PCA with Na replaced by averages

```
Dorg = [5   1   1   3;
     3   2   4   1;
     NaN 4   5   5;
     1   1   1   1;
     3   4   NaN NaN;
     5   1   1   1;
     4   2   NaN NaN;
     3   3   3   2;
     NaN NaN NaN NaN;
     1   1   3   4;
     4   2   NaN NaN;
     NaN 4   3   3];

nu=size(Dorg,1);
na=size(Dorg,2);
```

Get number of users (raters) and number of artists

```
Dmean = linspace(1,1,nu)'*mean(Dorg,'omitnan');
Dpca=Dmean.*(isnan(Dorg))+fillmissing(Dorg.*(not(isnan(Dorg))),'constant',0);

Dpca
```

```
Dpca = 12×4
    5.0000    1.0000    1.0000    3.0000
    3.0000    2.0000    4.0000    1.0000
    3.2222    4.0000    5.0000    5.0000
    1.0000    1.0000    1.0000    1.0000
    3.0000    4.0000    2.6250    2.5000
    5.0000    1.0000    1.0000    1.0000
    4.0000    2.0000    2.6250    2.5000
    3.0000    3.0000    3.0000    2.0000
    3.2222    2.2727    2.6250    2.5000
    1.0000    1.0000    3.0000    4.0000
      ⋮
```

Run PCA

```
[coeff,score,latent, tsq, expl, mu]= pca(Dpca,'algorithm','als','centered','on','NumComponents'
coeff
```

```
coeff = 4×4
   -0.1672    0.9645    0.0772    0.1894
    0.5642    0.2532   -0.5396   -0.5713
    0.6170   -0.0291   -0.1793    0.7658
    0.5226    0.0695    0.8190   -0.2267
```

```
score
```

```
score = 12×4
   -1.7566    1.4744    1.5249   -0.2938
   -0.0522   -0.4276   -1.3450    1.5066
    3.7462    0.5420    0.6897    0.2652
   -2.1330   -2.5225   -0.4221   -0.5982
    1.0117    0.2230   -0.9492   -1.0289
   -2.8017    1.3354   -0.1131    0.1595
```

```
   -0.2839    0.6811    0.2072    0.3031
    0.4176   -0.0758   -0.8863   -0.0571
   -0.0000    0.0000    0.0000   -0.0000
    0.6686   -2.3722    1.6764    0.2534
      ⋮
      ⋮
```

```
latent
```

```
latent = 4×1
    3.0437
    1.6018
    0.8847
    0.4346
```

Generate lower dimensional versions of the scores

```
score1=horzcat(score(:,1:1),zeros(nu,3));
score2=horzcat(score(:,1:2),zeros(nu,2));
score3=horzcat(score(:,1:3),zeros(nu,1));
score4=score;
```

Generate predictions

Predictions are taken by multiplying the user specific score vectors with the artist specifc coeff vector. For example for the one-factor model, we consider only the first dimension of the user specific score1 vector multiplied by the corresponding first dimension of the coeff vector.

Since we multiplying a column vector (12x1) with a row vector (1x4) we generate a full (12x4) predicted ratings matrix.

Rating_matrix = user_score_colvec * artist_coeff_rowvec + average_ratings_matrix

= [-1.7;-.05;3.7;-2.1;1.;-2.8 ...] * [-.16 .56 .61 .52] + [3.2 2.2 2.6 2.5; 3.2 2.2 ...]

For higher order (2 factor - 4 factor) models we include more dimensions of the user scores and artist coeffs.

```
Dpred1= score1*coeff'+repmat(mu,nu,1)
```

```
Dpred1 = 12×4
    3.5159    1.2816    1.5413    1.5821
    3.2310    2.2433    2.5928    2.4727
    2.5960    4.3864    4.9363    4.4576
    3.5788    1.0692    1.3090    1.3854
    3.0531    2.8436    3.2492    3.0287
    3.6906    0.6919    0.8965    1.0360
    3.2697    2.1125    2.4498    2.3516
    3.1524    2.5083    2.8826    2.7182
    3.2222    2.2727    2.6250    2.5000
    3.1105    2.6500    3.0375    2.8494
      ⋮
      ⋮
```

```
repmat(mu,nu,1) 3.2 2.2 2.6 2.5;3.2
```

```
ans = 12×4
    3.2222    2.2727    2.6250    2.5000
    3.2222    2.2727    2.6250    2.5000
    3.2222    2.2727    2.6250    2.5000
```

```
    3.2222    2.2727    2.6250    2.5000
    3.2222    2.2727    2.6250    2.5000
    3.2222    2.2727    2.6250    2.5000
    3.2222    2.2727    2.6250    2.5000
    3.2222    2.2727    2.6250    2.5000
    3.2222    2.2727    2.6250    2.5000
    3.2222    2.2727    2.6250    2.5000
     ⋮
     ⋮
```

```
Dpred2= score2*coeff'+repmat(mu,nu,1)
```

```
Dpred2 = 12×4
    4.9379    1.6549    1.4984    1.6846
    2.8185    2.1350    2.6052    2.4430
    3.1187    4.5237    4.9205    4.4953
    1.1459    0.4305    1.3824    1.2101
    3.2682    2.9000    3.2427    3.0442
    4.9785    1.0301    0.8576    1.1288
    3.9266    2.2850    2.4300    2.3990
    3.0793    2.4891    2.8848    2.7129
    3.2222    2.2727    2.6250    2.5000
    0.8225    2.0493    3.1065    2.6845
     ⋮
     ⋮
```

```
Dpred3= score3*coeff'+repmat(mu,nu,1)
```

```
Dpred3 = 12×4
    5.0557    0.8321    1.2250    2.9334
    2.7146    2.8607    2.8463    1.3415
    3.1720    4.1515    4.7969    5.0601
    1.1133    0.6583    1.4581    0.8644
    3.1949    3.4122    3.4129    2.2668
    4.9698    1.0911    0.8779    1.0361
    3.9426    2.1732    2.3929    2.5687
    3.0108    2.9674    3.0437    1.9871
    3.2222    2.2727    2.6250    2.5000
    0.9520    1.1447    2.8060    4.0574
     ⋮
     ⋮
```

```
Dpred4= score4*coeff'+repmat(mu,nu,1)
```

```
Dpred4 = 12×4
    5.0000    1.0000    1.0000    3.0000
    3.0000    2.0000    4.0000    1.0000
    3.2222    4.0000    5.0000    5.0000
    1.0000    1.0000    1.0000    1.0000
    3.0000    4.0000    2.6250    2.5000
    5.0000    1.0000    1.0000    1.0000
    4.0000    2.0000    2.6250    2.5000
    3.0000    3.0000    3.0000    2.0000
    3.2222    2.2727    2.6250    2.5000
    1.0000    1.0000    3.0000    4.0000
     ⋮
     ⋮
```

Average Rating Error

```
err1=sqrt(sum(fillmissing((Dpred1-Dorg),"constant",0).^2,"all")/(nu*na))
```

```
err1 = 0.8024
```

```
err2=sqrt(sum(fillmissing((Dpred2-Dorg),"constant",0).^2,"all")/(nu*na))
```

```
err2 = 0.5340
```

```
err3=sqrt(sum(fillmissing((Dpred3-Dorg),"constant",0).^2,"all")/(nu*na))
```

```
err3 = 0.2873
```

```
err4=sqrt(sum(fillmissing((Dpred4-Dorg),"constant",0).^2,"all")/(nu*na))
```

```
err4 = 1.6831e-15
```

For each Artist build a Regression Tree from the first two Score Dimension and Known (not Nan) Target Ratings

First prepare new xydata table with NaN entry rows removed

```
xydata_art1 = rmmissing(horzcat(score,Dorg(:,1)));
xydata_art2 = rmmissing(horzcat(score,Dorg(:,2)));
xydata_art3 = rmmissing(horzcat(score,Dorg(:,3)));
xydata_art4 = rmmissing(horzcat(score,Dorg(:,4)));
```

Build a tree model for each artist rating based on two PCA factors

```
rtree_1= fitrtree(xydata_art1(:,1:2),xydata_art1(:,end),"MinParentSize",4);
rtree_2= fitrtree(xydata_art2(:,1:2),xydata_art2(:,end),"MinParentSize",4);
rtree_3= fitrtree(xydata_art3(:,1:2),xydata_art3(:,end),"MinParentSize",4);
rtree_4= fitrtree(xydata_art4(:,1:2),xydata_art4(:,end),"MinParentSize",4);
```

Predict rating for each artist

```
pred_1 = predict(rtree_1, score(:,1:2));
pred_2 = predict(rtree_2, score(:,1:2));
pred_3 = predict(rtree_3, score(:,1:2));
pred_4 = predict(rtree_4, score(:,1:2));
```

Combine rating into table

```
pred = horzcat(pred_1, pred_2, pred_3, pred_4)
```

```
pred = 12×4
    5.0000    1.0000    1.0000    3.0000
    3.0000    2.0000    4.0000    1.0000
    3.0000    4.0000    5.0000    4.0000
    1.0000    1.0000    1.0000    1.0000
    3.0000    4.0000    3.0000    4.0000
    5.0000    1.0000    1.0000    1.0000
    4.0000    2.0000    4.0000    1.0000
    3.0000    3.0000    3.0000    2.0000
    3.0000    2.0000    4.0000    1.0000
    1.0000    1.0000    3.0000    4.0000
     ⋮
```

```
err=sqrt(sum(fillmissing((pred-Dorg),"constant",0).^2,"all")/(nu*na))
```

```
err = 0.2041
```

```
view(rtree_1)
```

```
Decision tree for regression
1  if x2<-1.39992 then node 2 elseif x2>=-1.39992 then node 3 else 3.22222
2  fit = 1
3  if x1<-0.168067 then node 4 elseif x1>=-0.168067 then node 5 else 3.85714
4  if x1<-1.02024 then node 6 elseif x1>=-1.02024 then node 7 else 4.5
5  fit = 3
6  fit = 5
7  fit = 4
```

```
view(rtree_2)
```

```
Decision tree for regression
1  if x1<0.840145 then node 2 elseif x1>=0.840145 then node 3 else 2.27273
2  if x1<-1.02024 then node 4 elseif x1>=-1.02024 then node 5 else 1.625
3  fit = 4
4  fit = 1
5  if x1<0.543075 then node 6 elseif x1>=0.543075 then node 7 else 2
6  if x1<0.182673 then node 8 elseif x1>=0.182673 then node 9 else 2.25
7  fit = 1
8  fit = 2
9  fit = 3
```

```
view(rtree_3)
```

```
Decision tree for regression
1  if x1<-0.904409 then node 2 elseif x1>=-0.904409 then node 3 else 2.625
2  fit = 1
3  if x1<2.60672 then node 4 elseif x1>=2.60672 then node 5 else 3.6
4  if x1<0.182673 then node 6 elseif x1>=0.182673 then node 7 else 3.25
5  fit = 5
6  fit = 4
7  fit = 3
```

```
view(rtree_4)
```

```
Decision tree for regression
1  if x1<0.543075 then node 2 elseif x1>=0.543075 then node 3 else 2.5
2  if x2<1.40489 then node 4 elseif x2>=1.40489 then node 5 else 1.6
3  fit = 4
4  if x1<0.182673 then node 6 elseif x1>=0.182673 then node 7 else 1.25
5  fit = 3
6  fit = 1
7  fit = 2
```

```
```