Tutorial:
Mean and Covariance Estimation with Missing Data

Prof. Robert J. Frey, Research Professor
Director, Program in Quantitative Finance
Applied Mathematics and Statistics
Stony Brook University

12 March 2015 (01)

The statistical estimation with missing data is a frequent problem faced by a financial analyst or researcher. This tutorial gives an overview of the techniques used in such problems through a simple example, viz., estimating the mean and covariance in the presence of missing data. As we work our way through the various concepts, we also demonstrate them via an example, embedding the working code in this Mathematica notebook.

The approach is deliberately informal with its objective to develop the student's intuition. The expectation is that he or she will thereafter be better able to understand and apply more advanced material. Complete solutions both using a *Mathematica* package developed by the author and MATLAB’s Financial Toolbox are executed and contrasted. A brief list of further readings with guidance is provided.

The Problem of Missing Data

Financial data involving multiple instruments frequently have missing values. Markets have different holidays or special periods when they were closed, *e.g.*, the New York markets post the 9/11 attacks. Securities have different periods during which they were active, creating “holes” in one time series relative another. Sometimes the data are simply not in the databases to which we have access. We will use the estimation of the mean and covariance as our archetypal problem.

If the number of missing data is not too large, a reasonable strategy is to drop the incomplete observations and get on with the estimation. Dealing with missing data requires us to make assumptions about the relationships among the components of an observation in order to infer the effects of missing data and often a “light hand” in dealing with data is the best approach. Unfortunately, we are often faced with instances in which the missing values will cause one to reject too many observations, throwing away perfectly useful information contained in the non-missing data within an observation.

A Simple Example

Consider the dataset below. Period 2 of security A, 10 of B, and 1 to 2 of C are missing. A quarter of our observations suffer from missing values. Those observations, though defective, contain *some* valid data. We want an approach to produce a reasonable estimate of the mean and covariance without throwing away the partial information in those defective observations.

We start with our working data with missing values using *Mathematica*’s standard Missing[]:

```
data = {
   {0.29882113381252573`, 0.13903821287345947`, Missing[]},
   {Missing[], 0.1777488052132532`, Missing[]},
   {0.16330221665216665`, -0.02493188254168839`, 0.15524566961797698`},
   {-0.03966328185565937`, 0.01761940471121591`, -0.04591426120657219`},
   {0.18427734057801887`, 0.0263177006187041`, -0.02986489586501037`},
   {0.3259976411483706`, 0.10470303902414277`, -0.08987806643420446`},
   {0.45545036866113553`, 0.18344791651424108`, 0.2101491321368833`},
   {0.19616270889741735`, 0.09160598543246382`, 0.20625564617936631`},
   {0.2919227964278637`, 0.21771812726125142`, 0.1814539005984894`},
   {0.37546677450207255`, Missing[], 0.08803547665646887`},
   {0.46293772750580564`, 0.2142960469280747`, 0.24584741899047644`},
   {0.03752699767762904`, 0.12522428724750828`, -0.02157123637991265`}
  };
```

We can use the standard Mean[ ] built into *Mathematica* but the Variance[ ] and Covariance[ ] functions produce unbiased estimates; *i.e.*, if N is the sample size, they compute expectations for variances and covariances with $N - 1$ and $N - 2$, respectively. We wish to use the maximum likelihood estimates (MLE) for the mean and covariance. Thus, we define these two functions

```
f = {x, y} ↦ x^y

Function[{x, y}, x^y]

f[3, 2]

9

varMLE = x ↦ Mean[x^2] - Mean[x]^2;
covMLE = x ↦ x^T.x / Length[x] - (KroneckerProduct[#, #] &) [Mean[x]];
```

Note that the notation "*f* = {*x*, *y*} ↦ *expr*" (where "↦" is entered as "⎋fn⎋") is equivalent to "*f* = Function[{*x*, *y*}, *expr*]". The function *f* would then be called using "*f*[*x*, *y*]".

We also define a function to print out a table of various versions of our dataset:

```
tbl = x ↦ TableForm[x, TableHeadings → {Automatic, {"A", "B", "C"}}];
```

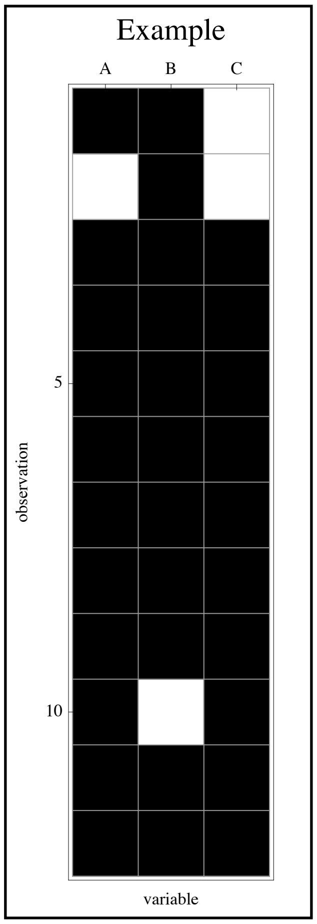
Our beginning data are below. *Mathematica* uses an expression with head "Missing". Frequently, an optional argument (usually a string) describing the reason for the missing point is added, *e.g.*, Missing["NotAvailable] or Missing["MarketClosed"].

```
tbl[data]

   | A           B            C
---+-------------------------------------
1  | 0.298821    0.139038     Missing[]
2  | Missing[]   0.177749     Missing[]
3  | 0.163302    -0.0249319   0.155246
4  | -0.0396633  0.0176194    -0.0459143
5  | 0.184277    0.0263177    -0.0298649
6  | 0.325998    0.104703     -0.0898781
7  | 0.45545     0.183448     0.210149
8  | 0.196163    0.091606     0.206256
9  | 0.291923    0.217718     0.181454
10 | 0.375467    Missing[]    0.0880355
11 | 0.462938    0.214296     0.245847
12 | 0.037527    0.125224     -0.0215712
```

ArrayPlot[ ] can be used to provide a visual signature that gives insight into the pattern of missing values. This approach works quite well even for large datasets.

```
Framed@ArrayPlot[
  Map[Boole@FreeQ[#, _Missing] &, data, {2}],
  ImageSize → 150,
  Mesh -> True,
  Frame → True,
  FrameLabel → {{"variable", Style["Example", FontSize → 16]}, {"observation", ""}},
  FrameTicks → {{Automatic, None}, {None, {{1, "A"}, {2, "B"}, {3, "C"}}}}
 ]
```



“Obvious” Workarounds

We first look at a few “obvious” but ultimately unsatisfactory approaches to dealing with missing data.

Drop Observations with Missing Values

The most obvious way to deal with missing data is to drop any observations that have missing values. It is useful to have a select valid function, selectVld[ ], which drops elements containing missing values from a list

```
selectVld = x ↦ Select[x, FreeQ[#, _Missing] &];
```

The data of only complete observations are

```
dropped = selectVld[data];
tbl[dropped]
```

```
   | A            B            C
---+-------------------------------------
1  | 0.163302     -0.0249319   0.155246
2  | -0.0396633   0.0176194    -0.0459143
3  | 0.184277     0.0263177    -0.0298649
4  | 0.325998     0.104703     -0.0898781
5  | 0.45545      0.183448     0.210149
6  | 0.196163     0.091606     0.206256
7  | 0.291923     0.217718     0.181454
8  | 0.462938     0.214296     0.245847
9  | 0.037527     0.125224     -0.0215712
```

The mean and covariance are

```
meanDropped = Mean[dropped];
MatrixForm[meanDropped]
covDropped = covMLE[dropped];
MatrixForm[covDropped]
```

$$\begin{pmatrix} 0.230879 \\ 0.106222 \\ 0.0901915 \end{pmatrix}$$

$$\begin{pmatrix} 0.0261755 & 0.00916384 & 0.0119005 \\ 0.00916384 & 0.00689875 & 0.0050928 \\ 0.0119005 & 0.0050928 & 0.0158338 \end{pmatrix}$$

Replace Missing Values with the Mean of the Non-Missing Values

A common approach that was used before the development of more powerful methods is to replace each missing value with the mean of the non-missing data. This is not unreasonable; *in the absence of other information* the mean is our best guess.

We can define a mean replace function, meanRpl[ ], which replaces missing values in a vector with the mean of the non-missing entries:

```
meanRpl = x ↦ x /. _Missing → Mean[selectVld[x]];
```

We then map this function column-wise to our data by transposing, mapping, and transposing back:

```
replaced = Transpose[meanRpl /@ Transpose[data]];
tbl[replaced]
```

```
    | A            B            C
----+-------------------------------------
1   | 0.298821     0.139038     0.0899759
2   | 0.2502       0.177749     0.0899759
3   | 0.163302     -0.0249319   0.155246
4   | -0.0396633   0.0176194    -0.0459143
5   | 0.184277     0.0263177    -0.0298649
6   | 0.325998     0.104703     -0.0898781
7   | 0.45545      0.183448     0.210149
8   | 0.196163     0.091606     0.206256
9   | 0.291923     0.217718     0.181454
10  | 0.375467     0.115708     0.0880355
11  | 0.462938     0.214296     0.245847
12  | 0.037527     0.125224     -0.0215712
```

```
meanReplaced = Mean[replaced];
MatrixForm[meanReplaced]
covReplaced = covMLE[replaced];
MatrixForm[covReplaced]
```

```
( 0.2502    )
( 0.115708  )
( 0.0899759 )
```

```
( 0.0214162   0.00710486  0.00890201 )
( 0.00710486  0.00560766  0.00381807 )
( 0.00890201  0.00381807  0.0118757  )
```

Apply All Available Cross-Products

Another—perhaps at first seemingly more promising—approach to improve the covariance estimate is to recognize that each entry in the covariance matrix is the paired covariance of each variable represented in the observation vector. Thus, to produce a given $\sigma_{i,j}$ we need only drop entries with missing values in the respective two variables and can ignore whether there are missing values in the others. Unfortunately, this approach can cause gross distortions in the covariance; in fact, the matrix is no longer even guaranteed to be positive semi-definite.

The estimate can be demonstrated with a simple for loop. We avoid unnecessary computation but make no further attempt to tighten up this code: This is not the ideal coding approach, but we are trying to keep things as accessible as possible in the tutorial.

```
covAvailable = Array[0. &, {#, #} &[Last@Dimensions[data]]];
For[i = 1, i ≤ Last@Dimensions[data], i++,
  covAvailable[[i, i]] = varMLE[selectVld[data[[All, i]]]];
  For[j = 1, j < i, j++,
   covAvailable[[i, j]] =
    covAvailable[[j, i]] = (Mean[#1 #2] - Mean[#1] × Mean[#2] &) @@
      (selectVld[data[[All, {i, j}]]]ᵀ)
  ]
 ];
MatrixForm[covAvailable]
```

```
( 0.0233631   0.00844812  0.0106824 )
( 0.00844812  0.00611744  0.0050928 )
( 0.0106824   0.0050928   0.0142509 )
```

Conditional Means by Linear Regression

Replacing a missing value with the mean is a reasonable guess in the absence of other evidence. We do, however, have other evidence: the other values in the observation. If the variables in the observation are correlated, they can be used to estimate a conditional mean for the missing value using linear regression. This can be a large improvement when the correlations are significant and no worse than mean replacement when they are uncorrelated.

Example Continued — Using Linear Regression

If we wish to estimate missing values for security C, then we can use the eight complete observations to fit a regression model and then apply using the values of A and B in the first two observations to estimate the missing values of C. Then we do the same for A and C to predict B, and B and C to predict A.

If we were designing general purpose software we would, of course, build functions to keep track of where the missing values were and what predictors to construct for each one depending upon the available data in each respective observations. Here we execute our code step-by-step to make the logic clear.

We begin by setting up a “regressed” matrix based on “data” into which we will replace missing data with their conditional means, *i.e.*, the predictions.

```
regressed = data;
```

We next perform a series of regressions and predictions to resolve each missing data element with its conditional mean. Note that the pattern of missing values in the specific variables depends upon which are included in the regression, and, therefore, we have to perform a selectVld[ ] on each case individually.

First, to predict the missing value in the first position of *C* (in ⟦1, 3⟧), we have both *A* (in ⟦1, 1⟧) and *B* (in ⟦1, 2⟧) available. Our dataset for LinearModelFit[ ] is selectVld[data]. The dependent variable is regressed⟦1, 3⟧ and is predicted by the independent variables {*A*, *B*} in regressed⟦1, {1, 2}⟧.

```
lmABtoC = LinearModelFit[selectVld[data], {a, b}, {a, b}];
regressed⟦1, 3⟧ = lmABtoC @@ regressed⟦1, {1, 2}⟧;
```

Then, to predict the missing value in the second position of *C* (in ⟦2, 3⟧), we see that *A* is also missing and, therefore, can use only *B* (in ⟦2, 2⟧). Our dataset for LinearModelFit[ ] is selectVld[data⟦All, {2, 3}⟧]. The dependent variable is regressed⟦2, 3⟧ and is predicted by the independent variable in regressed⟦2, 2⟧.

```
lmBtoC = LinearModelFit[selectVld[data]⟦All, {2, 3}⟧, {b}, {b}];
regressed⟦2, 3⟧ = lmBtoC @@ regressed⟦2, 2⟧;
```

Then, to predict the missing value in the second position of *A* (in ⟦2, 1⟧), we see that *C* is also missing and, therefore, must use only *B* (in ⟦2, 2⟧). Our dataset for LinearModelFit[ ] is selectVld[data⟦All, {2, 1}⟧]. The dependent variable is regressed⟦2, 1⟧ and is predicted by the independent variable in regressed⟦2, 2⟧.

```
lmBtoA = LinearModelFit[selectVld[data]⟦All, {2, 1}⟧, {b}, {b}];
regressed⟦2, 1⟧ = lmBtoA @@ regressed⟦2, 2⟧;
```

Finally, to predict the missing value in the tenth position of *B* (in ⟦10, 2⟧), we have both *A* (in ⟦10, 1⟧) and C (in ⟦10, 3⟧) available. Our dataset for LinearModelFit[ ] is selectVld[data⟦All, {1, 3, 2}⟧]. The dependent variable is regressed⟦10, 2⟧ and is predicted by the independent variables in regressed⟦10, {1, 3}⟧.

```
lmACtoB = LinearModelFit[selectVld[data]⟦All, {1, 3, 2}⟧, {a, c}, {a, c}];
regressed⟦10, 2⟧ = lmACtoB @@ regressed⟦10, {1, 3}⟧;
```

Preliminary Results

The resulting completed dataset, mean vector and covariance matrix are

```
tbl[regressed]
```

	A	B	C
1	0.298821	0.139038	0.123348
2	0.177749	0.177749	0.177749
3	0.163302	-0.0249319	0.155246
4	-0.0396633	0.0176194	-0.0459143
5	0.184277	0.0263177	-0.0298649
6	0.325998	0.104703	-0.0898781
7	0.45545	0.183448	0.210149
8	0.196163	0.091606	0.206256
9	0.291923	0.217718	0.181454
10	0.375467	0.150807	0.0880355
11	0.462938	0.214296	0.245847
12	0.037527	0.125224	-0.0215712

```
meanRegressed = Mean[regressed];
MatrixForm[meanRegressed]
covRegressed = covMLE[regressed];
MatrixForm[covRegressed]
```

```
( 0.244163 )
( 0.118633 )
( 0.100071 )

(  0.0218172  0.00711433 0.00856824 )
( 0.00711433  0.00570176 0.00430154 )
( 0.00856824  0.00430154  0.0125086 )
```

Mean Comparisons

```
TableForm[MatrixForm /@ Mean /@ {dropped, replaced, regressed},
 TableHeadings → {{"Dropped Missing", "Mean Replaced", "Mean Regressed"}, None}]
```

```
                ( 0.230879  )
Dropped Missing ( 0.106222  )
                ( 0.0901915 )
                (  0.2502   )
Mean Replaced   ( 0.115708  )
                ( 0.0899759 )
                ( 0.244163 )
Mean Regressed  ( 0.118633 )
                ( 0.100071 )
```

```
TableForm[MatrixForm /@ covMLE /@ {dropped, replaced, regressed},
 TableHeadings → {{"Dropped Missing", "Mean Replaced", "Mean Regressed"}, None}]
```

```
                (  0.0261755  0.00916384 0.0119005  )
Dropped Missing ( 0.00916384  0.00689875 0.0050928  )
                (  0.0119005   0.0050928 0.0158338  )
                (  0.0214162  0.00710486 0.00890201 )
Mean Replaced   ( 0.00710486  0.00560766 0.00381807 )
                ( 0.00890201  0.00381807  0.0118757 )
                (  0.0218172  0.00711433 0.00856824 )
Mean Regressed  ( 0.00711433  0.00570176 0.00430154 )
                ( 0.00856824  0.00430154  0.0125086 )
```

It should be obvious to you that replacing missing values with predictions (*i.e.*, conditional means) estimated by appropriate linear regressions is a better approach than either dropping all incomplete observations or replacing missing values with the unconditional mean. *We are still missing something however.*

Second-Order Effects

When we replace a missing value with its conditional mean, we are using the *most likely* value of the missing element given the available data. In an actual realization, however, a missing element would have displayed its normal variation about this conditional mean. *That additional variation is missing from our estimations. This causes us to underestimate the variances.*

We do, fortunately, have a place to look for this missing variation: the error variance of the regression used. The covariance matrix is made up of the paired covariances on the off-diagonal elements and the variances along the main diagonal. We look now as to how we can identify where the missing variability manifests itself and how this deficiency can be resolved.

Variances and Covariances

The variance of a random variable X_j is the expected value of the squared deviation from the mean

$$\sigma_j^2 = \mathbb{E}\left[\left(X_j - \overline{X}_j\right)^2\right]$$

For purposes of computation the above can be simplified by expanding terms, distributing the expectation, and simplifying:

$$\sigma_j{}^2 = \mathbb{E}\left[X_j{}^2\right] - \mathbb{E}[\overline{X}_j]^2$$

As we have already noted, the mean's estimation presents us with no further problems, so our estimate of $\mathbb{E}[\overline{X}_j]^2$ is fine. For the squared term, however, we must add the missing variability, Let $\hat{X}_{k,j}$ be the conditional mean estimated by regression as above for the missing value in the k–th position of the j–th variable and $\hat{\epsilon}_{k,j}$ the associated error term from that regression. Noting that the predictions and the errors are uncorrelated and that the mean error is 0, we have

$$\mathbb{E}\left[X_{k,j}{}^2\right] = \mathbb{E}\left[\left(\hat{X}_{k,j} + \hat{\epsilon}_{k,j}\right)^2\right] = \hat{X}_{k,j}{}^2 + \mathrm{Var}[\hat{\epsilon}_{k,j}]$$

Thus, the squared sum of the values of the completed data's estimates must be augmented by the error variances of any missing values.

The entries of the covariance matrix are also measures of dispersion, *viz*,.

$$\sigma_{i,j} = \mathbb{E}\left[\left(X_i - \overline{X}_i\right)\left(X_j - \overline{X}_j\right)\right]$$

For purposes of computation the above can be simplified using the same techniques we did with variance yielding

$$\sigma_{i,j} = \mathbb{E}[X_i\, X_j] - \overline{X}_i\, \overline{X}_j$$

The estimated means, $\overline{X}_i$ and $\overline{X}_j$, as before, need no further work. Consider an observed value $X_{k,i}$ and predicted missing value $\hat{X}_{k,j}$. We observe that the errors are uncorrelated; hence, the expected value of the cross product for these two data is

$$\mathbb{E}[X_{k,i}\, X_{k,j}] = X_{k,i}\left(\hat{X}_{k,j} + \hat{\epsilon}_{k,j}\right) = X_{k,i}\, \hat{X}_{k,j}$$

The results are analogous if both i and j are missing. *Thus, we can construct such cross-products using either actual observations or conditional means without further adjustment.*

A Brief Aside

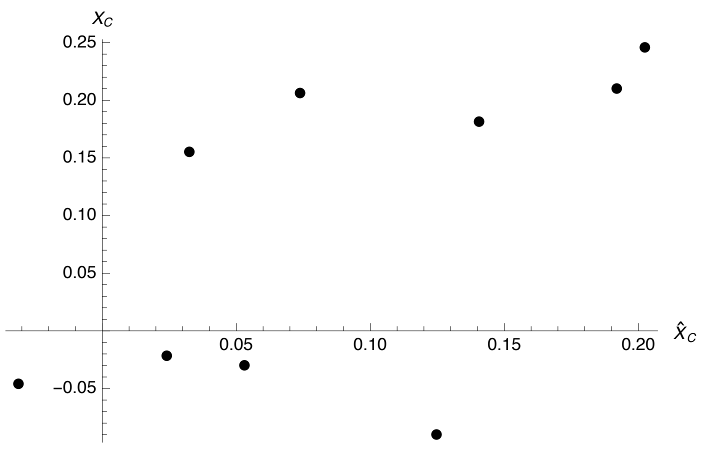
While the above is technically sufficient, this is a tutorial. We, therefore, go out of our way to make the student comfortable and conduct a brief demonstration. We will start with the set of all-valid observations. Then, we fit the regression $(A, B) \longmapsto C$. Finally, we will perform the expectations of variables and cross-products using both X_C and $\hat{X}_C$ to demonstrate our assertions above.

The first computations are to form the data, fit the model, and produce the conditional means.

```
{obsA, obsB, obsC} = selectVld[data]ᵀ;
lm = LinearModelFit[{obsA, obsB, obsC}ᵀ, {a, b}, {a, b}];
estC = lm @@ # & /@ ({obsA, obsB}ᵀ);
errC = lm["FitResiduals"];
```

A plot of the predicts *versus* observed shows the effect of noise.

```
ListPlot[{estC, obsC}ᵀ, AxesLabel → {"X̂C", "XC"}]
```



The first expectation we demonstrate is

$$\mathbb{E}[X_C] = \mathbb{E}[\hat{X}_C]$$

```
Mean[obsC] == Mean[estC]

True
```

The next expectation we demonstrate is the mean square value is equal to the mean square of the conditional mean and error variance:

$$\mathbb{E}\left[X_C^2\right] = \mathbb{E}[\hat{X}_C^{\,2}] + \mathrm{Var}[\hat{\epsilon}_C]$$

```
Mean[obsC^2] == Mean[estC^2] + varMLE[errC]

True
```

The final expectation is the product of the actual observations and the conditional means:

$$\mathbb{E}[X_A\, X_C] = \mathbb{E}\left[X_A\, \hat{X}_C\right]$$

```
Mean[obsA obsC] == Mean[obsA estC]

True
```

Example Continued — Fixing the Covariance

We now have what we need to fix our covariance estimate, covRegressed, which we see now has correctly estimated the covariances but has underestimated the variances.

```
MatrixForm[covRegressed]

( 0.0218172  0.00711433 0.00856824 )
( 0.00711433 0.00570176 0.00430154 )
( 0.00856824 0.00430154  0.0125086 )
```

We apply the necessary missing variable to the squared expectations and recompute a new covariance:

```
cross = regressed^T.regressed;
cross[[1, 1]] += varMLE[lmBtoA["FitResiduals"]];
cross[[2, 2]] += varMLE[lmACtoB["FitResiduals"]];
cross[[3, 3]] += varMLE[lmABtoC["FitResiduals"]] + varMLE[lmBtoC["FitResiduals"]];
covCompleted = cross / Length[regressed] - (KroneckerProduct[#, #] &[Mean[regressed]]);
MatrixForm[covCompleted]

( 0.0229841  0.00711433 0.00856824 )
( 0.00711433 0.00600244 0.00430154 )
( 0.00856824 0.00430154  0.014364  )
```

Are We Done?

While it may first seem that we have done everything we can, we next realize that we have more than an improved estimate of the mean and covariance. It turns out that the sums of variables and pairwise cross-products of variables represented by the mean and covariance contain precisely the information we need to perform a regression involving one variable predicted by any subset of the remaining variables. Thus, we can use the improved mean and covariance to repeat the regressions needed to estimate the first and second moments of missing variables which in turn allows us to compute a new covariance matrix.

We can repeat this cycle of missing value regressions followed by mean and covariance estimations until the process converges. If we do so in the right way, we can guarantee that the likelihood of the result will improve with each iteration. This is the approach embodied in the EM algorithm.

The EM Algorithm

The EM (Expectation-Maximization) Algorithm [Dempster *et al.*, 1977] is a powerful statistical estimation technique that can be used to compute the MLE (maximum likelihood estimation) of parameters when dealing with occult data; *i.e.*, a problem with missing, incomplete, or censored data or one that can be framed in that fashion). It is a formalization covering a range of intuitive *ad hoc* techniques that had been applied over many years by experienced data analysts.

As we have done elsewhere, we have relaxed formality for the sake of intuition. The EM algorithm did not appear in a single insight but was the synthesis and generalization of a number of creative and common sense approaches used by statisticians over the years to handle otherwise intractable problems. We are trying to capture that intuition in order to make the formal material more accessible to the student.

In the EM algorithm a problem is set up with the triple $(\mathbf{X}, \mathbf{Z}, \vartheta)$—the observed data $\mathbf{X}$, the occult (*i.e.*, missing, incomplete, or censored) data $\mathbf{Z}$, and the model parameters ϑ. The EM algorithm works by a series of successive expectation or E-steps and maximization or M-steps

In the E-step the observed data and an estimate of the parameters are used to produce values for the missing data

$$(\mathbf{X}, \vartheta) \longmapsto \mathbf{Z}$$

When we say that in the E-step we "produce values for the missing data", we mean "produce the sufficient statistics that we need to estimate ϑ".

The M-step produces a maximum likelihood estimate (MLE) of the parameters using the observed and estimated missing values

$$(\mathbf{X}, \mathbf{Z}) \longmapsto \vartheta$$

At each EM iteration the log likelihood function is computed and recorded.

$$\log \mathcal{L}(\vartheta \mid \mathbf{X}, \mathbf{Z})$$

When they are "sufficiently close" or when we have hit some predetermined number of maximum iterations, we stop and return the results; otherwise, we proceed to the next iteration.

Issues with Termination

When we use vague language such as "sufficiently close", we are unfortunately identifying some difficulties with the EM algorithm. There is no universal or obvious way to specify "sufficiently close". While the EM algorithm will increase the likelihood monotonically toward a maximum, that maximum may be a local one. It converges sub-linearly, *i.e.*, slowly. Finally, the likelihood can seem to be flattening out but then may suddenly begin to increase rapidly, behavior which can trick us into announcing convergence prematurely.

In many applications it is wise to set a very tight convergence criterion to avoid premature termination. If there is serious concern about multiple maxima, then several runs with different starting parameters is advisable. Often the strategy is dictated by your experience with the type of problem in question rather than any theoretical insight.

It is always useful to plot the growth of the log likelihood to gain insight into how the iterations are performing. If there is doubt that convergence has been achieved, then it is a simple matter to restart the algorithm with prior last estimate as the new initial estimate.

Issues with Patterns of Missing Values

In many applications of the EM algorithm and in particular to our problem here, certain assumptions must be made as to the pattern of missing values to avoid bias in the E-step. We are using observed values to estimate missing ones; therefore, the choice of which values are missing cannot be such that they introduce biases into the estimations. Normally, this means that we must that data are missing completely at random or MCAR.

That does not mean, of course, that there are no discernable patterns in the missing data. For example, for a given variable the missing values may all occur at the beginning because it represents a time series with a history shorter than the other variables. What is important, however, is that there is no special significance to those periods so that the regressions fit using the available periods embody the same underlying relationships among the variables.

Applying the EM Algorithm to Mean and Covariance

We can now outline how the EM algorithm is applied to estimating the mean and covariance with MCAR missing values.

Mean, Covariance, and Linear Regression

To gain additional insight as to how the EM iterations can be efficiently organized, consider the *augmented covariance matrix* Λ:

$$\Lambda = \begin{pmatrix} 1 & \mu^T \\ \mu & \Sigma \end{pmatrix}$$

We can use Λ to construct various regression among the variables. For example, the linear regression in which x_i and x_j are used to predict x_k

$$x_k = \beta_0 + \beta_i x_i + \beta_j x_j + \epsilon_k$$

is found by solving for $\{\beta_0, \beta_i, \beta_k\}$ in the normal equations, with N the sample size:

$$\begin{pmatrix} N & \Sigma x_i & \Sigma x_j \\ \Sigma x_i & \Sigma x_i^2 & \Sigma x_i x_j \\ \Sigma x_j & \Sigma x_i x_j & \Sigma x_j^2 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_i \\ \beta_j \end{pmatrix} = \begin{pmatrix} \Sigma x_k \\ \Sigma x_i x_k \\ \Sigma x_j x_k \end{pmatrix}$$

Equivalently, we can solve using expectations rather than sums:

$$\begin{pmatrix} 1 & \mathbb{E}[x_i] & \mathbb{E}[x_j] \\ \mathbb{E}[x_i] & \mathbb{E}\left[x_i^2\right] & \mathbb{E}[x_i x_j] \\ \mathbb{E}[x_j] & \mathbb{E}[x_i x_j] & \mathbb{E}\left[x_j^2\right] \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_i \\ \beta_j \end{pmatrix} = \begin{pmatrix} \mathbb{E}[x_k] \\ \mathbb{E}[x_i x_k] \\ \mathbb{E}[x_j x_k] \end{pmatrix}$$

and all of those expectations can be derived by appropriate row and column operations on the the augmented covariance matrix $\mathbf{\Lambda}$. Thus, $\mathbf{\Lambda}$ contains all of the information we need both for our means and covariances and to produce the regressions needed to estimate required first and second moments of missing data.

In the actual implementation of the EM algorithm the necessary row and column operations needed to produce $\mathbf{\Lambda}$ and the necessary regression and error estimates are done using the *sweep operator* which is covered in detail in the [Little and Rubin, 2002] reference below. If you have followed the basic approach of using regression to estimate missing values, then a careful reading of how the sweep operator works will make the implementation described in [Little and Rubin, 2002] clear.

Running the Algorithm

We start out with a preliminary estimate of the mean and covariance, $\boldsymbol{\vartheta}_1$. Often this can me quite general, *e.g.*, using just all valid observations or even a diagonal matrix of variance estimates. We estimate a starting log-likelihood $\log\mathcal{L}_1$. In the k–th iteration of the EM algorithm…

The E-Step

In the E-step we use the observed data and the prior iteration's estimates of mean and covariance to perform the regressions necessary to produce the sufficient statistics for the missing data.

$$(\mathbf{X}, \boldsymbol{\vartheta}_{k-1}) \longmapsto \mathbf{Z}_k$$

As we saw in the exposition above, for the mean and covariance that is the first and second moments of the missing data.

The M-Step

In the M-step we use the observed data and the revised missing data estimates to produce successor mean and covariance MLEs.

$$(\mathbf{X}, \mathbf{Z}_k) \longmapsto \boldsymbol{\vartheta}_k$$

In the case of the mean and covariance that meant combining the first and second moments of the observed data with those estimated by regression for the missing data.

Check the Likelihood

We then evaluate the likelihood function

$$\log\mathcal{L}_k$$

and compare $\log\mathcal{L}_{k-1}$ and $\log\mathcal{L}_k$. The most common distributional assumption for our problem here is that the mean and covariance are the parameters of a multivariate Normal. If the convergence criterion is met or a predetermined maximum number of iterations have occurred, then terminate; otherwise, return and execute another E- and M-step.

Mathematica and MATLAB Results

We now compute and compare estimates with missing values of the mean and covariance using the EM algorithm in both *Mathematica* and MATLAB. As this is a *Mathematica* notebook, those estimates will be performed *in situ*. The code necessary to perform the same estimate in MATLAB is also provided.

Mathematica

We load the *Mathematica* package written by the author containing the functions required. See the references for how to obtain this code if not bundled with the tutorial. The necessary package is assumed to reside in the same directory as the tutorial.

```
Get[FileNameJoin[{NotebookDirectory[], "MeanCovMissingMLE.m"}]]
```

The help message for the main function is

```
? xMeanCovMissingMLE
```

```
{{Mean, Cov}, {Completed, Cross}, LLHistory} =
    xMeanCovMissingMLE[Data, opts] – MLE estimate of mean and cov with missing data.

Data – Numeric data matrix; each row is an observation; missing values indicated by Missing[].
Option names are strings:
  "InitialEstimate" – "Diagonal" (default), "CompleteObs", and "ImputedMean" or {Mean, Cov}.
    If a string, then defines how the intial estimate is
    produced; otherwise, intial starting values for the mean vector and coveariance matrix.
  "ToleranceGoal" – Rate of change in log likelihood, criterion for EM termination, default 10^–6.
  "MaxIterations" – Max number of iterations, criterion for EM termination, default 100.
  "Verbose" – True to print iteration counts, default False.

Mean – Mean vector
Cov – Covariance matrix
Completed – Data with missing values replaced by expectations based on regression estimates.
Cross – Matrix for second order correction of cross products involving missing data.
LLHistory – History of log likelihood at each EM iteration.
```

We apply xMeanCovMissingMLE[ ] with a tight tolerance, using a preliminary mean and covariance estimate based on all-valid observations, *i.e.*, covDropped:

```
{{meanMma, covMma}, {comp, cross}, logL} =
  xMeanCovMissingMLE[data, "ToleranceGoal" → 10^-9, "InitialEstimate" → "CompleteObs"];
```

The algorithm converges fairly quickly for our simple problem.

```
MatrixForm[logL]
```

$$\begin{pmatrix} 58.8639 \\ 59.1686 \\ 59.1726 \\ 59.1727 \\ 59.1727 \\ 59.1727 \\ 59.1727 \end{pmatrix}$$

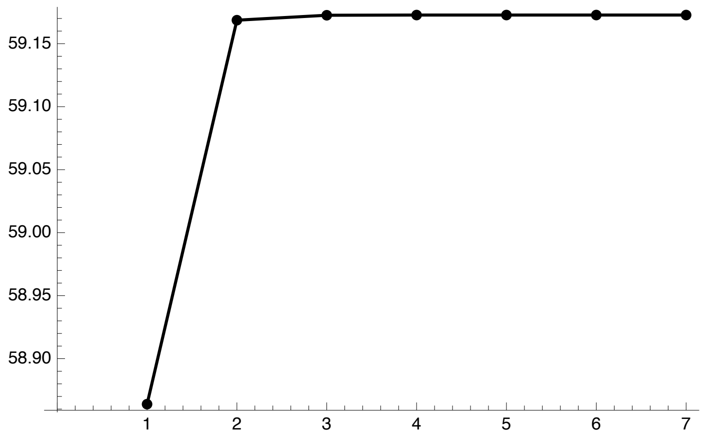
The convergence test is $(\log L_k - \log L_{k-1})/\log L_{k-1} \leq \epsilon$. Note $\log L_k \geq \log L_{k-1}$; hence, this test is always non-negative. The sequence of convergence-test results are below with the tolerance of 10^{-9} achieved at the final iteration, where it is 4×10^{-10}.

```
MatrixForm[(Differences@logL) / Most[logL]]
```

$$\begin{pmatrix} 0.00517736 \\ 0.0000663155 \\ 2.71142 \times 10^{-6} \\ 1.28893 \times 10^{-7} \\ 6.93419 \times 10^{-9} \\ 4.02574 \times 10^{-10} \end{pmatrix}$$

A plot of the log-likelihoods shows the last several iterations are virtually identical. There appears to be little evidence that we terminated the algorithm prematurely.

```
ListLinePlot[logL, PlotRange → All, Mesh → All]
```



MATLAB

As part of the optional Financial Toolbox available for MATLAB, the ecmnmle( ) function according to the online documentation "… estimates the mean and covariance of a data set. If the data set has missing values, this routine implements the ECM algorithm" where the ECM is a variant of the EM algorithm.

The "data" matrix can been written out as a CSV (comma separated value) file using the code below. Note that missing values have been set to "NaN" (Not-a-Number) which the value MATLAB expects for a missing value: the pattern matching rule _Missing → "NaN" replaces each element with a head Missing with the string "NaN".

```
Export[FileNameJoin[{NotebookDirectory[], "dataMATLAB.csv"}], data /. _Missing → "NaN"];
```

Once "data" is in a form expected by MATLAB, the mean and covariance can be estimated in a MATLAB session using the code below. The default initial estimate in ecmnmle( ) is the same as the "CompleteObs" setting used in the *Mathematica* call above.

```
% MATLAB code
%   '...' below represents the path to this notebook

data = csvread('...dataMATLAB.csv');

[meanMATLAB, covMATLAB] = ecmnmle(data);

csvwrite('...meanMATLAB.csv', meanMATLAB);
csvwrite('...covMATLAB.csv', covMATLAB);
```

After the code above has been executed in MATLAB, the result can be read back into *Mathematica* using the code below. Note that MATLAB represents vectors as 1-column matrices so the mean must be flattened after it has been imported. If you do not have access to MATLAB or the Financial Toolbox, the files below have been packaged in the same directory as this tutorial.

```
meanMATLAB = Flatten@Import[FileNameJoin[{NotebookDirectory[], "meanMATLAB.csv"}]];
covMATLAB = Import[FileNameJoin[{NotebookDirectory[], "covMATLAB.csv"}]];
```

Mathematica-MATLAB Comparison

The *Mathematica* and MATLAB estimates are virtually identical. The only difference appears to be that the *Mathematica* results carry more digits after the decimal place.

We can compare the means, including their difference and the L_2 – norm of the difference:

```
MatrixForm[meanMma]
MatrixForm[meanMATLAB]
MatrixForm[meanMma - meanMATLAB]
Norm[meanMma - meanMATLAB]
```

$$\begin{pmatrix} 0.257268 \\ 0.11901 \\ 0.0958446 \end{pmatrix}$$

$$\begin{pmatrix} 0.25727 \\ 0.11901 \\ 0.095844 \end{pmatrix}$$

$$\begin{pmatrix} -2.12069 \times 10^{-6} \\ -1.89226 \times 10^{-7} \\ 5.84339 \times 10^{-7} \end{pmatrix}$$

2.20785×10^{-6}

And do the same for the covariance, using the Frobenius matrix-norm:

```
MatrixForm[covMma]
MatrixForm[covMATLAB]
MatrixForm[covMma - covMATLAB]
Norm[covMma - covMATLAB, "Frobenius"]
```

$$\begin{pmatrix} 0.023008 & 0.00793362 & 0.00963754 \\ 0.00793362 & 0.00599427 & 0.00407703 \\ 0.00963754 & 0.00407703 & 0.0137816 \end{pmatrix}$$

$$\begin{pmatrix} 0.023008 & 0.0079336 & 0.0096375 \\ 0.0079336 & 0.0059943 & 0.004077 \\ 0.0096375 & 0.004077 & 0.013782 \end{pmatrix}$$

$$\begin{pmatrix} 2.99097 \times 10^{-9} & 2.11372 \times 10^{-8} & 4.2871 \times 10^{-8} \\ 2.11372 \times 10^{-8} & -2.8387 \times 10^{-8} & 3.38228 \times 10^{-8} \\ 4.2871 \times 10^{-8} & 3.38228 \times 10^{-8} & -3.65122 \times 10^{-7} \end{pmatrix}$$

3.75481×10^{-7}

R (r-project.org)

Undoubtedly the most accessible resource for these computations is the free, open-source R statistical analysis system, although it is viewed by many as a general programming environment. The “norm” package performs MLE multivariate Normal estimation with missing data. The function prelim.norm( ) prepares the data, and em.norm( ) estimates the mean and covariance. Running a test using the demonstration data above is left as an exercise for the student.

Comparisons with EM Result

Using meanMma and covMma as our proto-typical EM estimate we will compare the results against our earlier attempts. Note that there is not meanCompleted results as no correction for second-order effects were necessary for the regression-based estimate

```
TableForm[
 {{Norm[meanMma - meanDropped], Norm[meanMma - meanReplaced],
    Norm[meanMma - meanRegressed], "NA"},
   {Norm[covMma - covDropped, "Frobenius"], Norm[covMma - covReplaced, "Frobenius"],
    Norm[covMma - covRegressed, "Frobenius"],
    Norm[covMma - covCompleted, "Frobenius"]}}ᵀ,
 TableHeadings →
  {{"|Dropped-EM|", "|Replaced-EM|", "|Regressed-EM|", "|Completed-EM|"},
   {"Mean", "Cov"}}]
```

	Mean	Cov
|Dropped-EM|	0.0298635	0.00551317
|Replaced-EM|	0.00976195	0.00298422
|Regressed-EM|	0.0137752	0.00261806
|Completed-EM|	NA	0.00201741

Not surprisingly, the covCompleted estimate is the closest to the MLE solution. As evidenced by the plot of the log likelihood above, the first iteration gets us very close to the solution, and covCompleted is the essentially the covariance after one EM iteration.

It may seem strange that meanRegressed *isn't* the closest. Consider, however, the regressions that produce the conditional means for missing values. The first set of regressions were produced with the means and mean sums of squares associated with a rathter poor covariance estimate. As the covariance estimates evolve through repeated EM iterations, those improvements are likewise reflected in the improved regressions for each missing value.

References

[Dempster *et al*., 1977] is the seminal paper which formalized the EM algorithm although its development was foreshadowed by the work of many earlier researchers who used similar techniques to handle specific problems involving occult data. [Little and Rubin, 2002] is an excellent text covering a broad array of problems involving missing data; Chapter 7 deals with estimating the mean and covariance of mutlivariate Normal data when there are MCAR missing values. Finally, [McLachlan *et al*., 2008] provides a general overview of more recent developments in the EM Algorithm , including coverage a wide array of problems and the development of many algorithmic improvements.

- Dempster, A.P., N. M. Liard, and D. B. Rubin, "Maximum-Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society*, *Series B*, 39, 1977.
- Frey, Robert J., *Mathematica Package*, “MeanCovMLE”, Rev. 4 (2011-08-03), (bundled with this tutorial and available from author at Robert.Frey@StonyBrook.edu), 2011.
- Little, Roderick J. A., and Donald B. Rubin, *Statistical Analysis with Missing Data*, Wiley-Interscience, 2002.
- MATLAB, Version 2014b, *Financial Toolbox*, “ecmnmle( )”, 2014.
- McLachlan, G. J., and T. Krishnan, *The EM Algorithm and Extensions*, *2nd Ed*., Wiley-Interscience, 2008.
- R 3.0.3, “norm : EM algorithm for MLE of multivariate Normal data with missing values” (Version 1.0-9.5), functions prelim.norm( ) and em.norm( ), 2014.