A brief guide to Ensemble End-Member Mixing Analysis (EEMMA) using eemma.R

James W. Kirchner
Dept. of Environmental Systems Science, ETH Zurich, Zurich, Switzerland
Swiss Federal Research Institute WSL, Birmensdorf, Switzerland
Dept. of Earth and Planetary Science, University of California, Berkeley, CA, USA
kirchner@ethz.ch

v1.0, 14.04.2023

This document is intended as a brief user's guide to Ensemble End-Member Mixing Analysis (EEMMA), using the R script eemma.R. In this guide, EEMMA (upper case) will refer to the general theory and method, and eemma (lower case) will refer to the specific implementation in R. For an overview of the concepts and mathematics behind EEMMA, users should consult *Kirchner, J.W., Mixing models with multiple, overlapping, or incomplete end-members, quantified using time series of a single tracer* (Geophysical Research Letters, 2023), hereafter denoted K2023. References to equations and sections refer to that paper, unless otherwise stated. For the computational details, users should consult the code itself, which is extensively commented. This document presents a vignette of a simple application, explains the various options that users may wish to select, and provides some practical advice that may be useful. It is assumed that readers have some basic familiarity with R; for those who don't, many tutorials are available online.

There are three main sections below:

1. **Calling eemma** (an explanation of the options in the eemma function call)
2. **Outputs from eemma** (what the name says: an explanation of the outputs)
3. **A vignette** (what the name says: an illustrative vignette of some simple eemma applications)

1. Calling eemma

The function declaration for eemma, on line 450 of the code, may look intimidating:

```
eemma <- function(
    y,
    X,
    complete = TRUE,
    memory = FALSE,
    no.intercept = complete,
    method = c("OLS", "EIV"),
    err.all = 1,
    err.y = err.all,
    err.x = rep(err.all, NCOL(X)),
    sercorr.err.y = 0,
    rxx = 0,
    rxy = 0,
    uncert = c("relative", "absolute"),
    alpha = 1
)
```

Notice, however, that there are default options for nearly everything. Most of those default options will do what you wanted (even if you didn't know you wanted it).

Nonetheless, here is a complete list of the parameters and what they do. Much of this is already in the code, just below the function declaration.

y and X

There are only two things that eemma must have, for obvious reasons: **y**, a vector of tracer measurements in the mixture, and **X**, a vector, matrix, data table, or data frame containing tracer measurements in one or more end-members. If you have data for only one end-member (yes, eemma can estimate the mixing fraction of a single end-member, if you don't have measurements of the others…) and supply **X** as a vector, it must be the same length as **y**. If **X** is a matrix, data table, or data frame containing measurements from multiple end-members (the more usual case), its number of rows must equal the length of **y**. If the columns of **X** have names (assigned using the colnames function in R, for example), eemma will carry them through to the outputs; otherwise the end-members will be named "EM1", "EM2", and so forth. Missing (NA) values are allowed in both **y** and **X**, but only complete cases will be analyzed. Denote missing values with NA, not with a numerical code and particularly not with zero! (NA says you don't have a measurement. Zero says you actually have a measurement, and its value is zero. There's a difference.)

The sequence of measurements does not need to be an evenly spaced time series. However, it should go without saying (but just in case…) that EEMMA is based on the premise that each row of X, and the corresponding element of y, jointly comprise a set of measurements that belong together. Whether these measurements need to be simultaneous will depend on the specific situation; the math doesn't care one way or the other. The math does, however, care greatly that each set of measurements is actually a natural set. That means you shouldn't randomize the vectors with respect to each other, or sort them all into ascending order, or anything like that.

What should you do if you have different measurement frequencies for different components of your system? For example, what if you have measurements of xylem water (the mixture) and soil water (one end-member) every two weeks, but you have measurements of rainwater daily and groundwater every month? There are at least two options here. One is to aggregate or subsample all of the measurements at the lowest sampling frequency in the data set. This would mean calculating volume-weighted means of precipitation for each month (or each interval between pairs of groundwater measurements), and subsampling the xylem and soil water data at monthly frequency. Another pragmatic approach would be to create a data set at the sampling frequency of the mixture, aggregating anything sampled more frequently (in this case, precipitation) and interpolating anything sampled less frequently (in this case, groundwater).

complete

This is a flag for whether all of the end-members that contribute to the mixture are included in X (**complete**=TRUE), or whether some end-members are missing (**complete**=FALSE). If **complete** is TRUE (which is the default), the methods of Sections 1-3 are used, forcing the mixing fractions to sum to 1. If **complete** is FALSE, the methods of Section 4 are used, and eemma will estimate the mixing fractions of the available end-members (which do not need to sum to 1), along with the mixing fraction and time series of the unmeasured end-member (or the time series of their weighted sum, if there are multiple unmeasured end-members).

memory

This is a flag for whether eemma should estimate a first-order memory coefficient that quantifies how much of the mixture in each time step is carried over from the previous time step. If **memory**=TRUE, the methods of Section 5 are used to estimate the memory coefficient and the mixing fractions simultaneously. By default, **memory**=FALSE. Users may wish to check whether there is any evidence of a memory effect by setting **memory**=TRUE and seeing whether the memory coefficient (which will be named "lagmix" in the output) is significantly different from zero, but they should also consider whether such memory effects are physically or biologically realistic.

no.intercept

This is a flag that tells eemma not to fit an intercept (i.e., to fix the intercept at zero). Most users will not need to set any value for this flag. By default, **no.intercept** is TRUE whenever **complete** is TRUE and FALSE whenever **complete** is FALSE. When **complete**=TRUE, intercepts are omitted regardless of how **no.intercept** is set, because this is required by Equations 4, 7, and 13. Conversely, when **complete**=FALSE, an intercept is needed (i.e., **no.intercept**=FALSE by default) so that the missing end-member can be correctly estimated (see Equations 9, 10, and 12). So are there any situations where users might want to set **complete**=FALSE but **no.intercept**=TRUE? The only case would be where the set of end-members is known to be incomplete, but one of the end-members is constant or nearly so, and thus its coefficient could not be estimated if an intercept were used. Note, however, that in such cases it would only be valid to set **no.intercept**=TRUE if all missing end-members had mean values that were identical to the mean of the mixture. General advice: just don't touch **no.intercept**.

method

This flag determines whether the calculations will be performed by ordinary least squares (**method**="OLS") or by the methods of moments approach to errors in variables, using a variant of Fuller's (1987) method as described in Supporting Information Section S2 (**method**="EIV"). Setting method="OLS" (the default) will yield better-constrained estimates of mixing fractions and smaller residuals. However, mixing fraction estimates obtained by OLS will be biased when the end-member tracer concentrations have substantial sampling and measurement uncertainties (even when those measurements are unbiased).

As a rough guide, one can expect this bias to scale by the variance of the measurement errors, divided by the variance of the measurements themselves, so if the standard deviation of the measurement errors is half as large as the standard deviation of the data themselves, one can expect the bias to be on the order of 25%, whereas if the uncertainty in the measured values is 20% of their standard deviation, one can expect the bias to be on the order of 4% or so. These are only rough indications; the size of the bias will also depend on any correlations among the end-members, and among their uncertainties (as well as any constraints that arise from differences among end-member means if **complete**=TRUE or **no.intercept**=TRUE).

Where measurement and sampling errors are a substantial fraction of the total variance in the end-members, setting **method**="EIV" will typically result in mixing fraction estimates that are much less biased; that is, *on average* they will more closely correspond with the (unknown) true values. However, this comes at the cost of some increase in the scatter in the mixing fraction estimates; that is, *individual estimates* could potentially lie farther from the (unknown) true values. The EIV method requires estimates of the measurement/sampling errors in the end-members and mixture (see parameters below). If no such estimates are provided, all measurement/sampling uncertainties will be assumed to be equal and uncorrelated, and the magnitude of these uncertainties will be estimated within the EIV procedure itself.

All of the following parameters only pertain to the EIV method; they have no effect if method=OLS:

err.all

This is a single number quantifying the standard deviation of the measurement/sampling errors in the mixture and all of the end-members. Use **err.all** if you think all of your measurements are equally uncertain; otherwise provide values for **err.x** and **err.y** (see below), which will both override any value of **err.all**. By default, **err.all**=1 and **uncert**="relative" (see below); this has the effect of letting eemma pick a single value that best represents the uncertainties in all the measurements, based on the available data.

err.y

This is a single number quantifying the standard deviation of the measurement/sampling errors in the mixture. **err.y** inherits the value of **err.all**, unless users specify a different value here.

err.x

A vector or a single number quantifying the standard deviation of the measurement/sampling errors in the end-members. If **err.x** is specified as a single number, it will be applied to all of the end-members. If **err.x** is specified as a vector, it must have the same length as the number of end-members and each value will be applied to the corresponding end-member. If no value is specified for **err.x**, it will inherit the value of **err.all**.

sercorr.err.y

The lag-1 serial correlation in the measurement/sampling errors in the mixture. (Note that this is the serial correlation in the *errors*, not the serial correlation in the mixture values themselves.) This is only needed as an input to the bias-correction procedure when **method**="EIV" and **memory**=TRUE. Must be a single value between -1 and 1. The default value is 0.

rxx

The correlation matrix (note: *not covariance matrix!*) of the measurement/sampling errors of the end-members (note: *not of the end-member values themselves!*). **rxx** can be supplied as a single value or as a matrix. If a single value is supplied for **rxx**, eemma constructs a correlation matrix using that value for all of the cross-correlations. If **rxx** is supplied as a matrix, it must have as many rows and columns as the number of end-members, the diagonal elements must equal 1, and it must be positive definite (these are properties that all correlation matrices must satisfy). If **rxx** is not specified, the identity matrix will be used by default (implying no cross-correlations among the measurement/sampling errors in the end-members). General advice: users could test for cross-correlations among their measurements through replicate sampling and analysis, but note, again, here we are talking about correlations among the *errors*, not among the measurement *values* themselves. Correlated measurement errors could arise from (for example) day-to-day shifts in instrument calibration, if some sets of measurements are analyzed together on one day, and other sets are analyzed together on other days. But many users may instead just try several plausible **rxx** values to see whether they make a substantial difference.

rxy

A vector of correlations (note: *not covariances!*) between the measurement/sampling errors in the mixture and the measurement/sampling errors of the end-members (note: *not between the measurements themselves!*). **rxy** can be supplied as a single value or as a vector. If a single value is supplied for **rxy**, eemma constructs a vector using that value for all of the cross-correlations. If **rxy** is supplied as a vector, its length must equal the number of end-members. **rxy** values must be between -1 and 1. If **rxy** is not specified, these cross-correlations will be assumed to be zero.

uncert

This flag tells eemma whether the uncertainties provided in **err.y**, **err.x**, and **err.all** are expressed as relative values (**uncert**="relative", which is the default), or as absolute values (**uncert**="absolute"). Use **uncert**="relative" if you don't know how big the uncertainties are, but you can at least estimate their ratios. For example, you might not have a good estimate for any of your sampling and measurement uncertainties, but you might have a good basis for assuming that their standard deviations are twice as big in end-member #1 as in the mixture, and three times as big in end-member #2 as in end-member #1. In this case you would set **uncert**="relative", **err.y**=1, and **err.x**=c(2,6), or specify any other three numbers in the ratios of 1:2:6. (Note that these are ratios among the uncertainties; here, "relative" does not mean "as a fraction of the mean measured value".) Use **uncert**="absolute" if you can confidently estimate how big the uncertainties (error standard deviations) in your data really are. General advice: quantifying measurement and (especially) sampling uncertainties is difficult, so use **uncert**="relative" unless you have a good reason not to. That's why it's the default.

alpha

This is the alpha parameter for Fuller's method. It must be positive. Lower values give smaller bias but larger parameter variance; higher values give more bias but smaller parameter variance. Values between 1 and 4 have performed well in benchmark tests, usually with minor differences in bias and parameter variance.

2. Outputs from eemma

eemma returns a list containing the following objects:

$coeffs

This is a standard t-table, including the coefficients ("f") and their standard errors ("se"), as well as the t-statistics and their associated p-values. For example:

```
               f         se t_statistic      p_value
precip 0.2068911 0.01141577   18.123267 7.440396e-13
soil   0.2101869 0.04916945    4.274746 2.560387e-04
gw     0.5829221 0.05032645   11.582816 8.628136e-10
```

The p-values are ***one-tailed***, because mixing fractions less than zero automatically violate any mechanistic hypothesis. Two-tailed p-values can be obtained by doubling the one-tailed values.

$f_cov

This is the covariance (not correlation!) matrix of the estimated coefficients. For example:

```
              precip          soil            gw
precip  1.303198e-04 -7.601319e-06 -0.0001227185
soil   -7.601319e-06  2.417635e-03 -0.0024100334
gw     -1.227185e-04 -2.410033e-03  0.0025327519
```

The square root of the diagonal of this matrix is the vector of standard errors.

$RMSE

The root-mean-square error of the regression.

$f

The estimated coefficients. This is also the first column of **$coeffs**.

$se

The standard errors of the estimated coefficients. This is also the second column of **$coeffs**.

$e

The residuals of the fitted model.

$timeseries

The input time series, omitting any rows with NA (which also had to be omitted from the analysis). For example:

```
        mixture     precip       soil         gw
 [1,]  -9.807086 -10.069171  -9.498827 -10.076742
 [2,]  -9.705679  -8.567149  -9.100631 -10.165846
 [3,]  -9.194975  -7.711953  -8.680561 -10.127792
 [4,] -10.920072 -14.128967  -9.172344 -10.115267
 [5,] -10.088557  -9.355356  -9.755653 -10.275222
 [6,]  -9.414263  -7.385020  -9.675359 -10.071577
 [7,]  -9.793168  -9.108292 -10.159477 -10.075523
 [8,] -10.329509 -11.704551  -9.878755  -9.933583
```

If **complete**=FALSE, and **no.intercept**=FALSE, the intercept and the estimated time series of the missing source will be included in **$timeseries**, with the headings "intercept" and "unk", respectively.

$lambda

The $\hat{\lambda}$ parameter estimated using Equation S16 as part of the EIV algorithm. If **method**="OLS", this will be NA.

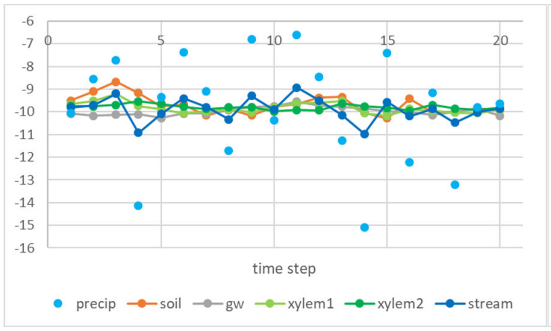
3. A vignette

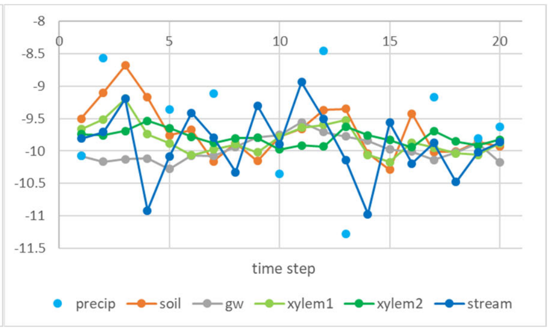
Here I present a brief worked example illustrating a potential application of eemma to a simple data set (in this case a synthetic one).

Imagine that we are interested in tracing where "blue water" (streamflow) and "green water" (transpiration) fluxes come from. So, over 20 time steps (say, weeks or months), we measured δ^{18}O in precipitation, soil water, groundwater, and streamflow, as well as xylem water in two species of trees. These (synthetic, make-believe) time series can be found in "vignette_data.txt":

timestep	precip	soil	gw	xylem1	xylem2	stream
1	-10.069	-9.499	-10.077	-9.665	-9.742	-9.807
2	-8.567	-9.101	-10.166	-9.513	-9.757	-9.706
3	-7.712	-8.681	-10.128	-9.194	-9.692	-9.195
4	-14.129	-9.172	-10.115	-9.739	-9.536	-10.920
5	-9.355	-9.756	-10.275	-9.879	-9.648	-10.089
6	-7.385	-9.675	-10.072	-10.067	-9.779	-9.414
7	-9.108	-10.159	-10.076	-9.975	-9.877	-9.793
8	-11.705	-9.879	-9.934	-9.899	-9.802	-10.330
9	-6.788	-10.152	-9.793	-10.018	-9.795	-9.301
10	-10.358	-9.776	-9.754	-9.783	-9.975	-9.898
11	-6.609	-9.659	-9.555	-9.639	-9.916	-8.936
12	-8.459	-9.369	-9.705	-9.595	-9.929	-9.507
13	-11.281	-9.349	-9.776	-9.522	-9.626	-10.135
14	-15.094	-10.035	-9.840	-10.059	-9.758	-10.974
15	-7.406	-10.290	-9.975	-10.173	-9.830	-9.560
16	-12.210	-9.425	-10.006	-9.870	-9.944	-10.194
17	-9.170	-10.017	-10.135	-9.949	-9.691	-9.874
18	-13.202	-10.010	-10.018	-10.037	-9.854	-10.475
19	-9.802	-9.853	-9.879	-10.061	-9.917	-10.021
20	-9.630	-9.933	-10.176	-9.879	-9.822	-9.862

One can visualize the time series as shown below (the right panel has expanded axes to more clearly show some of the signals). Precipitation, soil water, and groundwater ("gw") are shown in light blue, orange, and gray, respectively, the two trees (xylem1 and xylem2) are shown in light and dark green, and the streamwater is shown in dark blue.





Because these are synthetic data, we know how precipitation, soil water, and groundwater were combined to make xylem water and streamwater. (Needless to say, this is a luxury that we don't have in real-world cases.) The streamwater is derived 20% from precipitation, 20% from soil water, and 60% from groundwater. The xylem water in both trees is derived from 60% soil water and 40% groundwater (and no precipitation). The difference between the trees is that xylem1 is completely replaced from its sources in each time step, whereas 72% of xylem2 in each time step is carried over from the previous time step, with the other 28% coming from soil water and groundwater in the same 60:40 ratio. The soil water and groundwater also exhibit memory effects; 60% of each soil water value and 90% of each groundwater value are carried over from the previous time step. A small amount of random noise (standard deviation of 0.1 per mil) has been added to each value of each variable.

Let's try analyzing these data. The steps below are also reproduced in eemma_vignette_script.R. User inputs are shown in `blue` and console outputs from R are shown in `black text`.

As is typical in R, we begin by setting the working directory, cleaning up anything left in the environment, and sourcing the script that contains the eemma function:

```
setwd("~/EEMMA paper")   # set this to wherever eemma.R and your data are
rm(list=ls())            # clean up the environment
source("eemma.R")        # source the eemma script
```

We can then read in our data set…

```
dat <- read.table(file="vignette_data.txt", header=TRUE, sep="\t")
```

… and get an overview of what the data look like:

```
str(dat)
'data.frame':20 obs. of  7 variables:
 $ timestep: int  1 2 3 4 5 6 7 8 9 10 ...
 $ precip  : num  -10.07 -8.57 -7.71 -14.13 -9.36 ...
 $ soil    : num  -9.5 -9.1 -8.68 -9.17 -9.76 ...
 $ gw      : num  -10.1 -10.2 -10.1 -10.1 -10.3 ...
 $ xylem1  : num  -9.67 -9.51 -9.19 -9.74 -9.88 ...
 $ xylem2  : num  -9.74 -9.76 -9.69 -9.54 -9.65 ...
 $ stream  : num  -9.81 -9.71 -9.19 -10.92 -10.09 ...
```

For convenience, we can unbind the dataframe "dat" so that each column is a separate vector:

```
list2env(dat, envir=.GlobalEnv)
```

Now we can try a first analysis, looking at xylem1 as a mixture of precipitation, soil water, and groundwater. We can do this either by saving the eemma output and then examining the coefficients

```
mm <- eemma(y=xylem1, X=cbind(precip, soil, gw))
mm$coeffs
```

or by extracting the coefficients directly

```
eemma(y=xylem1, X=cbind(precip, soil, gw))$coeffs
```

Either approach yields the following coefficient table:

```
                f         se t_statistic      p_value
precip 0.01458161 0.01107822    1.316242 1.027826e-01
soil   0.56922839 0.04771557   11.929615 5.504653e-10
gw     0.41619000 0.04883837    8.521784 7.640218e-08
```

Here we see that the estimated contribution from precipitation is tiny (and is only a bit more than one standard error away from zero), whereas soil water and groundwater are present in a ratio of 60:40, within one standard error. Thus these results are consistent with what we know about how the data were created.

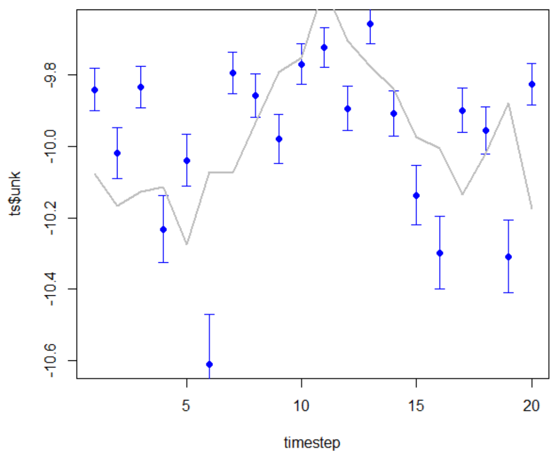
The analysis above satisfies the usual requirement of end-member mixing models, namely that all of the relevant end-members appear in the data set. What if this is not the case? What if, for example, we only had measurements of precipitation and soil water, but not groundwater? Using conventional end-member mixing methods, we would never know that anything was missing, unless the mixture lay outside the range of all of the measured end-members. EEMMA, by contrast, allows one to test whether important end-members are missing via the **complete**=FALSE option:

```
eemma(y=xylem1, X=cbind(precip, soil), complete=FALSE)$coeffs
                    f          se t_statistic      p_value
precip     0.01628097 0.01085917    1.499283 7.556737e-02
soil       0.52489645 0.06343070    8.275117 7.545625e-08
intercept -4.57869373 0.62528608    7.322558 4.219067e-07
unk        0.45882257 0.06443678    7.120508 6.173660e-07
```

The results above show that when we tell eemma that the mixing model is incomplete (**complete**=FALSE), eemma correctly estimates, as before, that the mixing fraction of precipitation is roughly zero within error, and the mixing fraction of soil water is roughly 60% within error. But it also tells us that there is an unknown end member "unk" that comprises roughly 40% of the mixture, with an associated standard error of only about 6%. Thus eemma is telling us that we are very clearly missing at least one substantial end-member.

It can also tell us something about the tracer signature of that missing end-member (or of the volume-weighted average of multiple missing end-members). As described in Section 5 of K2023, EEMMA allows us to estimate the tracer time series in the missing end-member "unk", and its standard error. Using the instructions below, we can plot eemma's estimate for the missing end-member, and its standard error, in blue, compared to the actual missing groundwater data (which we have the luxury of knowing, in this synthetic example) in gray:

```
ts <- eemma(y=xylem1, X=cbind(precip, soil), complete=FALSE)$timeseries
plot(timestep, ts$unk, col='blue', pch=19)
arrows(timestep, ts$unk-ts$unk_std_err, timestep, ts$unk+ts$unk_std_err,
     length=0.05, angle=90, code=3, col="blue")
lines(timestep, gw, col='gray', lwd=2)
```



The plot shows us that the missing end-member has an average value of about -10 per mil, but it does not precisely match the ups-and-downs of the actual groundwater signal, in part because the groundwater time series is strongly damped and thus does not have a strong influence on the xylem signal. The plot suggests that the missing values are somewhat higher in the middle of the time series, in agreement with the real groundwater signal, but more precise inferences cannot be drawn.

If we specify **complete**=FALSE when no end-members are actually missing, eemma should return an estimated mixing fraction of roughly zero (within error) for the unknown end-member, which is what happens in our case:

```
eemma(y=xylem1, X=cbind(precip, soil, gw), complete=FALSE)$coeffs
                    f          se t_statistic      p_value
precip     0.01502141 0.01067974    1.406534 8.879147e-02
soil       0.53569228 0.06267287    8.547435 7.326526e-08
gw         0.18118903 0.13817945    1.311259 1.036030e-01
intercept -2.67960750 1.57246935    1.704076 5.328729e-02
unk        0.26809728 0.15855543    1.690874 5.455549e-02
```

Readers may be concerned that when **complete**=FALSE, the estimated mixing fraction of groundwater is also roughly zero, within error. That result presumably arises because, as mentioned above, the groundwater signal is strongly damped. But considered together, the last two sets of results give a consistent picture. When groundwater is excluded, eemma says that an important end-member is missing; conversely, when groundwater is included, eemma does not give a strong signal that the model is incomplete.

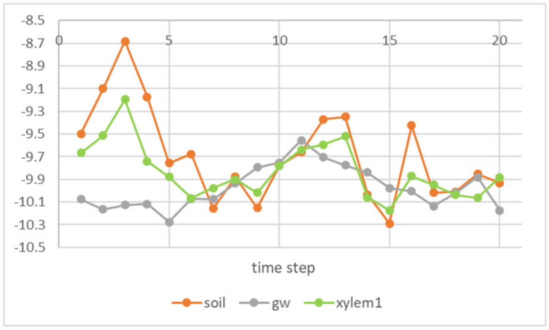
Note that one cannot determine whether an end-member is missing based on the estimated mixing fractions when **complete**=TRUE, as the following example illustrates:

```
eemma(y=xylem1, X=cbind(precip, soil))$coeffs
                f          se t_statistic      p_value
precip 0.03474711 0.02363558    1.470119 7.939670e-02
soil   0.96525289 0.02363558   40.838968 1.680702e-19
```

The math underlying EEMMA guarantees that when **complete**=TRUE, the mixing fractions must sum to 1, and thus will give no hint of a missing end-member. The right way to test for a missing end-member is to specify complete=FALSE, and see if the estimated mixing fraction of the unknown end-member (unk) is clearly different from zero.

Note also that when **complete**=TRUE, one cannot meaningfully compare t-statistics and p-values between models that have different sets of end-members. Thus, in the example above, the soil end-member's very large t-statistic and very small p-value do not mean that an end-member mixing model that excludes groundwater is better than our initial model that included groundwater.

We can also test for memory effects in the mixture. This is not as simple as testing for serial correlation in the xylem measurements, because the soil water and groundwater signals are also damped by memory effects, as the time series show:

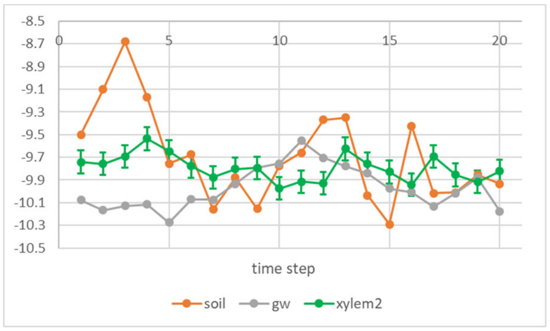


Instead, we test for memory effects by setting **memory**=TRUE:

```
eemma(y=xylem1, X=cbind(precip, soil, gw), memory=TRUE)$coeffs
                f          se t_statistic      p_value
precip 0.02102953 0.01485511    1.415643 8.865388e-02
soil   0.57081759 0.05857590    9.744922 3.506238e-08
gw     0.40815288 0.06163491    6.622105 4.060647e-06
lagmix 0.13393360 0.13135832    1.019605 1.620400e-01
```

The coefficient for the memory term (lagmix) is zero within error, indicating that memory effects are not substantial (which we know to be correct in this hypothetical case).

The hypothetical xylem from the second tree, by contrast, exhibits substantial memory effects, resulting in a very damped signal that is not much larger than the added random measurement errors (indicated by the error bars in the plot below):



When we test for a memory effect in this system, using the same model that we used for xylem1, we find that the coefficient for the memory term (lagmix) is clearly greater than zero, and is within one standard error of the true value (0.72):

```
eemma(y=xylem2, X=cbind(precip, soil, gw), memory=TRUE)$coeffs
                  f         se t_statistic      p_value
precip -0.002256761 0.05352275  0.04216451 4.834618e-01
soil    0.564638607 0.26108935  2.16262592 2.356617e-02
gw      0.437618153 0.26765053  1.63503562 6.142394e-02
lagmix  0.770389817 0.11561457  6.66343181 3.782361e-06
```

Here the mixing fractions for soil and groundwater are close to the true 60:40 ratio, although with large uncertainties because the xylem signal is so strongly damped that it is mostly obscured by measurement noise (see the figure above).

What happens if we ignore the memory effects detected above, and instead estimate the mixing model with **memory**=FALSE?

```
eemma(y=xylem2, X=cbind(precip, soil, gw), memory=FALSE)$coeffs
                 f         se t_statistic      p_value
precip -0.00887984 0.02256012   0.3936078 3.493817e-01
soil    0.40622847 0.09716985   4.1806019 3.136749e-04
gw      0.60265137 0.09945635   6.0594559 6.378656e-06
```

The statistics superficially look much nicer, because the standard errors are much smaller, but note that the estimated mixing fractions are substantially different from the true ones (a 40:60 ratio instead of a 60:40 ratio). This occurs because eemma has overestimated the contribution from the highly damped groundwater end-member, in order to account for the extra damping in the xylem (which instead arises from its own memory effects). This example illustrates similar lessons to those learned from the analysis shown above with **complete**=FALSE. Blindly chasing nicer-looking statistics can yield misleading results, in particular because the statistics typically cannot be compared between models with different end-members. And just as it is risky to set **complete**=TRUE when one has learned that an end-member is missing (i.e., when "unk" is clearly nonzero), it is similarly risky to set **memory**=FALSE when one has learned that there are substantial memory effects (i.e., when "lagmix" is clearly nonzero). Particularly because **complete**=TRUE and **memory**=FALSE by default, it will be important for users to test for missing end-members and memory effects unless they have strong prior reasons to believe that they are unimportant.

It bears emphasis that *any* mixing analysis, not just EEMMA, can potentially be affected by memory effects and missing end-members. But typical mixing analyses have no way to take these issues into account, or even to detect that they exist. EEMMA can do both.

Let's now look at the sources of streamflow. A straightforward analysis with all three end-members yields mixing fraction estimates that are equal, within error, to the true ratios of 20:20:60, as the following table shows:

```
eemma(y=stream, X=cbind(precip, soil, gw))$coeffs
               f         se t_statistic      p_value
precip 0.2068911 0.01141577   18.123267 7.440396e-13
soil   0.2101869 0.04916945    4.274746 2.560387e-04
gw     0.5829221 0.05032645   11.582816 8.628136e-10
```

When we test for missing end-members, we find no indication that they exist (the "unk" mixing fraction is roughly zero, within error):

```
eemma(y=stream, X=cbind(precip, soil, gw), complete=FALSE)$coeffs
                   f         se t_statistic      p_value
precip     0.2070744 0.01163847   17.792245 1.002847e-12
soil       0.1713226 0.06829907    2.508418 1.127782e-02
gw         0.4222108 0.15058395    2.803823 6.103389e-03
intercept -1.9784636 1.71363139    1.154544 1.321246e-01
unk        0.1993922 0.17278910    1.153963 1.322406e-01
```

And similarly, when we test for memory effects, we do not find any (the memory coefficient "lagmix" is roughly zero, within error):

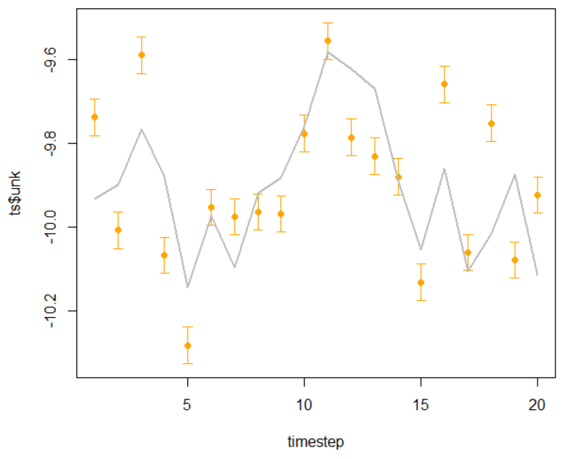
```
eemma(y=stream, X=cbind(precip, soil, gw), memory=TRUE)$coeffs
               f         se t_statistic      p_value
precip 0.2229728 0.02364249   9.4310196 5.369565e-08
soil   0.1774278 0.05755744   3.0826215 3.790791e-03
gw     0.5995994 0.05431148  11.0400117 6.696217e-09
lagmix 0.0552703 0.06634318   0.8330968 2.089292e-01
```

Now, if what if we only had measurements of precipitation and streamflow? Could we still correctly infer that precipitation makes up only 20% of streamwater?

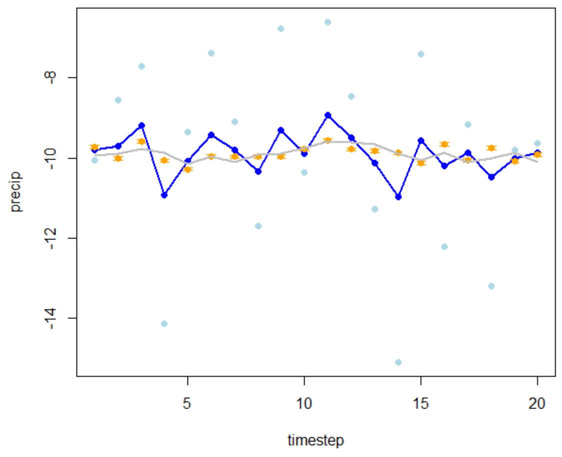
```
eemma(y=stream, X=cbind(precip), complete=FALSE)$coeffs
                   f         se t_statistic      p_value
precip     0.2098147 0.01459053    14.38020 5.767539e-12
intercept -7.8219550 0.14846348    52.68606 2.317173e-22
unk        0.7901853 0.01459053    54.15741 1.377555e-22
```

Indeed, the table above shows that one can constrain the mixing fraction of precipitation to within about 1%. Now, how well can we constrain the missing end-member and its temporal dynamics? In our hypothetical data set, soil water and groundwater make up 20 and 60 percent of streamflow, respectively; thus they jointly account for 80 percent of streamflow and can be considered as a single missing end-member that combines soil water and groundwater in a 20/60 (or 1:3) ratio. If we estimate this missing end-member (gray line below) and compare it to eemma's estimate of what the missing end-member must be (orange symbols, with error bars), we get the following:

```
ts <- eemma(y=stream, X=cbind(precip), complete=FALSE)$timeseries
plot(timestep, ts$unk, col='blue', pch=19, ylim=c(min(ts$unk-ts$unk_std_err),
     max(ts$unk+ts$unk_std_err)))
arrows(timestep, ts$unk-ts$unk_std_err, timestep, ts$unk+ts$unk_std_err,
     length=0.05, angle=90, code=3, col="blue")
lines(timestep, 0.25*soil+0.75*gw, col='gray', lwd=2)
```



In the figure above, the estimated values of the missing end-member (orange symbols) do not seem to be closely related to the true soil-groundwater mixture (gray line). But it is important to keep the axis scale in perspective. Both the estimated values and the true values are constrained to a narrow range of about 0.6-0.7 per mil, whereas the other end-member (precipitation) varies over a range of nearly 9 per mil. The figure below shows the orange symbols and gray line from the figure above, combined with the precipitation (light blue) and streamflow (dark blue) time series. From this perspective, one can see that eemma has estimated the value of the missing soil/groundwater end-member, and its limited dynamics, rather precisely relative to the much larger fluctuations in the precipitation and streamflow values that it has received as its only input data.


precip
timestep