ps9 Model Comparison

] # Question 1

prelims

```r
rm(list=ls())
set.seed(123)
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
##
## rstan version 2.32.5 (Stan version 2.32.2)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

```r
library(loo)
```

```
## This is loo version 2.6.0
```

```
## - Online documentation and vignettes at mc-stan.org/loo
```

```
## - As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' argument or
set options(mc.cores = NUM_CORES) for an entire session.
```

```
##
## Attaching package: 'loo'
```

```
## The following object is masked from 'package:rstan':
##
##     loo
```

```r
options(mc.cores = parallel::detectCores())
```

Data Processing

```r
# Data Processing
data = read.csv("Wages1.csv")
wages = data$wage
school = data$school
exper = data$exper
sex = ifelse(data$sex == "male", 0, 1)
N = length(wages)
```

step 1: Compare models using WAIC

```r
m1_w = stan_model("p9_q1m1_w.stan")
m2_w = stan_model("p9_q1m2_w.stan")
```

Model 1 WAIC

```r
fit_m1_w = sampling(m1_w, iter = 1000, chains = 4,
                 data = list(N = N,
                             log_wage = log(wages),
                             schooling = school,
                             experience = exper), refresh=0)
logLike_m1 = extract_log_lik(fit_m1_w, 'logLike')
WAIC_m1 = waic(logLike_m1)
print(WAIC_m1)
```

```
##
## Computed from 2000 by 3294 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic  -2944.1  73.3
## p_waic         6.6   0.7
## waic        5888.1 146.5
```

Model 2 WAIC

```r
fit_m2_w = sampling(m2_w, iter = 1000, chains = 4,
                 data = list(N = N,
                             log_wage = log(wages),
                             schooling = school,
                             experience = exper,
                             gender = sex), refresh=0)
logLike_m2 = extract_log_lik(fit_m2_w, 'logLike')
WAIC_m2 = waic(logLike_m2)
print(WAIC_m2)
```

```
##
## Computed from 2000 by 3294 log-likelihood matrix
##
##           Estimate    SE
## elpd_waic  -2877.3  74.1
## p_waic         9.6   0.9
## waic        5754.6 148.2
```

step 2: Compare models using CV estimates of the deviance

```r
m1_cv = stan_model("p9_q1m1_cv.stan")
```

```
## Warning in readLines(file, warn = TRUE): incomplete final line found on
## '/Users/thierno/Desktop/Caltech/Junior/Winter/Ec112-BayesianStatistics/ps9_package/p9_q1m1_cv.stan'
```

```r
m2_cv = stan_model("p9_q1m2_cv.stan")
```

Model 1 out-of-sample deviance measure of predictive fit

```r
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
wages = log(wages)
numFolds = 9
testIdx = createFolds(1:length(wages), k = numFolds, list = TRUE, returnTrain = FALSE)
nFolds = length(testIdx)
pointLogLikeTotal = vector()

for (i in 1:nFolds) {
  print(paste("fold #", i))
  X = cbind(school, exper)
  YTrain = wages[-testIdx[[i]]]
  XTrain = X[-testIdx[[i]], ]
  XTest = X[testIdx[[i]], ]
  YTest = wages[testIdx[[i]]]

  fit = sampling(m1_cv, iter = 1000, chains = 4, refresh=0,
               data = list(NTest = dim(XTest)[1],
                           NTrain = dim(XTrain)[1],
                           XTrain = XTrain,
                           XTest = XTest,
                           YTrain = YTrain,
                           YTest = YTest))
  logLike = extract_log_lik(fit, 'logLike')
  pointLogLikeTemp = colMeans(logLike)
  pointLogLikeTotal = c(pointLogLikeTotal, pointLogLikeTemp)
}
```

```
## [1] "fold # 1"
## [1] "fold # 2"
## [1] "fold # 3"
## [1] "fold # 4"
## [1] "fold # 5"
## [1] "fold # 6"
## [1] "fold # 7"
## [1] "fold # 8"
## [1] "fold # 9"
```

```r
paste("Out-of-sample deviance measure of predictive fit for model 1: ", sum(pointLogLikeTotal))
```

```
## [1] "Out-of-sample deviance measure of predictive fit for model 1:  -2947.37113880889"
```

Model 2 out-of-sample deviance measure of predictive fit

```r
pointLogLikeTotal_m2 = vector()

for (i in 1:nFolds) {
  print(paste("fold #", i))
  log_wage = wages[-testIdx[[i]]]
  schooling = school[-testIdx[[i]]]
  experience = exper[-testIdx[[i]]]
  gender = sex[-testIdx[[i]]]
  log_wage2 = wages[testIdx[[i]]]
  schooling2 = school[testIdx[[i]]]
  experience2 = exper[testIdx[[i]]]
  gender2 = sex[testIdx[[i]]]

  fit = sampling(m2_cv, iter = 1000, chains = 4, refresh=0,
               data = list(NTrain = length(log_wage),
                           NTest = length(log_wage2),
                           log_wage = log_wage,
                           schooling = schooling,
                           experience = experience,
                           gender = gender,
                           log_wage2 = log_wage2,
                           schooling2 = schooling2,
                           experience2 = experience2,
                           gender2 = gender2))
  logLike = extract_log_lik(fit, 'logLike')
  pointLogLikeTemp = colMeans(logLike)
  pointLogLikeTotal_m2 = c(pointLogLikeTotal_m2, pointLogLikeTemp)
}
```

```
## [1] "fold # 1"
## [1] "fold # 2"
## [1] "fold # 3"
## [1] "fold # 4"
## [1] "fold # 5"
## [1] "fold # 6"
## [1] "fold # 7"
## [1] "fold # 8"
## [1] "fold # 9"
```

```r
paste("Out-of-sample deviance measure of predictive fit for model 1: ", sum(pointLogLikeTotal_m2))
```

```
## [1] "Out-of-sample deviance measure of predictive fit for model 1:  -2882.09229211127"
```

step 3

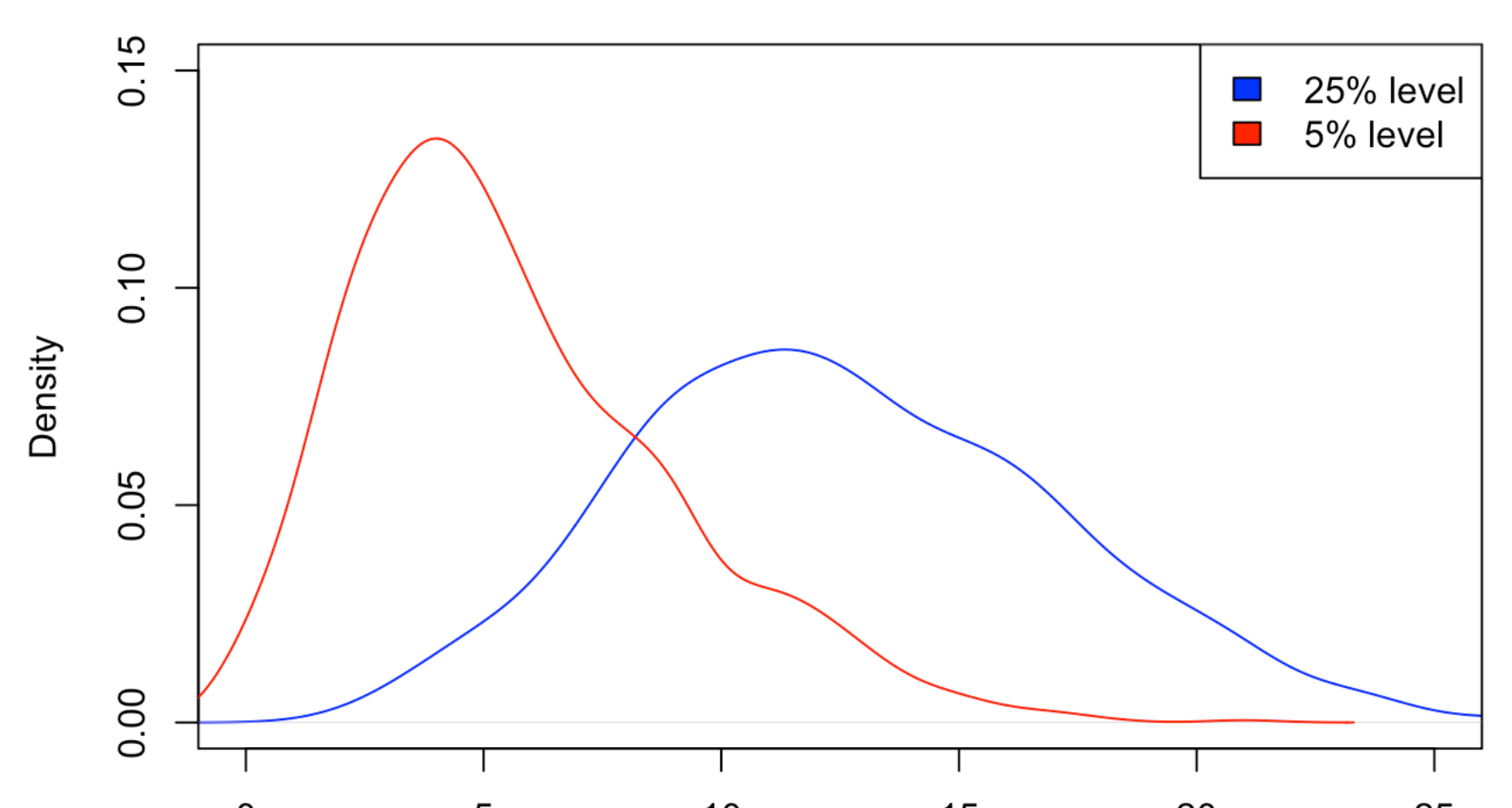
Model 2 is better because its WAIC is lower therefore it has better out of smaple performance. Model 2 is still has a deviance closer to zero using cross validation therefore it is still better than model model 1 hence both comparisons are consistent in hsowing model 2 is better.

Question 2

prelims

step 1

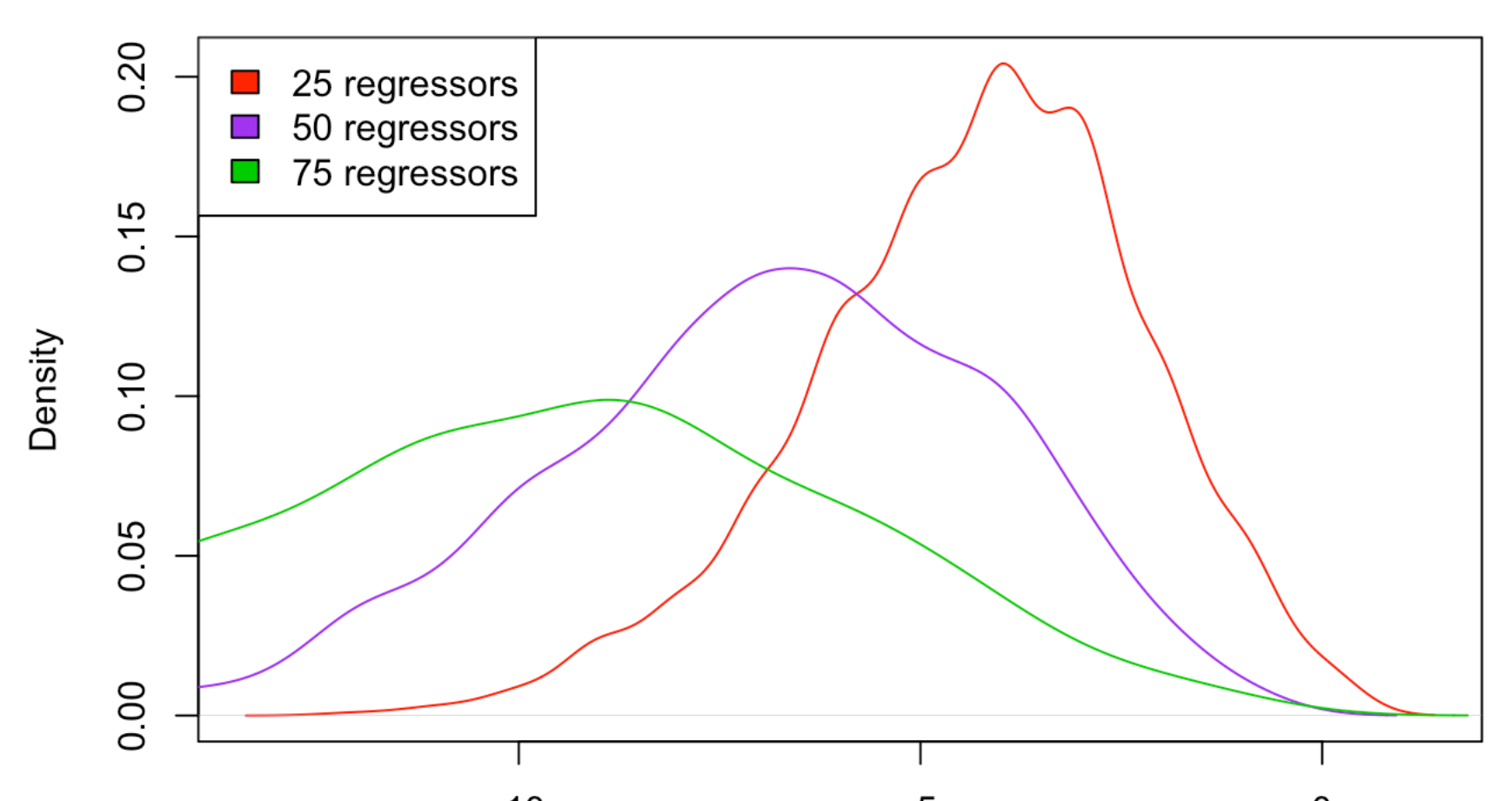
```r
surviced_1 <- vector("integer", 1000)
surviced_2 <- vector("integer", 1000)

for (i in 1:1000) {
  N = 100
  K = 50
  X = matrix(rnorm(N * K), ncol = K)
  Y = rnorm(N)

  full_model = lm(Y ~ X)
  summ_1 = summary(full_model)
  sig_1 = which(summ_1$coefficients[2:(K + 1), 4] <= 0.25)

  sub_model = lm(Y ~ X[ ,sig_1])
  summ_2 = summary(sub_model)
  sig_2 = which(summ_2$coefficients[2:(length(sig_1) + 1), 4] <= 0.05)

  surviced_1[i] = length(sig_1 <= 0.05)
  surviced_2[i] = length(sig_2 <= 0.05)
}
plot(main= "Distribution of Significant Coefficients",density(surviced_1),
     xlab= "Number of Coefficients", ylim=c(0, 0.15), xlim=c(0, 25), col="blue")
lines(density(surviced_2), col="red")
legend("topright", legend=c(" 25% level", " 5% level"), fill=c("blue", "red"))
```



step 2

```r
ensp_list <- list()

for (k in c(25, 50, 75)) {
  ensp <- vector("integer", 1000)
  for (i in 1:1000) {
    N = 100
    K = k
    X = matrix(rnorm(N * K), ncol = K)
    Y = rnorm(N)

    full_model = lm(Y ~ X)
    summ_1 = summary(full_model)
    sig_1 = which(summ_1$coefficients[2:(K + 1), 4] <= 0.25)

    if (length(sig_1) == 0) {next}

    sub_model = lm(Y ~ X[ ,sig_1])
    summ_2 = summary(sub_model)
    sig_2 = which(summ_2$coefficients[2:(length(sig_1) + 1), 4] <= 0.05)

    ensp[i] = length(sig_2 <= 0.05) - length(sig_1 <= 0.05)
  }
  ensp_list[[as.character(k)]] <- ensp
}
```

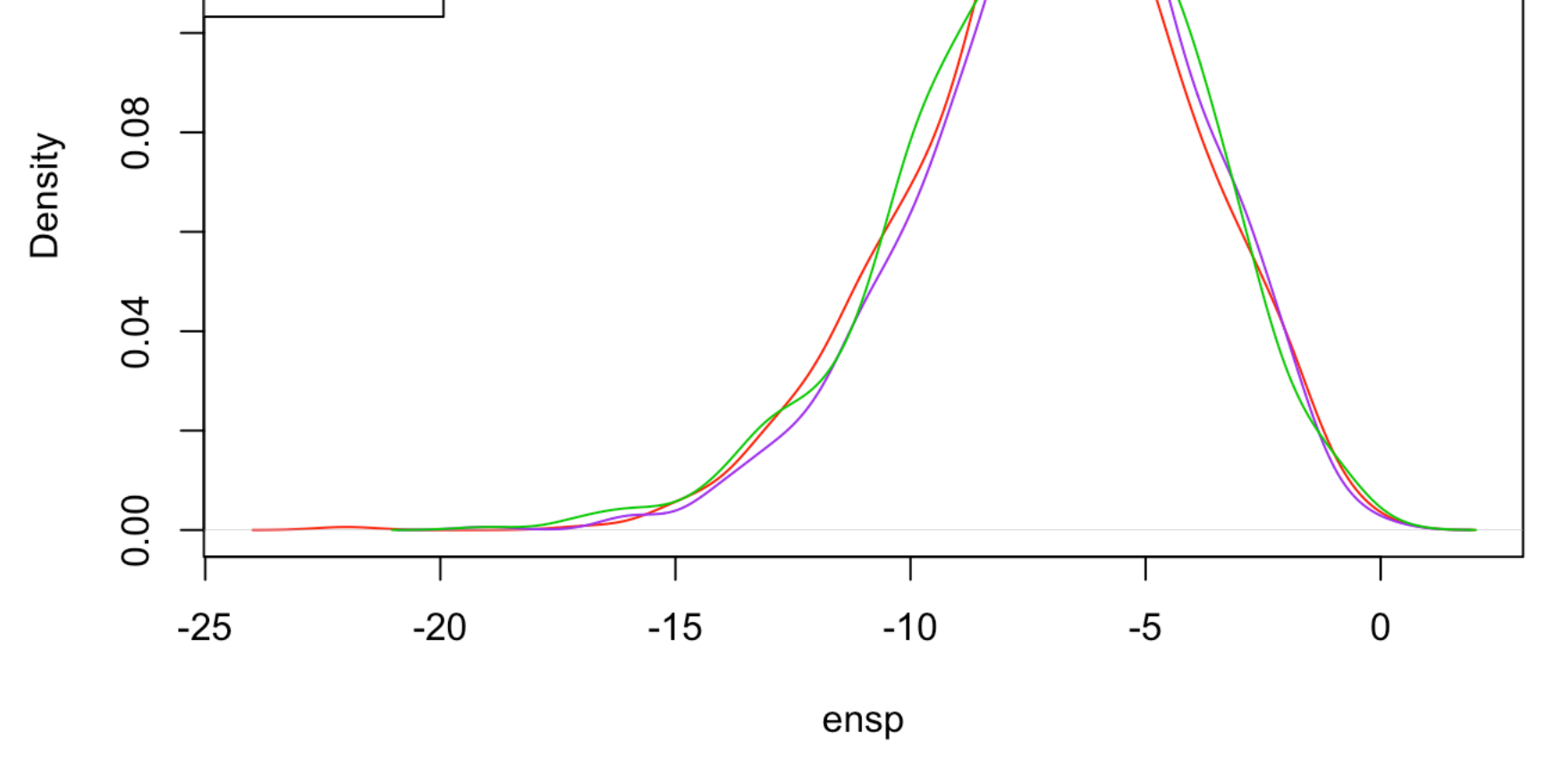
```r
plot(main="Distribution of ensp For Different Regressors.", xlab="ensp",
     density(ensp_list[["25"]]), col="red")
lines(density(ensp_list[["50"]]), col="purple")
lines(density(ensp_list[["75"]]), col="green3")
legend("topleft", legend = c("25 regressors", "50 regressors", "75 regressors"),
       fill=c("red", "purple", "green3"))
```



step 3

```r
ensp_list <- list()

for (sd in c(1, 5, 10)) {
  ensp <- vector("integer", 1000)
  for (i in 1:1000) {
    N = 100
    K = 50
    X = matrix(rnorm(N * K, 0, sd), ncol = K)
    Y = rnorm(N)

    full_model = lm(Y ~ X)
    summ_1 = summary(full_model)
    sig_1 = which(summ_1$coefficients[2:(K + 1), 4] <= 0.25)

    if (length(sig_1) == 0) {next}

    sub_model = lm(Y ~ X[ ,sig_1])
    summ_2 = summary(sub_model)
    sig_2 = which(summ_2$coefficients[2:(length(sig_1) + 1), 4] <= 0.05)

    ensp[i] = length(sig_2 <= 0.05) - length(sig_1 <= 0.05)
  }
  ensp_list[[as.character(sd)]] <- ensp
}
```

```r
plot(main="Distribution of ensp For Different Regressors.", xlab="ensp",
     density(ensp_list[["1"]]), col="red")
lines(density(ensp_list[["5"]]), col="purple")
lines(density(ensp_list[["10"]]), col="green3")
legend("topleft", legend = c("sd = 1", "sd = 5", "sd = 10"),
       fill=c("red", "purple", "green3"))
```



This is different because chnaging the sd of the predictors does not impact the relative likelihood of identifying a predictor. However, increasing the number regressions increases model complexity leading to overfitting resulting in more false positives.