Practical Session 2 - Modèles de Régression Linéaire

Richard CHEAM & Menghor THUO

01-11-2023

Contents

IV. Cookies Study

- First, we read the cookies dataset as a dataframe and load into a variable called **tab**. We then separate the **Y** dependent variable (fat) and the **X** explanatory variables, where the number of observations, $n = 32$, and the predictors, $p = 700$.

```
tab <- read.csv(file = 'cookies.csv', header = TRUE, sep = ',')
Y <- tab$fat
X <- subset(tab, select = -fat)
```

- As it is a high dimensional dataframe, trying to visualize it will be a naive approach, hence, the length of correlation matrix will be taken into account instead.

```
corr_matrix <- cor(tab)
length(corr_matrix)
```

```
## [1] 491401
```

- A length of 491401 as it is a 701x701 matrix

```
threshold <- 0.9 # 0.9 for a really strong correlation between variables
corr_matrix <- corr_matrix[corr_matrix > threshold]
length(corr_matrix)
```

```
## [1] 374051
```

- The output above demonstrates that there are still 374051 values left in which they are greater than 0.9. To be more precise, it suggests a severe multicollinearity which the correlation between input variables ($x_{i=1,..,700}$) will pose a huge problem on variable selection methods such as forward, backward, or stepwise selection.
- To illustrate, by using **regsubsets**() function from library **leaps** to perform forward selection, it returns the warning stated "669 linear dependencies found". Whereas, the **step**() function will not even work.

```
library(leaps)
regForward <- regsubsets(fat~., tab, method = "forward")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 669 linear dependencies found
```

- Henceforth, we will use the alternative approache that will address the multicollinearity better which is the regularization technique, shrinkage methods.

(a) Ridge Regression

- Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where the independent variables are highly correlated which the estimate coefficients $\hat{\beta}$ of this model is given by :

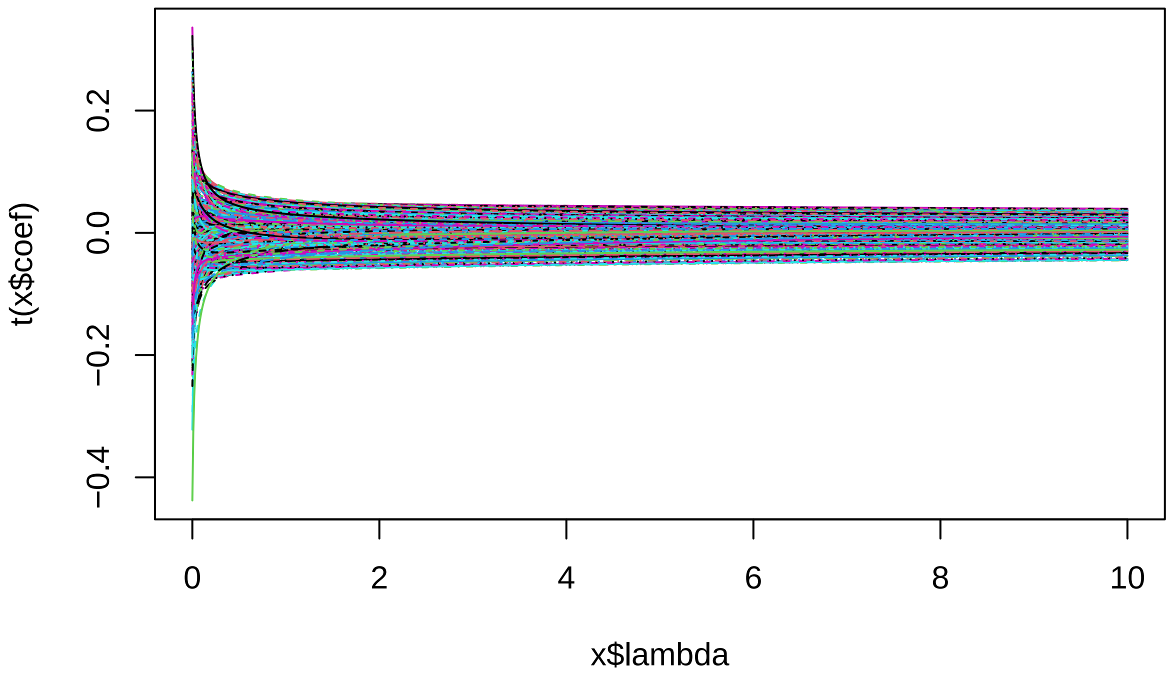
$$E(\beta, \lambda) = ||Y - X\beta||_2^2 + \lambda||\beta||_2^2$$
$$\Rightarrow \hat{\beta}_{RR} = (X^T X + \lambda I_p)^{-1} X^T Y$$

- $\hat{\beta}_{RR}$ minimizes the quadratic error term plus the l_2 shrinkage penalty term where $\lambda \geq 0$ is a tuning parameter, to be determined by using cross-validation in order to choose the best lambda value.
- Here we fit a linear model by ridge regression using **lm.ridge**() where λ is starting from 0 to 10 with an increment of 0.01.

```
ridge_mod <- lm.ridge(fat~., data = tab, lambda = seq(0, 10, 0.01))
```

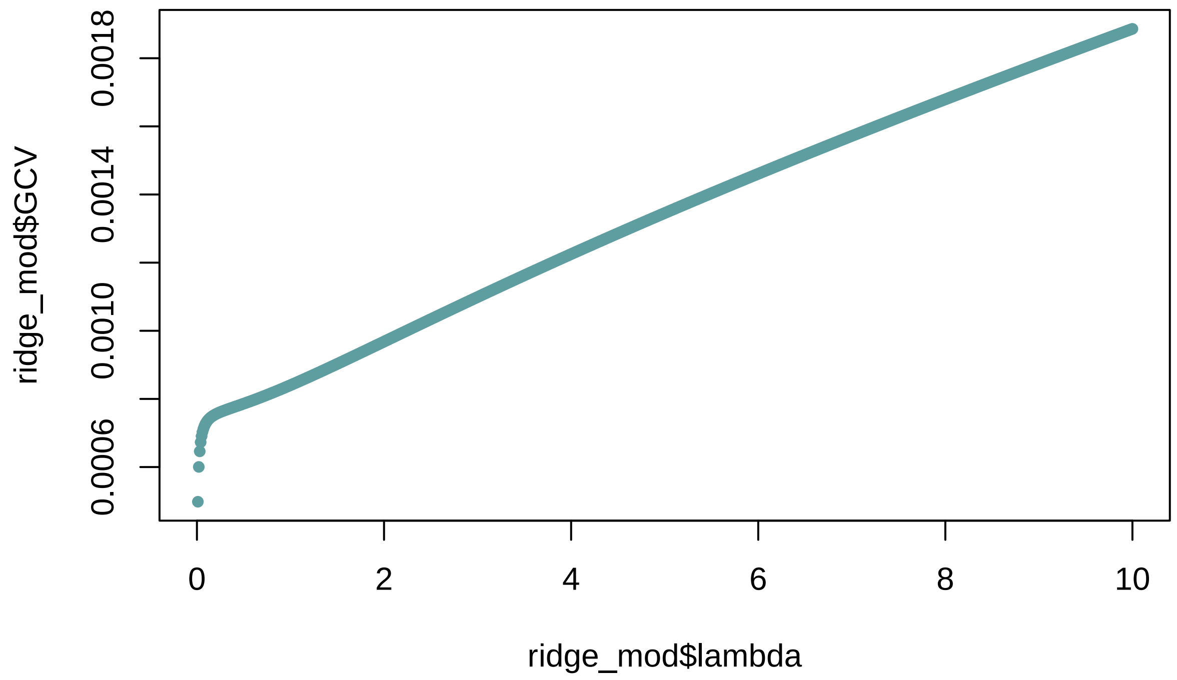
- The non-standardized ridge regression coefficients are displayed below for the cookies data set indicating its evolution given value λ

```
plot(ridge_mod)
```



- Graph below demonstrates the performances computed by cross-validation given the values of λ, where GCV stands for Generalized Cross Validation

```
plot(ridge_mod$lambda, ridge_mod$GCV, pch = 20, col = "cadetblue")
```



- We then use the Generalized Cross Validation (the smallest value) to choose the tuning parameter λ

```
best_lambda_ridge <- ridge_mod$lambda[which.min(ridge_mod$GCV)] #OR
#select(lm.ridge(tab$fat~., data = tab, lambda = seq(0, 10, 0.01)))
cat("Best lambda value:", best_lambda_ridge)
```

```
## Best lambda value: 0.01
```

- Then, we refit the model with the best lambda obtained
- Since the data is unscaled, hence, we use **coef**() to obtain the coefficients that are presented in their original, unscaled form, precisely, initial framework.

```
best_ridge <- lm.ridge(fat~., data = tab, lambda = best_lambda_ridge)
best_ridge_coef <- coef(best_ridge)
inter_ridge <- best_ridge_coef[1]
cat("Intercept:", inter_ridge)
```

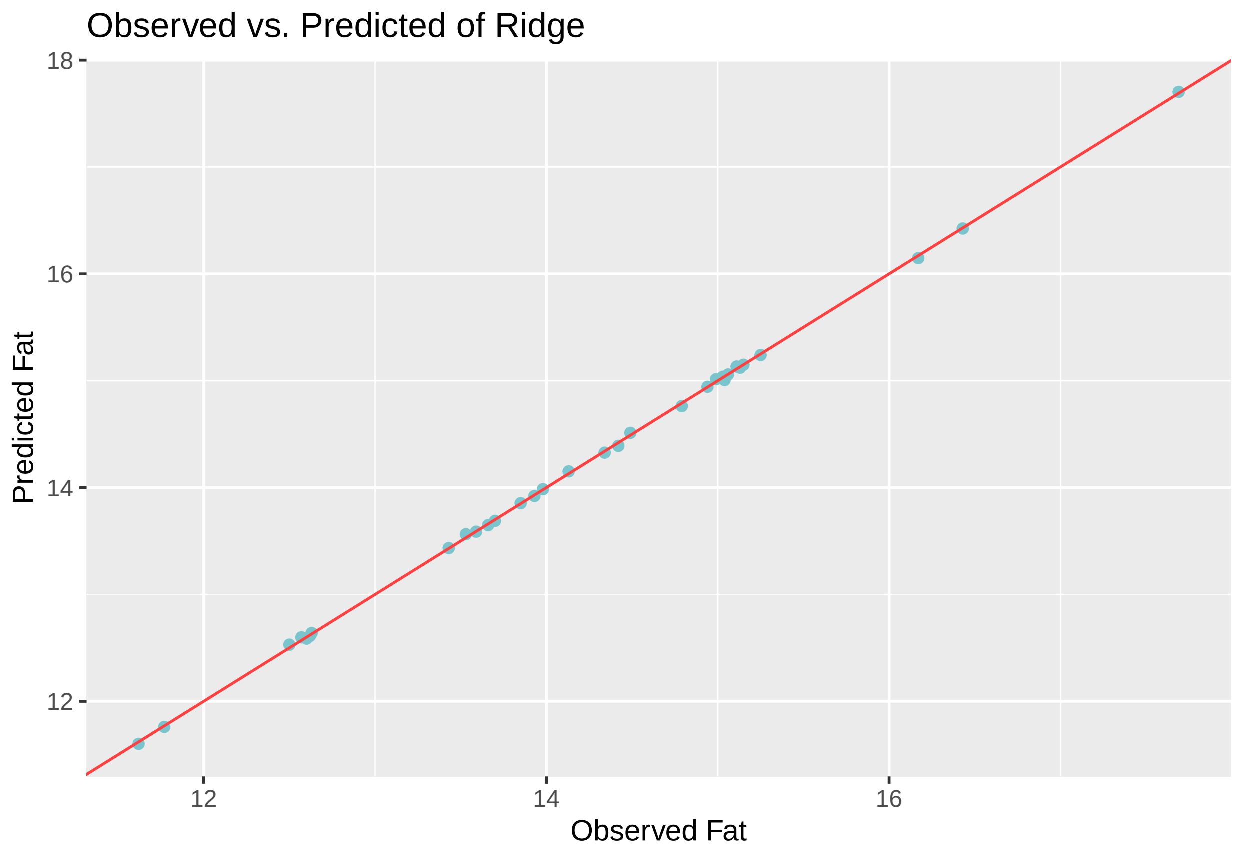
```
## Intercept: 15.98049
```

- As we do not penalize the intercept, then when finding $\hat{Y}$, intercept value is not included in the multiplication, and we will then add it to the predicted values afterward.

```r
Yridge <- (as.matrix(X) %*% as.vector(best_ridge_coef)[2:length(best_ridge_coef)])+inter_ridge
#OR cbind(matrix(1, nrow = nrow(tab), ncol = 1), as.matrix(X)) %*% as.vector(best_coef)
```

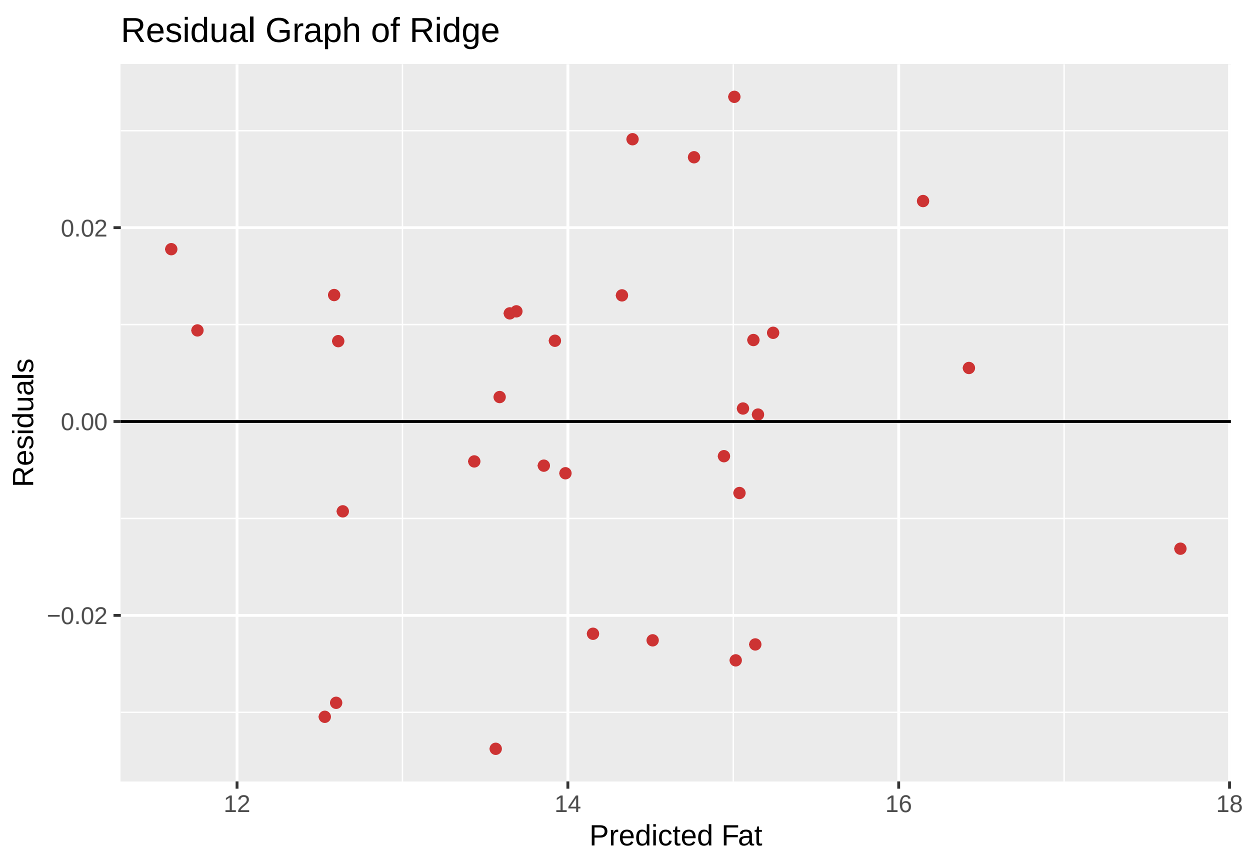
- Based on the graph below, scatter points are overlapped by the bisector line, indicating the very accurate prediction.

```r
ggplot(data.frame(Y, Yridge), aes(x = Y, y = Yridge)) +
geom_point(color = "cadetblue3") +
geom_abline(intercept = 0, slope = 1, color = "brown1") +
labs(title = "Observed vs. Predicted of Ridge", x = "Observed Fat",
y = "Predicted Fat")
```



- Whereas, The $(\hat{\epsilon}, \hat{y})$ plot below shows no discernible pattern. Thus, there is no information to be captured for such random distribution of residual values.

```r
ridge_residuals <- Y - Yridge
ggplot(data.frame(Yridge, ridge_residuals), aes(x = Yridge, y = ridge_residuals)) +
geom_point(color = "brown3") +
geom_hline(yintercept = 0) +
labs(title = "Residual Graph of Ridge", x = "Predicted Fat", y = "Residuals")
```



- We now calculate the mean squared error (MSE) of the best ridge regression model in order to measure the quality of this model, where the formula of the MSE denoted by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- Where Y_i is the observed values, and $\hat{Y}_i$ is the predicted values

```
#by using the MSE() function implemented above (do not include when knitting)
mse_ridge <- MSE(Y, Yridge)
cat("MSE:", mse_ridge)
```

```
## MSE: 0.0003120508
```

- MSE here is very small which indicates a good estimators quality of this mode. To be more precise, the difference between the observed values Y, and the predicted values $\hat{Y}$ is merely different. Hence, this is a good model and we will see if the Lasso has a smaller MSE.

(b) The Lasso

- Lasso is effective for variable selection by setting some coefficients to exactly zero, as for Ridge, it only tries to keep all coefficients relatively small. We now see whether the lasso can yield either a more accurate or a more interpretable model than ridge regression. We have:

$$E(\beta,\lambda) = ||Y - X\beta||_2^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

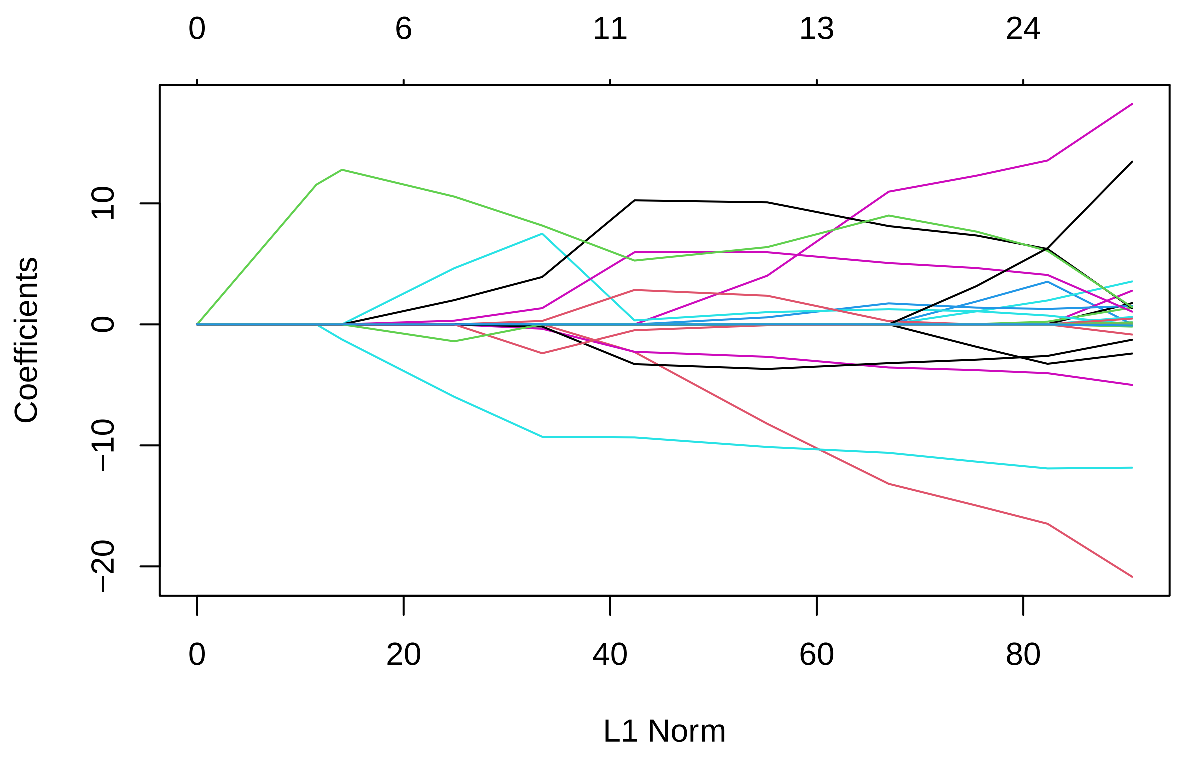
$$\Rightarrow \hat{\beta}_{Lasso} = |\hat{\beta}_{MCO}| - \frac{\lambda}{2}$$

- $\hat{\beta}_{Lasso}$ minimizes the quadratic error term plus the l_1 shrinkage penalty term where $\lambda \geq 0$ is a tuning parameter, to be determined by using cross-validation in order to choose the best lambda value.
- Here we use **glmnet**() to fit the model, where $\alpha = 1$ indicates lasso, also we have chosen to implement the function over a grid of values ranging from $\lambda = 10^{10}$ to $\lambda = 10^{-2}$, which it essentially covers the full range of scenarios from the null model containing only the intercept, to the least squares fit. (An Introduction to Statistical Learning with Applications in R)

```
#An Introduction to Statistical Learning with Applications in R
grid <- 10^seq(10, -2, length = 100)
```

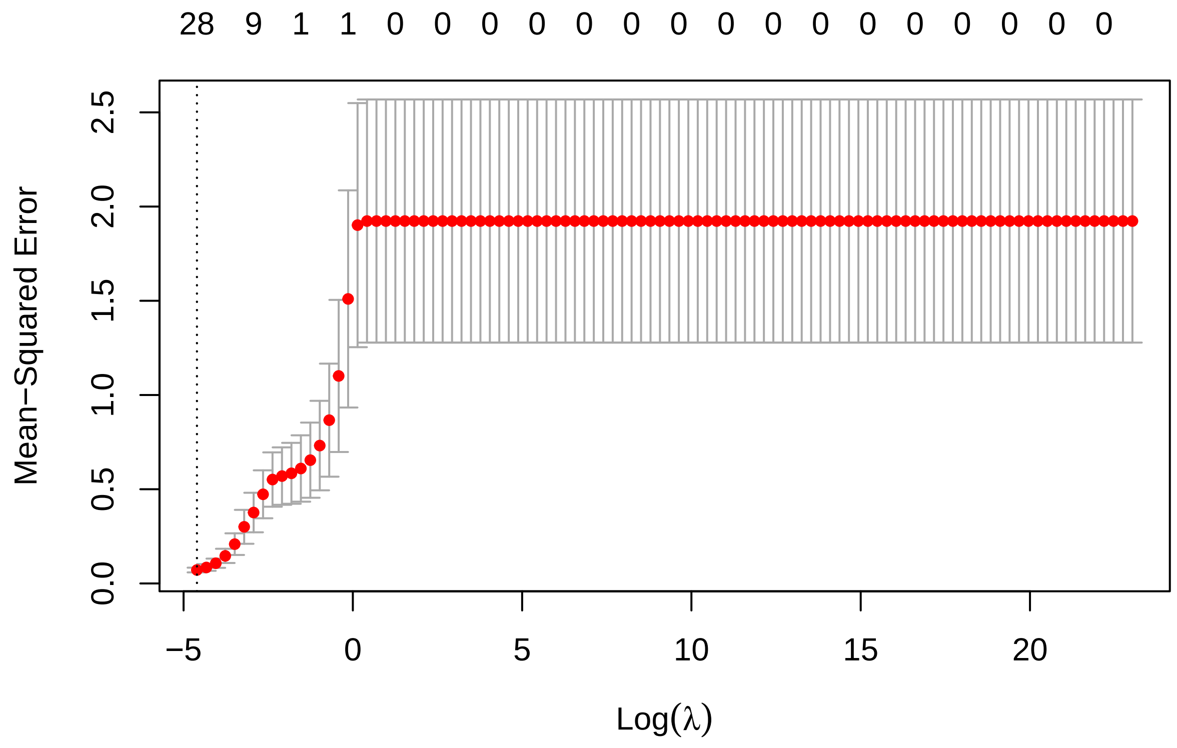
```
lasso_mod <- glmnet(as.matrix(X), Y, alpha = 1, lambda = grid)
plot(lasso_mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



- We can see from the coefficient plot that depending on the choice of tuning parameter, some of the coefficients will be exactly equal to zero. We now perform cross-validation by using **cv.glmnet**().

```
cv_out <- cv.glmnet(as.matrix(X), Y, alpha = 1, lambda = grid)
plot(cv_out)
```



- We now find the best lambda from the cross-validation

```
best_lambda_lasso <- cv_out$lambda.min
cat("Best lambda for lasso:", best_lambda_lasso)
```

```
## Best lambda for lasso: 0.01
```

- The lasso has a substantial advantage over ridge regression in that the resulting coefficient estimates are sparse. Here we see that only 23 coefficient estimates (without intercept) are different from zero. So the lasso model with λ chosen by cross-validation contains only 23 variables.

```
best_lasso <- glmnet(as.matrix(X), Y, alpha = 1, lambda = best_lambda_lasso)
best_lasso_coef <- as.numeric(coef(best_lasso)) #set as numeric values
best_lasso_coef <- best_lasso_coef[best_lasso_coef != 0] #see how many variables were selected
inter_lasso <- best_lasso_coef[1]
lasso_variables_remain <- rownames(coef(best_lasso))[which(coef(best_lasso)[, "s0"] != 0)]
```

- The variables selected are:

```
lasso_variables_remain
```

```
##  [1] "(Intercept)" "X1330"       "X1332"       "X1404"       "X1406"
##  [6] "X1410"       "X1720"       "X1722"       "X1882"       "X1884"
## [11] "X1886"       "X1888"       "X1890"       "X1978"       "X1980"
## [16] "X1982"       "X1984"       "X1986"       "X1988"       "X1990"
## [21] "X2072"       "X2074"       "X2076"       "X2302"
```

```
cat("Number of coefficients remaining (without intercept):", length(best_lasso_coef) - 1, "\n")
```

```
## Number of coefficients remaining (without intercept): 23
```

```
cat("Intercept:", inter_lasso)
```

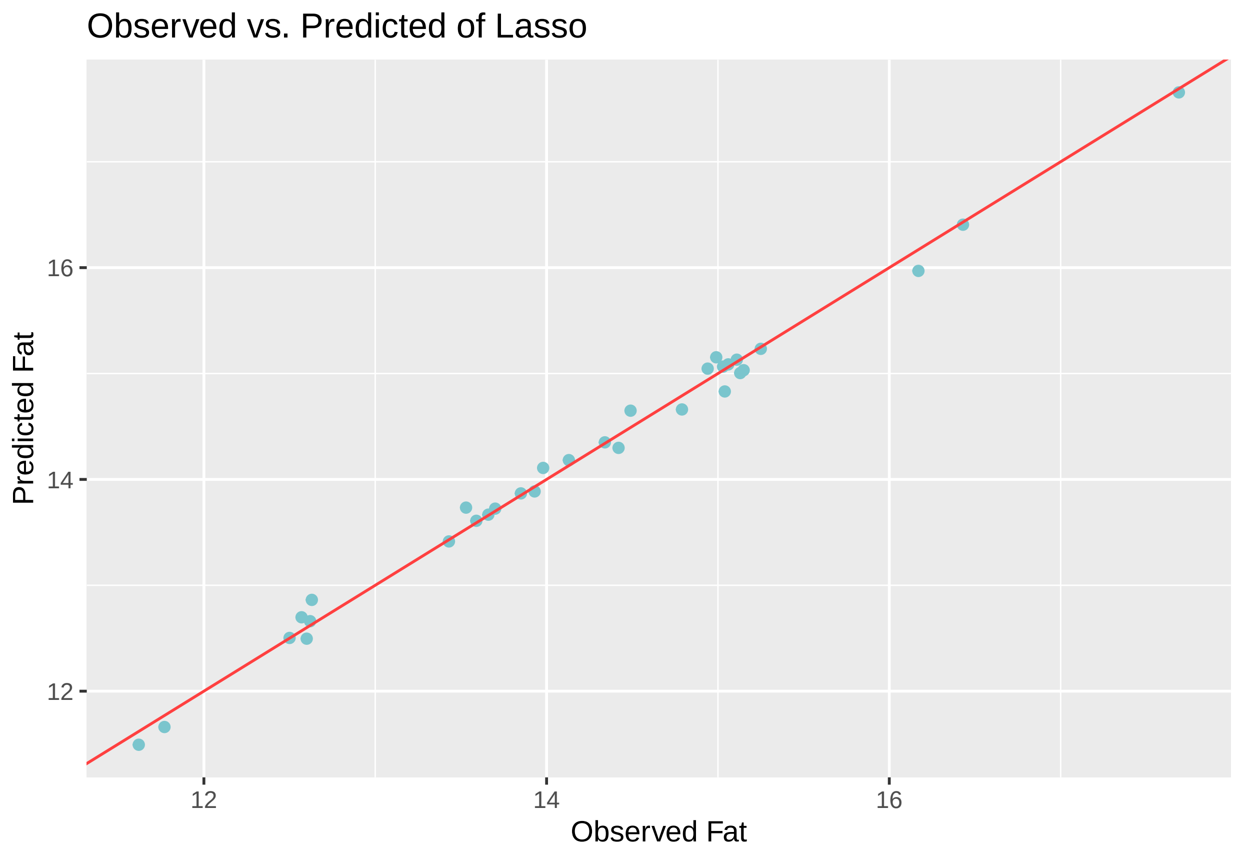
```
## Intercept: 15.41051
```

- We now find $\hat{Y}_{lasso}$ by using the best λ value obtained above

```
#use.Grame = FALSE for large p = 700
#lars() input args are both matrix
reslasso <- lars(as.matrix(X), as.matrix(Y) ,type="lasso", use.Gram = FALSE)
Ylasso <- predict.lars(reslasso, as.matrix(X), type="fit" , mode="lambda", s=best_lambda_lasso)
Ylasso <- Ylasso$fit
# OR Ylasso <- predict(best_lasso, s = best_lambda_lasso, newx = as.matrix(X))
```

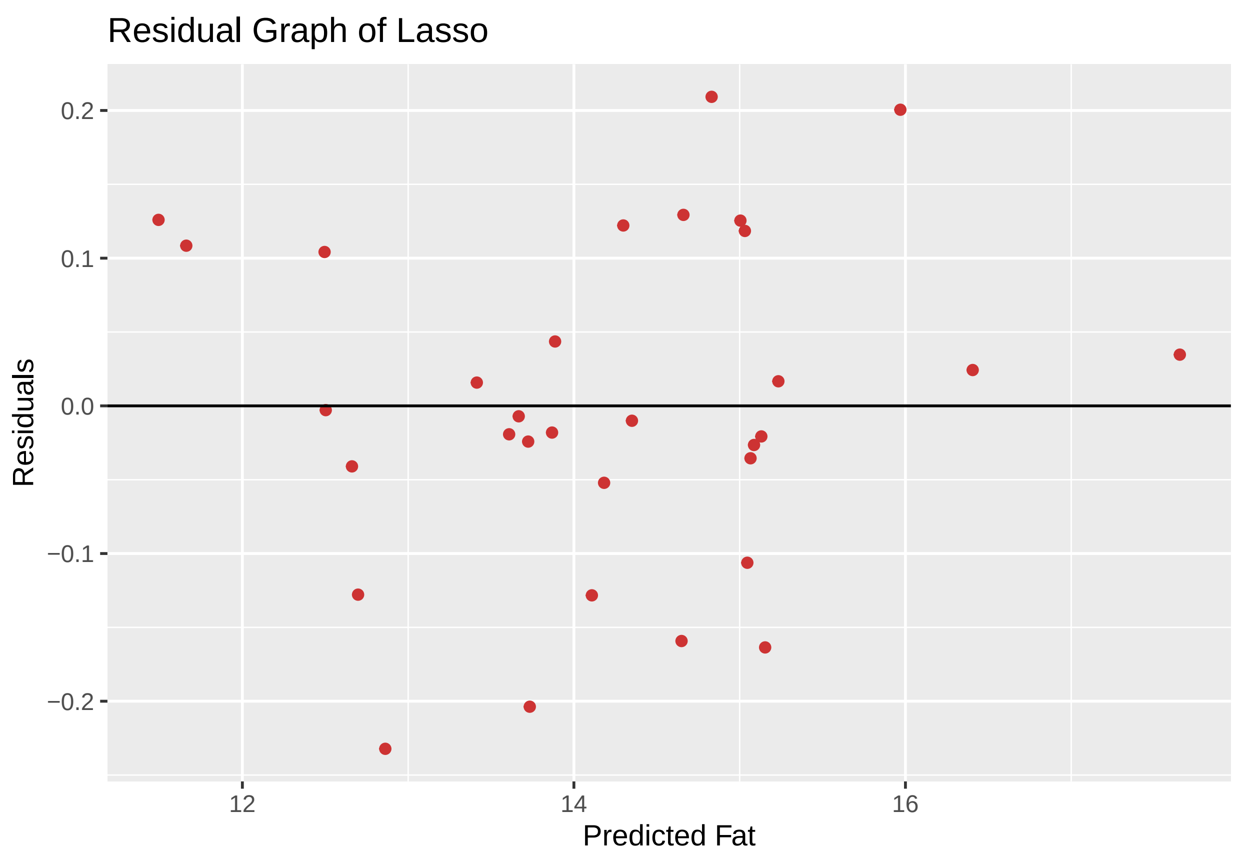
- According to graph below, we can see that most of the scatter points are overlapped by the bisector line, indicating the accurate prediction.

```
ggplot(data.frame(Y, Ylasso), aes(x = Y, y = Ylasso)) +
geom_point(color = "cadetblue3") +
geom_abline(intercept = 0, slope = 1, color = "brown1") +
labs(title = "Observed vs. Predicted of Lasso", x = "Observed Fat",
y = "Predicted Fat")
```



- The $(\hat{\epsilon}, \hat{y})$ plot below also shows no discernible pattern. Thus, there is no information to be captured for such random distribution of residual values.

```
lasso_residual <- Y - Ylasso
ggplot(data.frame(Ylasso, lasso_residual), aes(x = Ylasso, y = lasso_residual)) +
geom_point(color = "brown3") +
geom_hline(yintercept = 0) +
labs(title = "Residual Graph of Lasso", x = "Predicted Fat", y = "Residuals")
```



- Identically, we find the MSE for this model:

```
mse_lasso <- MSE(Y, Ylasso)
cat("Lasso's MSE: ", mse_lasso)
```

```
## Lasso's MSE:  0.01212286
```

- Here once again the MSE is small indicating an accuracy prediction.

(c) Conclusion

- Since R^2 increases mechanically with large number of variables, then only MSE is being considered as a main factor on measuring whether it is a good model.

```
cat("Ridge's MSE", mse_ridge)
```

```
## Ridge's MSE 0.0003120508
```

```
cat("Lasso's MSE", mse_lasso)
```

```
## Lasso's MSE 0.01212286
```

- Plus, the number of observations is quite low, 32; thus, it is not a good approach to split them into test and train data to see whether Ridge or Lasso has a better predictive power on a new dataset.
- In the final analysis, unlike Ridge Regression, The Lasso performs variable selection, and hence results in models that are easier to interpret. However, visibly, the Ridge has smaller error, meaning, in the case of cookies dataset here, Ridge is a better model than Lasso.