$\textbf{Assignment}_4$ R.C.

Bayesian Methods

Achladianakis Minas

A report presented for the R.C. Bayesian Methods

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

Department of Applied Mathematics

University of Crete

Greece, March 21, 2025

Abstract

This assignment is part of my evaluation for the R.C. Bayesian Methods.

List of Figures

BMS vs Ridge

1.1 Exercise Purpose & Data Preparation

For this assignment, I'll be using build-in R dataset "airquality" to develop a parsimonious model. In the first Part of the assignment, I'll focus in creating that prediction model using the BMS package for Bayesian model averaging. In the second part, I'll create the model based on Ridge regression. In the final section, I will compare these two approaches and derive conclusions.

Data prep

```
#The data
data("airquality")

# Function to replace missing values with the mean of the column
replace_na_with_mean <- function(x) {
  if (is.numeric(x)) {
    x[is.na(x)] <- mean(x, na.rm = TRUE)
  }
  return(x)
}

# Function to clean each column of the airquality dataset
airquality_clean <- apply(airquality, 2, replace_na_with_mean)

# Convert the result back to a data frame
airquality_clean <- as.data.frame(airquality_clean)

set.seed(123) # For reproducibility

rho <- 0.8 # Desired correlation coefficient

# Create Z1 correlated with Solar.R
Z1 <- rho * airquality_clean$Solar.R + sqrt(1 - rho^2) *
↪  rnorm(nrow(airquality_clean), sd = sd(airquality_clean$Solar.R))

# Create Z2 correlated with Wind
Z2 <- rho * airquality_clean$Wind + sqrt(1 - rho^2) *
↪  rnorm(nrow(airquality_clean), sd = sd(airquality_clean$Wind))

# Create Z3 correlated with Temp
Z3 <- rho * airquality_clean$Temp + sqrt(1 - rho^2) *
↪  rnorm(nrow(airquality_clean), sd = sd(airquality_clean$Temp))
```

```
# Add these to your dataset
airquality_clean$Z1 <- Z1
airquality_clean$Z2 <- Z2
airquality_clean$Z3 <- Z3

# Create polynomial and interaction terms
airquality_clean$Solar.R2 <- airquality_clean$Solar.R^2
airquality_clean$Wind2 <- airquality_clean$Wind^2
airquality_clean$Temp2 <- airquality_clean$Temp^2
airquality_clean$SolarR_Wind <- airquality_clean$Solar.R *
↪  airquality_clean$Wind
airquality_clean$SolarR_Temp <- airquality_clean$Solar.R *
↪  airquality_clean$Temp
airquality_clean$Wind_Temp <- airquality_clean$Wind *
↪  airquality_clean$Temp

> summary(airquality_clean)
     Ozone           Solar.R          Wind             Temp
      ↪  Month            Day
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00   Min.
 ↪  :5.000   Min.   : 1.0
 1st Qu.: 21.00   1st Qu.:120.0   1st Qu.: 7.400   1st Qu.:72.00   1st
 ↪  Qu.:6.000   1st Qu.: 8.0
 Median : 42.13   Median :194.0   Median : 9.700   Median :79.00   Median
 ↪  :7.000   Median :16.0
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88   Mean
 ↪  :6.993   Mean   :15.8
 3rd Qu.: 46.00   3rd Qu.:256.0   3rd Qu.:11.500   3rd Qu.:85.00   3rd
 ↪  Qu.:8.000   3rd Qu.:23.0
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00   Max.
 ↪  :9.000   Max.   :31.0
       Z1                  Z2                  Z3              Solar.R2
        ↪  Wind2             Temp2
 Min.   :-58.49   Min.   :-0.4045   Min.   :35.41   Min.   :    49   Min.
 ↪  :  2.89   Min.   :3136
 1st Qu.: 86.54   1st Qu.: 5.2944   1st Qu.:56.10   1st Qu.: 14400   1st
 ↪  Qu.: 54.76   1st Qu.:5184
 Median :155.90   Median : 7.7649   Median :62.28   Median : 37636
 ↪  Median : 94.09   Median :6241
 Mean   :147.84   Mean   : 7.8767   Mean   :61.50   Mean   : 42257   Mean
 ↪  :111.48   Mean   :6155
 3rd Qu.:199.92   3rd Qu.:10.2888   3rd Qu.:67.21   3rd Qu.: 65536   3rd
 ↪  Qu.:132.25   3rd Qu.:7225
 Max.   :348.56   Max.   :16.2390   Max.   :83.51   Max.   :111556   Max.
 ↪  :428.49   Max.   :9409
  SolarR_Wind      SolarR_Temp      Wind_Temp
 Min.   :  48.3   Min.   :  472   Min.   : 129.2
 1st Qu.: 984.0   1st Qu.: 8760   1st Qu.: 593.4
 Median :1575.5   Median :16184   Median : 745.2
 Mean   :1834.4   Mean   :14698   Mean   : 760.3
```

```
3rd Qu.:2564.5   3rd Qu.:20262   3rd Qu.: 924.6
Max.   :5878.8   Max.   :28101   Max.   :1490.4
```

1.2 Bayesian Model Averaging (BMA)

BMA is used to account for model uncertainty, it weighs the results of different models 2^K according to their **Posterior Model Probability (PMP)** calculated using Bayes' theorem. It also penalizes for the number of predictors, thus tents to create parsimonious models, that help us to avoid overfitting. One of the criteria by which the BMA evaluates the models is the **Model Posterior Inclusion Probability (PIP)**, which is the probability that a particular variable is included in the true model and is calculated y summing the PMPs of all models. BMS calculates the posterior mean and variance of the coefficients of each model which we will present shortly for the analysis of the "airquality" dataset. As I was instructed the predictors I am going to depend upon for my prediction are the linear and quadratic terms of Solar.R, Wind and Temp as well as their interactions Solar.R:Wind, Solar.R:Temp and Wind:Temp as well as the Z1, Z2, Z3 predictors correlated with rho 0.8 with Solar.R, Wind and Temp respectively, all created and include in our data as previously stated.

Uniformly-Informative Prior

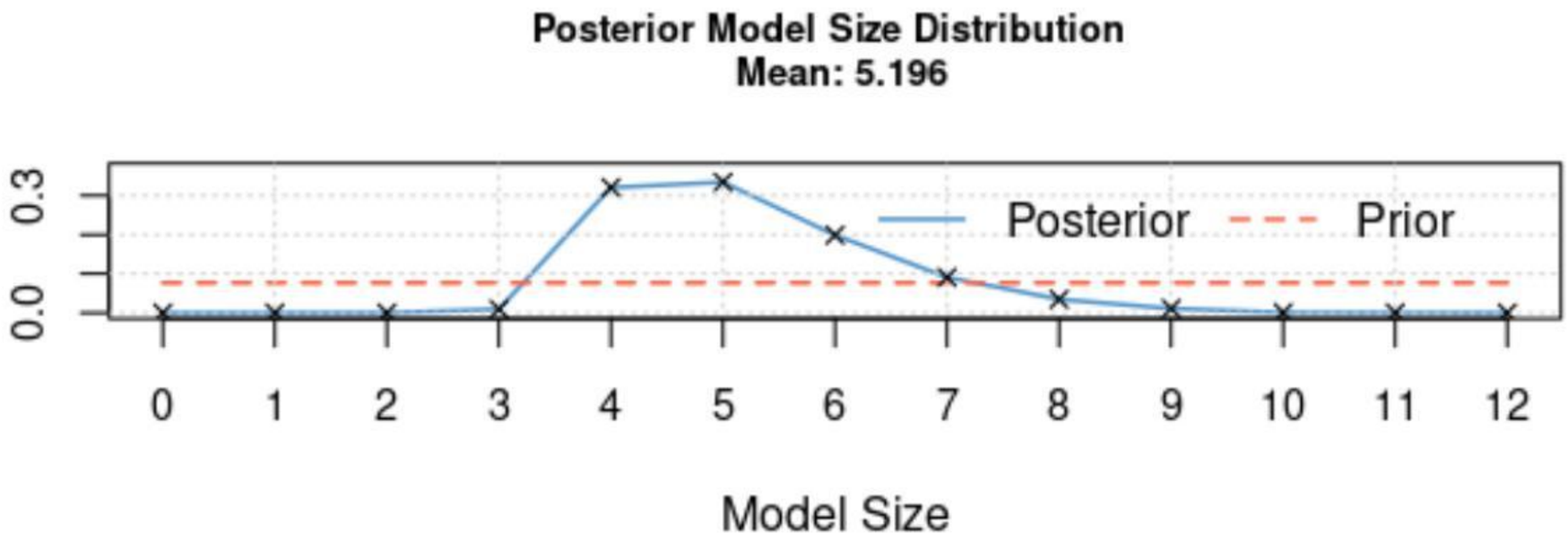
There is no shortage in the priors we can use for the bms, though one of the most commonly used is the UIP especially when you integrate prior knowledge into your model as we will, by specifying our prior knowledge about certain variables a.k.a the predictors I mentioned earlier.

```r
# Load the BMS package
library(BMS)

# Create a new data frame excluding 'Day' and 'Month'
airquality_bms <- airquality_clean[, !(names(airquality_clean) %in%
↪  c("Day", "Month"))]

# Perform BMA using the UIP prior and Birth-death MCMC method and the
↪  specified starting predictors
bms_result <- bms(airquality_bms,
                  mprior = "UIP",
                  mcmc = "bd")

# View the results
print(bms_result)
```

the results where

```
                PIP     Post Mean      Post SD Cond.Pos.Sign Idx
Wind2         0.9706667  3.083545e-01 0.0964287047    1.00000000   8
Temp2         0.8833333  2.160514e-02 0.0177606941    1.00000000   9
Wind_Temp     0.8023333 -9.097801e-02 0.0532148486    0.00457000  12
SolarR_Temp   0.4726667  7.300458e-04 0.0011205900    0.99647391  11
Solar.R       0.4450000  5.832274e-02 0.0966496867    0.96479401   1
```

```
Temp         0.3960000 -1.238571e+00 3.0097509205    0.28451178   3
SolarR_Wind 0.3556667 -3.075960e-03 0.0056128147    0.08153702  10
Solar.R2     0.2640000 -4.713745e-05 0.0001664889    0.25252525   7
Wind         0.2483333 -1.695109e+00 4.8950169833    0.18120805   2
Z3           0.1416667 -5.341967e-02 0.1722424333    0.00000000   6
Z1           0.1213333 -3.228641e-03 0.0131707579    0.02472527   4
Z2           0.0950000  6.373535e-02 0.2899527444    1.00000000   5

Mean no. regressors                  Draws                Burnins
↪  Time  No. models visited
          "5.1960"                  "3000"                  "1000"     "0.4056017
          ↪  secs"                  "1005"
      Modelspace 2^K                % visited          % Topmodels
       ↪  Corr PMP                No. Obs.
             "4096"                    "25"                   "100"
              ↪  "0.9667"                   "153"
        Model Prior                 g-Prior      Shrinkage-Stats
       "random / 6"                   "UIP"          "Av=0.9935"

Time difference of 0.4056017 secs
```



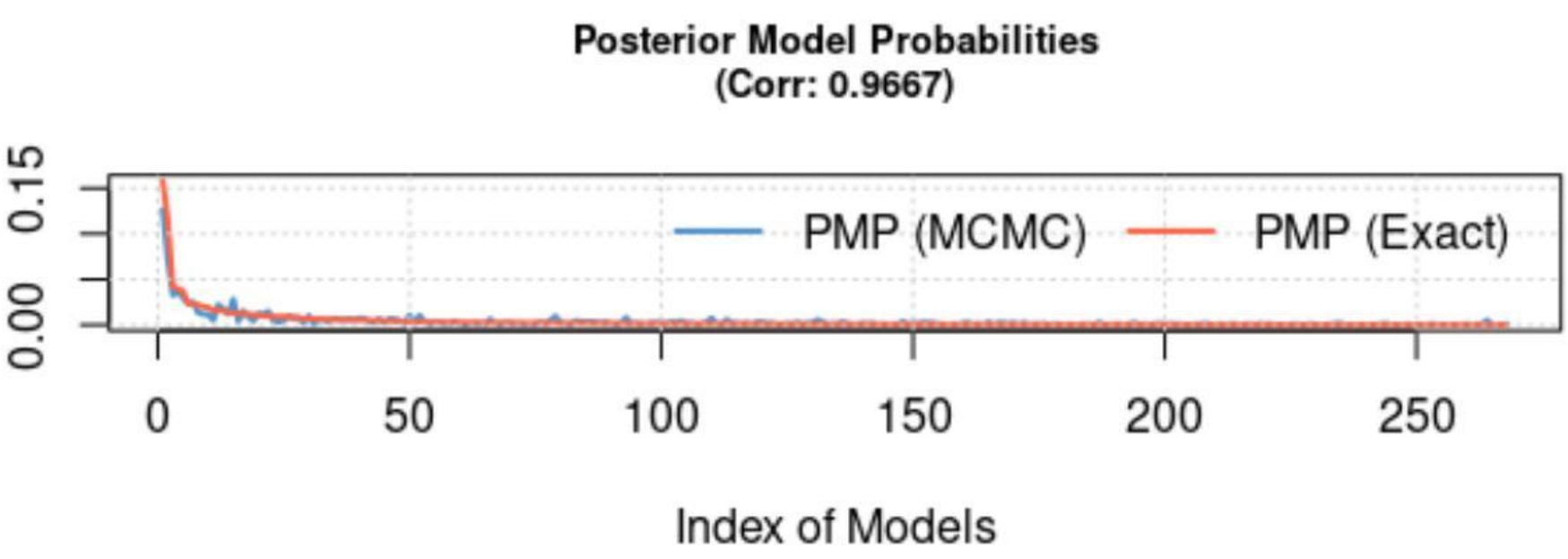


Figure 1.1: BMA using the UIP and bd method

The top performers are:

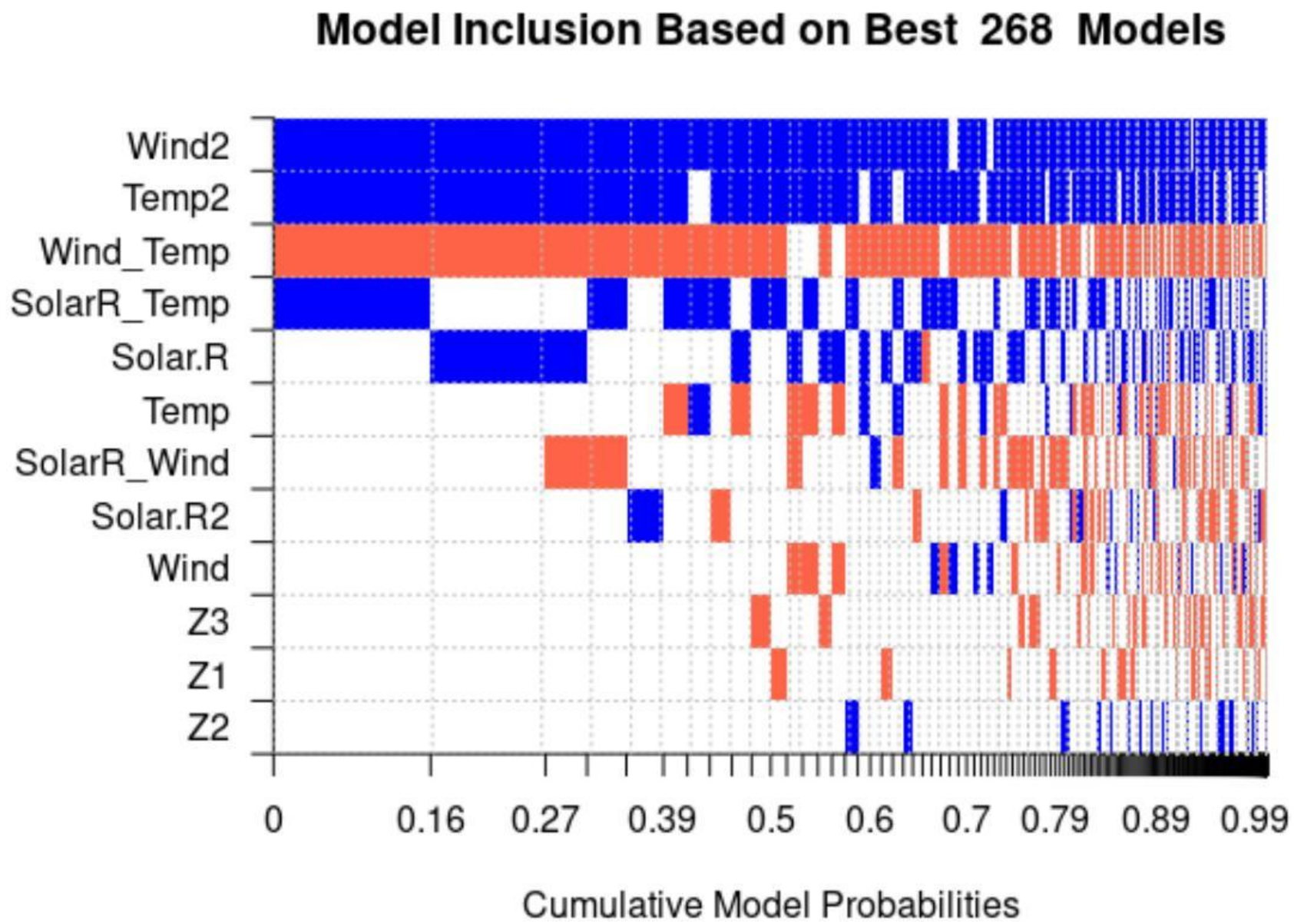


Figure 1.2: Model inclusion best 268 UIP

```
> topmodels.bma(bms_result)[,1:5]
                  001b      0819       081d       001f      0039
Solar.R      0.0000000 1.0000000 1.00000000 0.00000000 0.0000000
Wind         0.0000000 0.0000000 0.00000000 0.00000000 0.0000000
Temp         0.0000000 0.0000000 0.00000000 0.00000000 0.0000000
Z1           0.0000000 0.0000000 0.00000000 0.00000000 0.0000000
Z2           0.0000000 0.0000000 0.00000000 0.00000000 0.0000000
Z3           0.0000000 0.0000000 0.00000000 0.00000000 0.0000000
Solar.R2     0.0000000 0.0000000 0.00000000 0.00000000 1.0000000
Wind2        1.0000000 1.0000000 1.00000000 1.00000000 1.0000000
Temp2        1.0000000 1.0000000 1.00000000 1.00000000 1.0000000
SolarR_Wind 0.0000000 0.0000000 1.00000000 1.00000000 0.0000000
SolarR_Temp 1.0000000 0.0000000 0.00000000 1.00000000 0.0000000
Wind_Temp    1.0000000 1.0000000 1.00000000 1.00000000 1.0000000
PMP (Exact) 0.1580855 0.1157041 0.04196082 0.03952652 0.0372812
PMP (MCMC)  0.1266667 0.0710000 0.03266667 0.03633333 0.0300000
```

g- UIP

Running Bayesian Model Averaging (BMA) with the bms package can be tailored with various settings to fit different needs and research questions. Previously we used the prior UIP which assigns equal probabilities to all models of the same size but different probabilities to models of different sizes, based on a binomial distribution. Now we can use the **UIP g-prior** as described by our bms.pdf file which treats all models as equally likely regardless of their size with treatment of regressors within

each model, influencing the shrinkage of coefficients. Furthermore, the previous run didn't explicitly specify parameters bms allows us to adjust as we please as the argument denoted as nmodel, which indicates to the model the number of best-performing models to keep, to later present in the Inclusion Based illustrations (here 100).

```
# Perform BMA with different settings
bms_g_result <- bms(airquality_bms,
                    mprior = "uniform",
                    g = "UIP",
                    nmodel = 100)
```

with results

```
                PIP     Post Mean      Post SD Cond.Pos.Sign Idx
Wind2     0.9613074  0.2990936156 0.1000136019    0.99999962   8
Temp2     0.8797887  0.0200877098 0.0163384752    1.00000000   9
Wind_Temp 0.8501670 -0.0985239253 0.0524312709    0.00177522  12
SolarR_Temp 0.5311578  0.0008645308 0.0012247556    0.99640227  11
Solar.R   0.4282708  0.0465396155 0.0909441077    0.92346519   1
Temp      0.3548277 -0.9506726986 2.7139028589    0.30513553   3
SolarR_Wind 0.3428400 -0.0030325063 0.0054786096    0.05485804  10
Wind      0.2564377 -0.9436323260 4.9793134725    0.35407528   2
Solar.R2  0.2224023 -0.0000391788 0.0001559929    0.28178932   7
Z3        0.1585721 -0.0597414000 0.1808873511    0.00131958   6
Z1        0.1359659 -0.0039724556 0.0143073901    0.01489653   4
Z2        0.1156366  0.0773413156 0.3179372322    0.99951551   5

Mean no. regressors                Draws                Burnins
↪  Time  No. models visited
          "5.2374"                "4096"                   "0"    "0.3741758
          ↪  secs"                "4096"
     Modelspace 2^K            % visited          % Topmodels
      ↪  Corr PMP             No. Obs.
            "4096"                 "100"                "2.4"
             ↪  "NA"                 "153"
       Model Prior              g-Prior     Shrinkage-Stats
      "uniform / 6"               "UIP"         "Av=0.9935"

Time difference of 0.3741758 secs
```

[1]

The top 5 performers using g-UIP :

```
> topmodels.bma(bms_g_result)[,1:5]
                    001b          0819          081d          001f
                     ↪  021b
Solar.R     0.000000e+00 1.000000e+00 1.000000e+00 0.000000e+00
 ↪  0.000000e+00
Wind        0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 ↪  0.000000e+00
```

[1] **Commment** in retrospect I should have used 20 best

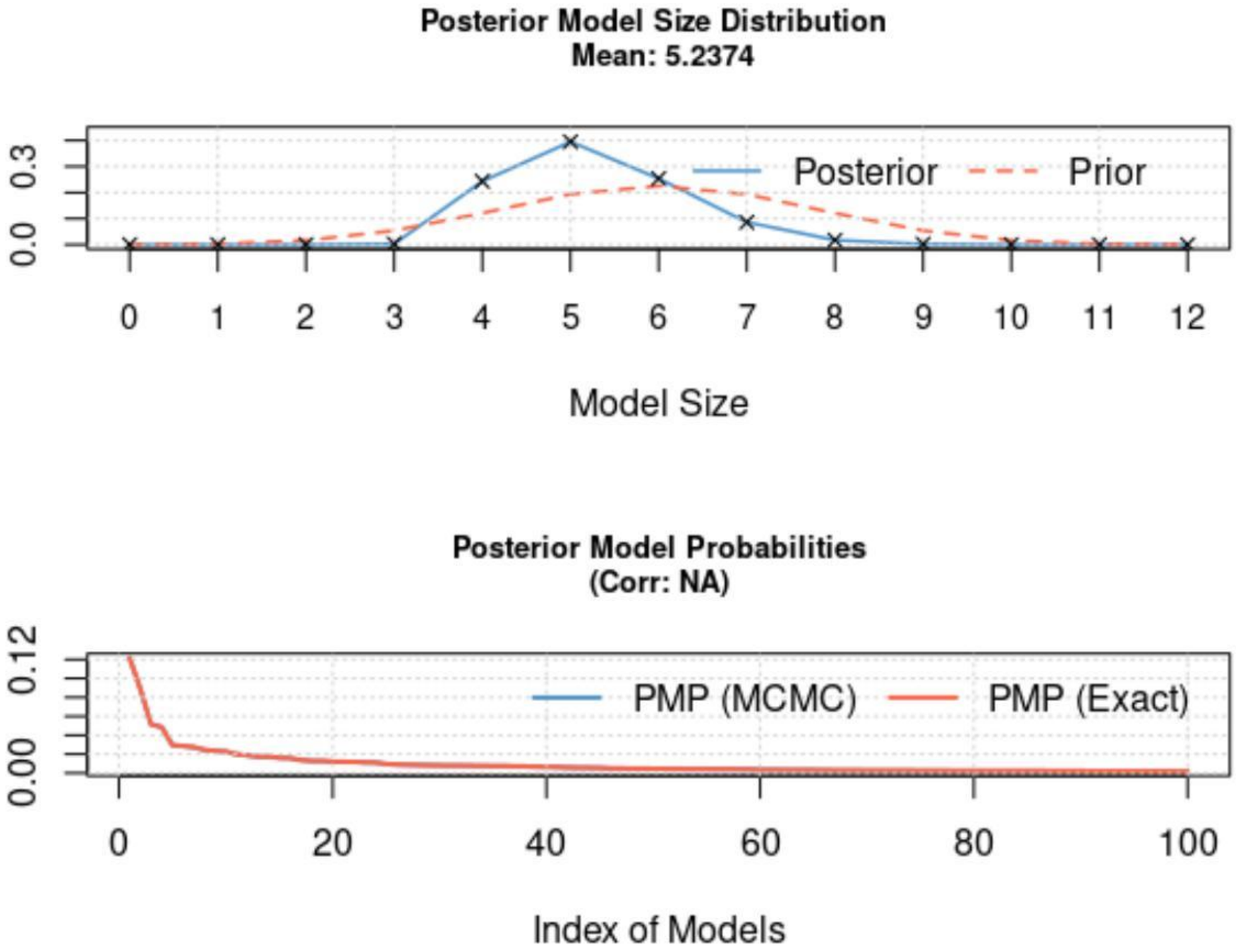


Figure 1.3: BMA using the g-UIP

```
Temp         0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 ↪  1.000000e+00
Z1           0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 ↪  0.000000e+00
Z2           0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 ↪  0.000000e+00
Z3           0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 ↪  0.000000e+00
Solar.R2     0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 ↪  0.000000e+00
Wind2        1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
 ↪  1.000000e+00
Temp2        1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
 ↪  1.000000e+00
SolarR_Wind 0.000000e+00 0.000000e+00 1.000000e+00 1.000000e+00
 ↪  0.000000e+00
SolarR_Temp 1.000000e+00 0.000000e+00 0.000000e+00 1.000000e+00
 ↪  1.000000e+00
Wind_Temp    1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
 ↪  1.000000e+00
PMP (Exact) 6.841381e+24 5.007263e+24 2.905466e+24 2.736909e+24
 ↪  1.663781e+24
```

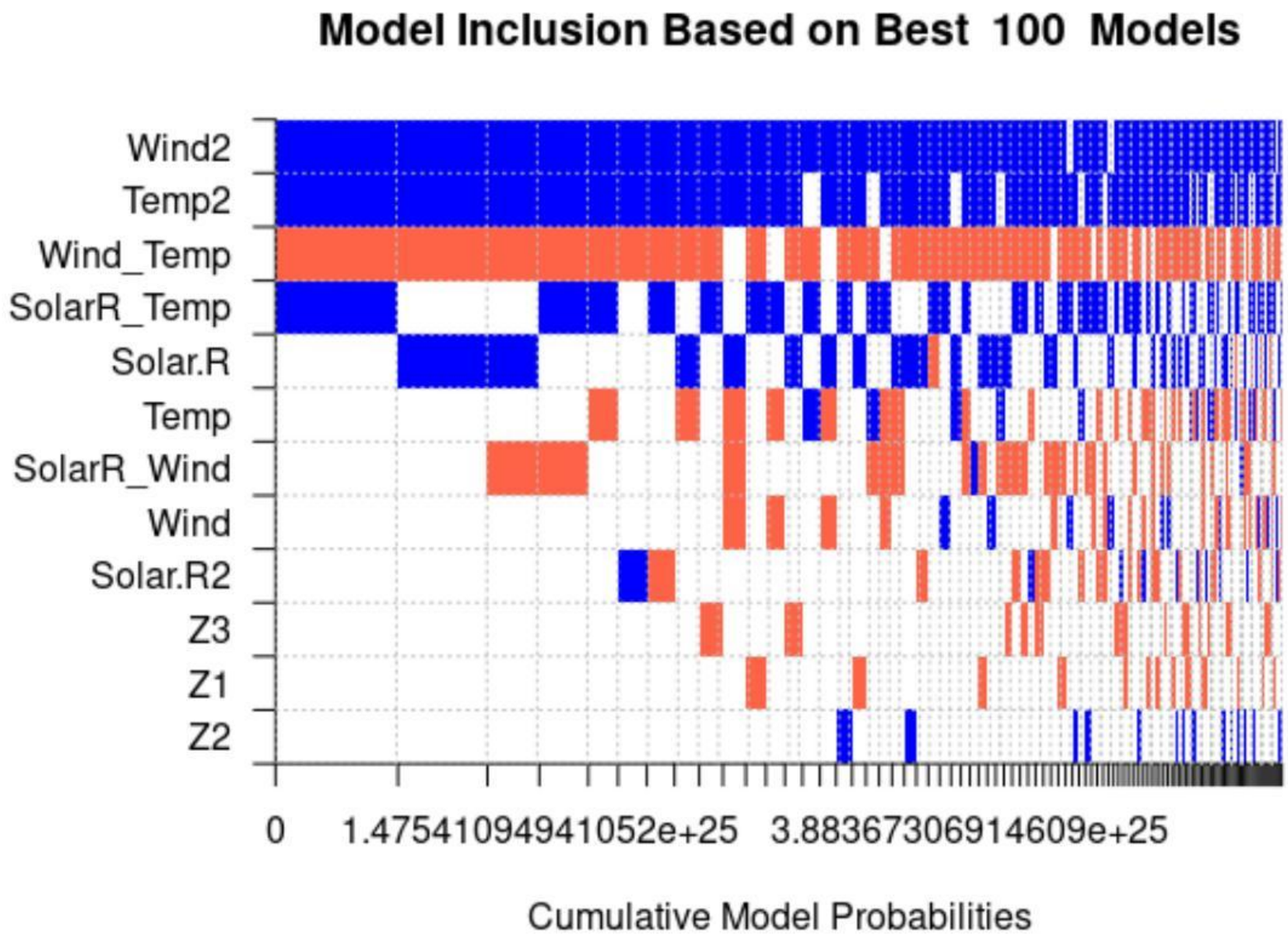


Figure 1.4: Model Inclusion best 100 g-UIP

```
PMP (MCMC)  6.841381e+24 5.007263e+24 2.905466e+24 2.736909e+24
↪  1.663781e+24
```

Fixed Prior

Now we will implement bma using a fixed prior as described in our guide bma.pdf

```
#fixed prior
bms_f_result <- bms(airquality_bms,
                    mprior = "fixed",
                    mcmc = "bd")
```

with results

```
                  PIP     Post Mean      Post SD Cond.Pos.Sign Idx
Wind2        0.9740000  0.2973097558 0.0885516761    1.00000000   8
Wind_Temp    0.9106667 -0.1041829080 0.0439344609    0.00036603  12
Temp2        0.8310000  0.0181202362 0.0153084090    1.00000000   9
SolarR_Temp  0.5873333  0.0009867707 0.0012724876    0.99829739  11
Temp         0.3893333 -0.6113245759 2.4920463001    0.41267123   3
Solar.R      0.3706667  0.0372784640 0.0822321201    0.92356115   1
SolarR_Wind  0.3390000 -0.0027243295 0.0052211486    0.07767945  10
Solar.R2     0.2483333 -0.0000490187 0.0001676128    0.23221477   7
Z3           0.1836667 -0.0680835341 0.1922053904    0.00362976   6
Wind         0.1576667 -0.5407043441 3.9744142335    0.34883721   2
Z2           0.1360000  0.0978682401 0.3545114476    1.00000000   5
```

```
Z1          0.1260000 -0.0036334723 0.0137337846    0.01587302   4

Mean no. regressors                  Draws                Burnins
 ↪  Time  No. models visited
          "5.2537"                 "3000"                "1000"    "0.3630452
          ↪  secs"                 "1044"
     Modelspace 2^K              % visited          % Topmodels
      ↪  Corr PMP              No. Obs.
            "4096"                  "25"                 "100"
             ↪  "0.9489"                "153"
        Model Prior              g-Prior       Shrinkage-Stats
        "fixed / 6"                "UIP"            "Av=0.9935"

Time difference of 0.3630452 secs
```

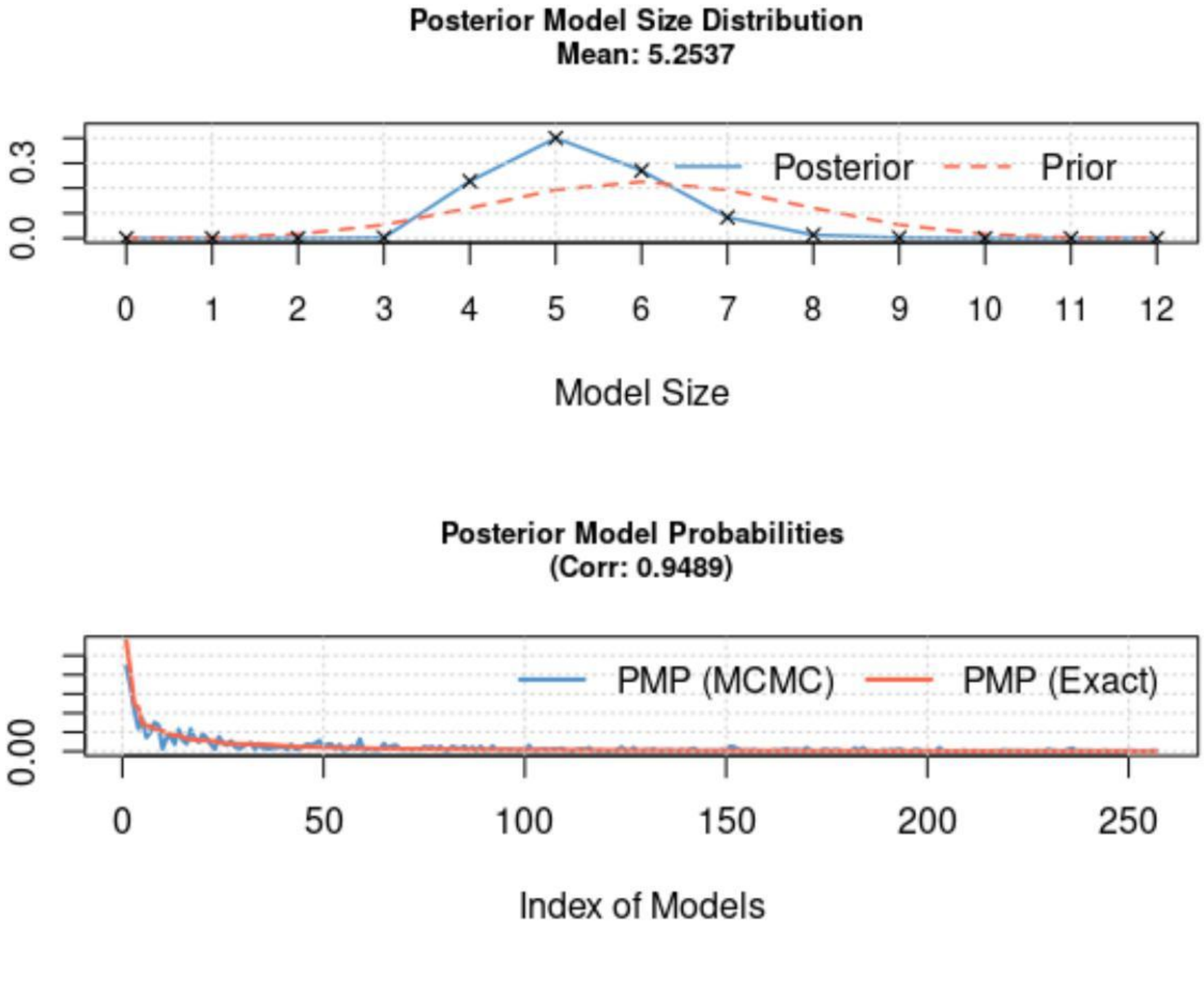


Figure 1.5: BMA using fixed prior

Best 5 performing models

	001b	0819	081d	001f	021b
Solar.R	0.00000000	1.00000000	1.00000000	0.00000000	0.00000000
Wind	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Temp	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000
Z1	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Z2	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000

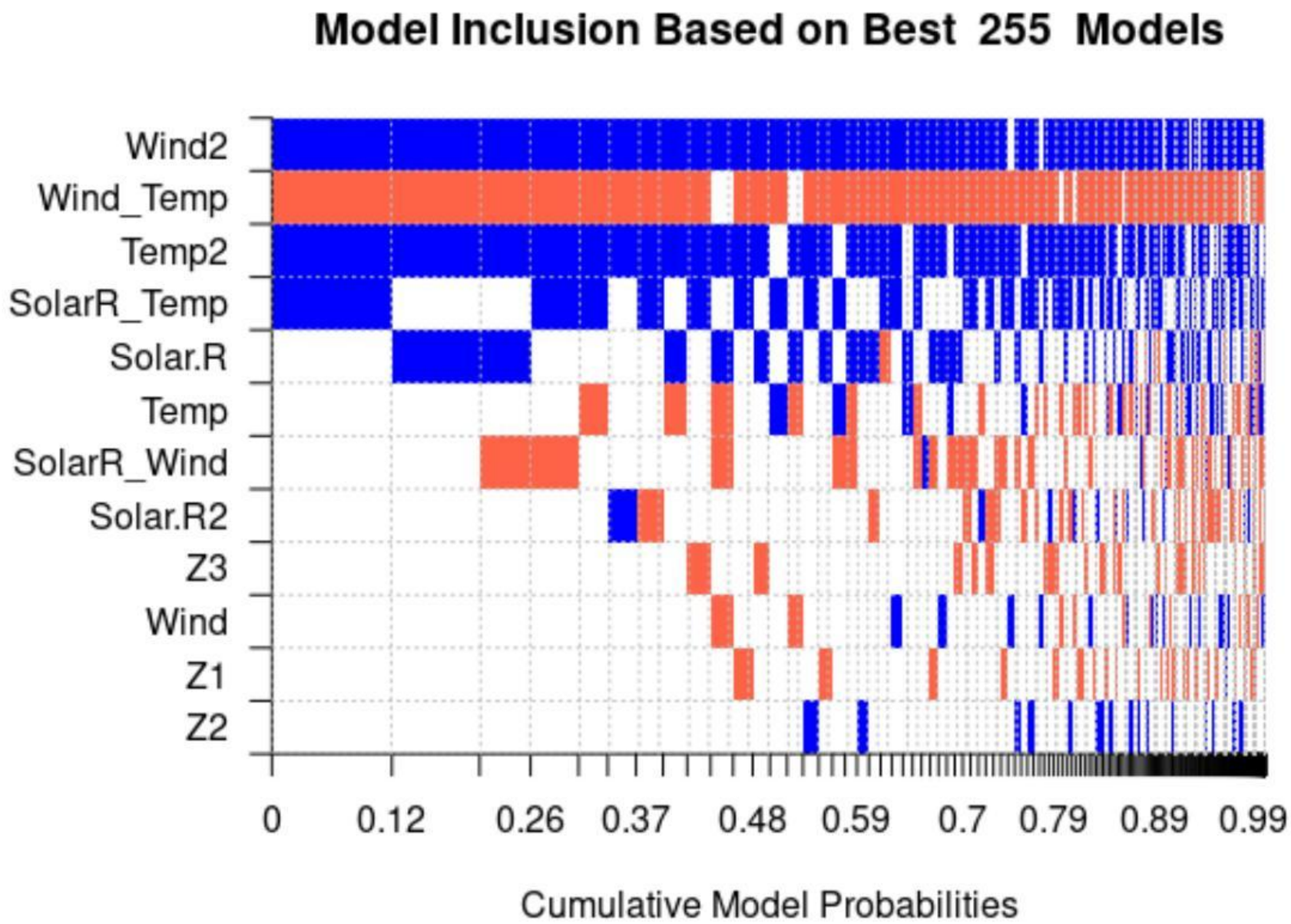


Figure 1.6: Model inclusion best 268 fixed

```
Z3           0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
Solar.R2     0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
Wind2        1.00000000 1.00000000 1.00000000 1.00000000 1.00000000
Temp2        1.00000000 1.00000000 1.00000000 1.00000000 1.00000000
SolarR_Wind 0.00000000 0.00000000 1.00000000 1.00000000 0.00000000
SolarR_Temp 1.00000000 0.00000000 0.00000000 1.00000000 1.00000000
Wind_Temp    1.00000000 1.00000000 1.00000000 1.00000000 1.00000000
PMP (Exact) 0.12076216 0.08838684 0.05128648 0.04831116 0.02936860
PMP (MCMC)  0.07466667 0.09866667 0.04666667 0.03800000 0.02066667
```

The results are better based on PMP relative to the precious run, though in both cases the coefficients of the predictors seem unsophisticated as they failed to eliminate the Zi predictors.

Random prior

Now we will implement the BMA using a random Beta-Binomial prior as described in bma.pdf

```
#  Beta-Binomial prior
bms_r_result <- bms(airquality_bms,
                    mprior = "fixed",
                    mcmc = "bd")
```

with results

```
                PIP     Post Mean      Post SD Cond.Pos.Sign Idx
Wind2      0.9876667  2.978548e-01 0.0808275638    1.00000000   8
Wind_Temp  0.9450000 -1.068954e-01 0.0389194100    0.00000000  12
Temp2      0.8883333  1.770049e-02 0.0133883012    1.00000000   9
SolarR_Temp 0.5130000  7.748697e-04 0.0011289670    0.99870045  11
Solar.R    0.4206667  4.799230e-02 0.0943423580    0.91204437   1
SolarR_Wind 0.3383333 -2.969352e-03 0.0053983468    0.04630542  10
Temp       0.2913333 -4.762095e-01 2.1409582321    0.38329519   3
Solar.R2   0.2526667 -2.642926e-05 0.0001502942    0.41160950   7
Z3         0.1580000 -6.148816e-02 0.1833445834    0.00000000   6
Wind       0.1410000 -2.777581e-01 3.2932790411    0.49172577   2
Z1         0.1236667 -3.887634e-03 0.0139142019    0.00000000   4
Z2         0.1033333  7.061031e-02 0.3039059127    1.00000000   5

Mean no. regressors                Draws                Burnins
↪  Time  No. models visited
          "5.1630"                "3000"                "1000"    "0.3619618
          ↪  secs"                "1004"
     Modelspace 2^K           % visited          % Topmodels
      ↪  Corr PMP             No. Obs.
            "4096"                "25"                 "100"
            ↪  "0.9406"                 "153"
        Model Prior              g-Prior     Shrinkage-Stats
        "fixed / 6"                "UIP"          "Av=0.9935"

Time difference of 0.3619618 secs
```

```
> topmodels.bma(bms_r_result)[,1:5]
                  001b       0819       081d       001f       021b
Solar.R     0.00000000 1.00000000 1.00000000 0.00000000 0.00000000
Wind        0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
Temp        0.00000000 0.00000000 0.00000000 0.00000000 1.00000000
Z1          0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
Z2          0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
Z3          0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
Solar.R2    0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
Wind2       1.00000000 1.00000000 1.00000000 1.00000000 1.00000000
Temp2       1.00000000 1.00000000 1.00000000 1.00000000 1.00000000
SolarR_Wind 0.00000000 0.00000000 1.00000000 1.00000000 0.00000000
SolarR_Temp 1.00000000 0.00000000 0.00000000 1.00000000 1.00000000
Wind_Temp   1.00000000 1.00000000 1.00000000 1.00000000 1.00000000
PMP (Exact) 0.11860897 0.08681089 0.05037204 0.04744977 0.02884495
PMP (MCMC)  0.09466667 0.07966667 0.06333333 0.02366667 0.03933333
```

The random approach provided better results regarding PMP. Though all of the implementations provide very similar results.

Conclusion Based on the results I observe that our first implementation yielded similar results to the other implementation based on PIP. Regarding PMP the second stood out from the rest and that the reason I did not proceed with the custom argument approach further. Furthermore, in all the implementations, the predictors Wind2, Temp2 and Wind_Temp seem to play a vital role in forecasting our Ozone

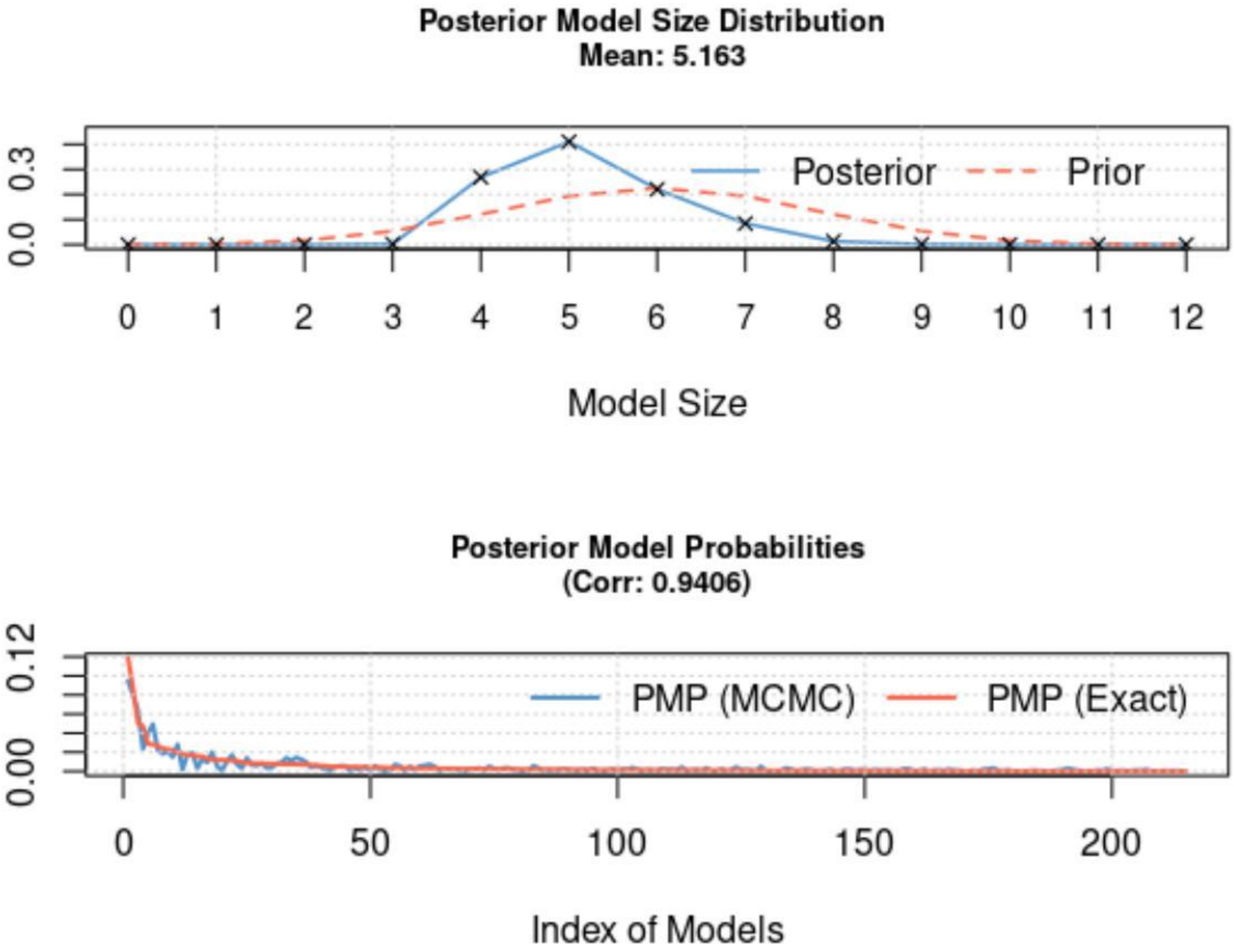


Figure 1.7: BMA using random prior

levels as they come up in almost every model all cases and the Solar_Temp with the Solar.R play an interesting role in all cases as it seems as if when one is present in a model the other is not. Additionally, in all the implementations the valuable predictors in the most parsimonious model are the Wind2, Wind_Temp, Temp2 and SolarR_Temp.
Some Marginal densities:

1.2.1 Train and Test Set

In this subsection, I will split the data into training data and test data. Then I will implement the uniform g, the fixed and the random as in our guide bms.pdf to compare them.

The 5 predictors observation

Running 10 fold CV for the bma model under the mprior "uniform" and g 'UIP'I noticed that the best model size was always 5 as seen in (A.1 and A.2)

```
library(BMS)
library(caret)

# The 10-fold cross-validation indices
```

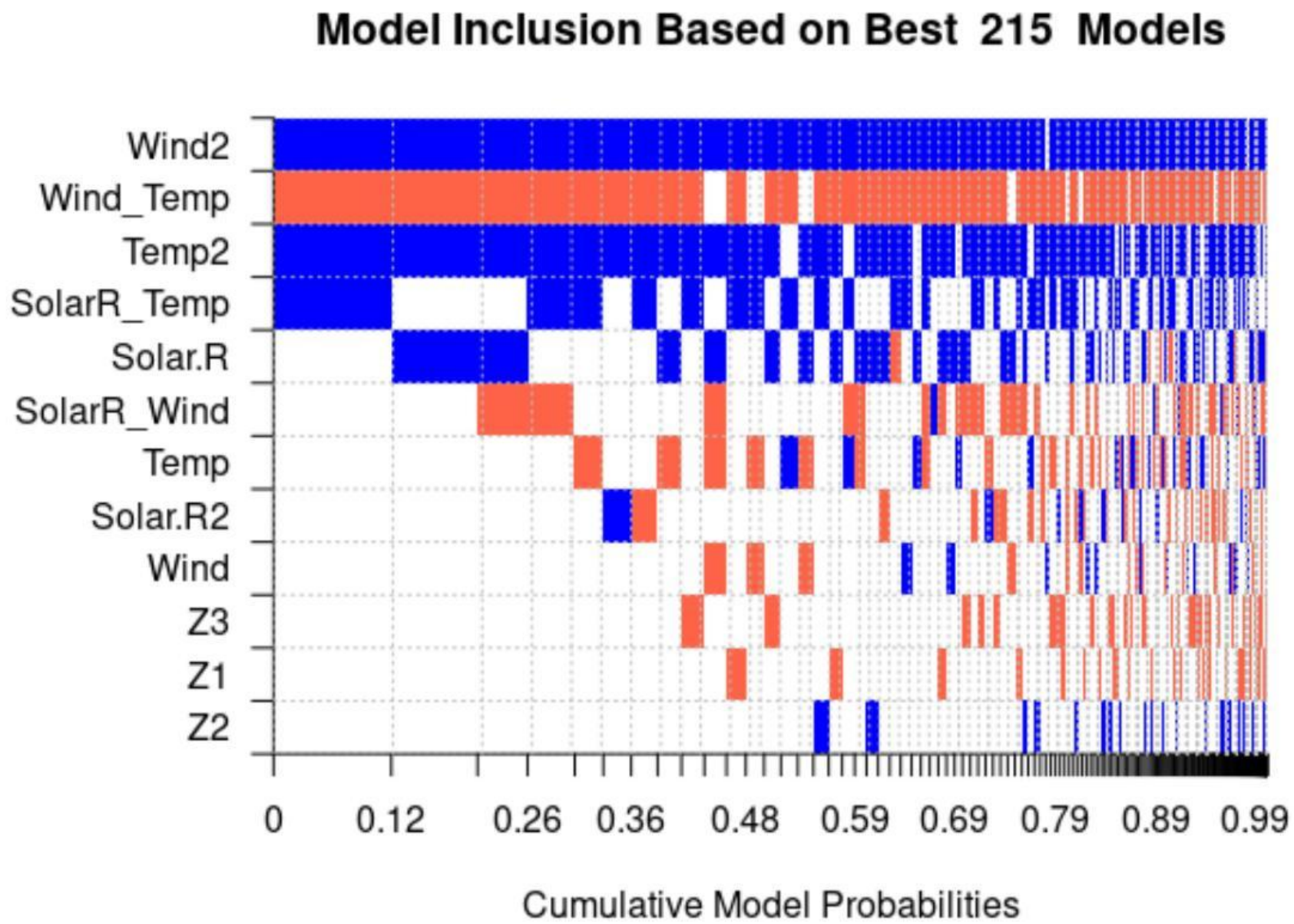


Figure 1.8: Model inclusion best 268 random

```r
set.seed(123)
folds <- createFolds(airquality_bms$Ozone, k = 10, list = TRUE)

# Initialization  list for the results
bma_results <- list()

# The loop over each fold
for(i in seq_along(folds)) {
  # The data spliting procedure ( training and test )
  training_set <- airquality_bms[-folds[[i]], ]
  test_set <- airquality_bms[folds[[i]], ]

  # BMA model fit on the training set
  bma_fit <- bms(training_set, mprior = "uniform", g = "UIP")

  # Prediction for the test
  test_set_no_response <- test_set[, !names(test_set) %in% "Ozone", drop =
  ↪  FALSE]
  predictions <- predict(bma_fit, newdata = test_set_no_response)

  # The results
  bma_results[[i]] <- list(
    model = bma_fit,
    predictions = predictions,
    true_values = test_set$Ozone
```

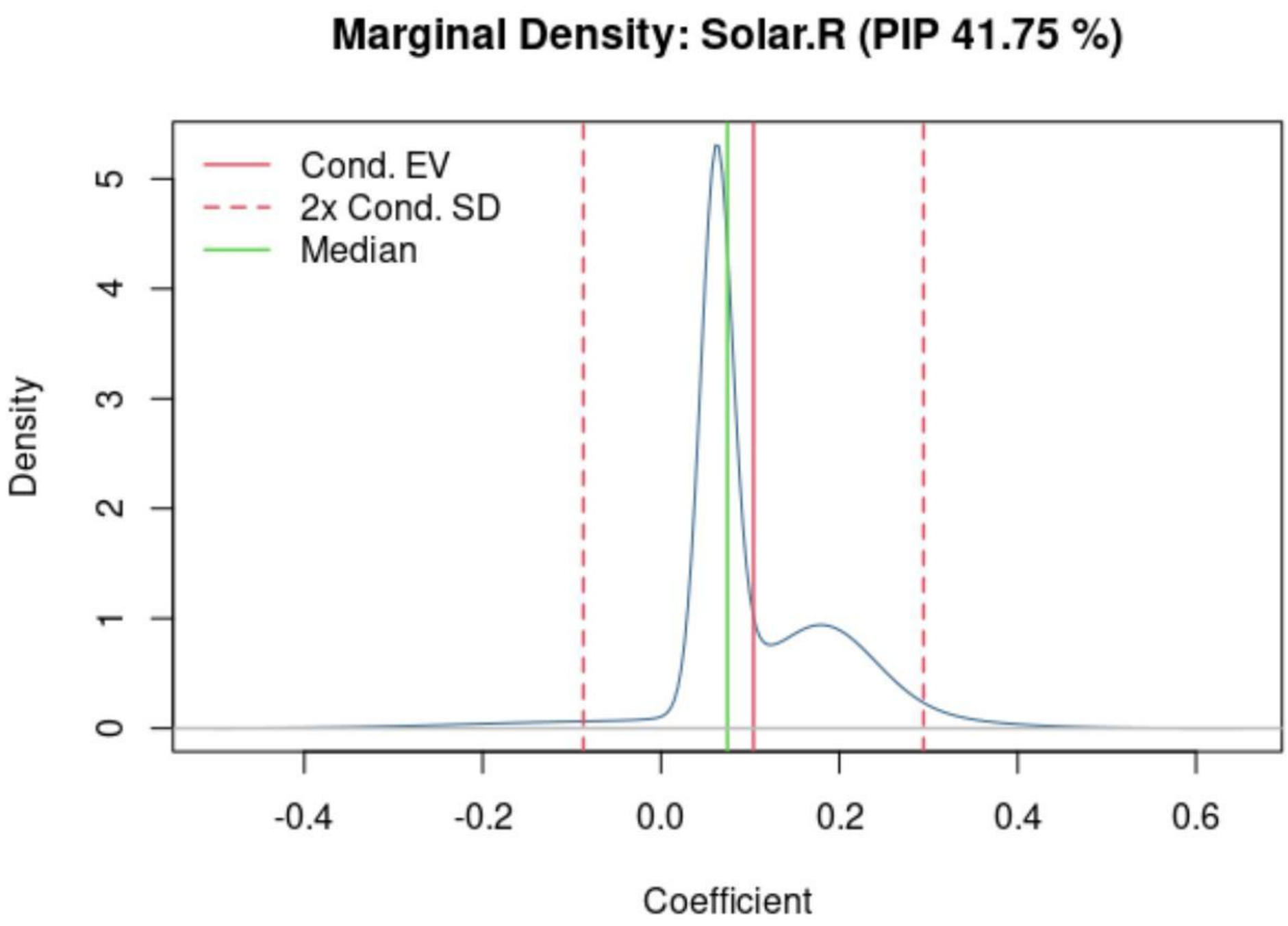


Figure 1.9: Marginal Density plot using uniform Solar.R

```
  )
}

# RMSE for each fold
rmse <- sapply(bma_results, function(res) {
  sqrt(mean((res$predictions - res$true_values)^2))
})

# The average RMSE across all folds
mean_rmse <- mean(rmse)

> print(mean_rmse)
[1] 19.0569
```

To procure the most parsimonious model

Using the 10 fold cross-validation scheme we can decide on the best out of sample procedure among the **birth-death mcmc procedure under our pre-specified priors, the Bric and the g=5** (Alternative fixed Zellner's g-prior) bms procedures to produce the most parsimonious model. We can decide based on the average RMSE results (There are other ways to decide e.g. decide to continue using the procedure that produced the model with the minimum RMSE).

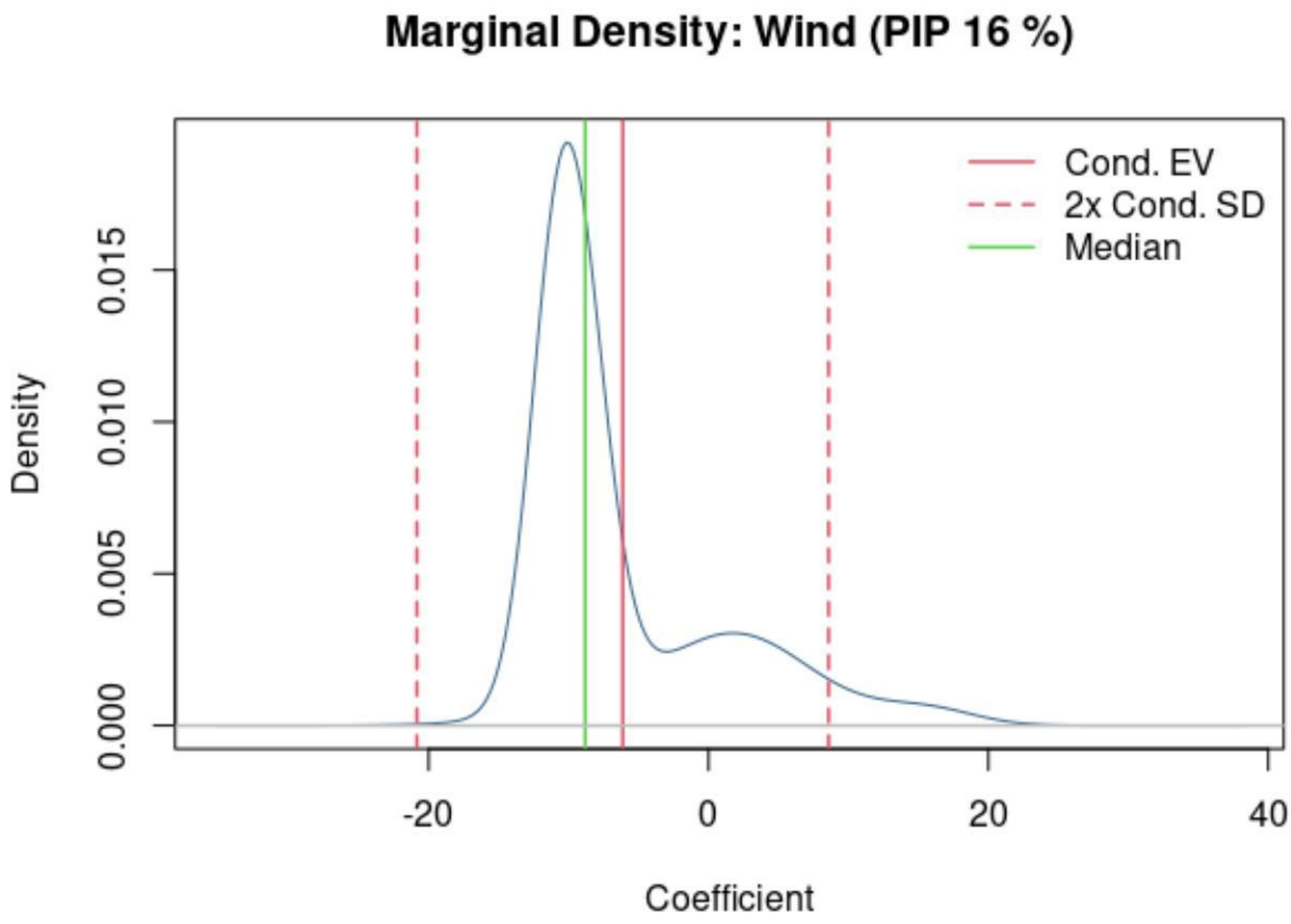


Figure 1.10: Marginal Density plot using uniform Wind

```
### 10 fold CV:

library(BMS)
library(caret)

# The 10-fold cross-validation indices
set.seed(123)
folds <- createFolds(airquality_bms$Ozone, k = 10, list = TRUE)

# Initialization for the lists of results
bma_results_UIP <- list()
bma_results_Fixed <- list()
bma_results_Random <- list()
bma_results_Bric <- list()
bma_results_Alternative <- list()

# The loop over each fold
for(i in seq_along(folds)) {
  # The data spliting procedure ( training and test )
  training_set <- airquality_bms[-folds[[i]], ]
  test_set <- airquality_bms[folds[[i]], ]

  # BMA model fits on the training set
```

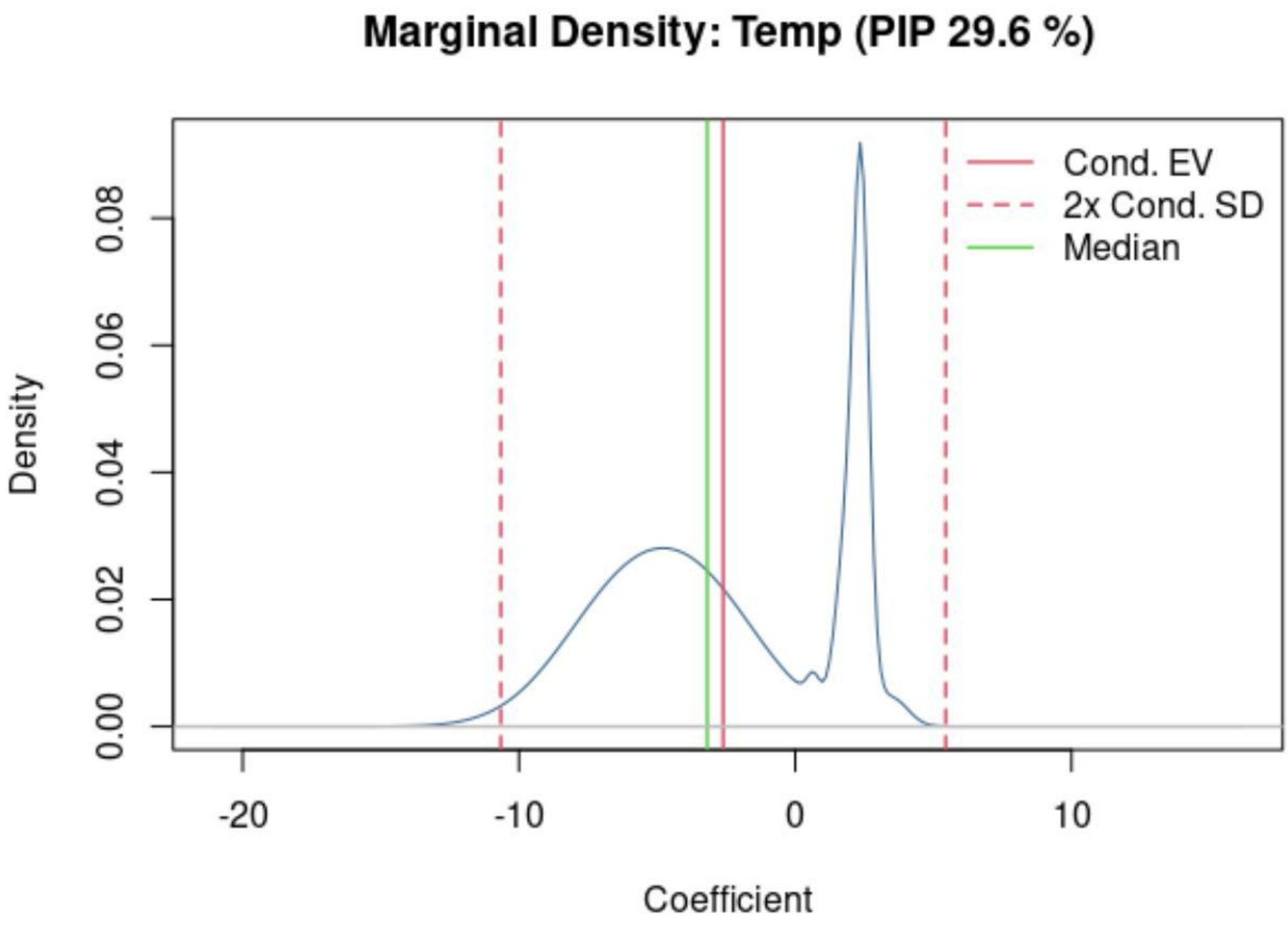


Figure 1.11: Marginal Density plot using uniform Temp

```
bma_fit_UIP <- bms(training_set, mprior = "uniform", g = "UIP", mcmc =
↪  "bd")
bma_fit_Fixed <- bms(training_set, mprior = "fixed", mcmc = "bd")
bma_fit_Random <- bms(training_set, mprior = "random", mcmc = "bd")
bma_fit_Bric <- bms(training_set, mprior = "uniform",g = "BRIC", mcmc =
↪  "bd")
bma_fit_Alternative <- bms(training_set, mprior = "uniform",g =5 , mcmc
↪  = "bd")

# Prediction for the test
test_set_no_response <- test_set[, !names(test_set) %in% "Ozone", drop =
↪  FALSE]
predictions_UIP <- predict(bma_fit_UIP, newdata = test_set_no_response)
predictions_Fixed <- predict(bma_fit_Fixed, newdata =
↪  test_set_no_response)
predictions_Random <- predict(bma_fit_Random, newdata =
↪  test_set_no_response)
predictions_Bric <- predict(bma_fit_Bric, newdata =
↪  test_set_no_response)
predictions_Alternative <- predict(bma_fit_Alternative, newdata =
↪  test_set_no_response)

#Predictive Densities
```

```
pd_UIP<- pred.density(bma_fit_UIP, newdata = test_set_no_response)
pd_Fixed<- pred.density(bma_fit_Fixed, newdata = test_set_no_response)
pd_Random<- pred.density(bma_fit_Random, newdata = test_set_no_response)
pd_Bric<- pred.density(bma_fit_Bric, newdata = test_set_no_response)
pd_Alternative<- pred.density(bma_fit_Alternative, newdata =
↪  test_set_no_response)

# The results
bma_results_UIP[[i]] <- list(
  model = bma_fit_UIP,
  predictions = predictions_UIP,
  true_values = test_set$Ozone,
  pdensities = pd_UIP
)
bma_results_Fixed[[i]] <- list(
  model = bma_fit_Fixed,
  predictions = predictions_Fixed,
  true_values = test_set$Ozone,
  pdensities = pd_Fixed
)
bma_results_Random[[i]] <- list(
  model = bma_fit_Random,
  predictions = predictions_Random,
  true_values = test_set$Ozone,
  pdensities = pd_Random
  )
  bma_results_Bric[[i]] <- list(
    model = bma_fit_Bric,
    predictions = predictions_Bric,
    true_values = test_set$Ozone,
    pdensities = pd_Bric
  )
  bma_results_Alternative[[i]] <- list(
    model = bma_fit_Alternative,
    predictions = predictions_Alternative,
    true_values = test_set$Ozone,
    pdensities = pd_Alternative
  )
}

# RMSE for each fold
rmse_UIP <- sapply(bma_results_UIP, function(res) {
  sqrt(mean((res$predictions - res$true_values)^2))
})
rmse_Fixed <- sapply(bma_results_Fixed, function(res) {
  sqrt(mean((res$predictions - res$true_values)^2))
})
rmse_Random <- sapply(bma_results_Random, function(res) {
  sqrt(mean((res$predictions - res$true_values)^2))
})
rmse_Bric <- sapply(bma_results_Bric, function(res) {
```

```
  sqrt(mean((res$predictions - res$true_values)^2))
})
rmse_Alternative <- sapply(bma_results_Alternative, function(res) {
  sqrt(mean((res$predictions - res$true_values)^2))
})


# The average RMSE across all folds
mean_rmse_UIP <- mean(rmse_UIP)
mean_rmse_Fixed <- mean(rmse_Fixed)
mean_rmse_Random <- mean(rmse_Random)
mean_rmse_Bric <- mean(rmse_Bric)
mean_rmse_Alternative <- mean(rmse_Alternative)
```

with results

```
> print(mean_rmse_UIP)
[1] 19.07014
> print(mean_rmse_Fixed)
[1] 19.01815
> print(mean_rmse_Random)
[1] 19.04674
> print(mean_rmse_Bric)
[1] 19.02298
> print(mean_rmse_Alternative)
[1] 18.87087
```

Based on these results the Alternative is better, though the results vary in every run as it used to be the Fixed and Bric procedures with the smallest value of **RMSE**, while based on the minimum rmse the Alternative g-Prior seems to conquer all as seen below (in any run).

```
> min(rmse_UIP)
[1] 11.90429
> min(rmse_Fixed)
[1] 11.84034
> min(rmse_Random)
[1] 11.81184
> min(rmse_Bric)
[1] 11.84127
> min(rmse_Alternative)
[1] 11.51756
```

Though these results could be Altered in every run in all times I ran the code the best results came from the same models with the model size varying between 4 and 7 for the mean value of the posterior model size. An interesting discovery was made by comparing the cases where the mean of the posterior model size distribution was 7, where the correlation in the posterior Model probabilities varied between 0.49 and 0.65, with the cases where the mean of the posterior model size probability was 4 or

5, where the correlation was high varying between 0.8- 0.95. The conclusion is that as the mean of the posterior model size distribution drops the correlation increases together with the variance of PMP. Thus the most parsimonious model was those of size 5 or 4 see (1.12).

The Predictive densities

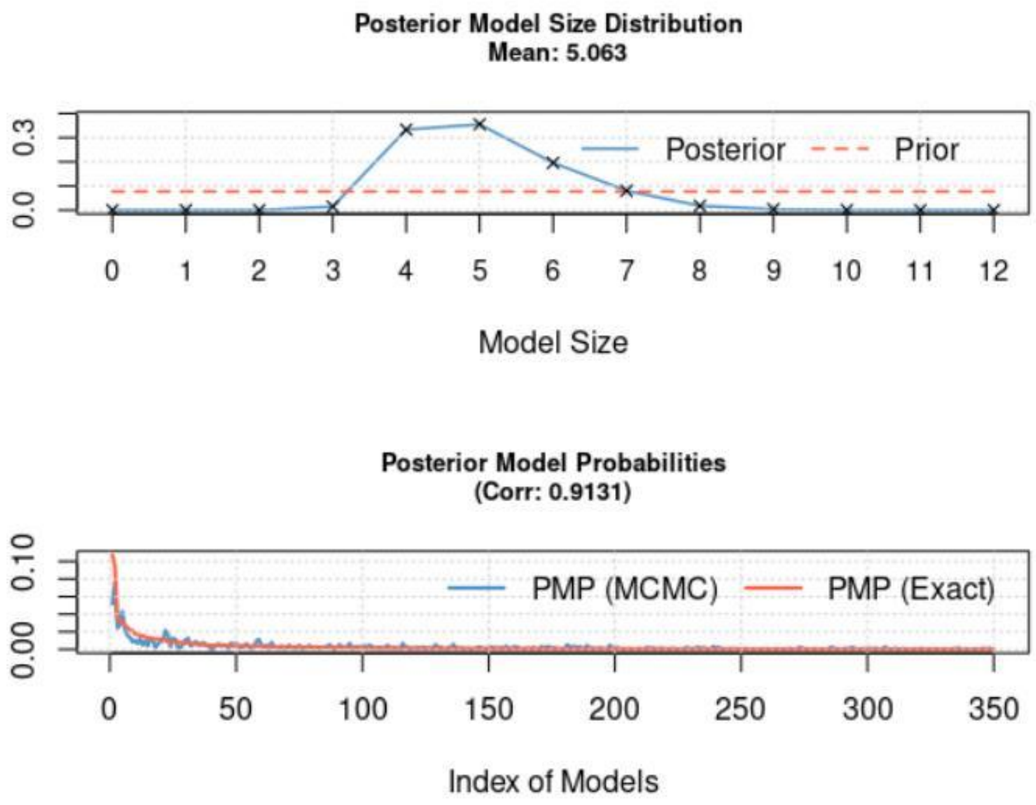
Using a small code insertion in the loop for the Predictive densities I was allowed to access their plots. As all of them plotted would be very time-consuming and of no-insight value, thus you will find below only the plots of the distributions with the minimun **RMSE** for each prior (the code can be found below).

```
# Finding the index of the model with minimum RMSE
min_index_UIP <- which.min(rmse_UIP)
min_index_Fixed <- which.min(rmse_Fixed)
min_index_Random <- which.min(rmse_Random)
min_index_Bric <- which.min(rmse_Bric)
min_index_Alternative <- which.min(rmse_Alternative)

# Accessing the predictive densities
pd_UIP_min <- bma_results_UIP[[min_index_UIP]]$pdensities
pd_Fixed_min <- bma_results_Fixed[[min_index_Fixed]]$pdensities
pd_Random_min <- bma_results_Random[[min_index_Random]]$pdensities
pd_Bric_min <- bma_results_Bric[[min_index_Bric]]$pdensities
pd_Alternative_min <-
↪  bma_results_Alternative[[min_index_Alternative]]$pdensities

# The Plotting the predictive densities one at a time
plot(pd_UIP_min, main = "Predictive Density - UIP")

# plot(pd_Fixed_min, main = "Predictive Density - Fixed")
# plot(pd_Random_min, main = "Predictive Density - Random")
# plot(pd_Bric_min, main = "Predictive Density - BRIC")
# plot(pd_Alternative_min, main = "Predictive Density - Alternative")
```

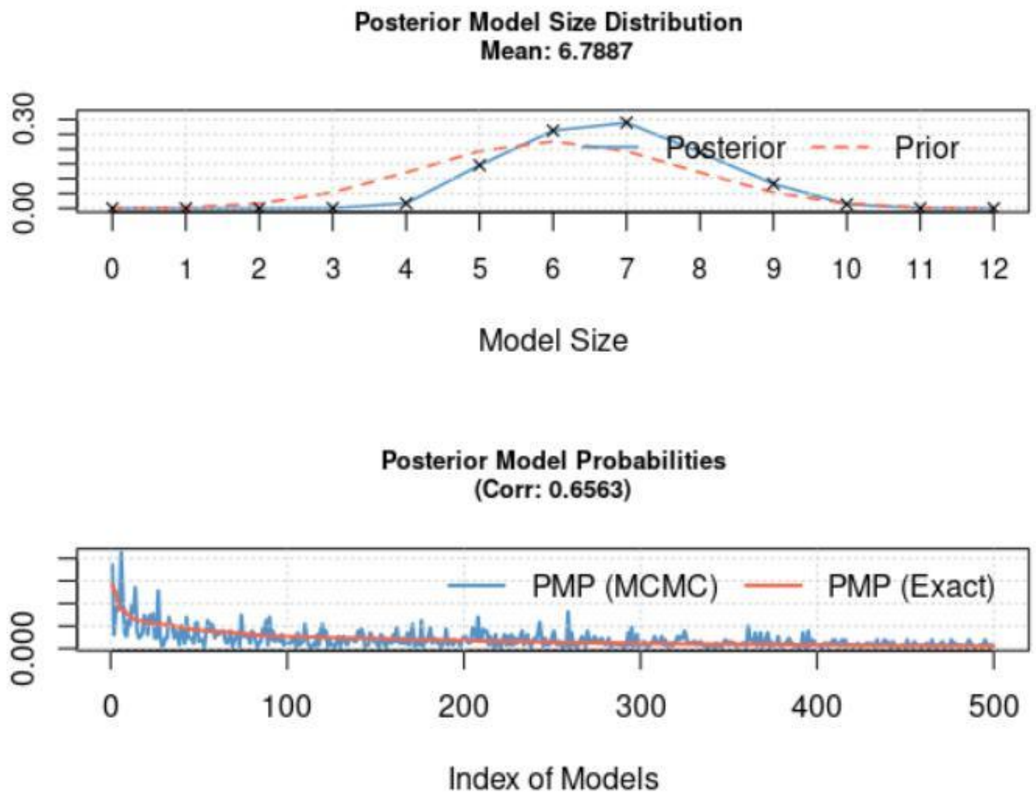
The similarity in the Predictive Densities is . All seem to be the same except the Pred. Density under the Alternative Prior which yield the model with the less RMSE (from all the models under any prior) and seems as the most conservative one (thought by less than 0.005). I am intrigued to see if the plots would deffer more if the magnitude of our variable in question (Ozone) had been transformed into it's logarithmic form (as well as all of the predictors). Given the magnitude of Ozone and the fluctuation of our predictors I am content with our resulted RMSE under any prior and stunned by the unique difference under the Alternative Prior.I will not proceed in finding the Confidence intervals as based on our plots would be far to close in any case

1.3 The Ridge Model

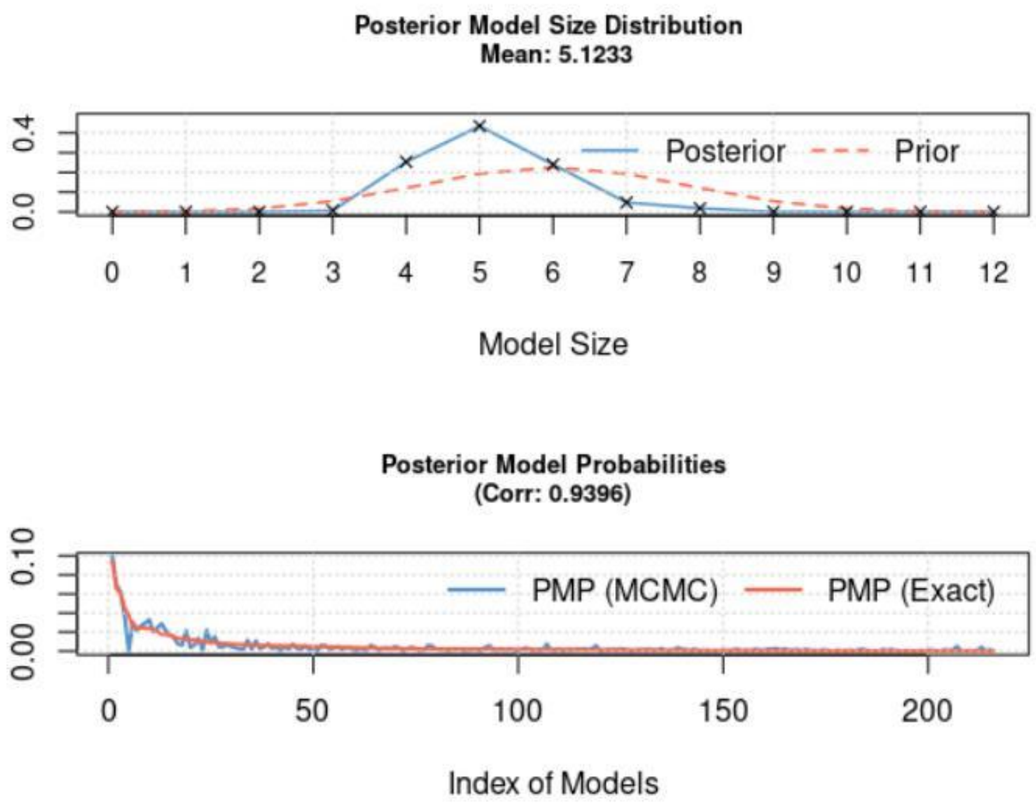
```
library(glmnet)
```



(a) $\mathrm{P}(M_{size} = 4) \approx \mathrm{P}(M_{size} = 5)$



(b) Low corelation



(c)

Figure 1.12: High corellation

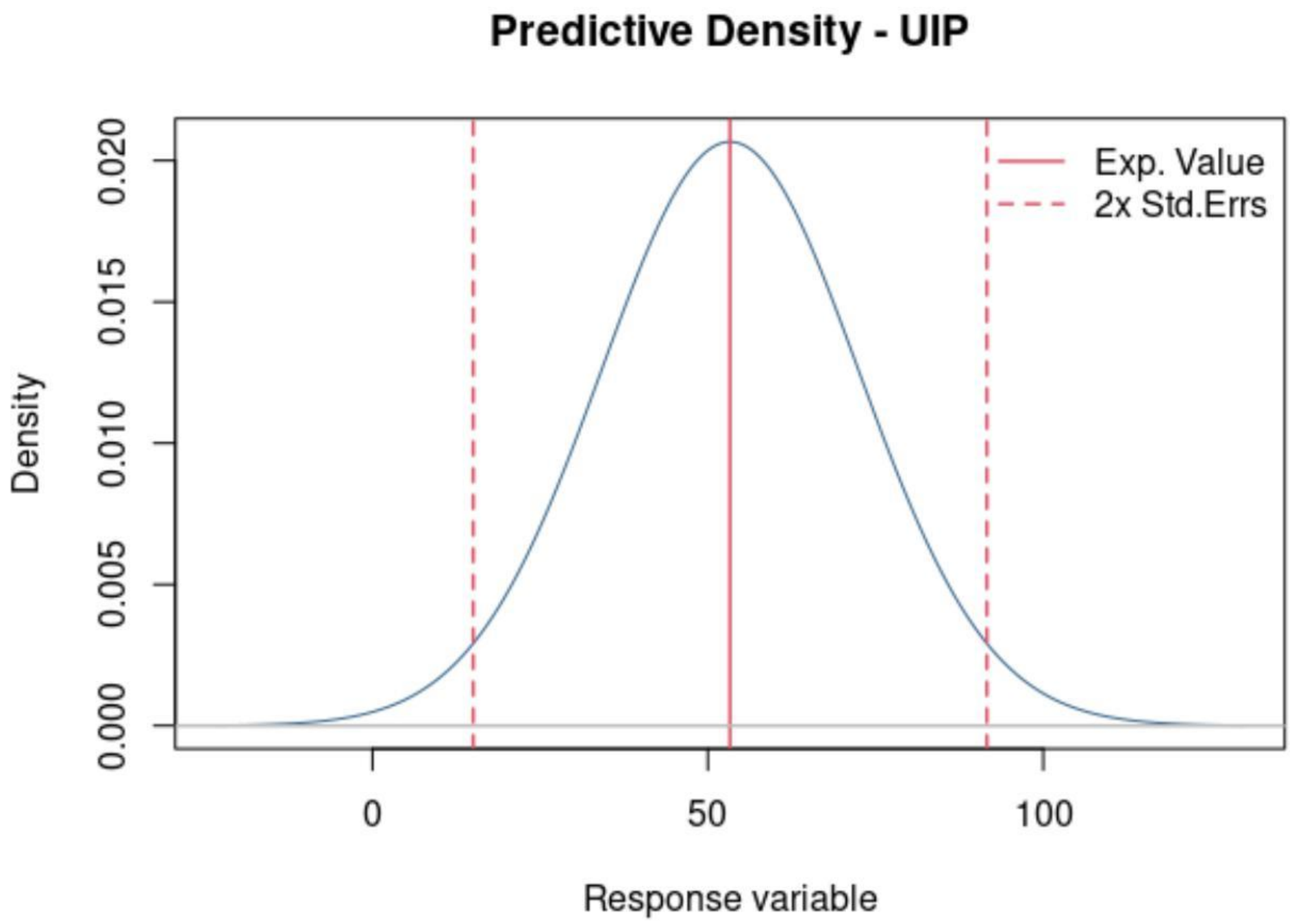


Figure 1.13: Predictive Density with prior UIP

```
# Prepare the data
# Exclude "Ozone", "Day", and "Month" columns to create the predictors
↪  matrix
predictors_matrix <- as.matrix(airquality_clean[,
↪  !(names(airquality_clean) %in% c("Ozone", "Day", "Month"))])

response_vector <- airquality_clean$Ozone

# Fit the Ridge Regression Model by a= 0
ridge_model <- glmnet(predictors_matrix, response_vector, alpha = 0)
```

1.3.1 Cross-Validation for best lambda

To decide upon the best λ value we can use default glmnet cv function.

```
lambda_values <- seq(0.01, 3, length = 1000)

# Cross-Validation to determine the best lambda
cv_ridge <- cv.glmnet(predictors_matrix, response_vector, alpha = 0,
↪  lambda = lambda_values, nfolds = 10)

# The best lambda value
best_lambda <- cv_ridge$lambda.min
```

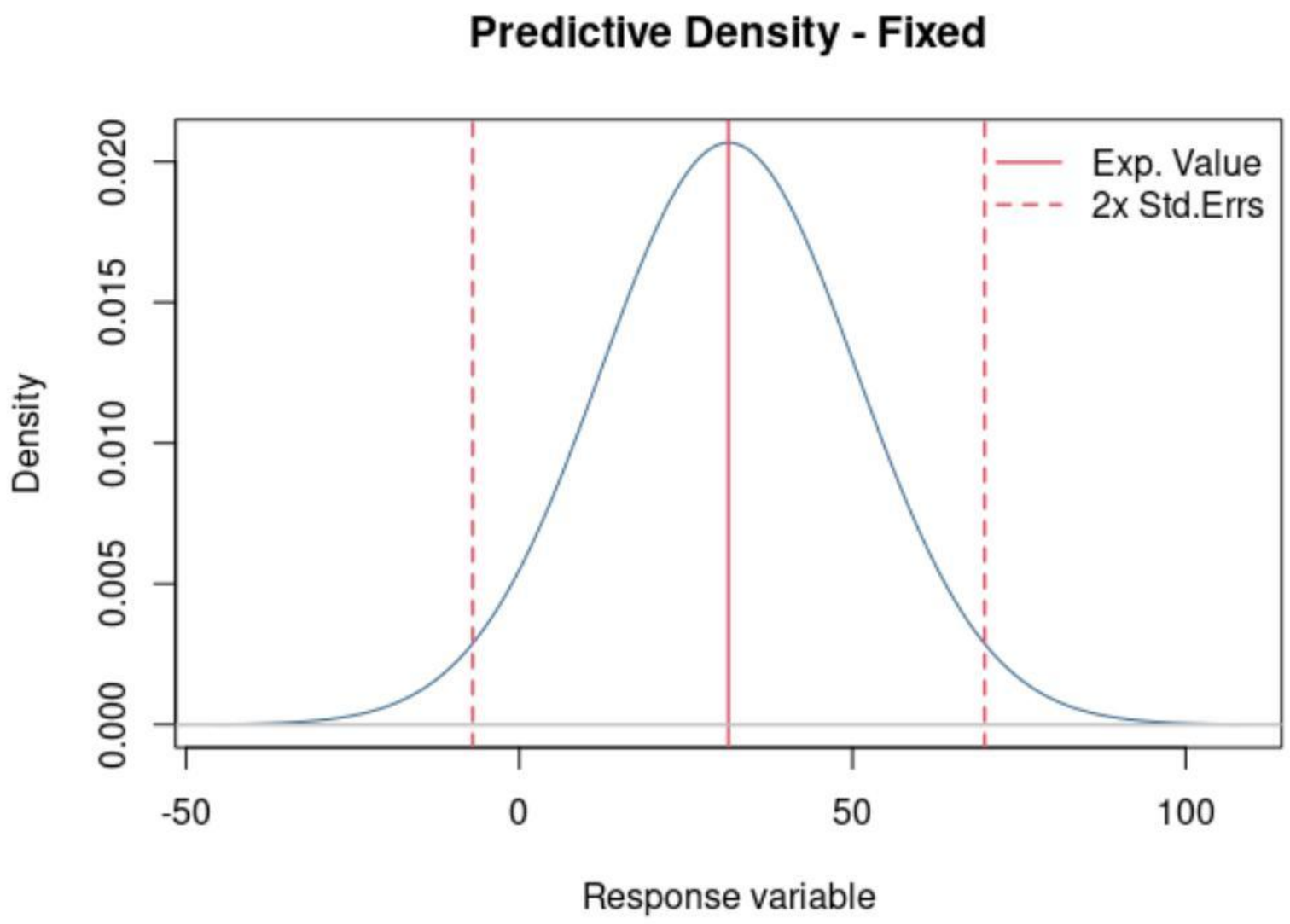


Figure 1.14: Predictive Density with fixed prior

```
print(best_lambda)
```

The best λ was found as **0.2560993** based on which we will calculate the final Ridge model as:

```
# Fit the model using the best lambda
final_ridge_model <- glmnet(predictors_matrix, response_vector, alpha = 0,
↪  lambda = best_lambda)
```

with resulting coefficients:

```
> coef(final_ridge_model)
13 x 1 sparse Matrix of class "dgCMatrix"
                        s0
(Intercept)  6.9790131781
Solar.R      0.1200787984
Wind        -1.2222397630
Temp         0.0528549764
Z1          -0.0334593228
Z2           0.7253684687
Z3          -0.4348750659
Solar.R2    -0.0002820133
Wind2        0.2541885515
Temp2        0.0122799082
SolarR_Wind -0.0078839361
SolarR_Temp  0.0019640846
Wind_Temp   -0.0783210763
```

As we can see the results are not as great as throught the bma. The Ridge regression failed, at it was unable to judge the Z1, Z2 and Z3 predictors as useless (see Z2)

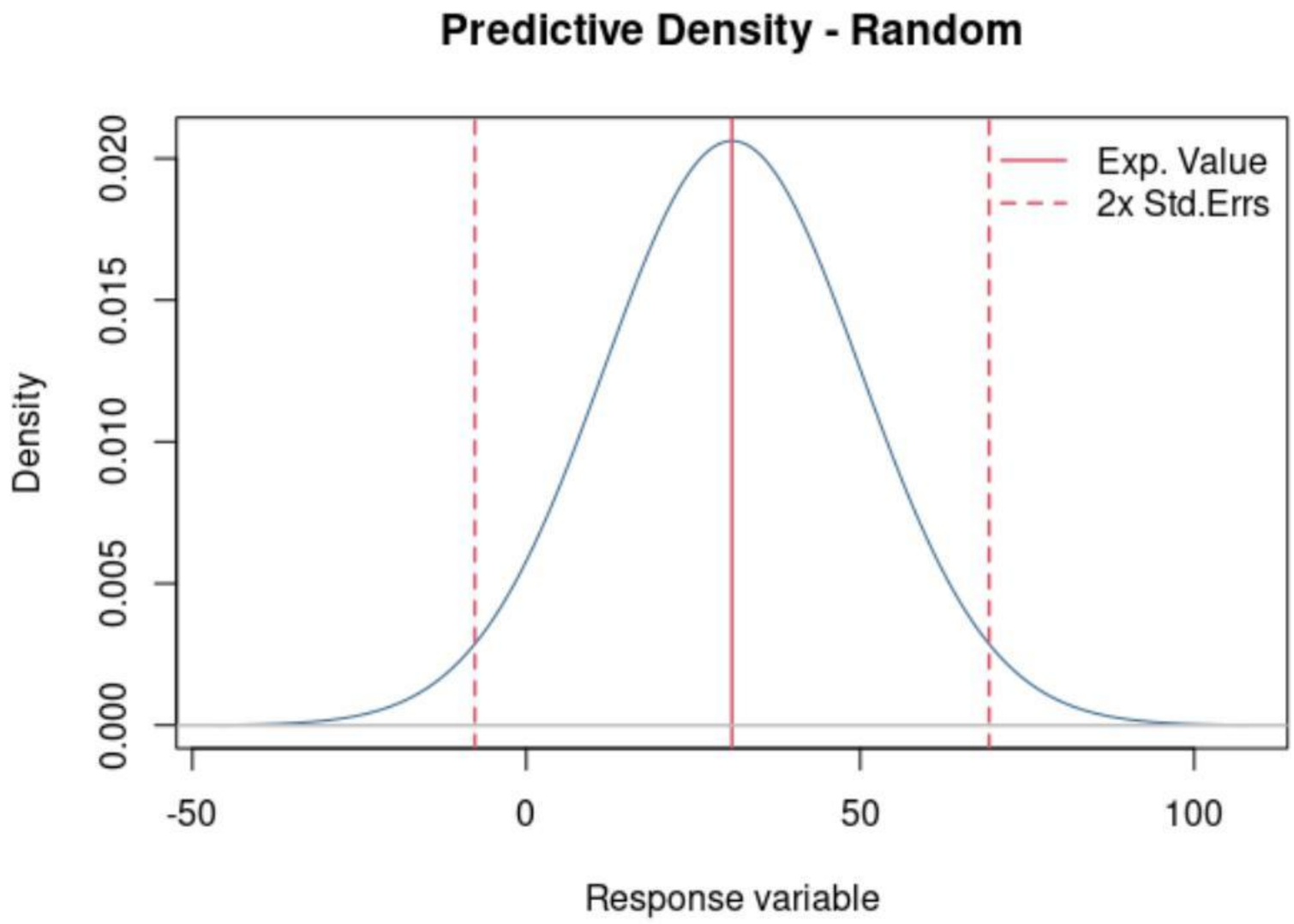


Figure 1.15: Predictive Density with random prior

and provide them with values close to zero, though that was expected as the whole implementation was created based on certain predictors.

Note! As I run repeatedly the below code for different intervals I observed an interesting chaotic effect those intervals have in the procedure of figuring out the best λ.

```
>lambda_values <- seq(0.01, 20, length = 10000)
>cv_ridge <- cv.glmnet(predictors_matrix, response_vector, alpha = 0,
↪  lambda = lambda_values, nfolds = 10)
best_lambda <- cv_ridge$lambda.min
> print(best_lambda)
[1] 0.3978448
```

then agin:

```
lambda_values <- seq(0.01, 2, length = 10000)
>cv_ridge <- cv.glmnet(predictors_matrix, response_vector, alpha = 0,
↪  lambda = lambda_values, nfolds = 10)
best_lambda <- cv_ridge$lambda.min
> print(best_lambda)
[1] 0.5017782
```

furthermore:

```
lambda_values <- seq(0.01, 7, length = 10000)
>cv_ridge <- cv.glmnet(predictors_matrix, response_vector, alpha = 0,
↪  lambda = lambda_values, nfolds = 10)
best_lambda <- cv_ridge$lambda.min
> print(best_lambda)
[1] 0.3462526
```

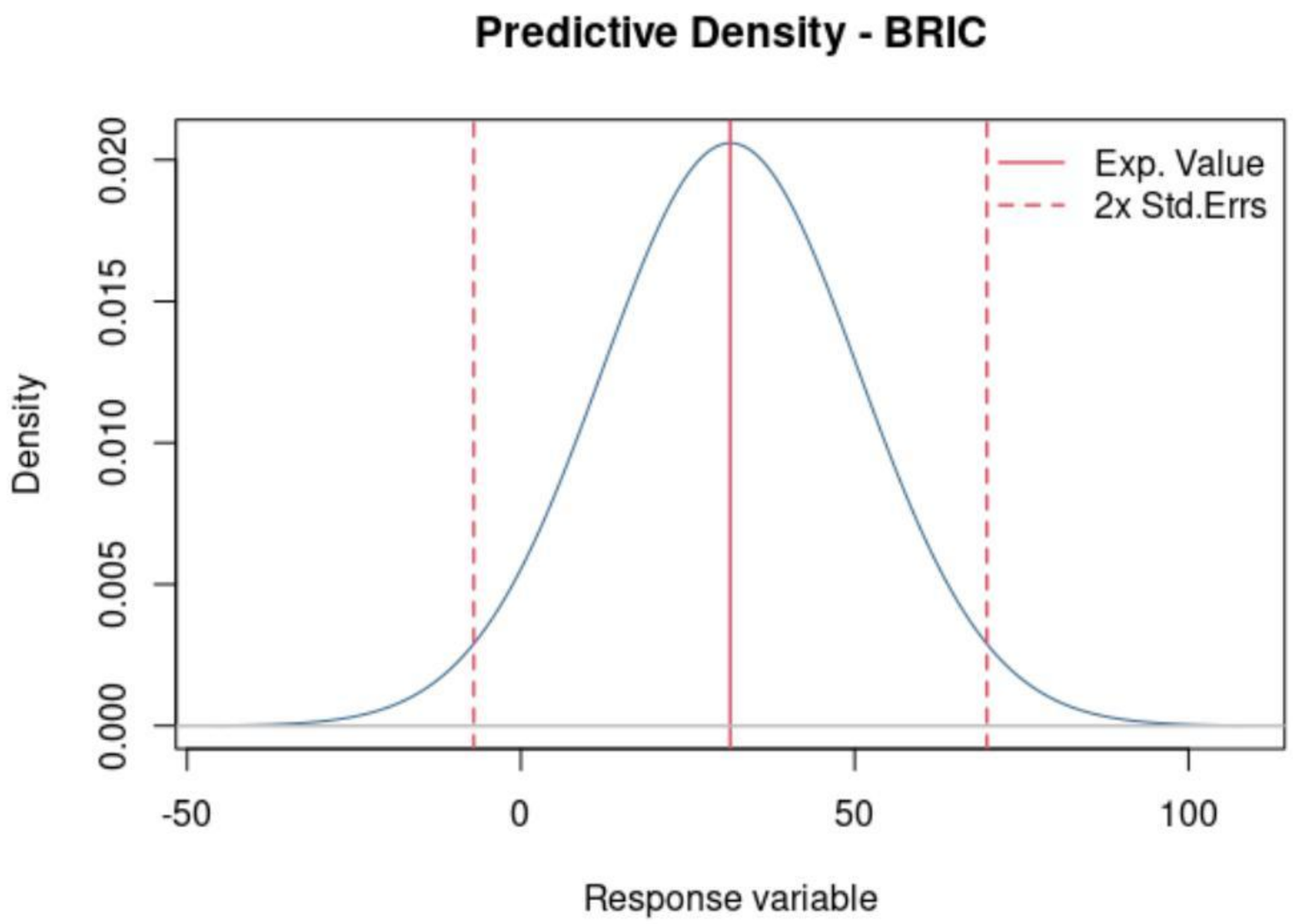


Figure 1.16: Predictive Density with BRIC prior

This is reason enough for me not to trust the outcomes of that procedure, as if you find a good model based on the provided lambda's sequence an other better one (or different good one) could have a λ away from the neighborhood of the previous one, which is troubling as a result. At least it rises the question the optimal sequence selection.

10-fold CV for RMSE

The correct way to do Cross-Validation would have been to include code for Ridge regression within the for loop I done 10 fold CV for the Bayesian (for a better fold compliance), thought I tried I would have to change all the earlier discoveries and revise the whole document accordingly and the mean RMSE results where not even that different, thus I will create a new 10-fold cross validation scheme:

```
# cv for RMSE
# 10-fold cross-validation indices
set.seed(123)
folds <- createFolds(response_vector, k = 10, list = TRUE)

# To store the RMSE for each fold
rmse_values <- numeric(length(folds))

#10-fold cross-validation loop
for(i in seq_along(folds)) {
  # Data split
  train_indices <- unlist(folds[-i])
  test_indices <- unlist(folds[i])
```

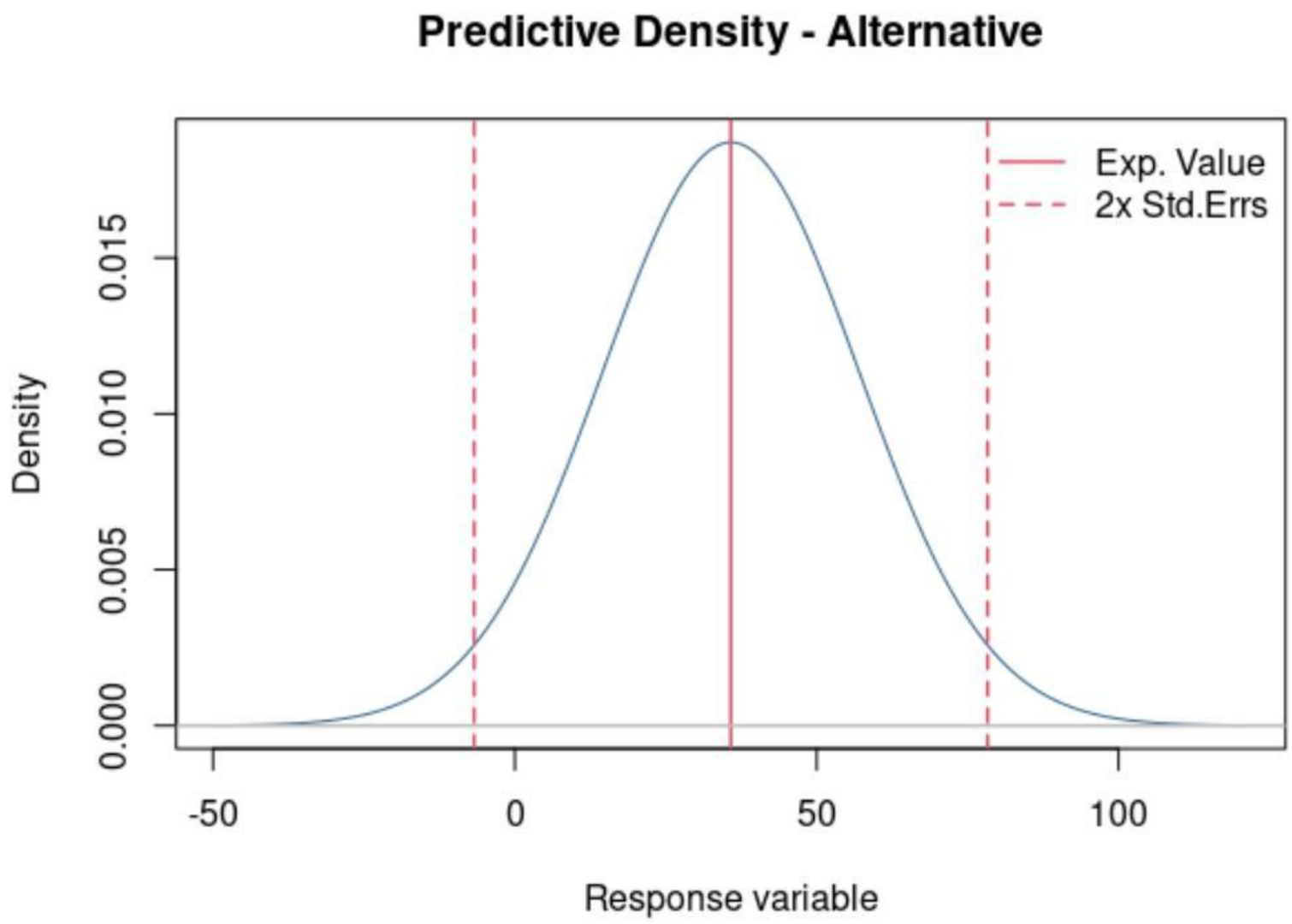


Figure 1.17: Predictive Density under the Alternative g-prior

```
  train_predictors <- predictors_matrix[train_indices, , drop = FALSE]
  train_response <- response_vector[train_indices]

  test_predictors <- predictors_matrix[test_indices, , drop = FALSE]
  test_response <- response_vector[test_indices]

  # The Ridge Regression model on the training set
  ridge_fit <- glmnet(train_predictors, train_response, alpha = 0, lambda
  ↪  = best_lambda)

  # To predict on the test set
  predictions <- predict(ridge_fit, newx = test_predictors, s =
  ↪  best_lambda)

  # The RMSE stored
  rmse_values[i] <- sqrt(mean((test_response - predictions)^2))
}

# The average RMSE across all folds
average_rmse <- mean(rmse_values)
```

The resulting Average RMSE and the minimum RMSE are better than the corresponding ones procured with BMS under different priors, thought we lose explainability as we are unable to see the Predictive Densities:

```
> average_rmse
[1] 18.6239
```

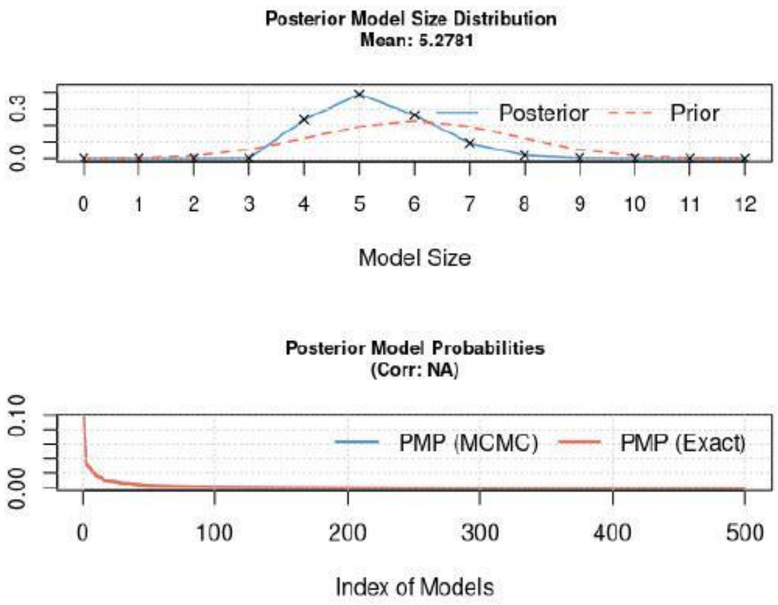
```
> min(rmse_values)
[1] 11.39362
```

1.4 Results and Future Work

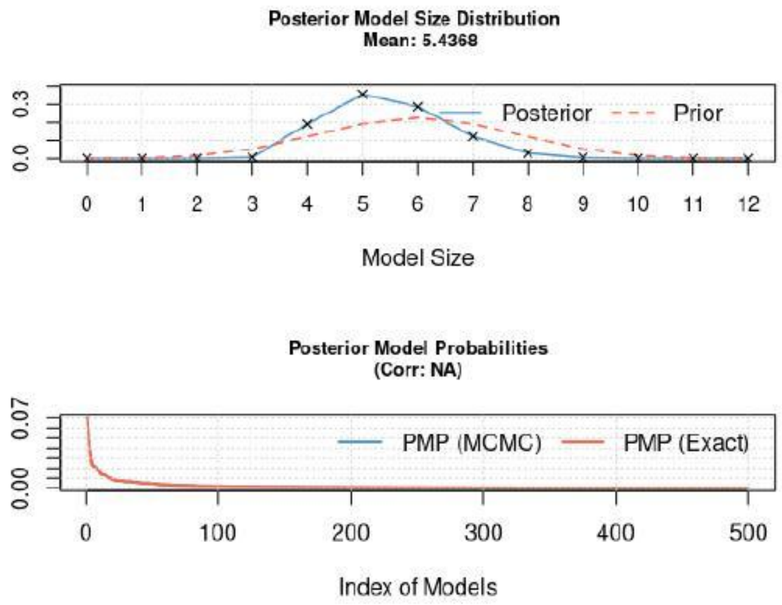
The difference between setting the prior to uniform, or fixed and even random was not substantial for our dataset. The BMA captured the best-performing predictor easily and quickly (best performers based on previous exercises and results seen through VIF, AIC and BIC). Through the BMA you can easily observe measures of interest and plots. Both of my approaches for BMA and Ridge need further improvement. More metrics could be discussed (e.g. MAE). The CV scheme yielded interesting results as we discussed about the relationship between the number of predictors and the correlation of the exact PMP to the evaluated. We saw that the Alternative g-Prior had the best results out of sample based on RMSE. Furthermore we plotted the Predictive Densities of the most parsimonious model under each prior and concluded that the most conservative predictive density had the less RMSE. Regarding Ridge regression's unstable according to λ results there should be more in depth analysis of the phenomenon, especially due to yielding the best results under the Average and minimum RMSE criterion. In conclusion I would like to note the capabilities of excellence for BMS procedure as it allows for prior knowledge to be included in the formulation of the predictive distribution and given the right experts opinion it could potentially surpass all other methods under many criteria having also the best explanatory power.

Appendix A

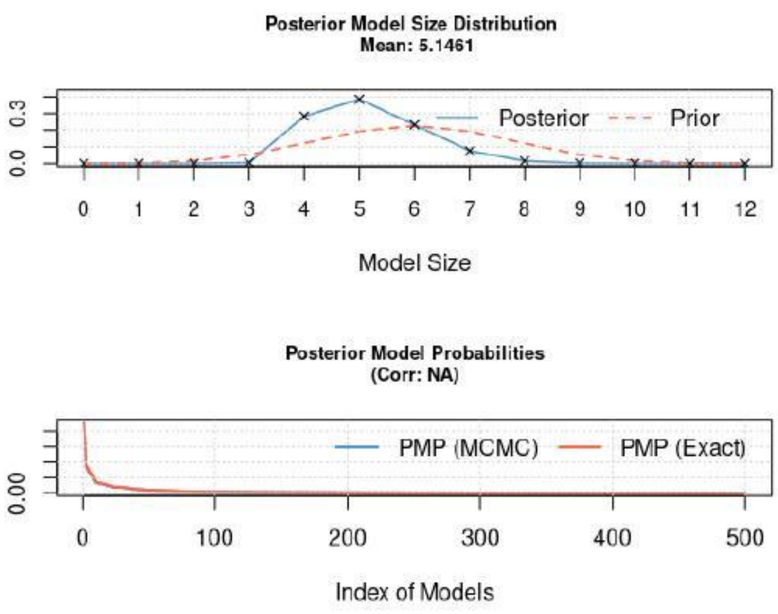
Appendix Title



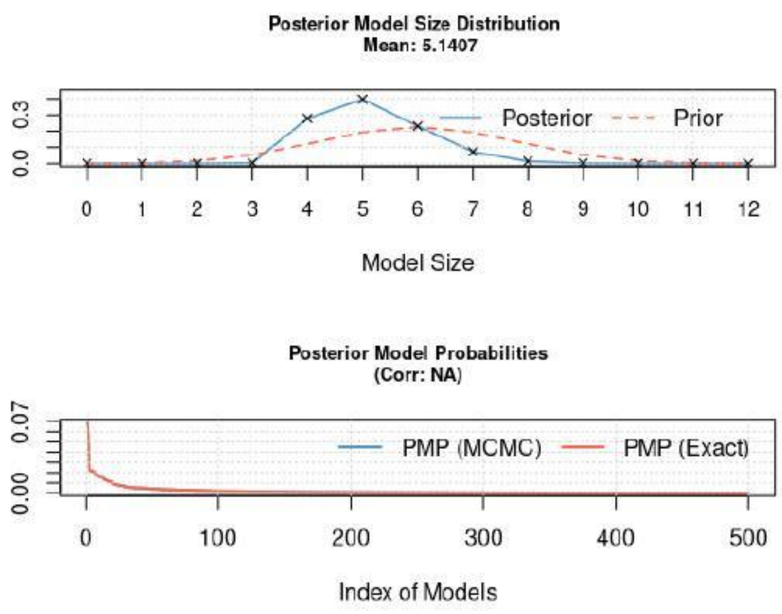
(a) 1_{st} set of folds



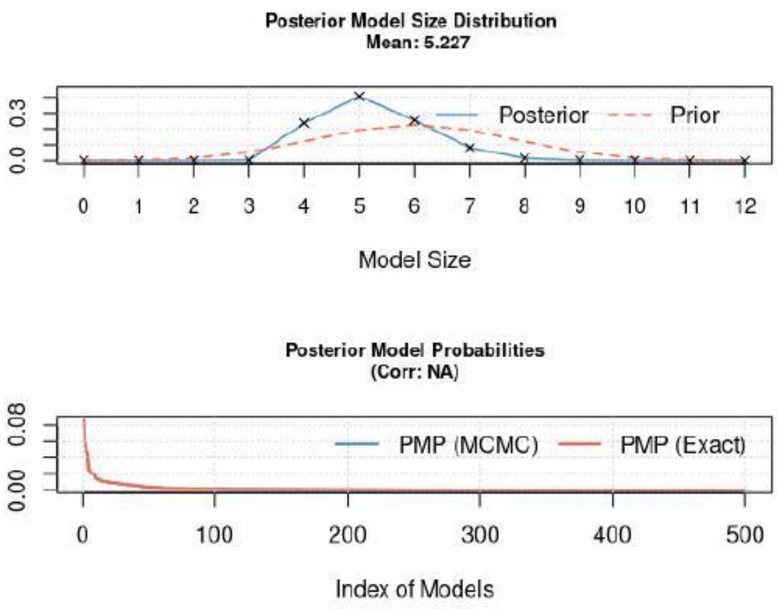
(b) 2_{nd} set of folds



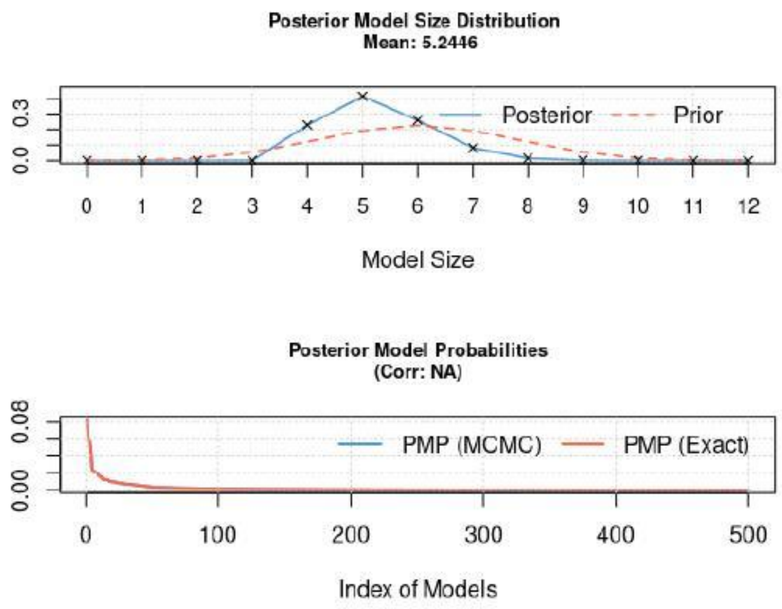
(c) 3_{rd} set of folds



(d) 4_{th} set of folds

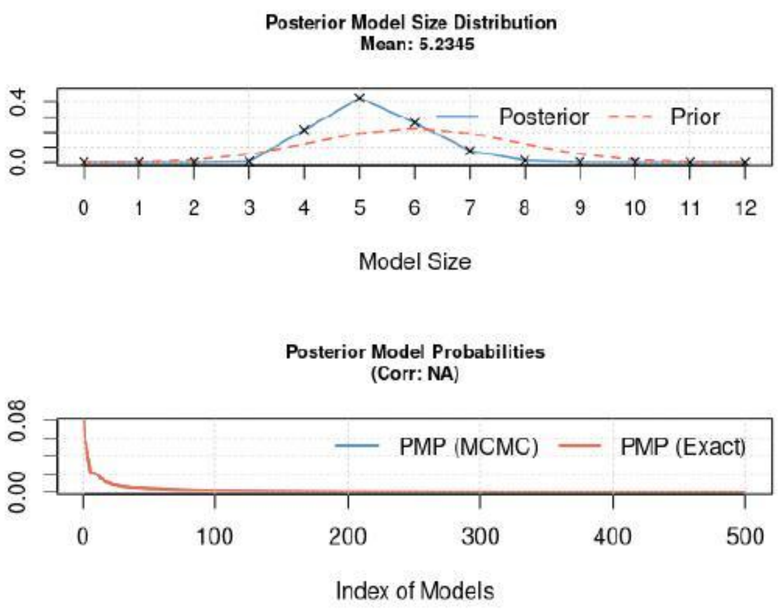


(e) 5_{th} set of folds

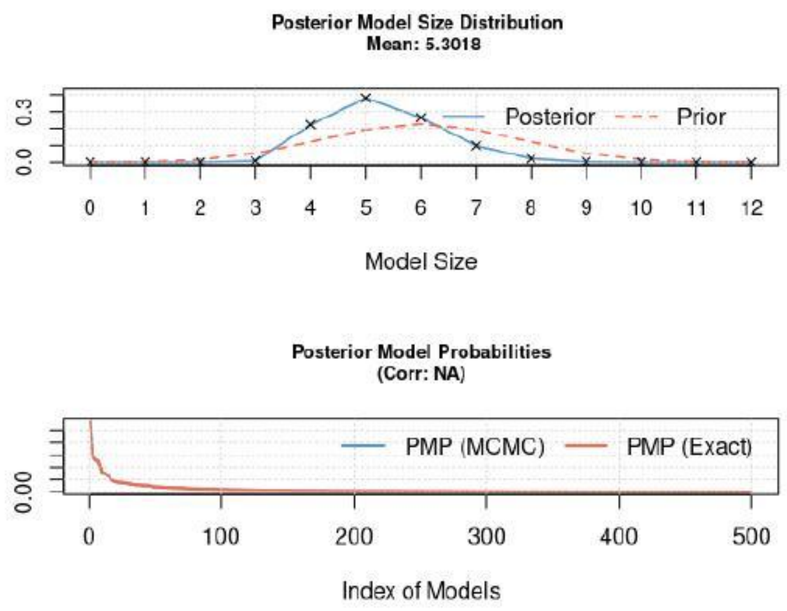


(f) 6_{th} set of folds

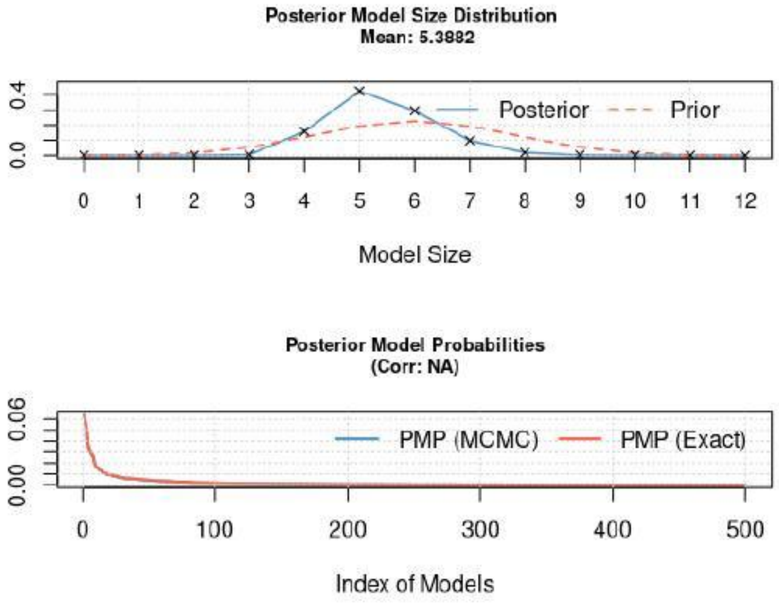
Figure A.1: The g-UIP 10 fold CV scheme (1-6)



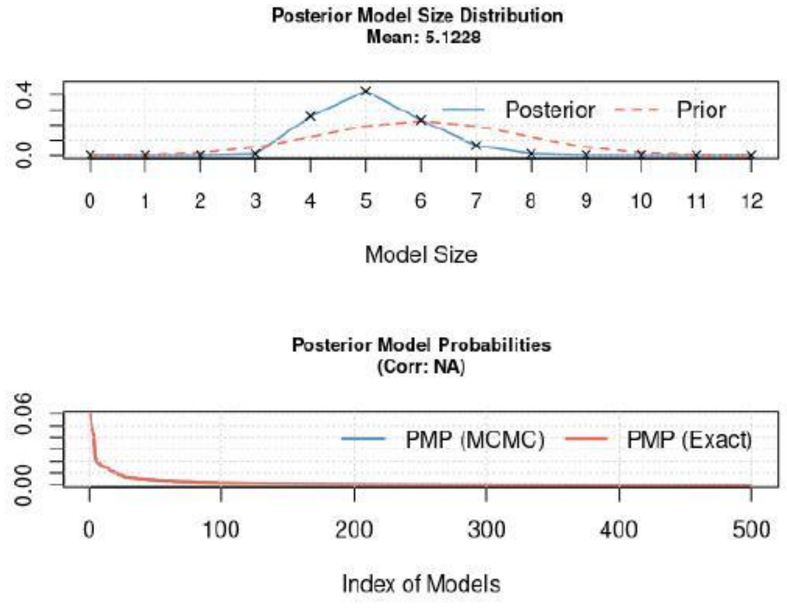
(a) 7_{th} set of folds



(b) 8_{th} set of folds



(c) 9_{th} set of folds



(d) 10_{th} set of folds

Figure A.2: The g-UIP 10 fold CV scheme (7-10)