Prediction with SEM: political democracy example

true

July 27, 2021

Intro

We made an R-function for the SEM based prediction rule and in this note we will analyze the `PoliticalDemocracy` data with this rule.

```
source('~/surfdrive/Predictive-Psychometrics/paper/SEM-Predictive Validity/versie2/Rcode/predict
predicty.lavaan
```

```
## function (object, newdata, xnames, ynames)
## {
##     Sxx = fitted(object)$cov[xnames, xnames]
##     Sxy = fitted(object)$cov[xnames, ynames]
##     mx = fitted(object)$mean[xnames]
##     my = fitted(object)$mean[ynames]
##     Xtest = as.matrix(newdata[, xnames])
##     Xtest = scale(Xtest, center = mx, scale = FALSE)
##     yhat = matrix(my, nrow = nrow(Xtest), ncol = length(ynames),
##         byrow = TRUE) + Xtest %*% solve(Sxx) %*% Sxy
##     return(yhat)
## }
```

Data set

The political democracy data set is the leading data set in lavaan, used for many examples of SEM models.

```
data(PoliticalDemocracy)
```

Let us first partition the data set in a training and test set. We use a small test test of size 10.

```
set.seed(1234)
id.test = sample(1:75, 10)
id.test
```

```
##  [1] 28 22  9  5 38 16  4 14 56 62
```

```
train = PoliticalDemocracy[-id.test, ]
test = PoliticalDemocracy[id.test, ]
```

Now define the SEM model as

```
model <- '
  # latent variable definitions
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + a*y2 + b*y3 + c*y4
  dem65 =~ y5 + a*y6 + b*y7 + c*y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'
```

and fit the model to the training set

```
fit <- sem(model, data = train, meanstructure = TRUE)
```

With the fitted model we can make predictions for the test set using the new function `predicty.lavaan`.

The code to obtain the predicted values is

```
xnames = colnames(PoliticalDemocracy)[-c(5,6,7,8)]
ynames = colnames(PoliticalDemocracy)[c(5,6,7,8)]
yhat = predicty.lavaan(fit, newdata = test, xnames = xnames, ynames = ynames)
yhat
```

```
##          y5          y6        y7         y8
## 28 3.329342  1.57139580  4.987360  2.4258206
## 22 2.377472  0.00294191  3.254321  0.9163651
## 9  3.383852  1.77037098  4.745584  2.5850493
## 5  8.492134  5.32778173  9.408443  7.7855507
## 38 4.809262  2.36605948  6.177126  4.2699479
## 16 7.277811  7.01565642  9.017963  7.3101672
## 4  8.681859  7.80303932 10.324046  9.0797876
## 14 6.967579  5.89242758  8.525011  6.6245157
## 56 2.373642  0.02862126  3.965996  1.2121382
## 62 1.098769 -1.83286311  1.584895 -1.1282093
```

These predicted values can be compared to the observed values

```
test[, ynames]
```

```
##          y5        y6        y7        y8
```

```
## 28 2.385559 0.000000 3.177568 1.116666
## 22 2.500000 0.000000 0.000000 0.000000
## 9  6.250000 3.333333 3.333333 3.333333
## 5  7.500000 3.333333 9.999998 6.666666
## 38 2.844997 0.000000 4.429657 1.485166
## 16 7.500000 1.100000 6.666666 6.666666
## 4  8.907948 8.127979 9.999998 4.615086
## 14 7.500000 0.000000 9.999998 0.000000
## 56 2.500000 0.000000 3.333333 3.333333
## 62 0.000000 0.000000 0.000000 0.000000
```

and we can compute the prediction error per variable

```
colSums((test[, ynames] - yhat)^2)
```

```
##       y5       y6       y7       y8
## 15.57043 87.66969 29.98534 82.12464
```

and overall

```
sum((test[, ynames] - yhat)^2)
```

```
## [1] 215.3501
```

The coefficients of the prediction rule can be obtained as

```
get.coef = function(fit, xnames, ynames){
  Sxx = fitted(fit)$cov[xnames , xnames]
  Sxy = fitted(fit)$cov[xnames , ynames]
  mx = fitted(fit)$mean[xnames]
  my = fitted(fit)$mean[ynames]
  gamma = solve(Sxx) %*% Sxy
  alpha = my - t(gamma) %*% mx
  output = list(
    "alpha" = alpha,
    "gamma" = gamma
  )
  return(output)
}

get.coef(fit, xnames, ynames)
```

```
## $alpha
##          [,1]
## y5 -0.4328722
## y6 -3.4310734
## y7  0.1092348
## y8 -2.9491356
##
## $gamma
##            y5          y6          y7          y8
```

```
## y1 0.51672039 0.26125703 0.25541633 0.32374037
## y2 0.03240832 0.39659157 0.05561578 0.05643788
## y3 0.06711971 0.11781779 0.23181502 0.14599560
## y4 0.12890187 0.09054014 0.22120795 0.36910000
## x1 0.10975755 0.15757648 0.14329531 0.17549622
## x2 0.19052790 0.27353669 0.24874602 0.30464352
## x3 0.03858141 0.05539048 0.05037043 0.06168953
```

Repeated 10 fold CV for varying models

With the following code we define four different SEM models for predicting the response variables, that are, the indicators for democracy in 1965. The first is the same model as used in the lavaan tutorial and used above. In model 2, the structural coefficient of ind60 to dem65 is left out; in Model 3 the residual correlations between dem60 and dem65 are left out; in model 4 both the structural coefficient and the residual correlations are left out.

```
model1 <- '
  # latent variable definitions
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + a*y2 + b*y3 + c*y4
  dem65 =~ y5 + a*y6 + b*y7 + c*y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

model2 <- '
  # latent variable definitions
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + a*y2 + b*y3 + c*y4
  dem65 =~ y5 + a*y6 + b*y7 + c*y8
  # regressions
  dem60 ~ ind60
  dem65 ~ dem60
  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'
```

```r
model3 <- '
  # latent variable definitions
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + a*y2 + b*y3 + c*y4
  dem65 =~ y5 + a*y6 + b*y7 + c*y8
  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60
'

model4 <- '
  # latent variable definitions
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + a*y2 + b*y3 + c*y4
  dem65 =~ y5 + a*y6 + b*y7 + c*y8
  # regressions
  dem60 ~ ind60
  dem65 ~ dem60
'
```

We perform 100 repetitions of 10 fold cross validation and compare the overall prediction error of the models. We also add a simple multivariate multiple linear regression. We focus on the cross-validated prediction error. Furthermore, we look at the estimated coefficients over the 100 x 10 fitted models.

```r
set.seed(1234)
repeats = 100
PE = data.frame(repetition = rep(1:repeats, each = 5),
                model = rep(1:5, repeats),
                pe = rep(0, 5 * repeats))

coefs1 = matrix(NA, 32, 1000)
coefs2 = matrix(NA, 32, 1000)
coefs3 = matrix(NA, 32, 1000)
coefs4 = matrix(NA, 32, 1000)
coefs5 = matrix(NA, 32, 1000)


folds = rep(1:10, length.out = 75)
t = 0
for (r in 1:repeats){
  yhat1 = yhat2 = yhat3 = yhat4 = yhat5 = matrix(NA, 75, 4)
  folds = sample(folds)
  for(k in 1:10){
    t = t + 1
    idx = which(folds == k)
    # approach 1
```

```
    fit <- sem(model1, data = PoliticalDemocracy[-idx, ], meanstructure = TRUE, warn = FALSE)
    yhat1[idx, ] = predicty.lavaan(fit, newdata = PoliticalDemocracy[idx, ], xnames = xnames, yn
    coefs = get.coef(fit, xnames, ynames)
    coefs1[, t] = rbind(matrix(coefs$alpha, 4, 1), matrix(coefs$gamma, 28, 1))

    # approach 2
    fit <- sem(model2, data = PoliticalDemocracy[-idx, ], meanstructure = TRUE, warn = FALSE)
    yhat2[idx, ] = predicty.lavaan(fit, newdata = PoliticalDemocracy[idx, ], xnames = xnames, yn
    coefs = get.coef(fit, xnames, ynames)
    coefs2[, t] = rbind(matrix(coefs$alpha, 4, 1), matrix(coefs$gamma, 28, 1))

    # approach 3
    fit <- sem(model3, data = PoliticalDemocracy[-idx, ], meanstructure = TRUE, warn = FALSE)
    yhat3[idx, ] = predicty.lavaan(fit, newdata = PoliticalDemocracy[idx, ], xnames = xnames, yn
    coefs = get.coef(fit, xnames, ynames)
    coefs3[, t] = rbind(matrix(coefs$alpha, 4, 1), matrix(coefs$gamma, 28, 1))

    # approach 4
    fit <- sem(model4, data = PoliticalDemocracy[-idx, ], meanstructure = TRUE, warn = FALSE)
    yhat4[idx, ] = predicty.lavaan(fit, newdata = PoliticalDemocracy[idx, ], xnames = xnames, yn
    coefs = get.coef(fit, xnames, ynames)
    coefs4[, t] = rbind(matrix(coefs$alpha, 4, 1), matrix(coefs$gamma, 28, 1))


    # linear regression model
    fit = lm(cbind(y5,y6,y7,y8) ~ ., data = PoliticalDemocracy[-idx, ])
    yhat5[idx, ]= predict(fit, newdata = PoliticalDemocracy[idx, ])
    coefs5[, t] = matrix(t(coef(fit)), 32, 1)
  }# end folds

  pe1 = sqrt(sum((PoliticalDemocracy[, ynames] - yhat1)^2)/300)
  pe2 = sqrt(sum((PoliticalDemocracy[, ynames] - yhat2)^2)/300)
  pe3 = sqrt(sum((PoliticalDemocracy[, ynames] - yhat3)^2)/300)
  pe4 = sqrt(sum((PoliticalDemocracy[, ynames] - yhat4)^2)/300)
  pe5 = sqrt(sum((PoliticalDemocracy[, ynames] - yhat5)^2)/300)
  PE$pe[((r-1)*5 + 1): (r*5)] = c(pe1, pe2, pe3, pe4, pe5)
} # end repetitions

save(PE, file = "xvalpoldem.Rdata")
save(coefs1, coefs2, coefs3, coefs4, coefs5, file = "xvalpoldemcoefs.Rdata")
```

We can make prediction error boxplots for the different approaches

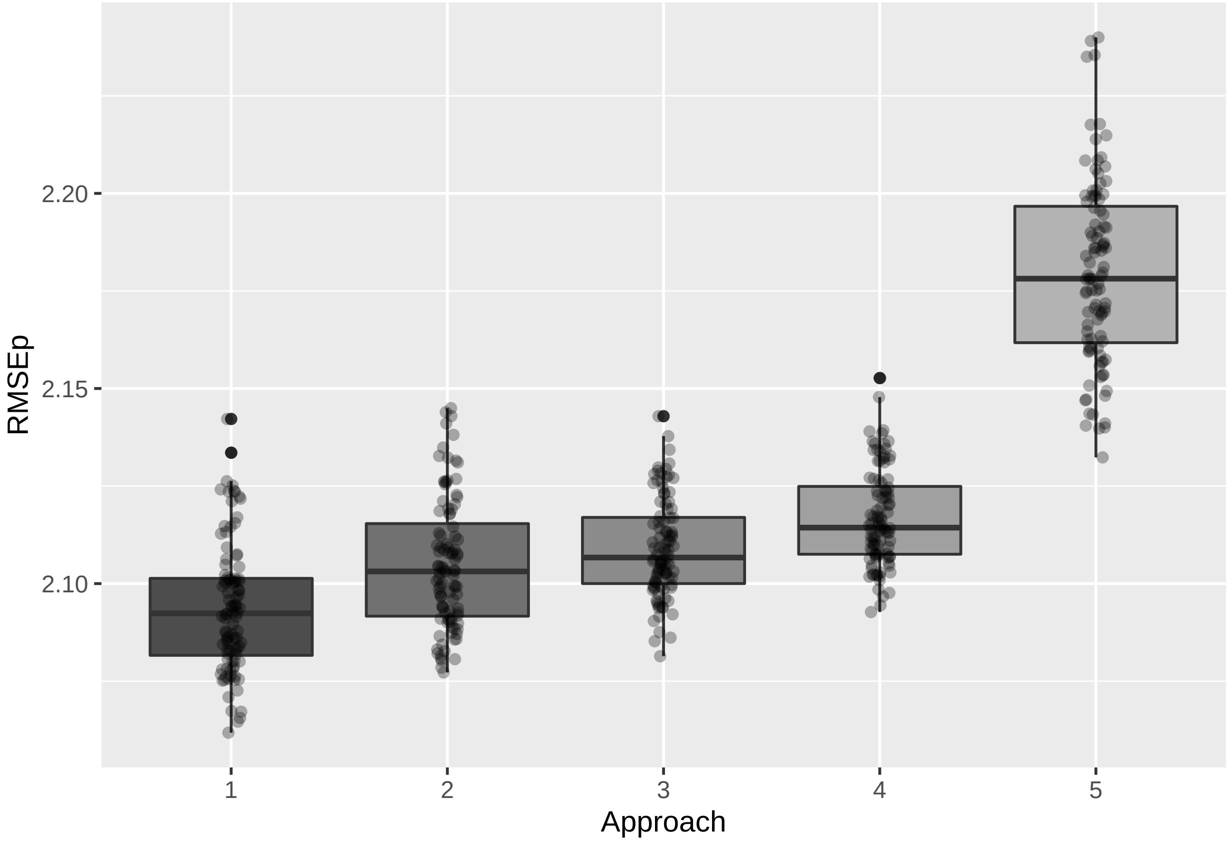
```
library(ggplot2)
PE$model = as.factor(PE$model)



p <- ggplot(PE, aes(x=model, y=pe, fill=factor(model))) +
```

```
    geom_boxplot(aes(group = factor(model))) +
    geom_jitter(width = 0.05, height = 0, colour = rgb(0,0,0,.3)) +
    xlab("Approach") + ylab("RMSEp") +
    theme(legend.position="none") +
    scale_fill_grey(start=.3,end=.7)

p
```




```
ggsave('~/surfdrive/Predictive-Psychometrics/paper/SEM-Predictive Validity/versie2/Figures/Polde
```

```
## Saving 6.5 x 4.5 in image
```

We can check the number of wins for each of the five approaches:

```
pe = cbind(PE[PE$model == 1, 3], PE[PE$model == 2, 3], PE[PE$model == 3, 3], PE[PE$model == 4, 3
table(apply(pe, 1, which.min))
```

```
##
##  1  2  3
## 95  1  4
```

Finally, we can take a look at the estimated coefficients in each of the 10 x 100 fitted models:

```
coefnames = c("a5", "a6", "a7", "a8", "gy15", "gy25", "gy35", "gy45", "gx15", "gx25", "gx35", "g
rownames(coefs1) = coefnames
rownames(coefs2) = coefnames
```

```r
rownames(coefs3) = coefnames
rownames(coefs4) = coefnames
rownames(coefs5) = coefnames
meancoefs = cbind(apply(coefs1, 1, mean),
      apply(coefs2, 1, mean),
      apply(coefs3, 1, mean),
      apply(coefs4, 1, mean),
      apply(coefs5, 1, mean))

meancoefs
```

```
##              [,1]        [,2]        [,3]        [,4]         [,5]
## a5   -0.71787763  0.43206489  0.05214320  0.96009565 -3.378331480
## a6   -3.45879811 -2.24737169 -3.56131864 -2.46259033 -1.289353598
## a7   -0.52350265  0.66671944  0.22536726  1.22231292  1.847380234
## a8   -2.80761877 -1.57159995 -2.55916851 -1.47871935 -2.042856150
## gy15  0.49634565  0.47183735  0.24769784  0.24810546  0.458637148
## gy25  0.03136181  0.03697160  0.09235532  0.09698387  0.448450214
## gy35  0.08248057  0.10274457  0.10517022  0.11148867  0.303232115
## gy45  0.11997414  0.16114393  0.21175985  0.23515372  0.321837989
## gx15  0.14531241  0.04417148  0.12692298  0.04764958  0.082806084
## gx25  0.21576626  0.06324066  0.19195322  0.06986930  0.358153451
## gx35  0.04580931  0.01373342  0.04014653  0.01494091  0.110493340
## gy16  0.27807842  0.29636621  0.31816194  0.32275421  0.104281912
## gy26  0.36300329  0.35861708  0.11914039  0.12667602  0.055992290
## gy36  0.12212555  0.14214866  0.13517890  0.14512740 -0.109021374
## gy46  0.03825125  0.07835195  0.27282419  0.30678258  0.274326007
## gx16  0.18048939  0.06120482  0.16297257  0.06194380  0.002726899
## gx26  0.26827460  0.08777686  0.24696066  0.09103282  0.038440971
## gx36  0.05701242  0.01910076  0.05163351  0.01945699  0.133400053
## gy17  0.28410679  0.29979439  0.29082017  0.29536754  0.203175268
## gy27  0.04779691  0.05202714  0.10854223  0.11555251  0.375505553
## gy37  0.26793915  0.28204027  0.12362848  0.13288987  0.876484016
## gy47  0.18365385  0.22751529  0.24896162  0.28029187 -0.239668292
## gx17  0.17975525  0.06199888  0.14896020  0.05670912 -0.538281761
## gx27  0.26666316  0.08854129  0.22528577  0.08314262  0.144611474
## gx37  0.05669985  0.01931485  0.04712732  0.01778491  0.185984787
## gy18  0.32048957  0.33667231  0.32126938  0.32759209  0.108110906
## gy28  0.03169588  0.03707115  0.12014400  0.12839997  0.282616372
## gy38  0.14115172  0.16182243  0.13650518  0.14730609  0.405538683
## gy48  0.32774574  0.36950286  0.27559068  0.31145094 -0.057846800
## gx18  0.19401717  0.06991388  0.16457721  0.06292445  0.311336864
## gx28  0.28792451  0.09985473  0.24924877  0.09239941  0.248501056
## gx38  0.06127479  0.02181331  0.05213539  0.01976213 -0.135068578
```

```r
stdcoefs = cbind(apply(coefs1, 1, sd),
      apply(coefs2, 1, sd),
      apply(coefs3, 1, sd),
```

```
      apply(coefs4, 1, sd),
      apply(coefs5, 1, sd))

stdcoefs
```

```
##             [,1]        [,2]        [,3]        [,4]       [,5]
## a5   0.247673074 0.180736209 0.279281754 0.204307773 0.59792299
## a6   0.282581924 0.227522572 0.275741969 0.191050200 1.09651584
## a7   0.307754233 0.229710948 0.317116537 0.247859586 0.91350786
## a8   0.267997852 0.231411929 0.252585805 0.190123280 0.88568227
## gy15 0.038537331 0.038377621 0.022986530 0.023186666 0.04639528
## gy25 0.006031673 0.006701559 0.008169283 0.008137232 0.05190423
## gy35 0.015282370 0.016665650 0.009249255 0.010030607 0.05701786
## gy45 0.015028390 0.015982607 0.013971515 0.014317463 0.05076422
## gx15 0.033095514 0.012791646 0.029548536 0.012105450 0.02658933
## gx25 0.033409939 0.013643651 0.028842058 0.011044942 0.04958269
## gx35 0.009673730 0.003584624 0.008572048 0.003439419 0.03016637
## gy16 0.032034255 0.032030773 0.022756419 0.023128409 0.03642152
## gy26 0.048483655 0.047424933 0.013640593 0.013720921 0.03582448
## gy36 0.010948888 0.012463004 0.010034551 0.011032890 0.03323860
## gy46 0.036828018 0.037102326 0.022610187 0.023449896 0.04514351
## gx16 0.038600611 0.015912489 0.036175708 0.015006845 0.04096285
## gx26 0.038887228 0.016470339 0.036876574 0.014305082 0.03431035
## gx36 0.011854622 0.004748091 0.010877325 0.004409331 0.05482957
## gy17 0.033664225 0.034075058 0.024836297 0.025160873 0.04874305
## gy27 0.007928024 0.008544703 0.010068948 0.009892770 0.05516196
## gy37 0.040042581 0.040076035 0.011715909 0.012794672 0.18675158
## gy47 0.023432885 0.024267747 0.018997813 0.019427604 0.35009908
## gx17 0.039032403 0.016578681 0.033896555 0.014150483 0.29500889
## gx27 0.035880973 0.015401440 0.032252093 0.012462801 0.27693674
## gx37 0.011526408 0.004782803 0.009851677 0.004024715 0.09203958
## gy18 0.028822481 0.028698876 0.022691845 0.023230421 0.13343539
## gy28 0.017613548 0.017843688 0.012123468 0.012038342 0.15430964
## gy38 0.013218303 0.014194104 0.010102318 0.011147132 0.14121811
## gy48 0.044547117 0.043999979 0.023918451 0.024607255 0.08550219
## gx18 0.041805082 0.019148482 0.036566755 0.015467321 0.11926599
## gx28 0.038961759 0.018292743 0.036223460 0.014462581 0.13180099
## gx38 0.012751577 0.005677270 0.010962987 0.004541299 0.11134679
```