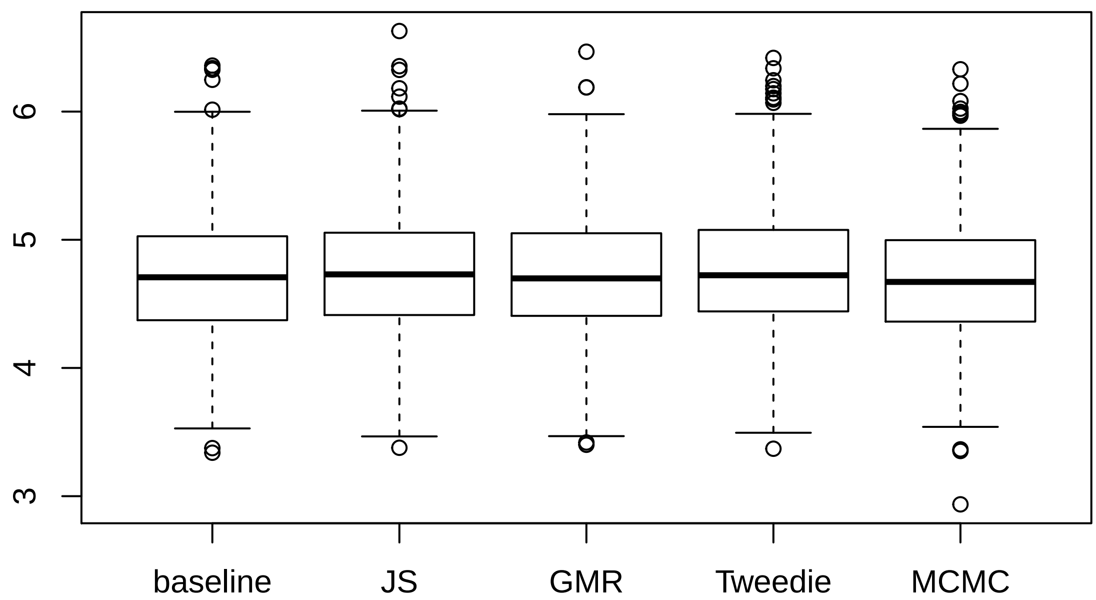
```
##
## Model Info:
##  function:     stan_glm
##  family:       Gamma [log]
##  formula:      js(childHeight) ~ gender + js(midparentHeight)
##  algorithm:    sampling
##  sample:       2000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 934
##  predictors:   3
##
## Estimates:
##                      mean   sd    10%   50%   90%
## (Intercept)          3.4    0.1   3.4   3.4   3.5
## gendermale           0.1    0.0   0.1   0.1   0.1
## js(midparentHeight)  0.0    0.0   0.0   0.0   0.0
## shape              312.5   14.7 293.9 311.9 331.9
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 66.7    0.2 66.5  66.7  66.9
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                     mcse Rhat n_eff
## (Intercept)         0.0  1.0  2933
## gendermale          0.0  1.0  1781
## js(midparentHeight) 0.0  1.0  2925
## shape               0.6  1.0   628
## mean_PPD            0.0  1.0  1695
## log-posterior       0.1  1.0   764
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```
for (i in 1:nrep){
  # partion training data and testing data
  train = sample(1:n, n-ntest, replace = FALSE)
  traindata <- GaltonFamilies[train,]
  testdata <- GaltonFamilies[-train,]
  #for(j in 1:nrow(testdata)) testdata$midparentHeight[j] =      js(c(traindata$midparentHeight,testda
```

```
  # make prediction based on the fitted model
  height_pred <- posterior_predict(glm_mcmc,testdata)
  # Evaluate model performance
  MSE[i,5] <- mean((testdata$childHeight - apply(height_pred,2,mean))^2)
}
MSE_info[1,5] = mean(MSE[,5])
MSE_info[2,5] = sd(MSE[,5])^2
```

Results

```
boxplot(MSE)
```




```
print(MSE_info)
```

```
##       baseline        JS       GMR   Tweedie      MCMC
## mean 4.7159450 4.7396502 4.7296105 4.7593013 4.6898292
## sd   0.2235418 0.2279211 0.2207221 0.2378679 0.2183873
```