

Translucent box

Tim Lucas

2020-05-24

Setup

```
# General
library(dplyr)
library(ggplot2)
library(patchwork)
library(doParallel)

# Modelling libraries
library(caret)
library(lime)
library(pdp)
library(ICEbox)
library(iml)
library(elasticnet)
library(INLA)

library(palettetown)

# Phylogenetic libraries.
library(ape)
library(caper)

source('helpers.R')

set.seed(100)
```

The data

First we need to read in the data.

```
p <- read.table(file = 'http://esapubs.org/archive/ecol/E090/184/PanTHERIA_1-0_WR05_Aug2008.txt',
  header = TRUE, sep = "\t", na.strings = c("-999", "-999.00"))
```

```
names(p)
```

## [1] "MSW05_Order"	"MSW05_Family"	"MSW05_Genus"	"MSW05"
## [6] "X1.1_ActivityCycle"	"X5.1_AdultBodyMass_g"	"X8.1_AdultForearmLen_mm"	"X13.1"
## [11] "X3.1_AgeatFirstBirth_d"	"X18.1_BasalMetRate_mL02hr"	"X5.2_BasalMetRateMass_g"	"X6.1_"
## [16] "X9.1_GestationLen_d"	"X12.1_HabitatBreadth"	"X22.1_HomeRange_km2"	"X22.2"
## [21] "X15.1_LitterSize"	"X16.1_LittersPerYear"	"X17.1_MaxLongevity_m"	"X5.3_"
## [26] "X21.1_PopulationDensity_n.km2"	"X10.1_PopulationGrpSize"	"X23.1_SexualMaturityAge_d"	"X10.2"
## [31] "X12.2_Terrestriality"	"X6.2_TrophicLevel"	"X25.1_WeaningAge_d"	"X5.4_"
## [36] "References"	"X5.5_AdultBodyMass_g_EXT"	"X16.2_LittersPerYear_EXT"	"X5.6_"
## [41] "X26.1_GR_Area_km2"	"X26.2_GR_MaxLat_dd"	"X26.3_GR_MinLat_dd"	"X26.4"
## [46] "X26.6_GR_MinLong_dd"	"X26.7_GR_MidRangeLong_dd"	"X27.1_HuPopDen_Min_n.km2"	"X27.2"
## [51] "X27.4_HuPopDen_Change"	"X28.1_Precip_Mean_mm"	"X28.2_Temp_Mean_01degC"	"X30.1"

```

sapply(p, function(x) mean(is.na(x))) %>% sort

##          MSW05_Order      MSW05_Family      MSW05_Genus
##          0.0000000      0.0000000      0.0000000
##          X26.2_GR_MaxLat_dd X26.3_GR_MinLat_dd X26.4_GR_MidRangeLat_dd X26.5_G
##          0.1381093          0.1381093          0.1381093
##          X27.1_HuPopDen_Min_n.km2 X27.2_HuPopDen_Mean_n.km2 X27.3_HuPopDen_5p_n.km2 X27.4_HuP
##          0.1381093          0.1381093          0.1381093
##          X30.1_AET_Mean_mm      X30.2_PET_Mean_mm      References      X5.1_Adu
##          0.2051329          0.2051329          0.2307976
##          X15.1_LitterSize      X6.1_DietBreadth      X6.2_TrophicLevel      X13.1_AdultHe
##          0.5382201          0.6009970          0.6009970
##          X25.1_WeaningAge_d      X5.3_NeonateBodyMass_g      X23.1_SexualMaturityAge_d      X17.1_Ma
##          0.7850812          0.7996677          0.8059453
##          X16.1_LittersPerYear      X22.1_HomeRange_km2      X10.2_SocialGrpSize      X14.1_Interbir
##          0.8349335          0.8694609          0.8694609
##          X18.1_BasalMetRate_mL02hr      X5.2_BasalMetRateMass_g      X5.4_WeaningBodyMass_g      X2.1_Ageat
##          0.8942024          0.8942024          0.9100812
##          X16.2_LittersPerYear_EXT      X10.1_PopulationGrpSize      X13.2_NeonateHeadBodyLen_mm      X7.1_Di
##          0.9272526          0.9283604          0.9582718

sapply(p, class)

##          MSW05_Order      MSW05_Family      MSW05_Genus
##          "factor"          "factor"          "factor"
##          X5.1_AdultBodyMass_g      X8.1_AdultForearmLen_mm      X13.1_AdultHeadBodyLen_mm      X2.1_Ageat
##          "numeric"          "numeric"          "numeric"
##          X5.2_BasalMetRateMass_g      X6.1_DietBreadth      X7.1_DispersalAge_d      X9.1_Ge
##          "numeric"          "numeric"          "numeric"
##          X22.2_HomeRange_Indiv_km2      X14.1_InterbirthInterval_d      X15.1_LitterSize      X16.1_Li
##          "numeric"          "numeric"          "numeric"
##          X13.2_NeonateHeadBodyLen_mm      X21.1_PopulationDensity_n.km2      X10.1_PopulationGrpSize      X23.1_SexualM
##          "numeric"          "numeric"          "numeric"
##          X12.2_Terrestriality      X6.2_TrophicLevel      X25.1_WeaningAge_d      X5.4_Weani
##          "numeric"          "integer"          "numeric"
##          X5.5_AdultBodyMass_g_EXT      X16.2_LittersPerYear_EXT      X5.6_NeonateBodyMass_g_EXT      X5.7_WeaningBo
##          "numeric"          "numeric"          "numeric"
##          X26.3_GR_MinLat_dd      X26.4_GR_MidRangeLat_dd      X26.5_GR_MaxLong_dd      X26.6_G
##          "numeric"          "numeric"          "numeric"
##          X27.2_HuPopDen_Mean_n.km2      X27.3_HuPopDen_5p_n.km2      X27.4_HuPopDen_Change      X28.1_Pr
##          "numeric"          "numeric"          "numeric"
##          X30.2_PET_Mean_mm      "numeric"          "numeric"

dim(p)

```

```
## [1] 5416 55
```

Now we need to choose a variable of interest and make some basic exploratory plots.

```
# Want something with quite a lot of data. Litter size?
```

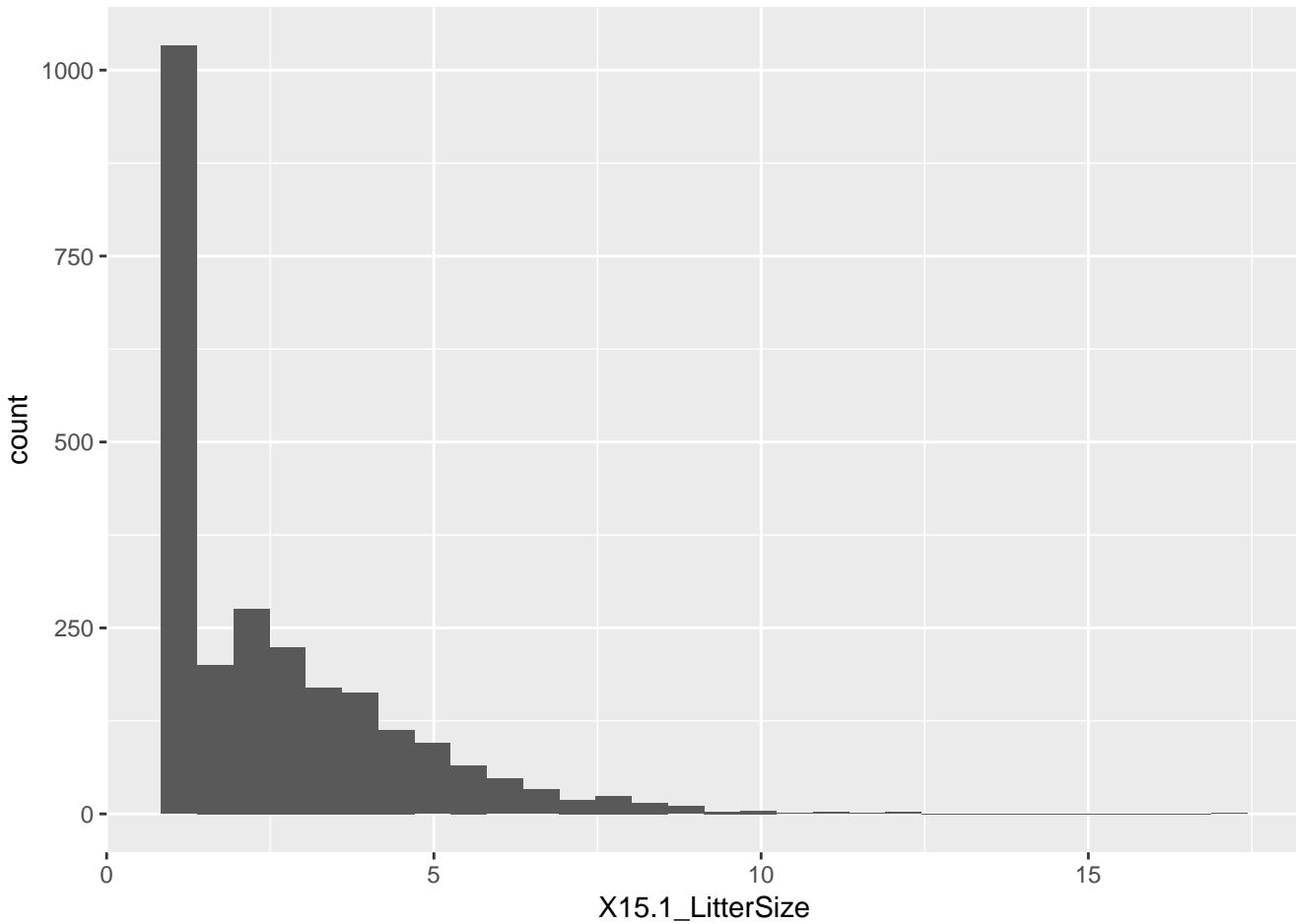
```
sum(!is.na(p$X15.1_LitterSize))
```

```
## [1] 2501
```

```
ggplot(p, aes(X15.1_LitterSize)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 2915 rows containing non-finite values (stat_bin).
```

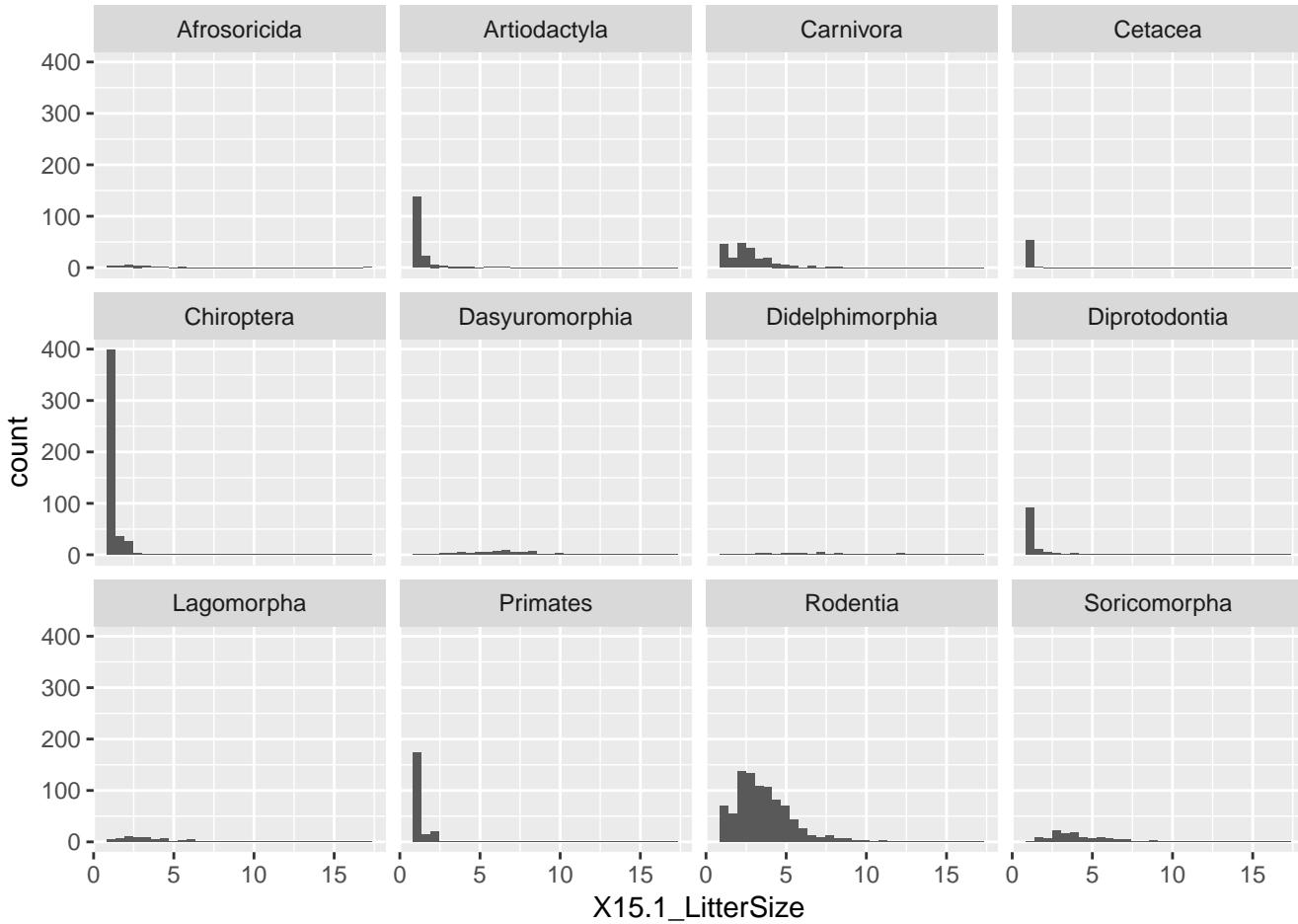


```
p$X15.1_LitterSize %>% summary
##      Min. 1st Qu.   Median     Mean 3rd Qu.    Max.    NA's
##  0.840   1.000   1.940   2.523   3.490  16.890  2915

large_orders <-
  p %>%
  filter(!is.na(sum(!is.na(p$X15.1_LitterSize)))) %>%
  group_by(MSW05_Order) %>%
  count() %>%
  arrange(desc(n)) %>%
  filter(n > 40) %>%
  pull(MSW05_Order)

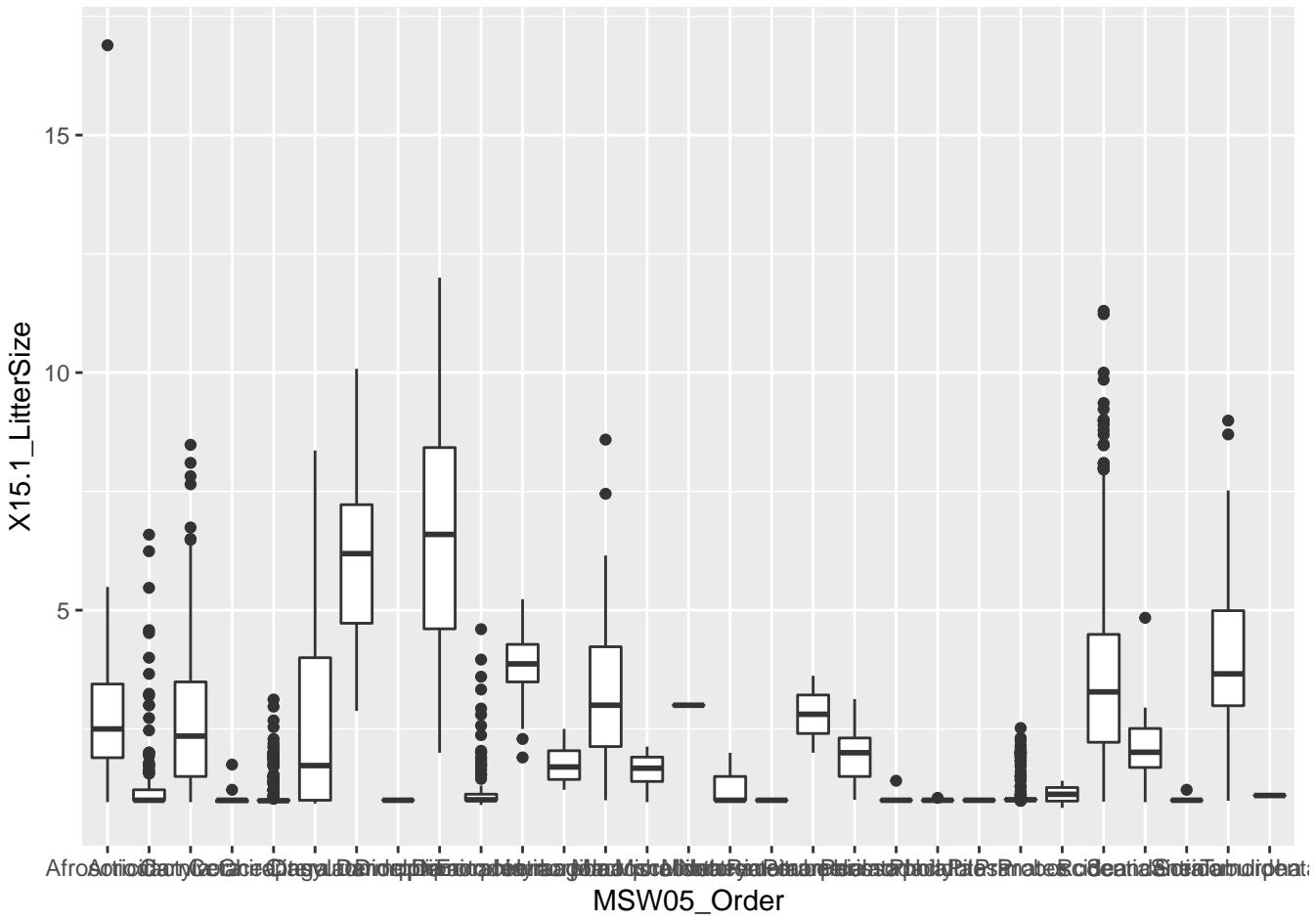
p %>%
  filter(MSW05_Order %in% large_orders) %>%
  ggplot(aes(X15.1_LitterSize)) +
  geom_histogram() +
  facet_wrap(~ MSW05_Order)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 2866 rows containing non-finite values (stat_bin).
```



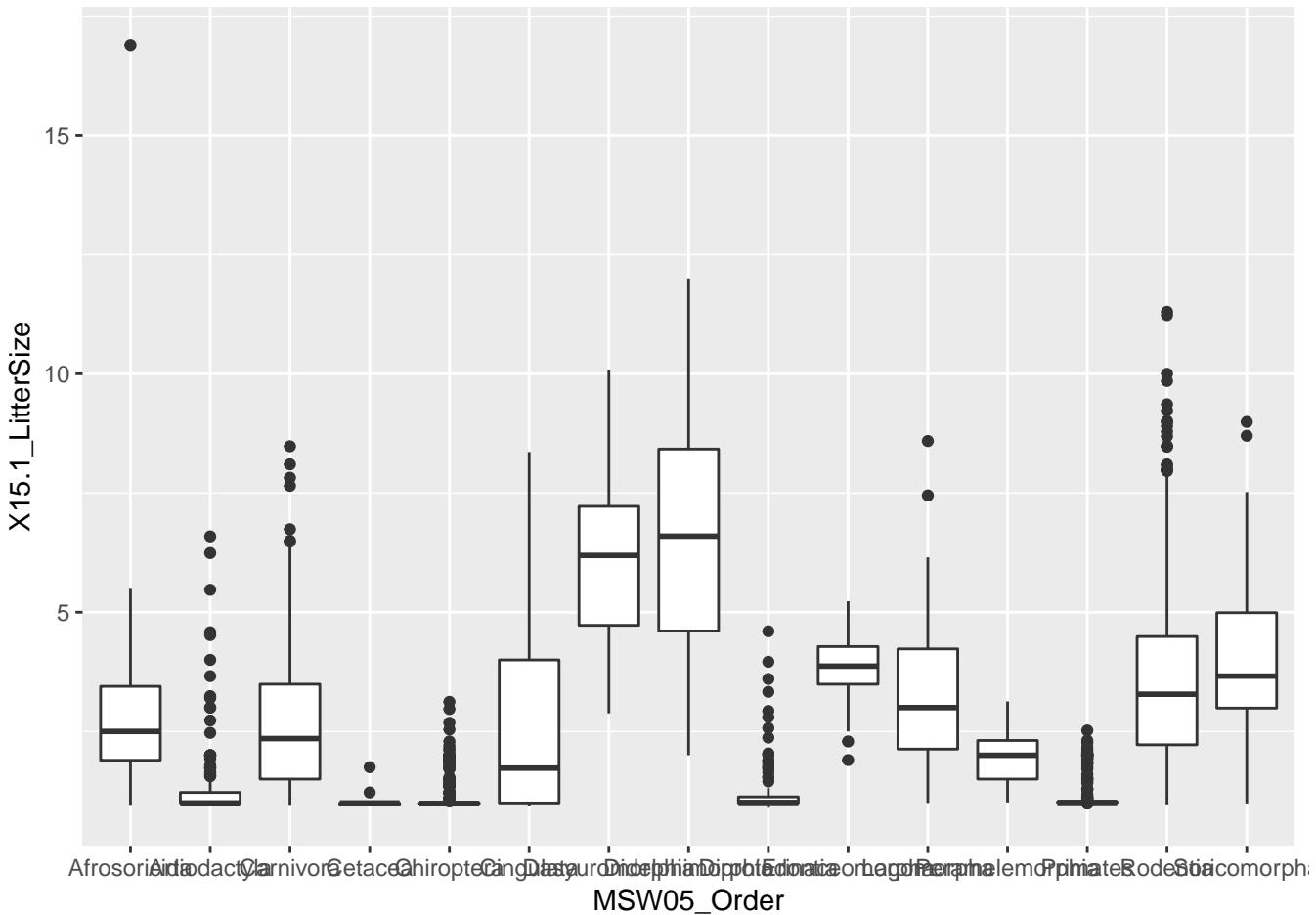
```
ggplot(p, aes(x = MSW05_Order, y = X15.1_LitterSize)) + geom_boxplot()
```

```
## Warning: Removed 2915 rows containing non-finite values (stat_boxplot).
```



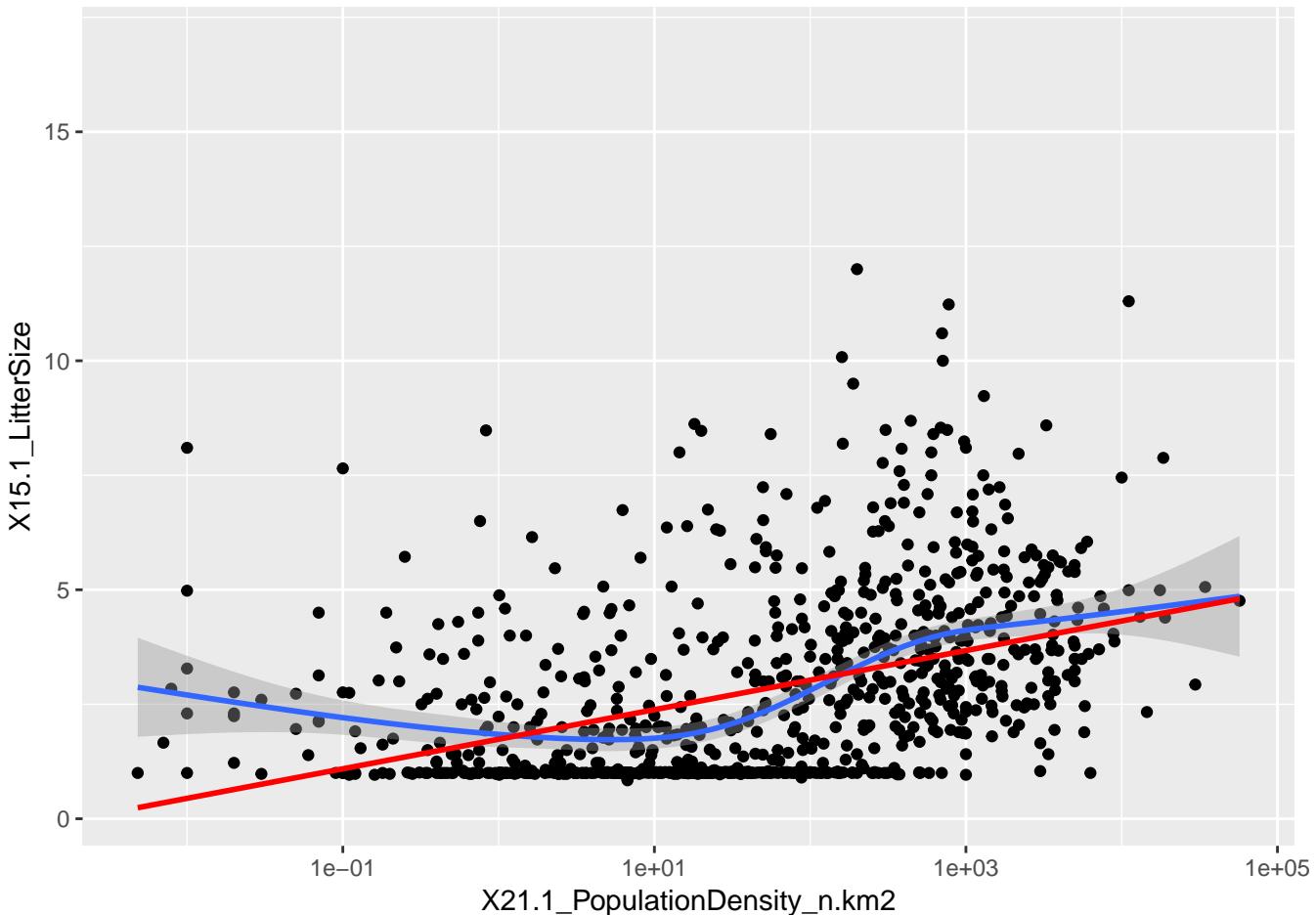
```
p %>%
  group_by(MSW05_Order) %>%
  add_tally %>%
  filter(n > 20) %>%
  ggplot(aes(x = MSW05_Order, y = X15.1_LitterSize)) + geom_boxplot()
```

```
## Warning: Removed 2889 rows containing non-finite values (stat_boxplot).
```



```
## Don't wish to do too many bivariate plots at this point. Going to do a priori variable selection and p value
ggplot(p, aes(x = X21.1_PopulationDensity_n.km2, y = X15.1_LitterSize)) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method = 'lm', se = FALSE, colour = 'red') +
  scale_x_log10()

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## Warning: Removed 4532 rows containing non-finite values (stat_smooth).
## Warning: Removed 4532 rows containing non-finite values (stat_smooth).
## Warning: Removed 4532 rows containing missing values (geom_point).
```



Some data cleaning

```
# Remove NAs in response and response where litter size is less than one (doesn't make sense).
p <- p %>%
  filter(!is.na(X15.1_LitterSize)) %>%
  filter(X15.1_LitterSize >= 1) %>%
  mutate(y = log1p(X15.1_LitterSize)) %>%
  dplyr::select(-X15.1_LitterSize, -References, -X24.1_TeatNumber)

p_notaxa <- p %>%
  dplyr::select(-contains('MSW05'))

preprocesses <- preProcess(p, method = 'medianImpute')
p_impute <- predict(preprocesses, p_notaxa)
```

Get phylogeny data.

read in phylogeny data.

```
# Read in trees
tree <- read.nexus('https://onlinelibrary.wiley.com/action/downloadSupplement?doi=10.1111%2Fj.1461-0248.2009.00740.x')

# Select best supported tree
tree <- tree[[1]]
tree$tip.label <- gsub('_', ' ', tree$tip.label)

# Check if species are available.
mean(p$MSW05_Binomial %in% tree$tip.label)
```

```

## [1] 0.9893813

in_phylo <- p$MSW05_Binomial %in% tree$tip.label

# Remove data that is not in the phylogeny.

p <- p %>% filter(in_phylo)
p_impute <- p_impute %>% filter(in_phylo)

```

Data summary

```

dim(p_impute)

## [1] 2143   48
names(p_impute)

## [1] "X1.1_ActivityCycle"          "X5.1_AdultBodyMass_g"
## [3] "X8.1_AdultForearmLen_mm"    "X13.1_AdultHeadBodyLen_mm"
## [5] "X2.1_AgeatEyeOpening_d"      "X3.1_AgeatFirstBirth_d"
## [7] "X18.1_BasalMetRate_mL02hr"   "X5.2_BasalMetRateMass_g"
## [9] "X6.1_DietBreadth"           "X7.1_DispersalAge_d"
## [11] "X9.1_GestationLen_d"         "X12.1_HabitatBreadth"
## [13] "X22.1_HomeRange_km2"        "X22.2_HomeRange_Indiv_km2"
## [15] "X14.1_InterbirthInterval_d"  "X16.1_LittersPerYear"
## [17] "X17.1_MaxLongevity_m"       "X5.3_NeonateBodyMass_g"
## [19] "X13.2_NeonateHeadBodyLen_mm" "X21.1_PopulationDensity_n.km2"
## [21] "X10.1_PopulationGrpSize"    "X23.1_SexualMaturityAge_d"
## [23] "X10.2_SocialGrpSize"        "X12.2_Terrestriality"
## [25] "X6.2_TrophicLevel"          "X25.1_WeaningAge_d"
## [27] "X5.4_WeaningBodyMass_g"      "X13.3_WeaningHeadBodyLen_mm"
## [29] "X5.5_AdultBodyMass_g_EXT"    "X16.2_LittersPerYear_EXT"
## [31] "X5.6_NeonateBodyMass_g_EXT"   "X5.7_WeaningBodyMass_g_EXT"
## [33] "X26.1_GR_Area_km2"          "X26.2_GR_MaxLat_dd"
## [35] "X26.3_GR_MinLat_dd"         "X26.4_GR_MidRangeLat_dd"
## [37] "X26.5_GR_MaxLong_dd"        "X26.6_GR_MinLong_dd"
## [39] "X26.7_GR_MidRangeLong_dd"    "X27.1_HuPopDen_Min_n.km2"
## [41] "X27.2_HuPopDen_Mean_n.km2"   "X27.3_HuPopDen_5p_n.km2"
## [43] "X27.4_HuPopDen_Change"       "X28.1_Precip_Mean_mm"
## [45] "X28.2_Temp_Mean_01degC"      "X30.1_AET_Mean_mm"
## [47] "X30.2_PET_Mean_mm"          "y"

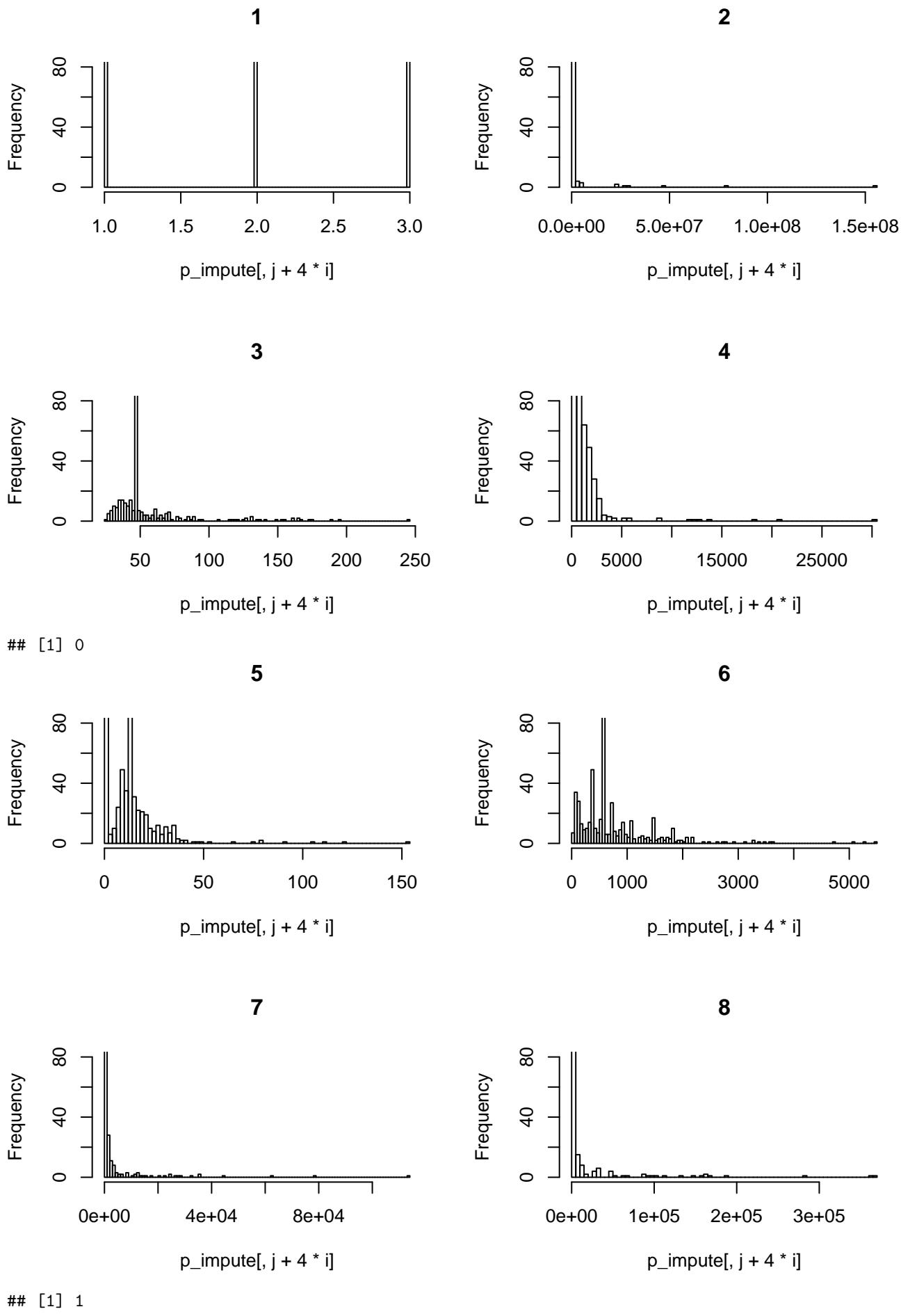
par(mfrow = c(2, 2))

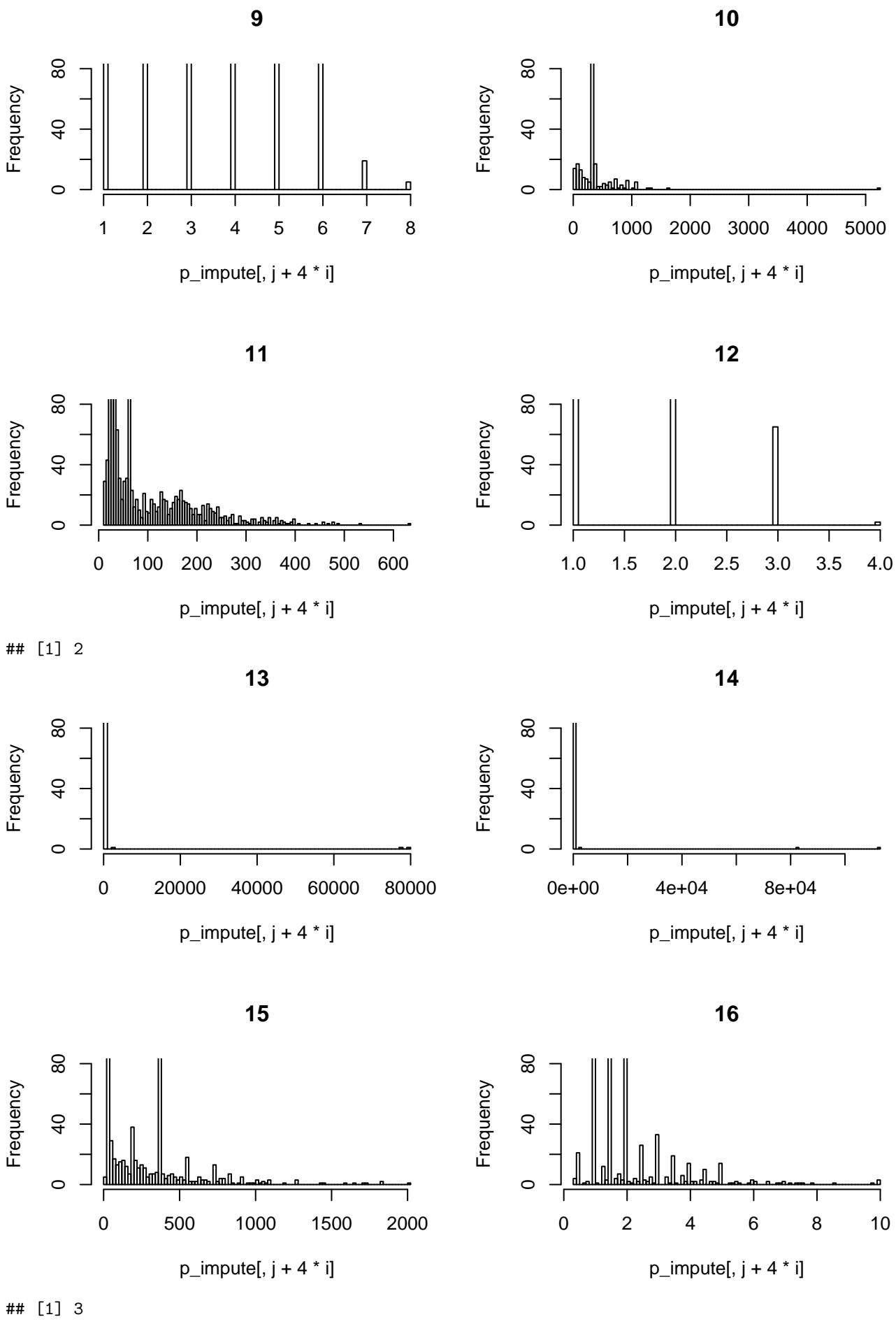
for(i in 0:11){
  for( j in 1:4){

    if(j + 4 * i <= ncol(p_impute)){
      hist(p_impute[, j + 4 * i], breaks = 100, ylim = c(0, 80), main = j + 4 * i)
    }

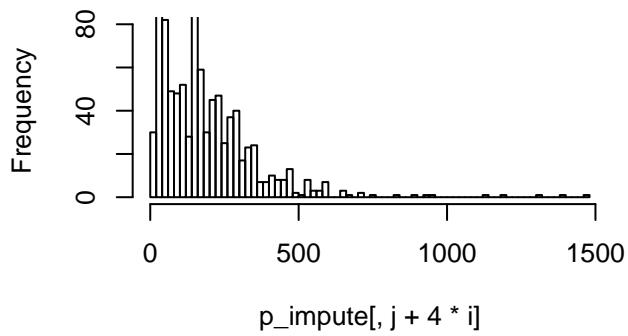
  }
  print(i)
  par(mfrow = c(2, 2))
}

```

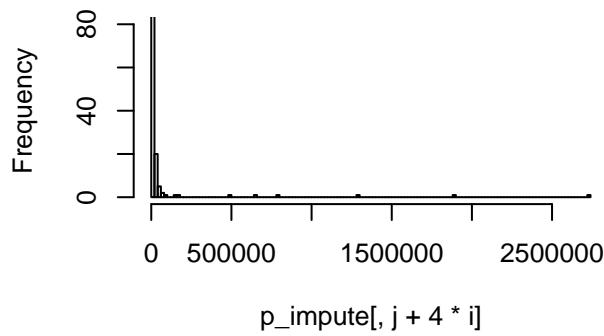




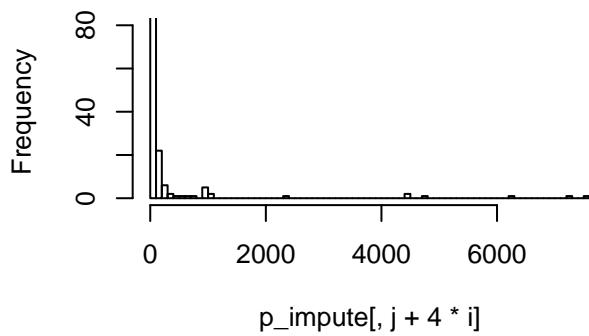
17



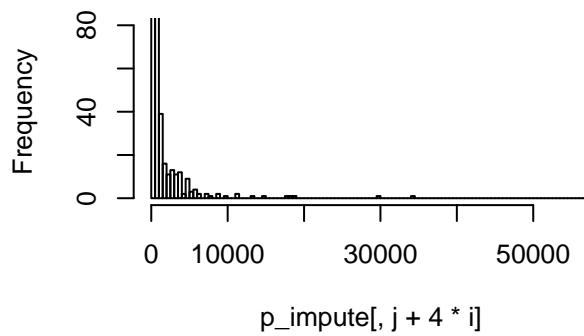
18



19

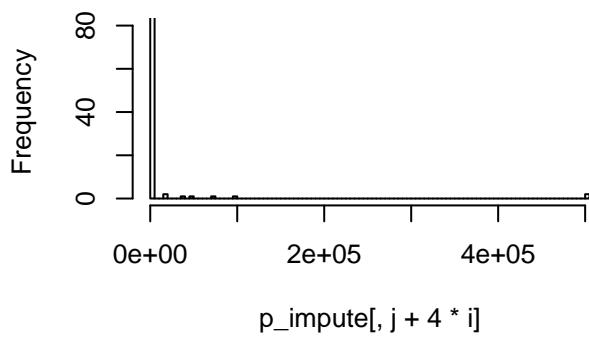


20

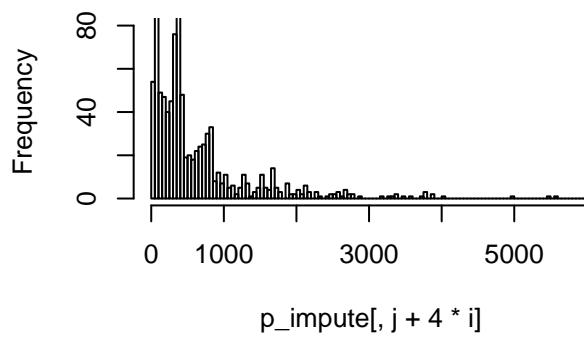


[1] 4

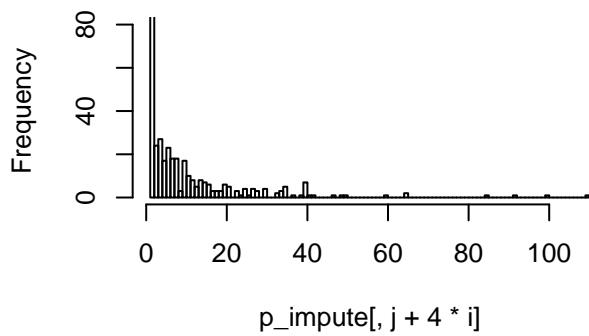
21



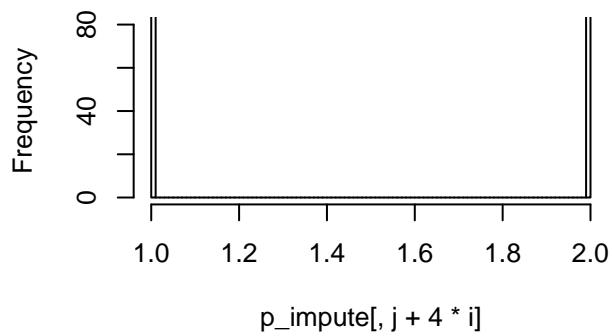
22



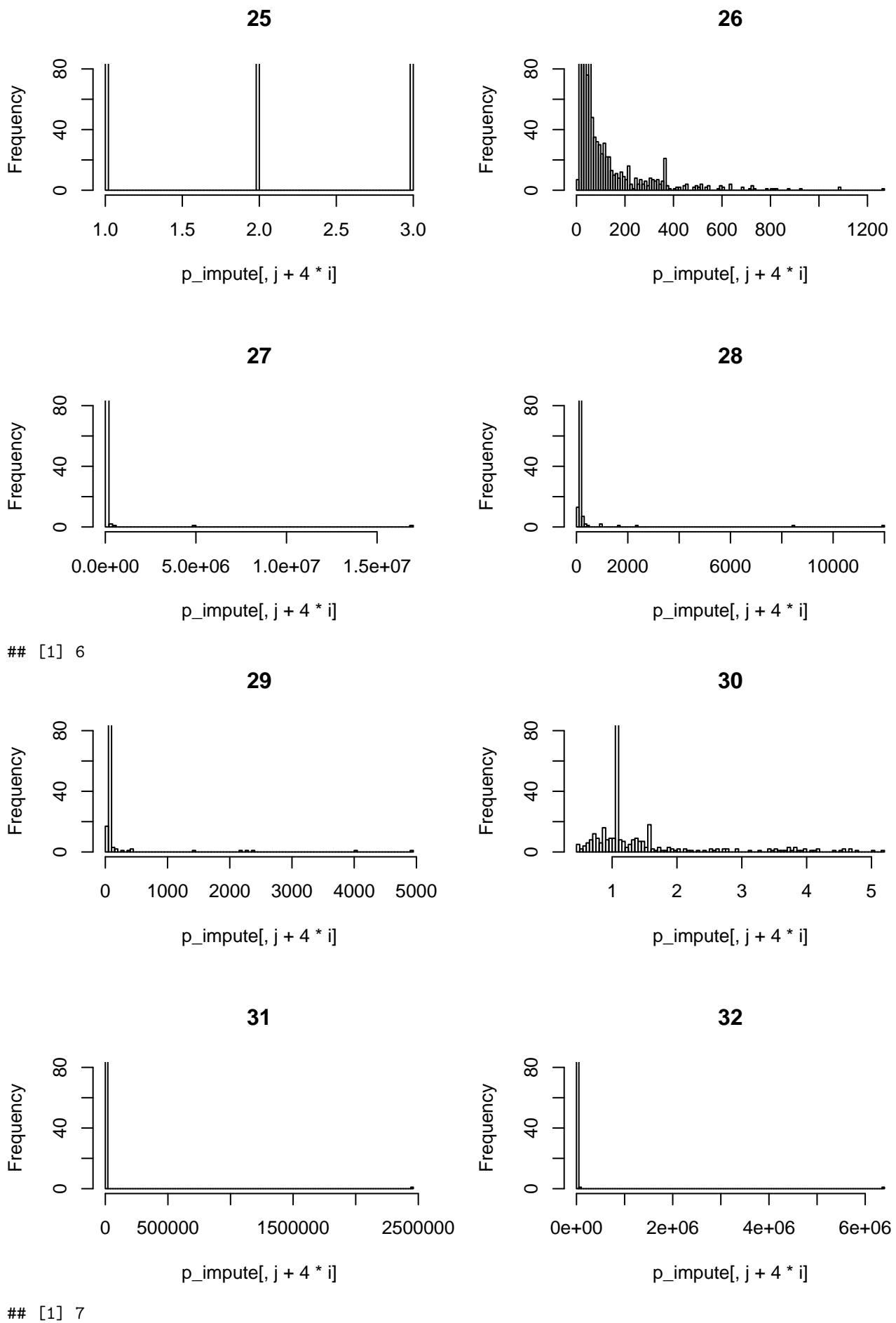
23



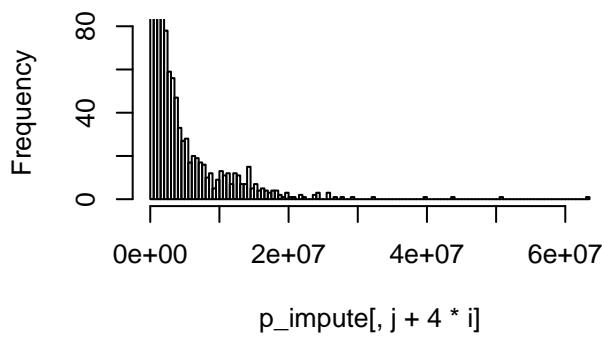
24



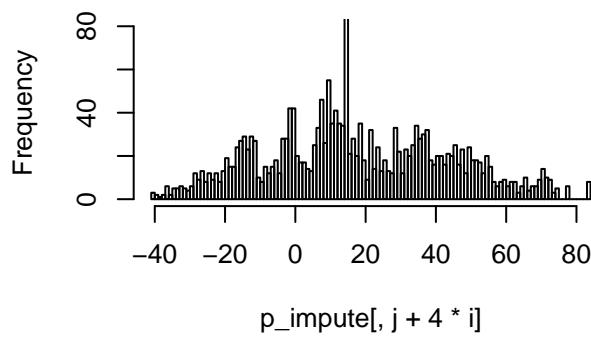
[1] 5



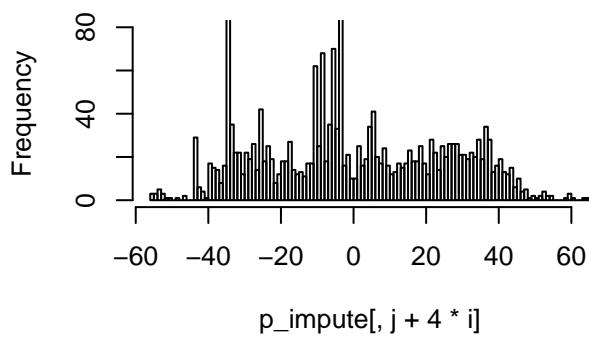
33



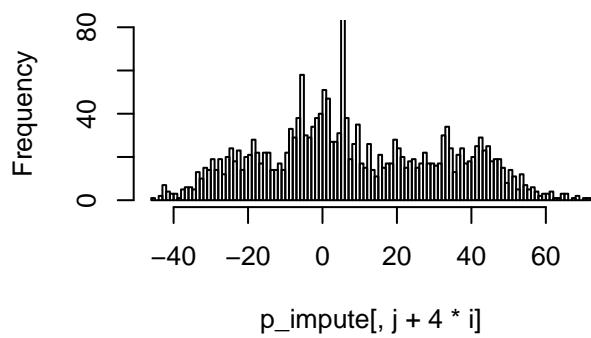
34



35

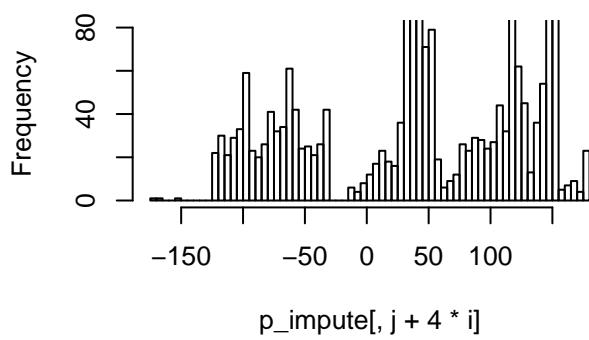


36

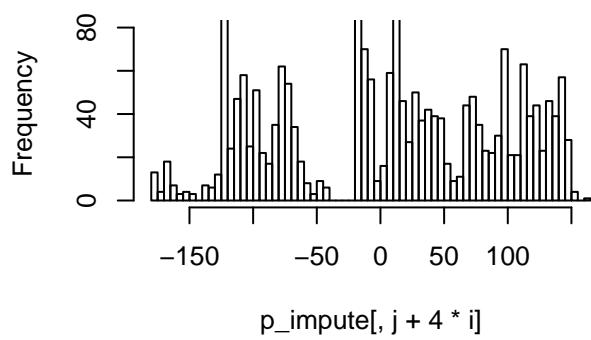


[1] 8

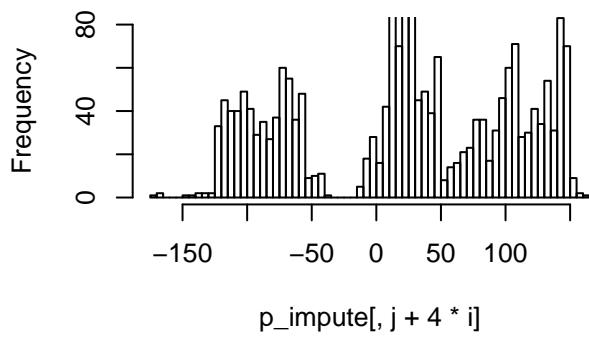
37



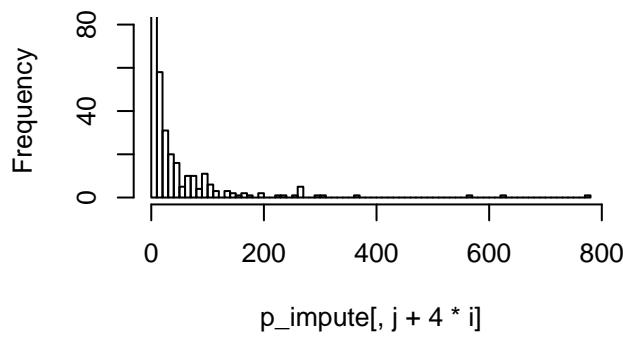
38



39

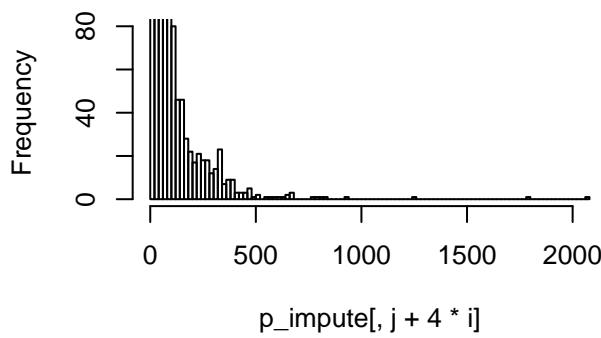


40

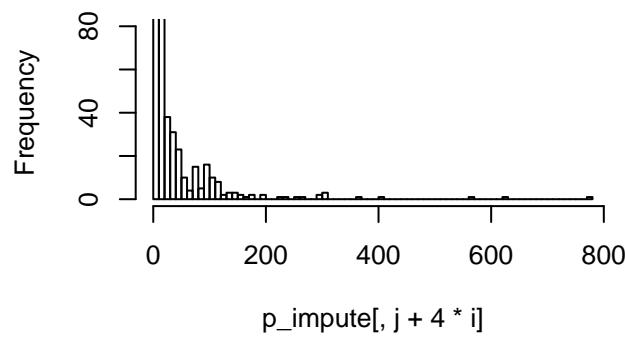


[1] 9

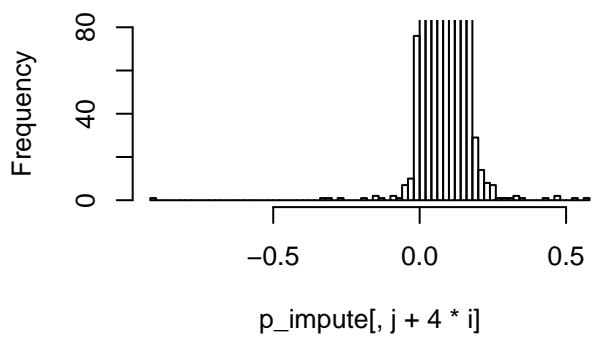
41



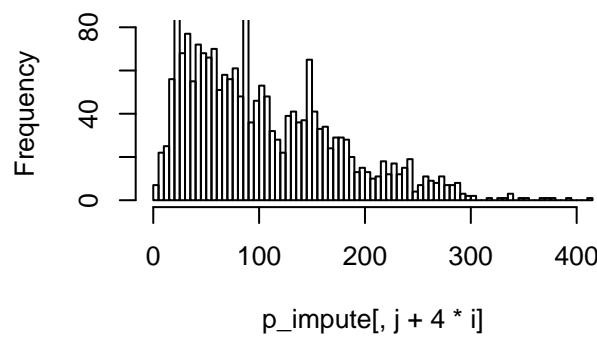
42



43

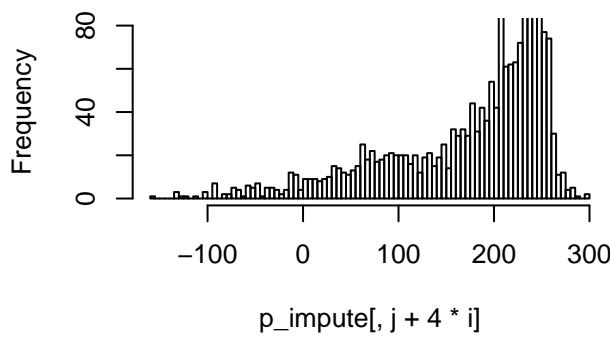


44

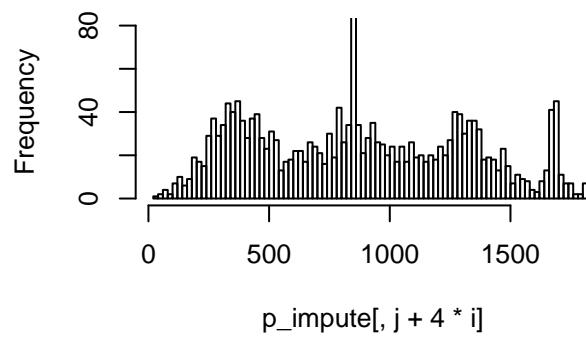


[1] 10

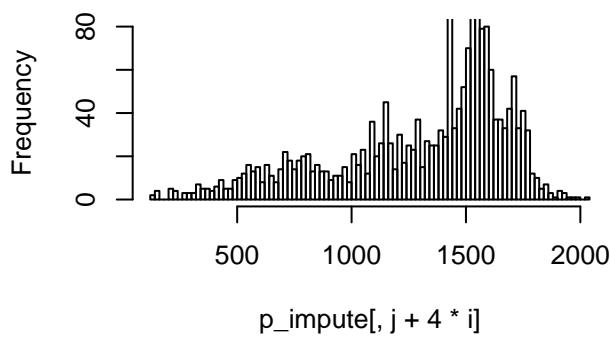
45



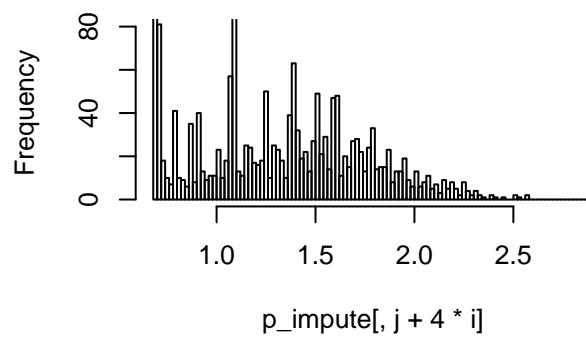
46



47



48



[1] 11

Log Transforms

```
log_cols <- c(2, 4, 7, 8,
             10, 11, 13, 14, 15, 17, 18, 19,
             20, 21, 22, 23, 26, 27, 28, 29,
             31, 32, 33,
             40, 41, 42)

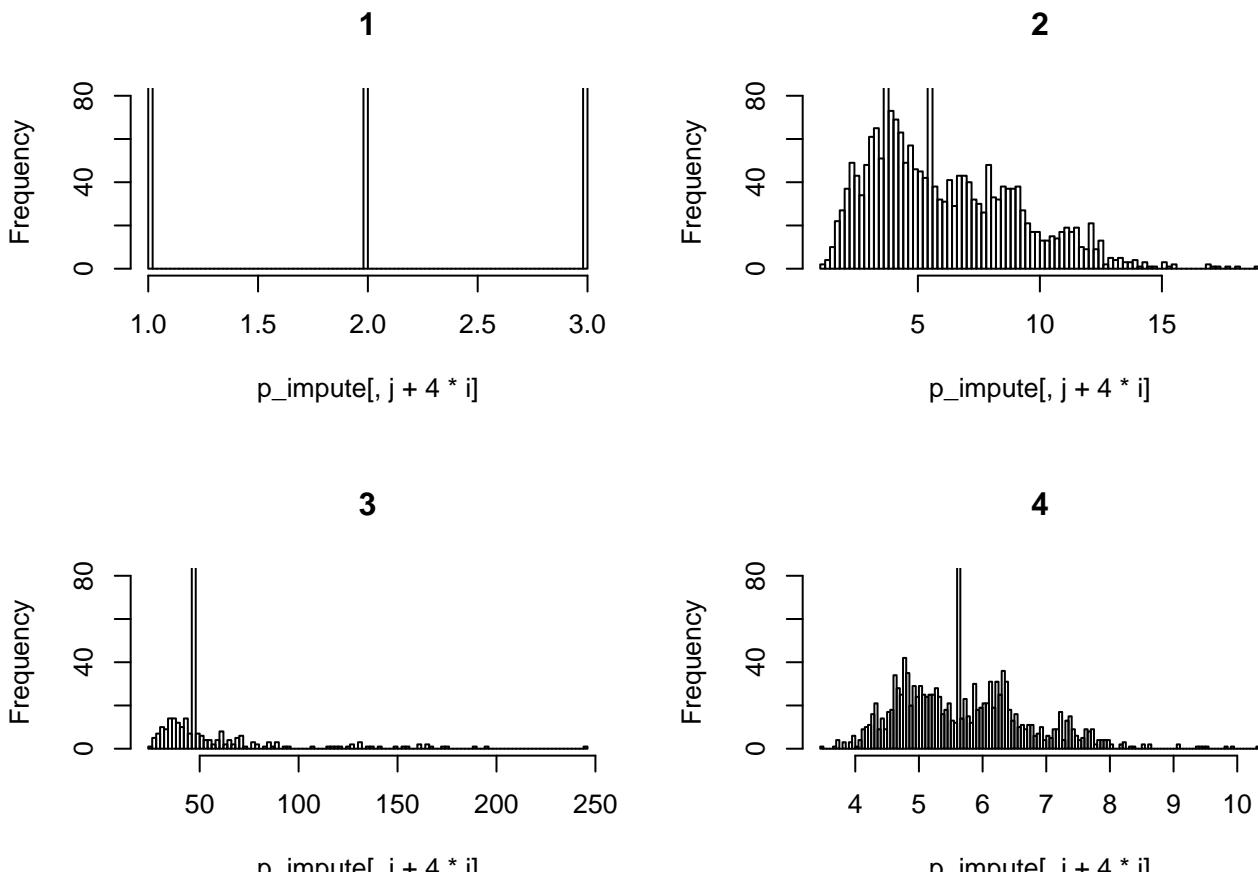
p_impute[, log_cols] <- log1p(p_impute[, log_cols])

par(mfrow = c(2, 2))

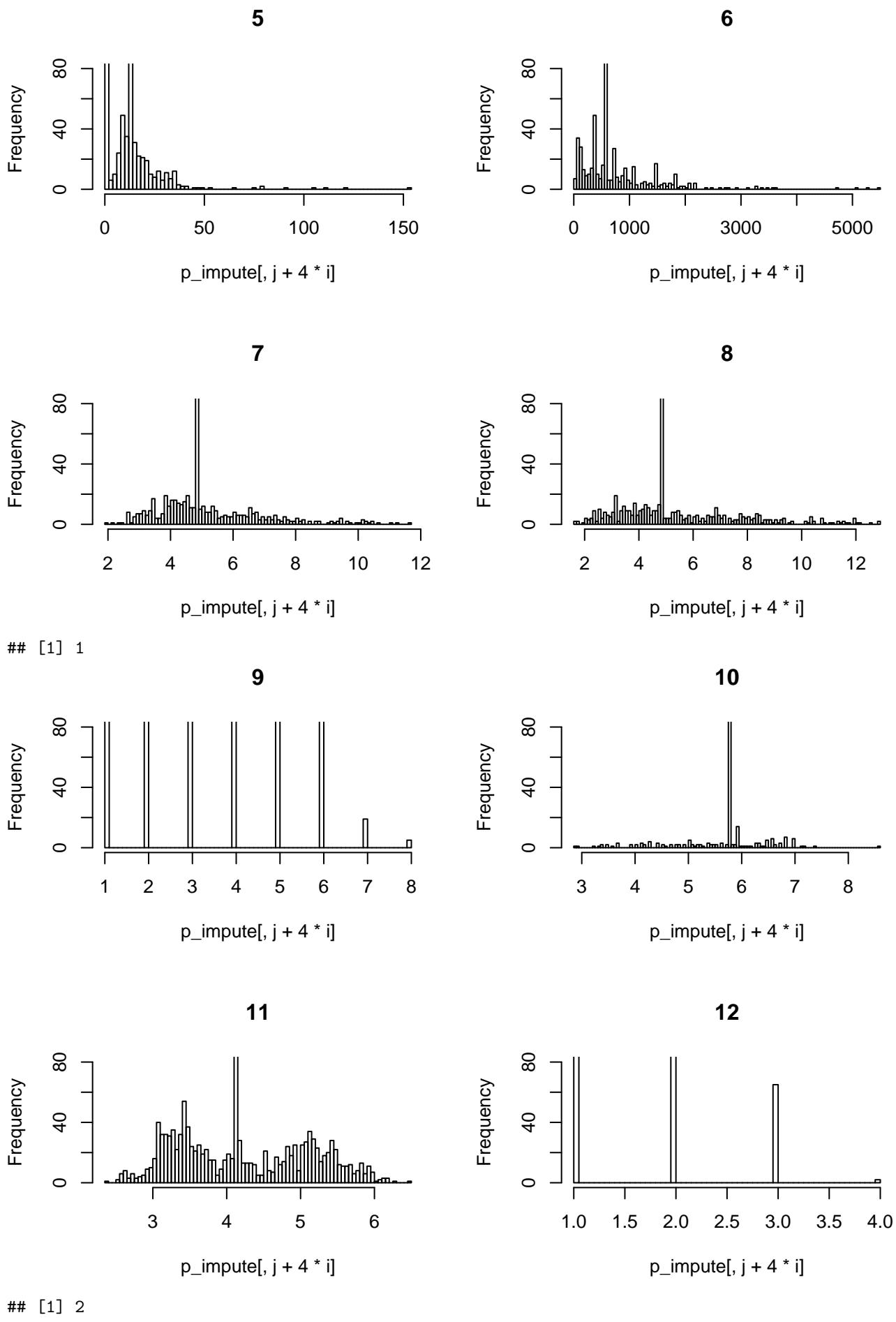
for(i in 0:11){
  for(j in 1:4){

    if(j + 4 * i <= ncol(p_impute)){
      hist(p_impute[, j + 4 * i], breaks = 100, ylim = c(0, 80), main = j + 4 * i)
    }

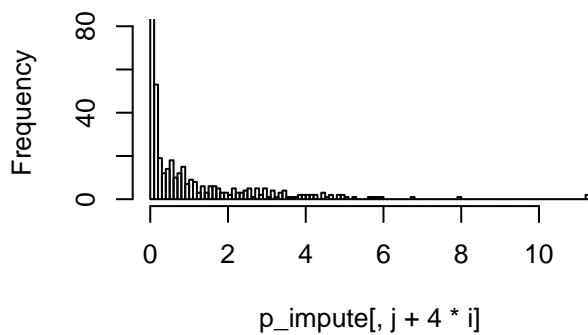
  }
  print(i)
  par(mfrow = c(2, 2))
}
```



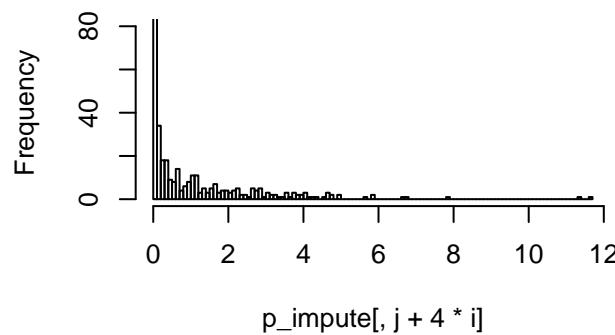
```
## [1] 0
```



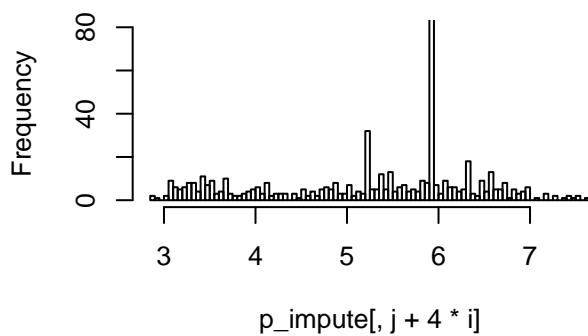
13



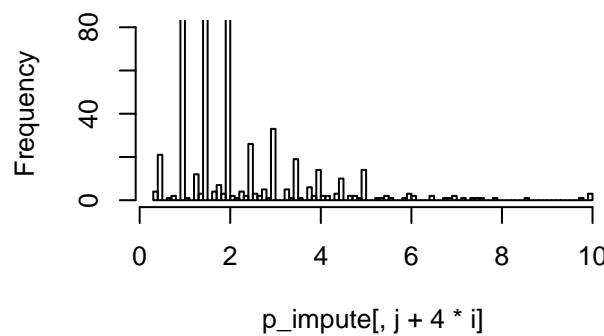
14



15

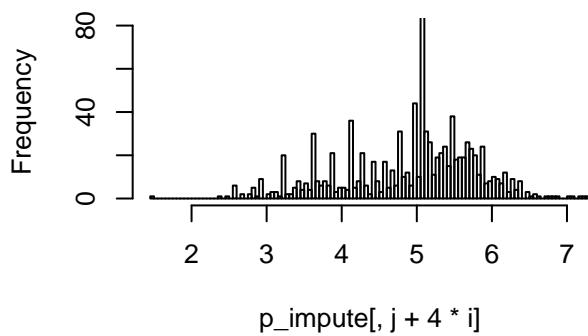


16

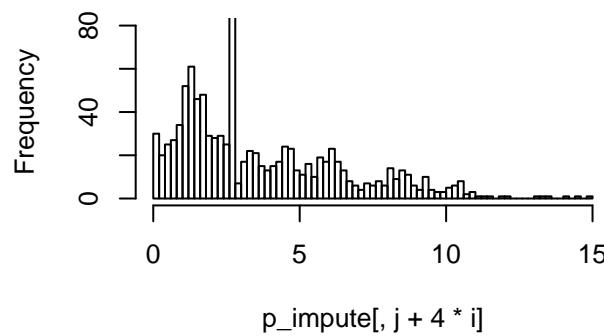


[1] 3

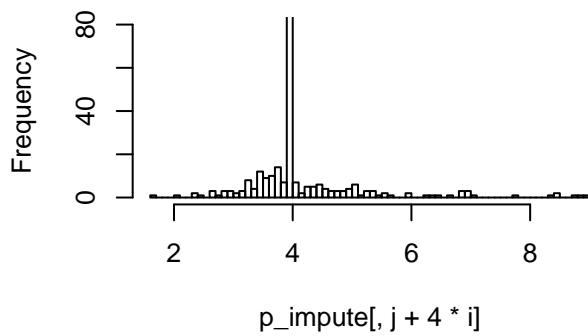
17



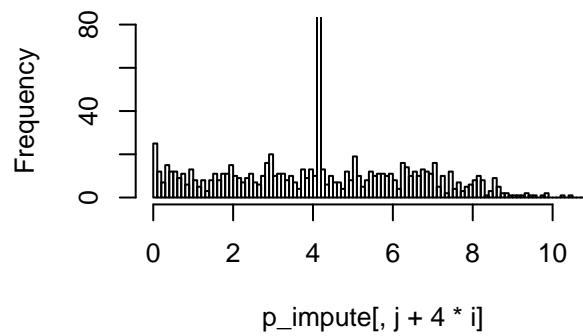
18



19

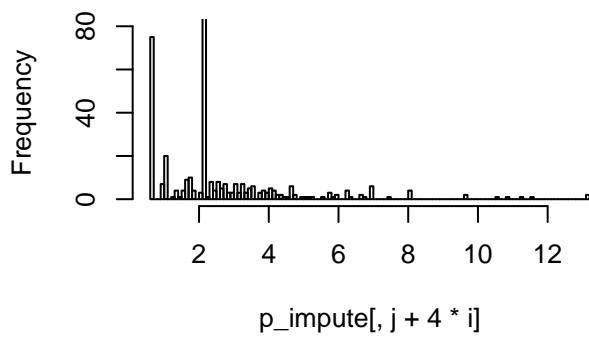


20

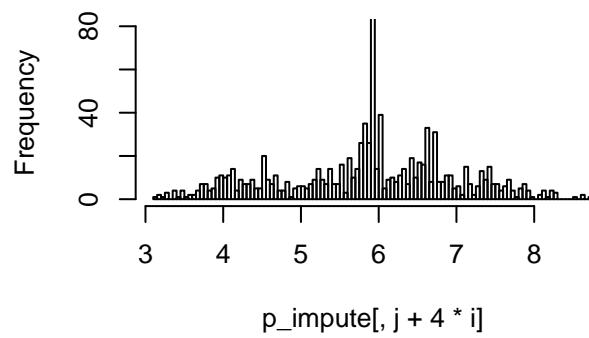


[1] 4

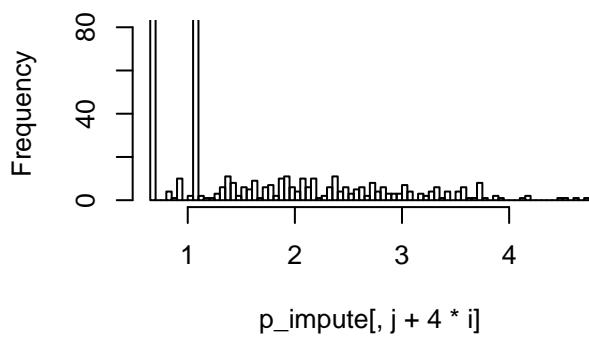
21



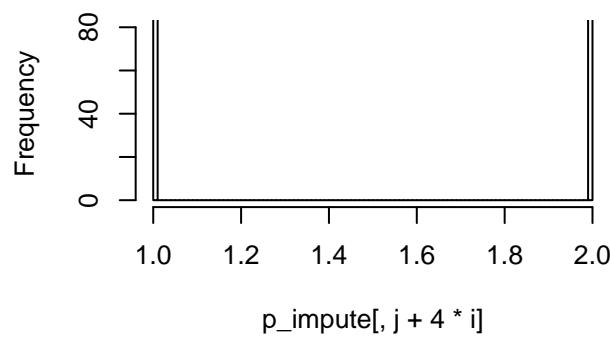
22



23

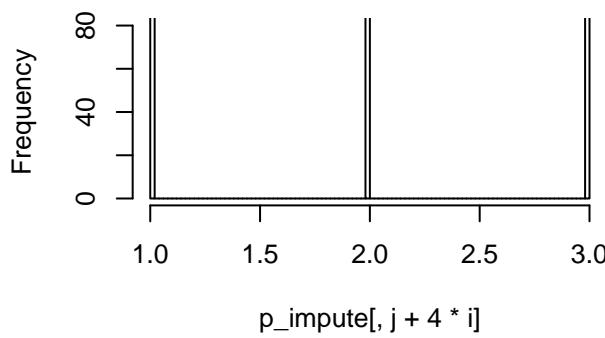


24

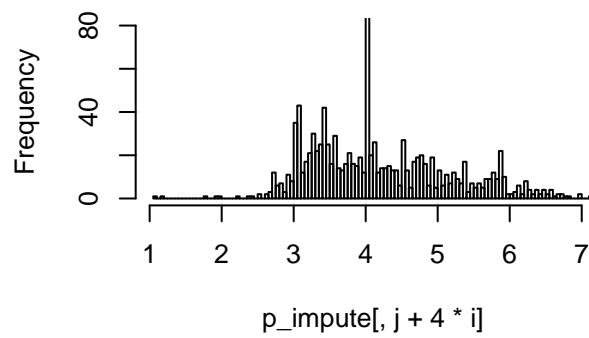


[1] 5

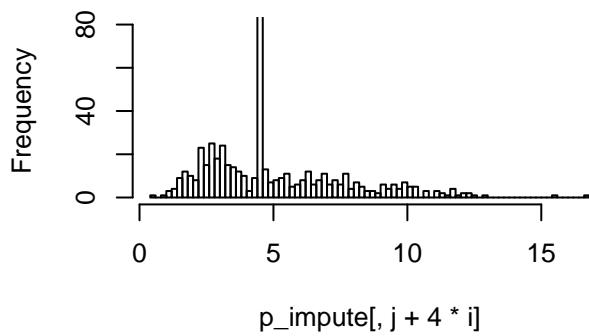
25



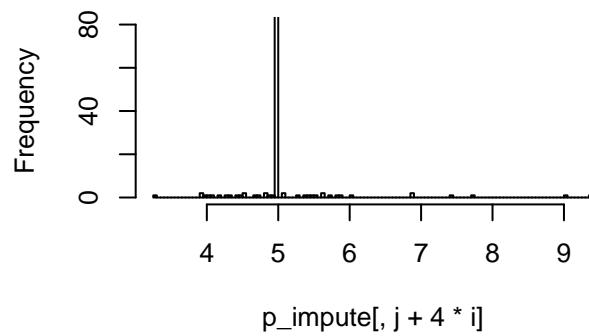
26



27

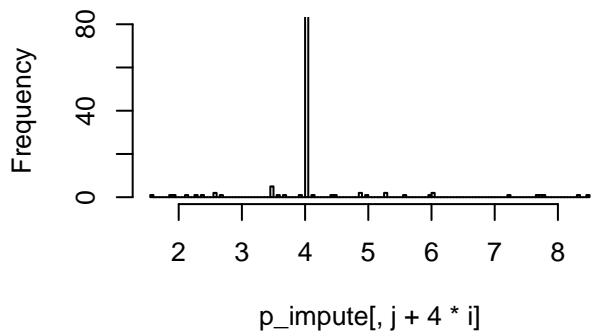


28

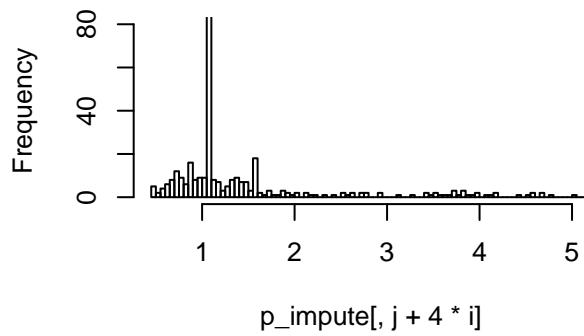


[1] 6

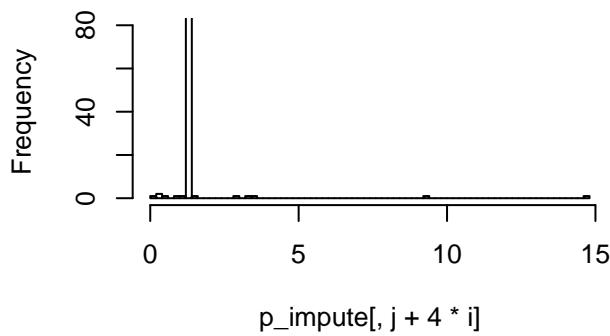
29



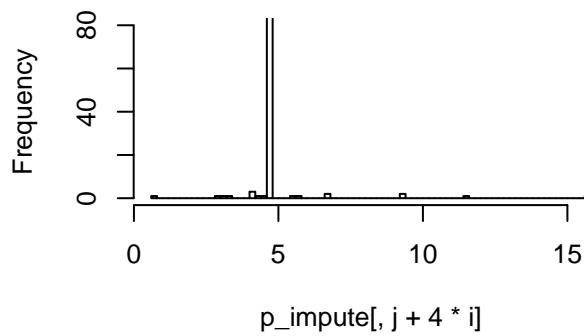
30



31

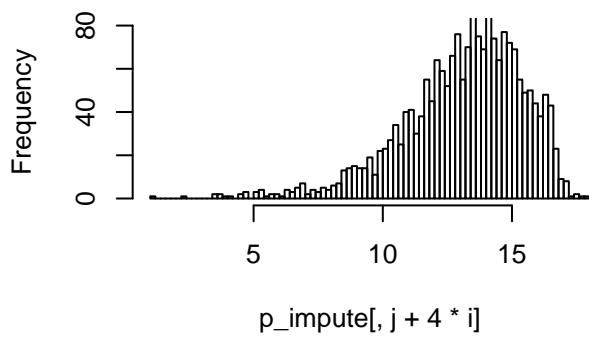


32

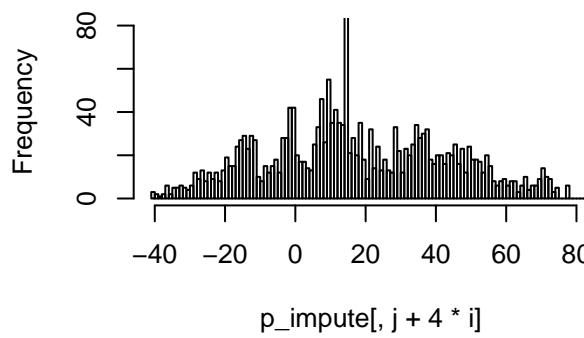


[1] 7

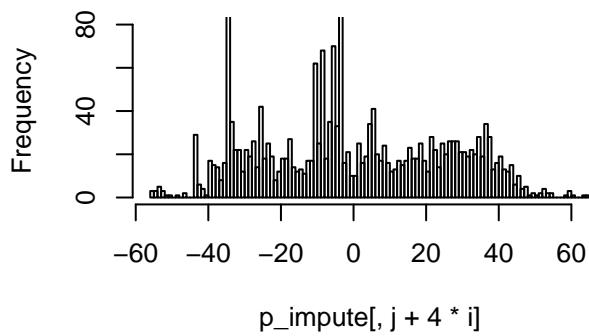
33



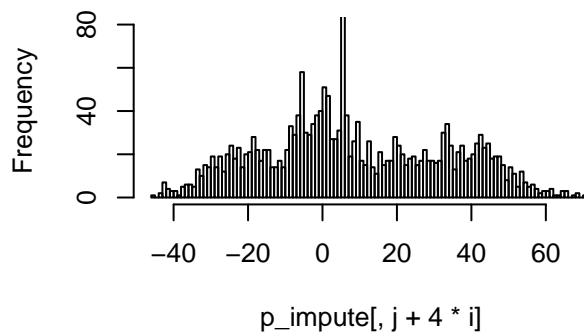
34



35

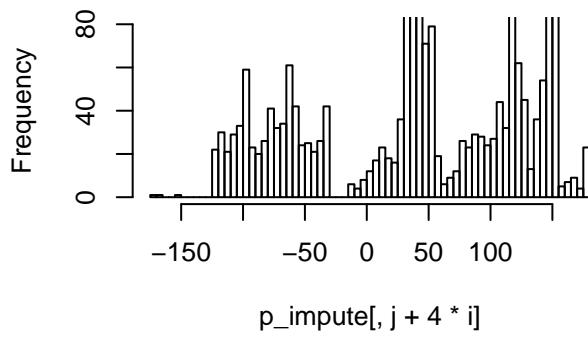


36

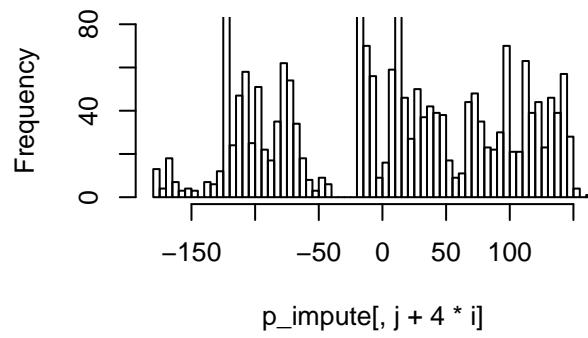


[1] 8

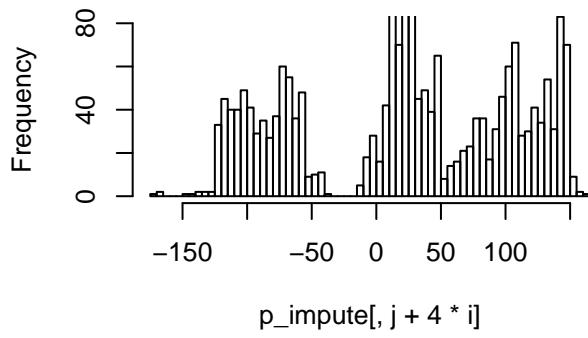
37



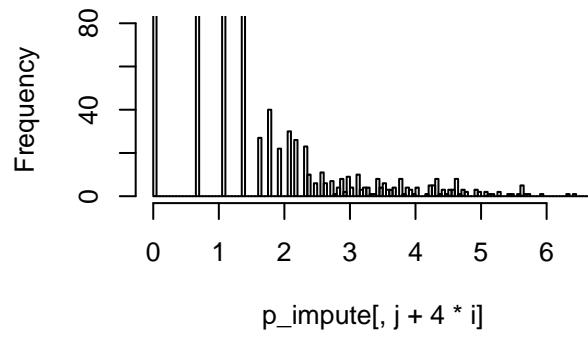
38



39

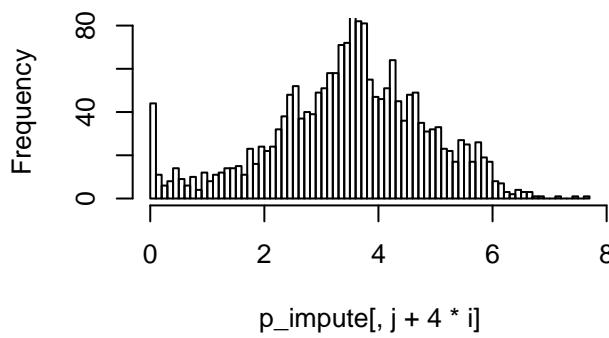


40

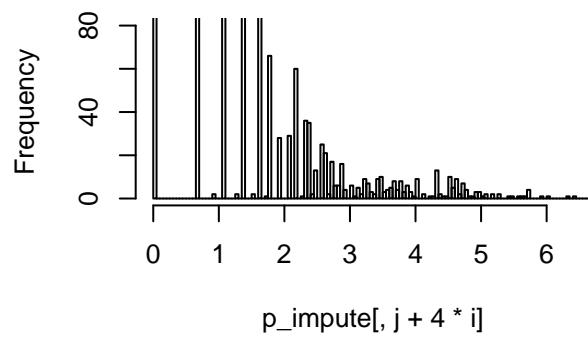


[1] 9

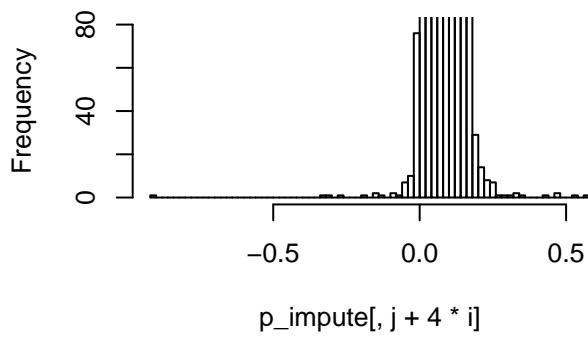
41



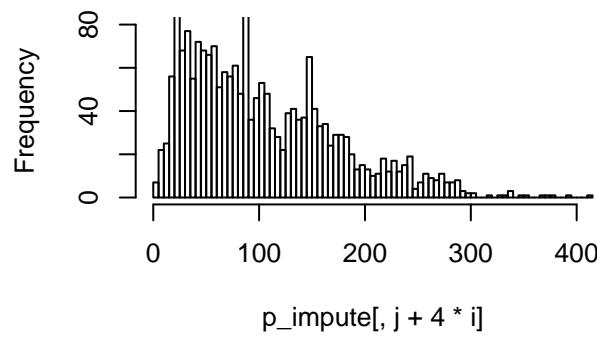
42



43

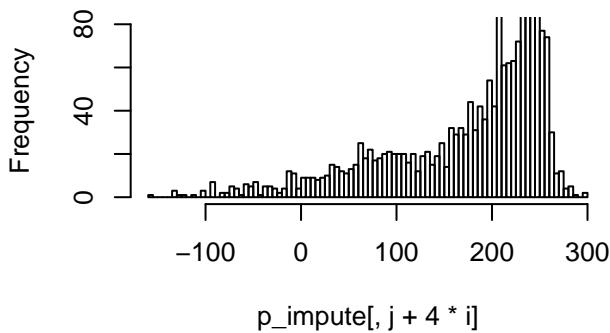


44

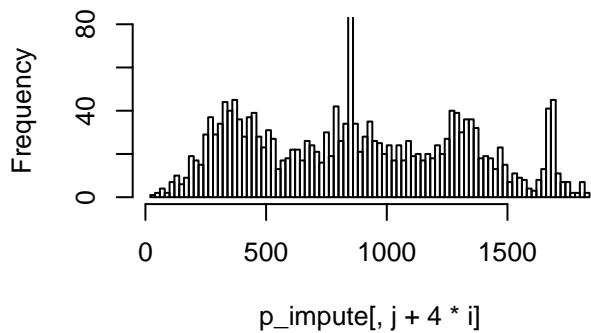


[1] 10

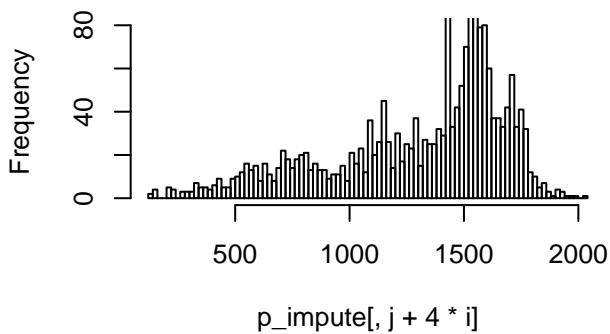
45



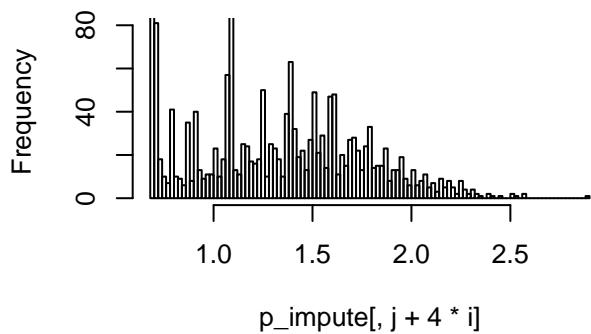
46



47



48



```
## [1] 11
```

Now fit 4 models.

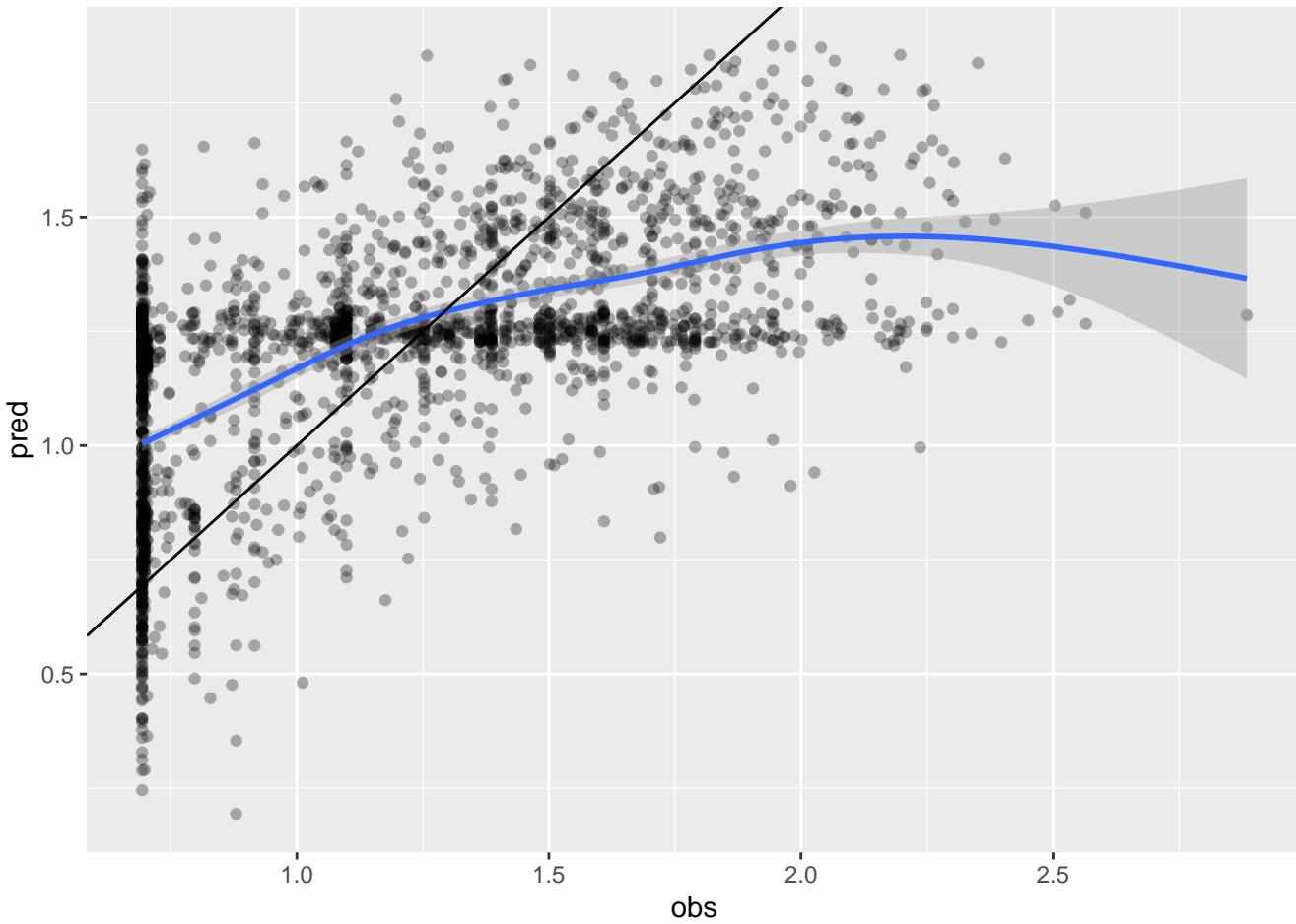
```
folds <- createFolds(p$y, k = 5, returnTrain = TRUE)
trcntrl <- trainControl(index = folds, savePredictions = TRUE, search = 'random')

cl <- makeForkCluster(6)
registerDoParallel(cl)

apriori_formula <- y ~ X5.1_AdultBodyMass_g + X3.1_AgeatFirstBirth_d + X18.1_BasalMetRate_mL02hr + X9.1_Gestat
m0_lm <- train(apriori_formula, data = p_impute, method = 'lm', trControl = trcntrl, na.action = na.omit)

plotCV(m0_lm)

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```
m0_lm
```

```
## Linear Regression
##
## 2143 samples
##   6 predictors
##
## No pre-processing
## Resampling: Bootstrapped (5 reps)
## Summary of sample sizes: 1714, 1715, 1714, 1715, 1714
## Resampling results:
##
##   RMSE      Rsquared    MAE
##   0.3705782  0.337926  0.2967044
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
summary(m0_lm$finalModel)

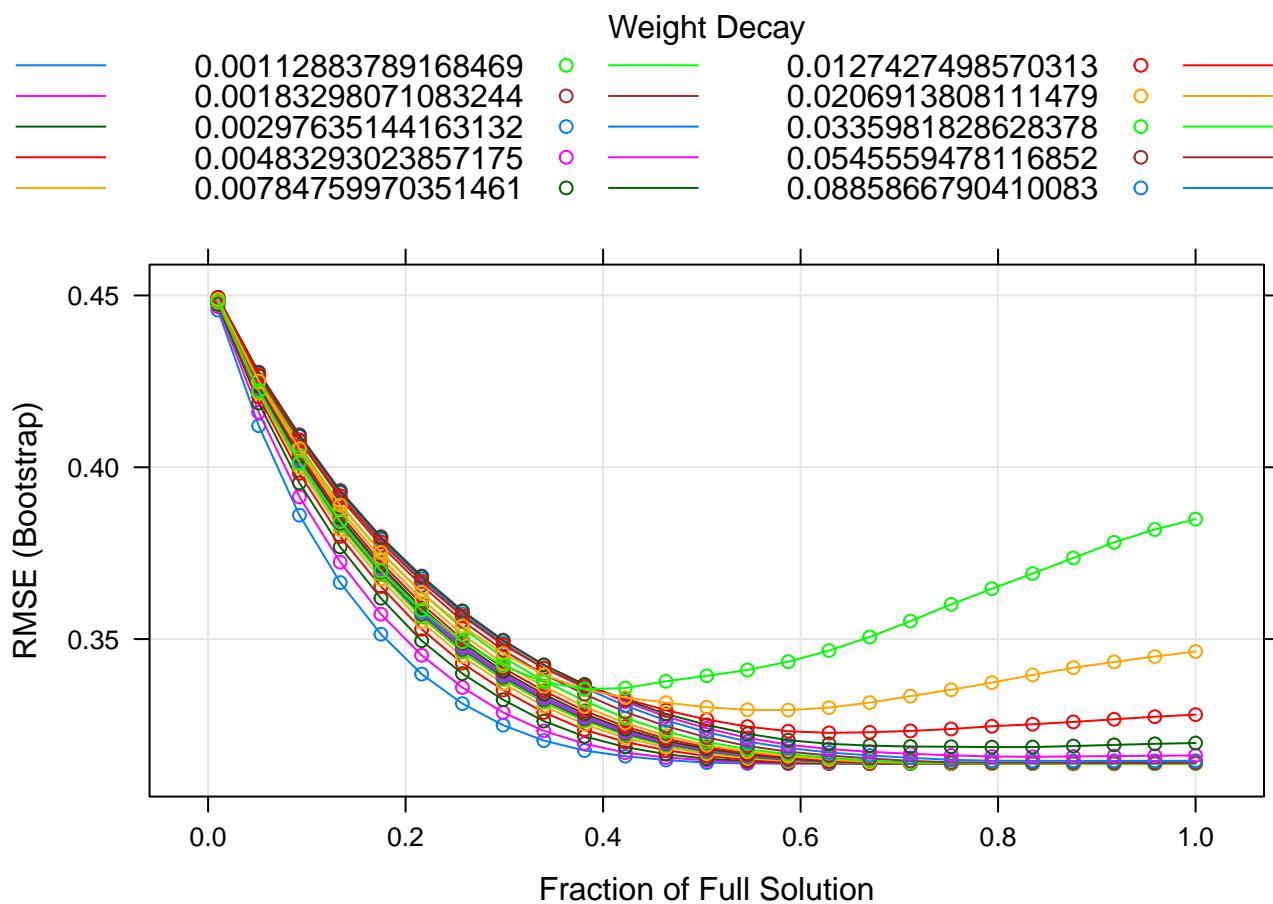
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##   Min     1Q     Median      3Q     Max 
## -0.92083 -0.24850 -0.01827  0.23556  1.59271
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 2.897e+00 9.188e-02 31.535 < 2e-16 ***
## X5.1_AdultBodyMass_g -1.732e-02 3.739e-03 -4.632 3.83e-06 ***
## 
```

```

## X3.1_AgeatFirstBirth_d      6.012e-06  2.455e-05   0.245   0.8066
## X18.1_BasalMetRate_mL02hr 1.368e-02  9.846e-03   1.389   0.1648
## X9.1_GestationLen_d        -2.710e-01 1.574e-02 -17.214 < 2e-16 ***
## X16.1_LittersPerYear       1.548e-02  9.368e-03   1.652   0.0987 .
## X17.1_MaxLongevity_m       -1.082e-01 1.624e-02 -6.662  3.42e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3704 on 2136 degrees of freedom
## Multiple R-squared:  0.3402, Adjusted R-squared:  0.3383
## F-statistic: 183.5 on 6 and 2136 DF,  p-value: < 2.2e-16

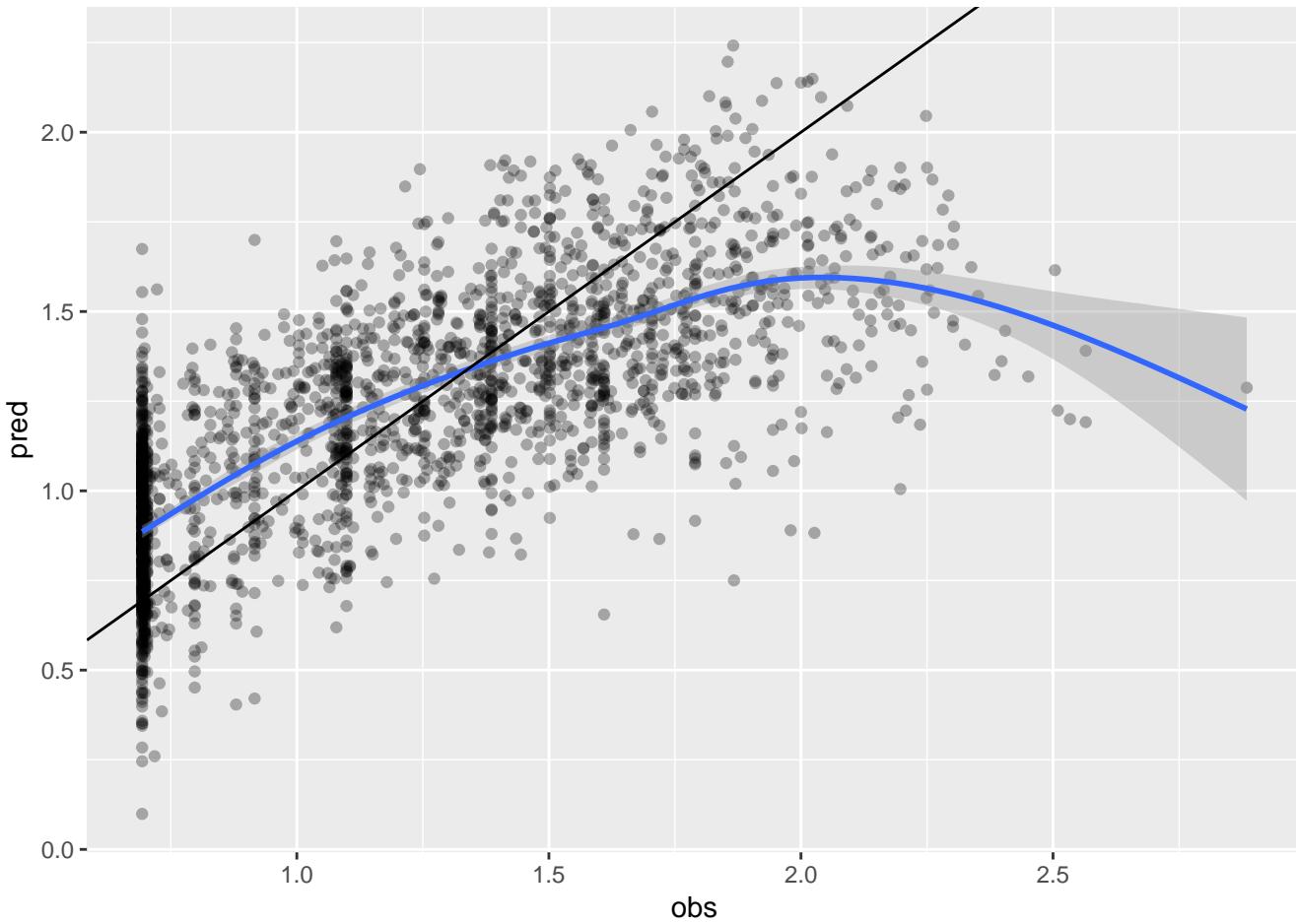
enet_gr <- expand.grid(lambda = 10 ^ seq(0, -4, length.out = 20), fraction = c(seq(0.01, 1, length.out = 25)))
m1_enet <- train(y ~ ., data = p_impute, method = 'enet', tuneGrid = enet_gr, trControl = trcntrl, na.action =
plot(m1_enet)

```



```
plotCV(m1_enet)
```

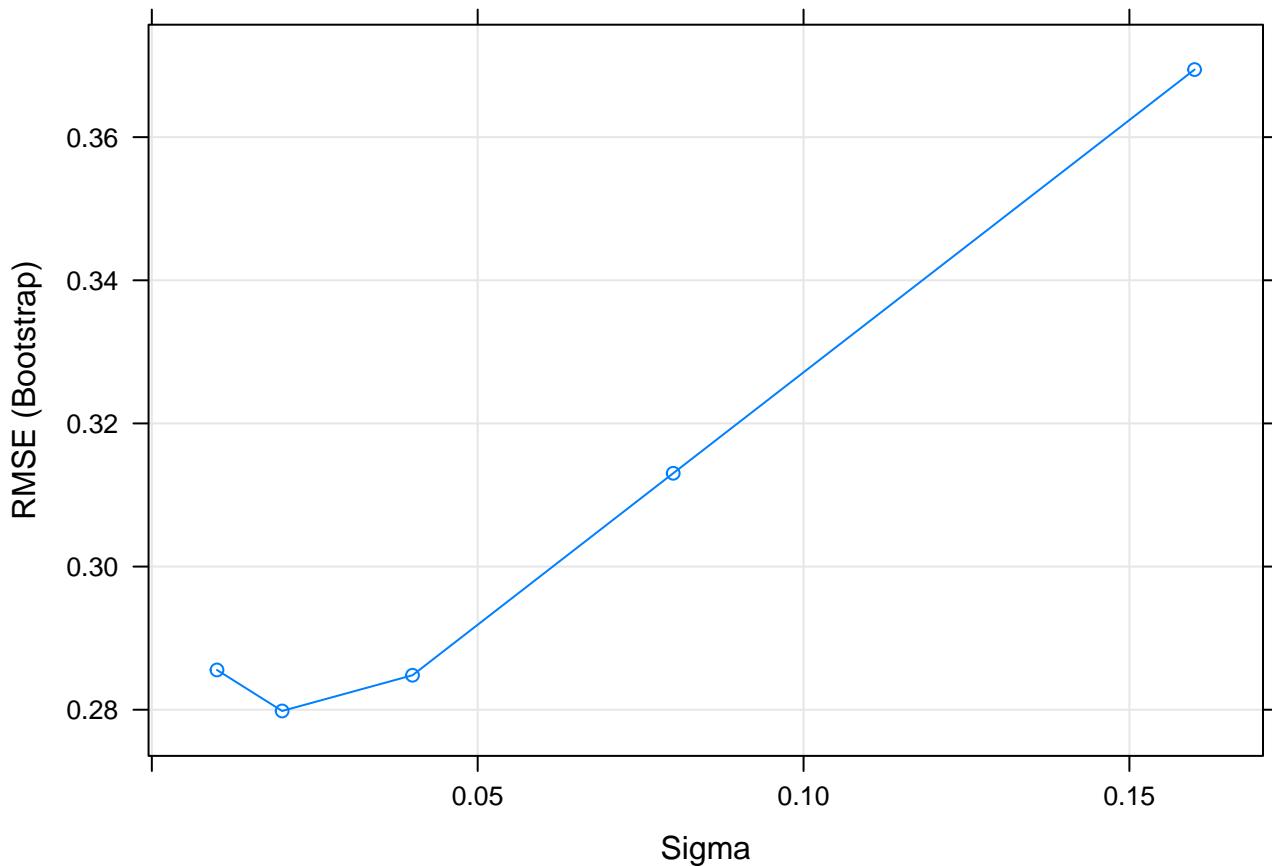
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```
m1_enet$results$R squared %>% max
```

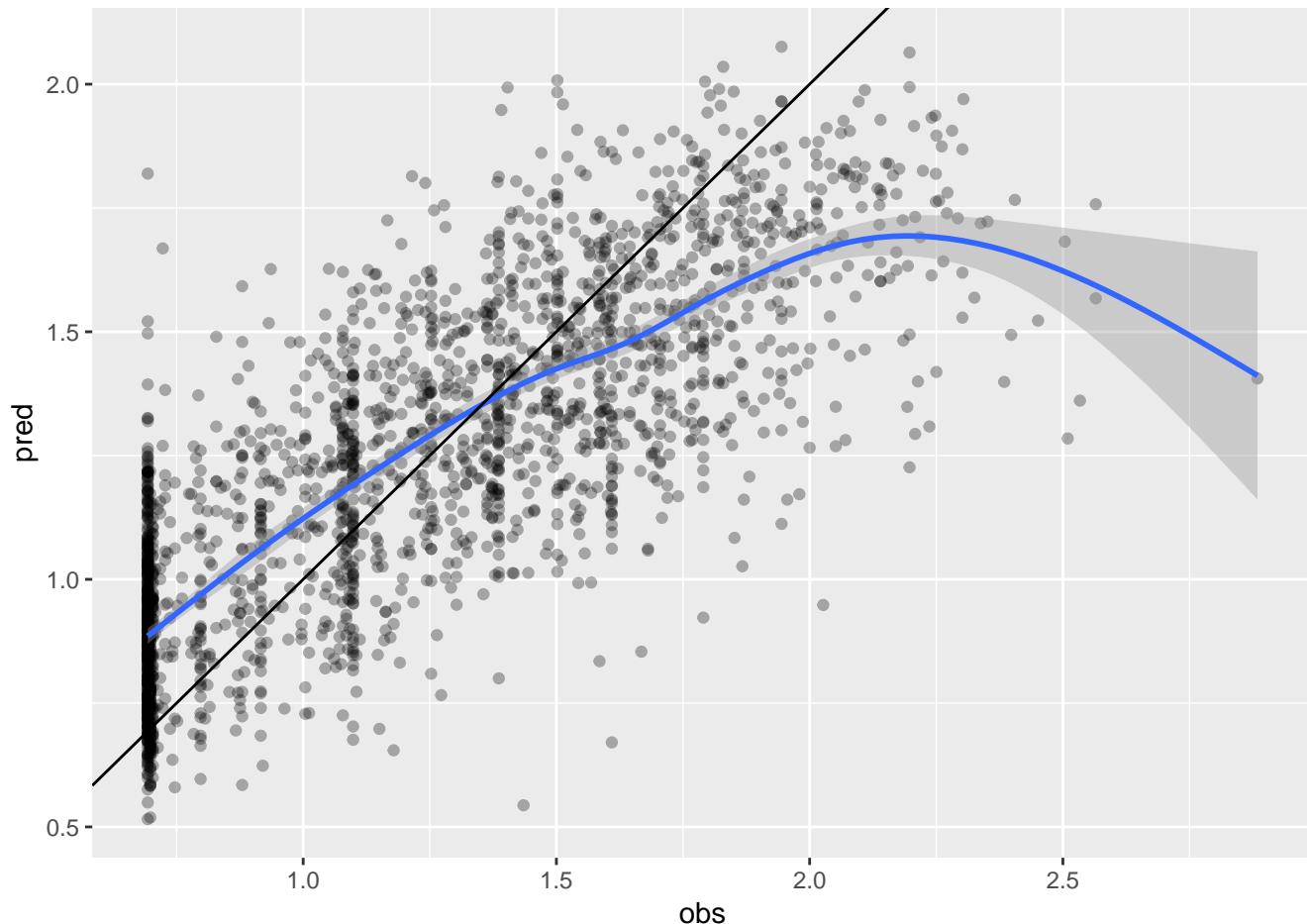
```
## [1] 0.526839
# Find the final parameters
#   https://stackoverflow.com/questions/40088228/how-to-retrieve-elastic-net-coefficients
final_pars <- predict.enet(m1_enet$finalModel, s=m1_enet$bestTune[1], "fraction"), type="coef", mode="fraction"
final_pars
sum(final_pars != 0)

gp_gr <- data.frame(sigma = c(0.01, 0.02, 0.04, 0.08, 0.16))
m2_gp <- train(y ~ ., data = p_impute, method = 'gaussprRadial', tuneGrid = gp_gr, trControl = trcntrl, na.action = na.omit)
plot(m2_gp)
```



```
plotCV(m2_gp)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```

m2_gp

## Gaussian Process with Radial Basis Function Kernel
##
## 2143 samples
##   47 predictors
##
## No pre-processing
## Resampling: Bootstrapped (5 reps)
## Summary of sample sizes: 1714, 1715, 1714, 1715, 1714
## Resampling results across tuning parameters:
##
##   sigma   RMSE      Rsquared     MAE
##   0.01    0.2855551  0.6095879  0.2218111
##   0.02    0.2798232  0.6266082  0.2183842
##   0.04    0.2848164  0.6178132  0.2253383
##   0.08    0.3130373  0.5515241  0.2540034
##   0.16    0.3694406  0.3992555  0.3080410
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was sigma = 0.02.

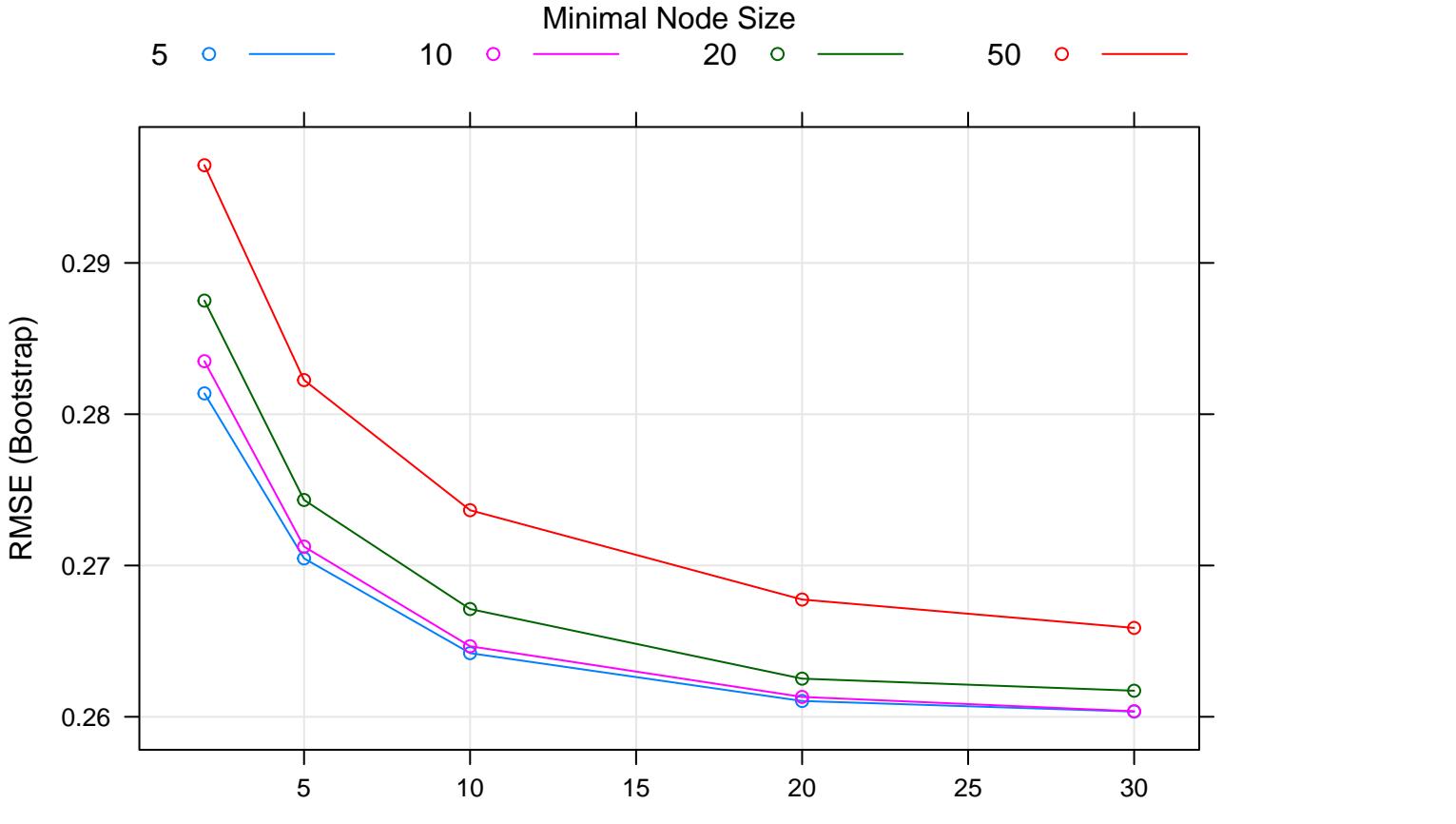
m2_gp$results$Rsquared %>% max

## [1] 0.6266082

rf_gr <- expand.grid(mtry = c(2, 5, 10, 20, 30), splitrule = 'variance', min.node.size = c(5, 10, 20, 50))
m3_rf <- train(y ~ ., data = p_impute, method = 'ranger', tuneGrid = rf_gr, trControl = trcntrl, na.action = na.omit)

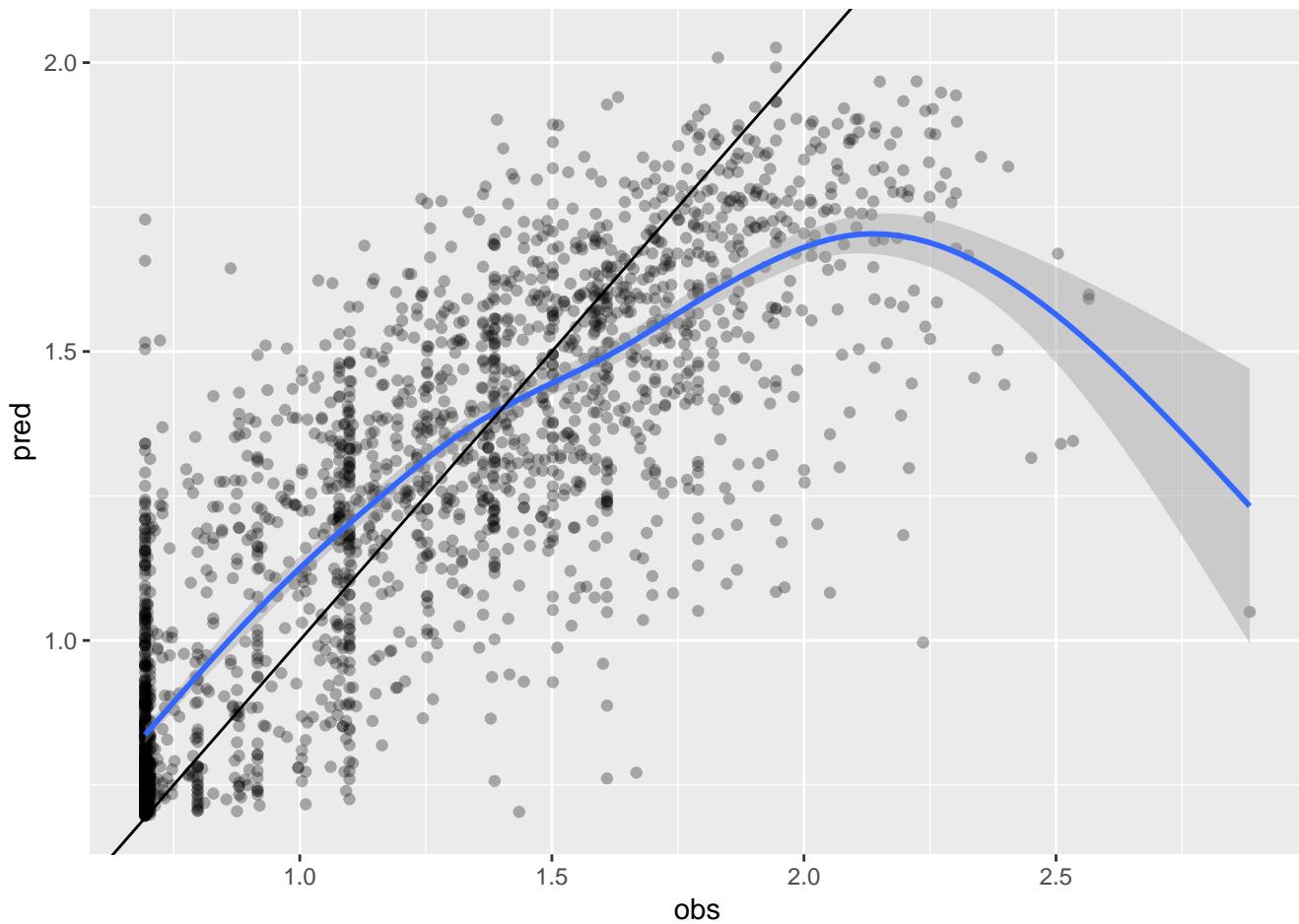
plot(m3_rf)

```



```
plotCV(m3_rf)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```
m3_rf
```

```
## Random Forest
##
## 2143 samples
##   47 predictors
##
## No pre-processing
## Resampling: Bootstrapped (5 reps)
## Summary of sample sizes: 1714, 1715, 1714, 1715, 1714
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  RMSE      Rsquared     MAE
##   2      5             0.2813761  0.6405606  0.2177796
##   2      10            0.2835044  0.6354284  0.2200403
##   2      20            0.2875085  0.6262929  0.2241080
##   2      50            0.2964571  0.6058430  0.2331574
##   5      5             0.2704681  0.6595899  0.2051341
##   5      10            0.2712469  0.6583178  0.2062279
##   5      20            0.2743309  0.6517933  0.2096598
##   5      50            0.2822554  0.6332837  0.2178622
##   10     5              0.2642059  0.6711550  0.1980803
##   10     10             0.2646623  0.6705596  0.1986097
##   10     20             0.2671232  0.6649359  0.2013062
##   10     50             0.2736562  0.6499851  0.2081583
##   20     5              0.2610452  0.6756351  0.1930300
##   20     10             0.2613128  0.6751686  0.1935159
##   20     20             0.2625209  0.6725189  0.1952154
##   20     50             0.2677504  0.6604168  0.2009672
```

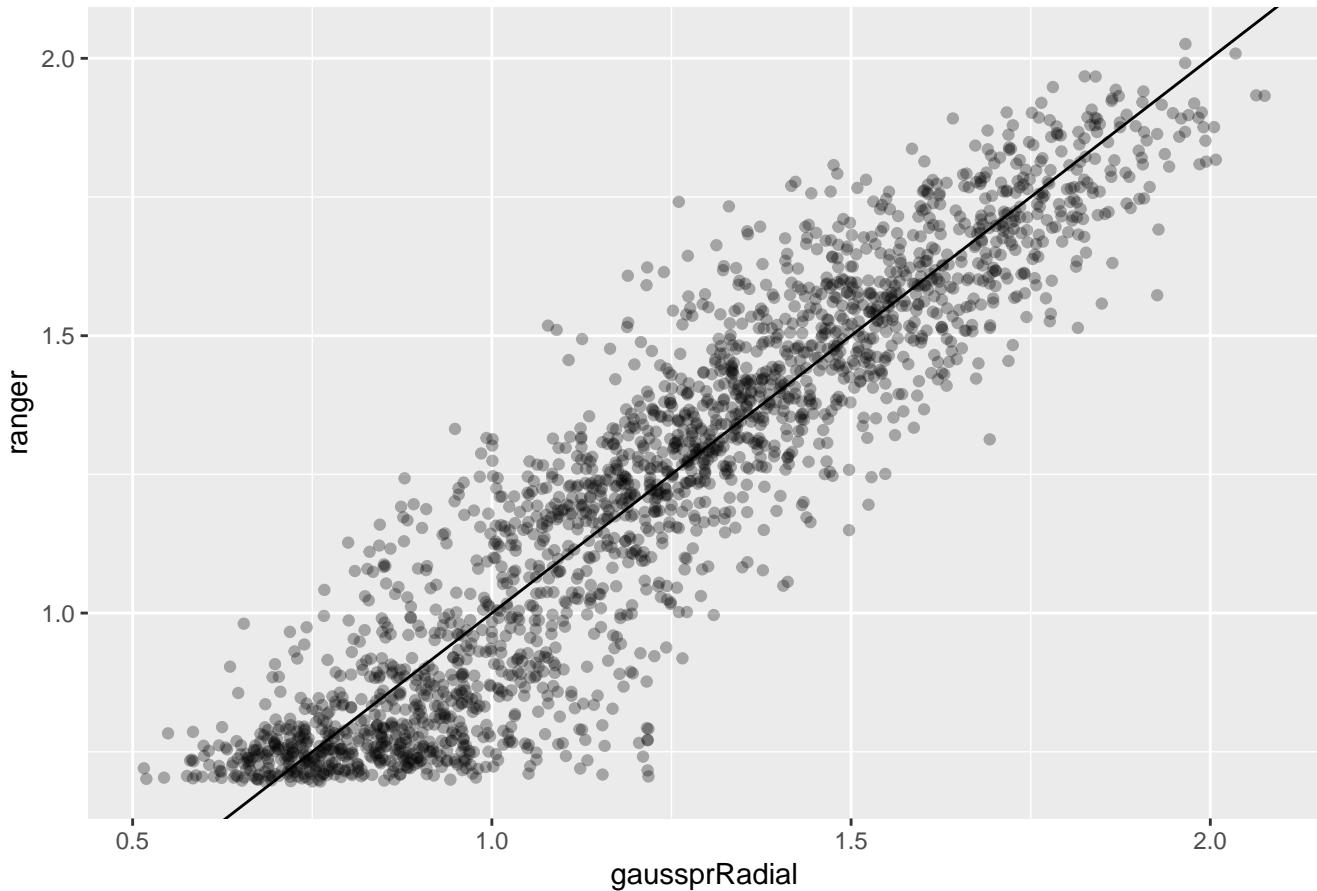
```

##   30      5      0.2603446  0.6761010  0.1913242
##   30     10      0.2603651  0.6761785  0.1915433
##   30     20      0.2617231  0.6728620  0.1931893
##   30     50      0.2658725  0.6631516  0.1981692
##
## Tuning parameter 'splitrule' was held constant at a value of variance
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 30, splitrule =
## variance and min.node.size = 5.

m3_rf$results$Rsquared %>% max
## [1] 0.6761785
compare_models(m2_gp, m3_rf)

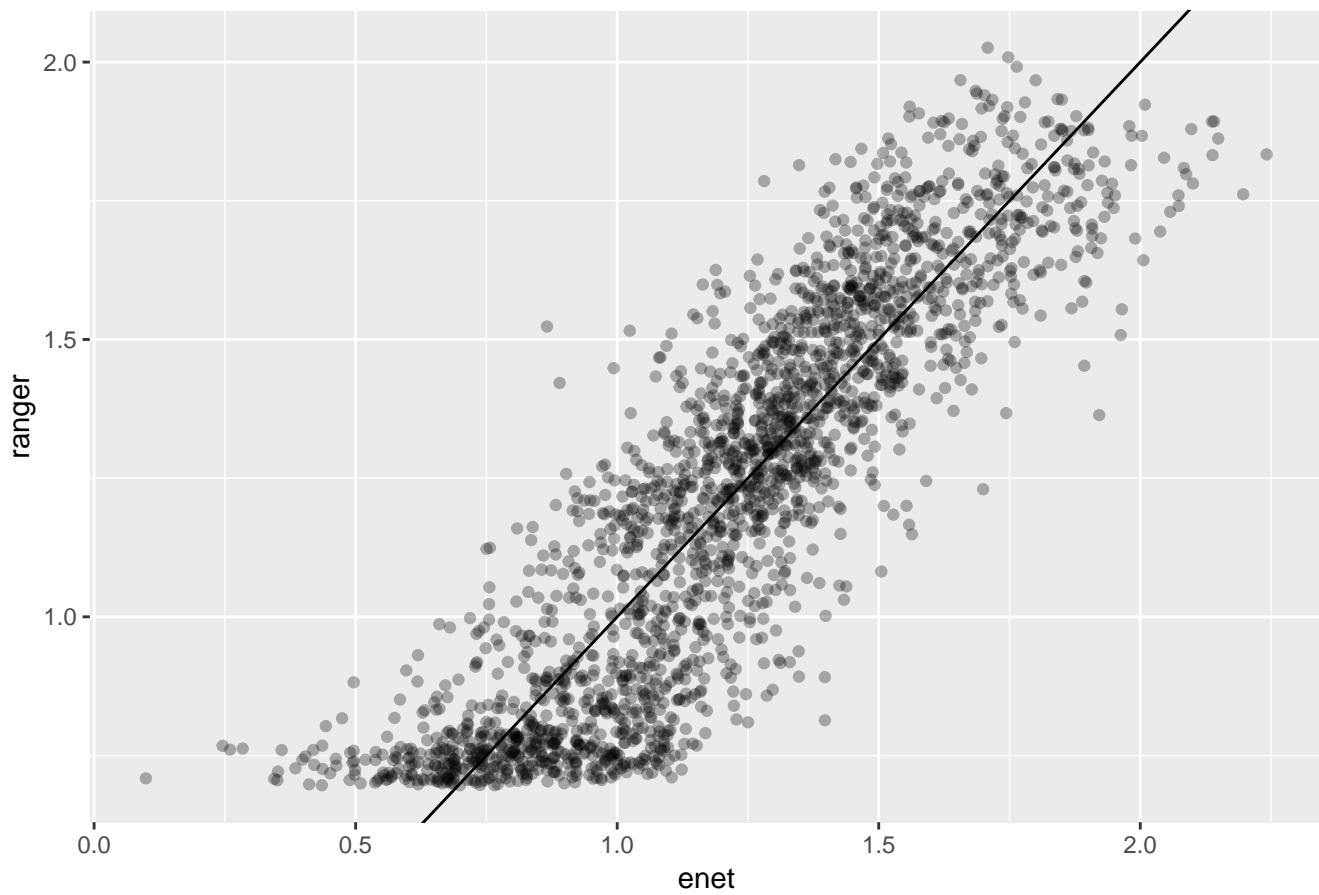
```

Correlation: 0.93. t1 RMSE: 0.28. t2 RMSE: 0.26



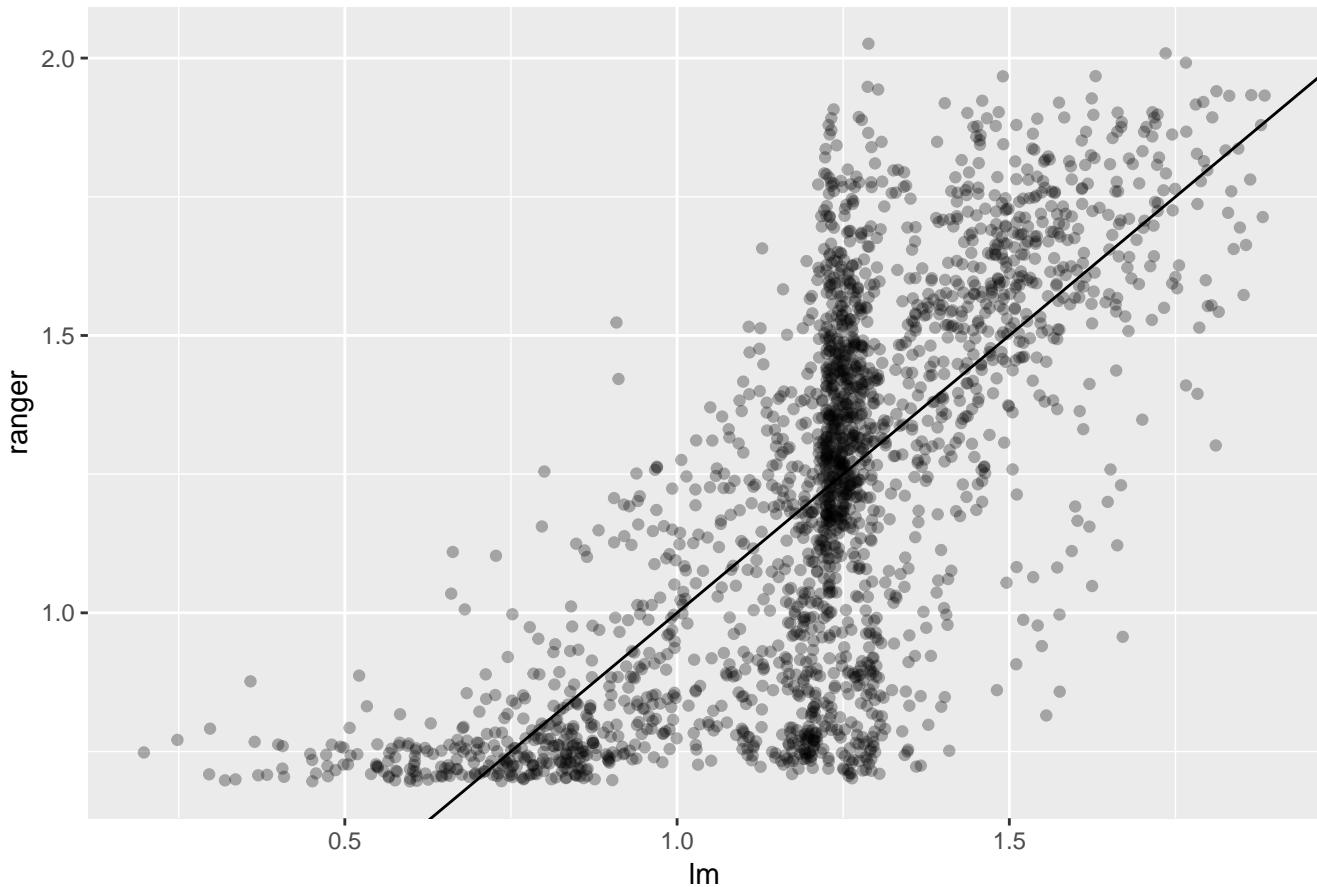
```
compare_models(m1_enet, m3_rf)
```

Correlation: 0.88. t1 RMSE: 0.31. t2 RMSE: 0.26



```
compare_models(m0_lm, m3_rf)
```

Correlation: 0.72. t1 RMSE: 0.37. t2 RMSE: 0.26



Global analysis.

- gain some understanding of a system
 - predictability
 - complexity
 - r2 # Generate hypotheses to test more formally.
- generate hypotheses (variable level)
 - var imp
 - interaction importance
 - ice etc. # Find variable importance for each model

```
varImp(m1_enet)
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : zero-width neighborhood. make span bigger
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at 4.8037
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.0031927
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0
## loess r-squared variable importance
##
## only 20 most important variables shown (out of 47)
##
##                                     Overall
## X9.1_GestationLen_d          100.00
## X5.1_AdultBodyMass_g         70.67
```

```

## X26.4_GR_MidRangeLat_dd      67.42
## X26.3_GR_MinLat_dd          62.73
## X30.2_PET_Mean_mm           61.68
## X26.2_GR_MaxLat_dd          59.09
## X30.1_AET_Mean_mm           56.67
## X23.1_SexualMaturityAge_d   54.73
## X28.2_Temp_Mean_01degC      53.08
## X25.1_WeaningAge_d           52.97
## X17.1_MaxLongevity_m        51.87
## X13.1_AdultHeadBodyLen_mm   51.01
## X28.1_Precip_Mean_mm         49.70
## X5.3_NeonateBodyMass_g       45.56
## X21.1_PopulationDensity_n.km2 39.86
## X5.4_WeaningBodyMass_g       24.52
## X12.2_Terrestriality         23.43
## X26.7_GR_MidRangeLong_dd     23.14
## X26.6_GR_MinLong_dd          21.45
## X3.1_AgeatFirstBirth_d       20.41

varImp(m2_gp)

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : zero-width neighborhood. make span bigger
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at 4.8037
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.0031927
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 0

## loess r-squared variable importance
##
## only 20 most important variables shown (out of 47)
##
##                                     Overall
## X9.1_GestationLen_d            100.00
## X5.1_AdultBodyMass_g           70.67
## X26.4_GR_MidRangeLat_dd        67.42
## X26.3_GR_MinLat_dd            62.73
## X30.2_PET_Mean_mm              61.68
## X26.2_GR_MaxLat_dd             59.09
## X30.1_AET_Mean_mm              56.67
## X23.1_SexualMaturityAge_d      54.73
## X28.2_Temp_Mean_01degC         53.08
## X25.1_WeaningAge_d              52.97
## X17.1_MaxLongevity_m           51.87
## X13.1_AdultHeadBodyLen_mm      51.01
## X28.1_Precip_Mean_mm            49.70
## X5.3_NeonateBodyMass_g          45.56
## X21.1_PopulationDensity_n.km2  39.86
## X5.4_WeaningBodyMass_g          24.52
## X12.2_Terrestriality            23.43
## X26.7_GR_MidRangeLong_dd        23.14
## X26.6_GR_MinLong_dd             21.45
## X3.1_AgeatFirstBirth_d          20.41

varImp(m3_rf)

## ranger variable importance
##

```

```

##   only 20 most important variables shown (out of 47)
##
##                               Overall
## X9.1_GestationLen_d      100.000
## X5.1_AdultBodyMass_g     58.065
## X8.1_AdultForearmLen_mm 26.915
## X26.4_GR_MidRangeLat_dd 25.800
## X30.2_PET_Mean_mm       22.957
## X30.1_AET_Mean_mm       20.838
## X26.2_GR_MaxLat_dd      19.898
## X12.2_Terrestriality    15.658
## X26.1_GR_Area_km2       11.726
## X28.2_Temp_Mean_01degC  11.042
## X25.1_WeaningAge_d      10.726
## X12.1_HabitatBreadth    10.305
## X26.3_GR_MinLat_dd      9.542
## X28.1_Precip_Mean_mm    8.629
## X27.2_HuPopDen_Mean_n.km2 7.841
## X13.1_AdultHeadBodyLen_mm 7.649
## X17.1_MaxLongevity_m    7.229
## X23.1_SexualMaturityAge_d 7.223
## X26.6_GR_MinLong_dd     7.097
## X26.5_GR_MaxLong_dd     6.010

```

Now plot some functional forms

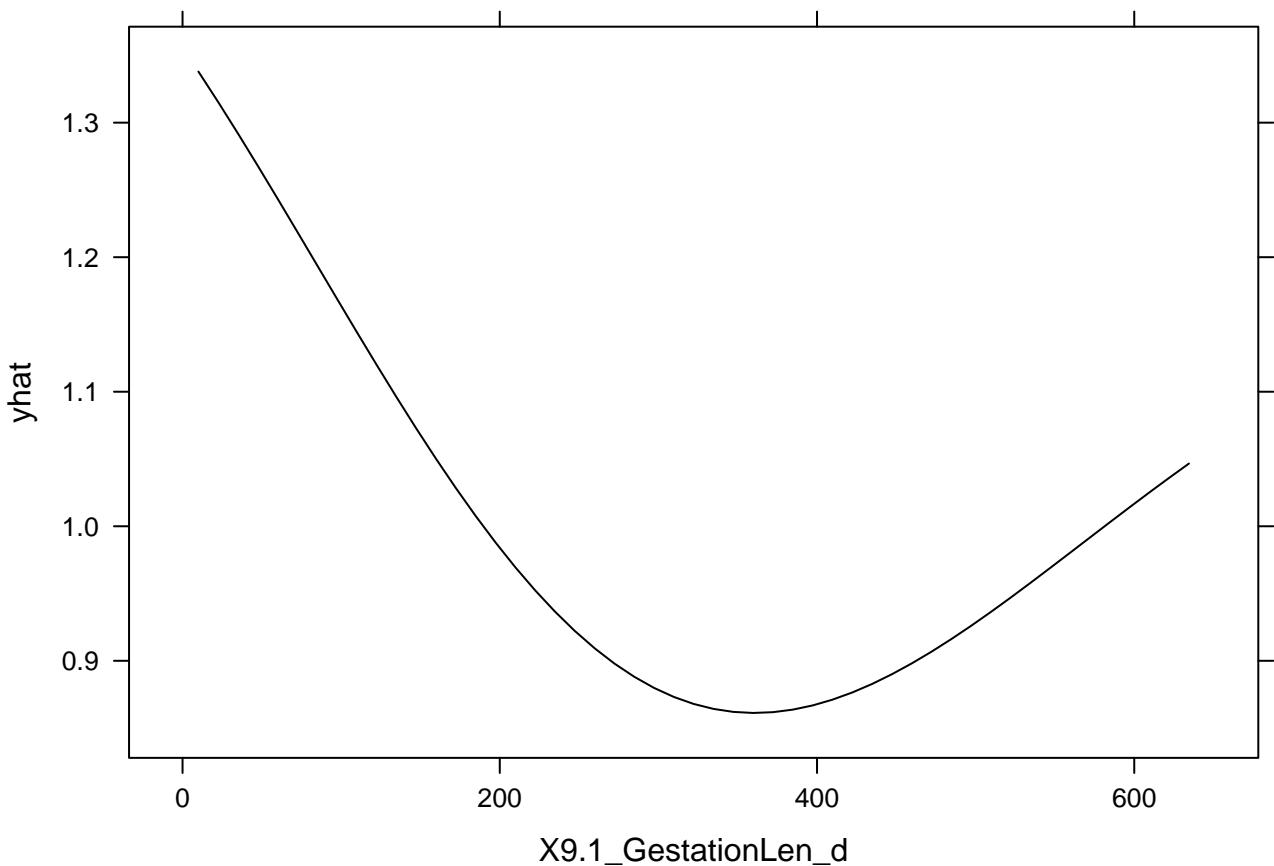
```

partial(m2_gp,
  pred.var = c('X9.1_GestationLen_d'),
  parallel = TRUE, plot = TRUE)

## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'

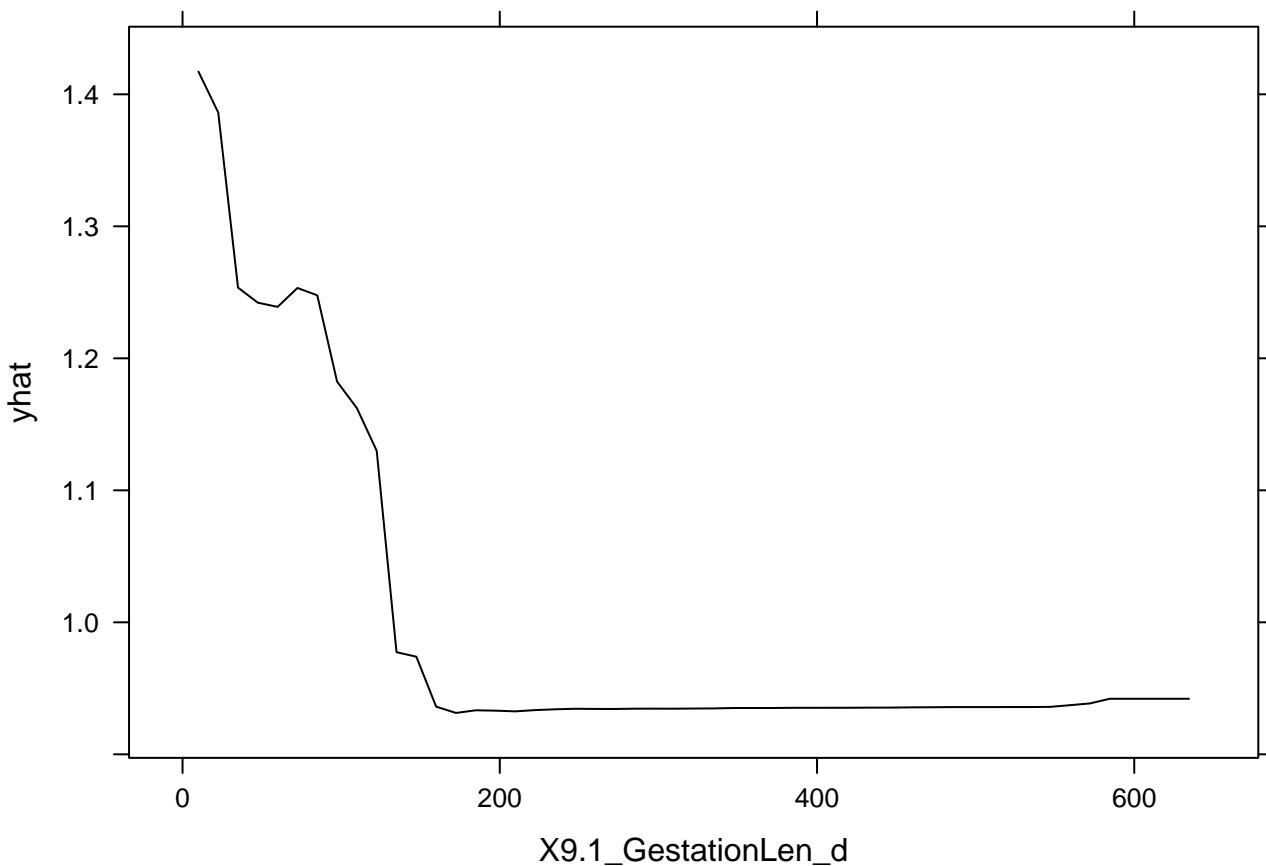
## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'

```



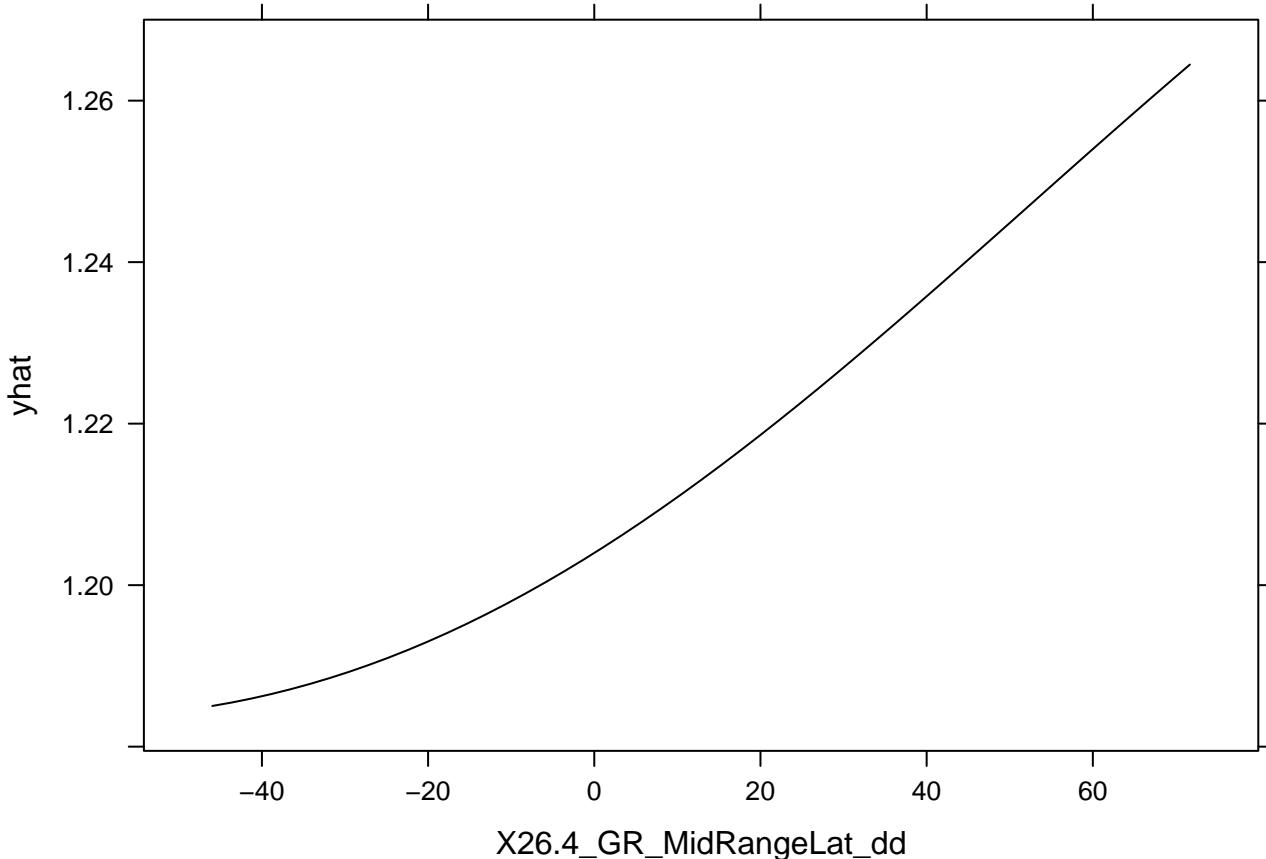
```
partial(m3_rf,
       pred.var = c('X9.1_GestationLen_d'),
       parallel = TRUE, plot = TRUE)

## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
```



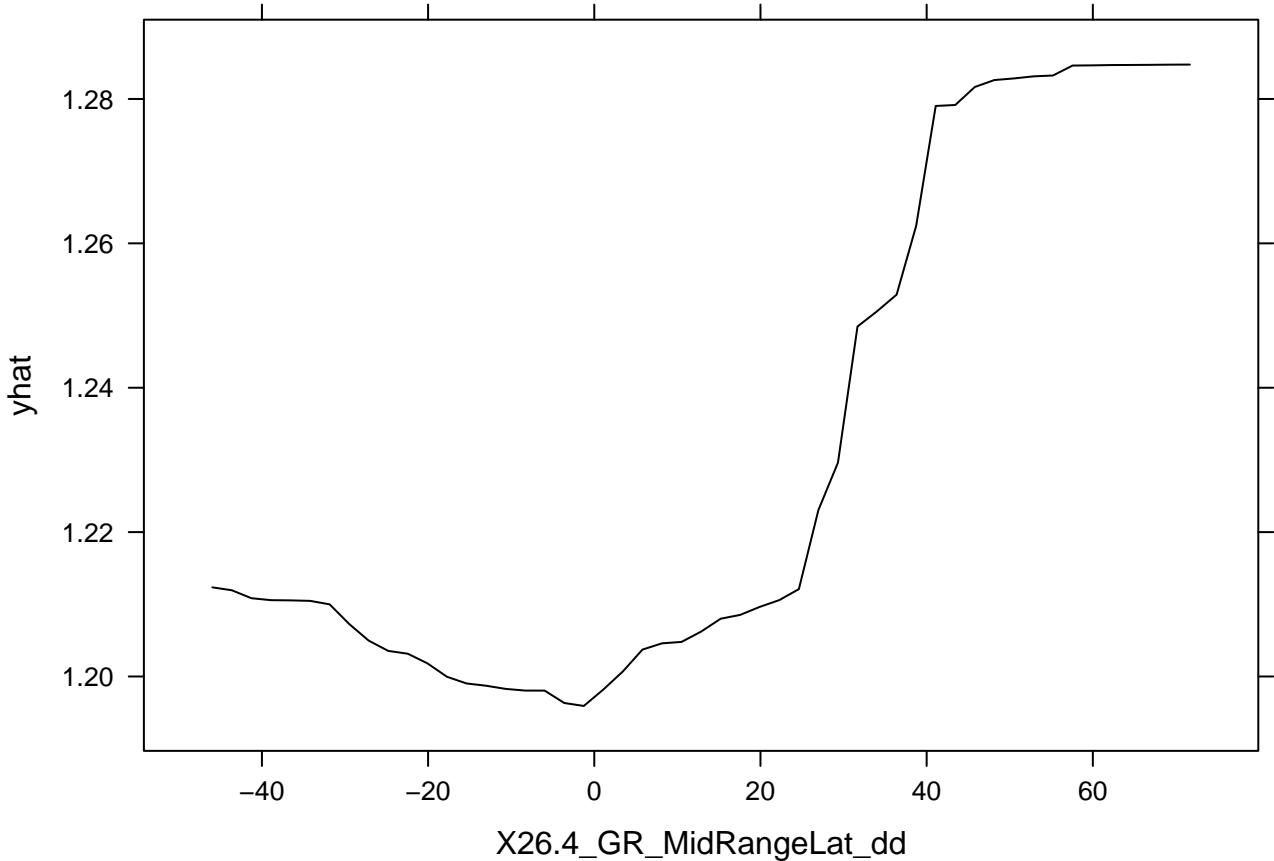
```
partial(m2_gp,
       pred.var = c('X26.4_GR_MidRangeLat_dd'),
       parallel = TRUE, plot = TRUE)

## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
```



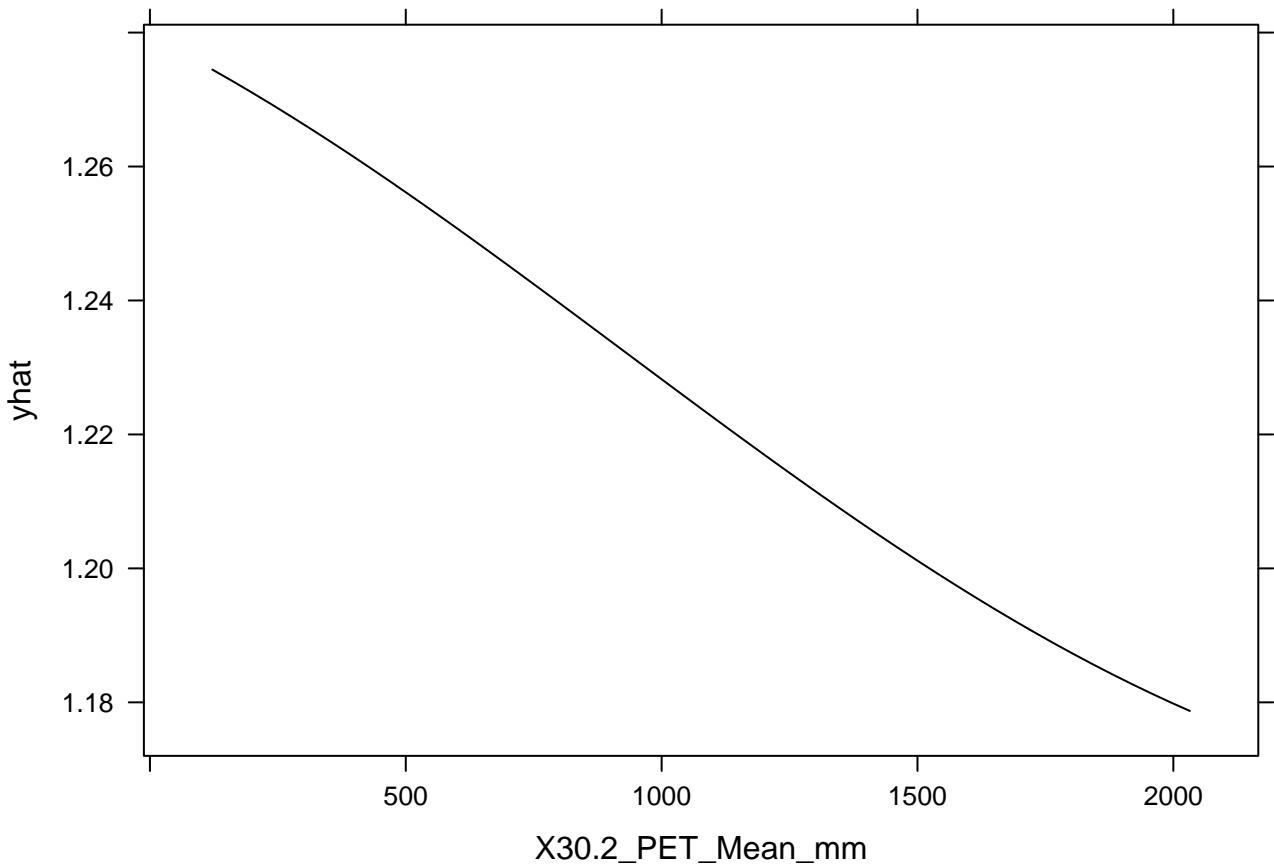
```
partial(m3_rf,
       pred.var = c('X26.4_GR_MidRangeLat_dd'),
       parallel = TRUE, plot = TRUE)

## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
```



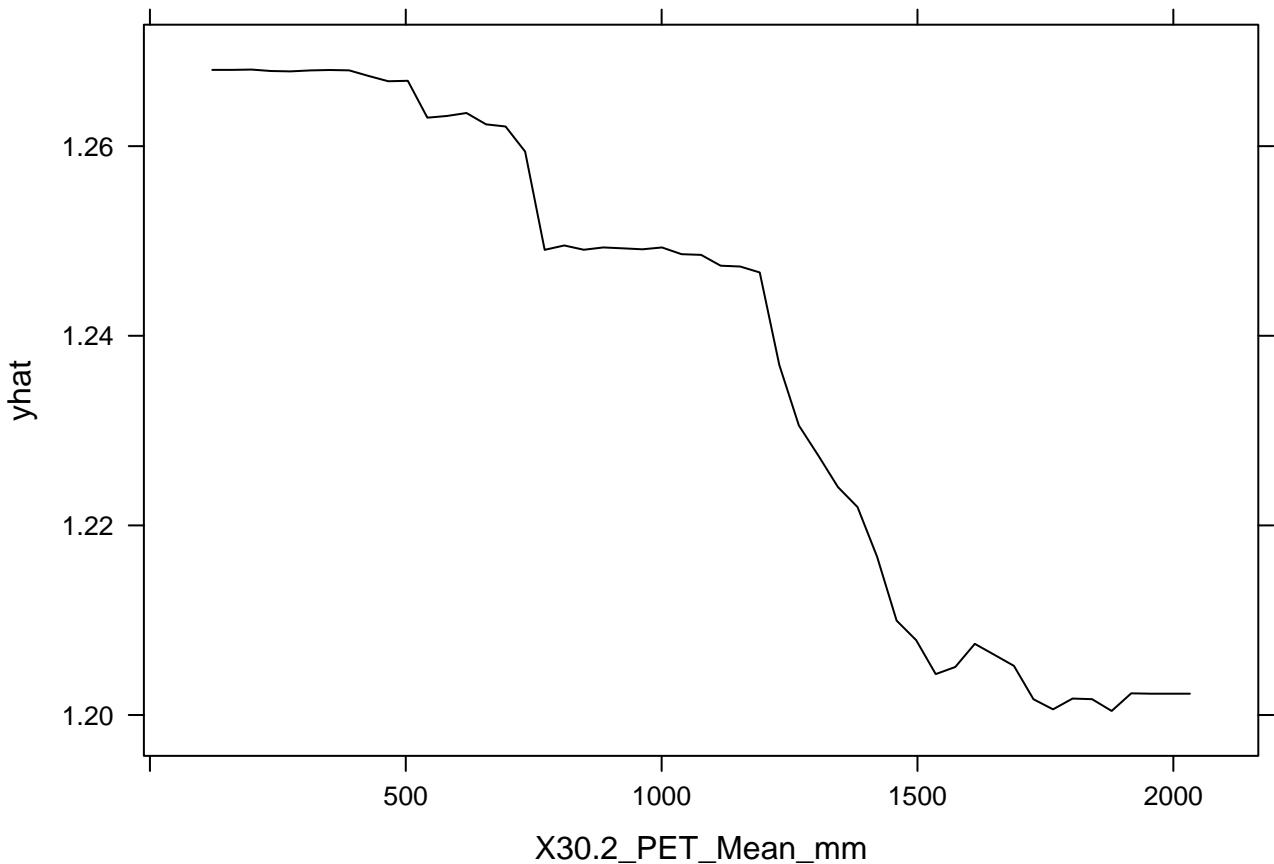
```
partial(m2_gp,
       pred.var = c('X30.2_PET_Mean_mm'),
       parallel = TRUE, plot = TRUE)

## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
```



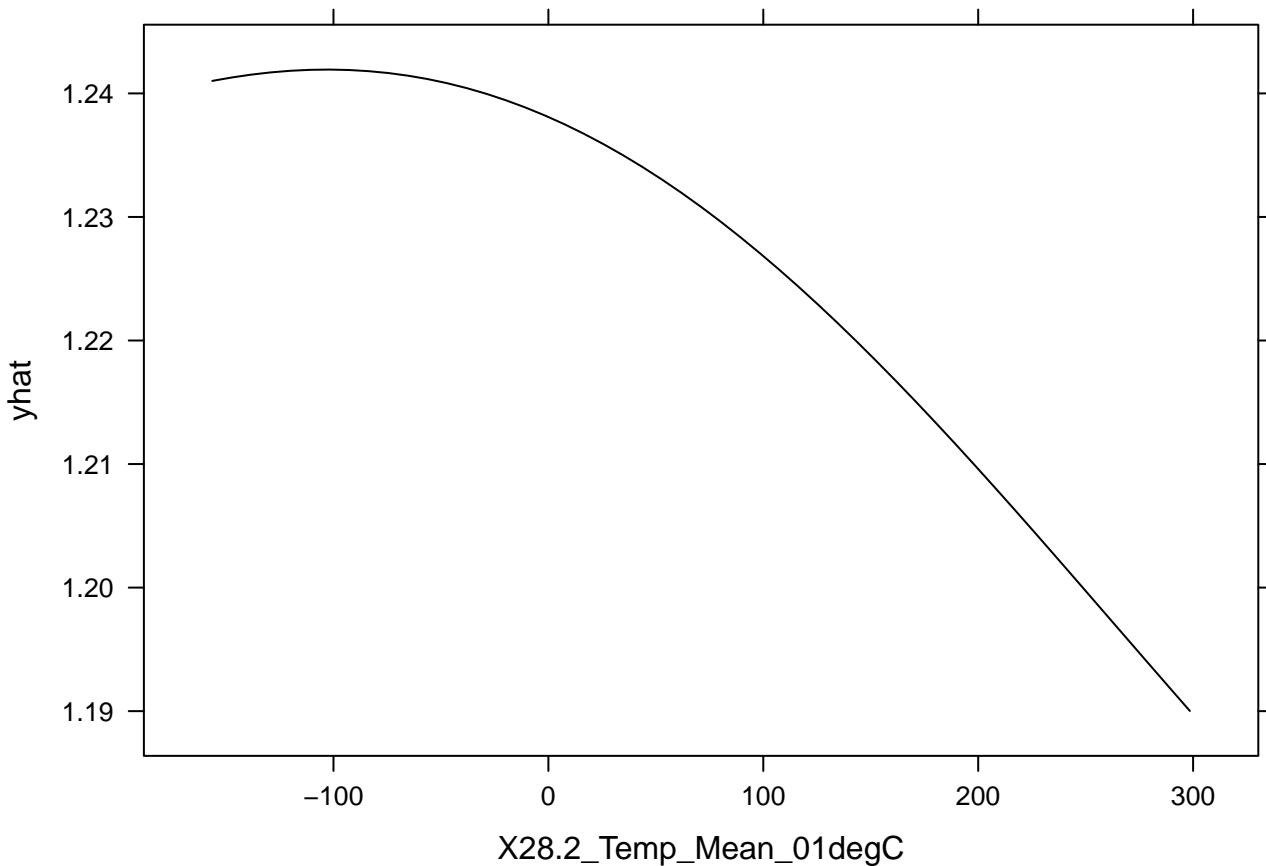
```
partial(m3_rf,
       pred.var = c('X30.2_PET_Mean_mm'),
       parallel = TRUE, plot = TRUE)

## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
```



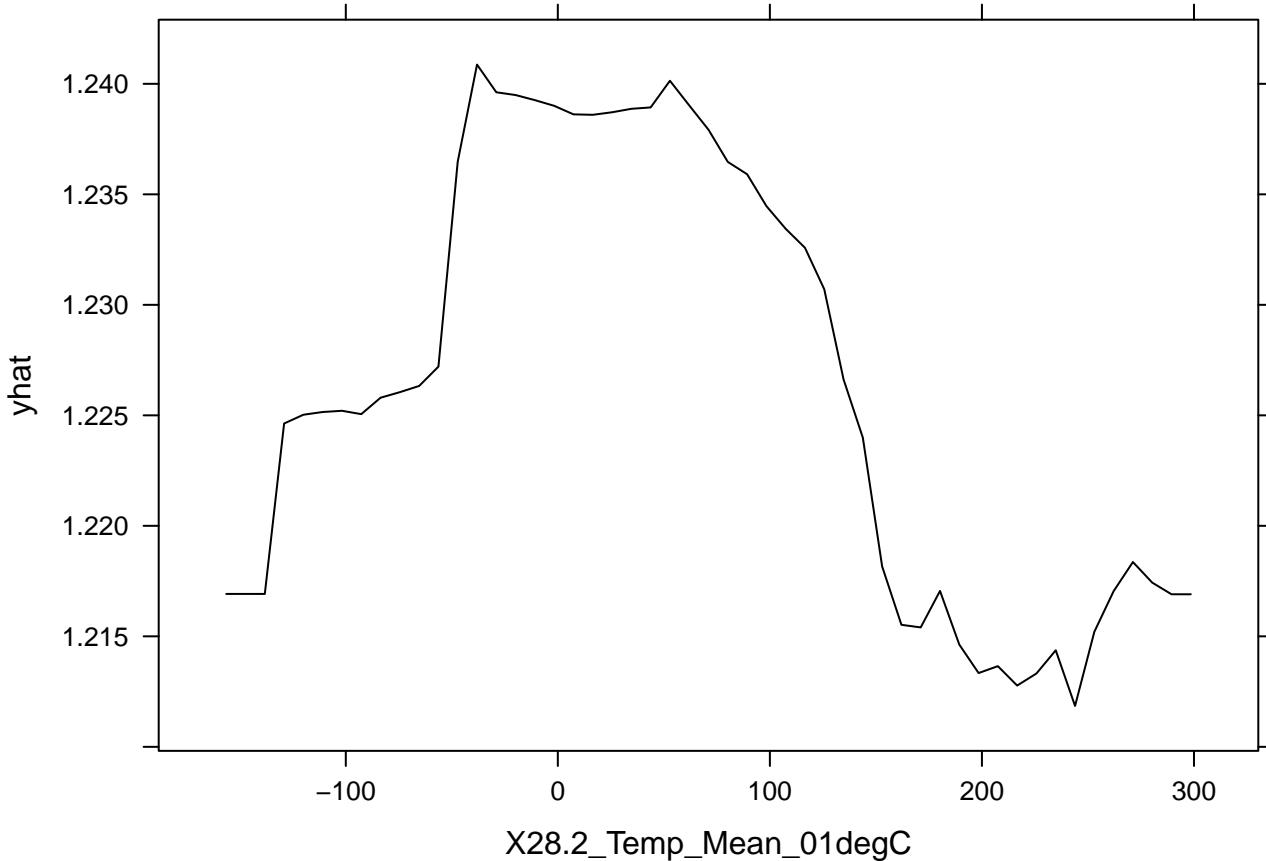
```
partial(m2_gp,
       pred.var = c('X28.2_Temp_Mean_01degC'),
       parallel = TRUE, plot = TRUE)

## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
```



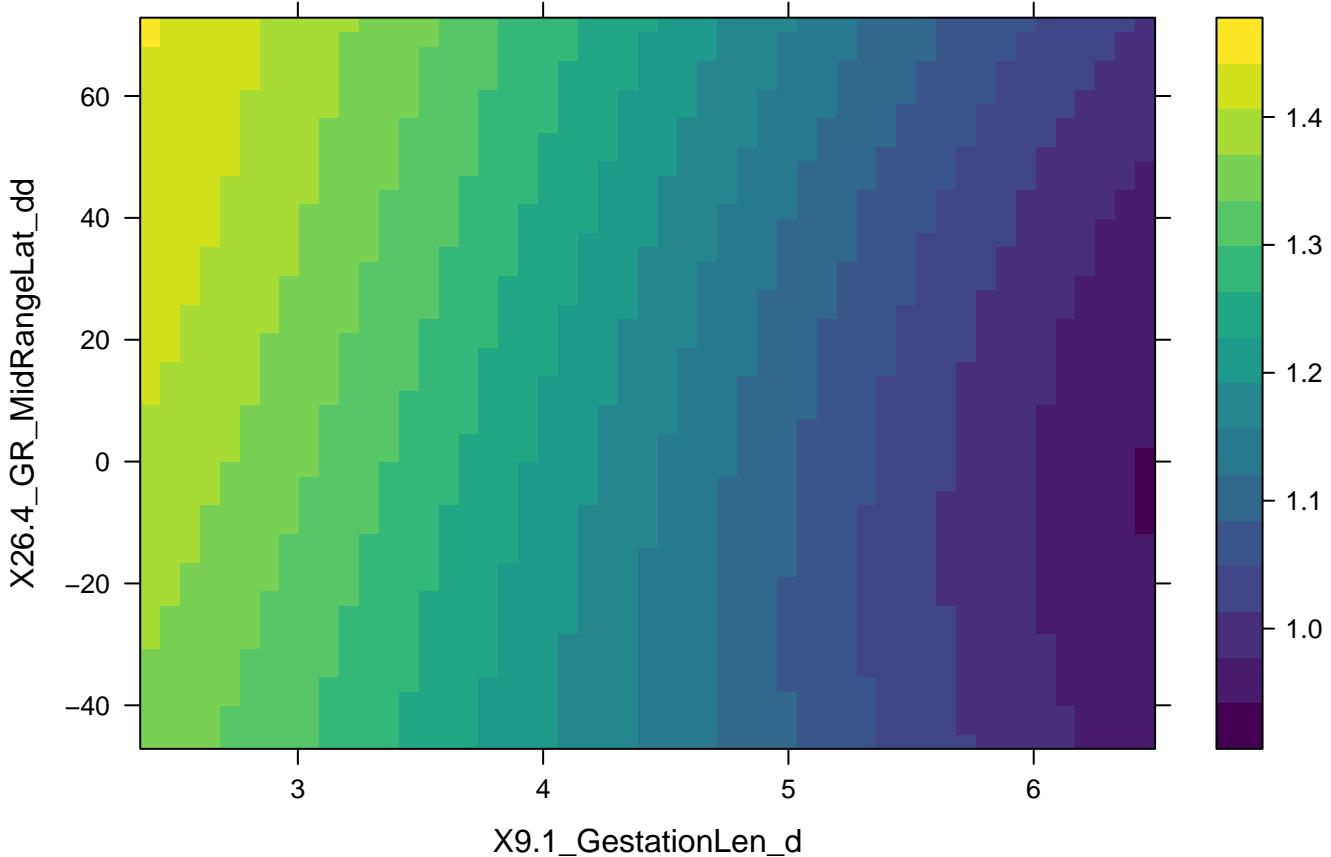
```
partial(m3_rf,
       pred.var = c('X28.2_Temp_Mean_01degC'),
       parallel = TRUE, plot = TRUE)

## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
```



```
partial(m2_gp,
       pred.var = c('X9.1_GestationLen_d', 'X26.4_GR_MidRangeLat_dd'),
       parallel = TRUE, plot = TRUE)

## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
```

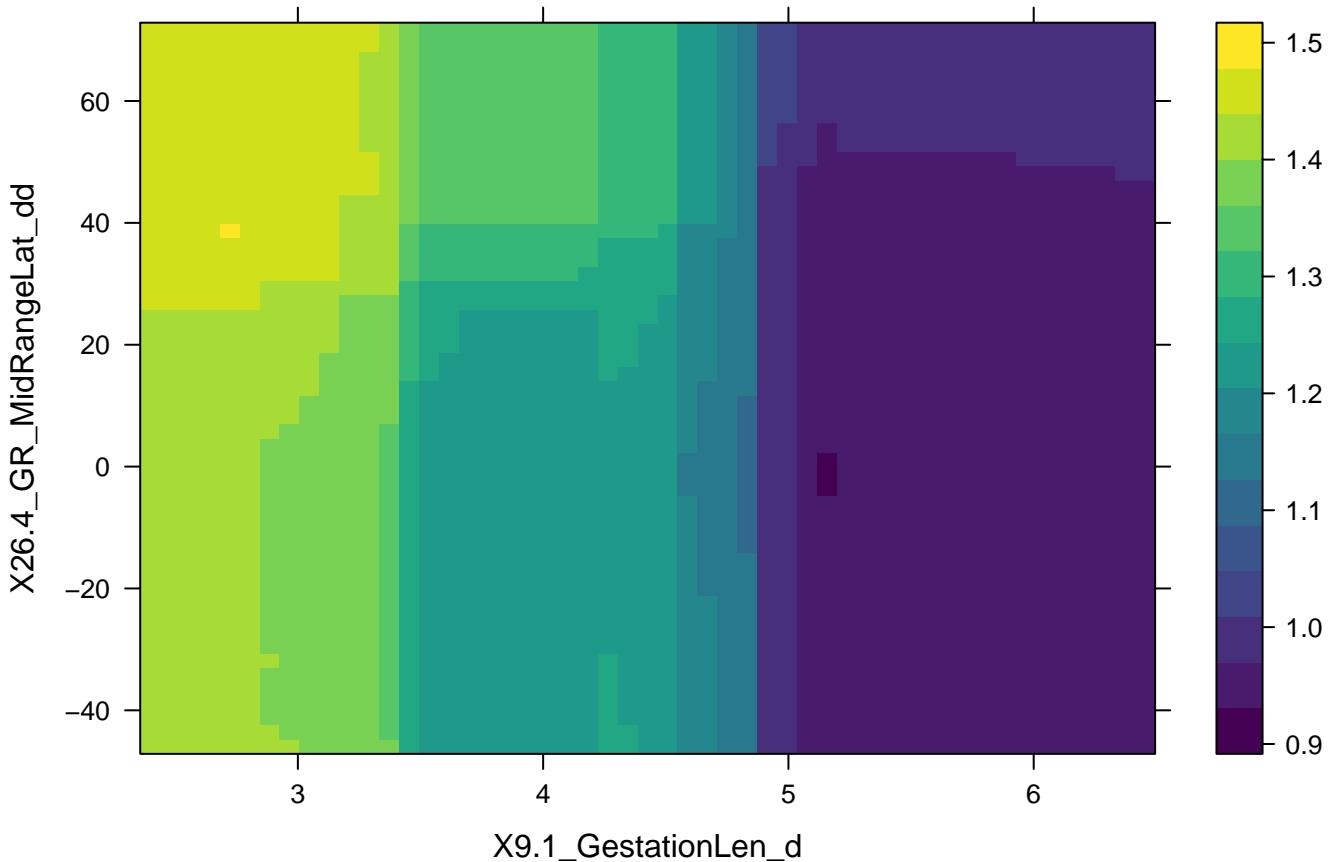


```

partial(m3_rf,
  pred.var = c('X9.1_GestationLen_d', 'X26.4_GR_MidRangeLat_dd'),
  parallel = TRUE, plot = TRUE)

## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'

```

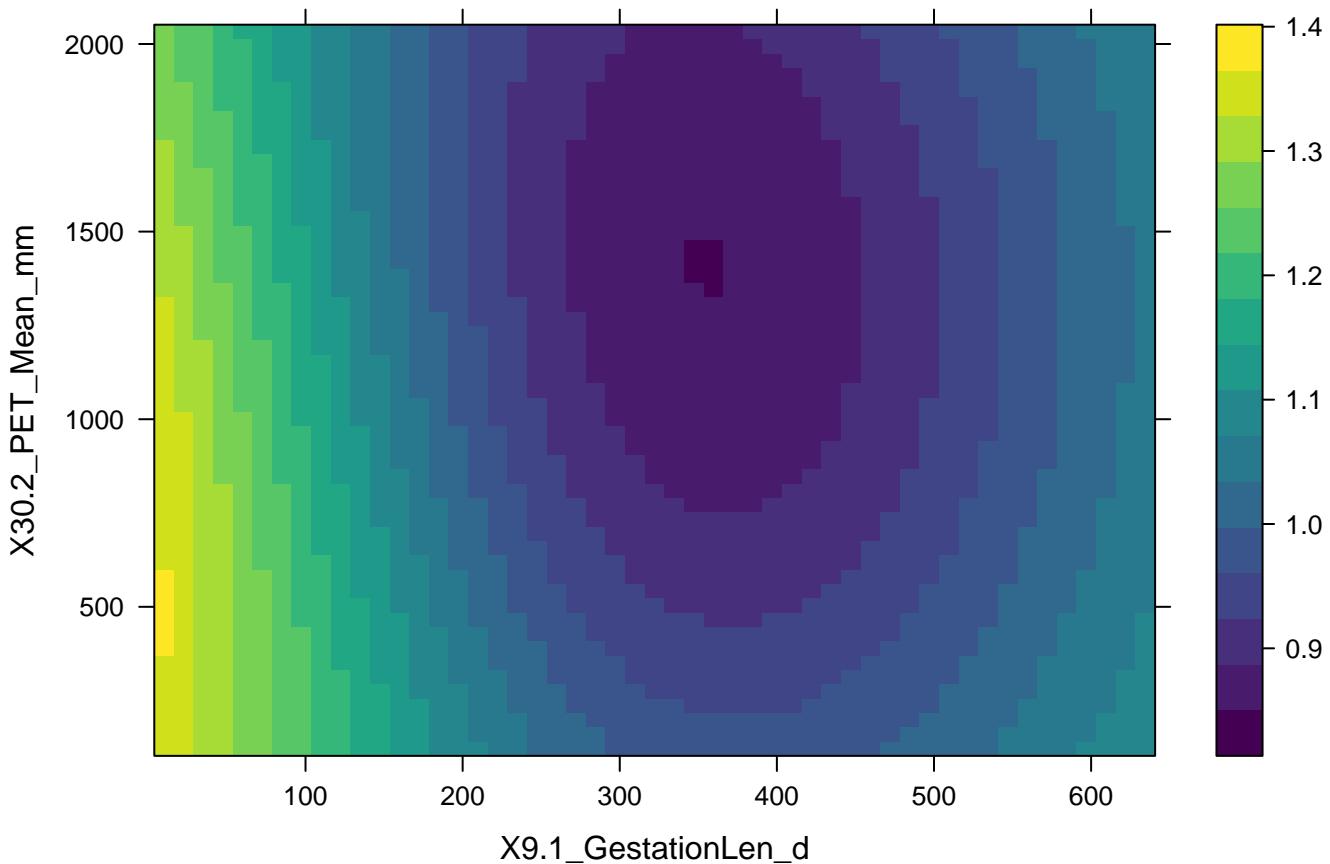


```

partial(m2_gp,
       pred.var = c('X9.1_GestationLen_d', 'X30.2_PET_Mean_mm'),
       parallel = TRUE, plot = TRUE)

## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'

```

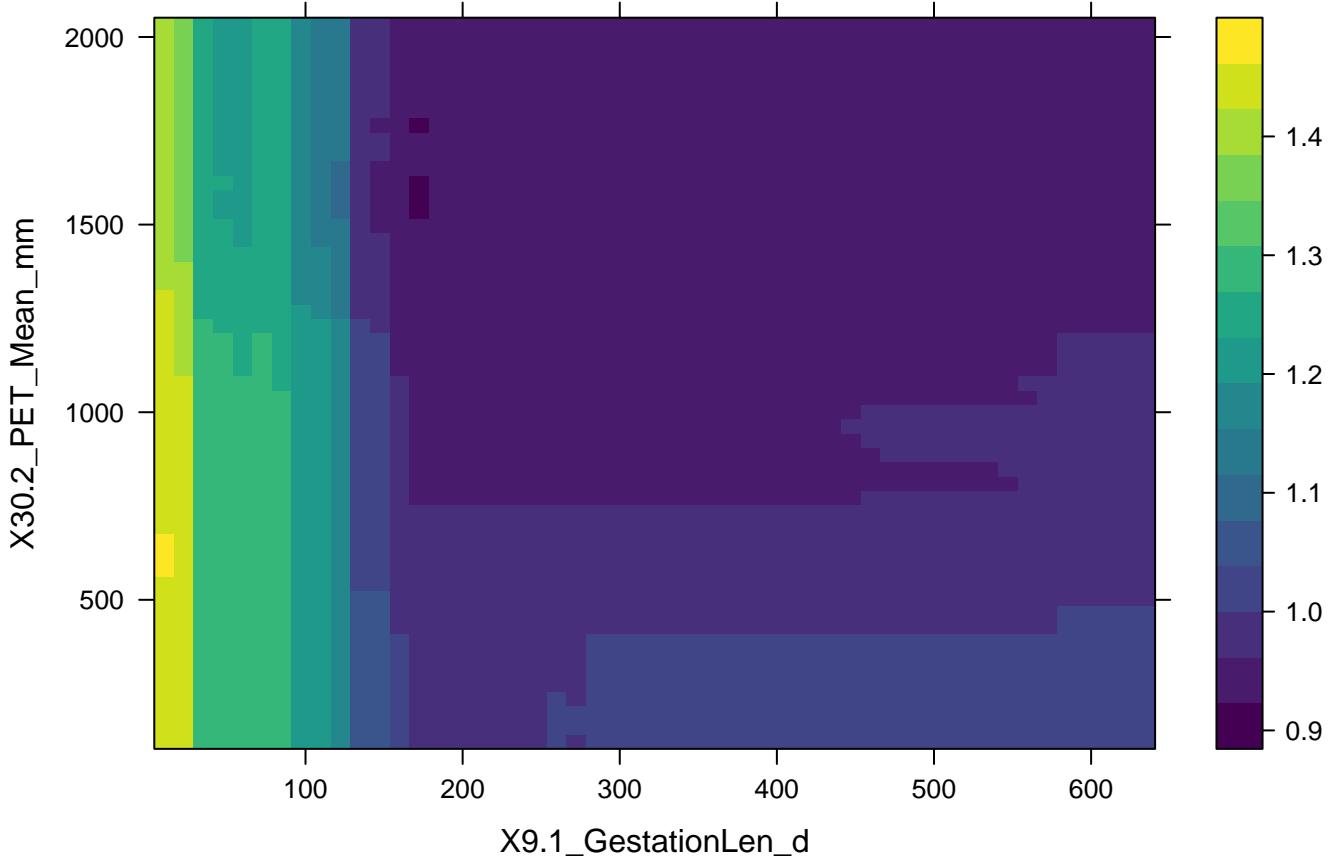


```

partial(m3_rf,
  pred.var = c('X9.1_GestationLen_d', 'X30.2_PET_Mean_mm'),
  parallel = TRUE, plot = TRUE)

## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'
## Warning: <anonymous>: ... may be used in an incorrect context: '.fun(piece, ...)'

```



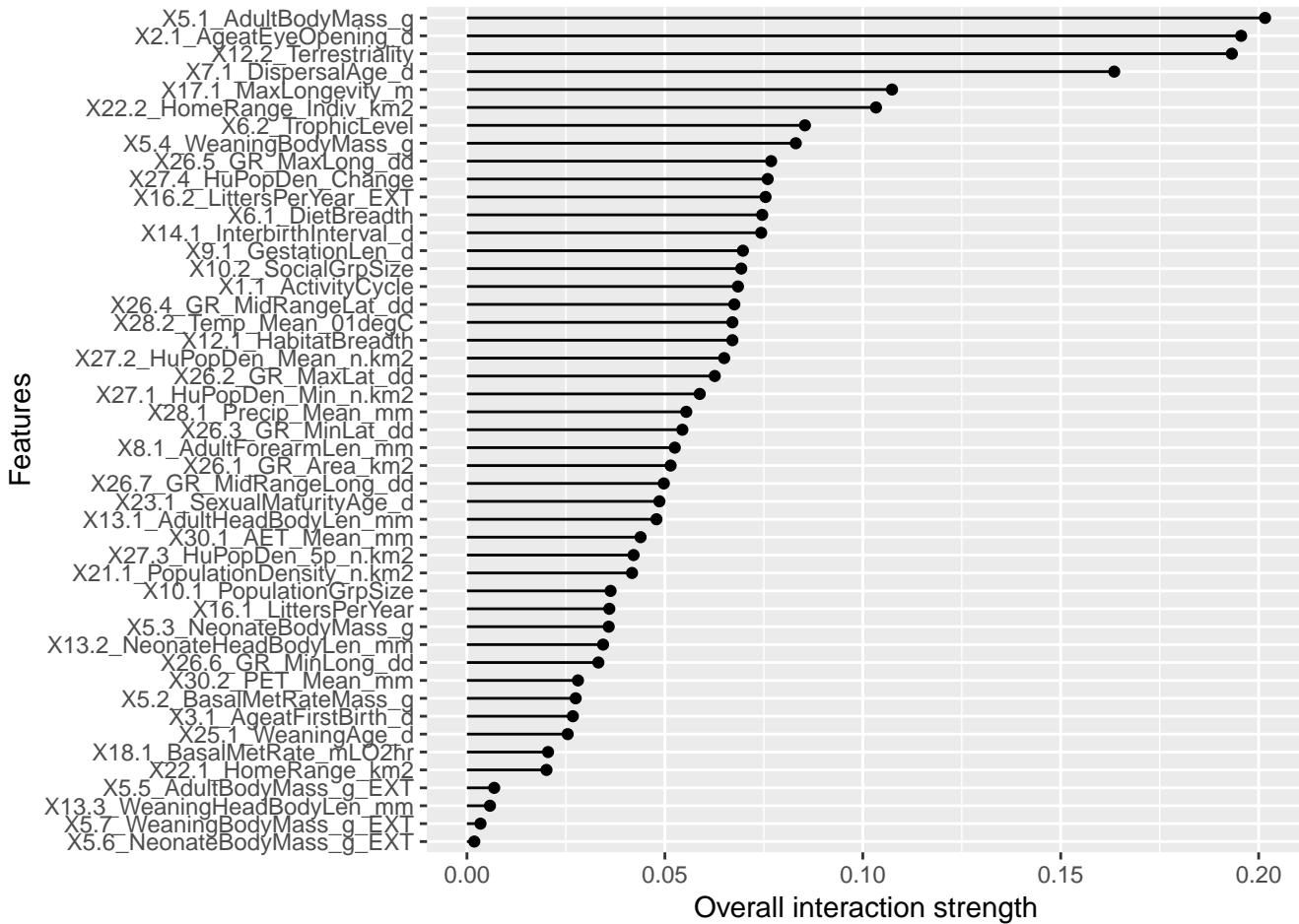
Test for interactions

```
predictor_gp = Predictor$new(m2_gp, data = dplyr::select(p_impute, -y), y = p_impute$y)
predictor_rf = Predictor$new(m3_rf, data = dplyr::select(p_impute, -y), y = p_impute$y)
```

```
interact_gp = Interaction$new(predictor_gp)
interact_gp$results %>% arrange(desc(.interaction)) %>% head
```

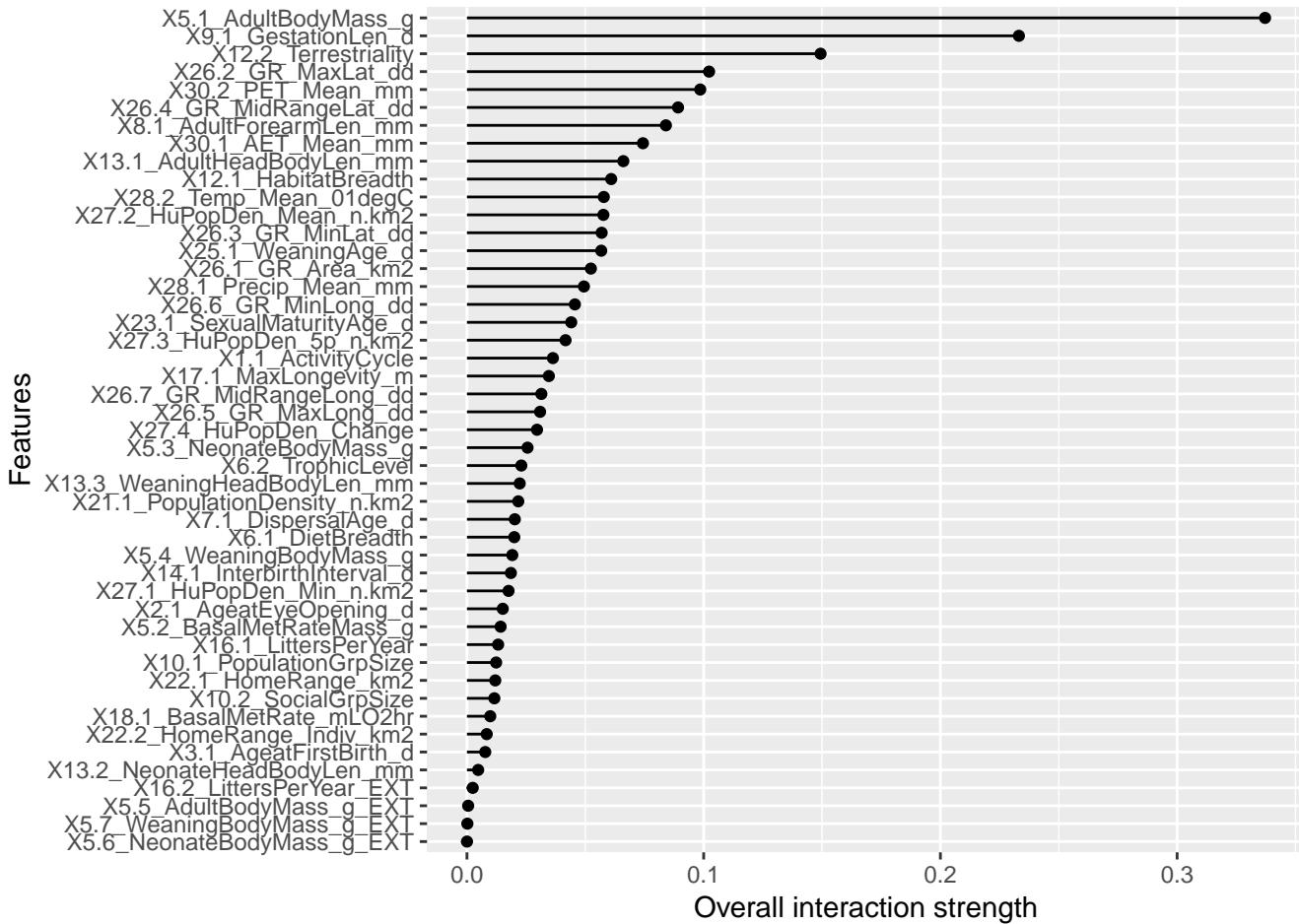
	.feature	.interaction
## 1	X5.1_AdultBodyMass_g	0.2016154
## 2	X2.1_AgeatEyeOpening_d	0.1955797
## 3	X12.2_Terrestriality	0.1932190
## 4	X7.1_DispersalAge_d	0.1635105
## 5	X17.1_MaxLongevity_m	0.1073744
## 6	X22.2_HomeRange_Indiv_km2	0.1033276

```
plot(interact_gp)
```



```
interact_rf = Interaction$new(predictor_rf)
interact_rf$results %>% arrange(desc(.interaction)) %>% head
```

```
##          .feature .interaction
## 1      X5.1_AdultBodyMass_g    0.3371445
## 2      X9.1_GestationLen_d    0.2331598
## 3      X12.2_Terrestriality   0.1493818
## 4      X26.2_GR_MaxLat_dd    0.1022850
## 5      X30.2_PET_Mean_mm     0.0985487
## 6 X26.4_GR_MidRangeLat_dd    0.0891687
```

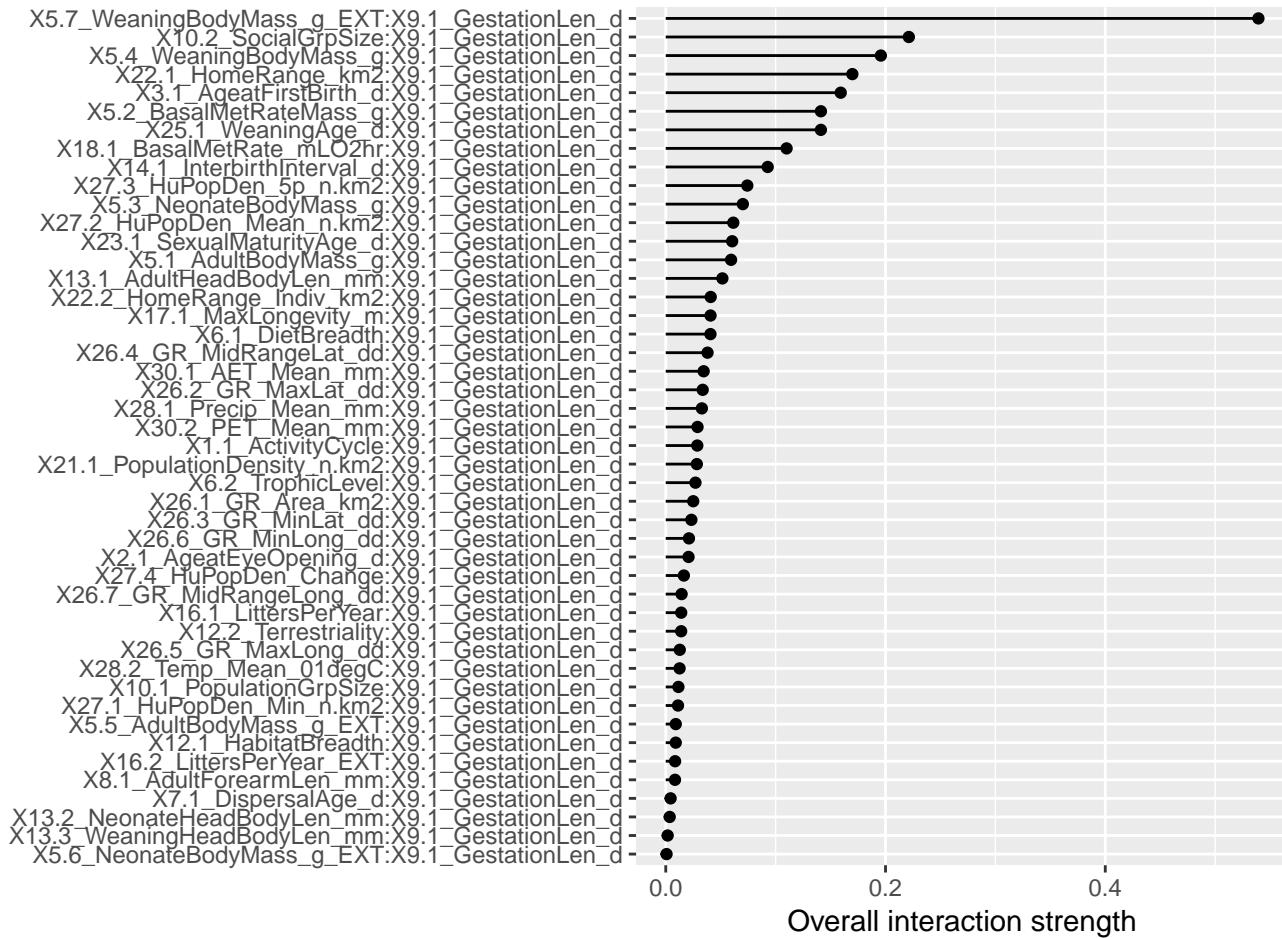


```
interact_ges_gp = Interaction$new(predictor_gp, feature = "X9.1_GestationLen_d")
interact_ges_gp$results %>% arrange(.interaction) %>% head
```

```
## .feature .interaction
## 1 X5.7_WeaningBodyMass_g_EXT:X9.1_GestationLen_d 0.5393717
## 2 X10.2_SocialGrpSize:X9.1_GestationLen_d 0.2212288
## 3 X5.4_WeaningBodyMass_g:X9.1_GestationLen_d 0.1957718
## 4 X22.1_HomeRange_km2:X9.1_GestationLen_d 0.1698334
## 5 X3.1_AgeatFirstBirth_d:X9.1_GestationLen_d 0.1592361
## 6 X5.2_BasalMetRateMass_g:X9.1_GestationLen_d 0.1412126
```

```
plot(interact_ges_gp)
```

Features

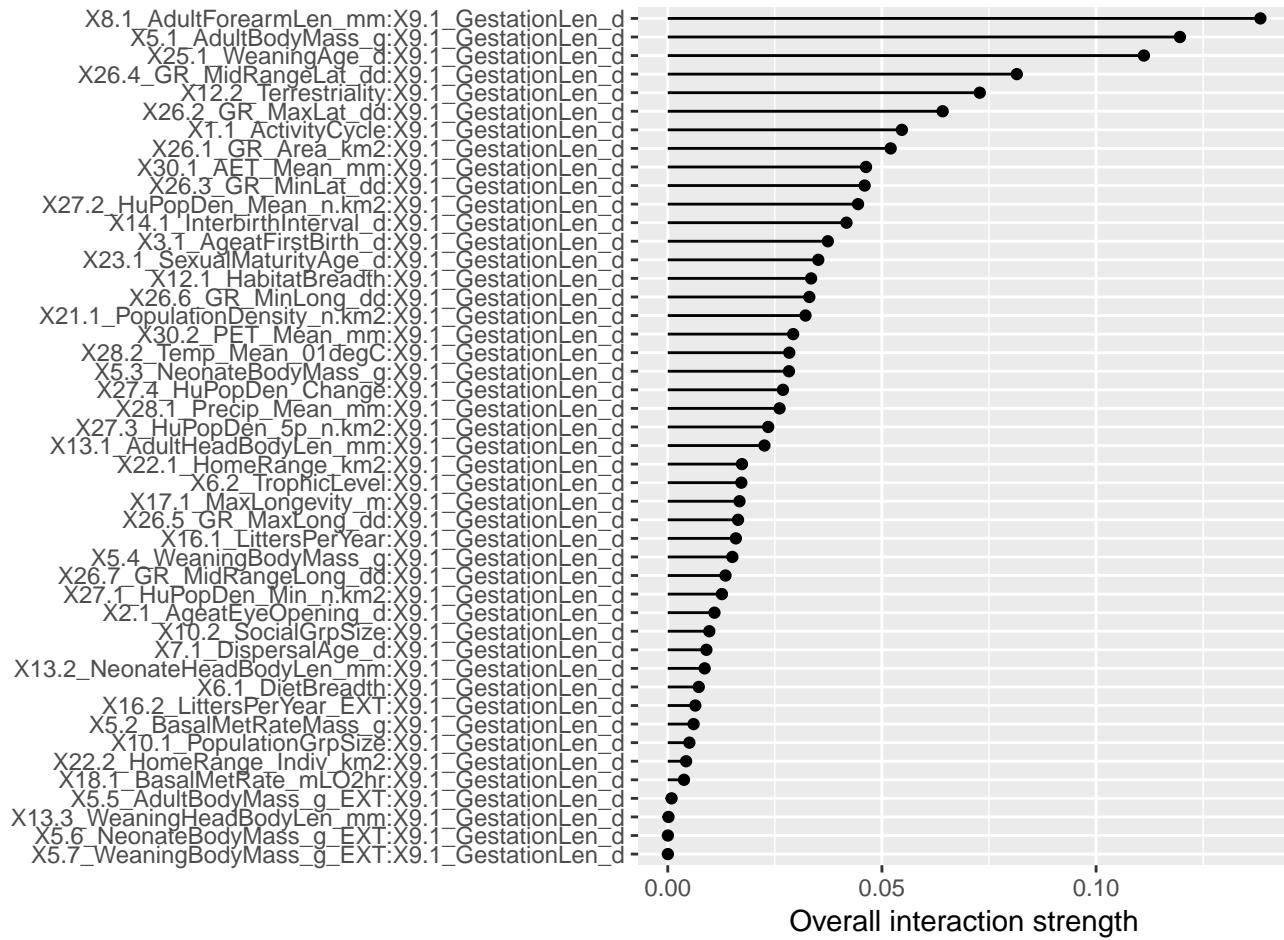


```
interact_ges_rf = Interaction$new(predictor_rf, feature = "X9.1_GestationLen_d")
interact_ges_rf$results %>% arrange(.interaction) %>% head
```

```
##                                     .feature .interaction
## 1 X8.1_AdultForearmLen_mm:X9.1_GestationLen_d 0.13836986
## 2 X5.1_AdultBodyMass_g:X9.1_GestationLen_d 0.11957721
## 3 X25.1_WeaningAge_d:X9.1_GestationLen_d 0.11116702
## 4 X26.4_GR_MidRangeLat_dd:X9.1_GestationLen_d 0.08149829
## 5 X12.2_Terrestriality:X9.1_GestationLen_d 0.07283942
## 6 X26.2_GR_MaxLat_dd:X9.1_GestationLen_d 0.06417600
```

```
plot(interact_ges_rf)
```

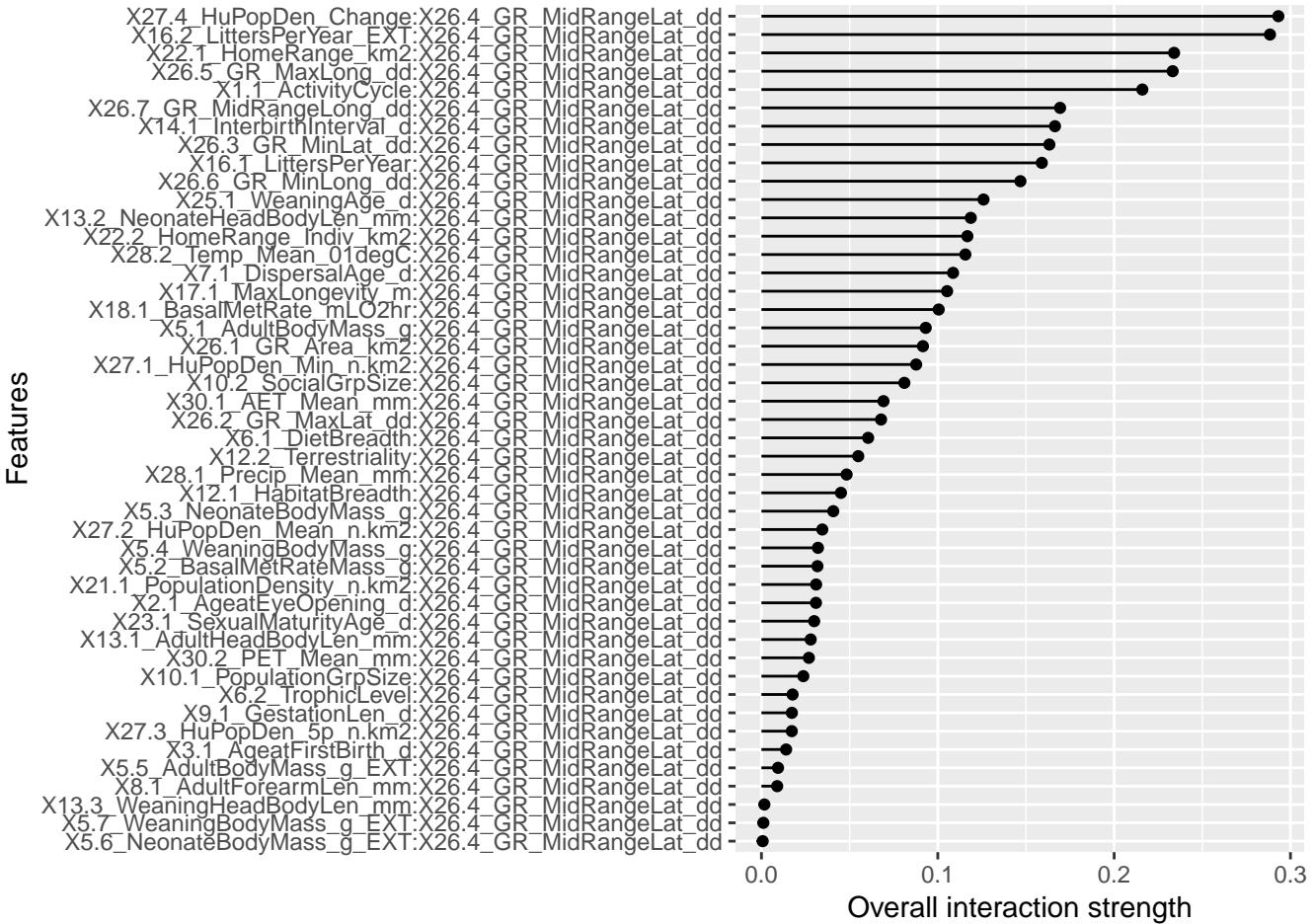
Features



```
interact_lat_gp = Interaction$new(predictor_gp, feature = "X26.4_GR_MidRangeLat_dd")
interact_lat_gp$results %>% arrange(.interaction) %>% head
```

```
##                                     .feature .interaction
## 1      X27.4_HuPopDen_Change:X26.4_GR_MidRangeLat_dd  0.2931253
## 2      X16.2_LittersPerYear_EXT:X26.4_GR_MidRangeLat_dd  0.2883728
## 3      X22.1_HomeRange_km2:X26.4_GR_MidRangeLat_dd  0.2339927
## 4      X26.5_GR_MaxLong_dd:X26.4_GR_MidRangeLat_dd  0.2331797
## 5      X1.1_ActivityCycle:X26.4_GR_MidRangeLat_dd  0.2159893
## 6 X26.7_GR_MidRangeLong_dd:X26.4_GR_MidRangeLat_dd  0.1693449
```

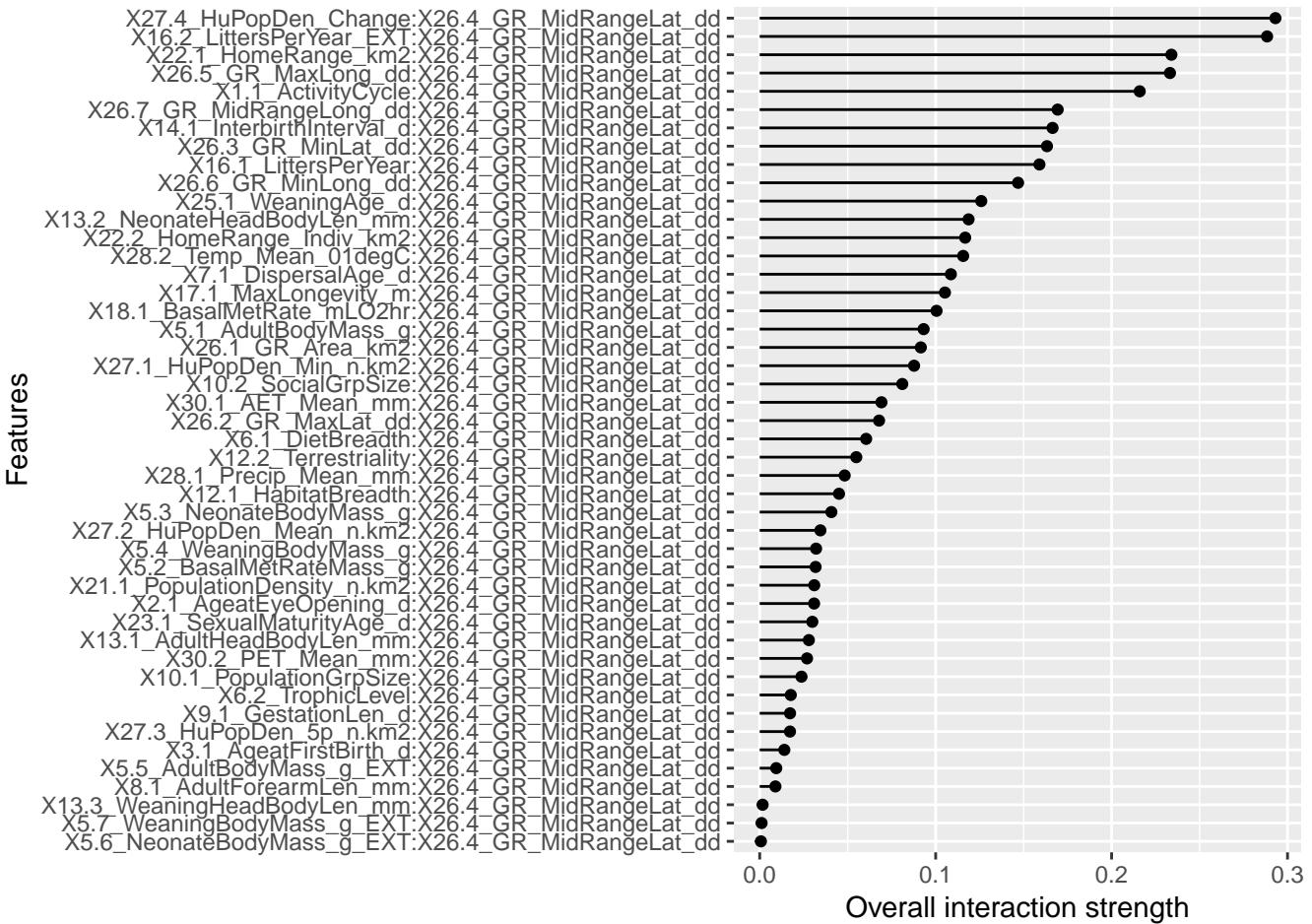
```
plot(interact_lat_gp)
```



```
interact_lat_rf = Interaction$new(predictor_rf, feature = "X26.4_GR_MidRangeLat_dd")
interact_lat_rf$results %>% arrange(desc(.interaction)) %>% head
```

```
##          .feature .interaction
## 1  X8.1_AdultForearmLen_mm:X26.4_GR_MidRangeLat_dd    0.1465875
## 2  X5.1_AdultBodyMass_g:X26.4_GR_MidRangeLat_dd     0.1254077
## 3  X28.1_Precip_Mean_mm:X26.4_GR_MidRangeLat_dd    0.1199153
## 4 X23.1_SexualMaturityAge_d:X26.4_GR_MidRangeLat_dd   0.1097579
## 5  X17.1_MaxLongevity_m:X26.4_GR_MidRangeLat_dd     0.1081483
## 6  X9.1_GestationLen_d:X26.4_GR_MidRangeLat_dd      0.0948204
```

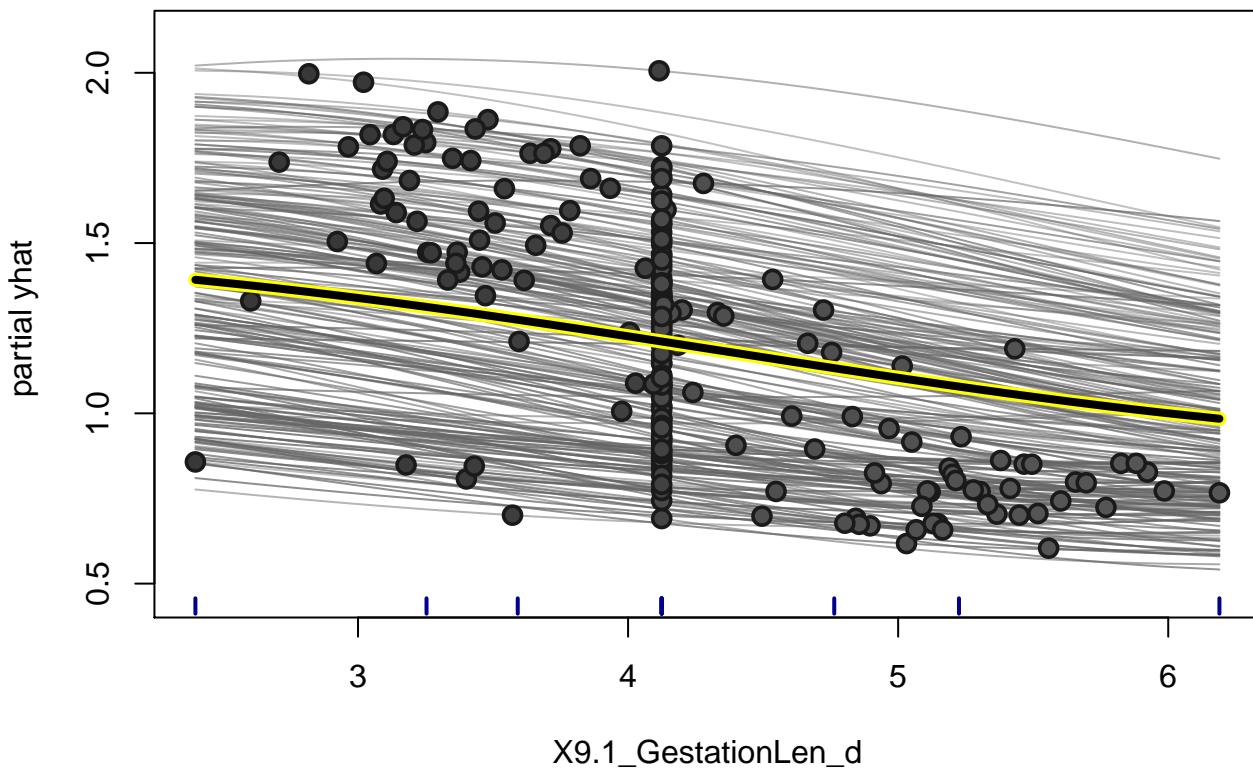
```
plot(interact_lat_gp)
```



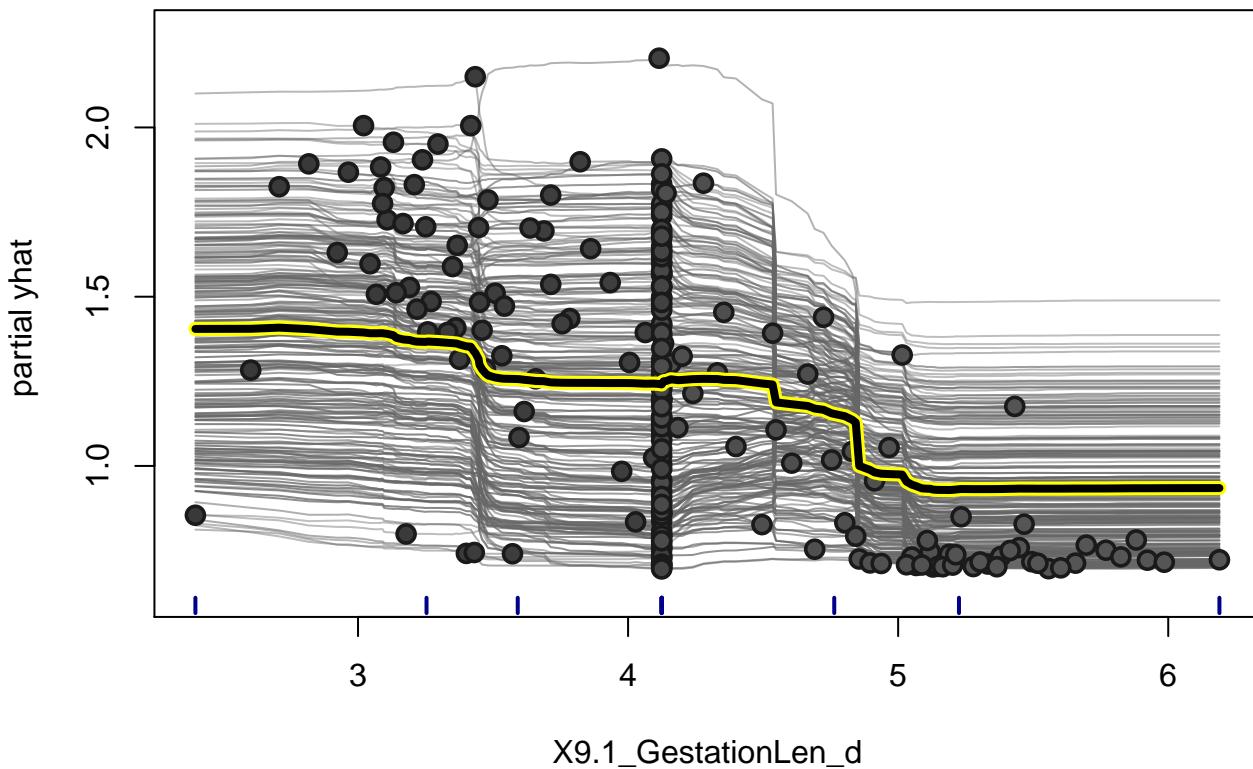
ICE plots

```
m2_gp_ice <- ice(m2_gp, p_impute, p_impute$y, 'X9.1_GestationLen_d', frac_to_build = 0.1)

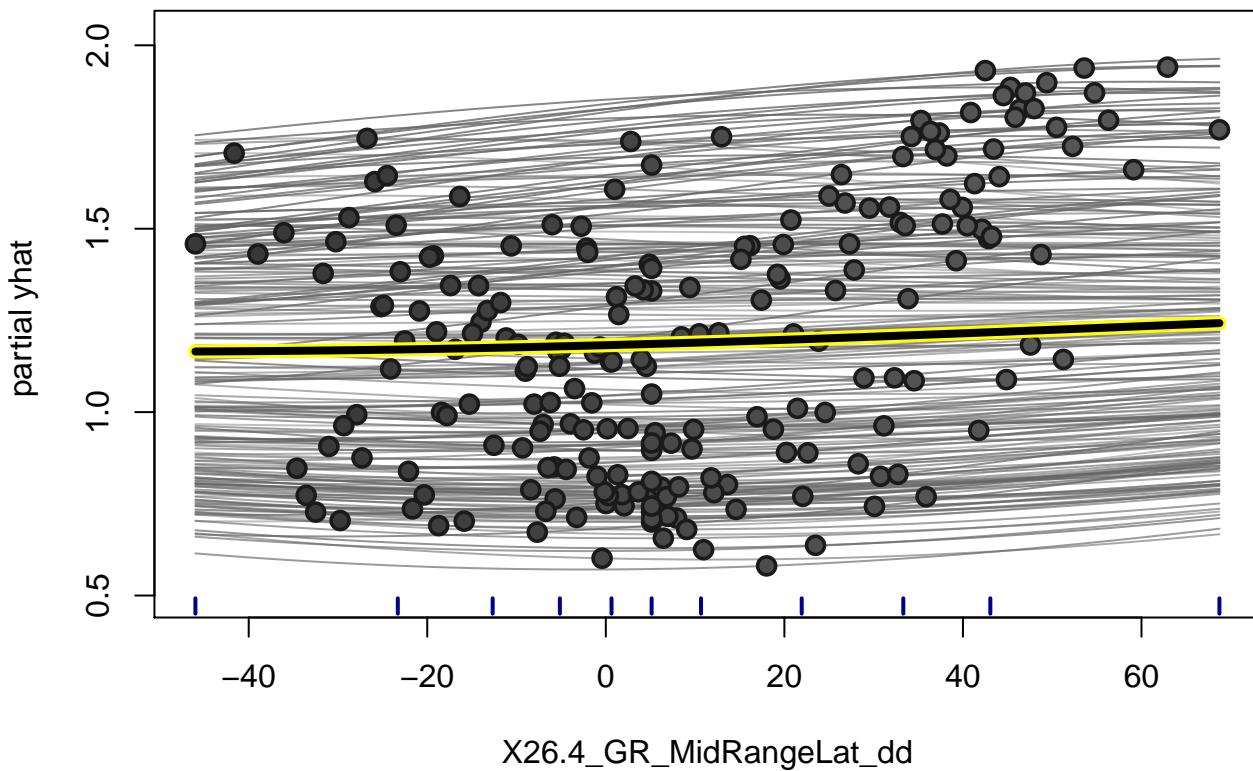
## .....
plot(m2_gp_ice)
```



```
m3_rf_ice <- ice(m3_rf, p_impute, p_impute$y, 'X9.1_GestationLen_d', frac_to_build = 0.1)
## .....
plot(m3_rf_ice)
```



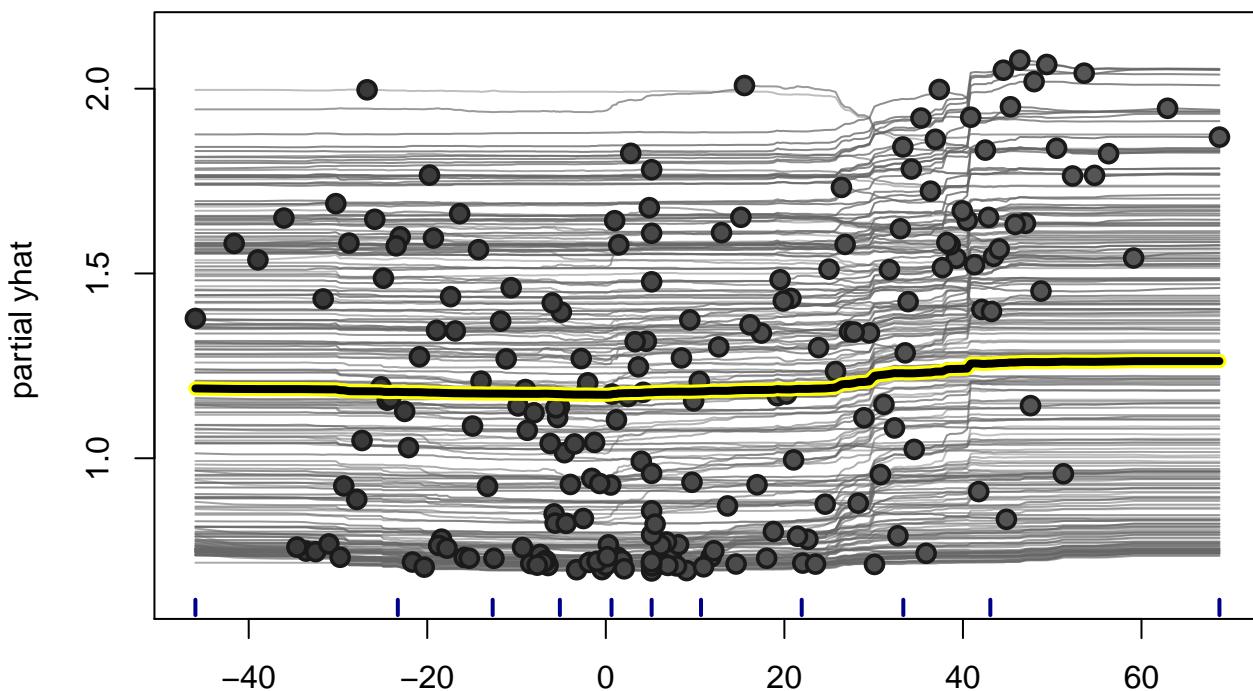
```
m2_gp_ice_lat <- ice(m2_gp, p_impute, p_impute$y, 'X26.4_GR_MidRangeLat_dd', frac_to_build = 0.1)
## .....
plot(m2_gp_ice_lat)
```



X26.4_GR_MidRangeLat_dd

```
m3_rf_ice_lat <- ice(m3_rf, p_impute, p_impute$y, 'X26.4_GR_MidRangeLat_dd', frac_to_build = 0.1)

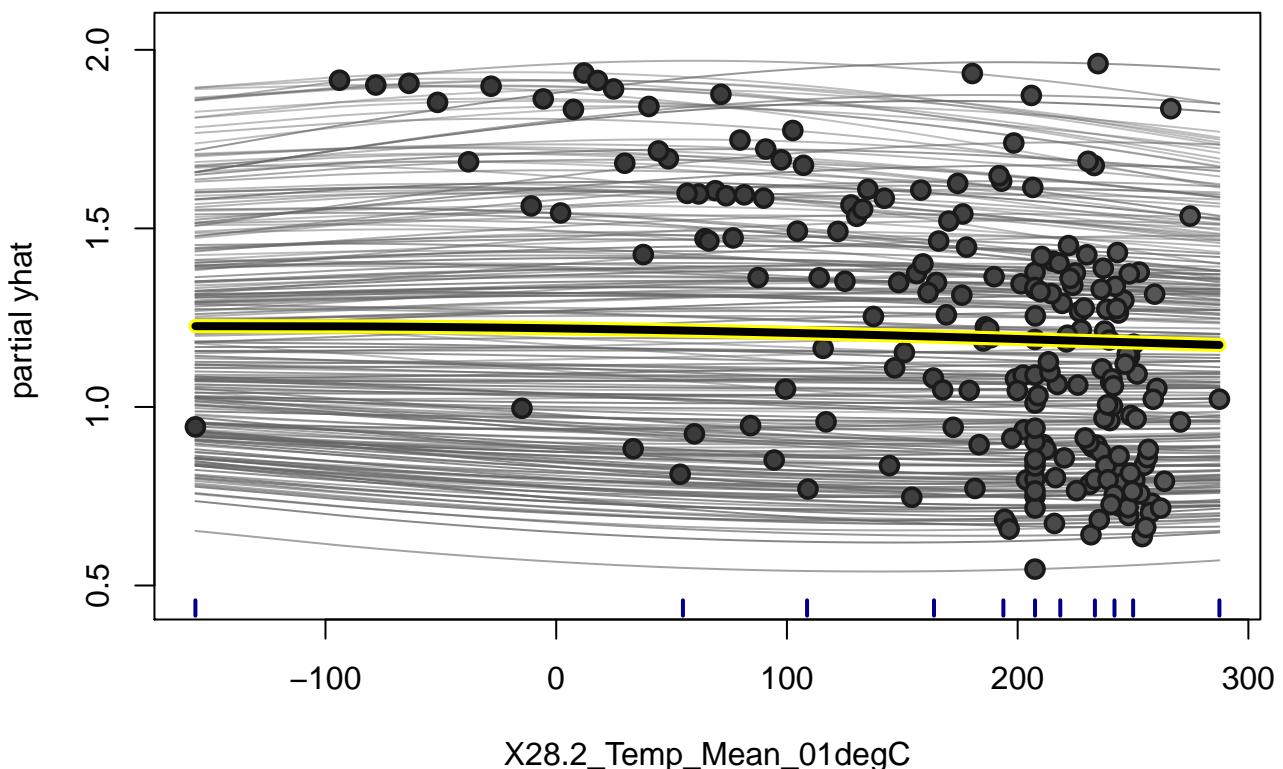
## .....
plot(m3_rf_ice_lat)
```



X26.4_GR_MidRangeLat_dd

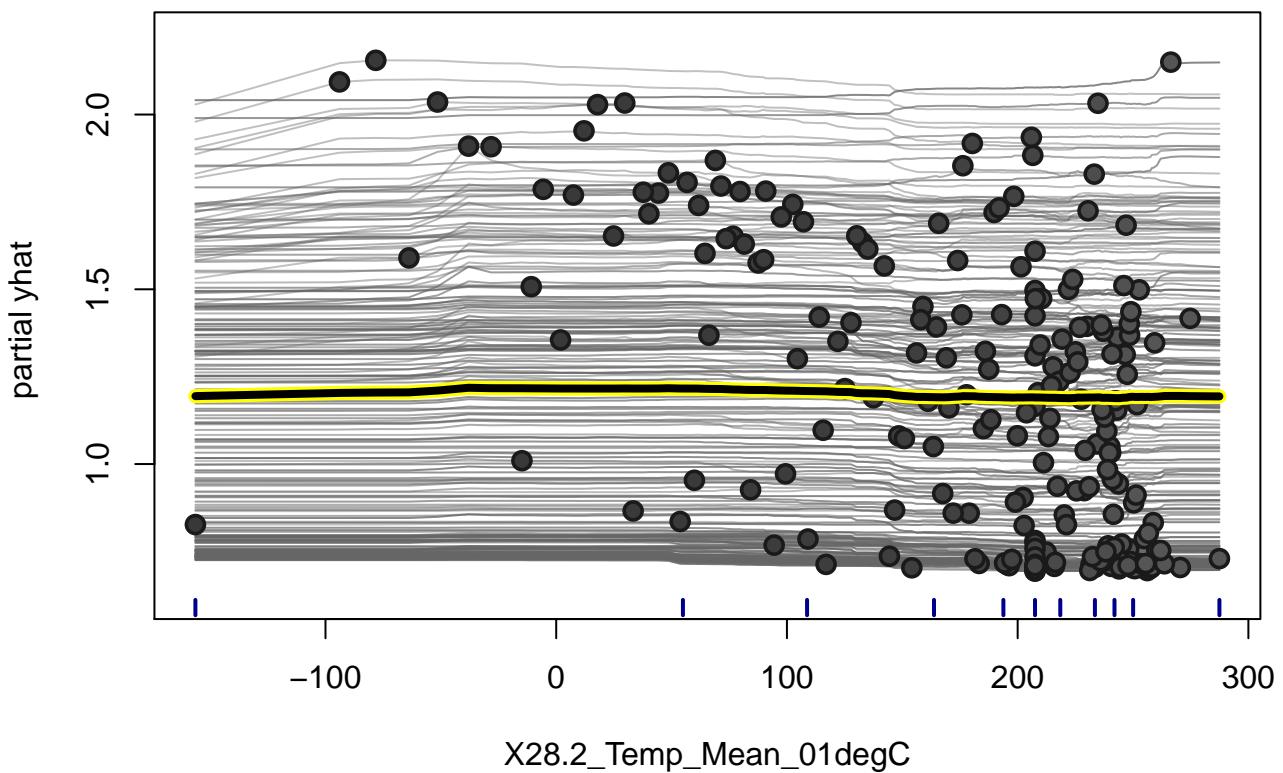
```
m2_gp_ice_temp <- ice(m2_gp, p_impute, p_impute$y, 'X28.2_Temp_Mean_01degC', frac_to_build = 0.1)

## .....
plot(m2_gp_ice_temp)
```



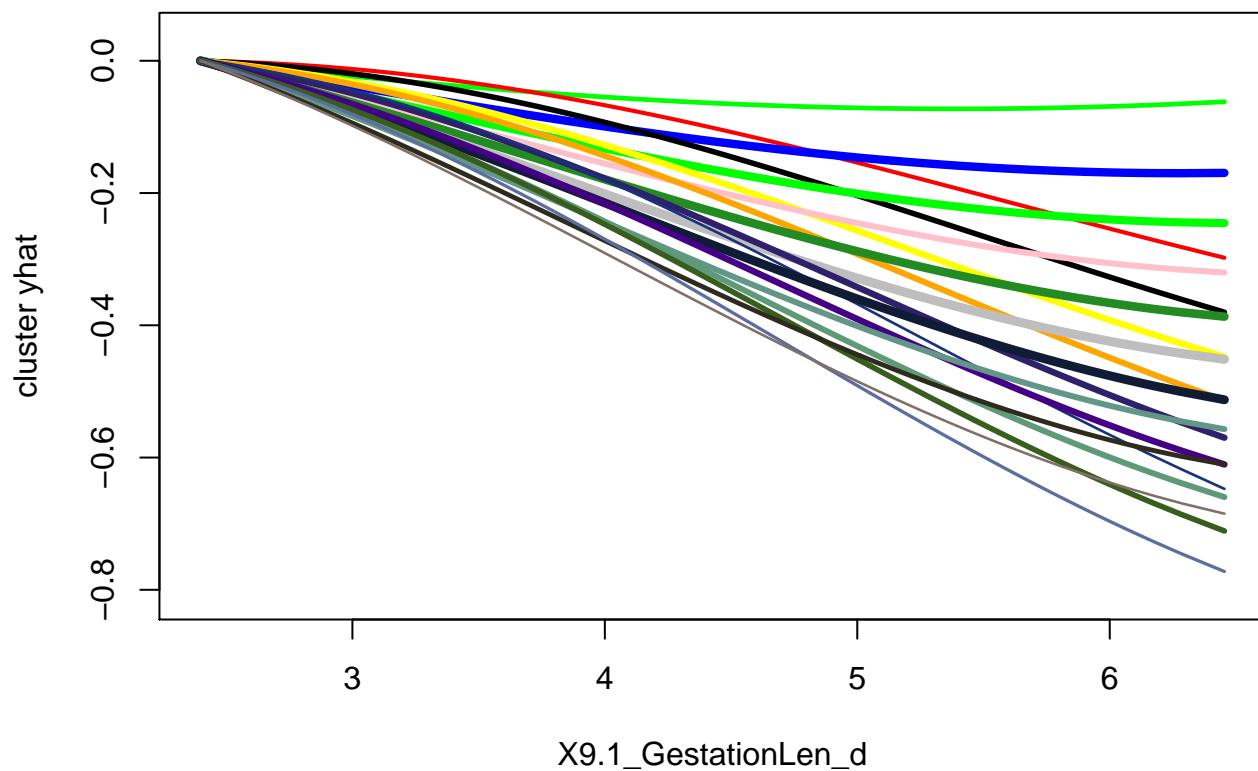
```
m3_rf_ice_temp <- ice(m3_rf, p_impute, p_impute$y, 'X28.2_Temp_Mean_01degC', frac_to_build = 0.1)

## .....
plot(m3_rf_ice_temp)
```

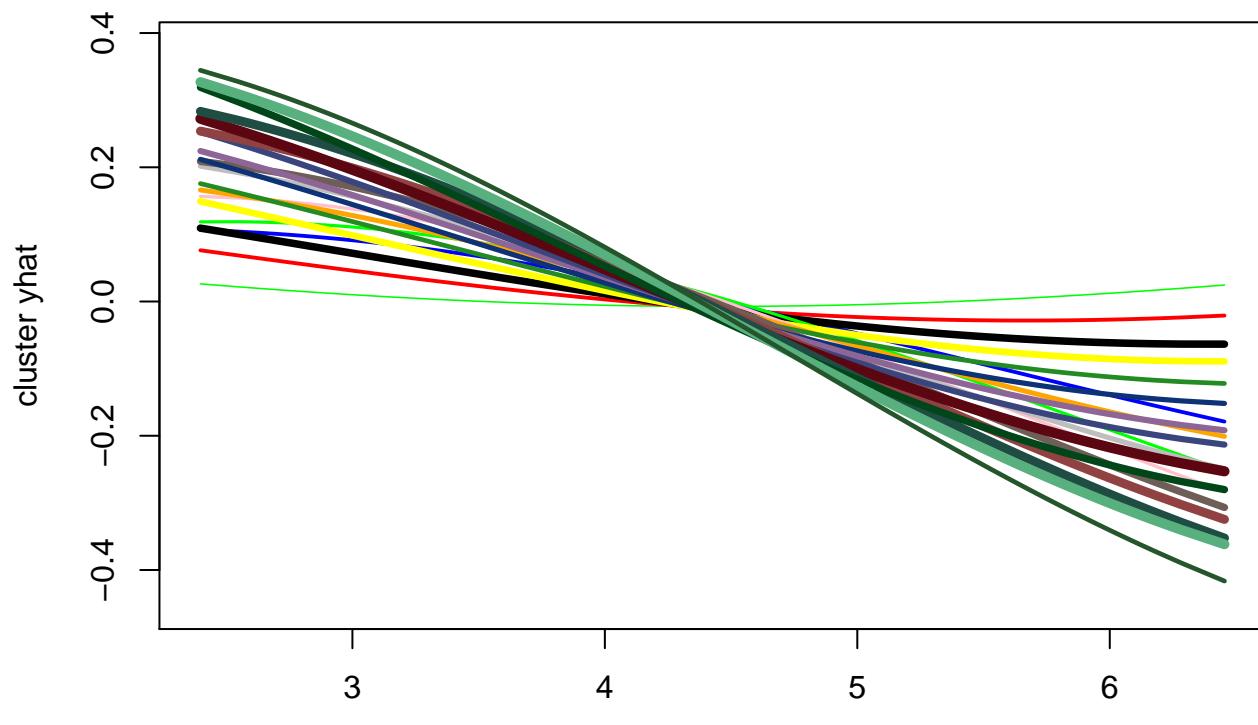


```
m2_gp_ice_c <- ice(m2_gp, p_impute, p_impute$y, 'X9.1_GestationLen_d')

## .....
clusterICE(m2_gp_ice_c, nClusters = 20, centered = TRUE)
```

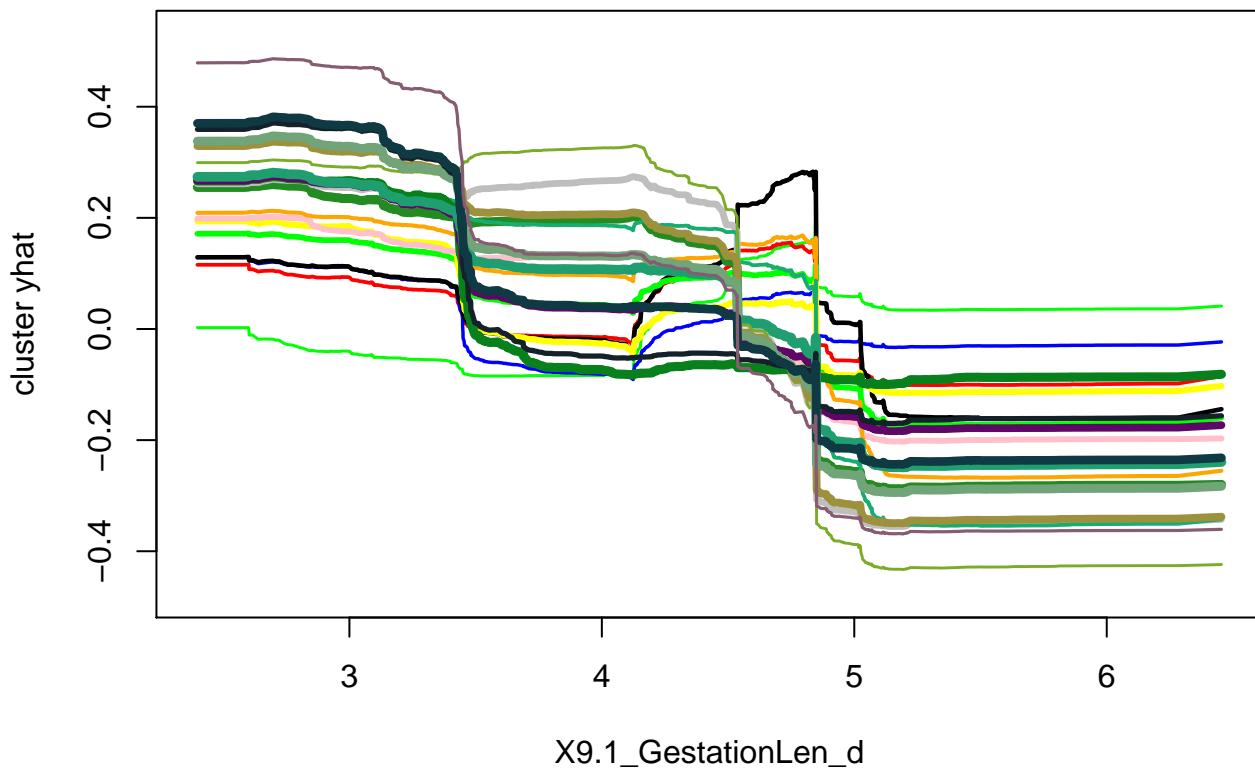


```
clusterICE(m2_gp_ice_c, nClusters = 20, centered = FALSE)
```



```
m3_rf_ice_c <- ice(m3_rf, p_impute, p_impute$y, 'X9.1_GestationLen_d')
```

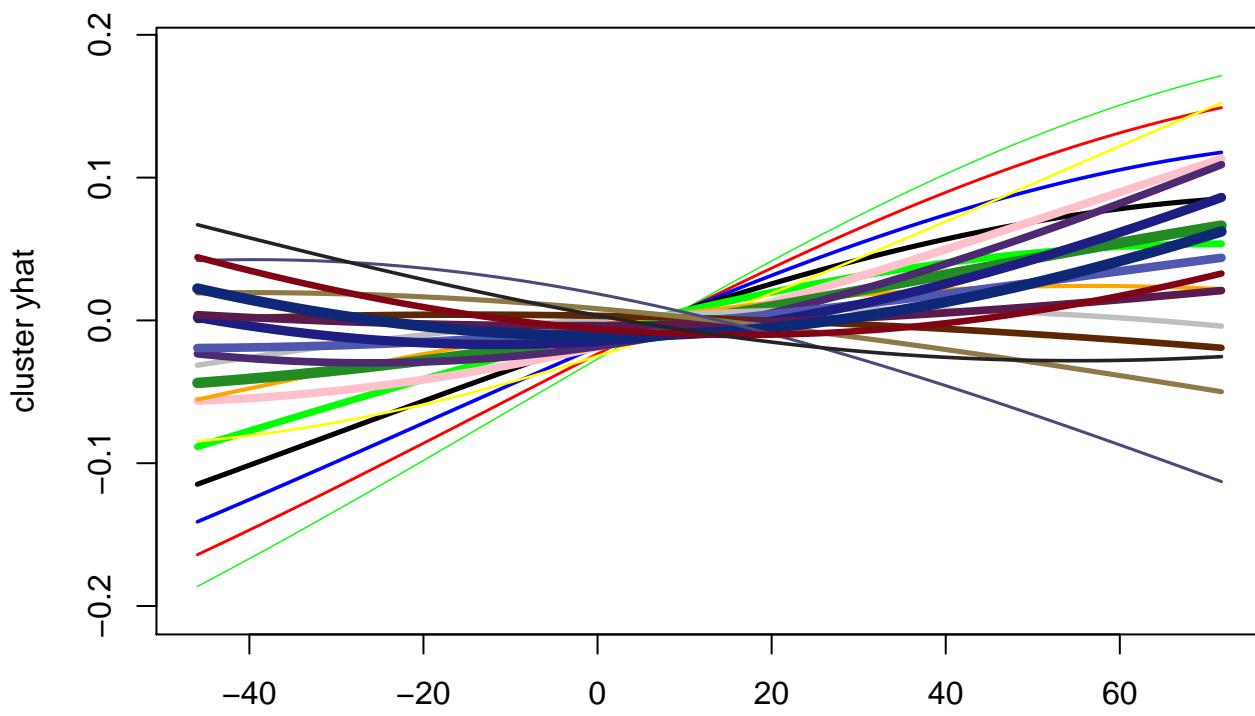
```
## .....  
clusterICE(m3_rf_ice_c, nClusters = 20, centered = FALSE)
```



X9.1_GestationLen_d

```
m2_gp_ice_lat_c <- ice(m2_gp, p_impute, p_impute$y, 'X26.4_GR_MidRangeLat_dd')

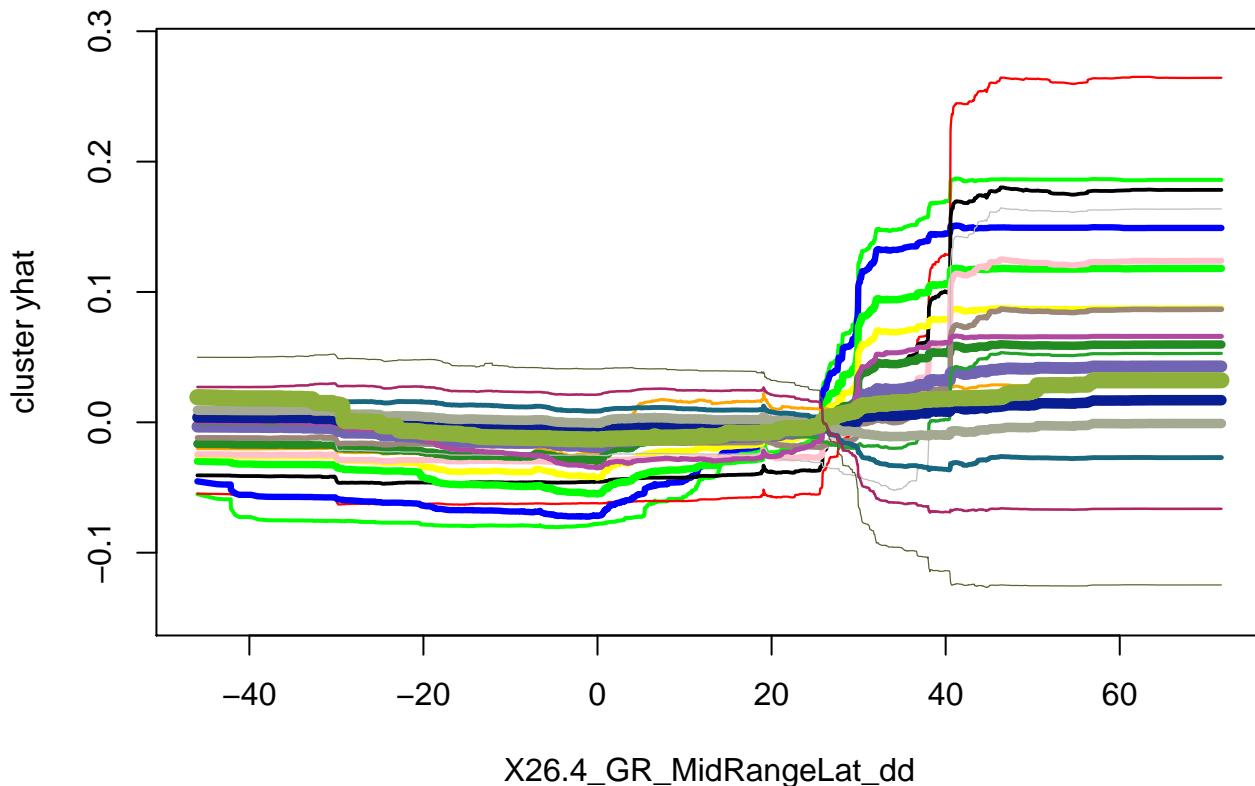
## .....
clusterICE(m2_gp_ice_lat_c, nClusters = 20, centered = FALSE)
```



X26.4_GR_MidRangeLat_dd

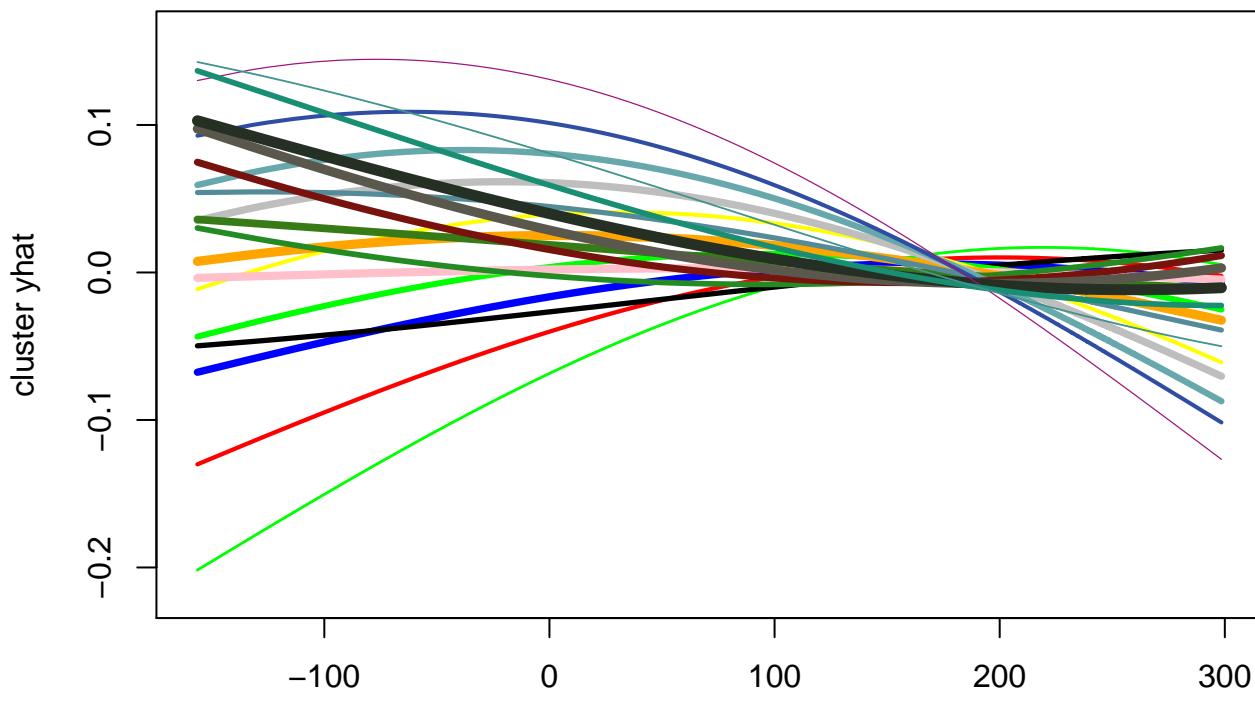
```
m3_rf_ice_lat_c <- ice(m3_rf, p_impute, p_impute$y, 'X26.4_GR_MidRangeLat_dd')

## .....
clusterICE(m3_rf_ice_lat_c, nClusters = 20, centered = FALSE)
```



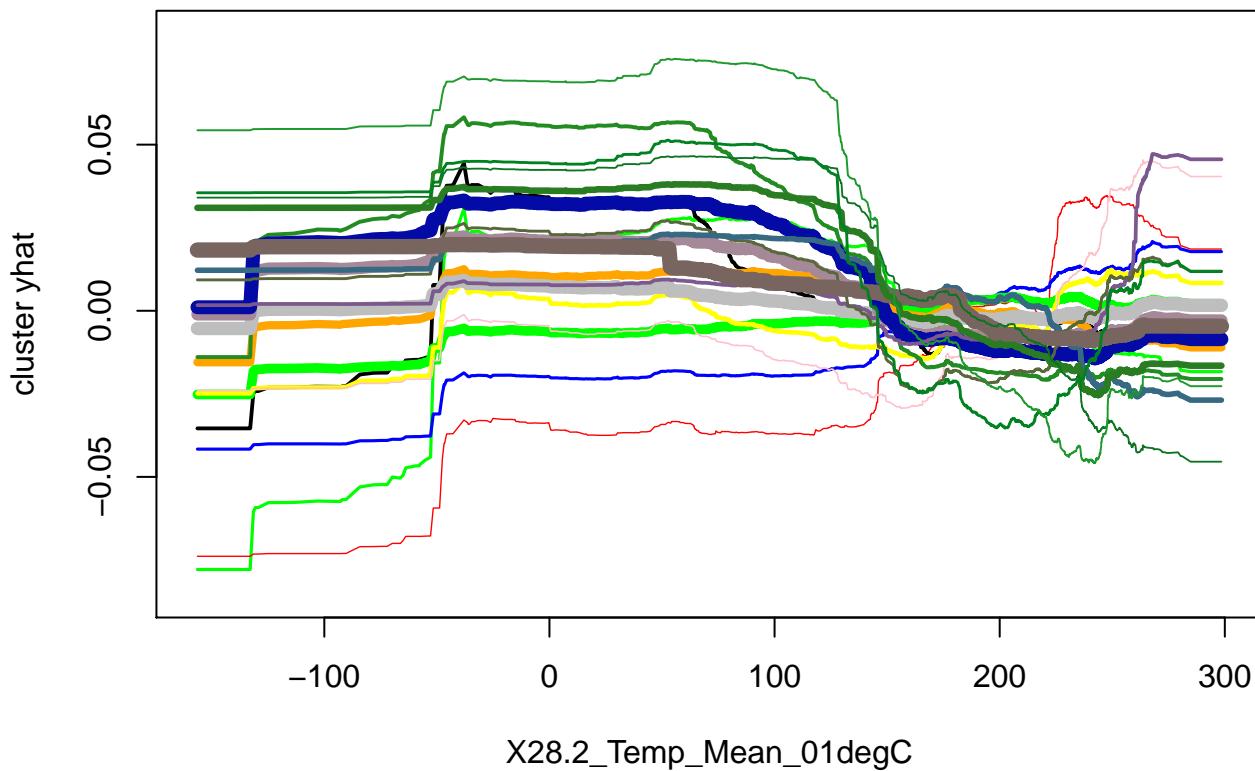
```
m2_gp_ice_temp_c <- ice(m2_gp, p_impute, p_impute$y, 'X28.2_Temp_Mean_01degC')

## .....
clusterICE(m2_gp_ice_temp_c, nClusters = 20, centered = FALSE)
```



```
m3_rf_ice_temp_c <- ice(m3_rf, p_impute, p_impute$y, 'X28.2_Temp_Mean_01degC')

## .....
clusterICE(m3_rf_ice_temp_c, nClusters = 20, centered = FALSE)
```



Examine correlation structure

- examine correlation structure (variable level)
 - random effects
 - mean zero
 - unseen values
 - control Vs use in prediction
 - shared power
 - feature engineering (eg t-1 value)
 - stacked generalisation
 - priors
 - random slopes are regularised interactions. dealt with fairly natively by RF.
 - RF on distance to points could be applied to phylogeny. ## Using genus as categorical variable.

```

genus_dummy <- dummyVars(~ MSW05_Genus, data = p)
genus_dummy_data <- data.frame(predict(genus_dummy, newdata = p))

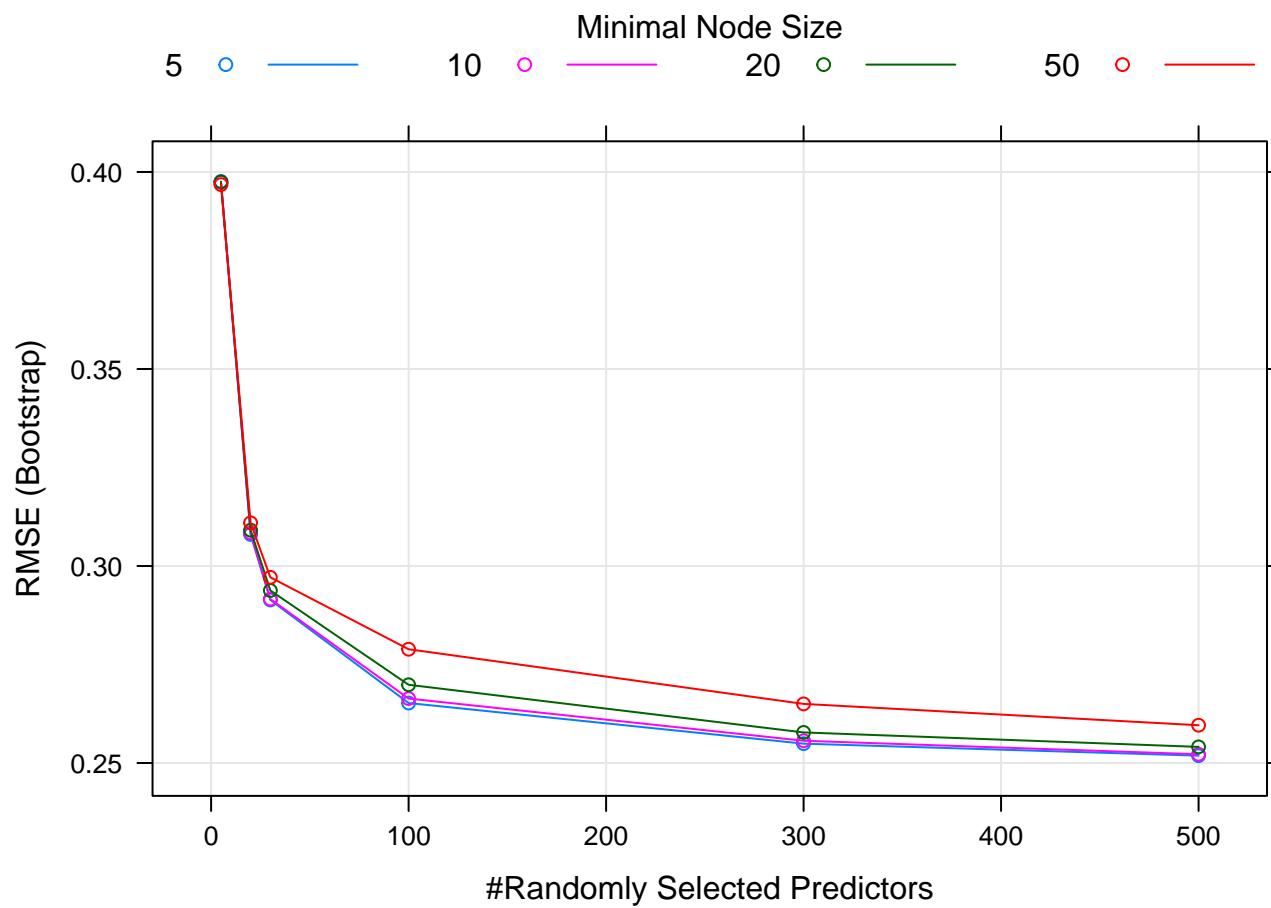
genus_data <- cbind(p_impute, genus_dummy_data)

rf_gr_gen <- expand.grid(mtry = c(5, 20, 30, 100, 300, 500), splitrule = 'variance', min.node.size = c(5, 10, 15))
m3_rf_gen <- train(y ~ ., data = genus_data, method = 'ranger', tuneGrid = rf_gr_gen,
                     trControl = trcntrl, na.action = na.omit, importance = 'impurity',
                     save.memory = TRUE, num.trees = 1000)

## Growing trees.. Progress: 38%. Estimated remaining time: 50 seconds.
## Growing trees.. Progress: 76%. Estimated remaining time: 19 seconds.

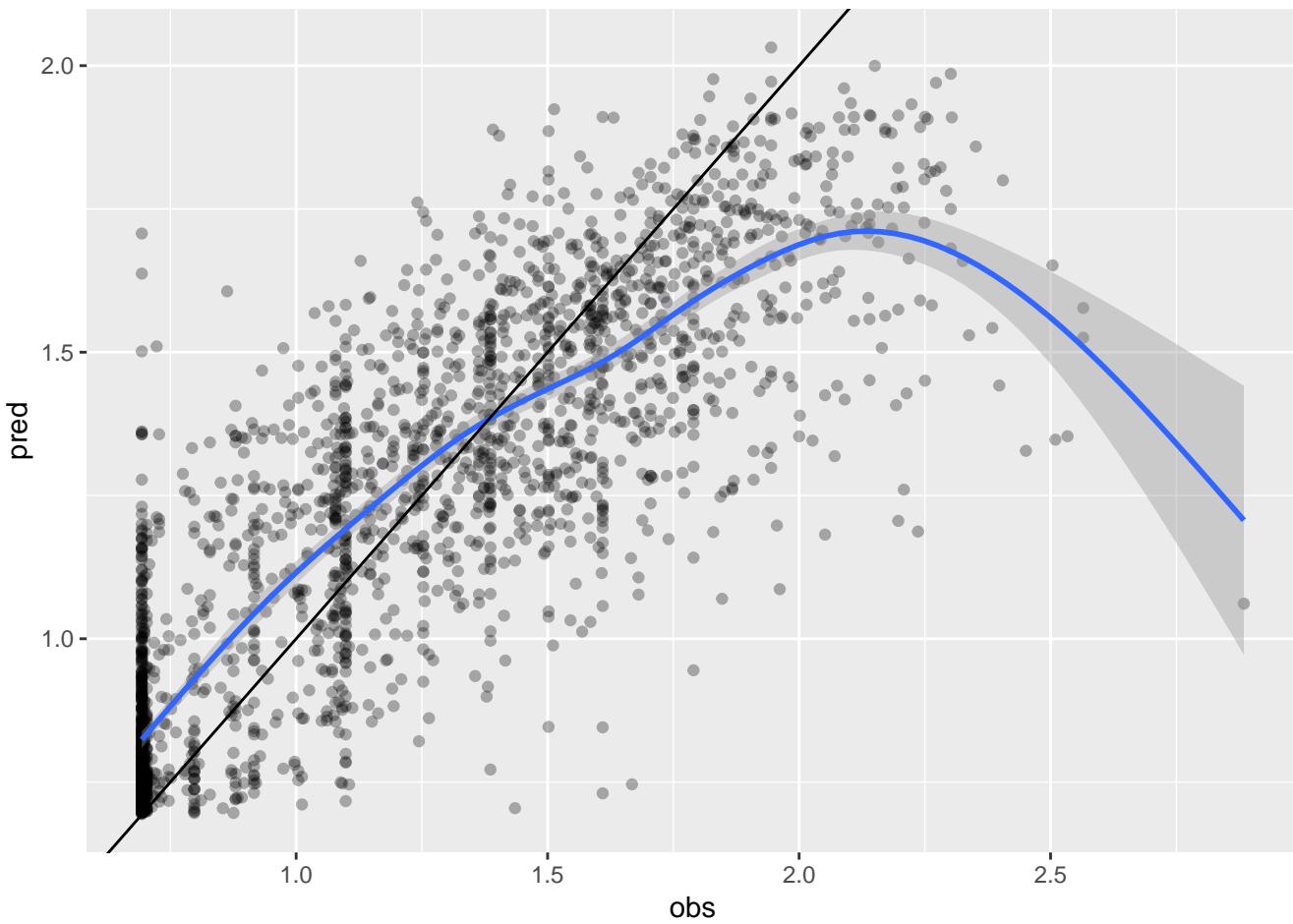
plot(m3_rf_gen)

```



```
plotCV(m3_rf_gen)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



m3_rf_gen

```
## Random Forest
## 
## 2143 samples
## 1277 predictors
## 
## No pre-processing
## Resampling: Bootstrapped (5 reps)
## Summary of sample sizes: 1714, 1715, 1714, 1715, 1714
## Resampling results across tuning parameters:
## 
##   mtry  min.node.size   RMSE      Rsquared     MAE
##   5      5             0.3975884  0.5200278  0.3385923
##   5      10            0.3973934  0.5146602  0.3383181
##   5      20            0.3974637  0.5194522  0.3385134
##   5      50            0.3967874  0.5226459  0.3378351
##   20     5             0.3080060  0.6052501  0.2482522
##   20     10            0.3084312  0.6036814  0.2486763
##   20     20            0.3090979  0.6013703  0.2491830
##   20     50            0.3109547  0.5965261  0.2508497
##   30     5             0.2913776  0.6320828  0.2309635
##   30     10            0.2916158  0.6313681  0.2312531
##   30     20            0.2937845  0.6257520  0.2330478
##   30     50            0.2971199  0.6184921  0.2363593
##   100    5             0.2652209  0.6761404  0.2016485
##   100    10            0.2663579  0.6739501  0.2031777
##   100    20            0.2698517  0.6664558  0.2067950
##   100    50            0.2788782  0.6464537  0.2160755
##   300    5             0.2549070  0.6935480  0.1888755
```

```

##   300    10      0.2556478  0.6922447  0.1898970
##   300    20      0.2577598  0.6876857  0.1925412
##   300    50      0.2650015  0.6716487  0.1999991
##   500    5       0.2518618  0.6982343  0.1843303
##   500    10      0.2522184  0.6975714  0.1847423
##   500    20      0.2540926  0.6934045  0.1870148
##   500    50      0.2595860  0.6814795  0.1931563
##
## Tuning parameter 'splitrule' was held constant at a value of variance
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 500, splitrule =
## variance and min.node.size = 5.
m3_rf_gen$results$Rsquared %>% max
## [1] 0.6982343

```

PGLS phylogenetic regression

Fit a standard phylogenetic regression with caper so we can check the later INLA models are reasonable.

```

comp_data <- comparative.data(tree, cbind(p_impute, MSW05_Binomial = p$MSW05_Binomial)[1:200, ], 'MSW05_Binomial')

# Model will not run without log transforming.
apriori_formula_phylo <- y ~ X5.1_AdultBodyMass_g + X3.1_AgeatFirstBirth_d + X18.1_BasalMetRate_mL02hr +
                           X9.1_GestationLen_d + X16.1_LittersPerYear + X17.1_MaxLongevity_m

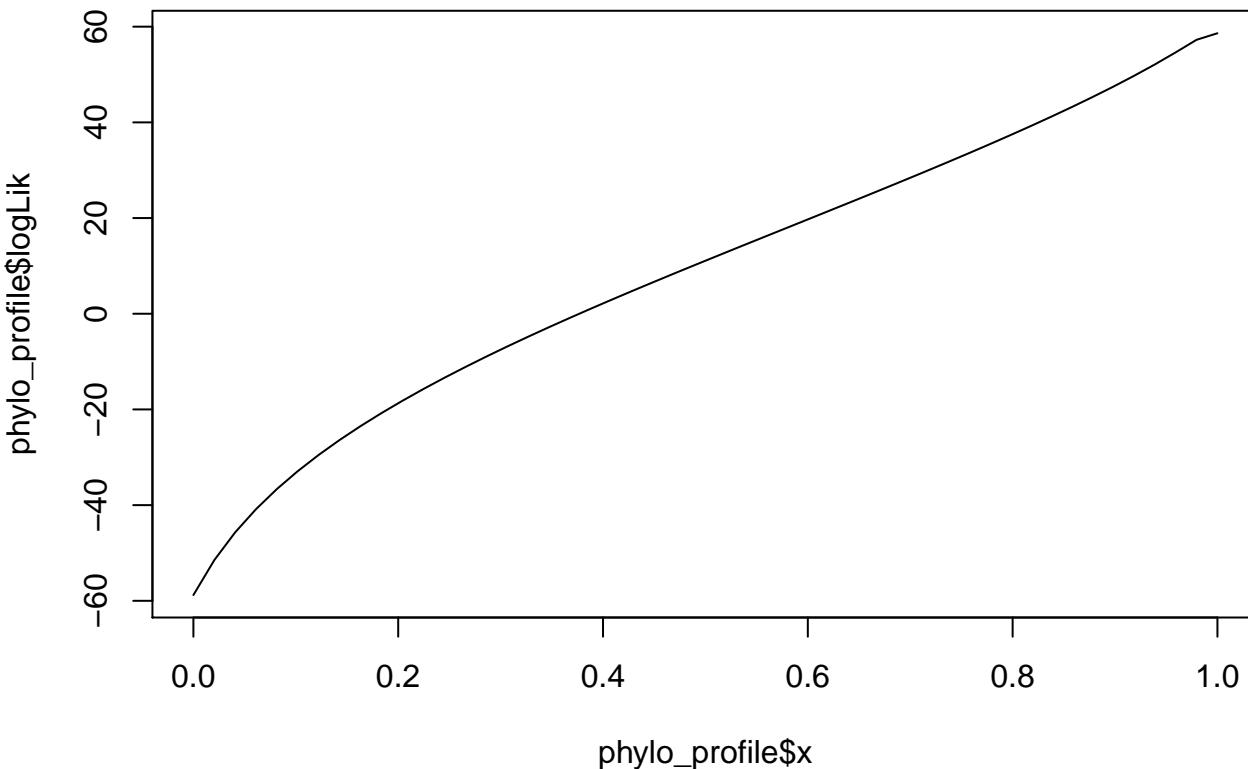
m0_pglm <- pgls(apriori_formula_phylo, data = comp_data)

m0_pglm

## 
## Call:
## pgls(formula = apriori_formula_phylo, data = comp_data)
## 
## Coefficients:
##             (Intercept)      X5.1_AdultBodyMass_g
##             1.176e+00      -1.109e-02
##             X3.1_AgeatFirstBirth_d  X18.1_BasalMetRate_mL02hr
##             -1.193e-05      2.628e-02
##             X9.1_GestationLen_d    X16.1_LittersPerYear
##             -3.959e-02      1.030e-02
##             X17.1_MaxLongevity_m
##             1.123e-02
phylo_profile <- pgls.profile(m0_pglm)

plot(phylo_profile$logLik ~ phylo_profile$x, type = 'l')

```



```

nspp <- length(comp_data$phy$tip.label)
Vphy <- ape::vcv.phylo(comp_data$phy)
Vphy <- solve(Vphy)

order <- match(p$MSW05_Binomial[1:200], colnames(Vphy))
Vphy <- Vphy[order, order] # same order as species levels

# Decide a decent prior.
# Want it to be considerably less than the sd of residuals from fixed effects model.
sd(m0_lm$finalModel$residuals)

## [1] 0.3698553

hyper.prior <- list(prec = list(prior="pc.prec", param = c(0.01, 0.00001)))

apriori_form_inla <- y ~ X5.1_AdultBodyMass_g + X3.1_AgeatFirstBirth_d + X18.1_BasalMetRate_mL02hr +
  X9.1_GestationLen_d + X16.1_LittersPerYear + X17.1_MaxLongevity_m +
  f(phylo, model = 'generic0', Cmatrix = Vphy, hyper = hyper.prior)

m0_inla_comp <- inla(apriori_form_inla, data = cbind(p_impute[1:200, ], phylo = 1:200), control.predictor = li

# autoplot(m0_inla_comp)

m0_inla_comp$summary.fixed

##                                     mean          sd    0.025quant
## (Intercept)      1.325296e+00 2.623426e-01 8.063124e-01
## X5.1_AdultBodyMass_g -1.335263e-02 1.058011e-02 -3.413838e-02
## X3.1_AgeatFirstBirth_d -1.540641e-05 2.714377e-05 -6.924512e-05
## X18.1_BasalMetRate_mL02hr 1.265097e-02 1.455241e-02 -1.709562e-02
## X9.1_GestationLen_d -6.843786e-02 2.940687e-02 -1.276682e-01
## X16.1_LittersPerYear 7.859127e-03 1.611157e-02 -2.396073e-02
## X17.1_MaxLongevity_m 2.424826e-02 2.965359e-02 -3.366914e-02
##                                     0.5quant      0.975quant        mode

```

```

## (Intercept)          1.326539e+00  1.837003e+00  1.329127e+00
## X5.1_AdultBodyMass_g -1.335375e-02  7.418337e-03 -1.335526e-02
## X3.1_AgeatFirstBirth_d -1.527497e-05  3.763627e-05 -1.502183e-05
## X18.1_BasalMetRate_mL02hr  1.305871e-02  4.016189e-02  1.389145e-02
## X9.1_GestationLen_d      -6.792392e-02 -1.205832e-02 -6.687515e-02
## X16.1_LittersPerYear     7.908972e-03  3.936747e-02  8.006255e-03
## X17.1_MaxLongevity_m     2.411722e-02  8.283189e-02  2.386250e-02
##                                     kld
## (Intercept)          1.976098e-13
## X5.1_AdultBodyMass_g  3.858528e-13
## X3.1_AgeatFirstBirth_d 1.341333e-12
## X18.1_BasalMetRate_mL02hr 1.201881e-11
## X9.1_GestationLen_d    6.401673e-13
## X16.1_LittersPerYear   1.027705e-12
## X17.1_MaxLongevity_m   1.213065e-12
m0_inla_comp$summary.hyperpar

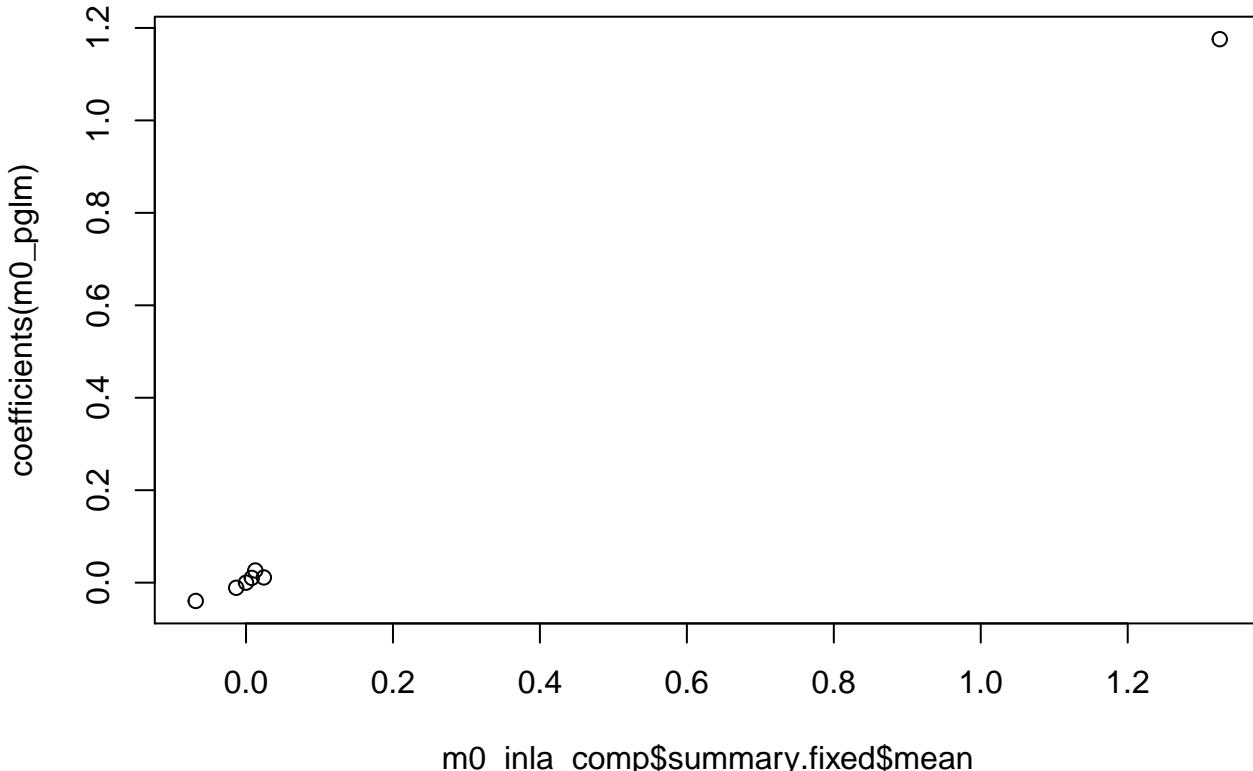
```

```

##                                         mean        sd 0.025quant
## Precision for the Gaussian observations 290.8399 187.4009  99.06641
## Precision for phylo                      906.8918 155.4347 616.07146
##                                         0.5quant 0.975quant mode
## Precision for the Gaussian observations 238.3625 787.0792 170.8831
## Precision for phylo                      904.3730 1222.1420 909.4108

```

```
plot(m0_inla_comp$summary.fixed$mean, coefficients(m0_pglm))
```



```

comp_data_full <- comparative.data(tree, cbind(p_impute, MSW05_Binomial = p$MSW05_Binomial), 'MSW05_Binomial')

# Code broadly copied from https://github.com/daijiang/phyr/blob/master/R/pglmm-utils.R#L54
nspp <- length(comp_data_full$phy$tip.label)
Vphy <- ape::vcv(comp_data_full$phy)
Vphy <- solve(Vphy)

order <- match(p$MSW05_Binomial, colnames(Vphy))
Vphy <- Vphy[order, order] # same order as species levels

```

```

# fit full model

m0_inla_comp_full <- inla(apriori_form_inla, data = cbind(p_impute, phylo = 1:nrow(p_impute)),
                           control.predictor = list(compute = TRUE),
                           control.inla= list(strategy = "gaussian", int.strategy = "eb"))

m0_inla_comp_full$summary.fixed

##                                     mean          sd 0.025quant
## (Intercept)           1.490076e+00 2.407773e-01 1.017349e+00
## X5.1_AdultBodyMass_g -1.599514e-02 5.103820e-03 -2.601566e-02
## X3.1_AgeatFirstBirth_d -5.429982e-07 1.553478e-05 -3.104301e-05
## X18.1_BasalMetRate_mL02hr 9.608507e-03 5.766556e-03 -1.713187e-03
## X9.1_GestationLen_d -5.141640e-02 1.207399e-02 -7.512172e-02
## X16.1_LittersPerYear 1.432133e-02 5.780201e-03 2.972849e-03
## X17.1_MaxLongevity_m -2.616186e-02 9.996875e-03 -4.578910e-02
##                                     0.5quant    0.975quant      mode
## (Intercept)           1.490070e+00 1.962409e+00 1.490076e+00
## X5.1_AdultBodyMass_g -1.599529e-02 -5.982988e-03 -1.599514e-02
## X3.1_AgeatFirstBirth_d -5.434357e-07 2.993156e-05 -5.429982e-07
## X18.1_BasalMetRate_mL02hr 9.608345e-03 2.092075e-02 9.608507e-03
## X9.1_GestationLen_d -5.141674e-02 -2.773087e-02 -5.141640e-02
## X16.1_LittersPerYear 1.432117e-02 2.566034e-02 1.432133e-02
## X17.1_MaxLongevity_m -2.616214e-02 -6.551004e-03 -2.616186e-02
##                                     kld
## (Intercept)           2.267975e-30
## X5.1_AdultBodyMass_g 0.000000e+00
## X3.1_AgeatFirstBirth_d 0.000000e+00
## X18.1_BasalMetRate_mL02hr 1.972152e-31
## X9.1_GestationLen_d 5.423419e-31
## X16.1_LittersPerYear 0.000000e+00
## X17.1_MaxLongevity_m 3.944305e-31

m0_inla_comp_full$summary.hyperpar

##                                     mean          sd 0.025quant
## Precision for the Gaussian observations 51.07758 3.845308 43.98966
## Precision for phylo                   698.17100 48.367376 607.33180
##                                     0.5quant    0.975quant      mode
## Precision for the Gaussian observations 50.9168 59.07455 50.58312
## Precision for phylo                   696.7450 797.44724 694.14836

# cross validation
m0_inla_comp2 <- list()

for(f in 1:5){

  cv_data <- cbind(p_impute, phylo = 1:nrow(p_impute))

  # remove hold out data
  cv_data$y[folds[[f]]] <- NA

  m0_inla_comp2[[f]] <- inla(apriori_form_inla, data = cv_data,
                             control.predictor = list(compute = TRUE),
                             control.inla= list(strategy = "gaussian", int.strategy = "eb"))
}

```

```

}

# calc rmse and scatter plot

inla_apriori_cvdat <- data.frame(y = p_impute$y, predy = NA)

for(f in 1:5){
  inla_apriori_cvdat$predy[folds[[f]]] <- m0_inla_comp2[[f]]$summary.fitted.values$mean[folds[[f]]]
}

sqrt(mean((inla_apriori_cvdat$predy - inla_apriori_cvdat$y) ^ 2))

## [1] 0.242759

cor(inla_apriori_cvdat$predy, inla_apriori_cvdat$y)

## [1] 0.8467217

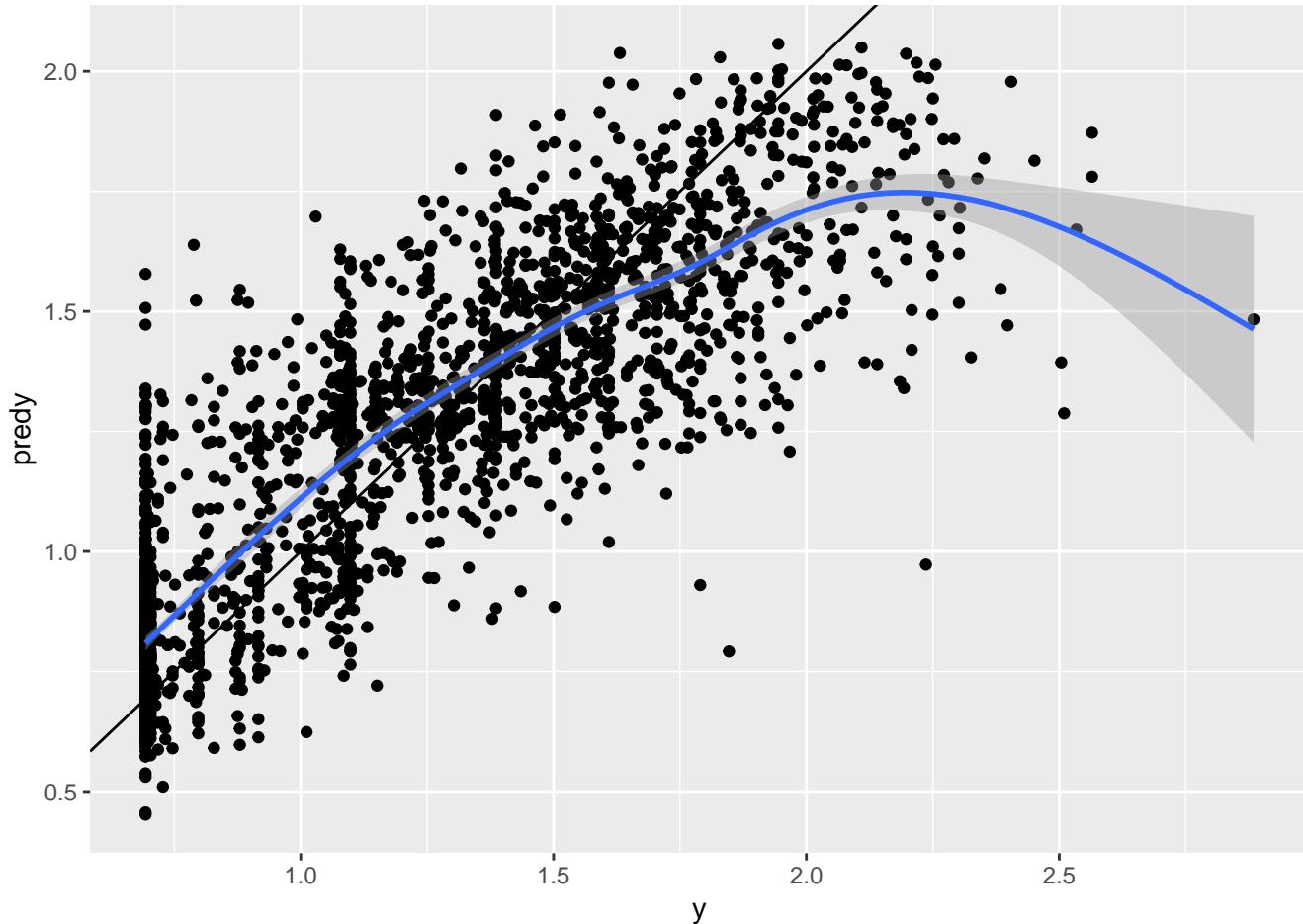
lm(inla_apriori_cvdat$predy ~ inla_apriori_cvdat$y) %>% summary %>% `\$`('r.squared')

## [1] 0.7169377

ggplot(inla_apriori_cvdat, aes(y, predy)) +
  geom_point() +
  #scale_y_continuous(limits = c(NA, 3)) +
  geom_abline(slope = 1, intercept = 0) +
  geom_smooth()

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

```



Regularised linear model

```
# fit full model

enet_form_inla <- paste0('y ~ ', paste(names(p_impute %>% dplyr::select(- y)), collapse = ' + '),
                           ' + f(phylo, model = \'generic0\', Cmatrix = Vphy, hyper = hyper.prior)') %>% formula()

# Select sd of fixed effect prior of 0.001. Gives us precision of 1e6.
m1_enet_inla_full <- inla(enet_form_inla, data = cbind(p_impute, phylo = 1:nrow(p_impute)),
                            control.predictor = list(compute = TRUE),
                            control.fixed = list(prec = 1e6),
                            control.inla= list(strategy = "gaussian", int.strategy = "eb"))

m1_enet_inla_full$summary.fixed
m1_enet_inla_full$summary.hyperpar

# cross validation
m1_enet_inla <- list()

for(f in 1:5){

  cv_data <- cbind(p_impute, phylo = 1:nrow(p_impute))

  # remove hold out data
  cv_data$y[folds[[f]]] <- NA

  m1_enet_inla[[f]] <- inla(enet_form_inla, data = cv_data,
                             control.predictor = list(compute = TRUE),
                             control.fixed = list(prec = 1e6),
                             control.inla= list(strategy = "gaussian", int.strategy = "eb"))

}

# calc rmse and scatter plot

inla_enet_cvdat <- data.frame(y = p_impute$y, predy = NA)

for(f in 1:5){
  inla_enet_cvdat$predy[folds[[f]]] <- m1_enet_inla[[f]]$summary.fitted.values$mean[folds[[f]]]
}

sqrt(mean((inla_enet_cvdat$predy - inla_enet_cvdat$y) ^ 2))

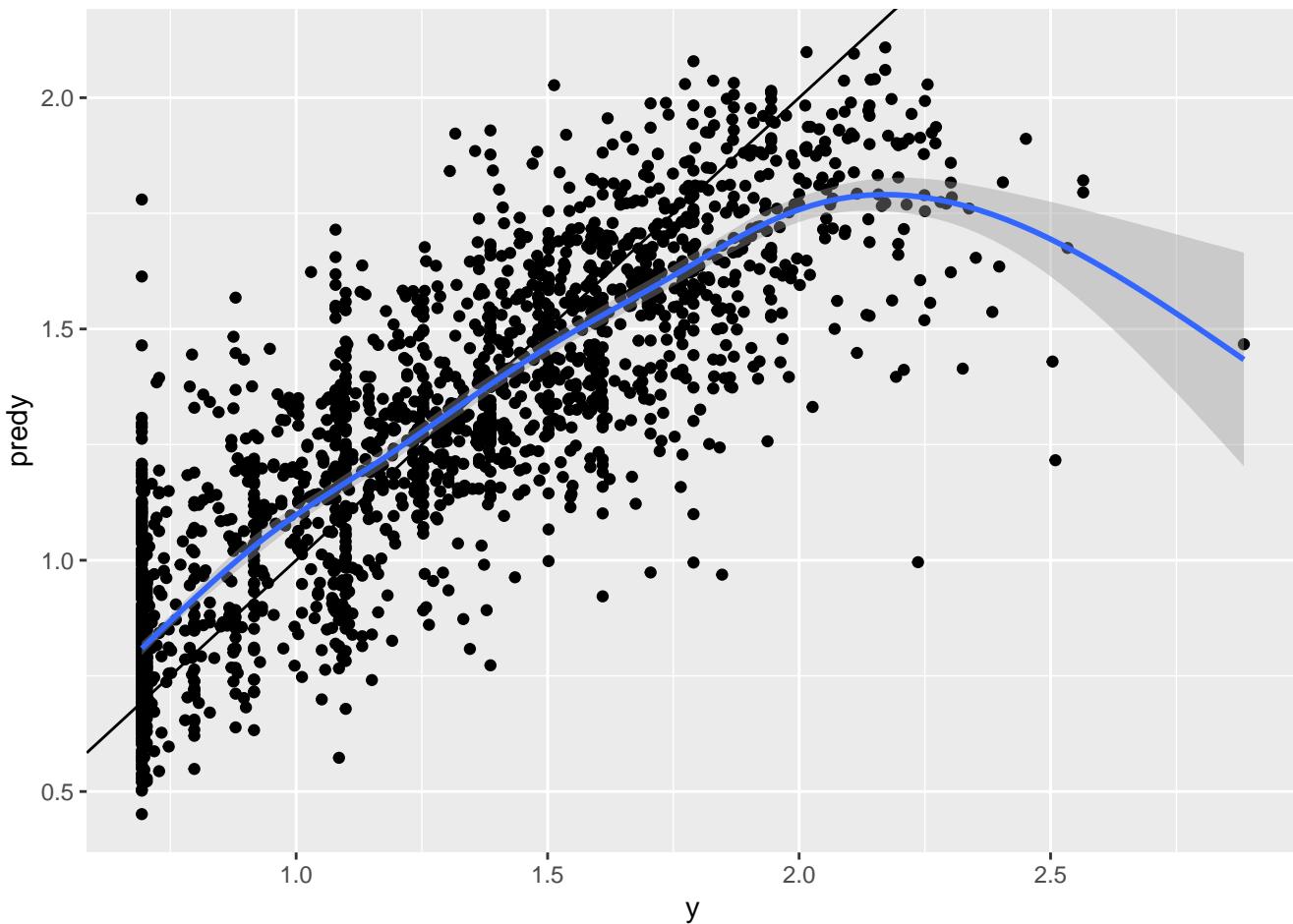
## [1] 0.2325696
cor(inla_enet_cvdat$predy, inla_enet_cvdat$y)

## [1] 0.8603091
lm(inla_enet_cvdat$predy ~ inla_enet_cvdat$y) %>% summary %>% `\$`('r.squared')

## [1] 0.7401317

ggplot(inla_enet_cvdat, aes(y, predy)) +
  geom_point() +
  #scale_y_continuous(limits = c(NA, 3)) +
  geom_abline(slope = 1, intercept = 0) +
  geom_smooth()

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```

mlist <- list(m1_enet, m2_gp, m3_rf)

cv_preds_raw <- lapply(mlist, best_tune_preds)
cv_preds <- lapply(seq_along(cv_preds_raw), function (x) cv_preds_raw[[x]]$pred)
cv_preds <- do.call(cbind, cv_preds) %>% data.frame

# Put into correct order
cv_preds <- cv_preds[order(cv_preds_raw[[1]]$rowIndex),]

names(cv_preds) <- c('enet', 'gp', 'rf')

cv_preds <- mutate(cv_preds, y = p_impute$y) # check order
cv_preds$phylo <- seq_len(nrow(cv_preds))

stacked_form <- y ~ 0 +
  f(enet, model='clinear', range=c(0, Inf), initial=0) +
  f(gp, model='clinear', range=c(0, Inf), initial=0) +
  f(rf, model='clinear', range=c(0, Inf), initial=0) +
  f(phylo, model = 'generic0', Cmatrix = Vphy, hyper = hyper.prior)

stacked_full <- inla(stacked_form, data = cv_preds,
                      control.predictor = list(compute = TRUE),
                      control.inla= list(strategy = "gaussian", int.strategy = "eb"))

stacked_full$summary.fixed

## data frame with 0 columns and 0 rows

```

```

stacked_full$summary.hyperpar

##                                     mean          sd
## Precision for the Gaussian observations 44.0704027 2.96503674
## Beta for enet                         0.1605476 0.02050353
## Beta for gp                           0.1835898 0.02345746
## Beta for rf                           0.3514981 0.03073841
## Precision for phylo                  1060.0785512 93.23983894
##                                         0.025quant   0.5quant
## Precision for the Gaussian observations 38.4920278 43.9863867
## Beta for enet                         0.1237593 0.1593646
## Beta for gp                           0.1417057 0.1821645
## Beta for rf                           0.2953634 0.3500297
## Precision for phylo                  890.1276376 1055.4593876
##                                         0.975quant   mode
## Precision for the Gaussian observations 50.1467747 43.8344459
## Beta for enet                         0.2042373 0.1571290
## Beta for gp                           0.2337631 0.1794077
## Beta for rf                           0.4159495 0.3470030
## Precision for phylo                  1255.8172631 1045.8258139

# cross validation

stacked_full2 <- list()

for(f in 1:5){

  cv_data <- cv_preds

  # remove hold out data
  cv_data$y[folds[[f]]] <- NA

  stacked_full2[[f]] <- inla(stacked_form, data = cv_data,
                               control.predictor = list(compute = TRUE),
                               control.inla= list(strategy = "gaussian", int.strategy = "eb"))

}

# calc rmse and scatter plot

stacked_cvdat <- data.frame(y = p_impute$y, predy = NA)

for(f in 1:5){
  stacked_cvdat$predy[folds[[f]]] <- stacked_full2[[f]]$summary.fitted.values$mean[folds[[f]]]
}

sqrt(mean((stacked_cvdat$predy - stacked_cvdat$y) ^ 2))

## [1] 0.2421599
cor(stacked_cvdat$predy, stacked_cvdat$y)

## [1] 0.8491472

lm(stacked_cvdat$predy ~ stacked_cvdat$y) %>% summary %>% `\$`('r.squared')

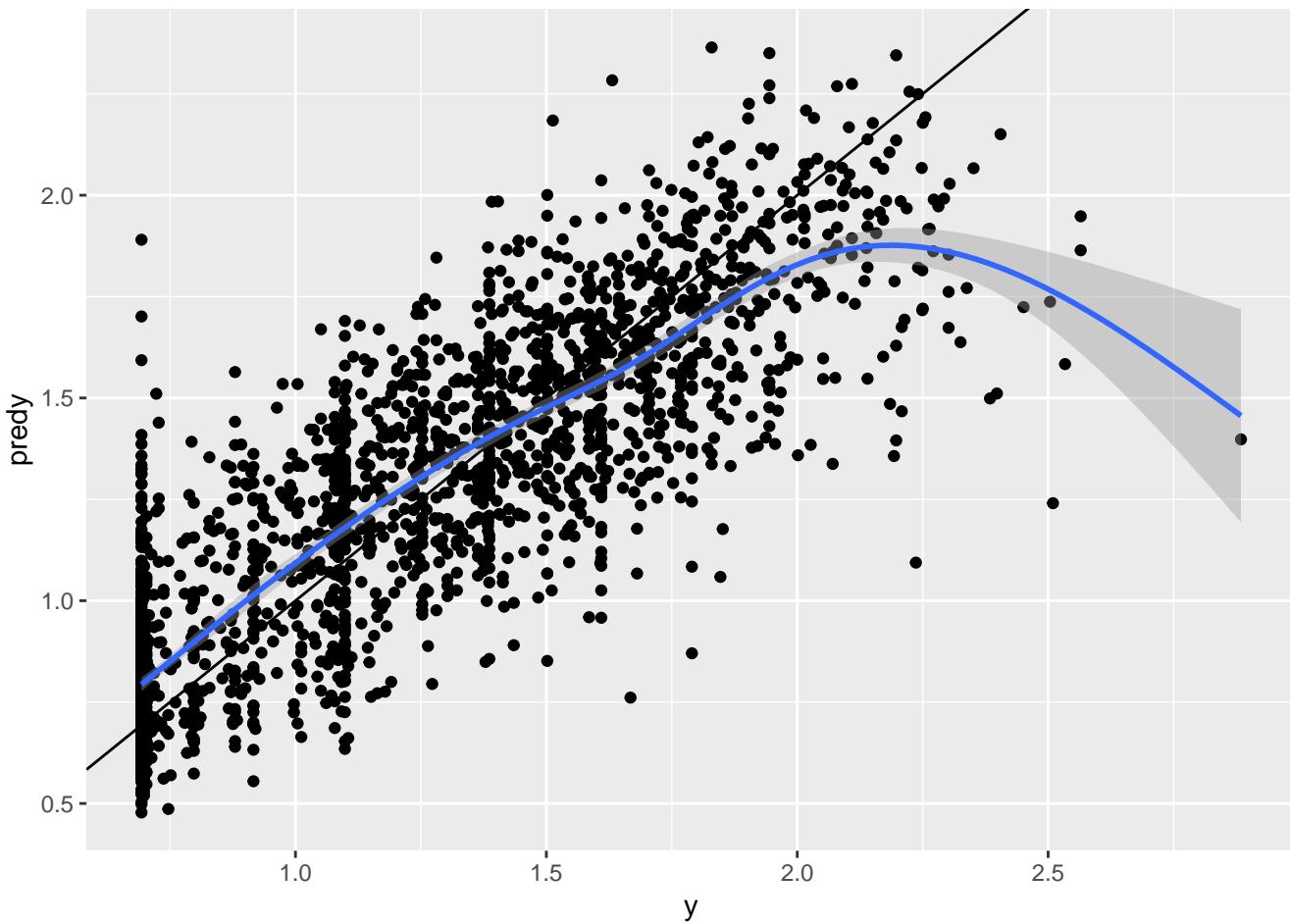
## [1] 0.721051

ggplot(stacked_cvdat, aes(y, predy)) +
  geom_point() +
  scale_y_continuous(limits = c(NA, 3)) +
  geom_abline(slope = 1, intercept = 0) +

```

```
geom_smooth()

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Using distance to each other as covariates

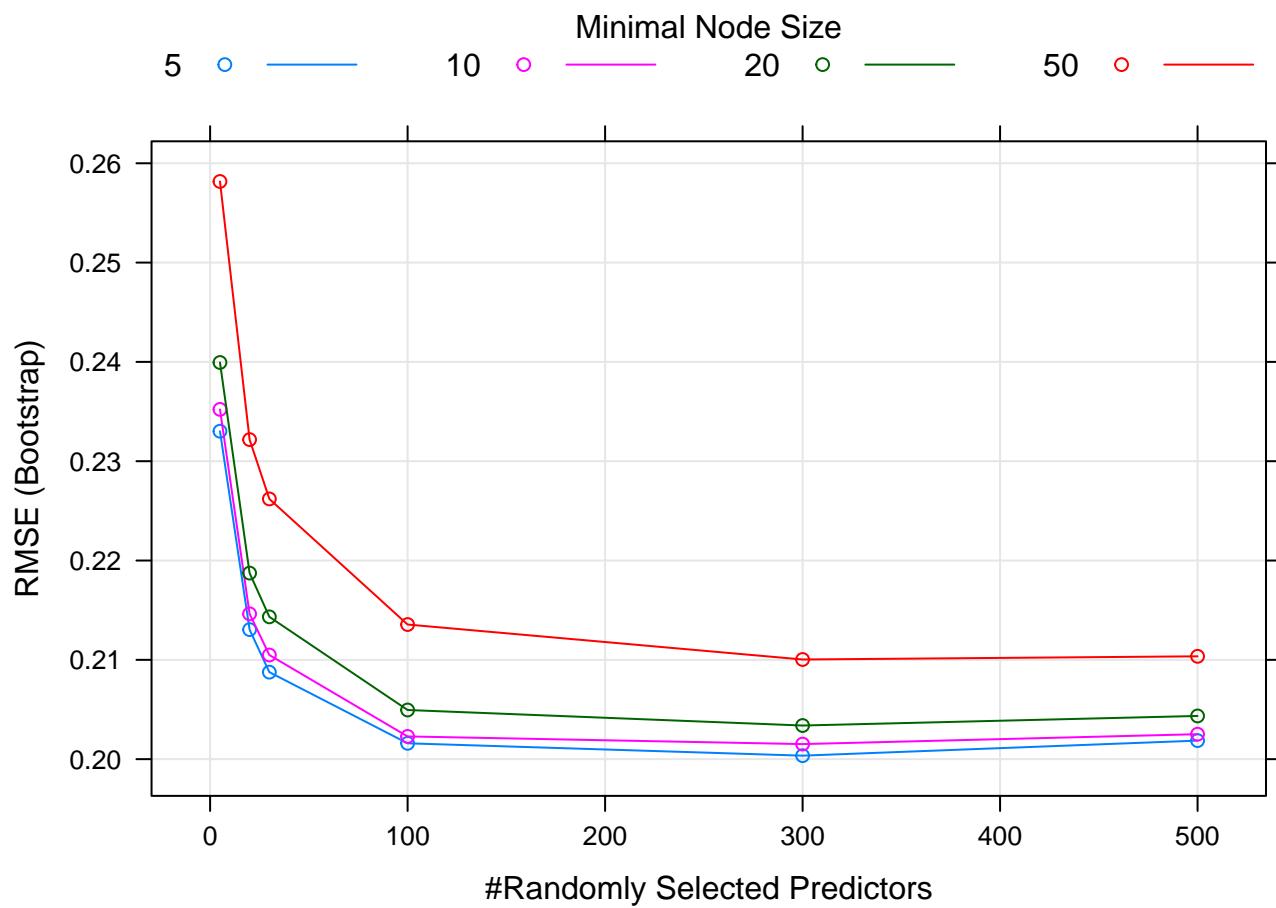
```
dist <- ape::cophenetic.phylo(comp_data_full$phy)
dist <- dist[order, order] # same order as species levels

dist_data <- cbind(p_impute, dist)

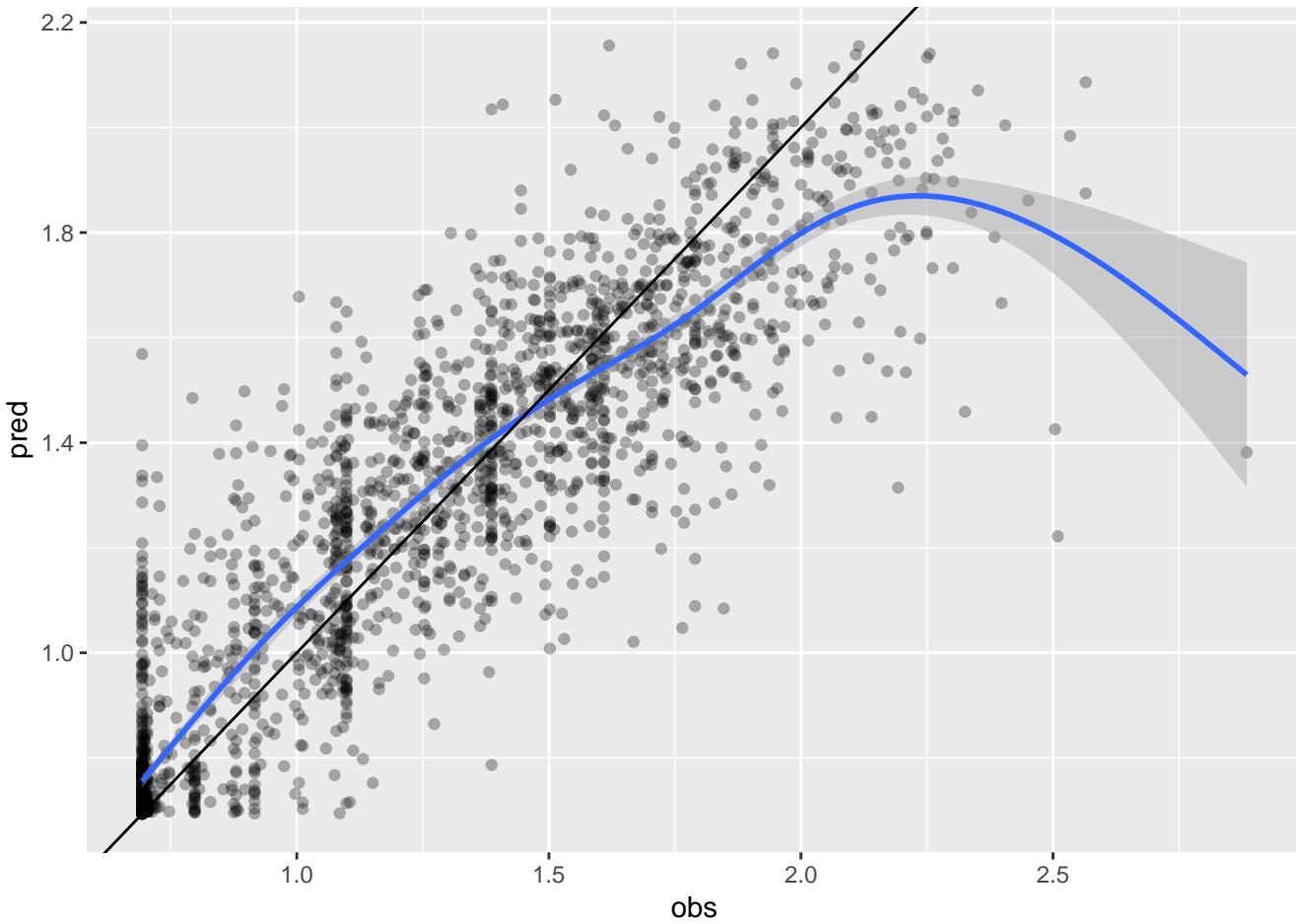
m3_rf_dist <- train(y ~ ., data = dist_data, method = 'ranger', tuneGrid = rf_gr_gen,
                      trControl = trcntrl, na.action = na.omit, importance = 'impurity',
                      save.memory = TRUE, num.trees = 1000)

## Growing trees.. Progress: 19%. Estimated remaining time: 2 minutes, 16 seconds.
## Growing trees.. Progress: 39%. Estimated remaining time: 1 minute, 38 seconds.
## Growing trees.. Progress: 58%. Estimated remaining time: 1 minute, 6 seconds.
## Growing trees.. Progress: 78%. Estimated remaining time: 35 seconds.
## Growing trees.. Progress: 98%. Estimated remaining time: 3 seconds.

plot(m3_rf_dist)
```



```
plotCV(m3_rf_dist)
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```
m3_rf_dist
```

```
## Random Forest
## 
## 2143 samples
## 2190 predictors
## 
## No pre-processing
## Resampling: Bootstrapped (5 reps)
## Summary of sample sizes: 1714, 1715, 1714, 1715, 1714
## Resampling results across tuning parameters:
## 
##   mtry  min.node.size   RMSE      Rsquared     MAE
##   5      5              0.2330206  0.7387793  0.1633385
##   5      10             0.2352168  0.7342702  0.1658948
##   5      20             0.2399383  0.7246471  0.1712113
##   5      50             0.2581654  0.6830884  0.1876714
##   20     5              0.2130462  0.7810381  0.1451401
##   20     10             0.2146317  0.7779104  0.1472152
##   20     20             0.2187356  0.7698582  0.1522700
##   20     50             0.2321760  0.7426681  0.1653027
##   30     5              0.2087589  0.7898324  0.1420413
##   30     10             0.2104853  0.7865754  0.1445030
##   30     20             0.2143276  0.7790480  0.1490468
##   30     50             0.2262028  0.7554625  0.1603118
##   100    5              0.2015960  0.8042570  0.1382527
##   100    10             0.2022899  0.8030139  0.1396084
##   100    20             0.2049504  0.7979310  0.1422218
##   100    50             0.2135562  0.7812562  0.1502735
##   300    5              0.2003519  0.8069550  0.1378101
```

```

##   300    10      0.2015170  0.8048209  0.1388489
##   300    20      0.2033859  0.8012281  0.1410010
##   300    50      0.2100360  0.7884138  0.1471126
##   500    5       0.2018675  0.8042896  0.1391457
##   500    10      0.2025135  0.8032176  0.1399057
##   500    20      0.2043507  0.7996878  0.1417907
##   500    50      0.2103543  0.7883724  0.1477301
##
## Tuning parameter 'splitrule' was held constant at a value of variance
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 300, splitrule =
## variance and min.node.size = 5.
m3_rf_dist$results$Rsquared %>% max

## [1] 0.806955

```

Point level understanding

- understand individual points
 - lime

```
# Explain Tenrec ecaudatus. Largest litter size. And other large litters
```

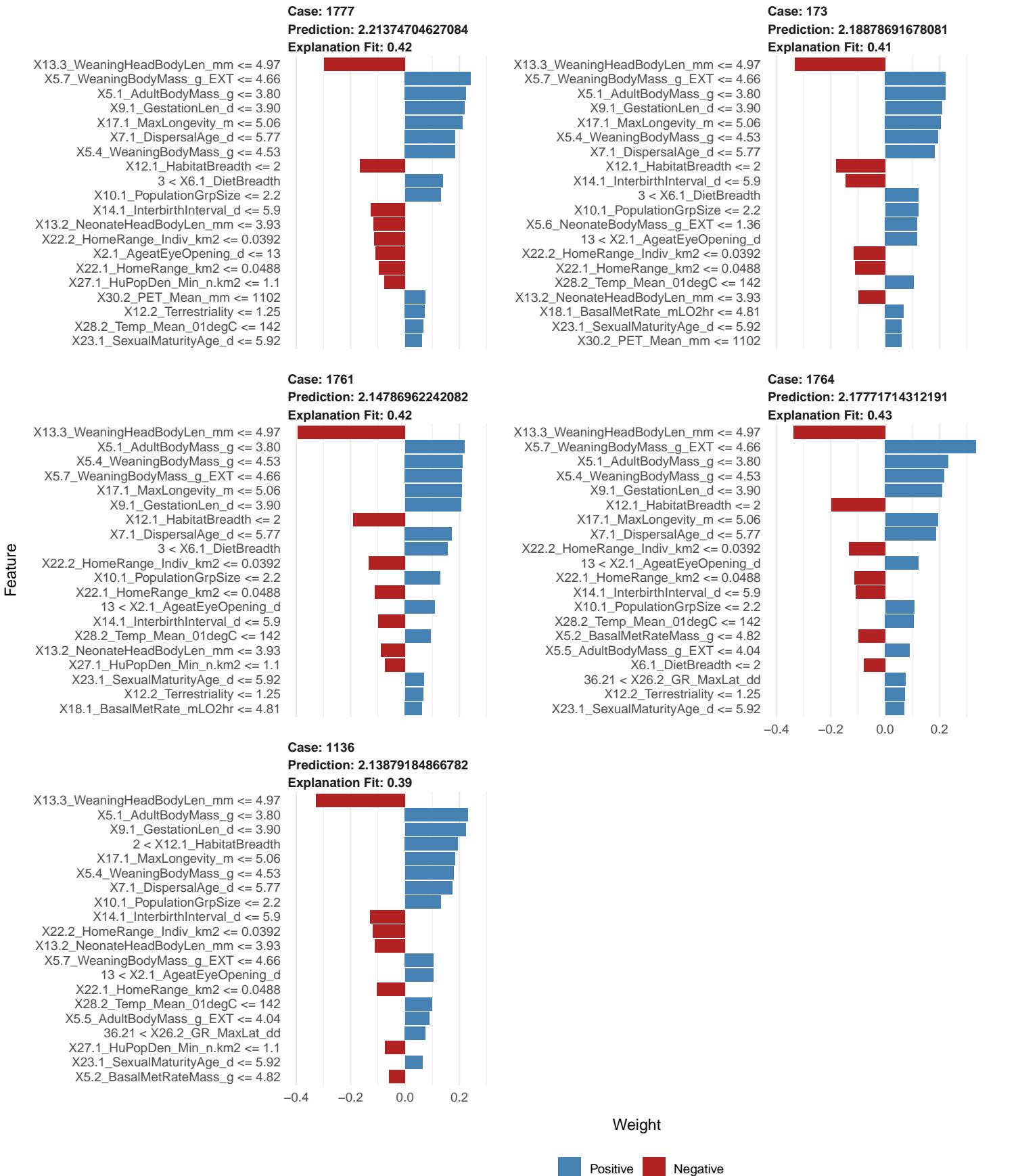
```
l1 <- pred_head(m1_enet)
l2 <- pred_head(m2_gp)
l3 <- pred_head(m3_rf)
```

```
m1_lime <- lime(p_impute, m1_enet)
```

```
## Warning: X12.2_Terrestriality does not contain enough variance to use quantile binning. Using standard binn
```

```
m1_explain <- explain(p_impute[l1, ], m1_lime, n_features = 20, feature_select = 'auto')
```

```
plot_features(m1_explain)
```

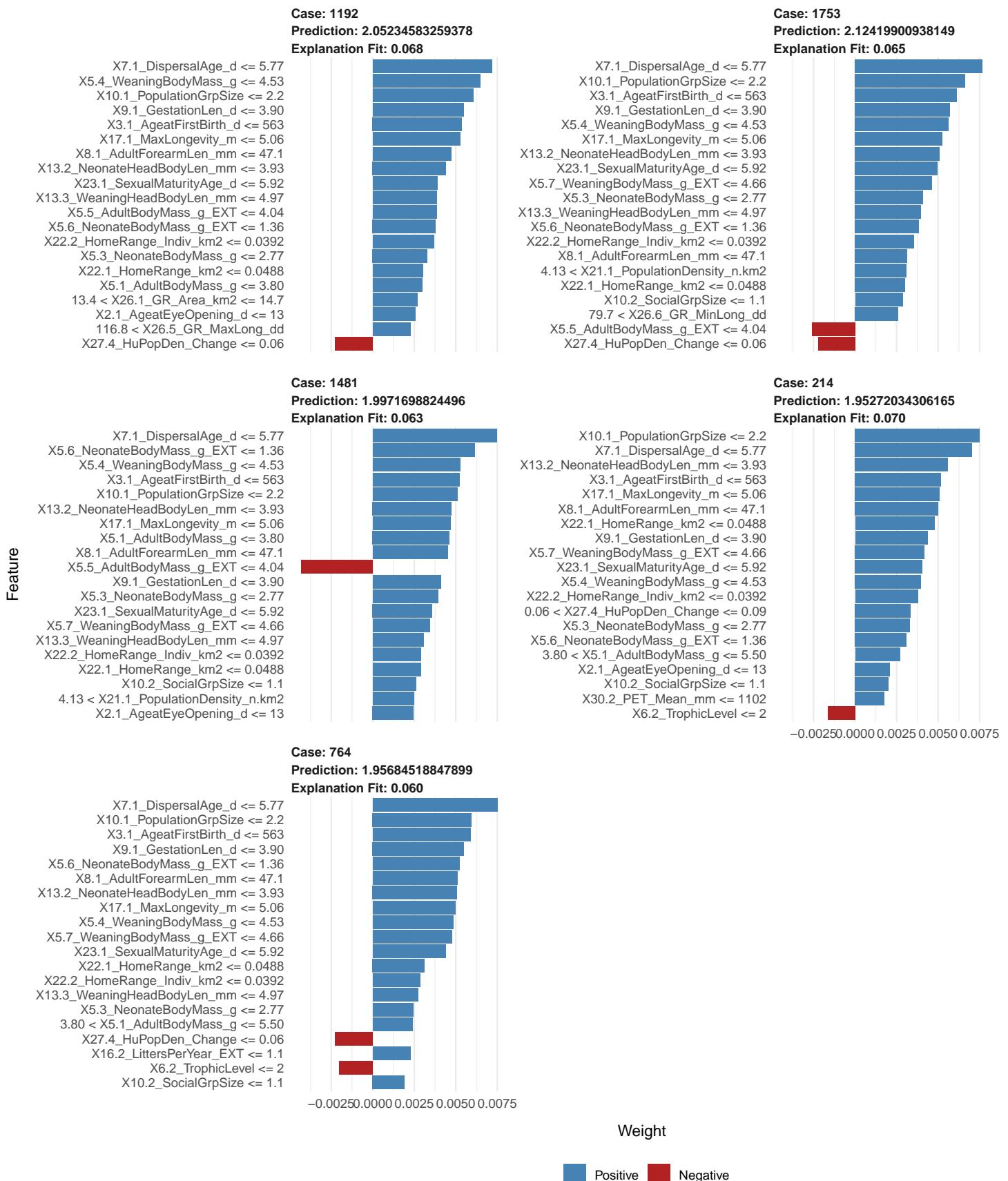


```
m2_lime <- lime(p_impute, m2_gp)
```

```
## Warning: X12.2_Terrestriality does not contain enough variance to use quantile binning. Using standard binn
```

```
m2_explain <- explain(p_impute[12, ], m2_lime, n_features = 20, feature_select = 'auto')

plot_features(m2_explain)
```

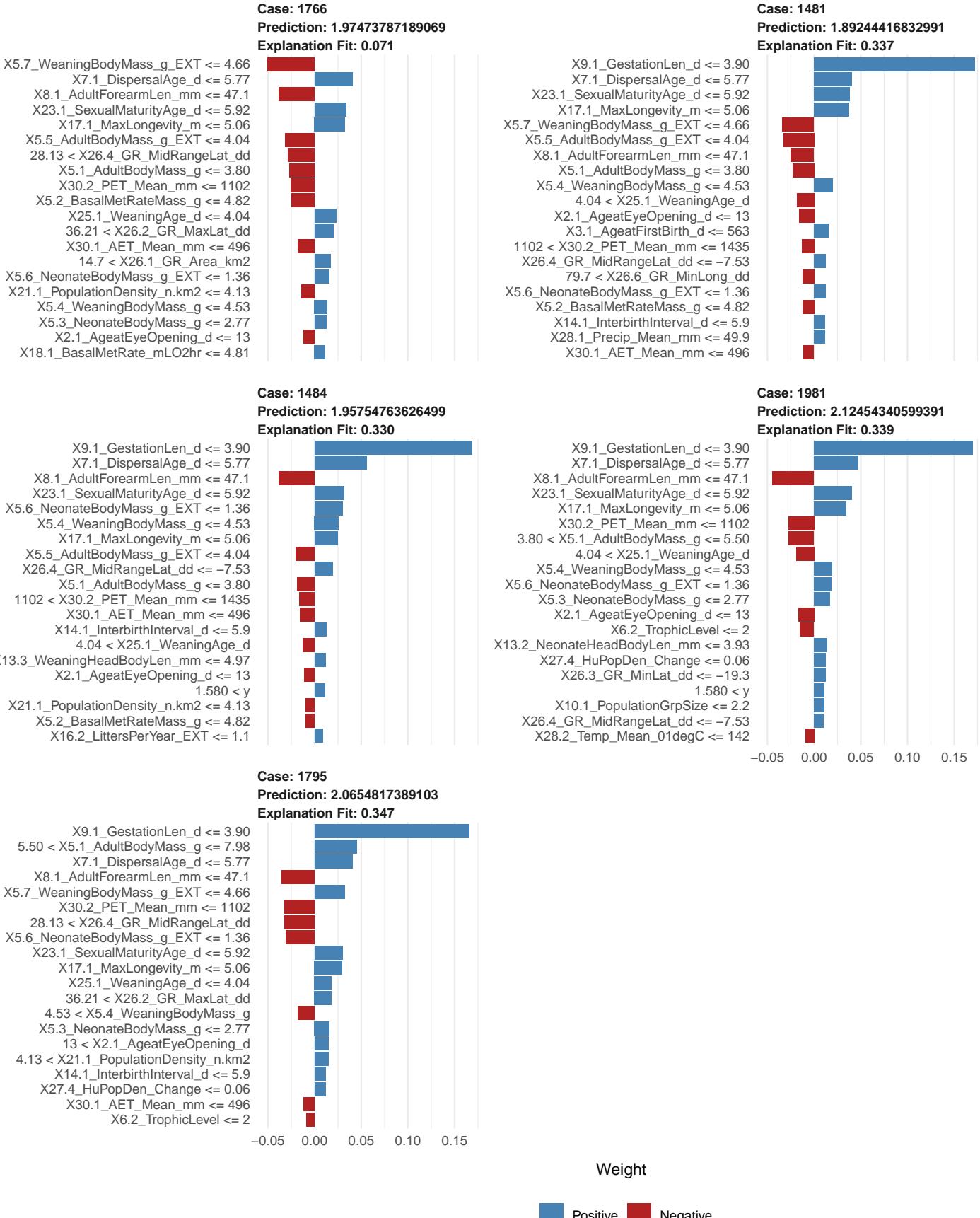


```
m3_lime <- lime(p_impute, m3_rf)

## Warning: X12.2_Terrestriality does not contain enough variance to use quantile binning. Using standard binn
m3_explain <- explain(p_impute[13, ], m3_lime, n_features = 20, feature_select = 'auto')

## Warning in predict.ranger.forest(forest, data, predict.all, num.trees, type, : Forest grown in ranger versi
plot_features(m3_explain)
```

Feature



```
12 <- pred_head(m2_gp, 2)
13 <- pred_head(m3_rf, 2)
```

```
m2_lime2 <- lime(p_impute, m2_gp, bin_continuous = FALSE, quantile_bins = FALSE)
```

```

m2_explain2 <- explain(p_impute[12, ], m2_lime2, n_features = 10, feature_select = 'auto')
m2_explain2$prediction <- round(m2_explain2$prediction, 2)

f6a <-
plot_features(m2_explain2)

m3_lime2 <- lime(p_impute, m3_rf, bin_continuous = FALSE, quantile_bins = FALSE)

m3_explain2 <- explain(p_impute[13, ], m3_lime2, n_features = 10, feature_select = 'auto')
m3_explain2$prediction <- round(m3_explain2$prediction, 2)

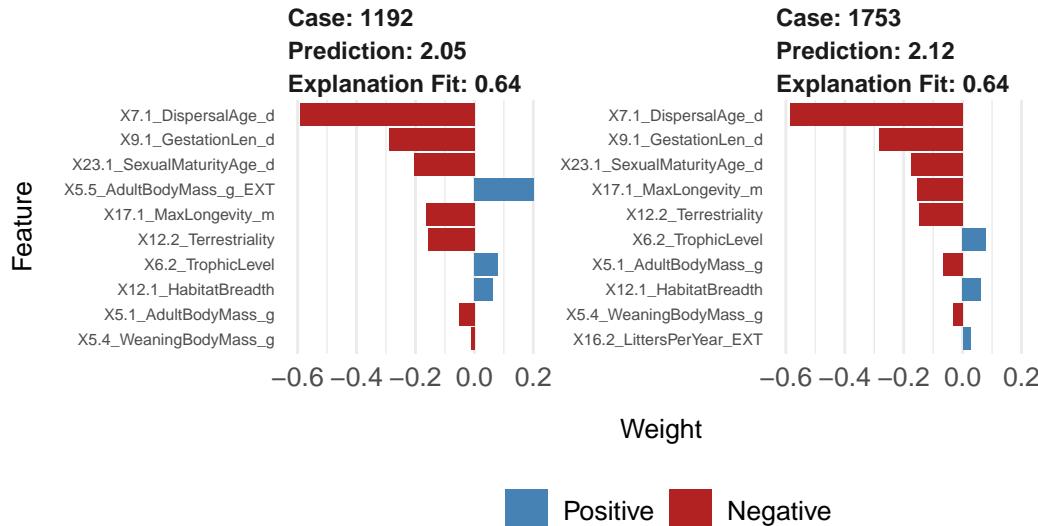
f6b <-
plot_features(m3_explain2)

f6 <- (f6a / f6b) +
  plot_annotation(tag_levels = 'A') &
  theme(text = element_text(size=7),
        axis.title=element_text(size=10),
        title = element_text(size=10),
        axis.text.x = element_text(size=10),
        legend.text = element_text(size=10) )

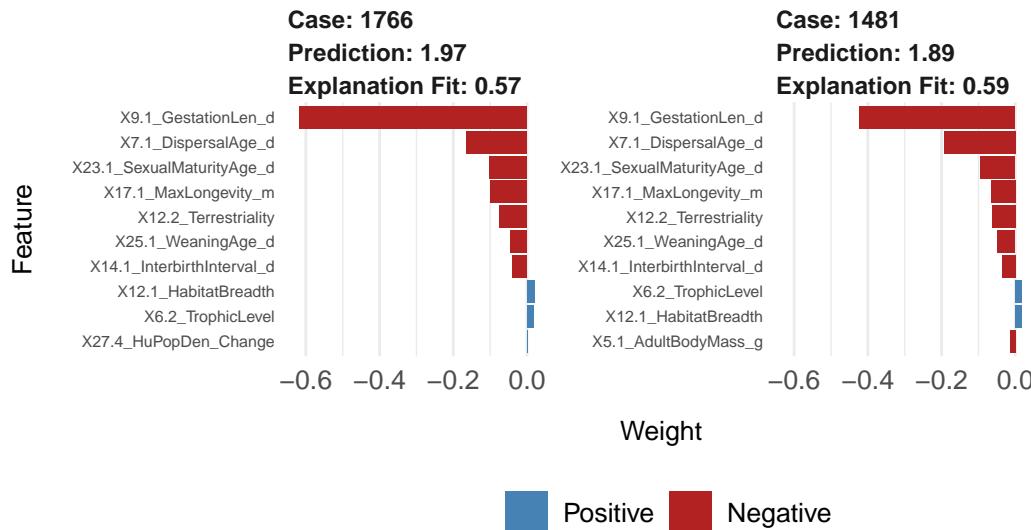
```

f6

A



B



```
f1a <-
m1_enet$results %>%
  ggplot(aes(fraction, Rsquared, colour = lambda, group = factor(lambda))) +
  geom_line() +
  geom_point() +
  scale_color_viridis_c(trans = 'log10') +
  xlab('Lasso/Ridge fraction') +
  ggtitle('Elastic net')

f1b <-
m2_gp$results %>%
  ggplot(aes(sigma, Rsquared)) +
  geom_line() +
  geom_point() +
  xlab('Sigma') +
  ggtitle('Gaussian process')

f1c <-
m3_rf$results %>%
  ggplot(aes(mtry, Rsquared, colour = factor(min.node.size), group = factor(min.node.size))) +
  geom_line() +
```

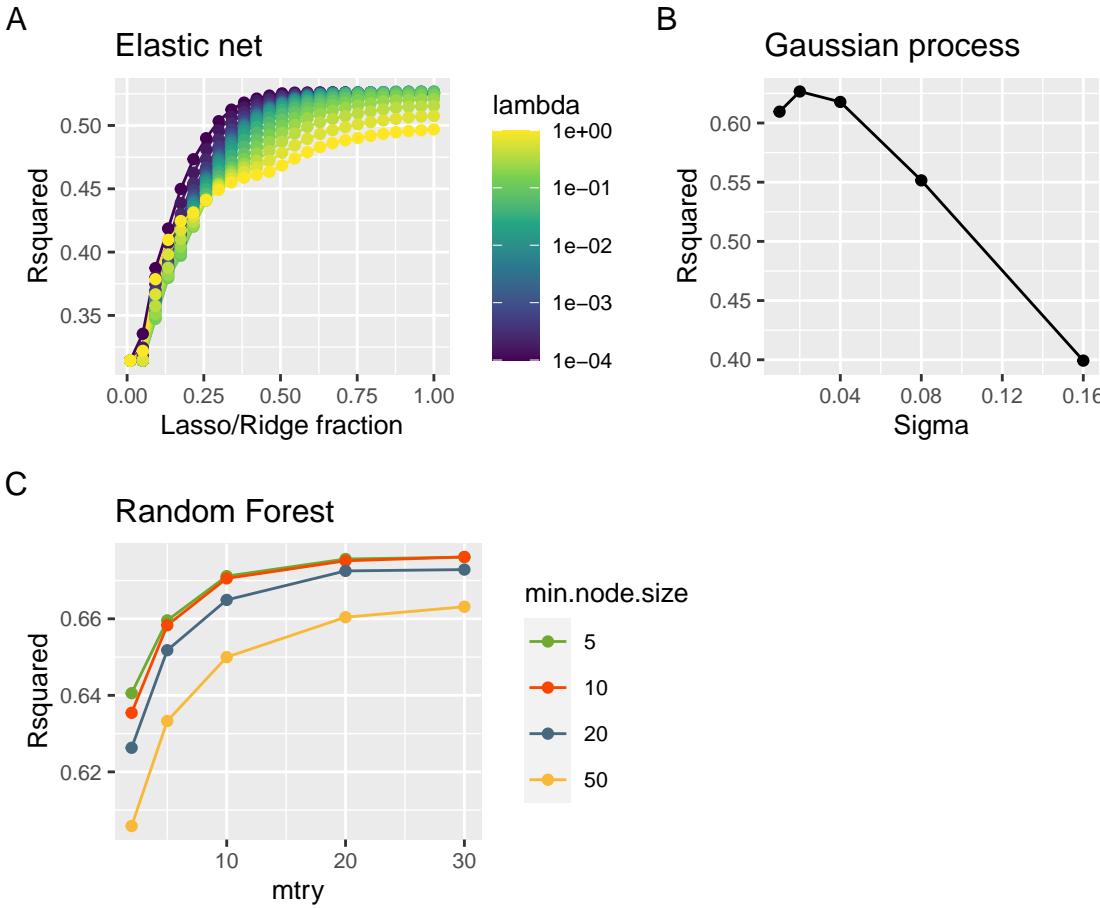
```

geom_point() +
scale_colour_poke(pokemon = 'oddish', spread = 4) +
xlab('mtry') +
labs(colour = 'min.node.size') +
ggtitle('Random Forest')

f1 <-
(f1a + f1b) / (f1c + plot_spacer()) +
plot_annotation(tag_levels = 'A') &
theme(text = element_text(size=10),
axis.title=element_text(size=10))

f1

```



```

f2a <-
plotCV(m0_lm, smooth = FALSE, print = FALSE) +
scale_colour_poke(pokemon = 'wartortle') +
xlab('Observed values') +
ylab('Predicted values') +
ggtitle('A priori model')

f2b <-
plotCV(m1_enet, smooth = FALSE, print = FALSE) +
scale_colour_poke(pokemon = 'blastoise') +
xlab('Observed values') +
ylab('Predicted values') +
ggtitle('Elastic net')

f2c <-
plotCV(m2_gp, smooth = FALSE, print = FALSE) +
scale_colour_poke(pokemon = 'parasect') +

```

```

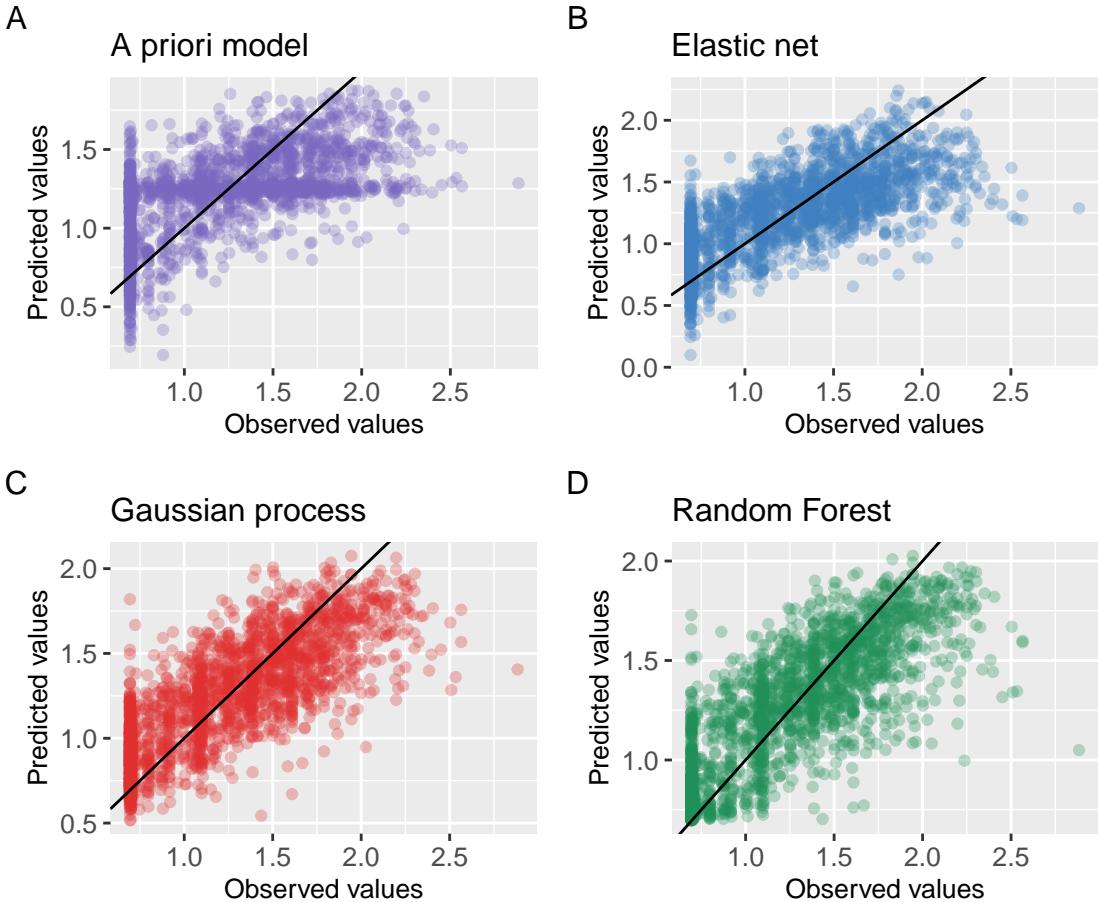
xlab('Observed values') +
ylab('Predicted values') +
ggtitle('Gaussian process')

f2d <-
plotCV(m3_rf, smooth = FALSE, print = FALSE) +
scale_colour_poke(pokemon = 'venusaur')+
xlab('Observed values') +
ylab('Predicted values') +
ggtitle('Random Forest')

f2 <-
(f2a + f2b) / (f2c + f2d) +
plot_annotation(tag_levels = 'A') &
theme(text = element_text(size=10),
axis.text = element_text(size = 10),
axis.title=element_text(size=10))

```

f2



```

f3a <-
partial(m1_enet,
pred.var = c('X9.1_GestationLen_d'),
parallel = TRUE, plot = TRUE, plot.engine = 'ggplot2') +
ggtitle('Elastic net') +
labs(x = 'Gestation Length')

f3b <-
partial(m2_gp,
pred.var = c('X9.1_GestationLen_d'),

```

```

parallel = TRUE, plot = TRUE, plot.engine = 'ggplot2')+
ggttitle('Gaussian process') +
labs(x = 'Gestation Length')

f3c <-
partial(m3_rf,
  pred.var = c('X9.1_GestationLen_d'),
  parallel = TRUE, plot = TRUE, plot.engine = 'ggplot2')+
ggttitle('Random Forest') +
labs(x = 'Gestation Length')

f3 <-
(f3a + f3b) / (f3c + plot_spacer()) +
plot_annotation(tag_levels = 'A') &
theme(text = element_text(size=10),
axis.title=element_text(size=10))

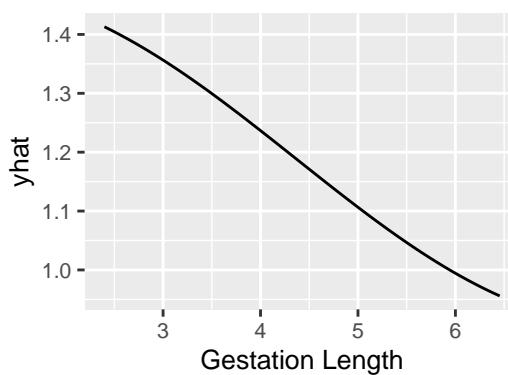
```

f3

A Elastic net B Gaussian process

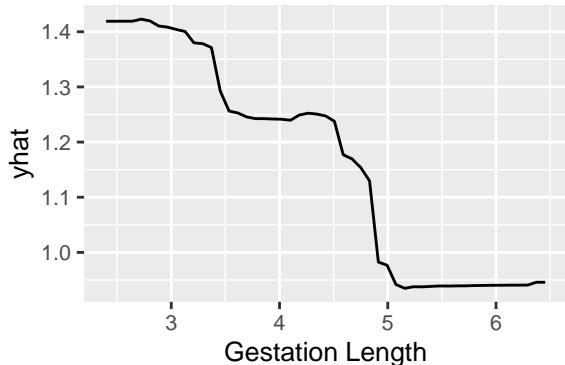


B Gaussian process



C

Random Forest



```

m1_enet_ice <- ice(m1_enet, p_impute, p_impute$y, 'X9.1_GestationLen_d', frac_to_build = 0.1)

## .....
m1_enet_ice_c <- ice(m1_enet, p_impute, p_impute$y, 'X9.1_GestationLen_d')

## .....
layout(mat = matrix(c(1,2,3,0,4,5), nrow = 2, byrow = TRUE))

par(cex = 0.7, mar = c(3.2, 3.1, 0.2, 0.1))
plot(m1_enet_ice, xlab = '')

```

```

mtext('Partial yhat', side = 2, line = 2)
mtext('Gestation Length', side = 1, line = 2.2)

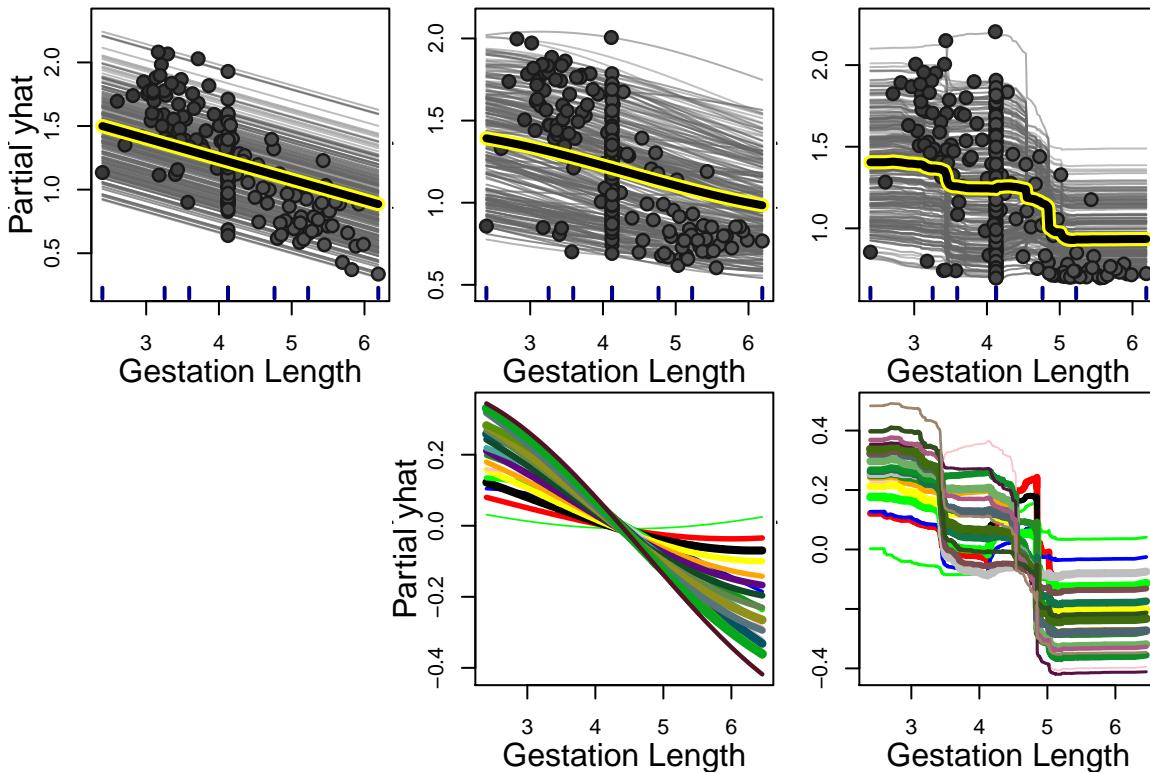
plot(m2_gp_ice, xlab = '')
mtext('Gestation Length', side = 1, line = 2.2)

plot(m3_rf_ice, xlab = '')
mtext('Gestation Length', side = 1, line = 2.2)

clusterICE(m2_gp_ice_c, nClusters = 20, centered = FALSE, plot_margin = 0)
mtext('Partial yhat', side = 2, line = 2)
mtext('Gestation Length', side = 1, line = 2.2)

clusterICE(m3_rf_ice_c, nClusters = 20, centered = FALSE, plot_margin = 0)
mtext('Gestation Length', side = 1, line = 2.2)

```



```

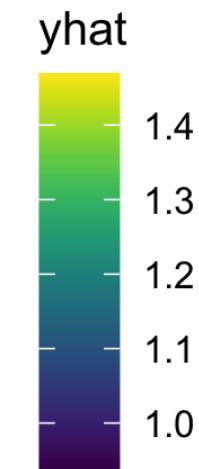
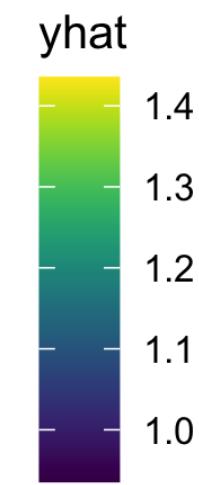
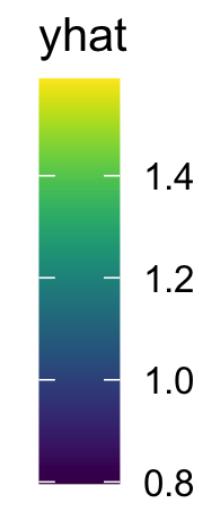
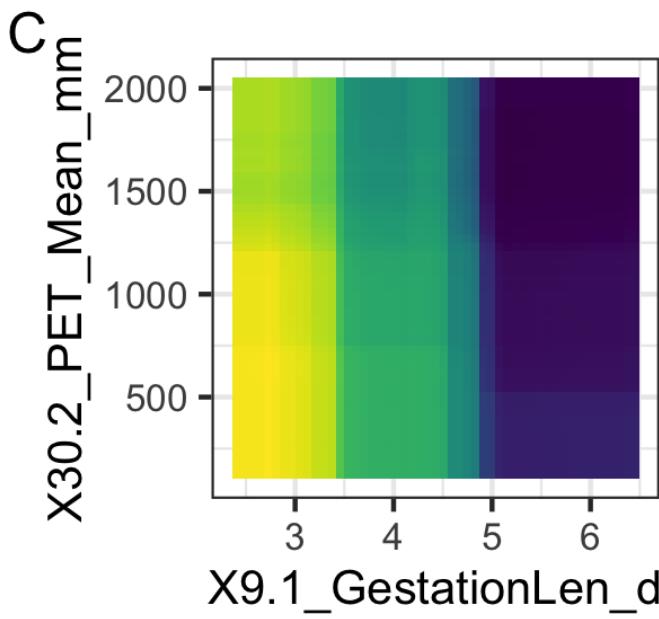
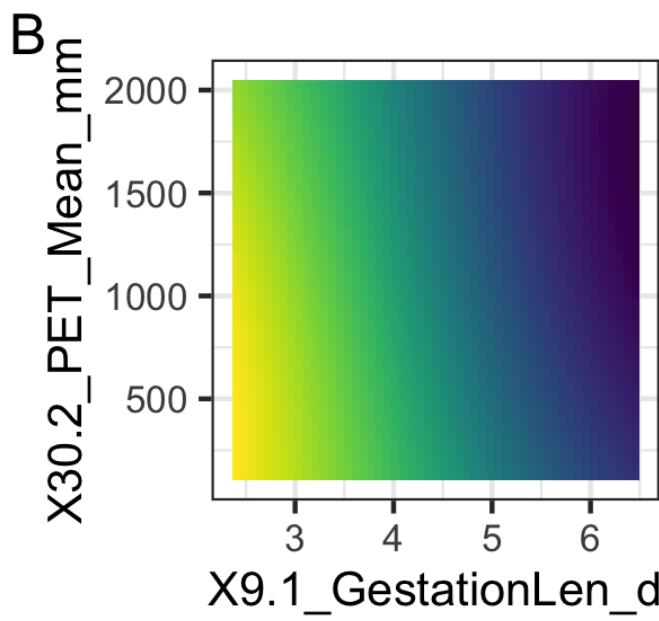
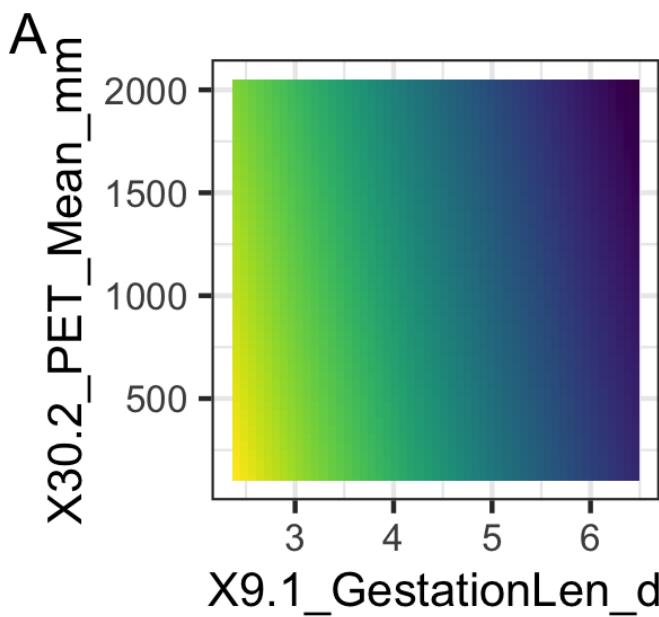
f5a <-
partial(m1_enet,
        pred.var = c('X9.1_GestationLen_d', 'X30.2_PET_Mean_mm'),
        parallel = TRUE, plot = TRUE, plot.engine = 'ggplot2')
f5b <-
partial(m2_gp,
        pred.var = c('X9.1_GestationLen_d', 'X30.2_PET_Mean_mm'),
        parallel = TRUE, plot = TRUE, plot.engine = 'ggplot2')

f5c <-
partial(m3_rf,
        pred.var = c('X9.1_GestationLen_d', 'X30.2_PET_Mean_mm'),
        parallel = TRUE, plot = TRUE, plot.engine = 'ggplot2')

f5 <-
(f5a / f5b / f5c) +
plot_annotation(tag_levels = 'A') &
theme(text = element_text(size=10),
      axis.title=element_text(size=10))

```

f5



```

sessionInfo()

## R version 3.4.4 (2018-03-15)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.5 LTS
##
## Matrix products: default
## BLAS: /usr/lib/libblas/libblas.so.3.6.0
## LAPACK: /usr/lib/lapack/liblapack.so.3.6.0
##
## locale:
## [1] LC_CTYPE=en_GB.UTF-8      LC_NUMERIC=C
## [3] LC_TIME=en_GB.UTF-8       LC_COLLATE=en_GB.UTF-8
## [5] LC_MONETARY=en_GB.UTF-8   LC_MESSAGES=en_GB.UTF-8
## [7] LC_PAPER=en_GB.UTF-8      LC_NAME=C
## [9] LC_ADDRESS=C              LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel    stats      graphics   grDevices  utils      datasets   methods
## [8] base
##
## other attached packages:
## [1] bindr_0.2.2      caper_1.0.1      mvtnorm_1.0-6
## [4] MASS_7.3-49      ape_5.2         INLA_0.0-1485844051
## [7] Matrix_1.2-14    sp_1.3-1        elasticnet_1.1.1
## [10] lars_1.2         iml_0.7.1       ICEbox_1.1.2
## [13] sfsmisc_1.1-2    pdp_0.7.0       lime_0.4.0
## [16] caret_6.0-79     lattice_0.20-38  doParallel_1.0.10
## [19] iterators_1.0.8   foreach_1.4.3   ggplot2_3.1.0
## [22] dplyr_0.7.7      rmarkdown_1.9
##
## loaded via a namespace (and not attached):
## [1] colorspace_1.3-2    class_7.3-14      rprojroot_1.2
## [4] DRR_0.0.3          MatrixModels_0.4-1 prodlim_1.6.1
## [7] lubridate_1.6.0     ranger_0.10.1    codetools_0.2-15
## [10] splines_3.4.4      mnormt_1.5-5    robustbase_0.92-8
## [13] libcoin_1.0-1      knitr_1.20.3    shinythemes_1.1.1
## [16] RcppRoll_0.2.2     Formula_1.2-2   Metrics_0.1.4
## [19] broom_0.4.2        ddalpha_1.3.2    kernlab_0.9-25
## [22] shiny_1.0.5        compiler_3.4.4  backports_1.1.0
## [25] assertthat_0.2.0   lazyeval_0.2.1   htmltools_0.3.6
## [28] tools_3.4.4        partykit_1.2-2  gtable_0.2.0
## [31] glue_1.3.0         reshape2_1.4.3  Rcpp_0.12.19
## [34] nlme_3.1-137      psych_1.7.3.21  timeDate_3043.102
## [37] inum_1.0-0        gower_0.1.2    stringr_1.3.1
## [40] mime_0.5          devtools_1.13.2 stringdist_0.9.5.1
## [43] DEoptimR_1.0-8    scales_1.0.0    ipred_0.9-6
## [46] yaml_2.1.19       memoise_1.1.0   gridExtra_2.3
## [49] rpart_4.1-13      stringi_1.2.4   randomForest_4.6-14
## [52] e1071_1.6-8       checkmate_1.8.3 lava_1.6.1
## [55] geometry_0.3-6    rlang_0.3.1     pkgconfig_2.0.2
## [58] evaluate_0.10.1   purrr_0.2.5    bindr_0.1.1
## [61] recipes_0.1.2     htmlwidgets_1.0  labeling_0.3
## [64] CVST_0.2-1        tidyselect_0.2.5 plyr_1.8.4
## [67] magrittr_1.5       R6_2.3.0       magick_2.0
## [70] dimRed_0.1.0      pillar_1.3.1   foreign_0.8-69
## [73] withr_2.1.2       mgcv_1.8-23    survival_2.43-3
## [76] abind_1.4-5       nnet_7.3-12    tibble_2.0.1
## [79] crayon_1.3.4      viridis_0.5.1  grid_3.4.4

```

```
## [82] data.table_1.12.0    ModelMetrics_1.1.0   digest_0.6.18
## [85] xtable_1.8-2          tidyR_0.8.2       httpuv_1.3.5
## [88] stats4_3.4.4          munsell_0.5.0     glmnet_2.0-16
## [91] viridisLite_0.3.0     magic_1.5-8
```

stopCluster(cl)