PREDICTING BETA-CAROTENE CONTENT IN PLASMA

Project 2

Zaide Montes Dmytro Perepolkin

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
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 Introduction

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)

We import the data

```r
plasma_df <- read_csv("data/plasma.txt", show_col_types = FALSE)
```

> 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

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

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)

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.

```
#head(plasma_df)
x <- model.matrix(log(betaplasma) ~ ., plasma_df)[, -1]
#head(x)
y <- log(plasma_df$betaplasma)
#head(y)
```

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 by splitting the data in 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)
```

```r
# 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 2.1 shows the ridge model coefficients are approaching to zero.

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

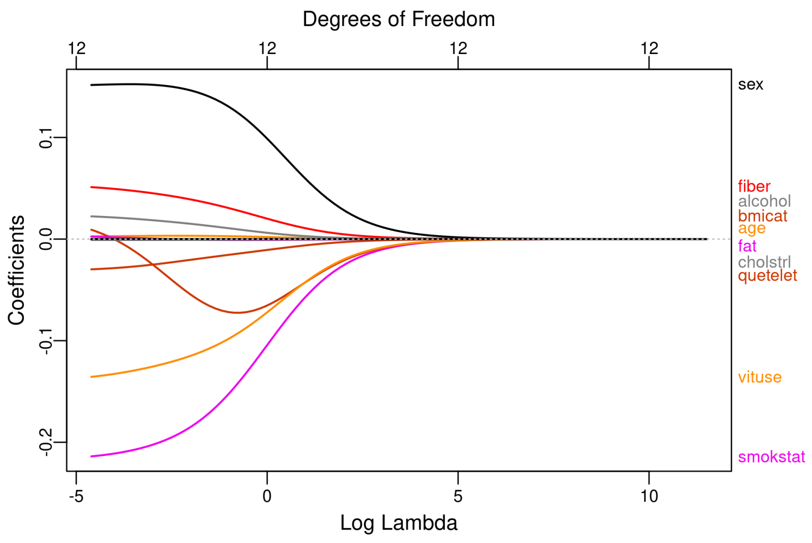


Figure 2.1: The ridge regression coefficients are displayed for the Plasma data set, as a function of .

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

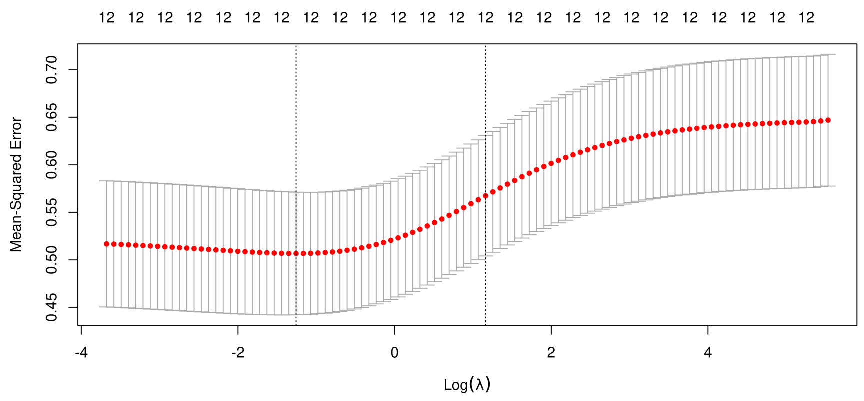


Figure 2.2: A graph of the ridge model coefficients for different lambdas. The dashed lines are the log values corresponding to the min (left dashed line) and 1 (right dashed line).

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

The best lambda is $\lambda = 0.2841768$.

```r
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")
```

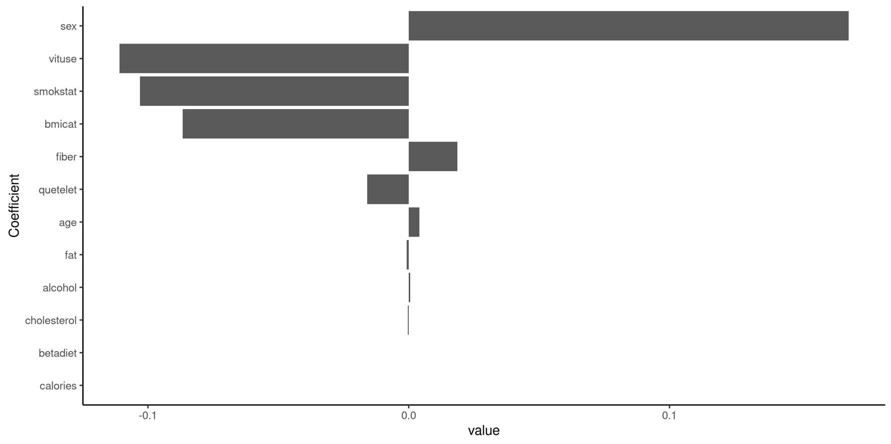


Figure 2.3: Coefficients for the ridge model under best lambda

```
# 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.

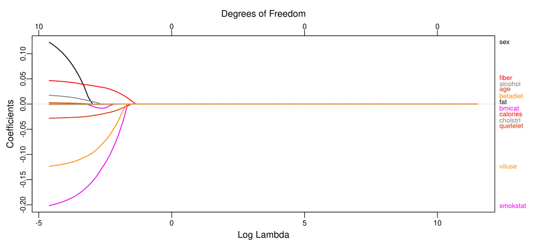
2.2 Lasso regression

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

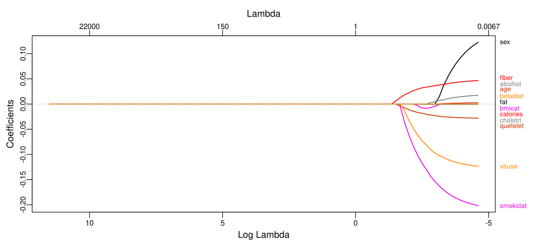
Figure 2.4 shows the coefficients against the penalty λ.

```
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 2.4: The Lasso regression coefficients

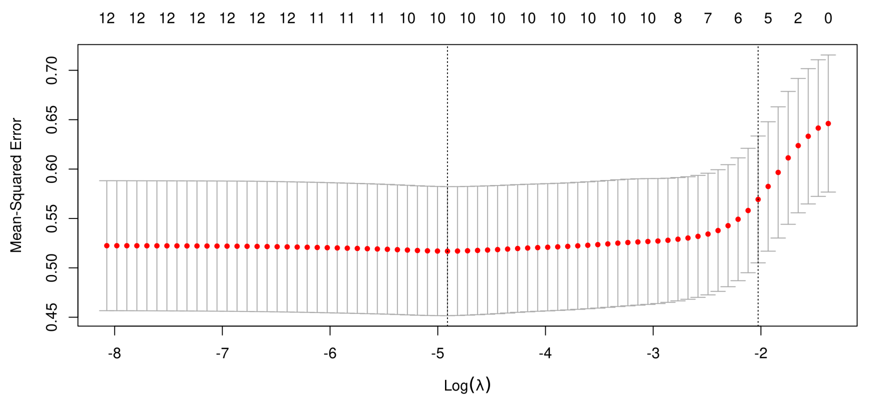


Figure 2.5: Cross-validated penalty for lasso model.

The Figure 2.5 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 for this choice of lambda or 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 turns out to be 0.4807286.

```
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=
```

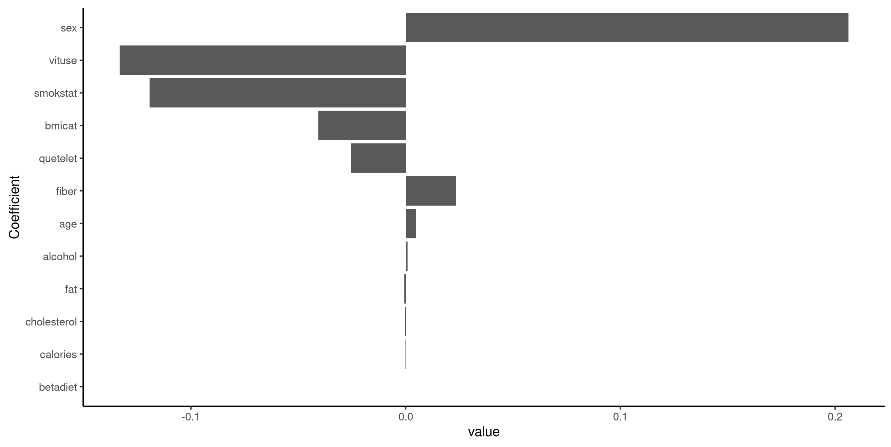


Figure 2.6: Coefficients for the lasso model under best lambda

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 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 λ. In the lasso regression we observed that the variables that influenced more in the model are the `sex`, `vituse`, `smokestat`, `bmicat` (Figure 2.6). To sum up, the lasso should perform better in a setting where a relatively small number of predictors have significant coefficients, and the remaining predictors have very small coefficients or that equal zero. In contrast, the ridge regression will perform better when the response is a function of many predictors, all with roughly equal-sized coefficients.

2.3 Principle component regression

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

Figure 2.7 shows the out-of-fold (cross-validated) MSE for each number of components in the PCR.

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

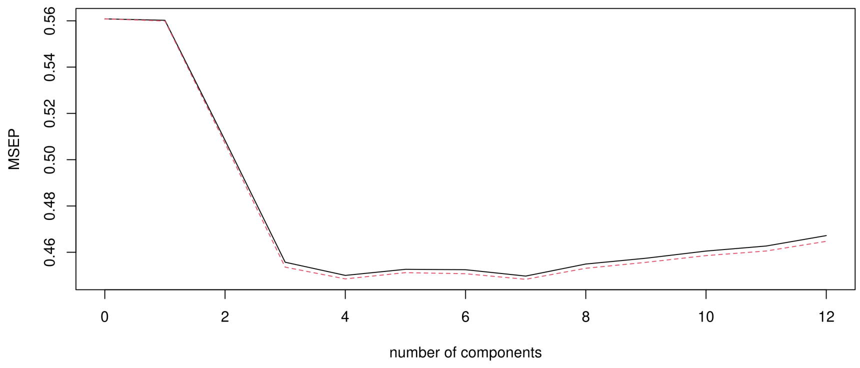


Figure 2.7: Cross-validated MSE for PCR model.

We definitely dont need more then 8 components. Let's perform CV on the training dataset and see if we can reduce the model even further.

```r
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")
```

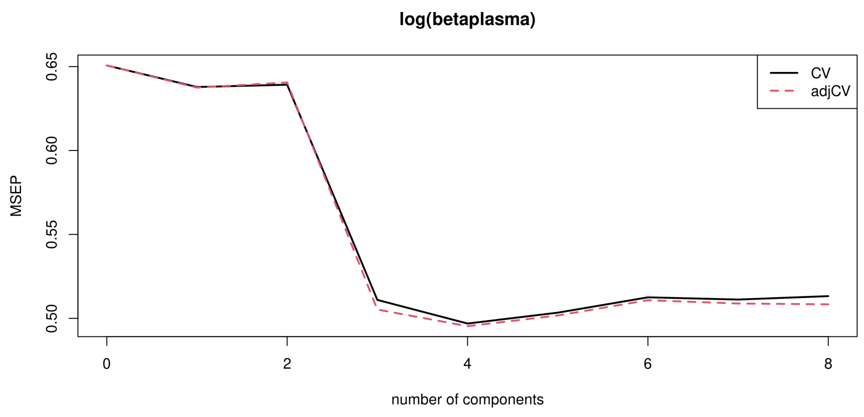


Figure 2.8: MSE for PCR model cross-validated on the training dataset.

We select the lowest number of components which minimizes MSE

```
pcr.pred <- predict(pcr.fit, x[test, ], ncomp = 3)
pcr.test.mse <- mean((pcr.pred - y.test)^2)
```

The test MSE for the PCR is 0.4074666.

3 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 3.1 shows the out-of-fold MSE for the models with different `mtry`.

```r
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
```

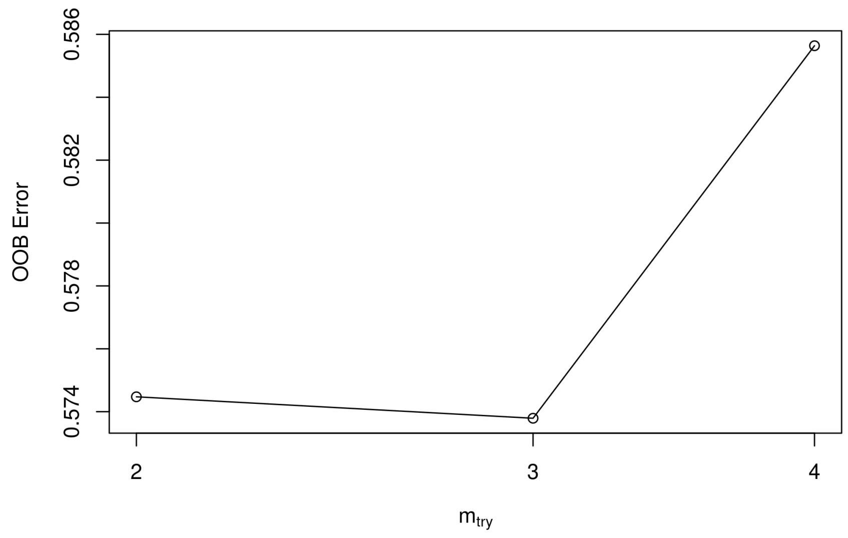


Figure 3.1: Cross-validated tuning of parameter mtry

The Figure 3.2 shows the predicted values against the true values for the test portion of our dataset.

```
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)
```

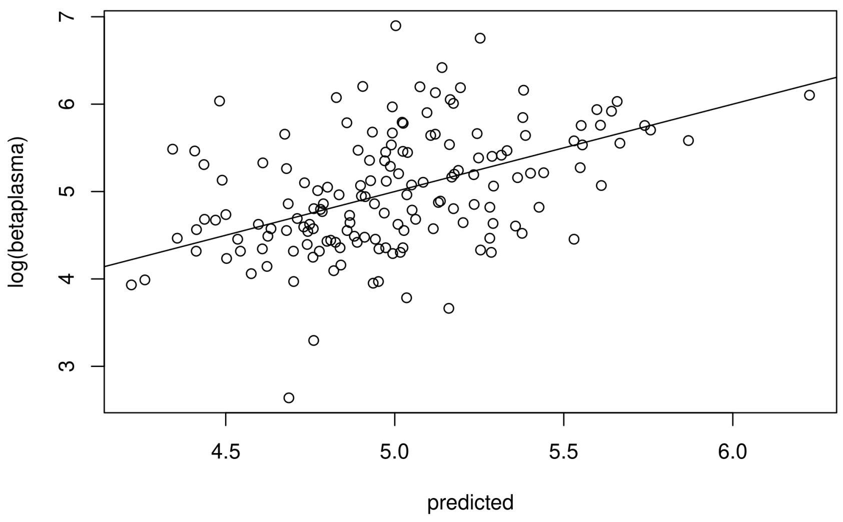


Figure 3.2: Predicted vs true values for random forest algorithm

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 3.3).

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

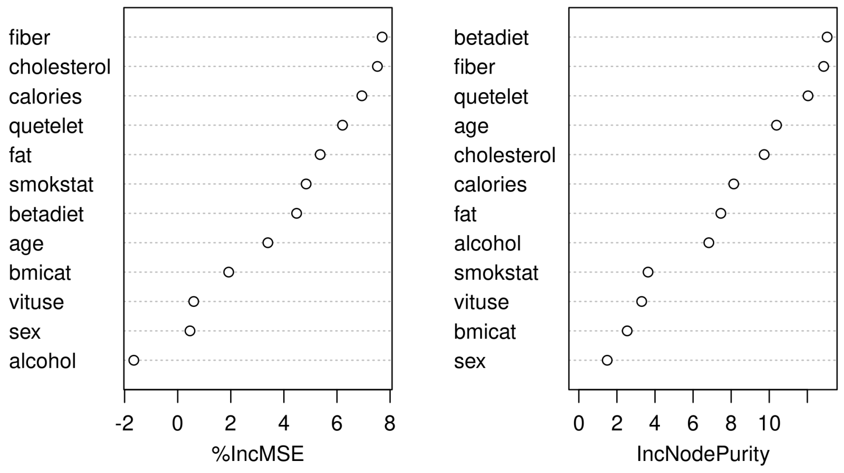


Figure 3.3: Variable importance plot for random forest with mtry 12

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

References

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