Bayesian-Linear-Regression-with-Shrinkage-Priors-MCnote

Duong Trinh

2022-08-08

1 About 2022.05.05_MonteCarlo (KST)

1.1 Result - 2022.06.27

- Function to generate various regression models: `GenRegr_sep2021.m`
- Monte Carlo exercise: `MC_main_2706.m` - there are 8 DGPs x 3 pairs (n = 100, p = [50, 100, 150]):
  - DGPs (1 + 2) – Uncorrelated predictors,
  - DGPs (3 + 4) – Spatially correlated predictors (rho = 0.4),
  - DGPs (5 + 6) – Spatially correlated predictors (rho = 0.8),
  - DGP (7) – Heteroskedastic errors,
  - DGP (8) – Stochastic Volatility.
  - DGPs (1 + 3 + 5) correspond to Rsquared = 0.4; DGPs (2 + 4 + 6) correspond to Rsquared = 0.8.
- Summary:

Table 1: BetaTrue: $n = 100$, $p = 50$

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8
b1	0.245	0.6	0.245	0.6	0.245	0.6	1.5	1.5
b2	-0.245	-0.6	-0.245	-0.6	-0.245	-0.6	-1.5	-1.5
b3	0.327	0.8	0.327	0.8	0.327	0.8	2.0	2.0
b4	-0.327	-0.8	-0.327	-0.8	-0.327	-0.8	-2.0	-2.0
b5	0.408	1.0	0.408	1.0	0.408	1.0	2.5	2.5
b6	-0.408	-1.0	-0.408	-1.0	-0.408	-1.0	-2.5	-2.5

Table 2: BetaTrue: $n = 100$, $p = 100$

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8
b1	0.245	0.6	0.245	0.6	0.245	0.6	1.5	1.5
b2	-0.245	-0.6	-0.245	-0.6	-0.245	-0.6	-1.5	-1.5
b3	0.327	0.8	0.327	0.8	0.327	0.8	2.0	2.0
b4	-0.327	-0.8	-0.327	-0.8	-0.327	-0.8	-2.0	-2.0
b5	0.408	1.0	0.408	1.0	0.408	1.0	2.5	2.5
b6	-0.408	-1.0	-0.408	-1.0	-0.408	-1.0	-2.5	-2.5

Table 3: BetaTrue: $n = 100$, $p = 150$

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8
b1	0.245	0.6	0.245	0.6	0.245	0.6	1.5	1.5
b2	-0.245	-0.6	-0.245	-0.6	-0.245	-0.6	-1.5	-1.5
b3	0.327	0.8	0.327	0.8	0.327	0.8	2.0	2.0
b4	-0.327	-0.8	-0.327	-0.8	-0.327	-0.8	-2.0	-2.0
b5	0.408	1.0	0.408	1.0	0.408	1.0	2.5	2.5
b6	-0.408	-1.0	-0.408	-1.0	-0.408	-1.0	-2.5	-2.5

Table 4: var(Epsilon)

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8
n = 100, p = 50	0.994	0.994	0.994	0.994	0.994	0.994	0.978	0.073
n = 100, p = 100	0.986	0.986	0.986	0.986	0.986	0.986	0.996	0.074
n = 100, p = 150	0.996	0.996	0.996	0.996	0.996	0.996	0.995	0.073

Table 5: SNR

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8
n = 100, p = 50	0.685	4.108	0.344	2.066	0.119	0.716	27.035	353.289
n = 100, p = 100	0.690	4.142	0.347	2.084	0.120	0.722	26.764	349.395
n = 100, p = 150	0.683	4.100	0.344	2.065	0.119	0.715	26.471	352.426

Table 6: Rsquared

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8
n = 100, p = 50	0.403	0.800	0.254	0.668	0.106	0.413	0.962	0.997
n = 100, p = 100	0.405	0.801	0.256	0.670	0.107	0.415	0.961	0.997
n = 100, p = 150	0.402	0.799	0.254	0.668	0.106	0.412	0.961	0.997

- Results:
  - https://duongtrinh.shinyapps.io/kst-ana1/
  - https://duongtrinh.shinyapps.io/kst-ana2/
- Issues:
  - Inconsistent Signal to Noise ratio (or R-squared) → Change functions to be used for DGPs.
  - *SSVS-Lasso-3* and *SSVS-Horseshoe-2* perform considerably worse than other Bayesian shrinkage priors, and even worse than No shrinkage sometimes (seem to induce too much shrinkage effect):
    - *SSVS-Lasso-3*: “kappa0 = NaN” in `BayesRegr.m` so that “tau0 = 1e-10” always!
    - *SSVS-Horseshoe-2*: The condition “tau1(tau1<1e-20) = 1e-20” and "tau0 = 1e-3*tau1" is the cause...

1.2 Result - 2022.07.14

- Function to generate various regression models: `GenRegr_july2022.m`

Table 9: BetaTrue: $n = 100$, $p = 150$

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8
b1	0.245	0.6	0.245	0.6	0.245	0.6	1.5	1.5
b2	-0.245	-0.6	-0.245	-0.6	-0.245	-0.6	-1.5	-1.5
b3	0.327	0.8	0.327	0.8	0.327	0.8	2.0	2.0
b4	-0.327	-0.8	-0.327	-0.8	-0.327	-0.8	-2.0	-2.0
b5	0.408	1.0	0.408	1.0	0.408	1.0	2.5	2.5
b6	-0.408	-1.0	-0.408	-1.0	-0.408	-1.0	-2.5	-2.5

- Monte Carlo exercise: `MC_main_1007.m` - there are 10 DGPs x 3 pairs (n = 100, p = [50, 100, 150]):
  - DGPs (1 + 2) – Uncorrelated predictors,
  - DGPs (3 + 4) – Spatially correlated predictors (rho = 0.4),
  - DGPs (5 + 6) – Spatially correlated predictors (rho = 0.8),
  - DGPs (7 + 8) – Heteroskedastic errors,
  - DGPs (9 + 10) – Stochastic Volatility.
  - Odd DGPs (1 + 3 + 5 + 7 + 9) correspond to Rsquared = 0.4; Even DGPs (2 + 4 + 6 + 8 +20) correspond to Rsquared = 0.8.
- Summary:

Table 7: BetaTrue: $n = 100$, $p = 50$

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8	DGP9	DGP10
b1	0.245	0.6	0.346	0.848	0.588	1.441	0.245	0.6	0.245	0.6
b2	-0.245	-0.6	-0.346	-0.848	-0.588	-1.441	-0.245	-0.6	-0.245	-0.6
b3	0.327	0.8	0.461	1.130	0.784	1.921	0.327	0.8	0.327	0.8
b4	-0.327	-0.8	-0.461	-1.130	-0.784	-1.921	-0.327	-0.8	-0.327	-0.8
b5	0.408	1.0	0.577	1.413	0.980	2.402	0.408	1.0	0.408	1.0
b6	-0.408	-1.0	-0.577	-1.413	-0.980	-2.402	-0.408	-1.0	-0.408	-1.0

Table 8: BetaTrue: $n = 100$, $p = 100$

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8	DGP9	DGP10
b1	0.245	0.6	0.346	0.848	0.588	1.441	0.245	0.6	0.245	0.6
b2	-0.245	-0.6	-0.346	-0.848	-0.588	-1.441	-0.245	-0.6	-0.245	-0.6
b3	0.327	0.8	0.461	1.130	0.784	1.921	0.327	0.8	0.327	0.8
b4	-0.327	-0.8	-0.461	-1.130	-0.784	-1.921	-0.327	-0.8	-0.327	-0.8
b5	0.408	1.0	0.577	1.413	0.980	2.402	0.408	1.0	0.408	1.0
b6	-0.408	-1.0	-0.577	-1.413	-0.980	-2.402	-0.408	-1.0	-0.408	-1.0

Table 10: var(Epsilon)

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8	DGP9	DGP10
n = 100, p = 50	0.994	0.994	0.994	0.994	0.994	0.994	0.978	0.978	0.985	0.985
n = 100, p = 100	0.986	0.986	0.986	0.986	0.986	0.986	0.996	0.996	1.003	1.003
n = 100, p = 150	0.996	0.996	0.996	0.996	0.996	0.996	0.995	0.995	0.984	0.984

Table 11: SNR

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8	DGP9	DGP10
n = 100, p = 50	0.685	4.108	0.688	4.125	0.688	4.128	0.721	4.326	0.692	4.153
n = 100, p = 100	0.690	4.142	0.694	4.161	0.694	4.166	0.714	4.282	0.679	4.076
n = 100, p = 150	0.683	4.100	0.687	4.122	0.687	4.124	0.706	4.235	0.690	4.140

Table 12: Rsquared

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8	DGP9	DGP10
n = 100, p = 50	0.403	0.800	0.404	0.800	0.403	0.799	0.412	0.804	0.405	0.801
n = 100, p = 100	0.405	0.801	0.406	0.801	0.405	0.801	0.408	0.801	0.401	0.798
n = 100, p = 150	0.402	0.799	0.403	0.799	0.403	0.799	0.407	0.800	0.405	0.801

- Results:
  - https://duongtrinh.shinyapps.io/KST-ana5/
  - https://duongtrinh.shinyapps.io/KST-ana6/
- Issues: While our goal is inference in coefficients, true β varies across DGPs.

1.3 Result - 2022.07.27

- Function to generate various regression models: `GenRegr_27072022.m`
- Monte Carlo exercise: `MC_main_1007.m` - there are 10 DGPs x 3 pairs (n = 100, p = [50, 100, 150]):
  - DGPs (1 + 2) – Uncorrelated predictors,
  - DGPs (3 + 4) – Spatially correlated predictors (rho = 0.4),
  - DGPs (5 + 6) – Spatially correlated predictors (rho = 0.8),
  - DGPs (7 + 8) – Heteroskedastic errors,
  - DGPs (9 + 10) – Stochastic Volatility.
  - Odd DGPs (1 + 3 + 5 + 7 + 9) correspond to Rsquared = 0.4; Even DGPs (2 + 4 + 6 + 8 +20) correspond to Rsquared = 0.8.
- Summary:

Table 13: BetaTrue: $n = 100$, $p = 50$

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8	DGP9	DGP10
b1	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
b2	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5
b3	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
b4	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0
b5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
b6	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5

Table 14: BetaTrue: $n = 100$, $p = 100$

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8	DGP9	DGP10
b1	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
b2	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5
b3	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
b4	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0
b5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
b6	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5

Table 15: BetaTrue: $n = 100$, $p = 150$

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8	DGP9	DGP10
b1	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
b2	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5	-1.5
b3	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
b4	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0
b5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
b6	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5	-2.5

Table 16: var(Epsilon)

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8	DGP9	DGP10
n = 100, p = 50	37.286	6.214	18.678	3.113	6.465	1.077	36.661	6.110	36.951	6.159
n = 100, p = 100	36.979	6.163	18.524	3.087	6.411	1.069	37.353	6.225	37.612	6.269
n = 100, p = 150	37.342	6.224	18.706	3.118	6.474	1.079	37.326	6.221	36.889	6.148

Table 17: SNR

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8	DGP9	DGP10
n = 100, p = 50	0.685	4.108	0.688	4.125	0.688	4.128	0.721	4.326	0.692	4.153
n = 100, p = 100	0.690	4.142	0.694	4.161	0.694	4.166	0.714	4.282	0.679	4.076
n = 100, p = 150	0.683	4.100	0.687	4.122	0.687	4.124	0.706	4.235	0.690	4.140

Table 18: Rsquared

	DGP1	DGP2	DGP3	DGP4	DGP5	DGP6	DGP7	DGP8	DGP9	DGP10
n = 100, p = 50	0.403	0.800	0.404	0.800	0.403	0.799	0.412	0.804	0.405	0.801
n = 100, p = 100	0.405	0.801	0.406	0.801	0.405	0.801	0.408	0.801	0.401	0.798
n = 100, p = 150	0.402	0.799	0.403	0.799	0.403	0.799	0.407	0.800	0.405	0.801

- Results:
  - https://duongtrinh.shinyapps.io/KST-ana7/

1.4 More thoughts

About the Signal-to-Noise Ratio (SNR):

- Formula 1:

$$\frac{R^2_{pop}}{1 - R^2_{pop}} = SNR = \frac{\left\|\Sigma^{1/2}\beta\right\|^2}{\sigma^2} = \frac{\beta'\Sigma\beta}{\sigma^2}$$

- Formula 2:

$$SNR = \frac{var(X\beta)}{\sigma^2}$$

- Formula 3:

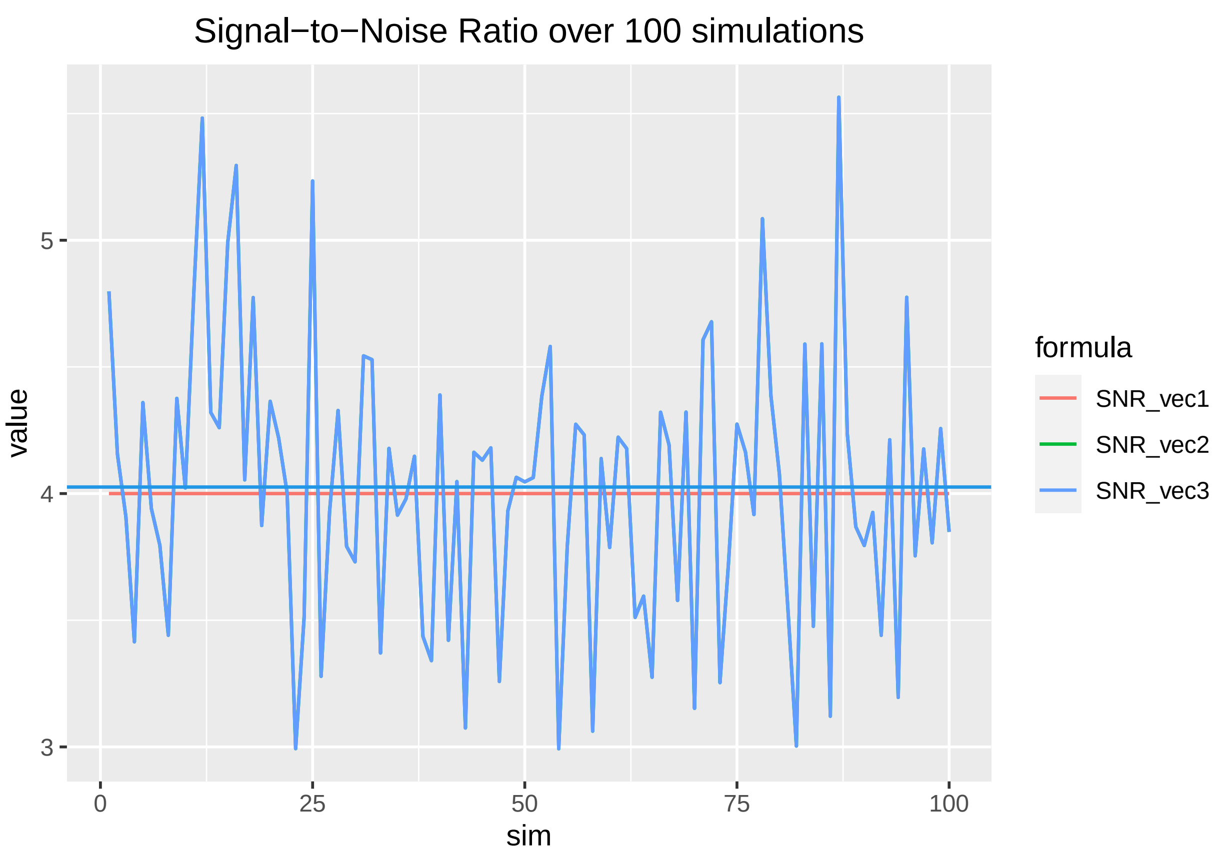
$$SNR = \frac{\beta' X' X\beta}{(n-1)\sigma^2}$$

```
# library(pracma) # for a (non-symmetric) Toeplitz matrix
GenRegr <- function(n,p,options) {
  # Generate predictors x
  if (options.corr == 0) {# Uncorrelated predictors
    C <-  diag(rep(1,p))
    x <-  matrix(rnorm(n*p),n,p)%*%chol(C)
  }
  else if (options.corr == 1) {# Spatially ncorrelated predictors
    C <-  toeplitz(options.rho^(0:(p-1)))
    x <-  matrix(rnorm(n*p),n,p)%*%chol(C)
  }
  else {
    print('Wrong choice of options.corr')
  }

  x <- data.matrix(sapply(data.frame(x), function(x) {(x-mean(x))/sd(x)})) # Standardize x

  # Generate coefficients
  beta <- rep(0,p)
  beta[1:6] <- c(1.5,-1.5,2,-2,2.5,-2.5)

  if (options.corr == 0) {
    signal_y <- sum(beta^2)
  }
  else if (options.corr == 1) {
    signal_y <- sum((chol(C)%*%beta)^2)
  }

  c <- signal_y*((1-options.R2)/options.R2) # mean(sigmasq) is c to obtain desirable options.R2 (or SNR

  # Generate epsilon
```

```r
  if (options.epsilon == 0) { # iid error
    sigmasq <- c
  }
  else if (options.epsilon == 1) {
    temp = (x%*%beta)
    sigmasq = c*temp/mean(temp)
  }

  epsilon = sqrt(sigmasq) * rnorm(n)

  # Generate y
  y = x%*%beta + epsilon

  return(list(y = y, x = x, beta = beta, C = C, sigmasq = sigmasq))
}
```

```r
set.seed(2907)
n = 100
p = 50
options.corr = 1
options.R2 = 0.8 # SNR = 4
options.epsilon = 0
options.rho = 0.4

df <- GenRegr(n, p, options)

y <- df$y
X <- df$x
beta_true <-  df$beta
C <- df$C
sigmasq <- df$sigmasq

# library(GGally)
# ggcorr(X, palette = "RdBu", label = FALSE)
#
# library(ggcorrplot)
# corr <- round(cor(X), 1)
# ggcorrplot(corr, hc.order = TRUE, outline.col = "white")
# ggcorrplot(C, hc.order = TRUE, outline.col = "white")


Nsim = 100
SNR_vec1 <- rep(NA,Nsim)
SNR_vec2 <- rep(NA,Nsim)
SNR_vec3 <- rep(NA,Nsim)

for (sim in 1 : Nsim)
{
  df <- GenRegr(n, p, options)
  set.seed(sim)
  y <- df$y
  X <- df$x
  beta_true <-  df$beta
```

```
  C <- df$C
  SNR_vec1[sim] <-  t(beta_true)%*%C%*%beta_true/sigmasq #sum((chol(C)%*%beta_true)^2)
  SNR_vec2[sim] <- var(X%*%beta_true)/sigmasq
  SNR_vec3[sim] <- t(beta_true)%*%t(X)%*%X%*%beta_true/(n-1)/sigmasq
}

# SNR_vec1
# SNR_vec2
# SNR_vec3
# SNR_vec2 == SNR_vec3
# SNR_vec1
# mean(SNR_vec2)

library(tidyverse)
df <- data.frame(sim = 1:Nsim,SNR_vec1,SNR_vec2,SNR_vec3)
df_long <- gather(df, formu, value, -c("sim"))

ggplot(df_long, aes(x = sim, y = value, group = formu)) +
  geom_line(aes(color = formu), size = 0.6) +
  geom_hline(yintercept = mean(SNR_vec2), col = 4, size = 0.6) +
  ggtitle("Signal-to-Noise Ratio over 100 simulations") +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(color = "formula")
```



Conclusion: Formula 2 and 3 are equivalent.

Theorem

If β is a vector and X is a random vector with mean μ and variance Σ then

$$\mathbb{E}(\beta^T X) = \beta^T \mu \quad \text{and} \quad \mathbb{V}(\beta^T X) = \beta^T \Sigma \beta$$

If B is a matrix then

$$\mathbb{E}(BX) = B\mu \quad \text{and} \quad \mathbb{V}(BX) = B\Sigma^T B$$

Choice of priors (and hyper-parameters)

- https://duongtrinh.shinyapps.io/KST-priors/