The Maximum Agreement Prediction via the Concordance Correlation Coefficient

Taeho Kim[*], George Luta[†], Matteo Bottai[‡], Pierre Chausse[§], Gheorghe Doros[¶], Edsel A. Pena[‖]

Abstract

The vignette explains how to use the malp package to compute maximum agreement prection, construct confidence intervals for the prediction and illustrate the result.

1 Introduction

Suppose we have a $p \times 1$ vector of covariates x_0 and a dependent variable Y. The MALP predictor is defined as:

$$\begin{aligned}\tilde{Y}^{\star}(x_0) &= \mu_Y + \frac{1}{\gamma}\Sigma'_{XY}\Sigma_X^{-1}(x_0-\mu_X) \\ &= (1-1/\gamma)\,\mu_Y + (1/\gamma)\,\tilde{Y}^{\dagger}(x_0)\,,\end{aligned}$$

where γ is the concordance correlation coefficient (CCC), μ_Y is the population mean of Y ,Σ_X is the covariance matrix of the random vector X, Σ_{XY} is the vector of covariance between X and Y and $\tilde{Y}^{\dagger}(x_0)$ is the best linear predictor defined as

$$\begin{aligned}\tilde{Y}^{\dagger}(x_0) &= \beta_1 + \beta'_2 x_0 \\ &= (\mu_Y - \Sigma'_{XY}\Sigma_X^{-1}\mu_X) + \Sigma'_{XY}\Sigma_X^{-1}x_0 \\ &= \mu_Y + \Sigma'_{XY}\Sigma_X^{-1}(x_0-\mu_X)\end{aligned}$$

The CCC γ is defined as

$$\begin{aligned}\gamma &= \frac{[\Sigma'_{XY}\Sigma_X^{-1}\Sigma_{XY}]^{1/2}}{\sigma_Y} \\ &= \frac{[\beta'_2\Sigma_X\beta_2]^{1/2}}{\sigma_Y}\,.\end{aligned}$$

It is therefore the square root of the coefficient of determination R^2 from the regression of Y oin X. Let X_0 be the $(p+1)\times 1$ vector $\{1,\ x'_0\}'$. Then the MALP predictor can also be written as:

[*]Lehigh University, tak422@lehigh.edu
[†]Georgetown University, George.Luta@georgetown.edu
[‡]Karolinska Institutet, matteo.bottai@ki.se
[§]University of Waterloo, pchausse@uwaterloo.ca
[¶]Boston University,, doros@bu.edu
[‖]University of South Carolina, pena@stat.sc.edu

$$\begin{aligned}\tilde{Y}^{\star}(x) &= (1-1/\gamma)\,\mu_{\mathrm{Y}} + (\beta_1 + x_0'\beta_2)/\gamma \\ &= [(1-1/\gamma)\mu_{\mathrm{Y}} + \beta_1/\gamma] + x_0'[\beta_2/\gamma] \\ &\equiv \alpha_1 + x_0'\alpha_2 \\ &\equiv X'\alpha\end{aligned}$$

Assuming we have an IID sample $\{Y_i, x_i\}$ of size n, a consistent estimator of the MALP at $x = x_0$ is:

$$\hat{Y}^{\star}(x_0) = \hat{\alpha}_1 + x_0'\hat{\alpha}_2$$

where $\hat{\alpha}_1 = (1 - 1/\hat{\gamma})\overline{\mathrm{Y}} + \hat{\beta}_1/\hat{\gamma}$, $\hat{\alpha}_2 = \hat{\beta}_2/\hat{\gamma}$, $\hat{\beta}_1$ and $\hat{\beta}_2$ are the least square estimators, $\overline{\mathrm{Y}}$ is the sample mean of Y and $\hat{\gamma}$ is the square root of the least square coefficient of determination.

To obtain the variance of the MALP predictor under general distribution, we first note that $\tilde{Y}^{\star}(x_0)$ can be written as a non-linear function $g(\theta; x_0)$, where

$$\theta = \begin{pmatrix} \mu_Y \\ \mu_X \\ \sigma_Y^2 \\ \Sigma_{XY} \\ \mathrm{Vec}(\Sigma_X) \end{pmatrix}$$

If we define the ith observation T_i from an iid sample as

$$T_i = \begin{pmatrix} Y_i \\ X_i \\ (Y_i - \bar{Y})^2 \\ (X_i - \bar{X})(Y_i - \bar{Y}) \\ \mathrm{Vec}((X_i - \bar{X})(X_i - \bar{X})') \end{pmatrix}.$$

and we assume that $\mathrm{E}(\|\{Y, X'\}'\|^4) < \infty$, then $\sqrt{n}(\bar{T} - \theta) \xrightarrow{d} N(0, \Xi)$, by the Lindeberg-Lévy Central Limit Theorem. Then, using the Delta Method, $\sqrt{n}[\hat{Y}^{\star}(x_0) - \tilde{Y}^{\star}(x_0)] \xrightarrow{d} N(0,\ \nabla g(\theta; x_0)'\Xi\nabla g(\theta; x_0))$, where

$$\nabla g(\theta; x_0) = \begin{pmatrix} 1 \\ -\frac{\beta_2}{\gamma} \\ \frac{\beta_2'(x_0 - \mu_X)}{2\gamma\sigma_Y^2} \\ \frac{\Sigma_X^{-1}(x_0 - \mu_X)}{\gamma} - \frac{[\beta_2'(x_0 - \mu_X)]\beta_2}{\gamma^3\sigma_Y^2} \\ -\frac{\mathrm{Vec}[\Sigma_X^{-1}(x_0 - \mu_X)\beta_2']}{\gamma} + \frac{[\beta_2'(x_0 - \mu_X)]\mathrm{Vec}(\beta_2\beta_2')}{2\gamma^3\sigma_Y^2} \end{pmatrix}$$

We can estimate this variance using the sample estimates of β_2, γ, σ_Y^2, Σ_X, μ_X and Ξ. The covariance matrix of $\hat{\alpha}$ can be obtained by using the fact that:

$$\mathrm{Var}[\hat{Y}^{\star}(x_0)] = \begin{pmatrix} 1 & x_0' \end{pmatrix} \mathrm{Var}(\hat{\alpha}) \begin{pmatrix} 1 \\ x_0 \end{pmatrix}$$

By rearranging, we get obtain the following result:

$$\mathrm{Var}\left[\sqrt{n}(\hat{\alpha} - \alpha)\right] \rightarrow \begin{pmatrix} (\mu_X' A'\Xi_{22} A\mu_X - 2\mu_x' A'\Xi_{21} B + \sigma_Y^2(1-\gamma)) & (B'\Xi_{12} A - \mu_X' A'\Xi_{22} A) \\ (A'\Xi_{21} B - A'\Xi_{22} A\mu_X) & A'\Xi_{22} A \end{pmatrix},$$

where

$$A = \begin{pmatrix} \frac{\beta_2'}{2\gamma\sigma_Y^2} \\ \frac{\Sigma_X^{-1}}{\gamma} - \frac{\beta_2\beta_2'}{\gamma^3\sigma_Y^2} \\ \frac{-\beta_2\otimes\Sigma_X^{-1}}{\gamma} + \frac{Vec(\beta_2\beta_2')\beta_2'}{2\gamma^3\sigma_Y^2} \end{pmatrix} \quad , \quad B = \begin{pmatrix} 1 \\ -\frac{\beta_2}{\gamma} \end{pmatrix}$$

and Ξ_{ij} are partitions of Ξ.

In general, there is no closed-form expression for the variance, but if we assume that the vector $\{Y, X'\}'$ is normally distributed, we can show that the above asymptotic variance can be written as:

$$\sigma^2_{\text{MA}}(x_0) \quad = \quad \sigma^2_{\text{Y}}(1-\gamma^2)\left[\frac{2}{1+\gamma} + \frac{(x_0-\mu_X)'\Sigma_X^{-1}(x_0-\mu_X)}{\gamma^2} - \frac{(1-\gamma^2)}{\sigma_Y^2\gamma^4}\left[\beta_2'(x_0-\mu_X)\right]^2\right]$$

Since the asymptotic variance of the estimated best linear predictor under the homoskedasticity assumption is

$$\sigma^2_{\text{LS}}(x_0) = \sigma^2_Y(1-\gamma^2)\left[1 + (x_0-\mu_X)\Sigma_X^{-1}(x_0-\mu_X)'\right]$$

We can write $\sigma^2_{\text{MA}}(x_0)$ as a function of $\sigma^2_{\text{LS}}(x_0)$:

$$\sigma^2_{\text{MA}}(x_0) = \frac{\sigma^2_{\text{LS}}(x_0)}{\gamma^2} + \frac{\sigma^2_Y(1-\gamma^2)}{\gamma^2} \times \left[\frac{2\gamma^2-\gamma-1}{1+\gamma} - \frac{(1-\gamma^2)}{\sigma_Y^2\gamma^2}\left[\beta_2'(x_0-\mu_X)\right]^2\right]$$

We obtain consistent estimator of the variances by replacing the population values of γ, $\sigma^2_{\text{LS}}(x_0)$, σ^2_Y, μ_X, Σ_X, β_2 and μ_Y by their sample estimates. Note that there is no unique way to estimate $\sigma^2_{\text{LS}}(x_0)$. For example, if we estimate it using the following:

$$\begin{aligned} \hat\sigma^2_Y &= \frac{1}{n-1}\sum_{i=1}^n (Y_i-\bar Y)^2 \\ \hat\gamma &= \frac{[\hat\beta_2'\hat\Sigma_X\hat\beta_2]^{1/2}}{\hat\sigma_Y} \\ \hat\Sigma_X &= \frac{1}{n-1}\sum_{i=1}^n (X_i-\bar X)(X_i-\bar X)'\,, \end{aligned}$$

which is the default method that we use in the package, we have the following relationship between this estimator and the one computed by the `predict` method for `lm` objects.

$$\hat\sigma^2_{\text{LS}}(x_0) = \frac{n-p-1}{n}\hat\sigma^{2(\text{R})}_{\text{LS}}(x_0) + \frac{\hat\sigma^2_Y(1-\hat\gamma^2)}{n}\,,$$

where $\hat\sigma^{2(\text{R})}_{\text{LS}}$ is the one computed by `predict.lm` and $\hat\sigma^2_{\text{LS}}(x_0)$ is the default estimator computed in the `malp` package. The difference comes from how the different estimators from the expression are adjusted for the loss of degrees of freedom. The package offer the option of using the `predict.lm` correction.

Using the same approach as for the general case, we can obtain the following closed-form expression for the asymptotic covariance matrix of $\sqrt{n}(\hat\alpha-\alpha)$ under the normality:

$$\mathrm{Var}\left[\sqrt{n}(\hat\alpha-\alpha)\right]\to\begin{pmatrix}\frac{2\sigma^2}{1+\gamma}+\mu_X\left(\frac{\sigma^2}{\gamma^2}\Sigma_X^{-1}-\frac{(1-\gamma^2)^2}{\gamma^4}\beta_2\beta_2'\right)\mu_X' & \mu_X\left(-\frac{\sigma^2}{\gamma^2}\Sigma_X^{-1}+\frac{(1-\gamma^2)^2}{\gamma^4}\beta_2\beta_2'\right)\\ \left(-\frac{\sigma^2}{\gamma^2}\Sigma_X^{-1}+\frac{(1-\gamma^2)^2}{\gamma^4}\beta_2\beta_2'\right)\mu_X' & \frac{\sigma^2}{\gamma^2}\Sigma_X^{-1}-\frac{(1-\gamma^2)^2}{\gamma^4}\beta_2\beta_2'\end{pmatrix},$$

where $\sigma^2=\sigma_Y^2(1-\gamma^2)$ is the variance of the least squares residuals. We can simply this expression further by using the fact that the asymptotic variance Ω of $\hat\beta_2$ under the homoskedasticity assumption, which is implied by the normality assumption, is $\sigma^2\Sigma_X^{-1}$ and $\mu_X'\beta_2=(\mu_Y-\beta_1)$:

$$\mathrm{Var}\left[\sqrt{n}(\hat\alpha-\alpha)\right]\to\begin{pmatrix}\frac{2\sigma^2}{1+\gamma}+\frac{1}{\gamma^2}\mu_{\mathrm{X}}\Omega\mu_{\mathrm{X}}'-\frac{(1-\gamma^2)^2}{\gamma^4}(\mu_{\mathrm{Y}}-\beta_1)^2 & -\frac{1}{\gamma^2}\mu_{\mathrm{X}}\Omega+\frac{(1-\gamma^2)^2}{\gamma^4}(\mu_{\mathrm{Y}}-\beta_1)\beta_2'\\ -\frac{1}{\gamma^2}\Omega\mu_{\mathrm{X}}'+\frac{(1-\gamma^2)^2}{\gamma^4}(\mu_{\mathrm{Y}}-\beta_1)\beta_2 & \frac{1}{\gamma^2}\Omega-\frac{(1-\gamma^2)^2}{\gamma^4}\beta_2\beta_2'\end{pmatrix},$$

2 The `malp` package

The main function is `malp`, which returns an object of class `malp`. The purpose of this function is to compute the estimates $\hat\alpha$. The function has two arguments: `formula` and `data`. The former is like the formula provided to `lm` for linear regressions and `data` is a `data.frame` containing all variables included in the formula. In the following example, we have one independent variable and one dependent variable.

```r
## Data just for the testing
library(malp)
set.seed(11223344)
x<-rnorm(100)
y<-1+2*x+rnorm(100)
dat <- data.frame(x,y)
fit <- malp(y~x, dat)
```

The `malp` object has its own `print` method that returns the coefficient estimates $\hat\alpha$.

```r
print(fit, digits=5)
```

```

Call:
malp(formula = y ~ x, data = dat)

Coefficients:
(Intercept)            x
     1.0396       2.3261
```

The object is a list with the following elements:

- `coefficients`: The vector $\hat\alpha$
- `gamma`: The value of $\hat\gamma$
- `lm`: The `lm` object from the least squares regression
- `varY`: The value of $\hat\sigma_Y^2$
- `na.action`: The observations that were removed due to missing values.
- `data`: The `data.frame` used for the estimation. If missing values were present, it excludes them.

2.1 The `vcov` method

By default, the covariance matrix of the MALP coefficients is computed without assuming normality. This is the best option, because it is also valid under normality. The method returns a $(p+1)\times(p+1)$ covariance matrix named by the variable names in the formula:

```r
vcov(fit)
```

```

             (Intercept)            x
(Intercept)  0.012846112 -0.001516036
x           -0.001516036  0.007753234
```

To compute the variance under the normality assumption using the closed-form expression presented in the previous section, we set the argument `method` to `"Normal"`:

```
vcov(fit, method="Normal")
```

```
             (Intercept)            x
(Intercept) 0.0129001202 0.0001546548
x           0.0001546548 0.0094871725
```

Since the covariance matrices derived in the previous section are based on the Delta method, we may want to rely on simulation techniques if the sample size is too small. The package offers two options: Bootstrap or Jackknife. For the bootstrap method, the number of bootstrap samples is set by the argument `B`. For example, the first method below computes the estimate using 100 bootstrap samples and the second computes the jackknife estimate:

```
v1 <- vcov(fit, method="Boot", B=100)
v1
```

```
             (Intercept)            x
(Intercept)  0.015941693 -0.003226687
x           -0.003226687  0.010299672
```

```
v2 <- vcov(fit, method="Jackknife")
```

By default, the covariance matrix under normality is not based on least squares degrees of freedom correction. If we want this correction, which only applies under normality, we set the argument `LSdfCorr` to `TRUE`.

```
vcov(fit, method="Normal", LSdfCorr=TRUE)
```

```
             (Intercept)            x
(Intercept) 0.0130317908 0.0001584821
x           0.0001584821 0.0097219557
```

2.2 The summary method

This method returns detailed information about the estimation. In particular, it returns the standard errors, t-ratios and p-values of the $\hat{\alpha}$. The arguments of the function are used to specify how to compute the standard errors. The options are the same as for `vcov`, but the argument that specifies the method is `vcovMet`. By default, the standard errors are based on the general expression for unknown distributions. The method returns an object of class `summary.malp`, which has its own `print` method.

```
print(summary(fit), digits=5)
```

```
Call:
malp(formula = y ~ x, data = dat)


Coefficients:
            Estimate Std. Error t value  Pr(>|t|)
(Intercept) 1.039611   0.113341  9.1724 < 2.2e-16 ***
x           2.326082   0.088052 26.4170 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

PCC: 0.90811
CCC: 0.90811
MSE: 1.2769
```

```
print(summary(fit, "Boot", B=100), digits=5)
```

```
Call:
malp(formula = y ~ x, data = dat)


Coefficients:
            Estimate Std. Error t value  Pr(>|t|)
(Intercept) 1.039611   0.108124   9.615 < 2.2e-16 ***
x           2.326082   0.092564  25.130 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
PCC: 0.90811
CCC: 0.90811
MSE: 1.2769
```

The three different good fit measures printed by `summary` can also be extracted manually for both the MALP and LS fits:

```
s <- summary(fit)
cbind(MALP=s$fitMALP, LSLP=s$fitLSLP)
```

```
         MALP      LSLP
PCC 0.9081066 0.9081066
CCC 0.9081066 0.9039040
MSE 1.2768623 1.2181947
```

2.3 The `predict` method

The method works like the `predict.lm` method. By default, it predicts the dependent variables for the same values of the covariates used to fit the model. The following prints the first four MALP predictors.

```
pr <- predict(fit)
pr[1:4]
```

```
         1          2          3          4
-1.2382594 -0.9168157  2.6455775  1.1622700
```

If the argument `se.fit` is set to `TRUE`, it returns a list with the element `fit` being the predicted values and the argument `se.fit` being the standard errors of the MALP predictions.

```
pr <- predict(fit, se.fit=TRUE)
pr$fit[1:4]
```

```
         1          2          3          4
-1.2382594 -0.9168157  2.6455775  1.1622700
```

```
pr$se.fit[1:4]
```

```
        1         2         3         4
0.1524812 0.1445030 0.1202019 0.1127288
```

As for `vcov` and `summary`, the default standard errors returned by `predict` are based on the asymptotic theory for unknown distributions. We can also use the closed-form expression under normality by setting the argument `vcovMet` to `"Normal"`. For standard errors based on the `predict.lm` version $\hat{\sigma}^{2(R)}_{\text{LS}}(x_0)$, we set the argument `LSdfCorr` to `TRUE` (it stands for Least Squares degrees of freedom Correction).

```
pr <- predict(fit, se.fit=TRUE, vcovMet="Normal", LSdfCorr=TRUE)
pr$se.fit[1:4]
```

```
        1         2         3         4
0.1485435 0.1402261 0.1338114 0.1144386
```

For small samples, we also have the option of using simulated methods. All we need is to set the argument `vcovMet` to either "Boot" or "Jackknife". For the "Boot" option, the number of bootstrap sample is set by the argument `Bse.`

```
pr <- predict(fit, se.fit=TRUE, vcovMet="Boot", Bse.=100)
pr$se.fit[1:4]
```

```
        1         2         3         4
0.1673680 0.1580201 0.1229206 0.1178964
```

If we want to predict Y for specific values of X, we can pass the specific values to the argument `newdata` as a `data.frame`. The data.frame must contain values for all covariates in the formula.

```
newd <- data.frame(x=c(-.3,.3,1.5))
predict(fit, newdata=newd)
```

```
        1         2         3
0.3417864 1.7374356 4.5287340
```

Note that the CCC can be computed manually using the `ccc` function included in the package. We first use predict to get the fitted values and then compute the CCC:

```
yhat <- predict(fit)
ccc(y,yhat)
```

```
[1] 0.9081066
```

2.4 Confidence Intervals

The confidence intervals for the predictor also comes from the `predict` method, but given the different options, it deserves its own section. By default, parametric confidence intervals are produced by setting the argument `interval` to "confidence".

```
pr <- predict(fit, newdata=newd, interval="confidence")
pr
```

```
        fit      lwr       upr
1 0.3417864 0.106154 0.5774187
2 1.7374356 1.517131 1.9577401
3 4.5287340 4.214266 4.8432015
```

The options for standard errors used to compute the parametric confidence intervals are explained in the previous section. For example, we can construct the intervals using the Jackknife standard errors this way:

```
pr <- predict(fit, newdata=newd, interval="confidence", vcovMet="Jackknife")
pr
```

```
        fit       lwr       upr
1 0.3417864 0.1035505 0.5800222
2 1.7374356 1.5155789 1.9592922
3 4.5287340 4.2089752 4.8484927
```

Note that if the argument `se.fit` is set to `TRUE`, it returns a list with the element `fit` being the intervals and the element `se.fit` being the standard errors.

It is also possible to compute bootstrap confidence intervals. The options are

- **norm**: Normal interval
- **basic**: Basic interval
- **stud**: Studentized intervals
- **perc**: Percentile intervals
- **bca**: Bias corrected intervals.

To obtain one of these bootstrap confidence intervals, we set the argument `bootInterval` to `TRUE`. The type of interval is obtained by setting the argument `bootIntType` to one of the above options. If set to "all", the default, the functions returns all interval in a list. These intervals are computed using `boot` and `boot.ci` from the `boot` package. For the studentized interval, `se.fit` must be set to `TRUE`. If `bootIntType` it set to "all" and `se.fit` to `FALSE`, the function will not return a studentized interval.

With `bootInterval=TRUE`, the function returns a list of intervals. The name of each element is the interval type. If `bootIntType` is not set to "all", the function return a list of length equal to 1.

```
pr <- predict(fit, newdata=newd, bootInterval=TRUE, se.fit=TRUE,
              vcovMet="Jackknife", B.=100)
```

```
Warning in norm.inter(t, adj.alpha): extreme order statistics used as endpoints
```

```
pr$norm
```

```
           fit     lower     upper
[1,] 0.3417864 0.1218451 0.5772914
[2,] 1.7374356 1.5073791 1.9544696
[3,] 4.5287340 4.1383931 4.8488801
```

```
pr$stud
```

```
           fit      lower     upper
[1,] 0.3417864 0.07929947 0.5843669
[2,] 1.7374356 1.47674792 1.9786607
[3,] 4.5287340 4.07366039 4.8227543
```

```
pr$bca
```

```
           fit    lower    upper
[1,] 0.3417864 0.154305 0.637135
[2,] 1.7374356 1.474833 1.950115
[3,] 4.5287340 4.212487 4.828053
```

```
pr$perc
```

```
           fit      lower     upper
[1,] 0.3417864 0.08343053 0.5883467
[2,] 1.7374356 1.51042212 1.9843750
[3,] 4.5287340 4.22511598 4.9117400
```

It is possible to speedup the process by changing the arguments `parallel`, `ncpus` and `cl`. See the help file of the `boot` function for more details. For example, the following would compute the bootstrap interval in parallel using 8 CPU's

```
pr <- predict(fit, newdata=newd, bootInterval=TRUE, se.fit=TRUE,
              vcovMet="Jackknife", B.=100, parallel="multicore",
              ncpus=8)
```

2.5 Prediction intervals

The prediction intervals for LSLP and MALP are respectively:

$$\left[\hat{Y}^\dagger(x_0) \pm z_{\alpha/2}\sqrt{S_{\mathrm{Y}}(1-\gamma^2)+\sigma^2_{\mathrm{LS}}(x_0)/n}\right]$$

and

$$\left[\hat{Y}^\star(x_0) + \hat{b}(x_0) \pm z_{\alpha/2}\sqrt{S_{\mathrm{Y}}(1-\gamma^2)+\sigma^2_{\mathrm{MA}}(x_0)/n}\right]$$

where $\hat{b}(x_0)$ is the prediction bias of MALP. It turns out that $\hat{Y}^\star(x_0)+\hat{b}(x_0)=\hat{Y}^\dagger(x_0)$, so the prediction interval for MALP can be written as

$$\left[\hat{Y}^\dagger(x_0) \pm z_{\alpha/2}\sqrt{S_{\mathrm{Y}}(1-\gamma^2)+\sigma^2_{\mathrm{MA}}(x_0)/n}\right]$$

Since $S_{\mathrm{Y}}(1-\gamma^2)$ is the estimated variance of the error term of least squares models, if the argument `LSdfCorr` is set to `TRUE`, $S_{\mathrm{Y}}(1-\gamma^2)$ is multiplied $(n-1)/df$, where df is the least squares residuals degrees of freedom. Note that bootstrap intervals are not available for prediction intervals.

```
pr1 <- predict(fit, newdata=newd, interval="prediction",
               includeLS=TRUE)
pr2 <- predict(fit, newdata=newd, interval="confidence")
pr1$MALP
```

```
        fit        lwr      upr
1 0.3417864 -1.7844510 2.589306
2 1.7374356 -0.5154542 3.855106
3 4.5287340  2.0078510 6.401394
```

```
pr2
```

```
        fit       lwr       upr
1 0.3417864 0.106154 0.5774187
2 1.7374356 1.517131 1.9577401
3 4.5287340 4.214266 4.8432015
```

Since only the variance differs between the prediction interval of LSLP amnd MALP, they are very close to each other:

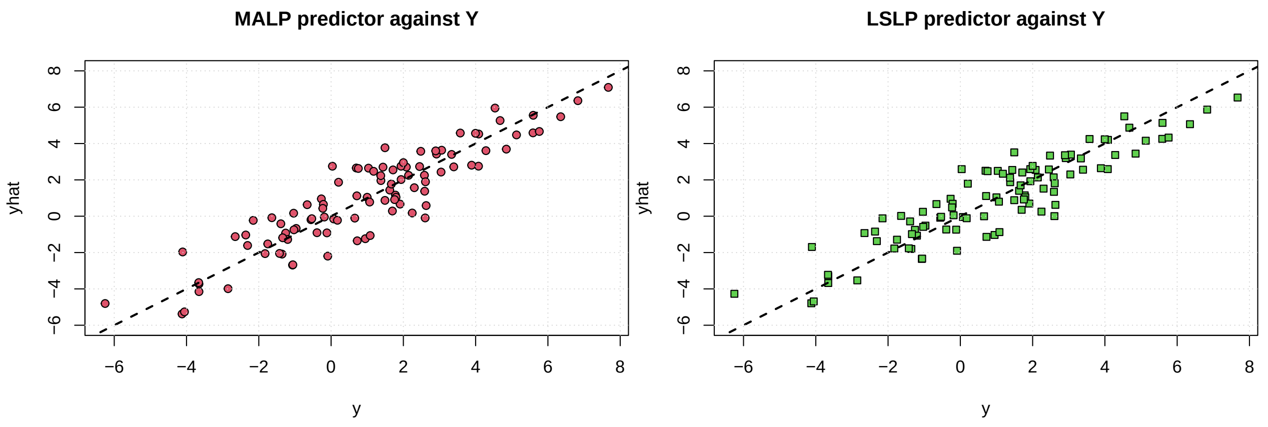
```
pr1$LSLP
```

```
        fit        lwr      upr
1 0.4024273 -1.7833582 2.588213
2 1.6698257 -0.5161263 3.855778
3 4.2046223  2.0000434 6.409201
```

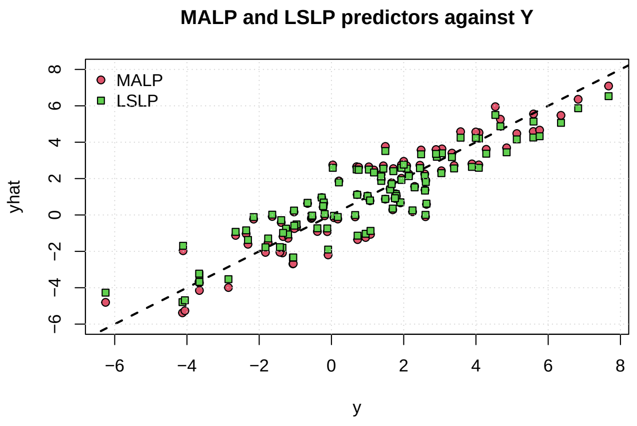
2.6 The `plot` method

This method produces a scatter plot of the dependent variable against the fitted values from the MALP, LSLP or both. A 45 degree line is added to better evaluate the level of agreement. The following shows one example with `which="MALP"`. The other options are “LSLP” and “Both”.

```
plot(fit, which="MALP", main="MALP predictor against Y", ylim=c(-6,8), col=c(1,1))
plot(fit, which="LSLP", main="LSLP predictor against Y", ylim=c(-6,8), col=c(1,1))
```



```
plot(fit, which="Both", main="MALP and LSLP predictors against Y", ylim=c(-6,8), col=c(1,1))
```


MALP and LSLP predictors against Y
MALP
LSLP
yhat
y