Advanced Bayesian Methods - Assignment 3

통계데이터사이언스학과
박주연 (2022311137)

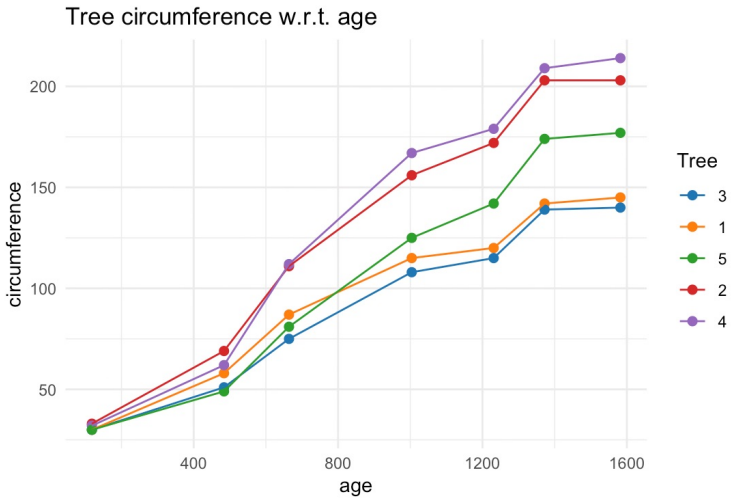
Q1

Nonlinear mixed model is given by

$$y_{ij} = \frac{\beta_1 + u_i}{1 + \exp\{-(AGE_{ij} - \beta_2)/\beta_3\}} + \varepsilon_{ij} \quad , \; u_i \sim N(0, \tau^2) \; , \; \varepsilon_{ij} \sim N(0, \sigma^2)$$

(a)

In following graph, the overall line resembles a logistic function. The variance of the 'circumference' increase with increasing 'age'. Thus it seems the data has heteroscedasticity. Furthermore, the sample standard deviation varies greatly depending on the tree.




```
> print(tree_stat)
  Tree group_mean group_sd
1    3   94.00000 42.98062
2    1   99.57143 43.29302
3    5  111.14286 58.85980
4    2  135.28571 66.32424
5    4  139.28571 71.89741
```

(b)

Based on the result of (a), it seems to be a with-in variability.

As a result, I give an informative prior for σ^2 and noninformative prior for the others.

① prior

$\sigma^2 \sim Gamma(3, 0.5)$ $\quad \tau^2 \sim Inv\text{-}Gamma(0.01, 0.01)$, $\underset{\sim}{\beta} = (\beta_1 \; \beta_2 \; \beta_3)' \sim N_3(0_3, 1000^2 I_3)$

② Result of MCMC

Posterior mean of τ^2 are much bigger than the posterior mean of σ^2. Therefore, the total variance of y is explained more by the between-group variance.

```
Inference for Stan model: bfb06beb4283423d17745c77181b8476.
4 chains, each with iter=3000; warmup=2000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

        mean se_mean    sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
tau1    0.00    0.00  0.00    0.00    0.00    0.00    0.00    0.00  1355 1.00
tau2    0.02    0.00  0.00    0.01    0.01    0.02    0.02    0.02  2167 1.00
beta1 193.25    0.95 24.50  143.48  179.52  192.72  206.71  246.15   661 1.01
beta2 736.96    0.93 39.79  667.11  708.94  733.17  761.97  821.15  1838 1.00
beta3 354.92    0.72 29.44  301.90  334.82  352.85  373.83  417.97  1691 1.00
u[1]  -36.98    0.91 23.76  -87.62  -49.60  -36.59  -23.79   11.21   676 1.00
u[2]  -29.22    0.93 23.86  -78.49  -42.16  -28.76  -16.07   20.32   658 1.01
u[3]   -4.60    0.92 23.80  -55.41  -17.12   -4.39    8.39   44.88   668 1.01
u[4]   32.86    0.92 23.73  -16.98   19.97   32.71   45.96   82.13   666 1.01
u[5]   41.37    0.92 23.83   -7.55   28.60   41.09   54.20   90.68   668 1.01
tau    45.73    0.87 23.50   20.89   30.74   39.59   53.37  108.15   736 1.01
sigma   8.25    0.02  1.14    6.42    7.42    8.13    8.90   10.81  2104 1.00
```

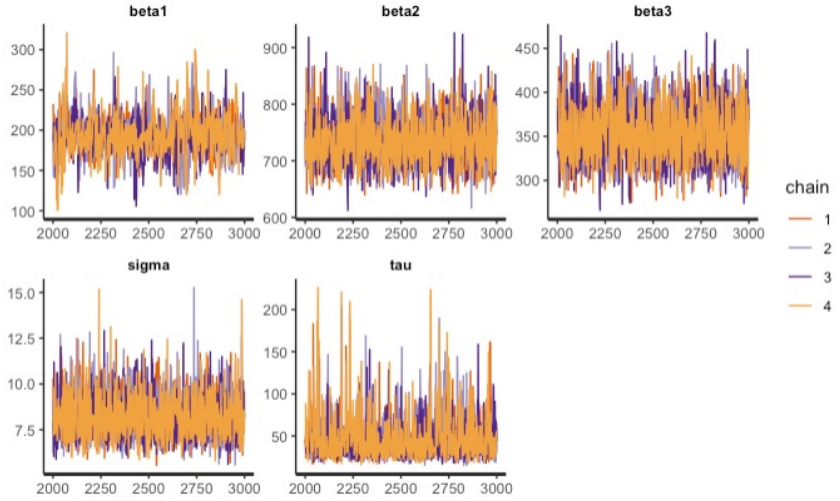
$\tau^2 \rightarrow$ tau

$\sigma^2 \rightarrow$ sigma

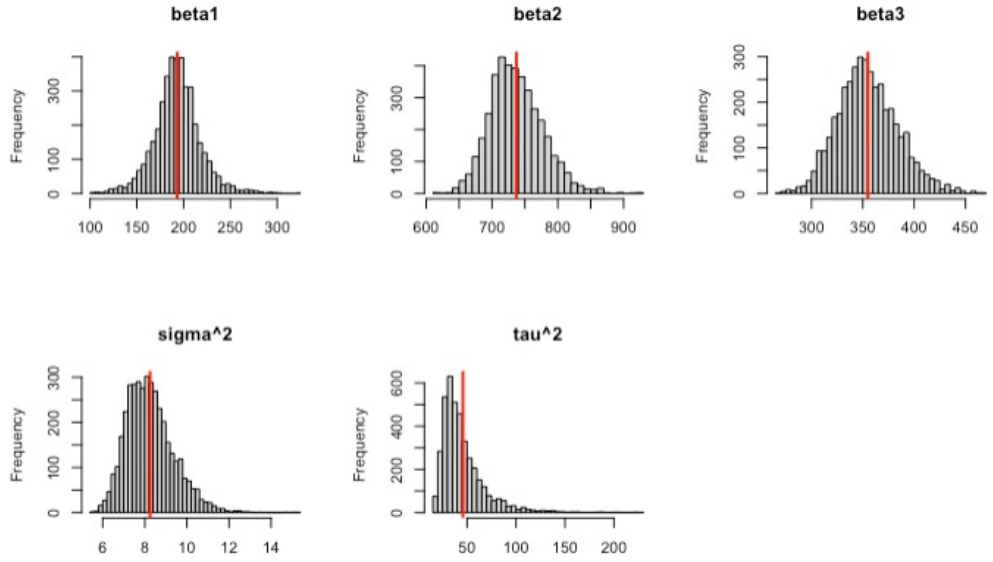
③ Model Diagnostic

· Traceplot

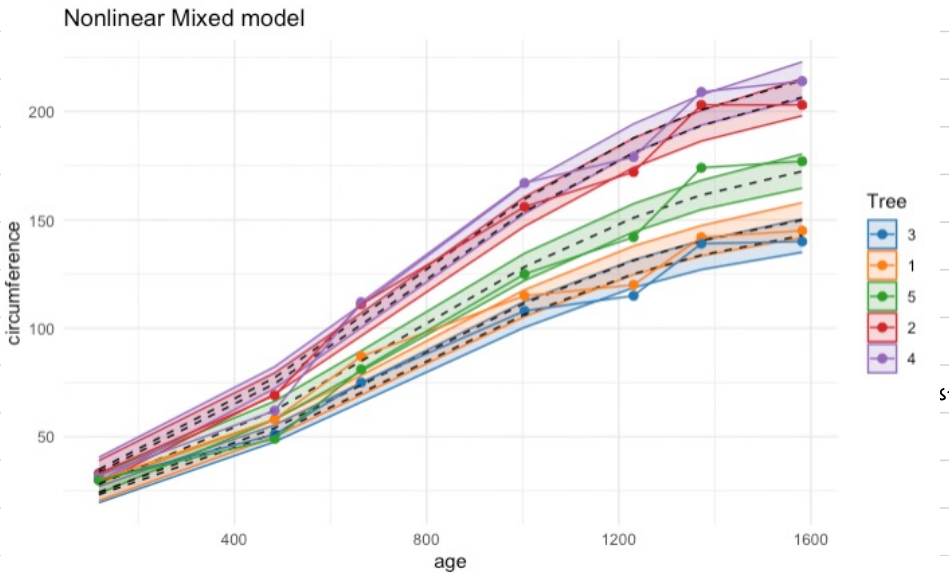
The chains are converged because the traceplots are bounded and wiggled



- Histogram of samples



· Visualization



sterior mean per Tree group

(c) Test quantity should reflect the homoscedastic.

① Define test quantity.

To make the test quantity reflects the homoscedastic, we should check each group has different variability. It can be confirmed by comparing the mean of error sum of squares. Suppose the group variance is defined as

$$\delta_i^2 = \frac{1}{n_i}\sum_{j=1}^{n_i}(y_{ij} - \mu_i)^2$$

where group mean $\mu_i = \frac{1}{n_i}\sum_{j=1}^{n_i} y_{ij}$. Then the test quantity is defined as

$$T(y, \theta) = \frac{1}{m}\sum_{i=1}^{m}(\delta_i^2 - \hat{\delta}_i^2)^2$$

② Posterior predictive check with model C1)

```
# predictive posterior p value
mean(group_diff_ss_rep > group_diff_ss_obs)
```

```
[1] 0.31925
```

The posterior predictive p-value is 0.319. It is not enough to judge the Homoscedasticity only with one model. So let's compare it with the comparative model.

③ Model with Heteroscedastic assumption

The model is

$$y_{ij} = \frac{\beta_1 + u_i}{1 + \exp\{-(AGE_{ij} - \beta_2)/\beta_3\}} + \varepsilon_{ij}, \quad u_i \sim N(0, \tau^2), \quad \varepsilon_i \sim N(0, \delta_i^2)$$

Then the posterior predictive p-value is 0.261.

```
# predictive posterior p value
mean(group_diff_ss_rep > group_diff_ss_obs)
```

```
[1] 0.261
```

Since the model with heteroscedasticity has lower pppvalue, we can conclude that the model in C1) is more appropriate for 'Orange' dataset.

④ Result

Since the model with heteroscedasticity has lower pppvalue, we can conclude that the model in C1) is more appropriate for 'Orange' dataset. Although it does not seems to satisfy the homoscedastic assumption, the variance according to tree group seems to have already explained through τ^2 in model C1). we've already seen that the total variance of y is explained more by between variance than within-variance.

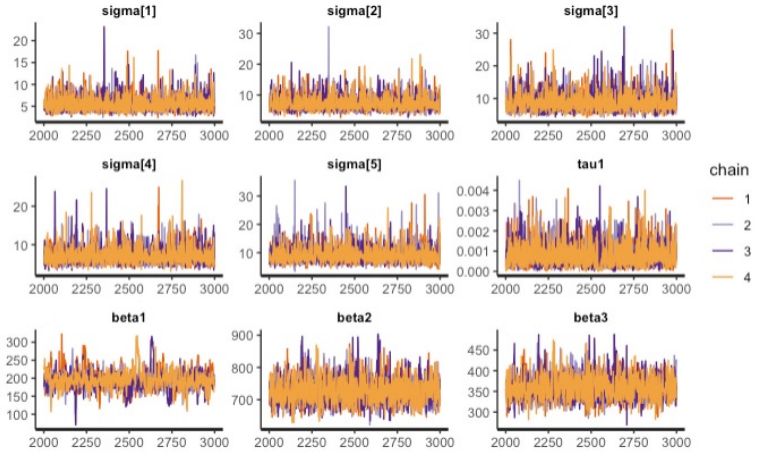
[Result of model in (c)]

- Result of stan

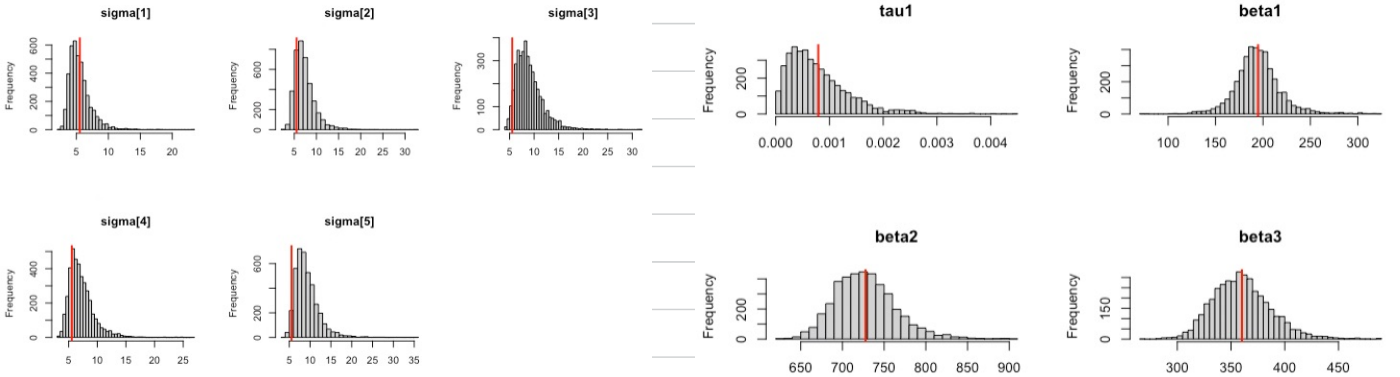
```
Inference for Stan model: f9ca6798a14637046260af5dd26a1463.
4 chains, each with iter=3000; warmup=2000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
tau1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2073	1.00
beta1	194.83	1.21	24.58	147.87	181.13	193.67	206.79	249.78	409	1.01
beta2	727.69	0.81	37.17	663.72	701.63	724.50	749.02	810.89	2116	1.00
beta3	360.15	0.58	27.39	312.84	340.96	358.50	376.48	419.99	2244	1.00
u[1]	-39.38	1.23	23.75	-93.41	-50.95	-38.42	-26.31	7.33	374	1.01
u[2]	-31.38	1.24	23.81	-86.75	-42.89	-30.11	-18.22	14.12	369	1.01
u[3]	-6.85	1.23	23.82	-62.99	-18.36	-5.79	6.17	38.98	374	1.01
u[4]	30.71	1.22	23.69	-24.01	19.09	31.50	43.24	77.90	375	1.01
u[5]	38.83	1.22	23.61	-15.07	27.10	39.67	51.57	85.35	376	1.01
tau	45.35	0.95	24.90	20.76	30.58	38.92	52.32	108.62	681	1.01
sigma[1]	5.54	0.03	1.72	3.34	4.38	5.18	6.28	9.75	2870	1.00
sigma[2]	7.29	0.04	2.30	4.21	5.74	6.84	8.32	13.19	3081	1.00
sigma[3]	9.02	0.05	2.81	5.28	7.05	8.48	10.32	15.57	2656	1.00
sigma[4]	7.19	0.05	2.25	4.22	5.69	6.75	8.20	12.93	2473	1.00
sigma[5]	9.23	0.06	2.91	5.50	7.28	8.69	10.46	16.56	2683	1.00

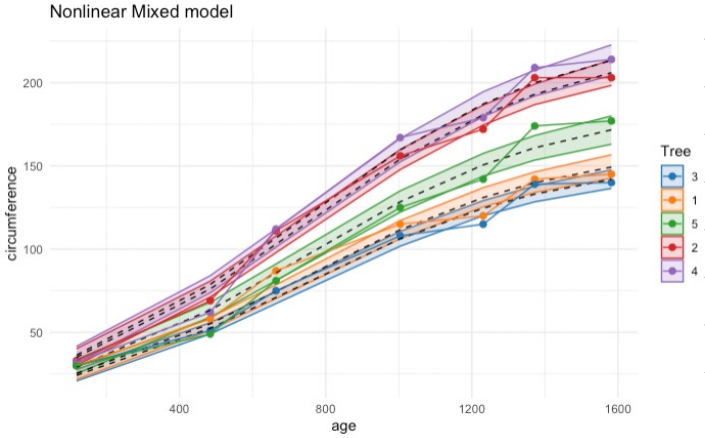
- Traceplot



- Histogram of samples



- Visualization

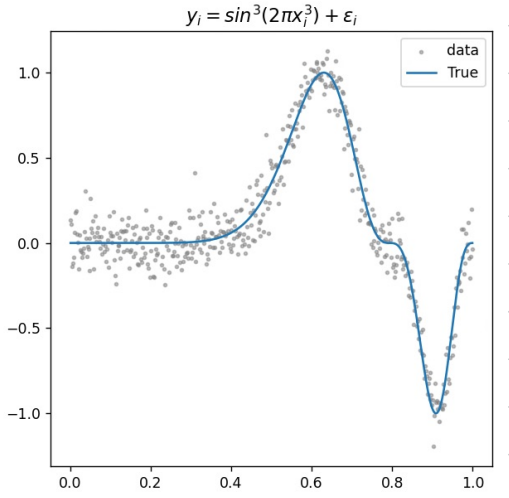


Q2

The nonparametric regression model is given as

$y_i \sim \sin^3(2\pi x_i^3) + \varepsilon_i$, $\varepsilon_i \sim N(0, 0.1^2)$, $i = 1, \cdots, n$.

Let $x_i = (2i-1)/1000$, $i = 1, \cdots, 500$. Then we can generate simulated data X and y. The data is follows.



(a) Verify that the column spaces of the three design matrices are identical.

To demonstrate that three design matrices span the same column space, we can show that their respective projection matrices are identical.

- Steps 1) Define uniform interior knots and basis functions
- step 2) Generate design matrices 'W_truncated', 'W_polynomial' and 'W_bspline'
- step 3) Compare the projection matrices 'proj_matrix_truncated', 'proj_matrix_polynomial' and 'proj_matrix_bspline'.

Do they span same column spaces?

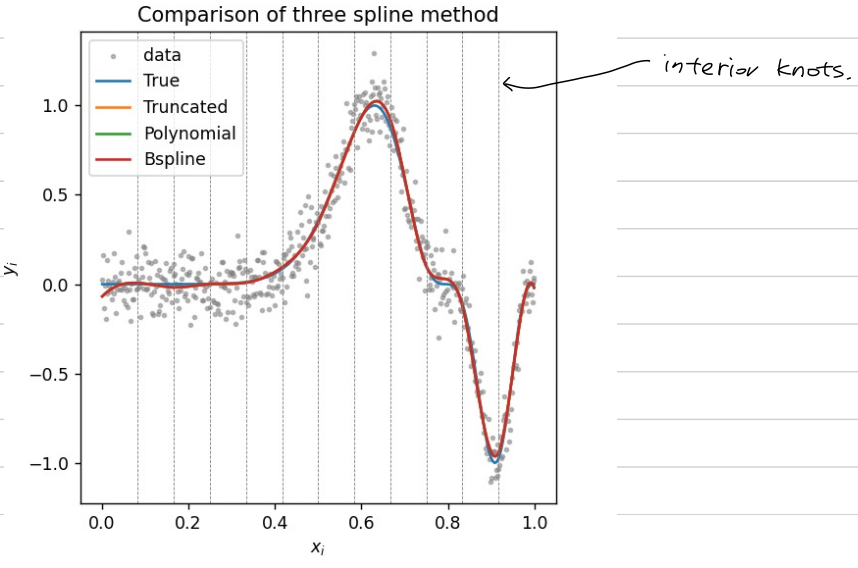
```
proj_matrix_truncated = Generate_proj_matrix(W_truncated)
proj_matrix_polynomial = Generate_proj_matrix(W_polynomial)
proj_matrix_bspline = Generate_proj_matrix(W_bspline)

# Check whether the projection matrices are the same in rounding 4 digits.
print(f'Are truncated projection matrix and polynomial projected matrix same?')
print(np.allclose(proj_matrix_truncated, proj_matrix_polynomial, rtol=0.01))
print(f'Are truncated projection matrix and bspline projected matrix same?')
print(np.allclose(proj_matrix_truncated, proj_matrix_bspline, rtol=0.01))
```

✓ 0.1s Python

```
Are truncated projection matrix and polynomial projected matrix same?
True
Are truncated projection matrix and bspline projected matrix same?
True
```

Moreover, through the graph below, all three fitted regression curves approximate the true function well. (The three fitted functions are overlapped.)



(b)

From the given likelihood and prior, we can drive posterior as

likelihood

$p(y|\beta_H, \sigma^2, H) \sim N_H\left(W_H\beta_H, \sigma^2 I_n\right)$

Prior

1. $p(\sigma^2) \propto (\sigma^2)^{-1}$
2. $p(H) = p(L) \sim Pois(1)$
3. $\beta_H|\sigma^2, H \sim N_H\left(0, g\sigma^2(W_H^T W_H)^{-1}\right)$ (g-prior)

Posterior

1. $p(\beta_H|\sigma^2, H, y) \sim N_H\left(\frac{g}{1+g}(W_H^T W_H)^{-1}W_H^T y, \frac{g\sigma^2}{1+g}(W_H^T W_H)^{-1}\right)$
2. $p(\sigma^2|H, y) \sim Inv-Gamma\left(\frac{n+1}{2}, \frac{1}{2}y^T(I_n - \frac{g}{g+1}W_H(W_h^T W_H)^{-1}W_H^T)y\right)$
3. $p(H|y) \propto p(H)\{\frac{1}{2}y^T(I_n - \frac{g}{g+1}W_H(W_h^T W_H)^{-1}W_H^T)y\}^{-\frac{n+1}{2}}$

※ The detailed derivation is attached in the Appendix.

To do model averaging method, the main steps are

Step 1. Calculate the normalized marginal posterior $p(H|y)$

To obtain the exact marginal posterior probability of H, assume that H can have maximum value of 30. Then the unnormalized posterior can be calculated of all the case H, and the normaling constant can be computed to obtain the normalized posterior $p(H|y)$.

By setting $max(H) = 30$, the normalized posterior $p(H|y)$ is

```
max_L = 30
posterior_H = posterior_H(X, max_L)
posterior_H
```

✓ 0.2s

```
array([0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.09947506, 0.89921775, 0.00115973, 0.00008728, 0.00000481,
       0.00000155, 0.00002809, 0.00002439, 0.00000111, 0.00000022,
       0.00000001, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ])
```

Step 2. Pointwise posterior of $\mu^{(t)}(x_0)$

① Draw $H^{(t)}$ from $p(H|y)$

② Given $H^{(t)}$, draw σ^2 from $p(\sigma^2|H, y)$

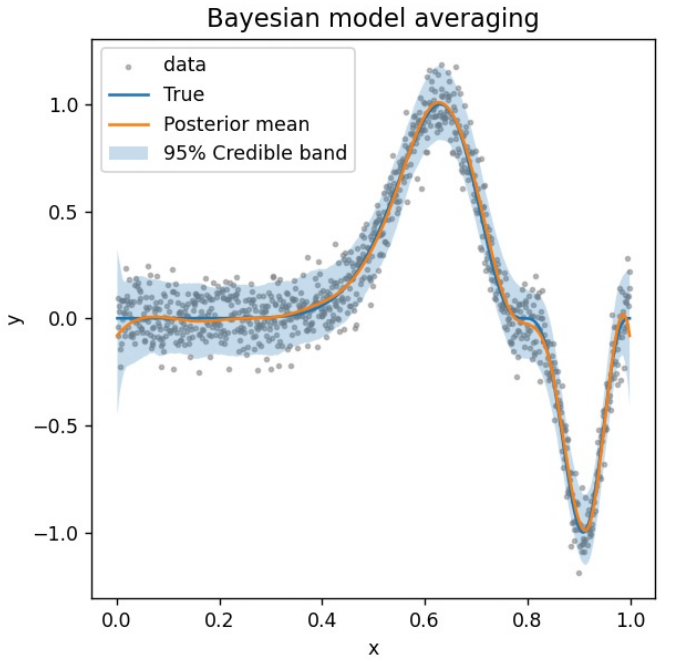
③ Given σ^2 and $H^{(t)}$, draw $\beta_1^{(t)}, \cdots, \beta_{H^{(t)}}^{(t)}$ from $p(\beta|\sigma^2, H, y)$

Step 3. Model averaging

The pointwise posterior is averaged over different models.

Result

We can see that the posterior mean almost coincides with the true function, and most of the observations fall within the 95% credible interval.



Appendix 1 - Detailed derivation of posterior in Q2

1. $p(\beta_H|\sigma^2, H, y)$

$$\begin{aligned}
p(\beta_H|\sigma^2, H, y) &\propto p(y, \beta_H|\sigma^2, H) \\
&= p(y|\beta_H, \sigma^2, H)p(\beta_H|\sigma^2, H) \\
&= N_H\left(W_H\beta_H, \sigma^2 I_n\right) \times N_H\left(0, g\sigma^2(W_H^TW_H)^{-1}\right) \\
&= (\sigma^2)^{-n/2}|g\sigma^2(W_H^TW_H)^{-1}|^{-1/2} \times exp\left(-\frac{1}{2}\{(y-W_H\beta_H)^T\frac{1}{\sigma^2}(y-W_H\beta_H) + \beta_H^T(\frac{1}{g\sigma^2}W_H^TW_H)\beta_H\}\right) \\
&\propto exp\left(-\frac{1}{2\sigma^2}\{(y-W_H\beta_H)^T(y-W_H\beta_H) + \beta_H^T(\frac{1}{g\sigma^2}W_H^TW_H)\beta_H\}\right) \\
&= exp\left(-\frac{1}{2\sigma^2}\{\beta_H^T(W_H^TW_H + \frac{1}{g}W_H^TW_H)\beta_H + 2(y^TW_H)\beta_H + y^Ty\}\right) \\
&\propto exp\left(-\frac{1}{2}(\beta_H-\mu_\beta)^T\Sigma_\beta^{-1}(\beta_H-\mu_\beta)\right) \quad \text{(for some } \mu_\beta,\ \Sigma_\beta\text{)}
\end{aligned}$$

For Some μ_β, Σ_β,

$$\Sigma_\beta = \frac{1}{\sigma^2}(W_H^TW_H + \frac{1}{g}W_H^TW_H)^{-1},\ \mu_\beta^T\Sigma_\beta^{-1} = \frac{1}{\sigma^2}y^TW_H$$
$$\therefore \Sigma_\beta = \frac{g\sigma^2}{1+g}(W_H^TW_H)^{-1},\ \mu_\beta = \frac{g}{1+g}(W_H^TW_H)^{-1}W_H^Ty$$

Therefore, $p(\beta_H|\sigma^2, H, y) \sim N_H\left(\frac{g}{1+g}(W_H^TW_H)^{-1}W_H^Ty, \frac{g\sigma^2}{1+g}(W_H^TW_H)^{-1}\right)$

2. $p(\sigma^2|H, y)$

$$\begin{aligned}
p(\sigma^2|H, y) &\propto p(y, \sigma^2|H) \\
&= \int p(y, \beta_H, \sigma^2|H)d\beta_H \\
&= \int p(y|\beta_H, \sigma^2, H)p(\beta_H|\sigma^2, H)p(\sigma^2)d\beta_H \\
&\propto (\sigma^2)^{-1}\int p(y|\beta_H, \sigma^2, H)p(\beta_H|\sigma^2, H)d\beta_H \\
&= (\sigma^2)^{-1}\int N_H\left(W_H\beta_H, \sigma^2 I_n\right) \times N_H\left(0, g\sigma^2(W_H^TW_H)^{-1}\right)d\beta_H \\
&= (\sigma^2)^{-n/2-1}|g\sigma^2(W_H^TW_H)^{-1}|^{-1/2} \times \int exp\left(-\frac{1}{2\sigma^2}\{\beta_H^T(W_H^TW_H + \frac{1}{g}W_H^TW_H)\beta_H + 2(y^TW_H)\beta_H + y^Ty\}\right)d\beta_H \\
&\propto (\sigma^2)^{-(n+1)/2-1}\int exp\left(-\frac{1}{2\sigma^2}\{\beta_H^T(W_H^TW_H + \frac{1}{g}W_H^TW_H)\beta_H + 2(y^TW_H)\beta_H + y^Ty + A\}\right)exp(\frac{1}{2\sigma^2}A)d\beta_H \quad \text{(for some } A\text{)} \\
&= (\sigma^2)^{-(n+1)/2-1}exp\left(-\frac{1}{\sigma^2}\left[\frac{1}{2}y^T(I_n - \frac{g}{g+1}W_H(W_h^TW_H)^{-1}W_H^T)y\right]\right) \\
&\sim Inv-Gamma\left(\frac{n+1}{2}, \frac{1}{2}y^T(I_n - \frac{g}{g+1}W_H(W_h^TW_H)^{-1}W_H^T)y\right)
\end{aligned}$$

Since $\beta_H|\sigma^2, H, y \sim N_H\left(\frac{g}{1+g}(W_H^TW_H)^{-1}W_H^Ty, \frac{g\sigma^2}{1+g}(W_H^TW_H)^{-1}\right)$, we can find A such that $\frac{1}{\sigma^2}y^Ty + \frac{1}{\sigma^2}A = \mu_\beta\Sigma_\beta^T\mu_\beta$. And A is

$$\begin{aligned}
A &= \sigma^2\mu_\beta\Sigma_\beta^T\mu_\beta - y^Ty \\
&= y^T\frac{g}{g+1}W_H(W_h^TW_H)^{-1}W_H^Ty - y^Ty \\
&= y^T(\frac{g}{g+1}W_H(W_h^TW_H)^{-1}W_H^T - I_n)y
\end{aligned}$$

3. $p(H|y)$

$$\begin{aligned}
p(H|y) &\propto p(H)p(y|H) \\
&= p(H)\iint p(y|\beta_H, \sigma^2, H)p(\beta_H|\sigma^2, H)p(\sigma^2)d\beta_H d\sigma^2 \\
&= p(H)\int (\sigma^2)^{-(n+1)/2-1}exp\left(-\frac{1}{\sigma^2}\left[\frac{1}{2}y^T(I_n - \frac{g}{g+1}W_H(W_h^TW_H)^{-1}W_H^T)y\right]\right)d\sigma^2 \\
&= p(H)\{\frac{1}{2}y^T(I_n - \frac{g}{g+1}W_H(W_h^TW_H)^{-1}W_H^T)y\}^{-\frac{n+1}{2}}
\end{aligned}$$

Appendix 2 – Code for Q1 and Q2

Q1 (in R)

```
setwd("~/wndus1712@gmail.com - Google Drive/내 드라이브/Lecture/02_BayesianAdv/hw3")
getwd()
library(rstan); library(ggplot2); library(dplyr)

data('Orange')
head(Orange)
summary(Orange) # 35 by 3
class(Orange$Tree)
class(Orange$age)
class(Orange$circumference)

#############################################################################
##
## Model (1): assume homoscedasticity
##
#############################################################################

#############################################################################
# 1. EDA
#############################################################################
colors = c("#1F77B4", "#FF7F0E", "#2CA02C", "#D62728", "#9467BD")

Orange %>%
  mutate(Tree = as.factor(Tree)) %>%
  ggplot(aes(x=age, y=circumference, group=Tree, color=Tree)) +
  geom_point(size=2) +
  geom_line() +
  scale_color_manual(values = colors) +
  labs(x='age',
       y='circumference',
       color='Tree',
       title='Tree circumference w.r.t. age') +
  theme_minimal()

## Check homoscedastic assumption
tree_means <- aggregate(Orange$circumference, by=list(Tree=Orange$Tree), FUN=mean)
tree_sd <- aggregate(Orange$circumference, by=list(Tree=Orange$Tree), FUN=sd)
tree_stat <- data.frame(tree_means, tree_sd$x)
colnames(tree_stat) <- c('Tree', 'group_mean', 'group_sd')
print(tree_stat)

#############################################################################
# 2. Model
# Group indicator: Orange$Tree
# predictor variable: Orange$age
# response variable: Orange$circumference
# number of observations: n=35
# number of group: L=5
# number of predictor variable: p=1
# prior
# - noninformative prior for beta1, beta2, beta3
# - informative prior for tau1, tau2
```

```
#############################################################################
## Define model
stanmodel <- "
data {
  int<lower=0> N;                  // num of observation 35
  int<lower=1> L;                  // num of group 5
  real<lower=0> y[N];              // response variable
  int<lower=1,upper=L> ll[N];   // group indicator
  real x[N];                       // predictor variable
}
parameters {
  real<lower=0> tau1;   // var of random effect u[L] (scalar, because of homoscedasticity)
  real<lower=0> tau2;   // var of epsilon
  real beta1;              // fixed effect
  real beta2;              // fixed effect
  real beta3;              // fixed effect
  real u[L];               // mixed effect
}
transformed parameters {
  real tau;
  real sigma;   // normal function in stan gets std for scale parameter
  real m[N];
  for (i in 1:N){
    m[i] = (beta1 + u[ll[i]]) / (1 + exp(-(x[i] - beta2) / beta3));
  }
  sigma = 1 / sqrt(tau2);
  tau = 1 / sqrt(tau1);
}
model {
  // priors for fixed effect
  beta1 ~ normal(0.0, 1000);
  beta2 ~ normal(0.0, 1000);
  beta3 ~ normal(0.0, 1000);   // noninformative prior
  tau2 ~ gamma(0.3, 0.5);
  // prior for random effect
  tau1 ~ gamma(0.01, 0.01);
  for (l in 1:L){
    u[l] ~ normal(0, tau);
  }
  // likelihood
  for (n in 1:N){
   y[n] ~ normal(m[n], sigma);
  }
}
generated quantities{
  // calculated when the mcmc sampling is done
  real y_mean[N];
  real y_rep[N];
  //real ppp_value_group_mean;
  for(i in 1:N){
    // Posterior parameter distribution of the mean
    y_mean[i] = (beta1 + u[ll[i]]) / (1 + exp(-(x[i] - beta2) / beta3));
    // Posterior predictive distribution
    y_rep[i] = normal_rng(y_mean[i], sigma);
  }
}"
```

```
## Model fitting
n = 35; L = 5
data = list(y=Orange$circumference,
                N=n,
                L=L,
                x=Orange$age,
                ll=as.integer(Orange$Tree)
)

fit <- stan(
  model_code = stanmodel,
  data = data,
  chain = 4,
  warmup = 2000,
  iter = 3000,
  cores = 4
)

print(fit)

#############################################################################
## 3. Model diagnostic
#############################################################################
## Convergence diagnostic
# a. traceplot
traceplot(fit,pars=c("beta1", "beta2", "beta3", "sigma", "tau"))

# b. histogram of parameters
param = extract(fit)
par(mfrow=c(2, 3))
hist(param$beta1, breaks=40, main='beta1', xlab='')
abline(v=mean(param$beta1), col='red', lwd=2)
hist(param$beta2, breaks=40, main='beta2', xlab='')
abline(v=mean(param$beta2), col='red', lwd=2)
hist(param$beta3, breaks=40, main='beta3', xlab='')
abline(v=mean(param$beta3), col='red', lwd=2)
hist(param$sigma, breaks=40, main='sigma^2', xlab='')
abline(v=mean(param$sigma), col='red', lwd=2)
hist(param$tau, breaks=40, main='tau^2', xlab='')
abline(v=mean(param$tau), col='red', lwd=2)

## Visualize
y_mean <- extract(fit, "y_mean")
y_mean_cred <- apply(y_mean$y_mean, 2, quantile, c(0.05, 0.95))
y_mean_mean <- apply(y_mean$y_mean, 2, mean)

y_rep <- extract(fit, "y_rep")
y_rep_cred <- apply(y_rep$y_rep, 2, quantile, c(0.05, 0.95))
y_rep_mean <- apply(y_rep$y_rep, 2, mean)   ## y replicated

df <- data.frame(
  age = Orange$age,
  circumference = Orange$circumference,
  group = as.factor(Orange$Tree),
  lower = y_mean_cred[1,], # Assuming this is the lower bound of the CI
  upper = y_mean_cred[2,],  # Assuming this is the upper bound of the CI
  mean_prediction = y_mean_mean
```

```
)

ggplot(df, aes(x=age, y=circumference, group=group, color=group)) +
  geom_ribbon(aes(ymin=lower, ymax=upper, fill=group), alpha=0.2) +
  geom_point(size=2) +
  geom_line(aes(y=mean_prediction), linetype="dashed", color="black") +
  geom_line() +
  scale_color_manual(values = colors) +
  scale_fill_manual(values = colors) +
  labs(x='age',
       y='circumference',
       color='Tree',
       fill='Tree',
       title='Nonlinear Mixed model') +
  theme_minimal()

#############################################################################
## 4. Posterior predictive check
#############################################################################
## Posterior predictive check
par(mfrow=c(1,1))
res.obs=sqrt(colMeans((Orange$circumference - t(y_mean$y_mean))^2))
res.rep=sqrt(rowMeans((y_rep$y_rep-y_mean$y_mean)^2))          # just same as the posterior dist of
 sigma; hist(1/sqrt(param$tau))
plot(res.obs,res.rep); abline(a=0,b=1,col="red")

chisq.obs=colMeans((Orange$circumference - t(y_mean$y_mean))^2/t(y_mean$y_mean))
chisq.rep=rowMeans((y_rep$y_rep-y_mean$y_mean)^2/y_mean$y_mean)
plot(chisq.obs,chisq.rep); abline(a=0,b=1,col="red")

## Posterior predictive p-value
# Group matrix
group <- matrix(0, n_mcmc, 35)
for (i in 1:n_mcmc){
  group[i, ] <- Orange$Tree
}
# Define residual test quantity
group_diff_ss_rep <- rep(0, n_mcmc)
group_diff_ss_obs <- rep(0, n_mcmc)
for(k in 1:n_mcmc){
  # within group variability of estimated y and replicated y
  group_sd_mean <- rep(0, 5)
  group_sd_rep <- rep(0, 5)
  group_sd_obs <- rep(0, 5)
  for(i in 1:5){
    group_sd_mean[i] <- sd(y_mean$y_mean[k, ][group[k, ] == i])
    group_sd_rep[i] <- sd(y_rep$y_rep[k, ][group[k, ] == i])
    group_sd_obs[i] <- sd(Orange$circumference[group[k, ] == i])
  }
  # sum of squared within group variability
  group_diff_rep <- group_sd_rep - group_sd_mean
  group_diff_obs <- group_sd_obs - group_sd_mean
  group_diff_ss_rep[k] <- (t(group_diff_rep) %*% group_diff_rep)/5
  group_diff_ss_obs[k] <- (t(group_diff_obs) %*% group_diff_obs)/5
}

# predictive posterior p value
```

```
mean(group_diff_ss_rep > group_diff_ss_obs)



#############################################################################
##
## Model (c): assume heteroscedasticity
## epsilon_i ~ N(0, sigma^2_i), i = 1, ..., m (# group)
##
#############################################################################

#############################################################################
## 1. Model
#############################################################################
## Define model
stanmodel <- "
data {
  int<lower=0> N;                  // num of observation 35
  int<lower=1> L;                  // num of group 5
  real<lower=0> y[N];              // response variable
  int<lower=1,upper=L> ll[N];   // group indicator
  real x[N];                       // predictor variable
}
parameters {
  real<lower=0> tau1;        // variance for random effect ui
  real<lower=0> tau2[L];   // var of epsilon, no homoscedasticity assumption
  real beta1;              // fixed effect
  real beta2;              // fixed effect
  real beta3;              // fixed effect
  real u[L];               // mixed effect
}
transformed parameters {
  real tau;
  real sigma[L];   // normal function in stan gets std for scale parameter
  real sigma_temp[N];   // for sampling y
  real m[N];

  tau = 1 / sqrt(tau1);
  for (l in 1:L){
    sigma[l] = 1 / sqrt(tau2[l]);
  }
  for (i in 1:N){
    m[i] = (beta1 + u[ll[i]]) / (1 + exp(-(x[i] - beta2) / beta3));
    sigma_temp[i] = sigma[ll[i]];
  }
}
model {
  // priors for fixed effect
  beta1 ~ normal(0.0, 1000);
  beta2 ~ normal(0.0, 1000);
  beta3 ~ normal(0.0, 1000);   // noninformative prior

  // prior for random effect
  tau1 ~ gamma(0.01, 0.01);
  for (l in 1:L){
    u[l] ~ normal(0, tau);
  }
```

```
  // likelihood
  for (i in 1:N){
    y[i] ~ normal(m[i], sigma_temp[i]);
  }
}
generated quantities{
  // calculated when the mcmc sampling is done
  real y_mean[N];
  real y_rep[N];
  //real ppp_value_group_mean;
  for(i in 1:N){
    // Posterior parameter distribution of the mean
    y_mean[i] = (beta1 + u[ll[i]]) / (1 + exp(-(x[i] - beta2) / beta3));
    // Posterior predictive distribution
    y_rep[i] = normal_rng(y_mean[i], sigma_temp[i]);
  }
}"

## Model fitting
n = 35; L = 5
data = list(y=Orange$circumference,
            N=n,
            L=L,
            x=Orange$age,
            ll=as.integer(Orange$Tree)
)

fit <- stan(
  model_code = stanmodel,
  data = data,
  chain = 4,
  warmup = 2000,
  iter = 3000,
  cores = 4
)

print(fit)

#############################################################################
## 2. Model diagnostic
#############################################################################
## Convergence diagnostic
## a. traceplot
traceplot(fit,pars=c("sigma[1]", 'sigma[2]', 'sigma[3]', 'sigma[4]', 'sigma[5]',
                     'tau1', 'beta1', 'beta2', 'beta3'))

## b. histogram of parameters
param = extract(fit)
par(mfrow=c(2, 3))
hist(param$sigma[, 1], breaks=40, main='sigma[1]', xlab='')
abline(v=mean(param$sigma[, 1]), col='red', lwd=2)
hist(param$sigma[, 2], breaks=40, main='sigma[2]', xlab='')
abline(v=mean(param$sigma[, 1]), col='red', lwd=2)
hist(param$sigma[, 3], breaks=40, main='sigma[3]', xlab='')
abline(v=mean(param$sigma[, 1]), col='red', lwd=2)
hist(param$sigma[, 4], breaks=40, main='sigma[4]', xlab='')
```

```
abline(v=mean(param$sigma[, 1]), col='red', lwd=2)
hist(param$sigma[, 5], breaks=40, main='sigma[5]', xlab='')
abline(v=mean(param$sigma[, 1]), col='red', lwd=2)
par(mfrow=c(2, 2))
hist(param$tau1, breaks=40, main='tau1', xlab='')
abline(v=mean(param$tau1), col='red', lwd=2)
hist(param$beta1, breaks=40, main='beta1', xlab='')
abline(v=mean(param$beta1), col='red', lwd=2)
hist(param$beta2, breaks=40, main='beta2', xlab='')
abline(v=mean(param$beta2), col='red', lwd=2)
hist(param$beta3, breaks=40, main='beta3', xlab='')
abline(v=mean(param$beta3), col='red', lwd=2)

## Visualize
y_mean <- extract(fit, "y_mean")
y_mean_cred <- apply(y_mean$y_mean, 2, quantile, c(0.05, 0.95))
y_mean_mean <- apply(y_mean$y_mean, 2, mean)

y_rep <- extract(fit, "y_rep")
y_rep_cred <- apply(y_rep$y_rep, 2, quantile, c(0.05, 0.95))
y_rep_mean <- apply(y_rep$y_rep, 2, mean)  ## y replicated

df <- data.frame(
  age = Orange$age,
  circumference = Orange$circumference,
  group = as.factor(Orange$Tree),
  lower = y_mean_cred[1,], # Assuming this is the lower bound of the CI
  upper = y_mean_cred[2,],  # Assuming this is the upper bound of the CI
  mean_prediction = y_mean_mean
)

ggplot(df, aes(x=age, y=circumference, group=group, color=group)) +
  geom_ribbon(aes(ymin=lower, ymax=upper, fill=group), alpha=0.2) +
  geom_point(size=2) +
  geom_line(aes(y=mean_prediction), linetype="dashed", color="black") +
  geom_line() +
  scale_color_manual(values = colors) +
  scale_fill_manual(values = colors) +
  labs(x='age',
       y='circumference',
       color='Tree',
       fill='Tree',
       title='Nonlinear Mixed model') +
  theme_minimal()

#########################################################################
## 3. Posterior predictive check
#########################################################################
## Posterior predictive check
par(mfrow=c(1,1))
res.obs=sqrt(colMeans((Orange$circumference - t(y_mean$y_mean))^2))
res.rep=sqrt(rowMeans((y_rep$y_rep-y_mean$y_mean)^2))          # just same as the posterior dist of
 sigma; hist(1/sqrt(param$tau))
plot(res.obs,res.rep); abline(a=0,b=1,col="red")

chisq.obs=colMeans((Orange$circumference - t(y_mean$y_mean))^2/t(y_mean$y_mean))
chisq.rep=rowMeans((y_rep$y_rep-y_mean$y_mean)^2/y_mean$y_mean)
```

```
plot(chisq.obs,chisq.rep);  abline(a=0,b=1,col="red")

## Posterior predictive p-value
# Group matrix
group <- matrix(0, n_mcmc, 35)
for (i in 1:n_mcmc){
  group[i, ] <- Orange$Tree
}
# Define residual test quantity
group_diff_ss_rep <- rep(0, n_mcmc)
group_diff_ss_obs <- rep(0, n_mcmc)
for(k in 1:n_mcmc){
  # within group variability of estimated y and replicated y
  group_sd_mean <- rep(0, 5)
  group_sd_rep <- rep(0, 5)
  group_sd_obs <- rep(0, 5)
  for(i in 1:5){
    group_sd_mean[i] <- sd(y_mean$y_mean[k, ][group[k, ] == i])
    group_sd_rep[i] <- sd(y_rep$y_rep[k, ][group[k, ] == i])
    group_sd_obs[i] <- sd(Orange$circumference[group[k, ] == i])
  }
  # sum of squared within group variability
  group_diff_rep <- group_sd_rep - group_sd_mean
  group_diff_obs <- group_sd_obs - group_sd_mean
  group_diff_ss_rep[k] <- (t(group_diff_rep) %*% group_diff_rep)/5
  group_diff_ss_obs[k] <- (t(group_diff_obs) %*% group_diff_obs)/5
}

# predictive posterior p value
mean(group_diff_ss_rep > group_diff_ss_obs)
```

Q2 (In Python)

In [1]:

```
### library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings

from scipy.stats import norm, multivariate_normal, invgamma, poisson
from numpy.linalg import inv, det, cholesky # linalg.pinv: Compute the (Moore-Penr
from numpy import matmul

from patsy import dmatrix  # For bspline `bs'

warnings.filterwarnings('ignore')
warnings.simplefilter(action='ignore', category=FutureWarning)
plt.rcParams['figure.facecolor'] = 'white'
np.set_printoptions(suppress=True)
```

Q2.

Consider a nonparametric regression model

$$y_i = sin^3(2\pi x_i^3) + \epsilon_i, \quad \epsilon_i \sim \mathrm{N}(0, 0.1^2), \quad i = 1, \dots, n.$$

Let $x_i = (2i - 1)/1000, i = 1, \dots, n$ with $n = 500$.

(a)

We approximate the target function using the cubic spline with L interior uniform knots such that there are $H = L + 4$ basis terms. Fro $H = 15$, generate design matries using the truncated power basis, polynomial radial basis, and B-sline basis (`bs` function with and intercept). Using software, verify that the column spaces of the three design matrices are identical.

In [53]:

```
#####################################################################
## Generate X and y
#####################################################################
def generate_simulated_data(n, sigma2):
    X = np.array([(2 * i - 1)/1000 for i in range(1, n+1)])
    mu_y = (np.sin(2 * np.pi * (X**3)))**3
    y = np.random.normal(mu_y, np.sqrt(sigma2))

    return X, y

def true_function(x):
    return (np.sin(2 * np.pi * (x**3)))**3

#####################################################################
## Generate uniform knots
#####################################################################
def generate_unif_knots(X, L):
    knots = np.linspace(0, 1, L+2)[1:L+1]

    return knots

#####################################################################
## Basis function 1
# 1. Truncated power basis
# 2. Polynomial radial basis
# 3. B-Spline basis
#####################################################################
def truncated_power_basis(x, k, knot):
    """ Truncated power basis for univariate x with one knot """
    return max(0, x - knot)**k

def polynomial_radial_basis(x, k, knot):
    """ Polynomial radial basis for univariate x with one knot """
    return np.abs(x - knot)**k

def evalutation(x, k, knots, function):
    """ Array of basis funtion evaluated in univarate x"""
    L = len(knots)
    basis = np.zeros(L + k + 1)

    # Start with the terms associated with the knots
    basis[:L] = np.array([function(x, k, knots[i]) for i in range(L)])

    # Append polynomial terms
    for i in range(k+1):
        basis[L+i] = x**i

    return basis

def bspline_basis(X, k, knots):
    """ Projection matrix using B-spline basis with X """
    bspline_proj = dmatrix("bs(X, knots=knots, degree=k, include_intercept=True) -
                           {"X": X, "knots": knots, "k": k})
    return np.asarray(bspline_proj)

#####################################################################
## Fit the model
# Generate projection matrix
# Prediction
```

```
#################################################################
from numpy.linalg import pinv
def Generate_proj_matrix(matrix):
    return (matrix @ inv(matrix.T @ matrix)) @ matrix.T

def pred(y, design_matrix):
    pred = design_matrix @ ((pinv(design_matrix.T @ design_matrix) @ design_matrix
    return pred
```

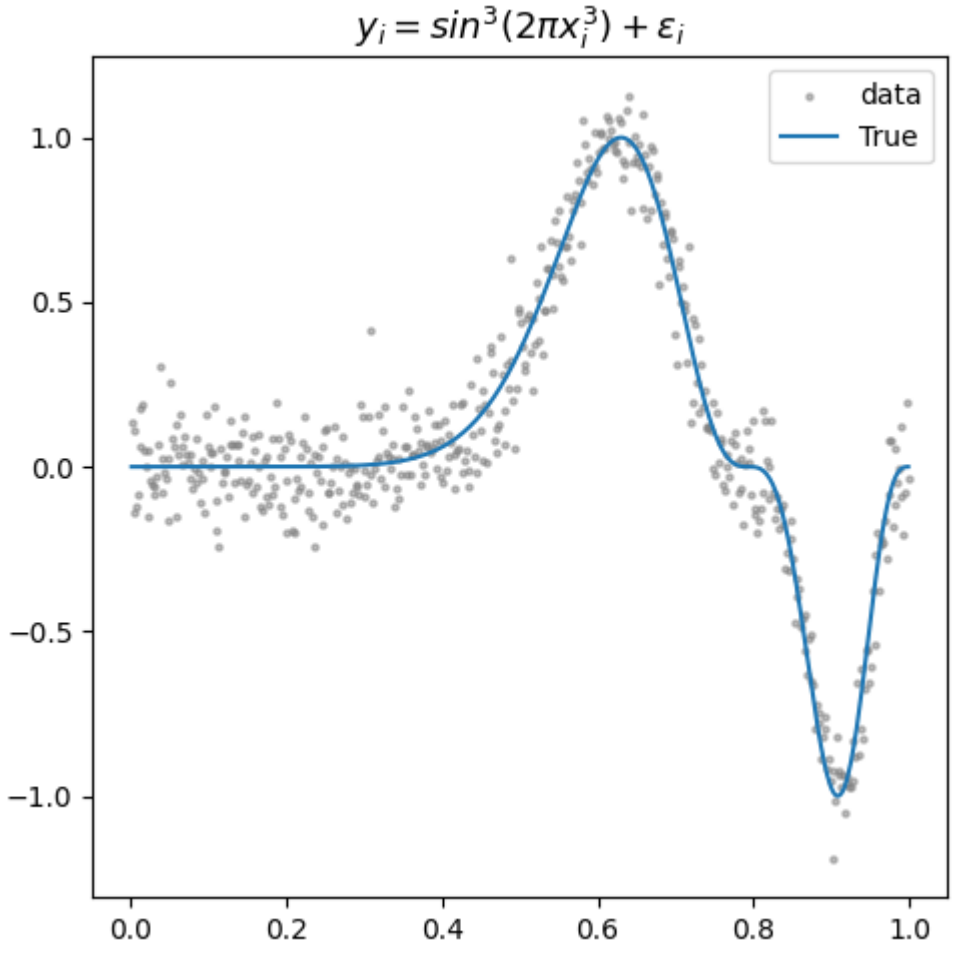
Generate simulated data, X and y

In [54]:

```
n = 500; sigma2 = 0.1**2
X, y = generate_simulated_data(n, sigma2)

plt.figure(figsize=(5, 5))
plt.scatter(X, y, label='data', s=5, alpha=0.5, color='gray')
plt.plot(X, true_function(X), label='True')

plt.legend()
plt.title(f'$y_i = sin^3(2 \\pi x_i^3) + \\epsilon_i$', fontsize=13)
plt.tight_layout()
plt.savefig('Q2_b.png', dpi=125)
plt.show()
```



Generate interior uniform knots

In [5]:

```
# Basic information
k = 3   # degree, cubic spline, order K+1
H = 15  # number of basis
L = H - (k + 1)  # num of interior knots (11)

# Generate uniform knots
knots = generate_unif_knots(X, L)
knots
```

Out[5]:

```
array([0.08333333, 0.16666667, 0.25      , 0.33333333, 0.41666667,
       0.5       , 0.58333333, 0.66666667, 0.75      , 0.83333333,
       0.91666667])
```

Design matrix

In [6]:

```
# 1. Using truncated power basis
W_truncated = np.zeros((n, H))
for i in range(n):
    W_truncated[i] = evalutation(X[i], k, knots, truncated_power_basis)

# 2. Using polynomial radial basis
W_polynomial = np.zeros((n, H))
for i in range(n):
    W_polynomial[i] = evalutation(X[i], k, knots, polynomial_radial_basis)

# 3. Using B-spline basis
W_bspline = bspline_basis(X, k, knots)
```

Do they span same column spaces?

In [7]:

```
proj_matrix_truncated = Generate_proj_matrix(W_truncated)
proj_matrix_polynomial = Generate_proj_matrix(W_polynomial)
proj_matrix_bspline = Generate_proj_matrix(W_bspline)

# Check whether the projection matrices are the same in rounding 4 digits.
print(f'Are truncated projection matrix and polynomial projected matrix same?')
print(np.allclose(proj_matrix_truncated, proj_matrix_polynomial, rtol=0.01))
print(f'Are truncated projection matrix and bspline projected matrix same?')
print(np.allclose(proj_matrix_truncated, proj_matrix_bspline, rtol=0.01))
```

```
Are truncated projection matrix and polynomial projected matrix same?
True
Are truncated projection matrix and bspline projected matrix same?
True
```

Linear regression fit

In [8]:

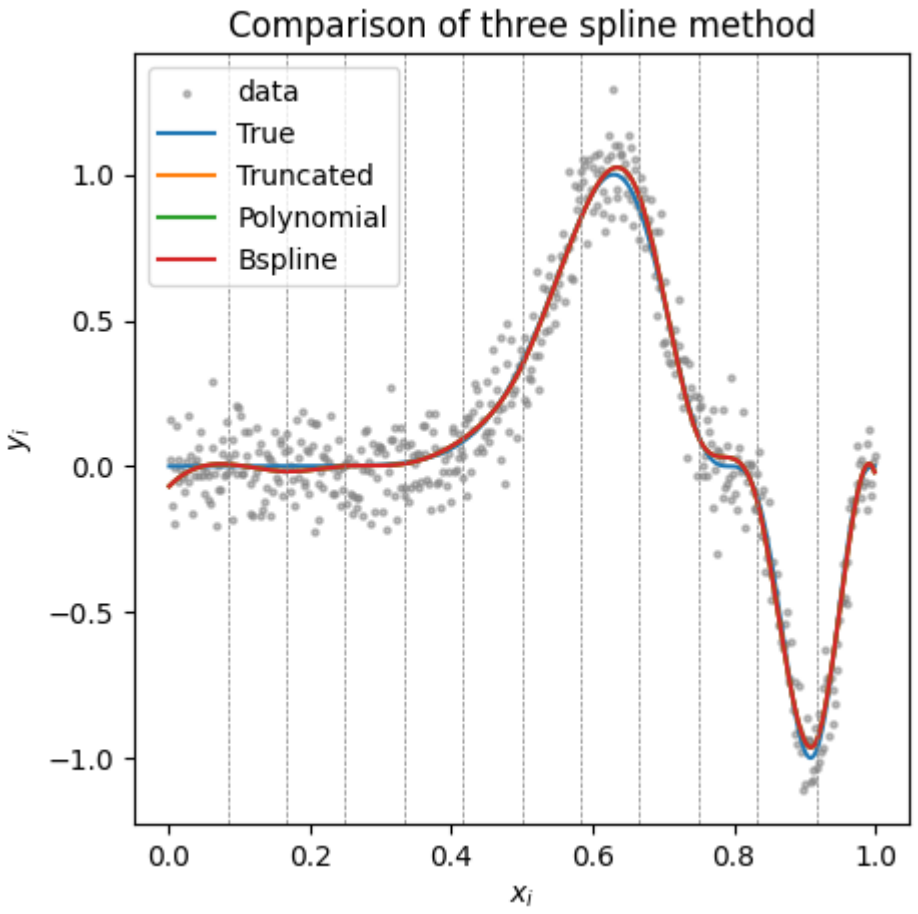
```
pred_truncated = pred(y, W_truncated)
pred_polynomial = pred(y, W_polynomial)
pred_bspline = pred(y, W_bspline)
```

In [52]:

```
plt.figure(figsize=(5, 5))

plt.scatter(X, y, label='data', s=5, alpha=0.5, color='gray')
plt.plot(X, true_function(X), label='True')
plt.plot(X, pred_truncated, label='Truncated')
plt.plot(X, pred_polynomial, label='Polynomial')
plt.plot(X, pred_bspline, label='Bspline')
[plt.axvline(knot, linestyle='dashed', linewidth=0.5, color='gray') for knot in kn
plt.axvline(knots[-1], linestyle='dashed', linewidth=0.5, color='gray')

plt.xlabel(f'$x_i$')
plt.ylabel(f'$y_i$')
plt.title(f'Comparison of three spline method')
plt.legend()
plt.savefig('Q2_b_1.png', dpi=125)
plt.tight_layout
plt.show()
```



(b)

Rather than fixing H, we now explore a model-averaged estimate of the target function. Using the above model and x_i with n=500, generate y_i, $i = 1, \dots, n$. With the design matrix generated by the B-spline, put the g-prior on the coefficients β_H with $g = n$, the Jeffreys prior on σ^2, and a Poisson prior Pois(1) on L. For $x_0 = j/1000, j = 1, \dots, 999$, obtain the model average pointwise posteriors for μ_0. Draw the posterior mean and the 95% credible interval for every x_0, which portrays the posterior mean curve and the 95% credible band on [0, 1]

likelihood

$p(y|\beta_H, \sigma^2, H) \sim N_H\left(W_H\beta_H, \sigma^2 I_n\right)$

Prior

1. $p(\sigma^2) \propto (\sigma^2)^{-1}$
2. $p(H) = p(L) \sim Pois(1)$
3. $\beta_H|\sigma^2, H \sim N_H\left(0, g\sigma^2(W_H^T W_H)^{-1}\right)$ (g-prior)

Posterior

1. $p(\beta_H|\sigma^2, H, y) \sim N_H\left(\frac{g}{1+g}(W_H^T W_H)^{-1}W_H^T y, \frac{g\sigma^2}{1+g}(W_H^T W_H)^{-1}\right)$
2. $p(\sigma^2|H, y) \sim Inv-Gamma\left(\frac{n+1}{2}, \frac{1}{2}y^T(I_n - \frac{g}{g+1}W_H(W_h^T W_H)^{-1}W_H^T)y\right)$
3. $p(H|y) \propto p(H)\{\frac{1}{2}y^T(I_n - \frac{g}{g+1}W_H(W_h^T W_H)^{-1}W_H^T)y\}^{-\frac{n+1}{2}}$

Step 1) Calculate normalize marginal posterior $p(H|y)$

- To obtain exact marginal posterior probability of H, let's assume that H can have a maximum value of 30. Then the unnormalized posterior can be calculated of all cases of H, and the normalizing constant can be computed to obtain the normalized posterior of H.

In [10]:

```
def unnormalized_posterior_H(X, L):
    ## Define Design matrix W_H for model H with interior knots L
    k = 3
    W = bspline_basis(X, k, generate_unif_knots(X, L))
    n, H = W.shape
    g = n  # for g-prior

    # log prior
    log_prior = poisson.logpmf(L, mu=1)
    # log marginal likelihood
    epsilon = 1e-4
    factor1 = (W @ inv(W.T @ W)) @ W.T
    log_marginal_likelihood = (- (n + 1) / 2) * \
        np.log(((y.T @ (np.identity(n) - (g / (g + 1)) * factor1)) @ y) / 2)

    # unnormalized posterior
    posterior = np.exp(log_prior + log_marginal_likelihood)
    return posterior

def posterior_H(X, max_L):
    constant = 0
    unnormalized = np.zeros(max_L)
    for l in range(max_L):
        unnormalized[l] = unnormalized_posterior_H(X, l+1)
        constant += unnormalized[l]

    # Normalize posterior with constant
    posterior_H = unnormalized / constant

    return posterior_H
```

In [11]:

```
max_L = 30
posterior_H = posterior_H(X, max_L)
posterior_H
```

Out[11]:

```
array([0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.09947506, 0.89921775, 0.00115973, 0.00008728, 0.00000481,
       0.00000155, 0.00002809, 0.00002439, 0.00000111, 0.00000022,
       0.00000001, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ])
```

Step 2. Pointwise posterior of $\mu^{(t)}(x_0)$

- Draw $H^{(t)}$ from $p(H|y)$
- Draw σ^2 from $p(\sigma^2|H, y)$ given $H^{(t)}$
- Draw $\beta_1^{(t)}, \dots \beta_{H^{(t)}}^{(t)}$ from $p(\beta|\sigma^2, H, y)$ given $H^{(t)}, \sigma^2$

In [12]:

```
def sampling(X, posterior_H):
    """ For one iteration """

    # 1. Sampling H from p(H|y)
    max_L = len(posterior_H)
    L_sample = np.random.choice(np.arange(1, max_L + 1), p=posterior_H)

    # 2. Generate design matrix for H
    k = 3
    W = bspline_basis(X, k, generate_unif_knots(X, L_sample))
    n, H = W.shape
    g = n  # for g-prior

    # 3. Sampling sigma^2 from p(sigma^2 | H, y)
    alpha = (n + 1) / 2
    factor1 = (W @ inv(W.T @ W)) @ W.T
    beta = ((y.T @ (np.identity(n) - (g / (g + 1)) * factor1)) @ y) / 2
    sigma2_sample = invgamma.rvs(a=alpha, scale=beta, size=1)

    # 4. Sampling beta_H from p(beta_H | sigma^2, H, y)
    mean = (inv(W.T @ W) @ W.T) @ y * (g / (1 + g))
    A = cholesky(inv(W.T @ W))
    cov = np.sqrt(g * sigma2_sample / (1 + g)) * A
    beta_samples = multivariate_normal.rvs(mean, cov)
    beta_samples

    return W, L_sample, sigma2_sample, beta_samples
```

Step 3. Model averaging

- The pointwise posterior is averaged over different models (**Bayesian model averaging**)

In [13]:

```
def model_averaging(X, X_new, n_mcmc):

    k = 3
    averaging_values = np.zeros((len(X_new), n_mcmc))
    for i in range(n_mcmc):
        # 1. sampling H^(t) and beta^t_1, ..., beta^t_H
        W, L, sigma2, betas = sampling(X, posterior_H)

        # 2. Model averaging
        W_new = bspline_basis(X_new, k, generate_unif_knots(X_new, L))
        estimated_X = W_new @ betas

        averaging_values[:, i] = estimated_X

        if i % 100 == 0:
            print(i)

    # 3. posterior mean and 95% credible interval
    posterior_mean = np.mean(averaging_values, axis=1)
    posterior_interval = np.quantile(averaging_values, [0.05, 0.95], axis=1)

    return averaging_values, posterior_mean, posterior_interval
```

In [14]:

```
n_mcmc = 1000  # number of mcmc samples
X_new = np.array([j / 1000 for j in range(1, 1000)])  # new X
averaging_values, posterior_mean, posterior_interval = model_averaging(X, X_new, n
```

```
0
100
200
300
400
500
600
700
800
900
```

In [51]:

```
sigma2 = 0.1**2
y_new= np.random.normal((np.sin(2 * np.pi * (X_new**3)))**3, np.sqrt(sigma2))

plt.figure(figsize=(5, 5))
plt.scatter(X_new, y_new, label='data', s=5, alpha=0.5, color='gray')
plt.plot(X_new, true_function(X_new), label='True')
plt.plot(X_new, posterior_mean, label='Posterior mean')
plt.fill_between(X_new, posterior_interval[0], posterior_interval[1], alpha=0.25,
plt.legend()
plt.title(f'Bayesian model averaging', fontsize=13)
plt.ylabel('y')
plt.xlabel('x')
plt.tight_layout()
plt.savefig('Q2_b_2.png', dpi=125)
plt.show()
```