Team 8 Final Project

Main Findings

	LINEAR	LOGISTIC
PLS	✧ PCA1: First 5 components can explain up to 98% response. ✧ #7948: First 9 components can explain up to 98% response.	✧ Extract the major components (5 for Z & 9 for Y) in linear scenarios to do the logistic regression. ✧ The coefficients remains approximately the same trend but of different scale.
LASSO	✧ Use LASSO to reduce numbers of predictors from 600 to: ● PCA1,p: 30 ● PCA1,b: 38 ● #7948,p: 59 ● #7948,b: 25	✧ Logistic regression reduces number of predictors further. ✧ Common predictors have same sign. ✧ Many common predictors.
STEPWISE	✧ Use BIC criterion and stepwise selection to find out best model : ● PCA1,p: stop at the 20th step. ● PCA1,b: stop at 44th step. ● #7948,p: stop at 12th steps ● #7948,b: stop at 17th steps ✧ Remove 2 outliers: MEGA2_4, MEGA_2_5 for #7948,p	✧ PCA1: coefficients of both p and b are very different with original ones; Low significance. ✧ #7948: coefficients of both p and b are closer with original ones; High significance (some)
GMC	✧ For GMC variable selection, we use GMC as the criterion and selected n/log(n)≈40 variables which best explain the response. After grid searching for the best hyper parameter and optimization of the objective function, we get the best lambda, the maximal value of the function, as well as the corresponding coefficients betas. ✧ We found that the value of λ_1 will not affect RSS results. And result tables are:	

● Maximize first formula:

Maximal	#7948	PCA1
Set p	0.9032722 $k = 5, \lambda_2 = 0.01$	0.9263179 $k = 5, \lambda_2 = 0.01$
Set b	0.5501994 $k = 4, \lambda_2 = 0.1$	0.9111035 $k = 5, \lambda_2 = 0.01$

● Maximize second formula:

Maximal	#7948	PCA1
Set p	0.9032722 $k = 5, \lambda_2 = 0.01$	0.9263179 $k = 5, \lambda_2 = 0.01$
Set b	0.8838306 $k = 2, \lambda_2 = 0.01$	0.9111035 $k = 5, \lambda_2 = 0.01$

Data Preparation

Extract and Processing Data

We extract the data and organize them into dataframe from the .txt file. And after deleting repeated or non-existence genes among all 37 genes in Figure 5, we have a subset of 34 genes with 211 cells.

Find out Two Response Variables

Extract PCA1 From Subset: Z

We use princomp () function to execute Principle Component Decomposition and select the first component as our PCA1 for future processing.

Construct the Best Fit Model Among Subset of 34 Variables: Y

We use R-square as the major criteria of seleting the best model.

```
##The best model is to regress the 25th gene, X7948, with the rest genes. It c
omes up with the best R-square value 0.9020006.
```

Convert the Response Variables into Binary Format (for logistic regression)

We use smaller than 0 or larger than 0 to divide our response into 0 and 1.

Partial Least Square Regression

Linear regression part

We executed Partial Least Squares Regression on Y, Z and their corresponding predictors. Partial Least Squares is a shrinkage method select orthogonal components of predictors according to the descriptive ability towards response.

We use the pls package in R to do the decomposition as well as the regression and return the result shown in appendix (Table 1). We can see that for Z and Y response, first 5 and 9 components can explain up to 98% their information. Compared with the 597 and 599 genes, the volume of predictors has been extremely shrunk without significant information loss.

Partial Least Squares is a method similar with Principle Component Analysis, but we choose to do Partial Least Squares For two reasons. Theoretically speaking, as we still have to do a regression on latter procedures, we cannot extract major components regardless of responses. Partial Least Squares is exactly the one that considers the relationship between the predictors and response variables. Practically Speaking, as we have so much isolated genes, the explain percentage is heavily limited (see to the upper row of the Table 1) if we only do Principle Components Analysis. So generally, PLS is the better approach in variable extraction.

Then we do the Partial Least Squares Regression using the extracted variables.
We can see that from the figures (residual plots for Z, Y in Figure1, 2) regress on Z and Y with 98% explanation components (first 5 and first 9).

Logistic regression part

After this linear regression model, we do the dichotomization and set up a logistic regression model with the binary response and the original predictors. The detailed comparison of coefficients is plotted in the Figure 5, 6, 7, 8 in appendix. They shows similar trend but of different scale because dichotomization is another way of rescaling.

Check the residual plots of residuals when we fit the model with components that explained 98% variance of response. We can see that the residual is symmetrically distributed and shows no trend towards the fitted model. We can assume that the residual follows the Gauss-Markov Assumption.

Weakness

The residual is evenly distributed with both index and fitted value, which is one feature of Gauss-Markov Assumption. But, in residual-fitted value plot, the residual is equal on both side of 0 but asymmetrically distributed. It's against the assumption. We tried to do log transformation but as Y and Z are not always positive, it doesn't work at all. We haven't come up with a valid remedy for this problem.

Lasso Regression

Linear regression part

Step of doing lasso regression is as following:

- Generate design matrices for 2 datasets
- Fit two lasso models for different datasets using different lambdas
- Plot out the change of coefficients of each predictor when lambda change
- Use cv to find out the lambda minimize the RSS+penalty
- Fit model using best lambda and predict response
- Plot out residuals

The reason why we use Lasso instead of ridge is because that the penalty in Lasso can force the insignificant coefficient to be zero. Since we've got more predictors than observations, we need to focus on avoiding overfitting. Thus, we'll first use cross validation to choose tuning parameter and then fit a Lasso regression model to reduce number of variables.

From the plots of cross validation (Figure 9, 10), we can see the model minimize mean-squared error corresponds to really small lambda. Here we use lambda 1se provided by function "cv.glmnet"as our tuning parameter. The criterion this function use is actually cross validation, which prevent overfitting effectively. After fitting the model using chosen tuning parameter, we can see the number of predictors are reduced significantly. Using the predict function we can check the residual plots, which show that models fit quite well.(Figure 11)

	Dataset b	**Dataset p**
Response Z	`Lambda=0.109` `#of predictors: 38`	`Lambda=0.12` # of predictors: 30
Response Y	`Lambda=0.089` `#of predictors: 25`	`Lambda=0.03` `#of predictors: 59`

Logistic Regression part

Here we'll use the raw data (full model of predictors) to fit lasso model for two responses. After fitting the new models using similar methods in continuous response case, we can find out that doing lasso regression for binary response will reduce the number of predictors as reported. There are also many predictors in common for models which using the same dataset for both binary and constant type of response.

- ✧ **Number of common predictors**

- For response PCA1, the number of predictors change from 38 to 13 for dataset b and 30 to 28 for dataset p if we change PCA1 into a binary variable. Lasso and logistic regression has 8 predictors in common for dataset b and 7 predictors in common for dataset p.
- For response #7948, the number of predictors change from 25 to 14 for dataset b and 59 to 23 for dataset p if we change PCA1 into a binary variable. For #7948, lasso and logistic regression has 9 predictors in common for dataset b and 14 predictors in common for dataset p.

✧ **Same sign for common predictors**

- Except for the intercept, all the coefficients for common predictors in 2 datasets have same sign. This means changing response into binary will not affect common predictors' influence on them (from the aspect of sign). Also, change the response into binary can reduce the number of predictors to some extent.

Weakness

The main weakness of lasso regression is that the numbers of predictors are still too large. From the result of stepwise we can see that linear regression model of 14 predictors already provide quite good result (high r square and significance of coefficients).

stepwise selection

Linear regression part

In this part, we attempt to fit is model selection using stepwise methods. In this way, we can solve the problem of multicollinearity and get a simpler model. To obtain a simpler model, since BIC has a higher penalize, we choose BIC as criterion. For some cases, when we choose “both” as the direction, the procedure wouldn't stop even though the warning of "It's meaningless to select variables for a perfect fitted model" has been shown. And if we choose “backward” as the direction, the BIC is negative infinity for the full model. We believe this kind of error happens because p is larger than n.

So we limit the steps under 50 and choose “both” as the direction. We draw a plot of adjusted R square and BIC. By considering the significance of selected variables and the adjusted R2, we can decide where to stop. (Figure 12)

- For PCA1 response variable, when using set p, we choose to stop at 20 steps. The adjusted R2 is 0.9831 and all the variables are very significant. What’s more, the BIC value doesn’t change too much at step 20. When using set b, BIC did reach the lowest at 44 steps, so we choose to stop at 44 steps. The adjusted R2 is 0.9896 and all the variables are very significant.

- For response variable #7948, when using set p, we choose to stop at 12 steps However, the diagnostic plot shows two outliers: MEGA2_4, MEGA2_5. After removing the outliers, The adjusted R2 is 0.9421 and all the variables are very significant. When using set b, we choose to stop at 17 steps. The adjusted R2 is 0.932 and all the variables are very significant. What’s more, the BIC value doesn’t change too much at step 17.

Logistic regression part

To generate a logistic regression, we used the variables selected in linear case.

- For PCA1, in both data set b and p, all variables are not significant with coefficients very different with the original coefficients (original range[-1,1], new range[-100,100]).
- Response #7948, compared with PCA1, its coefficients are much closer with some significant coefficients (original range[-1,1], new range[-4,4]). This will be shown more clearly in the figure contained in appendix. (Figure 13)

Weakness

The ways we used to choose best model depends on personal decision very much. It's very hard to find a balance between BIC, adjusted R square and significance. As for logistic parts, the coefficients are very different.

GMC

We explain the GMC variable selection separately from the model comparison.

- For Part 1, we have to fit

$$y = g(x_1\beta_1 + \cdots + x_p\beta_p) + e$$

  , with

$$g(x_1\beta_1 + \cdots + x_p\beta_p) = poly(x_1\beta_1 + \cdots + x_p\beta_p, k)$$

  which means the k degree polynomial of the linear combination $x_1\beta_1 + \cdots + x_p\beta_p$.
  And then maximize

$$\frac{Var(g(x))}{Var(g(x)) + Var(e)} - \lambda_1 |corr(g(x), e)| - \lambda_2 \sum_{i=1}^{p} |\beta_i| \qquad (*)$$

  There are so many variables, parameters and hyperparameters to consider in both fitting and maximization, so we split the whole task into 3 steps:
  - First, we select reasonable variables with the integrated and professional R package to calculate the GMC value. As introduced in lecture, GMC($y|x_i$) reports the level of how x_i explains y, even though there's no linear relationship between x and y. It's a perfect tool for us to measure the usefulness of predictors. We calculated all the GMC($y|x_i$), $\mathrm{i} = 1, \ldots, \mathrm{n}$, $\mathrm{n} =$ feature dimension set p or b, $\mathrm{y} =$ response Y or Z . After ranking them from high to low, the first $\frac{\mathrm{n}}{\log(\mathrm{n})} \cong 40$ variables are selected to give explanation to the response. The specific variable names see to appendix.
  - Then, for any fixed k, λ_1, λ_2, we put the following in a integrated `R function`:
    - generate the polynomial design matrix $g(x)$ of the linear combination of the selected predictors with temporarily unidentified $\beta_1, \ldots, \beta_p$, using function `poly()`
    - fit the design matrix towards y using regular linear model
    - the error term e automatically come up after the model is built up.
  - To specify the best hyperparameters k, λ_1, λ_2, we use k-fold cross validation with 5 folds. The criterion of choosing tuning hyperparameters is using grid searching with different k, λ_1, λ_2 values and fill the corresponding RSS of the grid. The k, λ_1, λ_2 values that offers the best RSS is the final hyperparameters we choose in the final model. They may vary from training set to training set.

Here comes a very interesting result: with fixed k, λ_2, no matter what value λ_1 takes, the RSS results remain the same. We investigate the penalty term that λ_1 punishes: $|corr(g(x), e)|$, and it returns `5.38226e-17`, which indicates that the model `y ~ g(x)` has been fully fitted with the error term e completely orthogonal with the polynomial term $g(x)$. And the fact that $|corr(g(x), e)|$ approximately equals zero also gives sensible reason toward the insignificance of the value of λ_1. We assume this is the reason why we have only one λ to be tuned in the second part of the question.

Then the optimization function $(*)$ is all determined except for the best βs we have to search for. For each loop with different combination of k, λ_1, λ_2, We use function `optim()` to find out the best parameters $\beta_1, \dots, \beta_p$, and the corresponding maximal value of $(*)$. The values of hyperparameters are commented.

Maximal of $(*)$	Y	Z
Set p	0.9032722 $k = 5, \lambda_2 = 0.01$	0.9263179 $k = 5, \lambda_2 = 0.01$
Set b	0.5501994 $k = 4, \lambda_2 = 0.1$	0.9111035 $k = 5, \lambda_2 = 0.01$

The specified values of $\beta_1, \dots, \beta_p$ see to appendix.

- For Part 2, the steps are similar. We maximize the function

$$\mathrm{GMC}(y|g(x)) - \lambda \sum_{i=1}^{p} |\beta_i| \qquad (**)$$

and do cross validation on each candidate value on k, λ. Then `optim()` in each grid search combination of hyperparameter. The result is listed below:

Maximal of $(**)$	Y	Z
Set p	0.9032722 $k = 5, \lambda_2 = 0.01$	0.9263179 $k = 5, \lambda_2 = 0.01$
Set b	0.8838306 $k = 2, \lambda_2 = 0.01$	0.9111035 $k = 5, \lambda_2 = 0.01$

The specified values of $\beta_1, \dots, \beta_p$ see to appendix.

Appendix

PLS part, linear models

"Z ~ Predictors(p)" % variance explained

	1 comps	2 comps	3 comps	4 comps	5 comps
X	11.47	22.74	34.58	50.05	53.85
Z	90.85	94.00	96.14	97.48	98.28

"Z ~ Predictors(b)" % variance explained

	1 comps	2 comps	3 comps	4 comps	5 comps
X	12.02	26.12	46.80	49.53	52.33
Z	89.06	93.91	94.88	97.39	98.28

"Y ~ Predictors(p)" % variance explained

	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps	7 comps	8 comps	9 comps
X	10.78	20.68	31.24	49.20	56.73	58.93	61.43	63.96	65.18
Y	74.64	82.55	88.44	91.08	93.23	95.88	96.81	97.50	98.30

"Y ~ Predictors(b)" % variance explained

	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps	7 comps	8 comps	9 comps
X	11.10	23.87	43.36	49.76	55.36	58.28	60.22	63.29	64.81
Y	73.47	81.81	85.05	91.08	93.47	95.48	96.58	97.24	98.09

Table 1(press the label to return to the text)

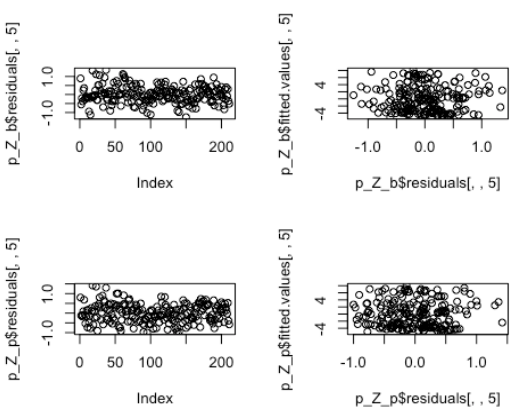


Figure 1

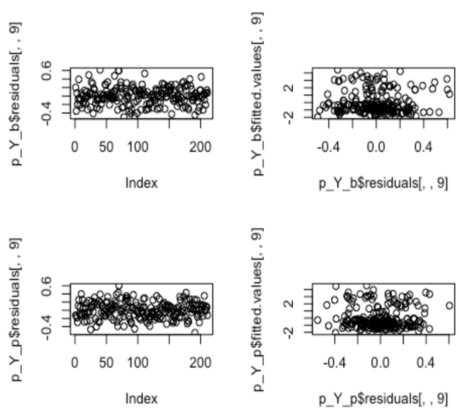


Figure 2

PLS part, logistic models

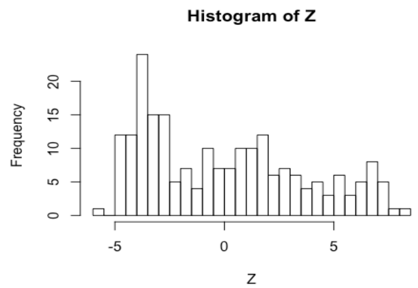


Figure 3

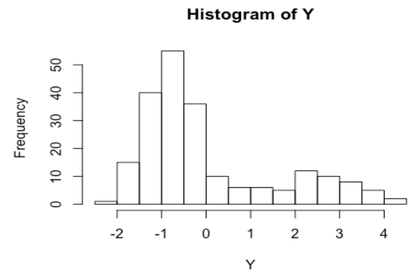


Figure 4

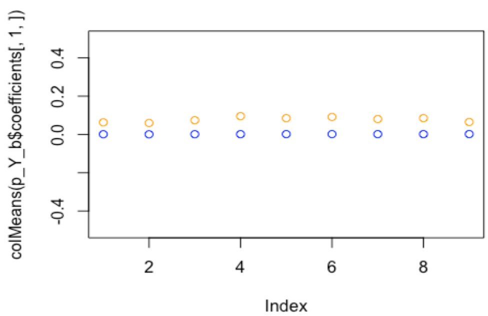


Figure 5

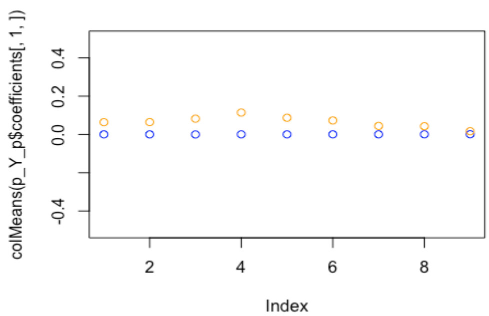


Figure 6

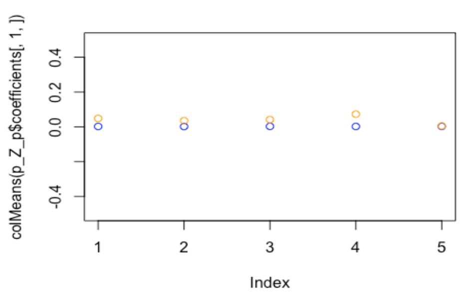


Figure 7

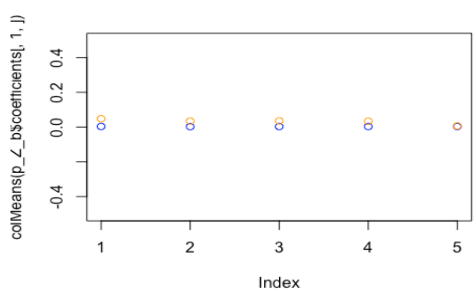


Figure 8

Lasso part

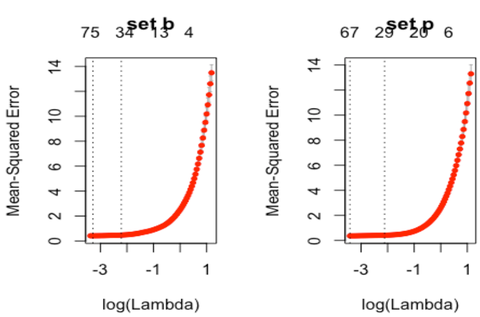


Figure 9 : Response Z

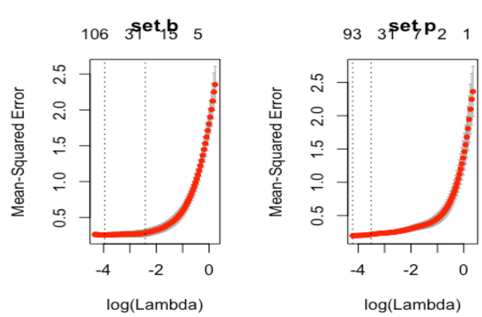
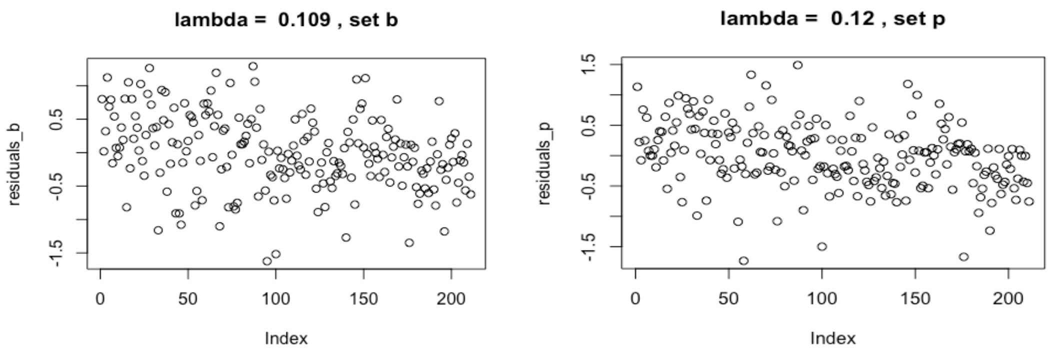


Figure 10: Response Y



Response Z

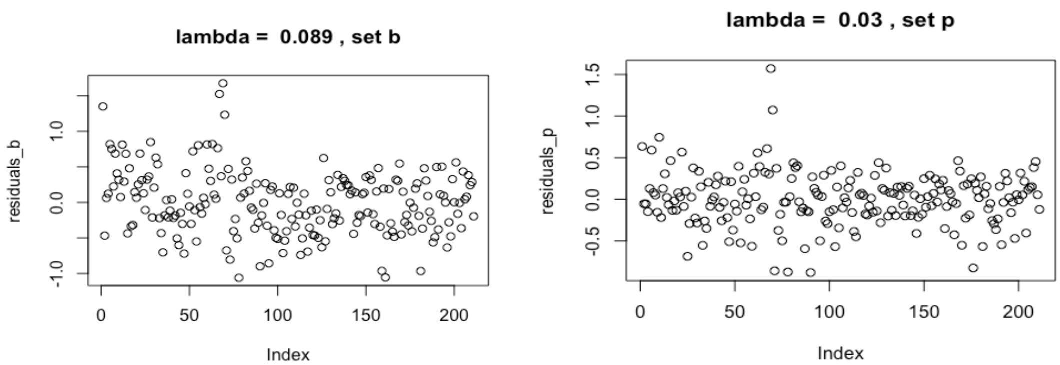


Figure 11:Response Y

Stepwise part, linear models

```
PCA1~set p

##(Intercept)      X8005      X7813       X8042       X8214       X8252
1.428757e-10  8.011449e-01  3.061296e-01 -1.966035e-01  1.561845e-01 -4.231188
e-01
   X8116         X8126      X8158       X8164       X7891       X8037
-1.554706e-01 -8.914259e-02  3.525173e-01 -2.837342e-01 -3.338751e-01  1.68670
2e-01
   X7780
 2.259363e-01

PCA1~set b

##    (Intercept)          X630          X736          X1004          X1140
## 2.537171e-10  4.124217e-01 -2.486528e-01  2.921273e-01  4.405964e-01
##          X879         X1165         X1154         X1150          X719
## 3.510490e-01 -1.191637e-01  3.708921e-01  4.043328e-01 -4.009163e-01
```

```
##          X725          X1062           X750          X1122          X1156
## 1.499412e-01  2.827686e-01  5.487230e-01 -3.171017e-01 -4.746130e-01
##          X913           X867          X1023           X724          X1008
## 2.121685e-01 -2.206994e-01  3.542756e-01  2.404564e-01 -2.971988e-01
##          X740           X905          X1151           X974          X1028
## 2.212294e-01 -3.243139e-01  3.869443e-01  2.311873e-01 -3.196056e-01
##          X998          X1060          X1191           X983           X708
## 2.940633e-01 -2.043588e-01 -2.346388e-01  1.684545e-01  3.246562e-01
##         X1172           X676
## 7.091478e-02 -2.728498e-01
```

```
#7948~set b
```

```
##  (Intercept)           X630          X1182           X740          X1106
## 4.321004e-10  4.694213e-01  2.756774e-01  5.318479e-01 -7.104435e-02
##         X1140           X751          X1004           X787           X742
## 1.891106e-01 -2.676646e-01  1.628844e-01  2.266107e-01  2.822262e-01
##          X841          X1120          X1108           X750           X609
## 9.560649e-01  1.304785e-01 -3.294914e-01  2.285697e-01 -3.344508e-01
##          X940           X668          X1107
## 1.113261e-01  3.856523e-01 -7.264128e-01
```

```
#7948~set p
```

```
##  (Intercept)          X8005          X7813          X8042          X8214
## 1.428757e-10  8.011449e-01  3.061296e-01 -1.966035e-01  1.561845e-01
##         X8252          X8116          X8126          X8158          X8164
##-4.231188e-01 -1.554706e-01 -8.914259e-02  3.525173e-01 -2.837342e-01
##         X7891          X8037          X7780
##-3.338751e-01  1.686702e-01  2.259363e-01
```

```
#7948~set p outlier removed
```

```
## (Intercept)       X8005       X7813       X8042       X8214       X8252
## -0.01700308 0.73948827 0.34758018 -0.18566977 0.13461592 -0.36458051
##       X8116       X8126       X8158       X8164       X7891       X8037
##-0.15343510 -0.09104487 0.30484721 -0.28340087 -0.36323558 0.18791569
##       X7780
## 0.18076963
```

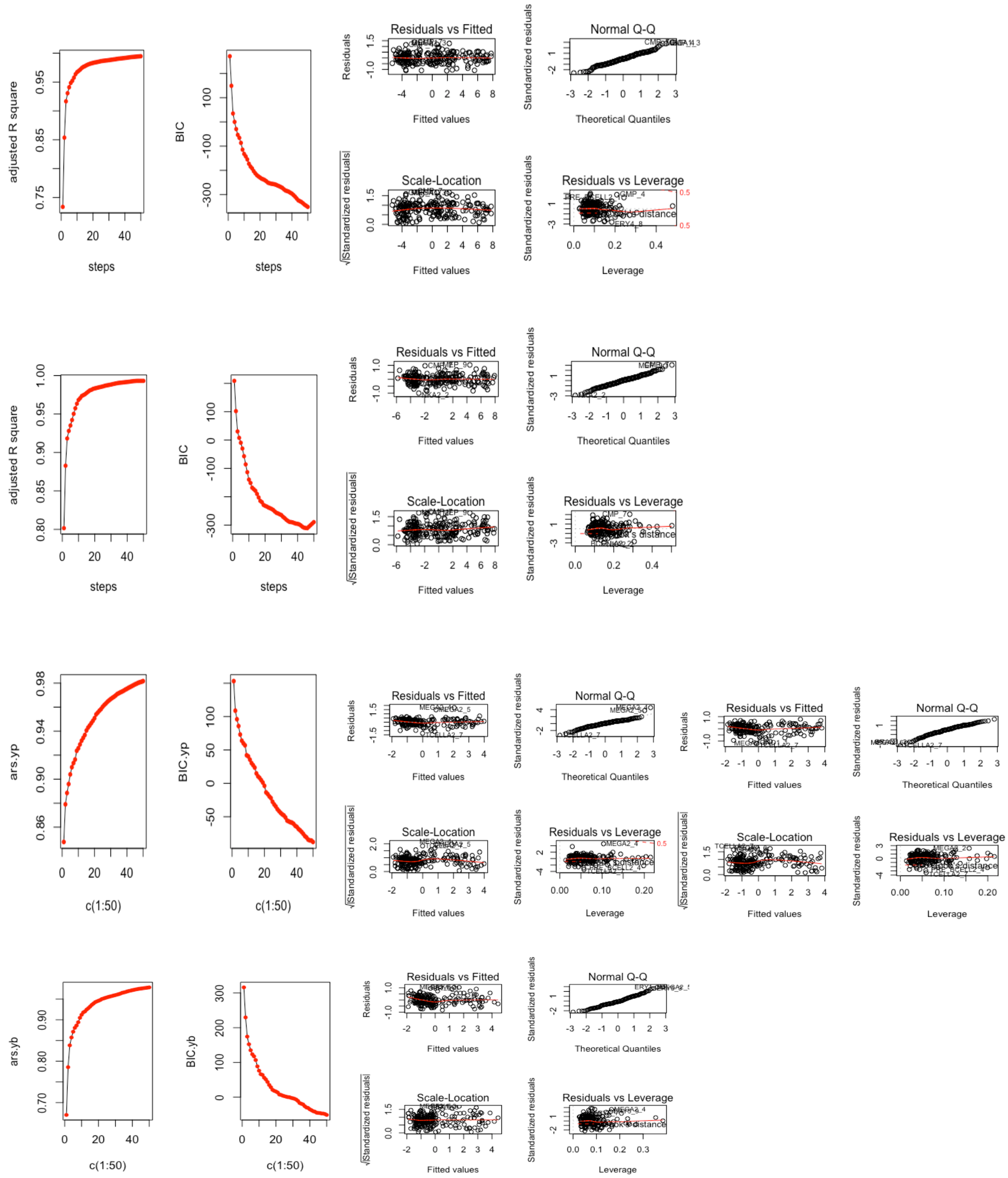

adjusted R square
steps
BIC
Residuals vs Fitted
Residuals
Fitted values
Normal Q-Q
Standardized residuals
Theoretical Quantiles
Scale-Location
√|Standardized residuals|
Residuals vs Leverage
Leverage
ars.yp
BIC.yp
c(1:50)
ars.yb
BIC.yb

Figure 12

Stepwise part, logistic models

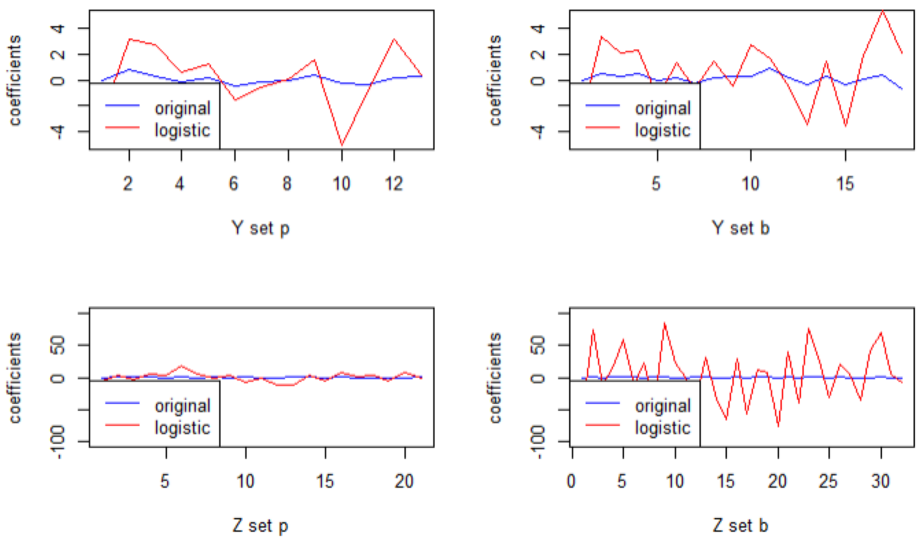


Figure 13

GMC part

```
> out_py$par
        x8005         x7813         x7995         x8009         x8323         x7871         x8304 
 3.359170e-01  4.742422e-02  2.505746e-01 -2.871190e-04 -7.935834e-03  5.143530e-02  2.835965e-01 
        x8149         x7860         x7918         x7909         x7807         x8033         x8339 
 8.473347e-03 -2.140105e-02  1.507227e-01  5.721075e-02 -1.196949e-02  1.046420e-01  4.048151e-01 
        x8274         x8335         x8047         x8141         x7869         x8114         x8174 
 1.124583e-01  2.070772e-02 -2.228519e-02 -2.012962e-02  4.300179e-02 -1.371591e-01 -6.341373e-02 
        x8001         x8334         x7888         x7865         x7952         x8186         x7775 
-2.428381e-03  6.309681e-02  3.137581e-01  3.488028e-03 -1.314340e-02 -8.058395e-02 -1.129000e-01 
        x8104         x7820         x8259         x8295         x8239         x7927         x8039 
 1.879246e-02  4.751939e-02  6.120995e-03 -2.979733e-02 -6.942221e-02 -3.148367e-02  1.580235e-02 
        x7997         x7938         x8020         x8062         x8362 
-1.737547e-01 -8.228882e-02  5.525119e-02  2.072825e-05  1.856799e-01 
> out_by$par
         x630          x946          x673         x1154          x874         x1117          x879 
 0.2634395713 -0.0904480129  0.0007833109 -0.0115214773  0.0783761776  0.3065875472  0.0049984808 
        x1140         x1024         x1182         x1139          x913          x650         x1177 
 0.2500924255 -0.0021010558  0.2396645090  0.2628124971 -0.0790347001 -0.1865846173 -0.2116881329 
         x736          x663          x714          x983          x907          x932         x1138 
 0.0217975900 -0.0107429618  0.2167499342  0.1695656491  0.0074426580  0.0745390537  0.1179032406 
         x824         x1062          x869          x629          x665         x1120          x653 
-0.1894206540 -0.1050786133  0.0111677207  0.1182499528 -0.0013463798  0.0192205946 -0.0457132516 
         x955          x940         x1103         x1034          x752         x1009          x989 
-0.0930555011  0.0269586211  0.0460201358  0.0034972316  0.1369496491 -0.0471866904  0.0864927535 
        x1133          x725          x858          x894          x945 
 0.0261708862  0.0423928703 -0.0273125231 -0.0214698572 -0.0446810996 
> out_pz$par
        x8009         x8005         x8335         x7995         x7813         x8323         x7807 
 0.0898821808  0.3465726070  0.0830033468  0.2328024088  0.3455616004 -0.0416046114  0.0071079659 
        x7888         x8245         x7909         x8275         x8353         x8149         x8322 
 0.0645857816 -0.0791824100  0.1116572184 -0.2475391649  0.2175562405  0.2137252341  0.3629479133 
        x8174         x7871         x8350         x8177         x7775         x7858         x8214 
-0.1671860535 -0.1466783601  0.1582962544 -0.1785522106  0.0309142966  0.1429546343  0.2367986667 
        x8304         x8339         x7860         x8117         x7953         x7904         x8278 
 0.2749540912  0.0371537360  0.0211573089  0.0069696768  0.0065753561  0.1697455031 -0.0275844903 
        x7917         x7825         x8114         x8033         x8274         x7918         x8017 
-0.1833667867  0.1574479364 -0.0136681282 -0.0321433053 -0.0423788866  0.0004044455  0.4020605592 
        x8239         x8063         x7869         x7824         x8047 
 0.0179380826  0.0187267345  0.0816164760 -0.0536786896  0.2565035766 
```

```
> out_bz$par
        x630        x1024         x932         x736         x874         x989         x946 
 0.549906322  0.342695172  0.009401298 -0.024132011  0.190572833  0.030660747 -0.016585611 
       x1140         x650         x879         x907         x891        x1121        x1117 
 0.285238160  0.053511427  0.210509597 -0.019637744 -0.075262859  0.046861612  0.435104491 
       x1084        x1154        x1017        x1020        x1139         x663         x673 
-0.127989923  0.134375389  0.019384616 -0.050778893 -0.160001983 -0.109037090  0.278466083 
        x629         x701        x1103         x824         x866        x1133         x913 
 0.096332845 -0.011028228 -0.002834551 -0.237975290  0.174797050 -0.013690114  0.013425314 
        x934         x752         x691         x876         x739         x887        x1157 
 0.554177139  0.044360319 -0.188254629  0.006470443  0.008685916  0.043838862 -0.152705407 
        x983         x864         x741         x850        x1034 
 0.215573756 -0.178425752  0.178251834  0.011798168  0.103077427 
> out_py$par
        x8005         x7813         x7995         x8009         x8323         x7871         x8304 
 3.359170e-01  4.742422e-02  2.505746e-01 -2.871190e-04 -7.935834e-03  5.143530e-02  2.835965e-01 
        x8149         x7860         x7918         x7909         x7807         x8033         x8339 
 8.473347e-03 -2.140105e-02  1.507227e-01  5.721075e-02 -1.196949e-02  1.046420e-01  4.048151e-01 
        x8274         x8335         x8047         x8141         x7869         x8114         x8174 
 1.124583e-01  2.070772e-02 -2.228519e-02 -2.012962e-02  4.300179e-02 -1.371591e-01 -6.341373e-02 
        x8001         x8334         x7888         x7865         x7952         x8186         x7775 
-2.428381e-03  6.309681e-02  3.137581e-01  3.488028e-03 -1.314340e-02 -8.058395e-02 -1.129000e-01 
        x8104         x7820         x8259         x8295         x8239         x7927         x8039 
 1.879246e-02  4.751939e-02  6.120995e-03 -2.979733e-02 -6.942221e-02 -3.148367e-02  1.580235e-02 
        x7997         x7938         x8020         x8062         x8362 
-1.737547e-01 -8.228882e-02  5.525119e-02  2.072825e-05  1.856799e-01 
> out_by$par
         x630          x946          x673         x1154          x874         x1117          x879 
 1.567791e-01 -4.687153e-02  9.109179e-02 -2.673263e-02  8.619870e-02  3.852413e-01  3.065474e-02 
        x1140         x1024         x1182         x1139          x913          x650         x1177 
 2.424047e-01 -2.196111e-02  2.103772e-01  8.547345e-02 -9.538970e-02 -9.931913e-02 -1.733928e-01 
         x736          x663          x714          x983          x907          x932         x1138 
 4.037205e-02 -6.799111e-02  1.924413e-01  2.836662e-01  3.097414e-02  5.546866e-02  1.294159e-01 
         x824         x1062          x869          x629          x665         x1120          x653 
-1.929874e-01 -7.837323e-02 -2.243330e-03  1.754583e-01  2.781755e-03  3.690097e-02 -3.876570e-02 
         x955          x940         x1103         x1034          x752         x1009          x989 
-9.063334e-02  6.424939e-02  3.892274e-02  9.187777e-05  1.341535e-01 -6.172521e-02  6.654758e-02 
        x1133          x725          x858          x894          x945 
 1.062932e-02  6.425529e-02 -5.231352e-02 -6.321462e-02 -1.633537e-01 
> out_pz$par
        x8009         x8005         x8335         x7995         x7813         x8323         x7807 
 0.0898821808  0.3465726070  0.0830033468  0.2328024088  0.3455616004 -0.0416046114  0.0071079659 
        x7888         x8245         x7909         x8275         x8353         x8149         x8322 
 0.0645857816 -0.0791824100  0.1116572184 -0.2475391649  0.2175562405  0.2137252341  0.3629479133 
        x8174         x7871         x8350         x8177         x7775         x7858         x8214 
-0.1671860535 -0.1466783601  0.1582962544 -0.1785522106  0.0309142966  0.1429546343  0.2367986667 
        x8304         x8339         x7860         x8117         x7953         x7904         x8278 
 0.2749540912  0.0371537360  0.0211573089  0.0069696768  0.0065753561  0.1697455031 -0.0275844903 
        x7917         x7825         x8114         x8033         x8274         x7918         x8017 
-0.1833667867  0.1574479364 -0.0136681282 -0.0321433053 -0.0423788866  0.0004044455  0.4020605592 
        x8239         x8063         x7869         x7824         x8047 
 0.0179380826  0.0187267345  0.0816164760 -0.0536786896  0.2565035766 
```

```
> out_bz$par
        x630        x1024         x932         x736         x874         x989         x946 
 0.549906322  0.342695172  0.009401298 -0.024132011  0.190572833  0.030660747 -0.016585611 
       x1140         x650         x879         x907         x891        x1121        x1117 
 0.285238160  0.053511427  0.210509597 -0.019637744 -0.075262859  0.046861612  0.435104491 
       x1084        x1154        x1017        x1020        x1139         x663         x673 
-0.127989923  0.134375389  0.019384616 -0.050778893 -0.160001983 -0.109037090  0.278466083 
        x629         x701        x1103         x824         x866        x1133         x913 
 0.096332845 -0.011028228 -0.002834551 -0.237975290  0.174797050 -0.013690114  0.013425314 
        x934         x752         x691         x876         x739         x887        x1157 
 0.554177139  0.044360319 -0.188254629  0.006470443  0.008685916  0.043838862 -0.152705407 
        x983         x864         x741         x850        x1034 
 0.215573756 -0.178425752  0.178251834  0.011798168  0.103077427 
```