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 and a dependent variable Y. The MALP predictor is defined as:

$$\tilde{Y}^{\star}(x) = (1 - 1/\gamma)\,\mu_{\mathrm{Y}} + (1/\gamma)\,\tilde{Y}^{\dagger}(x)\,,$$

where γ is the concordance correlation coefficient (CCC), μ_{Y} is the population mean of Y and $\tilde{Y}^{\dagger}(x)$ is the best linear predictor. For any predictor $\tilde{Y}$, the CCC is defined as

$$\gamma = \frac{2\sigma_{\mathrm{Y}\tilde{\mathrm{Y}}}}{\sigma^2_{\mathrm{Y}} + \sigma^2_{\tilde{\mathrm{Y}}} + (\mu_{\mathrm{Y}} - \mu_{\tilde{\mathrm{Y}}})}\,,$$

where σ_{xy} is the covariance between x and y and σ^2_x is the variance of x. When the predictor is the MALP defined above, the CCC is equal to the square root of the coefficient of determination R^2 of the best linear predictor.

Let $X = \{1, x'\}'$ and let the best linear predictor $\tilde{Y}^{\dagger}(x)$ be $X'\beta \equiv \beta_1 + x'\beta_2$, where $\beta_1 = \mu_{\mathrm{Y}} - \mu'_{\mathrm{x}}\beta_2$ and $\beta_2 = \mathrm{Var}(X)^{-1}\mathrm{Cov}(X, Y)$, then the MALP can be written as:

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

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

$$\hat{Y}^{\star}(x_0) = \hat{\alpha}_1 + x'_0\hat{\alpha}_2$$

*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

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.

If we assume that $\{Y, x\}$ are jointly normal, the MALP predictor $\hat{Y}^{\star}(x_0)$ is asymptotically normal with the following variance:

$$\begin{aligned}
\sigma^2_{\mathrm{MA}}(x_0) &= \sigma^2_{\mathrm{Y}}(1-\gamma^2) \times \\
&\quad \left[\frac{2}{1+\gamma} + \frac{1}{\gamma^2}(x_0 - \mu_{\mathrm{X}})'\mathrm{Var}(\mathrm{X})^{-1}(x_0 - \mu_{\mathrm{X}}) - \frac{(1-\gamma^2)}{\sigma^2_{\mathrm{Y}}\gamma^4}\left[\mathrm{Cov}(\mathrm{X},\mathrm{Y})'\mathrm{Var}(\mathrm{X})^{-1}(x_0 - \mu_{\mathrm{X}})\right]^2\right] \\
&= \sigma^2_{\mathrm{Y}}(1-\gamma^2) \times \\
&\quad \left[\frac{2}{1+\gamma} + \frac{1}{\gamma^2}(x_0 - \mu_{\mathrm{X}})'\mathrm{Var}(\mathrm{X})^{-1}(x_0 - \mu_{\mathrm{X}}) - \frac{(1-\gamma^2)}{\sigma^2_{\mathrm{Y}}\gamma^4}\left[\tilde{Y}^{\dagger}(x_0) - \mu_{\mathrm{Y}}\right]^2\right]
\end{aligned}$$

Since the asymptotic variance of the estimated best linear predictor is

$$\sigma^2_{\mathrm{LS}}(x_0) = \sigma^2_{\mathrm{Y}}(1-\gamma^2)\left[1 + (x_0 - \mu_{\mathrm{X}})\Sigma^{-1}_{\mathrm{XX}}(x_0 - \mu_{\mathrm{X}})'\right]$$

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

$$\sigma^2_{\mathrm{MA}}(x_0) = \frac{\sigma^2_{\mathrm{LS}}(x_0)}{\gamma^2} + \frac{\sigma^2_{\mathrm{Y}}(1-\gamma^2)}{\gamma^2} \times \left[\frac{2\gamma^2 - \gamma - 1}{1+\gamma} - \frac{(1-\gamma^2)}{\sigma^2_{\mathrm{Y}}\gamma^2}[\tilde{Y}^{\dagger}(x_0) - \mu_{\mathrm{Y}}]^2\right]$$

We obtain consistent estimator of the variances by replacing the population values of γ, $\sigma^2_{\mathrm{LS}}(x_0)$, σ^2_{Y}, μ_{X}, Σ_{XX}, $\tilde{Y}^{\dagger}(x_0)$ and μ_{Y} by their sample estimates. Note that there is no unique way to estimate $\sigma^2_{\mathrm{LS}}(x_0)$. For example, if we estimate it using the following:

$$\begin{aligned}
\hat{\sigma}^2_{\mathrm{Y}} &= \frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2 \\
\hat{\gamma} &= \frac{\sum_{i=1}^{n}(Y_i - \bar{Y})(\hat{Y}^{\star}_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2 \sum_{i=1}^{n}(\hat{Y}^{\star}_i - \bar{Y})^2}} \\
\hat{\Sigma}_{\mathrm{XX}} &= \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})',
\end{aligned}$$

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.

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

where $\hat{\sigma}^{2(R)}_{\mathrm{LS}}$ is the one computed by `predict.lm` and $\hat{\sigma}^2_{\mathrm{LS}}(x_0)$ is the default estimator computed in the `malp` package. The difference comes for 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.

Alternatively, we can obtain $\sigma^2_{\mathrm{MA}}(x_0)$ indirectly by noting that

$$\sigma^2_{\mathrm{MA}}(x_0) = \{1, x_0\}V\{1, x_0\}',$$

where V is the asymptotic variance of $\sqrt{n}(\hat{\alpha}-\alpha)$ defined as:

$$V = \begin{pmatrix} \frac{2\sigma^2}{1+\gamma} + \mu_X \left(\frac{\sigma^2}{\gamma^2}\mathrm{Var}(X)^{-1} - \frac{(1-\gamma^2)^2}{\gamma^4}\beta_2\beta_2'\right)\mu_X' & \mu_X\left(-\frac{\sigma^2}{\gamma^2}\mathrm{Var}(X)^{-1} + \frac{(1-\gamma^2)^2}{\gamma^4}\beta_2\beta_2'\right) \\ \left(-\frac{\sigma^2}{\gamma^2}\mathrm{Var}(X)^{-1} + \frac{(1-\gamma^2)^2}{\gamma^4}\beta_2\beta_2'\right)\mu_X' & \frac{\sigma^2}{\gamma^2}\mathrm{Var}(X)^{-1} - \frac{(1-\gamma^2)^2}{\gamma^4}\beta_2\beta_2' \end{pmatrix},$$

where $\sigma^2 = \sigma_Y^2(1-\gamma^2)$ is the variance of the least squares residuals. We can simply this expression by using the fact that the asymptotic variance Ω of $\hat{\beta}_2$ under the homoskedasticity assumption is $\sigma^2\mathrm{Var}(X)^{-1}$ and $\mu_X\beta_2 = (\mu_Y - \beta_1)$:

$$V = \begin{pmatrix} \frac{2\sigma^2}{1+\gamma} + \frac{1}{\gamma^2}\mu_X\Omega\mu_X' - \frac{(1-\gamma^2)^2}{\gamma^4}(\mu_Y-\beta_1)^2 & -\frac{1}{\gamma^2}\mu_X\Omega + \frac{(1-\gamma^2)^2}{\gamma^4}(\mu_Y-\beta_1)\beta_2' \\ -\frac{1}{\gamma^2}\Omega\mu_X' + \frac{(1-\gamma^2)^2}{\gamma^4}(\mu_Y-\beta_1)\beta_2 & \frac{1}{\gamma^2}\Omega - \frac{(1-\gamma^2)^2}{\gamma^4}\beta_2\beta_2' \end{pmatrix},$$

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.

```
## 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 $\hat{\alpha}$.

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

```
##
## Call:
## malp(formula = y ~ x, data = dat)
##
## Coefficients:
## (Intercept)           x
##      1.0396       2.3261
```

2.1 The `vcov` method

Under the normality assumption, we can estimate the variance of $\hat{\alpha}$ by $\hat{V}/n$. To estimate $\hat{V}$, we simple replace the β_i's by the least squares estimates, Ω by the least squares covariance matrix times n, the population means by the sample means, σ^2 by the estimated variance of the least squares residuals and γ by the square root of the R^2. This is the default method of `vcov`:

```
vcov(fit)
```

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

If we relax the normality assumption, we have to rely on simulation methods. There are two options: Bootstrap or Jackknife. For the bootstrap method, the number of bootstrap samples is set by the argument `B`. For example:

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

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

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

By default, the covariance matrix is not based on least squares degrees of freedom correction, we set the argument `LSdfCorr` to `TRUE`.

```
vcov(fit, 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 to specify how to compute the standard errors (the default is "Asymptotic"). The options are the same as for `vcov`. 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   0.113579  9.1532 < 2.2e-16 ***
## x           2.326082   0.097402 23.8812 < 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
```

If we only want to see the value of the MSE, PCC and CCC, we can avoid computing the standard errors, which can be a problem for very large samples, by setting the argument `se` to `FALSE`:

```
(s <- summary(fit, se=FALSE))
```

```
##
## Call:
## malp(formula = y ~ x, data = dat)
##
## (Summary without standard errors)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.0396         NA      NA       NA
## x             2.3261         NA      NA       NA
##
## PCC: 0.90811
## CCC: 0.90811
## MSE: 1.2769
```

These "good fit" measures can also be extracted from the `summary` output. It includes the measure for both MALP and LSLP

```
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.

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

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

In that case, it returns predicted values only. 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.

```
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
## 0.1472929 0.1391092 0.1328005 0.1137665
```

By default, the standard errors returned by `predict` are based on the asymptotic theory under the assumption of the joint normality of $\{Y, x\}$. It uses the expression from the Introduction section for $\hat{\sigma}^2_{\text{MA}}(x_0)$ with $\hat{\sigma}^2_{\text{LS}}(x_0)$. 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, LSdfCorr=TRUE)
pr$se.fit[1:4]
```

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

If we are not willing to assume normality, we have the option of computing the standard errors using the expression $\sqrt{X_0' \text{Var}(\hat{\alpha}) X_0}$, where $X_0 = \{1, x_0'\}'$ and $\text{Var}(\hat{\alpha})$ is computed with `vcov`. All we need to do 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
```

Alternatively, we can compute the standard errors manually using the standard errors of $\hat{\alpha}$. For the above predictions, we have the following standard errors:

```
V <- vcov(fit)
X <- cbind(1,c(-.3,.3,1.5))
apply(X, 1, function(x) c(sqrt(t(x)%*%V%*%x)))
```

```
## [1] 0.1168810 0.1176723 0.1863068
```

Which is identical to the result from the `predict` method:

```
predict(fit, newdata=newd, se.fit=TRUE)$se.fit
```

```
##         1         2         3
## 0.1168810 0.1176723 0.1863068
```

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 0.1127038 0.5708689
## 2 1.7374356 1.5068022 1.9680689
## 3 4.5287340 4.1635793 4.8938886
```

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 return 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")
pr$norm
```

```
##            fit     lower    upper
## [1,] 0.7371137 0.1438565 1.249577
## [2,] 1.1866577 0.6047423 1.710968
## [3,] 2.0857456 1.0321086 3.128154
```

```
pr$stud
```

```
##            fit       lower    upper
## [1,] 0.7371137 -0.02094787 1.317056
## [2,] 1.1866577  0.68668149 1.971728
## [3,] 2.0857456  1.40123686 3.537060
```

```
pr$bca
```

```
##            fit     lower    upper
## [1,] 0.7371137 0.3879852 1.639559
## [2,] 1.1866577 0.8820619 2.471166
## [3,] 2.0857456 1.5885509 4.612881
```

```
pr$perc
```

```
##            fit     lower    upper
## [1,] 0.7371137 0.3773472 1.581598
## [2,] 1.1866577 0.8181034 1.809035
## [3,] 2.0857456 1.3319190 3.120020
```

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. For valid intervals in case of non-normality, we can set the argument `vcovMet` to either "Boot" or "Jackknife" to obtain a consistent estimator of $\sigma^2_{\mathrm{MA}}(x_0)$.

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

```
##         fit        lwr      upr
## 1 0.3417864 -1.7847330 2.589588
## 2 1.7374356 -0.5156112 3.855263
## 3 4.5287340  2.0070874 6.402157
```

```
pr2
```

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

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", bg=NA, col=1)
```