PREDICTING BETA-CAROTENE CONTENT IN PLASMA

Project 2

Zaide Montes Dmytro Perepolkin

LUNDS UNIVERSITET

Group 1 Lund University
Lund, Sweden
https://lu.se

```r
#install.packages("corrgram")
library(tidyverse)
library(broom)
library(corrgram) # visualisation of correlations
library(lmtest)  # more linear regression tools
library(hrbrthemes) #ggplot styling
library(GGally) # Pair plot
library(ggplot2) # ggplot 2
library(hrbrthemes) # theme for ggplot
library(kableExtra)
library(leaps)
library(glmnet)
library(patchwork)
library(plotmo)
library(randomForest) # for random forest chapter
library(pls) # for principal component regression
library(rsample) # for cross-validation
library(modelr) # evaluation of models
library(splines)
ggplot2::theme_set(theme_classic())

extrafont::loadfonts(device = "all", quiet = TRUE)

knitr::opts_chunk$set(dev = 'png')
options(device = function(file, width, height) {
  png(tempfile(), width = width, height = height)
})
```

1 Regression and tree-based models

1.1 Introduction

1.1.1 Data description

The dataset is from `gamlss.data` package (Harrell 2002). It is a cross-sectional study to investigate the relationship between personal characteristics and dietary factors, and plasma concentrations.

An original data frame has 315 observations of the following variables

Variable	Description
`age`	age(years)
`sex`	sex, 1=male, 2=female
`smokstat`	smoking status 1=never, 2=former, 3=current Smoker
`bmi`	body mass index weight/(height^2)
`vituse`	vitamin use 1=yes, fairly often, 2=yes, not often, 3=no
`calories`	number of calories consumed per day
`fat`	grams of fat consumed per day
`fiber`	grams of fiber consumed per day
`alcohol`	number of alcoholic drinks consumed per week
`cholesterol`	cholesterol consumed (mg per day)
`betadiet`	dietary beta-carotene consumed (mcg per day)
`retdiet`[1]	dietary retinol consumed (mcg per day)
`betaplasma`	plasma beta-carotene (ng/ml)
`retplasma`[2]	plasma retinol (ng/ml)

> Observational studies have suggested that low dietary intake or low plasma concentrations of retinol, beta-carotene, or other carotenoids might be associated with increased risk of developing certain types of cancer… We designed a cross-sectional study to investigate the relationship between personal characteristics and dietary factors, and plasma concentrations of retinol, beta-carotene and other carotenoids. (Harrell 2002)

[1]Not present in the current version of the dataset
[2]Not present in the current version of the dataset

1.2 Multivariate regression

`glmnet` requires a model-matrix as the model specification. A model matrix is a matrix with a column for each predictor (i.e. x-variable) in the model (including the extra columns for polynomials, dummy variables etc), and each row is for a unique observation. Sometimes the model-matrix includes a column for the intercept, in `glmnet` it should (typically) not be required, therefore we explicitly remove the first column.

```
plasma_df <- read_csv("data/plasma.txt", show_col_types = FALSE)
x <- model.matrix(log(betaplasma) ~ ., plasma_df)[, -1]
y <- log(plasma_df$betaplasma)
```

1.2.1 Ridge regression

```
# Grid of lambdas
grid <- 10^seq(5, -2, length = 100) #grid

# Run ridge regression (alpha =0) for each lambda in grid
ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

The `ridge.mod` contains 13 rows and 100 columns corresponding to the number of λ values we created earlier.

```
# Prepare for cross validation. Create training and validation set
set.seed(42)
train <- sample(1:nrow(x), nrow(x) / 2)
test <- (-train)
y.test <- y[test]
```

```
# Fit ridge regression on the training data, for our grid of lambda
ridge.mod <- glmnet(x[train, ], y[train], alpha = 0,
    lambda = grid, thresh = 1e-12)
# Find predictions for our testdata for a specific lambda (here lambda =4)
ridge.pred <- predict(ridge.mod, s = 4, newx = x[test, ])
```

The MSE for the $\lambda = 4$ is 0.4227268. Figure 1.1 shows the ridge model coefficients approaching to zero, as lambda increases.

```
#plot(ridge.mod,xvar="lambda",lwd=1.5)
plot_glmnet(ridge.mod,xvar="lambda",lwd=1.5)
```

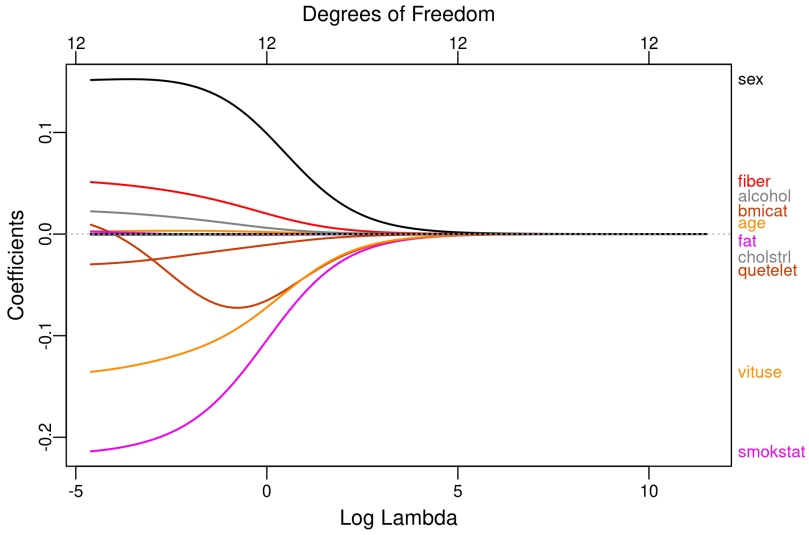


Figure 1.1: The ridge regression coefficients for the plasma data set.

In the Figure 1.2 the dashed lines are the $\ln(\lambda)$ values corresponding to the λ_{min} (left dashed line) and λ_{1se} (right dashed line).

```
set.seed(1)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 0)
plot(cv.out)
```

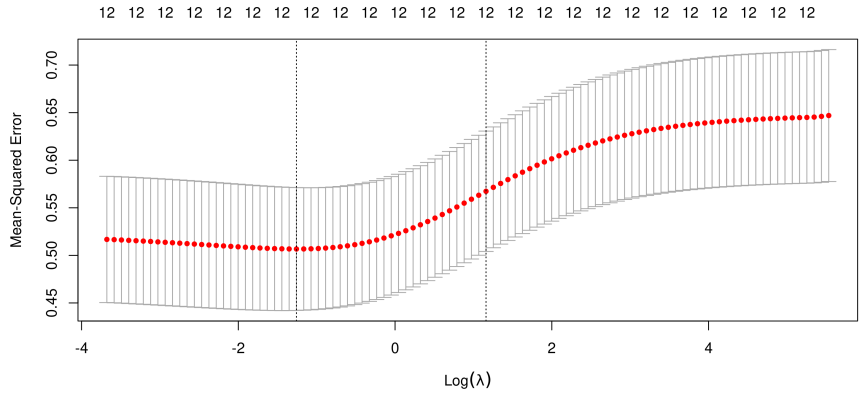


Figure 1.2: A graph of the ridge model coefficients for different lambdas.

```
# Choose the best:
bestlam <- cv.out$lambda.min
```

Figure 1.3 shows the coefficients for the ridge regression corresponding to the best lambda value $\lambda = 0.2841768$.

```
out <- glmnet(x, y, alpha = 0, lambda = grid)
ridge.coef <- predict(out, type = "coefficients",
    s = bestlam)

ridge.coef %>% as.matrix() %>% as.data.frame() %>%
  rownames_to_column()%>%
  arrange(-abs(s1)) %>%
    filter(!str_detect(rowname, "Intercept")) %>%
  ggplot()+geom_col(aes(x=s1,y=fct_reorder(rowname, abs(s1))))+
  labs(y="Coefficient", x="value")
```

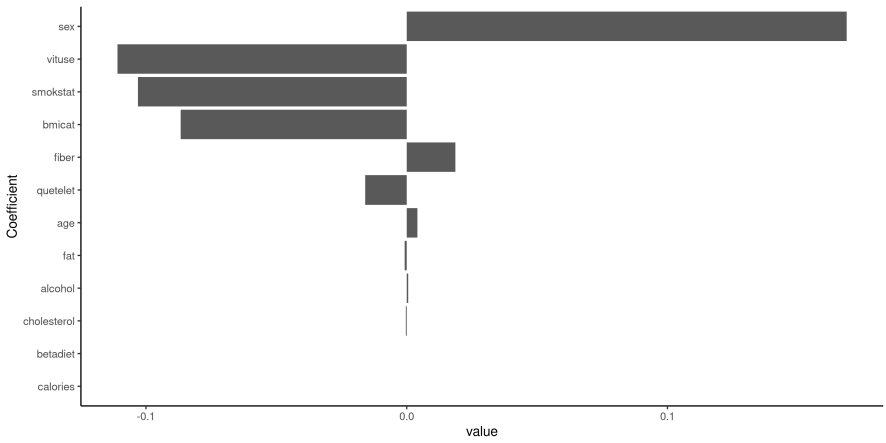


Figure 1.3: Coefficients for the ridge model under best lambda

```
# MSE associated with this lambda
ridge.pred <- predict(ridge.mod, s = bestlam,
    newx = x[test, ])
ridge.test.mse <- mean((ridge.pred - y.test)^2)
```

The MSE turns out to be 0.4320415.

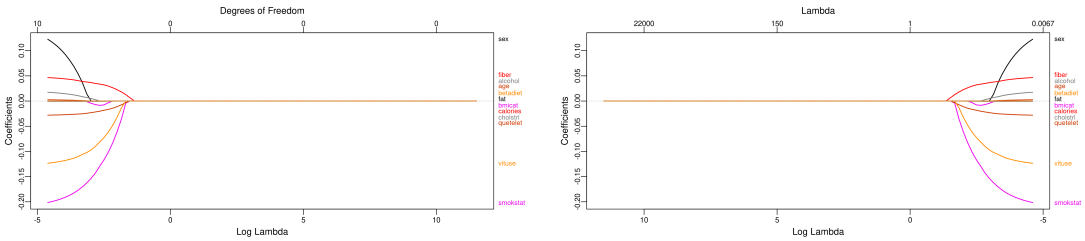
1.2.2 Lasso regression

```
# Fit lasso model: alpha=1. Use only the training data
lasso.mod <- glmnet(x[train, ], y[train], alpha = 1,
    lambda = grid)
```

Figure 1.4 shows the coefficients against the penalty λ.

```
plot_glmnet(lasso.mod,label=TRUE,xvar="lambda",lwd=1.5)
plot_glmnet(lasso.mod,label=TRUE,lwd=1.5)
```

```
set.seed(1)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 1)
plot(cv.out)
```



(a) against df. (b) against lambda.

Figure 1.4: The Lasso regression coefficients

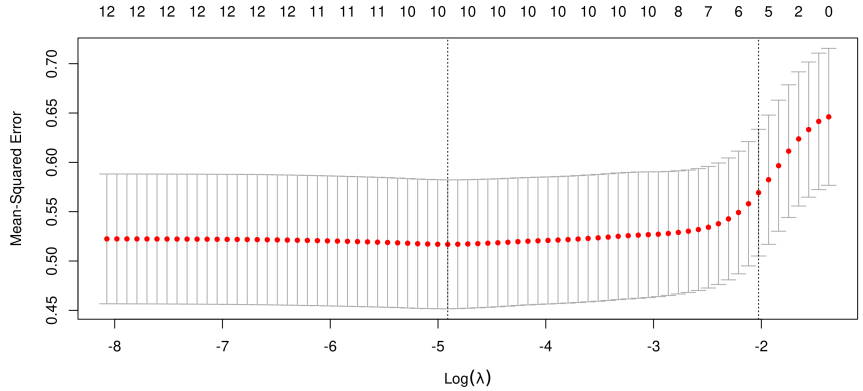


Figure 1.5: Cross-validated penalty for the lasso model.

The Figure 1.5 illustrates the cross validation process for picking the best lambda in lasso regression. The dashed lines are the log λ values corresponding to the λ_{min} (left dashed line) and λ_{1se} (right dashed line)

```
bestlam2 <- cv.out$lambda.min
```

The best lambda value that minimizes the test MSE turns out to be $\lambda = 0.0073745$.

```
# compute the test error (MSE) associated with this lambda
lasso.pred <- predict(lasso.mod, s = bestlam2,
    newx = x[test, ])
lasso.test.mse <- mean((lasso.pred - y.test)^2)
```

The test MSE corresponding to the best lambda is 0.4807286. The Figure 1.6 shows the coefficients of lasso regression corresponding to the best lambda.

```
out <- glmnet(x, y, alpha = 1, lambda = grid)

lasso.coef <- predict(out, type = "coefficients",
    s = bestlam2)

lasso.coef %>% as.matrix() %>% as.data.frame() %>%
  rownames_to_column()%>%
  arrange(-abs(s1)) %>%
  filter(!str_detect(rowname, "Intercept")) %>%
  ggplot()+geom_col(aes(x=s1,y=fct_reorder(rowname, abs(s1))))+  labs(y="Coefficient", x=
```

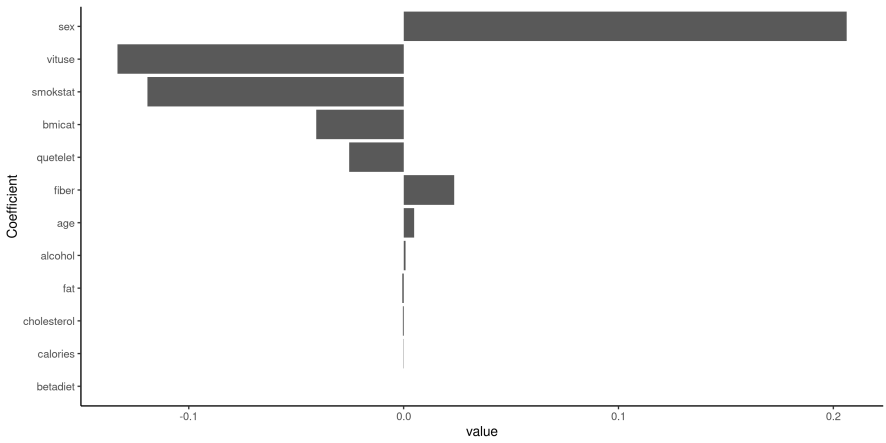


Figure 1.6: Coefficients for the lasso model under best lambda

The variance of ridge regression is slightly lower than the variance of the lasso. Consequently, the minimum MSE of ridge regression is slightly smaller than that of the lasso. In addition, the models generated from the lasso were much easier to interpret than those produced by ridge regression. The lasso yields sparse models that involve only a subset of the variables. Hence, depending on the value of λ, the lasso can produce a model involving reduced number of variables. In contrast, ridge regression will always include all of the variables in the model, although the magnitude of the coefficient estimates will depend on the λ. In the lasso regression we observed that the variables that influenced more in the model are the `sex`, `vituse`, `smokestat`, `bmicat` (Figure 1.6). To sum up, the lasso should perform better in a setting where a relatively small number of predictors have significant coefficients. The remaining predictors will be regularized to have have very small coefficients or will be completely compressed to zero. In contrast, the ridge regression will perform better when the response is a function of many predictors, all with roughly equal-sized coefficients.

1.2.3 Principle component regression

```
pcr_mod <- pls::pcr(log(betaplasma)~., data=plasma_df, scale=TRUE, validation="CV")
```

Figure 1.7 shows the out-of-fold (cross-validated) MSE for each number of components in the PCR.

```
validationplot(pcr_mod, val.type = "MSEP", main="")
```

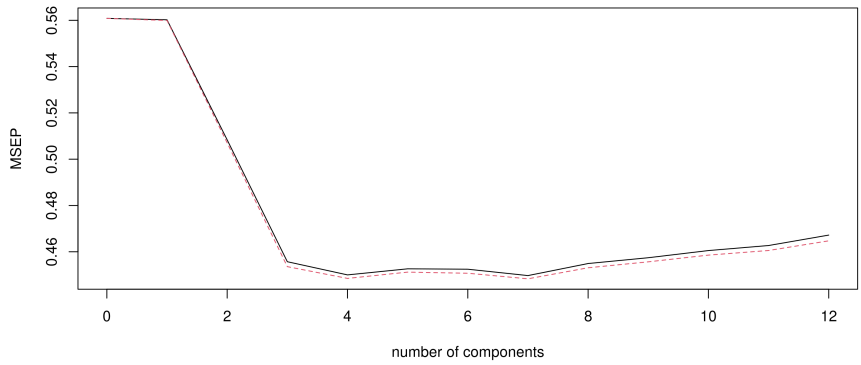


Figure 1.7: Cross-validated MSE for PCR model.

The first 8 principal components seems to be sufficient to explain most of the variability in the data, as well as the relationship with the response. Let's perform CV on the training dataset and see if we can reduce the model even further.

```
set.seed(1)
pcr.fit <- pcr(log(betaplasma)~., data=plasma_df, ncomp=8,
               subset = train, scale = TRUE, validation = "CV")
validationplot(pcr.fit, val.type = "MSEP",lwd=2,legendpos = "topright")
```

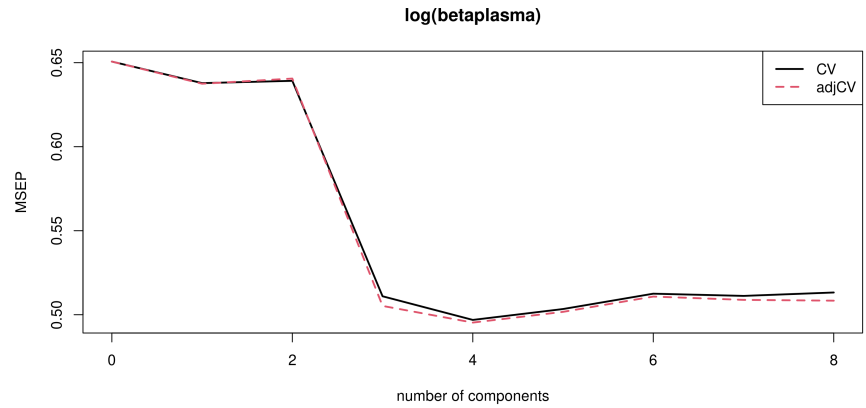


Figure 1.8: MSE for PCR model cross-validated on the training dataset.

The figure Figure 1.8 indicates that performing PCR with 4 components can results in substantial improvement over the least squares. Thus, we select the lowest number of components which minimizes MSE.

```
pcr.pred <- predict(pcr.fit, x[test, ], ncomp = 4)
pcr.test.mse <- mean((pcr.pred - y.test)^2)
```

The test MSE for the PCR is 0.41016.

1.2.4 Random Forest

We will now use the random forest algorithm implemented in `randomForest` package in R.

The parameter `mtry` in random forest algorithm determines how many columns are consered for each split. We will have to cross-validate this parameter to make sure we are not overfitting to our dataset. We will now determine the best value of `mtry` parameter to use in our random forest model. Figure 1.9 shows the out-of-fold MSE for the models with different `mtry`. The `tuneRF` function uses the out-of-bag predictions, therefore we will adjust the step factor and increase the number of trees to produce a reliable estimate.

```
set.seed(42)
tuneRF(x[train,], y[train], mtryStart = 3,
       stepFactor=1.5, ntreeTry = 1000)
```

```
mtry = 3  OOB error = 0.5737914
Searching left ...
mtry = 2    OOB error = 0.5744733
-0.001188488 0.05
Searching right ...
mtry = 4    OOB error = 0.5856382
-0.02064658 0.05
```

```
  mtry  OOBError
2    2 0.5744733
3    3 0.5737914
4    4 0.5856382
```

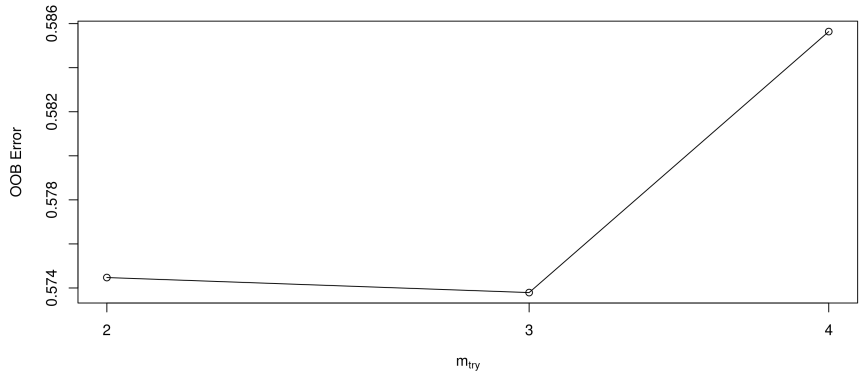


Figure 1.9: Cross-validated tuning of parameter mtry

The Figure 1.10 shows the predicted values against the true values for the test portion of our dataset using the best `mtry` value.

```
rf_fit <- randomForest(log(betaplasma)~., data=plasma_df, mtry=3,
                       subset = train, importance=TRUE)
```

```
rf_pred <- predict(rf_fit, newdata = plasma_df[test,])
plot(rf_pred, log(plasma_df$betaplasma)[test],
     ylab="log(betaplasma)", xlab="predicted")
abline(0,1)
rf.mse <- mean((rf_pred-y.test)^2)
```

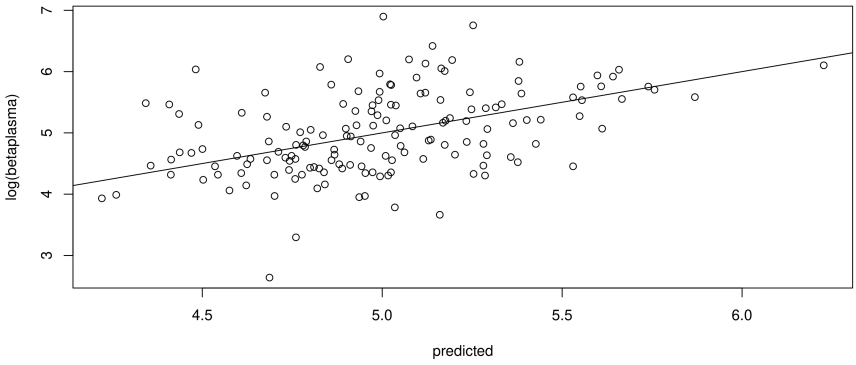


Figure 1.10: Predicted vs true values for random forest algorithm

The test MSE for random forest is 0.3815397, which is pretty impressive. As for any tree-based algorithm, interpretability is usually an issue. We can use the special function in the `randomForest` package to see which variables were most frequently used in the tree splits (Figure 1.11).

```
varimp_m <- importance(rf_fit) #matrix form
varImpPlot(rf_fit, main="")
```

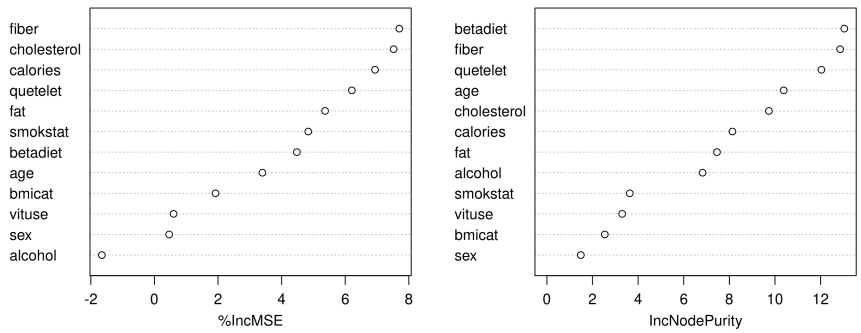


Figure 1.11: Variable importance plot for the tuned random forest

It seems like fiber, cholesterol, calories, quetelet, and fat are the most important variables for predicting the level of beta-carotene in plasma.

Table 1.2: Test MSE for various models of plasma dataset

OLS	Ridge	Lasso	PCR	RF
0.505	0.432	0.481	0.41	0.382

1.3 Conclusion

```
# Fit ordinary least squares regression on the training data
ols.mod <- lm(log(betaplasma) ~ ., data=plasma_df, subset = train)
# Find predictions for our testdata
ols.pred <- predict(ols.mod, newdata = plasma_df[test, ])
ols.test.mse <- mean((ols.pred - y.test)^2)

res <- tibble::tibble(
  ols=ols.test.mse,
  ridge=ridge.test.mse,
  lasso=lasso.test.mse,
  pcr=pcr.test.mse,
  rf=rf.mse
)
kableExtra::kbl(res, digits=3, booktabs = TRUE,
                col.names = c("OLS", "Ridge", "Lasso", "PCR", "RF"))
```

In conclusion, all three regression methods offer some improvement over the least squares (Table 1.2). However, PCR and ridge regression slightly outperform the lasso regression. Thus, although PCR tends to work exceptionally well in many practical settings, this method only allows models to be developed based on a small set of features. Thus, PCR is more closely related to the ridge regression than to lasso regression.

However, our tree-based (random forest) model blows all of the regression models out of the water with an impressive drop in MSE below that of the principle component regression. Although tree-based models may suffer from overfitting, our random forest model with cross-validated `mtry` parameter provides a robust alternative with a reliable predictive power, albeit without the interpretability of the regression-based models.

2 Non-linear effects

2.1 Introduction

Now we are proceeding to developing a model for predicting the Greenhouse gas emissions using the non-linear effects, namely *splines*. The Berkeley Earth data (`Temp_BE.csv`) provides the annual anomalies from the global mean temperature, calculated from the period Jan 1951-Dec 1980, which amounted to 14.105°C with 95% CI (14.080°C, 14.130°C). The greenhouse gas emission dataset (`GHG_1880_2014.csv`) records the greenhouse gas emission pathways for every country and Kyoto gas covering the years 1850 to 2017 (PRIMAP). We join the annual `KYOTOGHG` observations to the temperature anomalies data.

```
TEMPBE_df <- read_csv("data/Temp_BE.csv", show_col_types = FALSE) %>%
  mutate(abs_temp=14.105+anomaly_1y) %>%
  select(year, abs_temp)

GHG_df <- read_csv("data/GHG_1880_2014.csv", show_col_types = FALSE) %>%
  rename(GHG=KYOTOGHG) %>%
  mutate(GHG_cum=cumsum(replace_na(GHG, 0)))

GHG_TEMP_df <- left_join(GHG_df, TEMPBE_df, by="year")
```

2.2 Natural spline model

We proceed to fit the non-linear (natural spline) model and the baseline (OLS) models for comparison. For our first model we choose natural splines with 4 degrees of freedom.

```
ns_mod <- lm(GHG_cum~ns(abs_temp, df=4), data=GHG_TEMP_df)
```

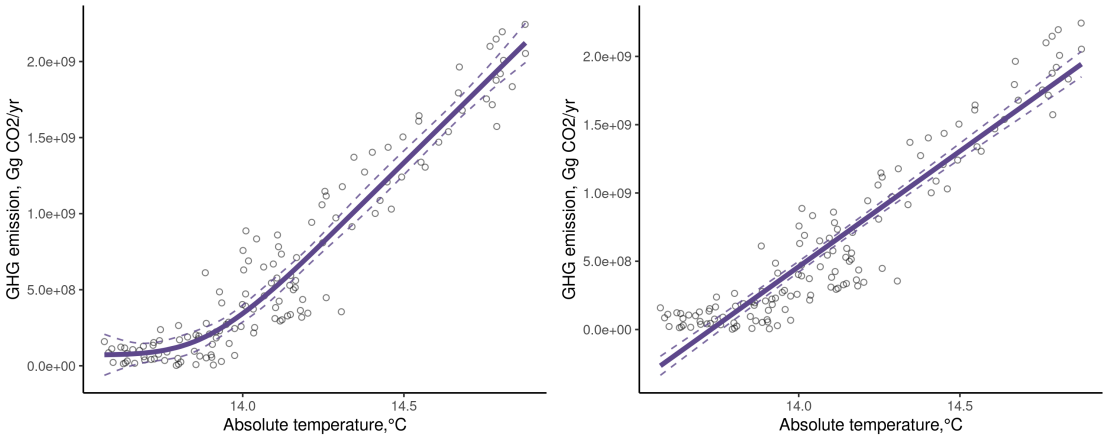
```
lm_mod <- lm(GHG_cum~abs_temp, data=GHG_TEMP_df)
```

Figure 2.1 shows the comparison of the spline model to the linear model.

```
predns_df <- predict(ns_mod, se = TRUE) %>%
  as_tibble() %>%
  bind_cols(GHG_TEMP_df)
predlm_df <- predict(lm_mod, se=TRUE) %>%
  as_tibble() %>%
  bind_cols(GHG_TEMP_df)

GHG_TEMP_df %>% ggplot(aes(x=abs_temp,y=GHG_cum))+
  geom_point(shape=21,alpha=0.5) +
  geom_line(data=predns_df,aes(y=fit),
            color="mediumpurple4",linewidth=1.5) +
  geom_line(data=predns_df,aes(y=fit + 2*se.fit) ,
            color="mediumpurple4",linewidth=0.5,
            lty="dashed",alpha=0.8) +
  geom_line(data=predns_df,aes(y=fit - 2*se.fit) ,
            color="mediumpurple4",linewidth=0.5,
            lty="dashed",alpha=0.8)+
  labs(x="Absolute temperature,°C", y="GHG emission, Gg CO2/yr")

GHG_TEMP_df %>% ggplot(aes(x=abs_temp,y=GHG_cum))+
  geom_point(shape=21,alpha=0.5) +
  geom_line(data=predlm_df,aes(y=fit),
            color="mediumpurple4",linewidth=1.5) +
  geom_line(data=predlm_df,aes(y=fit + 2*se.fit) ,
            color="mediumpurple4",linewidth=0.5,
            lty="dashed",alpha=0.8) +
  geom_line(data=predlm_df,aes(y=fit - 2*se.fit) ,
            color="mediumpurple4",linewidth=0.5,
            lty="dashed",alpha=0.8)+
  labs(x="Absolute temperature,°C", y="GHG emission, Gg CO2/yr")
```

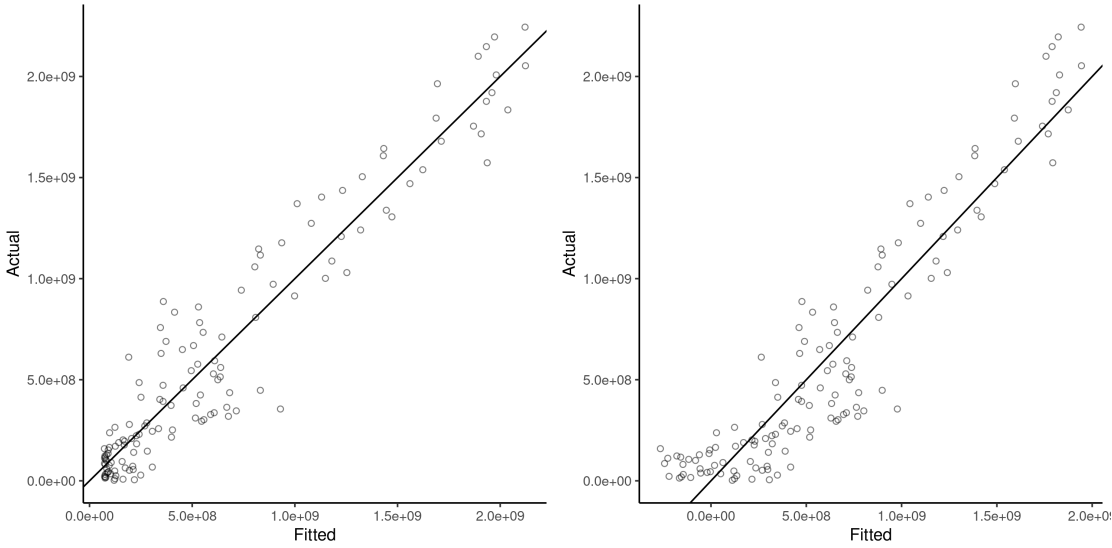
We also compare the fitted vs. the actual values in Figure 2.2.

```
predns_df %>% ggplot(aes(x=fit,y=GHG_cum))+
  geom_point(shape=21,alpha=0.5) +
  geom_abline(slope=1, intercept = 0)+
  labs(x="Fitted", y="Actual")

predlm_df %>% ggplot(aes(x=fit,y=GHG_cum))+
  geom_point(shape=21,alpha=0.5) +
  geom_abline(slope=1, intercept = 0)+
  labs(x="Fitted", y="Actual")
```



(a) natural spline model (b) linear model

Figure 2.1: Original data with the fitted values (including standard error)



(a) natural spline model (b) linear model

Figure 2.2: Fitted vs actual values

Now, we tune the spline model changing the degrees of freedom and measuring the out-of-fold error. Figure 2.3 shows the raw and adjusted ten-fold cross-validated MSE for splines with various degrees of freedom fit to the data[1].

```
set.seed(42)
cv_error <- matrix("numeric", nrow=10, ncol=2)
for(i in 1:10){
  ghg_glm_fit <- glm(GHG_cum~ns(abs_temp, df=i),
                     data=GHG_TEMP_df)
  cv_error[i,] <- boot::cv.glm(GHG_TEMP_df, ghg_glm_fit, K=10)$delta
}
#cv_error
plot(cv_error[,1], type="l", xlab="df", ylab="MSE")
lines(cv_error[,2], lty=2, col="firebrick")
```

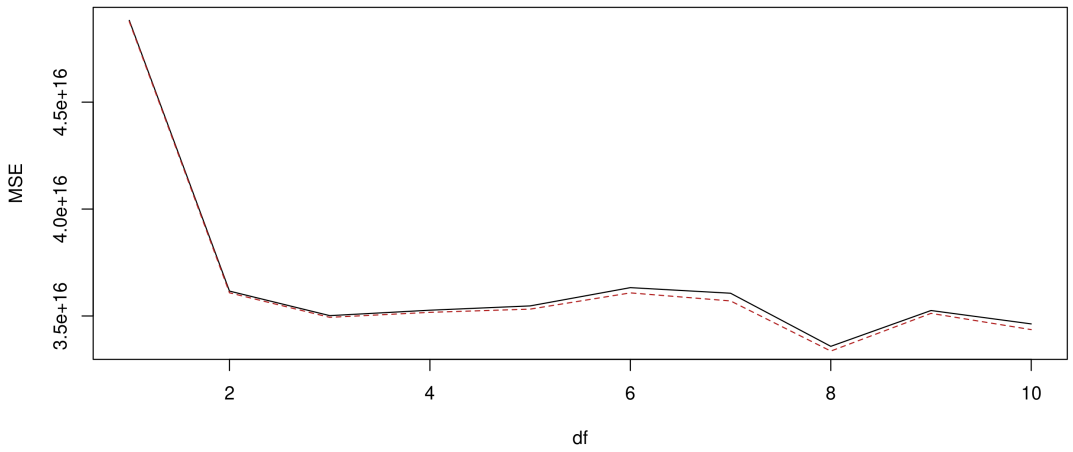


Figure 2.3: Out-of-fold MSE for different values of df

```
ns_mod3 <- lm(GHG_cum~ns(abs_temp, df=3), data=GHG_TEMP_df)
ns_mod8 <- lm(GHG_cum~ns(abs_temp, df=8), data=GHG_TEMP_df)
```

We can compare splines of different complexity (Figure 2.4)

[1]The first element in the `delta` output from the `boot::cv.glm` is the raw cross-validation estimate of prediction error. The second component is the adjusted cross-validation estimate. The adjustment is designed to compensate for the bias introduced by not using leave-one-out cross-validation.

```
predns3_df <- predict(ns_mod3, se = TRUE) %>%
  as_tibble() %>%
  bind_cols(GHG_TEMP_df)
predns8_df <- predict(ns_mod8, se = TRUE) %>%
  as_tibble() %>%
  bind_cols(GHG_TEMP_df)

GHG_TEMP_df %>% ggplot(aes(x=abs_temp,y=GHG_cum))+
  geom_point(shape=21,alpha=0.5) +
  geom_line(data=predns3_df,aes(y=fit),
            color="mediumpurple4",linewidth=1.5, alpha=0.5) +
  geom_line(data=predns3_df,aes(y=fit + 2*se.fit) ,
            color="mediumpurple4",linewidth=0.5,
            lty="dashed",alpha=0.8) +
  geom_line(data=predns3_df,aes(y=fit - 2*se.fit) ,
            color="mediumpurple4",linewidth=0.5,
            lty="dashed",alpha=0.8)+
    geom_line(data=predns8_df,aes(y=fit),
            color="firebrick",linewidth=1.5, alpha=0.5) +
  geom_line(data=predns8_df,aes(y=fit + 2*se.fit) ,
            color="firebrick",linewidth=0.5,
            lty="dashed",alpha=0.8) +
  geom_line(data=predns8_df,aes(y=fit - 2*se.fit) ,
            color="firebrick",linewidth=0.5,
            lty="dashed",alpha=0.8)+
  labs(x="Absolute temperature,°C", y="GHG emission, Gg CO2/yr")
```

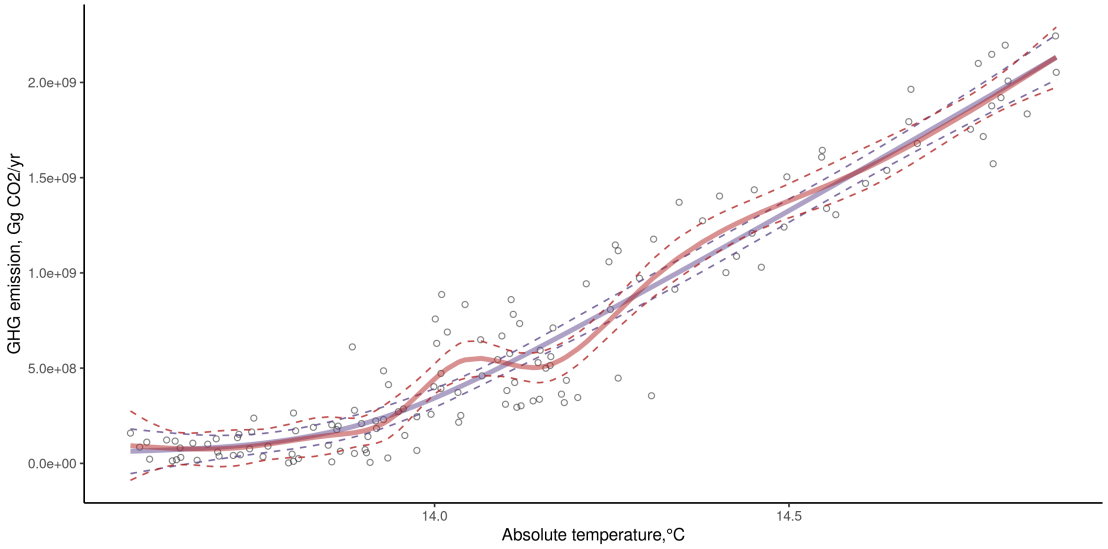


Figure 2.4: Original data with the fitted values (natural splines with df 3 and 8)

2.3 Conclusion

Figure Figure 2.4 compares a natural cubic spline with 3 degrees of freedom, purple line, to a degree-8 polynomial, red line, on the data set. The flexibility in the degree-8 polynomial produces a slightly better fit to the data, capturing the deviations around 14.0-14.2 degrees. However, this variance in the observations might just represent the noise and as a result, the more complex model would overfit. It would be, therefore, prudent to choose a simpler model with a robust predictive power, which has a higher chance of performing equally well on the new observations.

References

Harrell, Frank. 2002. “Plasma Retinol and Beta-Carotene Dataset.” 2002. https://hbiostat.org/data/repo/plasma.html.