Gaussian Data Example

Eric Dunipace

6/18/2019

Setup

Predictors

We can measure variable importance in a variety of settings. Our first example is in a setting with Normally distributed data. We generate $n = 1024$ predictor variables, X, from a Multivariate Normal,

$$X_i \sim N(0, \Sigma),$$

with $\Sigma_{i,i} = 1$ for all i from 1 to $p = 30$. The first 5 dimensions of X are correlated, dimensions 6-10 are correlated, dimensions 10-20 are correlated, and dimensions 21 to 30 are correlated with correlations 0.5. Thus, Σ is block diagonal.

Parameters

We generate parameters as follows

$$\beta_{0:5} \sim \text{Unif}(1, 2),$$
$$\beta_{6:10} \sim \text{Unif}(-2, -1),$$
$$\beta_{11:20} \sim \text{Unif}(0, 0.5),$$
$$\beta_{21:30} = 0,$$

and

$$\sigma^2 = 1.$$

Outcome

We sample the outcome as

$$Y_i \sim N(\beta_0 + X_i^\top \beta_{1:20}, \sigma^2)$$

```
#### Load packages ####
require(SparsePosterior)
require(ggplot2)
require(CoarsePosteriorSummary)
require(rstan)
require(ggsci)
require(doParallel)
require(ggridges)
require(data.table)

rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores()-1)

group.names <- c("Selection","Loc./Scale", "Projection")

#### generate data ####
```

```
set.seed(9507123)
n <- 1024
p <- 31
nsamp <- 1000
nlambda <- 100
lambda.min.ratio <- 1e-10
gamma <- 1
pseudo.observations <- 0

target <- get_normal_linear_model()
param <- target$rparam()
p_star <- length(param$theta)

target$X$corr <- 0.5
X <- target$X$rX(n, target$X$corr, p)
Y <- target$rdata(n,X[,1:p_star], param$theta, param$sigma2)

x <- target$X$rX(1, target$X$corr, p)

true_mu_full <- X[,1:p_star, drop=FALSE] %*% c(param$theta)
true_mu <- x[,1:p_star,drop=FALSE] %*% c(param$theta)
```

Posterior Estimation

We put conjugate priors on our parameters,

$$\beta_{0:p} \sim N(0, 1)$$

and

$$\sigma^2 \sim \text{Inv-Gamma}(10, 10),$$

which facillitates quick estimation via known posterior distributions.

```
hyperparameters <- list(mu = NULL, sigma = NULL,
                        alpha = NULL, beta = NULL,
                       Lambda = NULL)
hyperparameters$mu <- rep(0,p)
hyperparameters$sigma <- diag(1,p,p)
hyperparameters$alpha <- 10
hyperparameters$beta <- 10
hyperparameters$Lambda <- solve(hyperparameters$sigma)

posterior <- target$rpost(nsamp, X, Y, hyperparameters = hyperparameters, method="conjugate")


cond_mu <- x %*% posterior$theta
cond_mu_full <- X %*% posterior$theta
```

We then utilize our 1) selection variable, 2) location/scale, and 3) projection methods to find coarse posteriors close to our full one. Note: this code may take a bit to run.

```
penalty.factor <- rep(1, p)
force <- NULL

theta_selection <- W2L1(X=X, Y = cond_mu_full,
```

```
                    theta=posterior$theta, penalty="selection.lasso",
                    nlambda = nlambda, lambda.min.ratio = lambda.min.ratio,
                    infimum.maxit=1e4, maxit = 1e3, gamma = gamma,
                    pseudo_observations = 0, display.progress = TRUE,
                    penalty.factor = penalty.factor,
                    method="selection.variable")

theta_proj <- W2L1(X=X, Y=cond_mu_full,
                    theta=posterior$theta, penalty="mcp",
                    nlambda = nlambda, lambda.min.ratio = lambda.min.ratio,
                    infimum.maxit=1, maxit = 1e3, gamma = gamma,
                    pseudo_observations = pseudo.observations,
                    display.progress = TRUE,
                    penalty.factor = penalty.factor, method="projection")

theta_SA <- WPSA(X=X, Y=cond_mu_full,
                    theta=posterior$theta, force = force,
                    p = 2, model.size = c(1:p),
                    iter=10, temps = 10,
                    pseudo.obs = 0,
                    proposal = proposal.fun,
                    options = list(method = c("selection.variable"),
                                   energy.distribution = "boltzman",
                                   cooling.schedule = "exponential"),
                    display.progress = TRUE)

theta_SW <- WPSW(X=X, Y=cond_mu_full,
                    theta=posterior$theta, force = force, p = 2,
                    direction = "backward",
                    method="selection.variable",
                    display.progress = TRUE)
```

Distance Plots

It is useful to look at plots to see how our coarsened posteriors are doing. We can look at how close the coarsened posterior parameters are to the full posterior parameters in terms of 2-Wasserstein distance

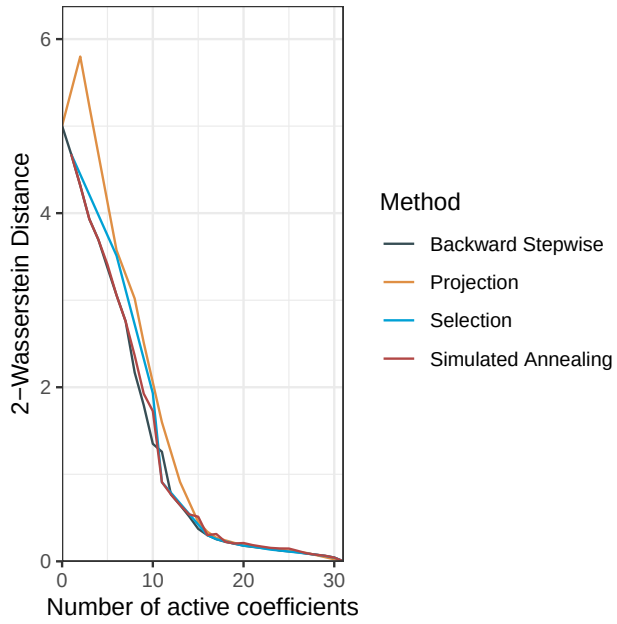


Figure 1: 2-Wasserstein distance between the full posterior for the regression coefficients and our coarsened version.

and how far away the coarsened posteriors are in terms of mean-squared error (MSE) from the true data generating parameters.

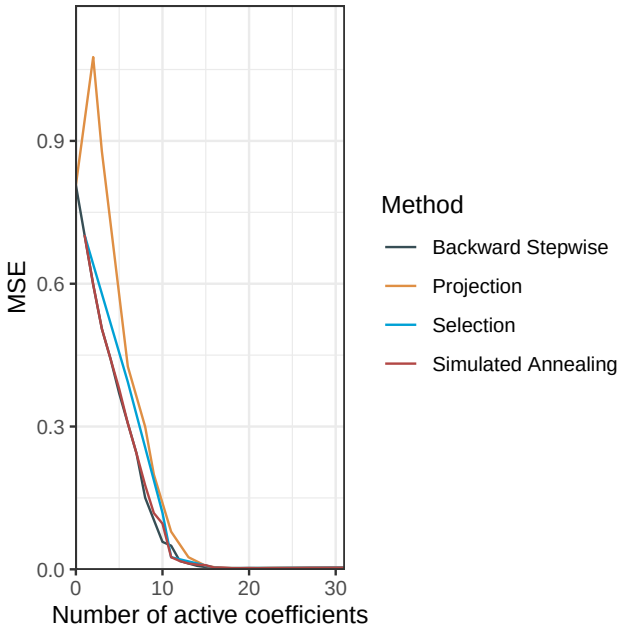


Figure 2: MSE between the full posterior for the regression coefficients and the true data generating parameters.

Moreover, we can look at the differences the posterior predictive mean and the full posterior predictive mean, as well as the MSE between the coarse posterior and true mean.

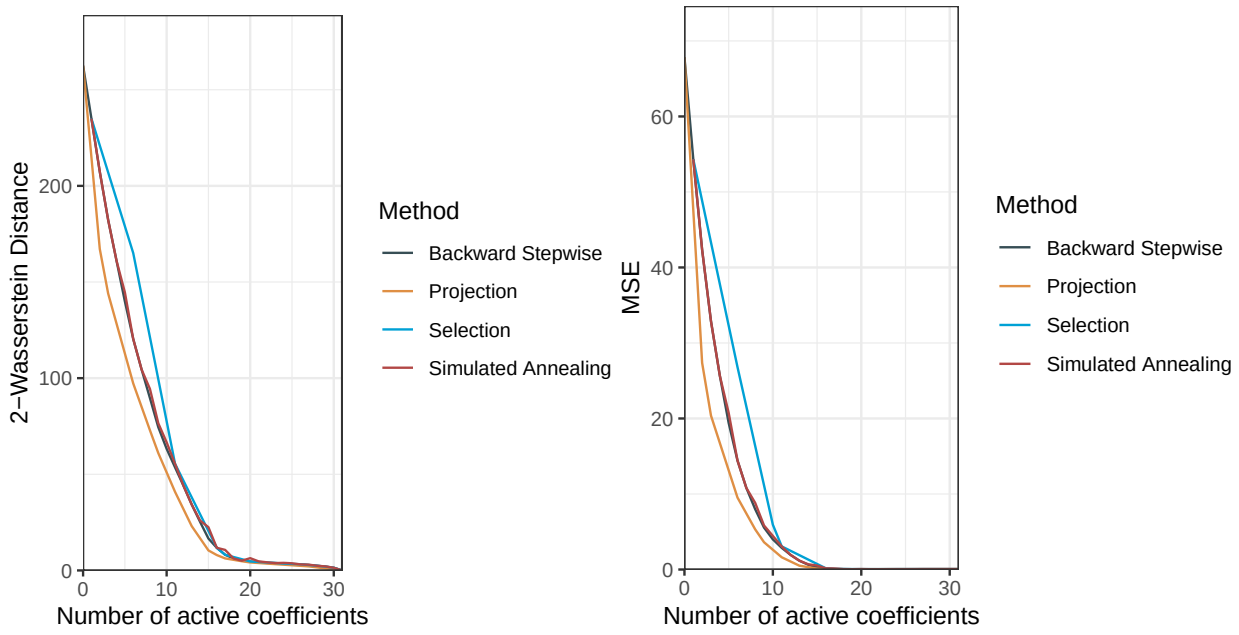


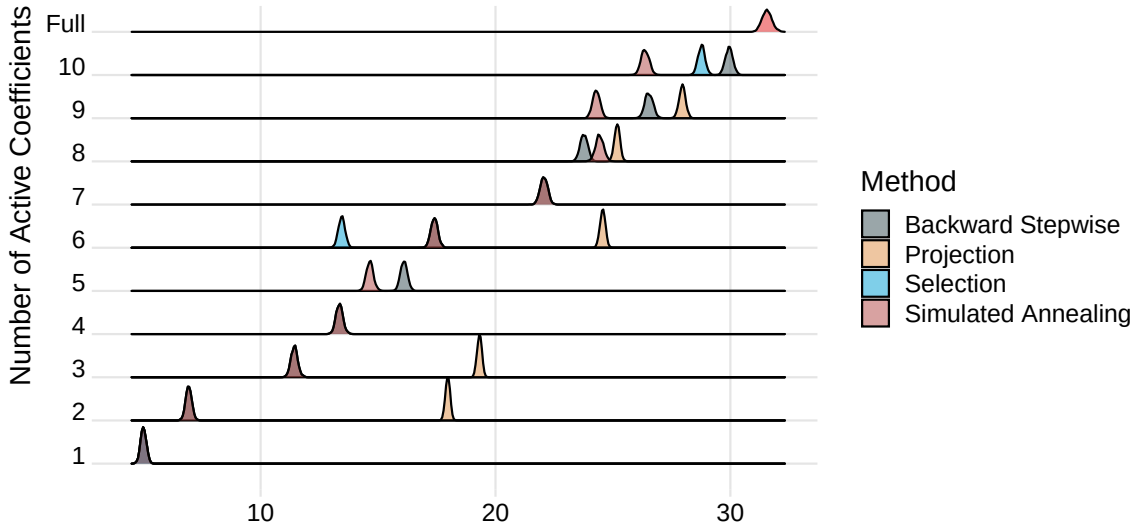
Figure 3: 2-Wasserstein distance between coarse and full posterior predictive means and MSE between posterior predictive means and the true means (calculated from the true parameters)

We can see as more covariates are active, the coarse distribution gets close to the full posterior and the true mean.

Ridgeline Plots

We may be interested to see how well the coarasened posterior is approximating an individuals mean, relative to the full model, for a given model size. To examine this relationship, we can look at ridgeline plots of the posterior predictive means from the coarsened posteriors. Take, for example, individual $i = 512$.

```
idx <- 512
ridge512 <- ridgePlot(models, index=idx, maxCoef = 10,
                      scale =1 , alpha =0.5, full = cond_mu_full)
```



We can see that as we add more coefficients up to our desired size, 10, the models perform better. For model size 10, there is no prediction for the Projection method, but it looks like the backward stepwise method performs best. If we really wanted to include the Projection method, we could try model size 9 or try a larger model size where it was present.

Of course, we may be interested in looking at how good or how bad the predictions from the coarsened posterior can be for the entire sample, and not just one person chosen at random. Towards this end, we have implemented a function that for a given model size, will calculate the average p-Wasserstein distance between the coarsened predictions and the full posterior, and rank all individual observations by this distance. Then this ranking function will give the observations at user provided quantiles.

```
ranks <- ranking(models, cond_mu_full, p = 2,
                 maxCoef = 10, quantiles = c(0,0.3, 0.67, 1))
```

Note, when we provide this function more than one model, it will average all of the calculated distances for a selected model size. S

Next, we can plot ridgeline plots of the predictions from 1 coefficient up to the given model size. This allows us to see if smaller models perform equally well.

```
ridgePlots <- ridgePlot(models, index=ranks$index,
                        maxCoef = ranks$ncoef, scale =1 , alpha =0.5, full = cond_mu_full)
print(ridgePlots)
```

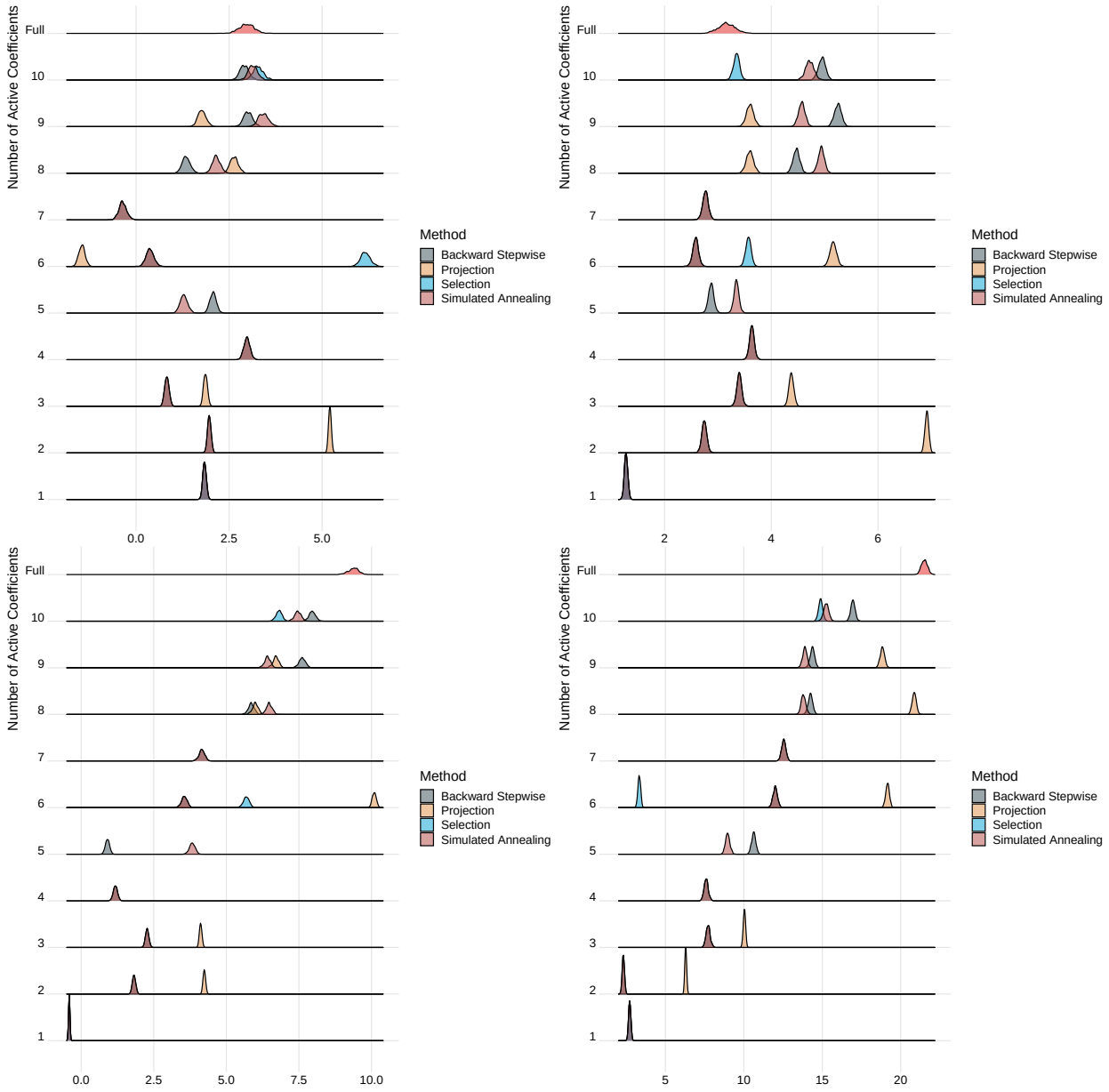


Figure 4: Ridgeline plots for 1 to 10 coefficients. The top left plot is the observation whose coarsened posterior prediction is closest to the true posterior predictive distribution in terms of 2-Wasserstein distance (0^{th} percentile in terms of ranks), and the bottom right is the observations whose is furthest from the true posterior predictive distribution (100^{th} percentile in terms of ranks). Top row, left to right: 0^{th} and 33^{rd} percentiles. Bottom row, left to right: 67^{th} and 100^{th} percentiles.

If we had wanted a smaller model size, say 6, that would be possible as well

```
ranks2 <- ranking(models, cond_mu_full, p = 2,
                  maxCoef = 6, quantiles = c(0,0.3, 0.67, 1))
ridgePlots2 <- ridgePlot(models, index=ranks2$index,
                         maxCoef = ranks2$ncoef, scale =1 , alpha =0.5, full = cond_mu_full)
print(ridgePlots2)
```

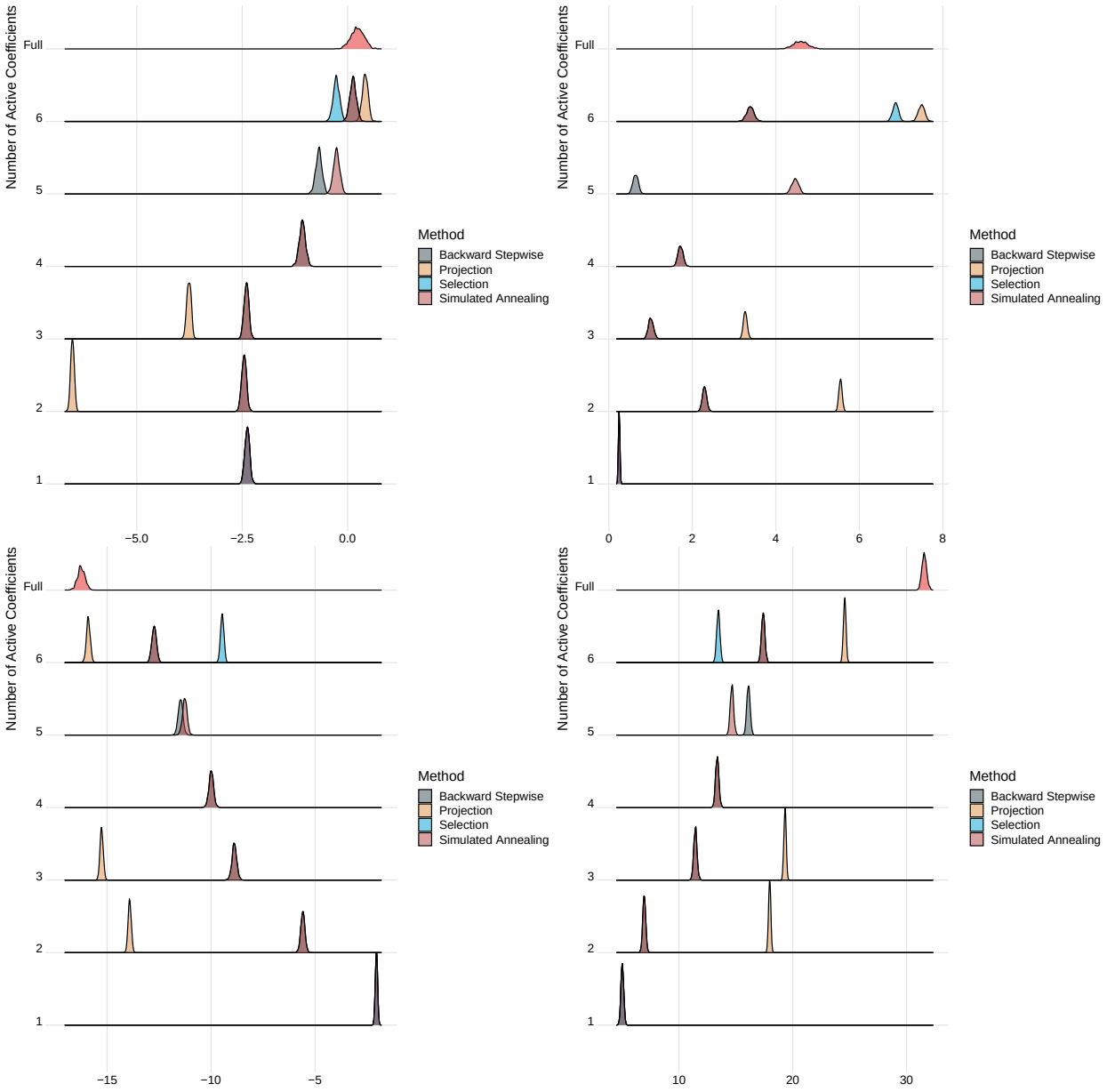


Figure 5: Ridgeline plots for 1 to 6 coefficients. The top left plot is the observation whose coarsened posterior prediction is closest to the true posterior predictive distribution in terms of 2-Wasserstein distance (0^{th} percentile in terms of ranks), and the bottom right is the observations whose is furthest from the true posterior predictive distribution (100^{th} percentile in terms of ranks). Top row, left to right: 0^{th} and 33^{rd} percentiles. Bottom row, left to right: 67^{th} and 100^{th} percentiles.