EM Algorithm Example

MATH 6630

2024-11-12

Bivariate Normal with one variable missing depending on value of non-missing variable

In this example, missingness of X_2 depends the value of X_1 being above a cut-off value, but the method used here works as long as the probability of missingness is a function of the observed value of X_1. The manner of dependency need not be monotone, and it need not be deterministic. The probability of missingness could be a completely arbitrary, e.g. discontinuous, and unknown function of X_1.

```
knitr::opts_chunk$set(fig.height=6)
library(spida2)
library(lattice)
library(latticeExtra)
library(latex2exp)
library(mvtnorm)    # multivariate normal and t
```

1,000 observation from bivariate normal

```
set.seed(1232)
help(p=mvtnorm)
?rmvnorm
N <- 1000
Sigma <- cbind(c(1.1, .9), c(.9, 1.2))
Sigma
```

```
##      [,1] [,2]
## [1,]  1.1  0.9
## [2,]  0.9  1.2
```

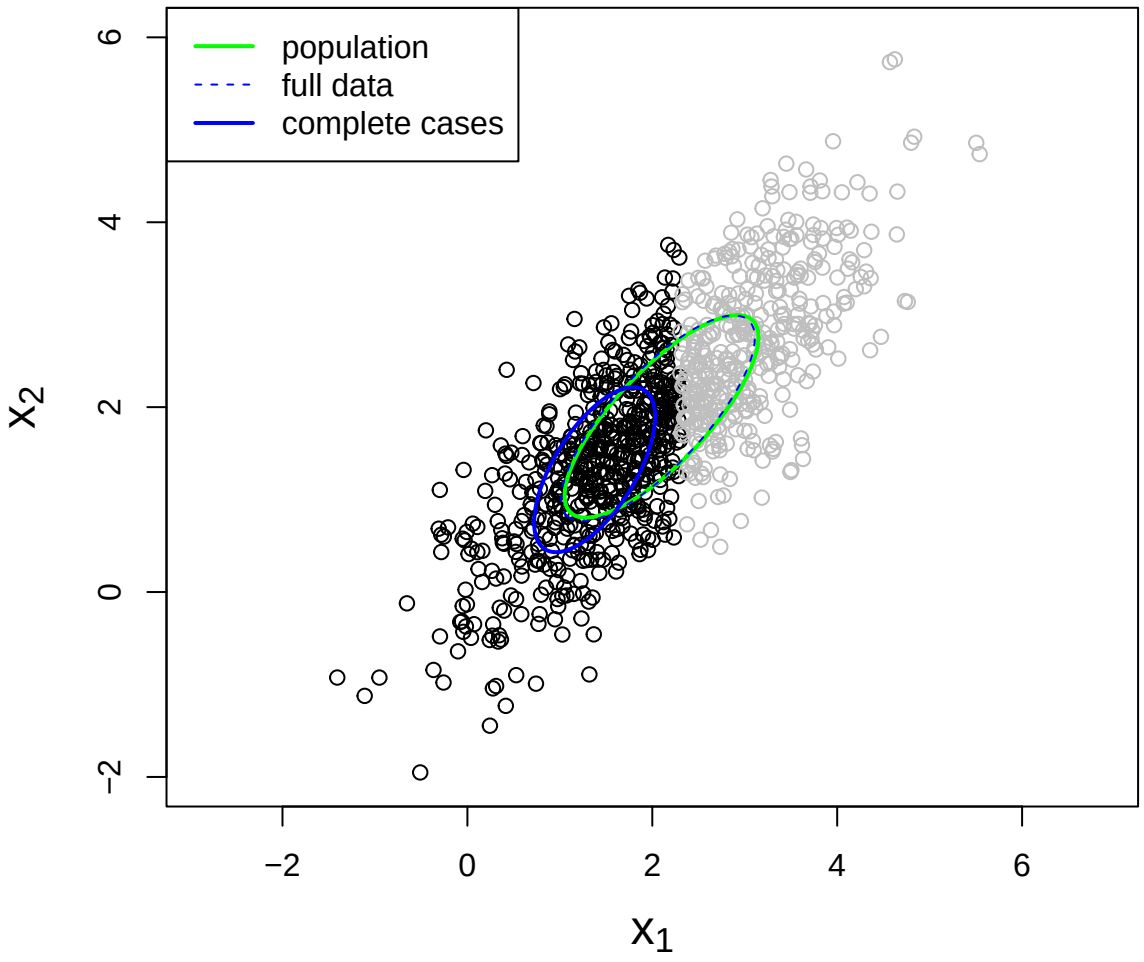
```
mu <- c(2.1,1.9)
cutoff <- 2.3
Xf <- rmvnorm(N, mean = mu, sigma = Sigma)   # full data set
dim(Xf)
```

```
## [1] 1000    2
```

```
head(Xf)
```

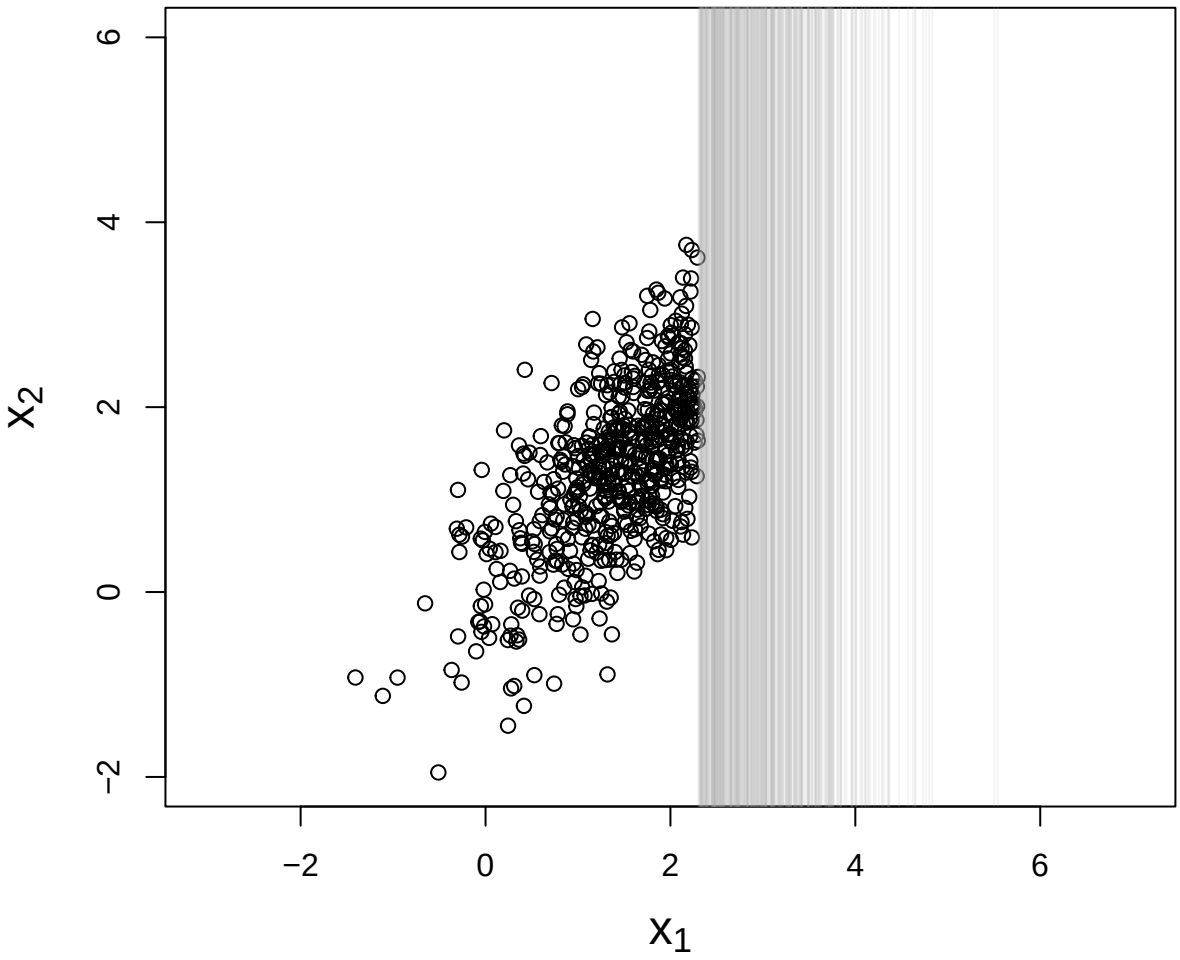
```
##           [,1]       [,2]
## [1,] 4.1458202  3.9086654
## [2,] 2.7078360  1.7887896
## [3,] 0.3633627 -0.5145991
## [4,] 1.5405261  1.5486942
## [5,] 0.5832166 -0.2397467
## [6,] 3.2016066  2.8082384
```

```r
par(mar = c(5,5,4,2) + .1)
xlab <- list(TeX('$x_1$'), cex = 1.5)
ylab <- list(TeX('$x_2$'), cex = 1.5)

plot(Xf, xlim = c(-2,6), ylim = c(-2,6), asp = 1, type = 'n',
     xlab = xlab,
     ylab = ylab
)
Xc <- Xf[Xf[,1] < cutoff,]          # both variables available: complete cases
Xm <- Xf[Xf[,1] >= cutoff,]         # only x1 available
Xm[,2] <- NA

X <- rbind(Xc, Xm)


points(Xc)
points(Xf[Xf[,1] >= cutoff,], col = 'gray')

n <- nrow(Xf)
nc <- nrow(Xc)
nm <- n - nc

lines(ell(mu, Sigma), col = 'green', lwd = 2)   # 'true' population parameters
lines(dell(Xf), col = 'blue', lwd = 1, lty = 2)           # estimated from full data
lines(dell(Xc), col = 'blue', lwd = 2, lty = 1)           # estimated from complete cases

legend('topleft', legend = c('population','full data', 'complete cases'),
       lwd = c(2,1,2), lty = c(1,2,1), col = c('green','blue','blue'))
```



What we actually have:

```
plot(Xf, xlim = c(-2,6), ylim = c(-2,6), asp = 1, type = 'n',
     xlab = xlab,
     ylab = ylab
)
points(Xc)
abline(v = Xm[,1], col = '#AAAAAA22')
```



What if we don't get y, only x, if x > 2.3

```
em_step <- function(X, theta_t) {
  #
  # Assumes X is n x 2 with some (at least 2) missing values in 2nd column
  # Validity assumes probability of missingness is a function of x1
  # theta_t is a list of 2 elements:
  #   mu_t is a vector of length 2 with last value of mu
  #   Sigma_t is a 2x2 positive definite matrix with last value of Sigma
  #
  #   - TODO: function needs to be robustified to deal with one or none missing values
  #
  mu_t <- theta_t[[1]]
  Sigma_t <- theta_t[[2]]
  #
  # E Step
  #
  Xc <- X[!is.na(X[,2]),]
  Xm <- X[is.na(X[,2]),]

  nc <- nrow(Xc)
  nm <- nrow(Xm)

  e2.1 <- mu_t[2] + (Sigma_t[1,2]/Sigma_t[1,1])*(Xm[,1] - mu_t[1])
```

```
  s_m <- c(sum(Xm[,1]), sum(e2.1))

  S_m_12 <- sum(Xm[,1] * e2.1)
  S_m_11 <- sum(Xm[,1]^2)
  S_m_22 <- sum(e2.1^2) + nm * (Sigma_t[2,2] - Sigma_t[2,1]^2/Sigma_t[1,1])
  S_m <- cbind(c(S_m_11, S_m_12), c(S_m_12, S_m_22) )

  SSCP <- crossprod(Xc) + S_m
  sx <- colSums(Xc) + s_m

  #
  # M Step
  #

  mu_hat <- sx / n                          # MLE of mu
  Sigma_hat <- (SSCP - outer(sx,sx)/n)/n    # MLE of Sigma

  list(mu_hat, Sigma_hat)

}

theta <- list(c(0,0), diag(2))
{
  (theta <- em_step(X, theta))
}
```

```
## [[1]]
## [1] 2.0618208 0.7930248
##
## [[2]]
##            [,1]       [,2]
## [1,]  1.0963574 -0.3201657
## [2,] -0.3201657  1.2985775
```

```
em_step(X, list(c(0,0), diag(2)))
```

```
## [[1]]
## [1] 2.0618208 0.7930248
##
## [[2]]
##            [,1]       [,2]
## [1,]  1.0963574 -0.3201657
## [2,] -0.3201657  1.2985775
```

Let's add a stopping rule

```
em_bivariate_normal <- function(X, start, eps = 10e-7, maxiter = 100, verbose = FALSE) {
  theta_t <- start
  iter <- 0
  crit <- 1 + eps
  if(verbose) print(theta_t)
  while(crit > eps && iter < maxiter) {
    theta_tp1 <- em_step(X, theta_t)
    crit <- sum((unlist(theta_tp1) - unlist(theta_t))^2)    # very crude
    theta_t <- theta_tp1
```

```r
    iter <- iter + 1
    if(verbose) {
      cat(iter,":", crit, unlist(theta_t), '\n')
    }
  }
  if(iter == maxiter) warning("maxiter reached")
  theta_t
}

em_bivariate_normal(X, list(c(0,0), diag(2)), verbose = T)
```

```
## [[1]]
## [1] 0 0
##
## [[2]]
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
##
## 1 : 5.183439 2.061821 0.7930248 1.096357 -0.3201657 -0.3201657 1.298577
## 2 : 0.1181229 2.061821 0.9910866 1.096357 -0.157672 -0.157672 1.137066
## 3 : 0.09774476 2.061821 1.131383 1.096357 0.00651348 0.00651348 0.9816698
## 4 : 0.06444515 2.061821 1.249149 1.096357 0.147836 0.147836 0.8785576
## 5 : 0.04197033 2.061821 1.349316 1.096357 0.2682497 0.2682497 0.8243539
## 6 : 0.02858308 2.061821 1.434593 1.096357 0.3707757 0.3707757 0.8073894
## 7 : 0.02058581 2.061821 1.507197 1.096357 0.4580672 0.4580672 0.8160394
## 8 : 0.01548733 2.061821 1.569013 1.096357 0.5323875 0.5323875 0.8409219
## 9 : 0.0119471 2.061821 1.621643 1.096357 0.5956641 0.5956641 0.8751173
## 10 : 0.009308064 2.061821 1.666452 1.096357 0.649538 0.649538 0.9137873
## 11 : 0.007255422 2.061821 1.704603 1.096357 0.6954065 0.6954065 0.9536883
## 12 : 0.005630161 2.061821 1.737085 1.096357 0.7344591 0.7344591 0.9927379
## 13 : 0.004340137 2.061821 1.764741 1.096357 0.7677087 0.7677087 1.029674
## 14 : 0.00332167 2.061821 1.788286 1.096357 0.7960175 0.7960175 1.063798
## 15 : 0.002524459 2.061821 1.808333 1.096357 0.8201197 0.8201197 1.094794
## 16 : 0.001906263 2.061821 1.825401 1.096357 0.8406404 0.8406404 1.122593
## 17 : 0.001431202 2.061821 1.839933 1.096357 0.8581119 0.8581119 1.147281
## 18 : 0.001069129 2.061821 1.852305 1.096357 0.8729871 0.8729871 1.169041
## 19 : 0.0007951713 2.061821 1.862839 1.096357 0.8856519 0.8856519 1.188105
## 20 : 0.0005891894 2.061821 1.871808 1.096357 0.8964348 0.8964348 1.204724
## 21 : 0.0004351565 2.061821 1.879444 1.096357 0.9056154 0.9056154 1.219156
## 22 : 0.0003205052 2.061821 1.885945 1.096357 0.9134318 0.9134318 1.231648
## 23 : 0.0002355045 2.061821 1.89148 1.096357 0.9200868 0.9200868 1.242432
## 24 : 0.0001726984 2.061821 1.896193 1.096357 0.9257528 0.9257528 1.251721
## 25 : 0.0001264248 2.061821 1.900206 1.096357 0.9305769 0.9305769 1.259707
## 26 : 9.241462e-05 2.061821 1.903622 1.096357 0.9346841 0.9346841 1.266563
## 27 : 6.746963e-05 2.061821 1.90653 1.096357 0.938181 0.938181 1.272441
## 28 : 4.920571e-05 2.061821 1.909007 1.096357 0.9411583 0.9411583 1.277476
## 29 : 3.585343e-05 2.061821 1.911115 1.096357 0.9436932 0.9436932 1.281784
## 30 : 2.61043e-05 2.061821 1.91291 1.096357 0.9458514 0.9458514 1.285467
## 31 : 1.899369e-05 2.061821 1.914438 1.096357 0.9476889 0.9476889 1.288614
## 32 : 1.381227e-05 2.061821 1.91574 1.096357 0.9492533 0.9492533 1.291302
## 33 : 1.003956e-05 2.061821 1.916848 1.096357 0.9505853 0.9505853 1.293596
## 34 : 7.294392e-06 2.061821 1.917791 1.096357 0.9517194 0.9517194 1.295554
## 35 : 5.29803e-06 2.061821 1.918594 1.096357 0.9526849 0.9526849 1.297224
```

```
## 36 : 3.846916e-06 2.061821 1.919278 1.096357 0.953507 0.953507 1.298648
## 37 : 2.792563e-06 2.061821 1.91986 1.096357 0.9542069 0.9542069 1.299862
## 38 : 2.026755e-06 2.061821 1.920355 1.096357 0.9548028 0.9548028 1.300897
## 39 : 1.47069e-06 2.061821 1.920777 1.096357 0.9553101 0.9553101 1.301779
## 40 : 1.067025e-06 2.061821 1.921137 1.096357 0.9557421 0.9557421 1.30253
## 41 : 7.740535e-07 2.061821 1.921443 1.096357 0.9561099 0.9561099 1.30317
```

```
## [[1]]
## [1] 2.061821 1.921443
##
## [[2]]
##           [,1]      [,2]
## [1,] 1.0963574 0.9561099
## [2,] 0.9561099 1.3031703
```

```
em_bivariate_normal(X, list(c(0,0), diag(2)))
```

```
## [[1]]
## [1] 2.061821 1.921443
##
## [[2]]
##           [,1]      [,2]
## [1,] 1.0963574 0.9561099
## [2,] 0.9561099 1.3031703
```

```
em_bivariate_normal(X, list(c(0,0), diag(2)), maxiter = 10)
```

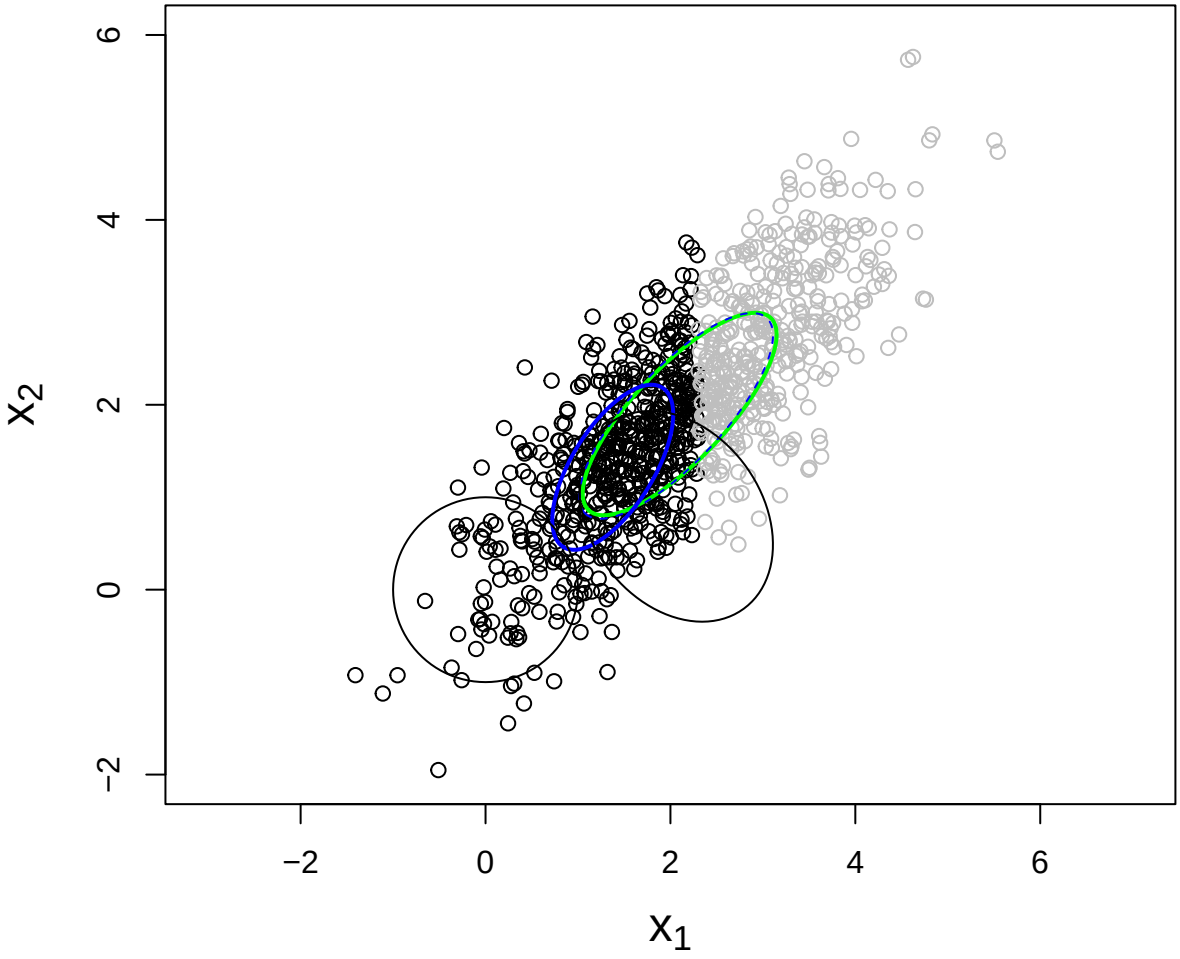
```
## Warning in em_bivariate_normal(X, list(c(0, 0), diag(2)), maxiter = 10):
## maxiter reached
```

```
## [[1]]
## [1] 2.061821 1.666452
##
## [[2]]
##          [,1]      [,2]
## [1,] 1.096357 0.6495380
## [2,] 0.649538 0.9137873
```

Visualizing the process

```
{
  plot(Xf, xlim = c(-2,6), ylim = c(-2,6), asp = 1, type = 'n',
       xlab = xlab,
       ylab = ylab
  )

  points(Xc)
  points(Xf[Xf[,1] >= cutoff,], col = 'gray')

  lines(ell(mu, Sigma), col = 'green', lwd = 2)   # 'true' population parameters
  lines(dell(Xf), col = 'blue', lwd = 1, lty = 2)           # estimated from full data
  lines(dell(Xc), col = 'blue', lwd = 2, lty = 1)           # estimated from complete cases
}

theta_0 <- list(c(0,0), diag(2))
```

```
theta_0 <- list(c(-2,6), diag(2))
theta_0 <- list(c(4,4), cbind(c(4,-3),c(-3,2)))
theta_0 <- list(c(0,0), diag(2))
theta_0 <- list(c(0,0), diag(2))

theta_t <- theta_0
lines(do.call(ell,theta_0))
{ # iterate
  theta_t <- em_step(X, theta_t)
  lines(do.call(ell, theta_t))
}
```



Won't work if missing Y depends on Y not through X

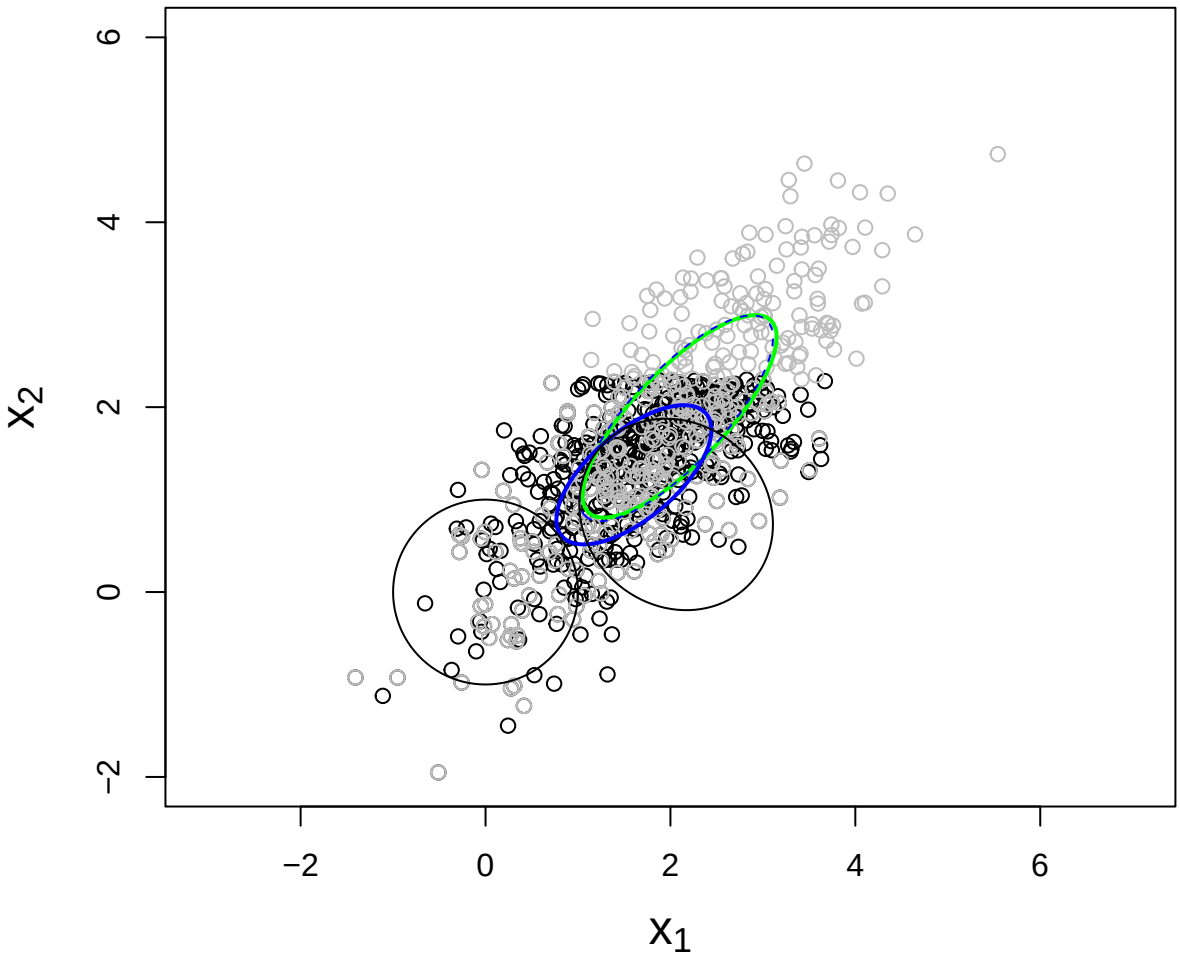
Missingness not 'at random' given X_observed, i.e. probability of missingness not a function of X_observed

```
head(Xf)
```

```
##           [,1]       [,2]
## [1,] 4.1458202  3.9086654
## [2,] 2.7078360  1.7887896
## [3,] 0.3633627 -0.5145991
## [4,] 1.5405261  1.5486942
```

```
## [5,] 0.5832166 -0.2397467
## [6,] 3.2016066  2.8082384
```

```
Xnmar <- Xf
Xnmar[Xnmar[,2] > 2.3,2] <- NA
Xc <- Xnmar[!is.na(Xnmar[,2]),]

{
  plot(Xf, xlim = c(-2,6), ylim = c(-2,6), asp = 1, type = 'n',
       xlab = xlab,
       ylab = ylab
  )

  points(Xnmar)
  points(Xf[is.na(X[,2]),], col = 'gray')

  lines(ell(mu, Sigma), col = 'green', lwd = 2)   # 'true' population parameters
  lines(dell(Xf), col = 'blue', lwd = 1, lty = 2)           # estimated from full data
  lines(dell(Xc), col = 'blue', lwd = 2, lty = 1)           # estimated from complete cases
}


theta_0 <- list(c(0,0), diag(2))

theta_t <- theta_0
lines(do.call(ell,theta_0))
{ # iterate
  theta_t <- em_step(Xnmar, theta_t)
  lines(do.call(ell, theta_t))
}
```



Alternative analysis using mixed models

```
head(X)
```

```
##             [,1]       [,2]
## [1,]  0.36336271 -0.5145991
## [2,]  1.54052607  1.5486942
## [3,]  0.58321658 -0.2397467
## [4,] -0.02960784  0.5586251
## [5,]  2.22090465  1.3464217
## [6,]  2.04887101  2.0036473
```

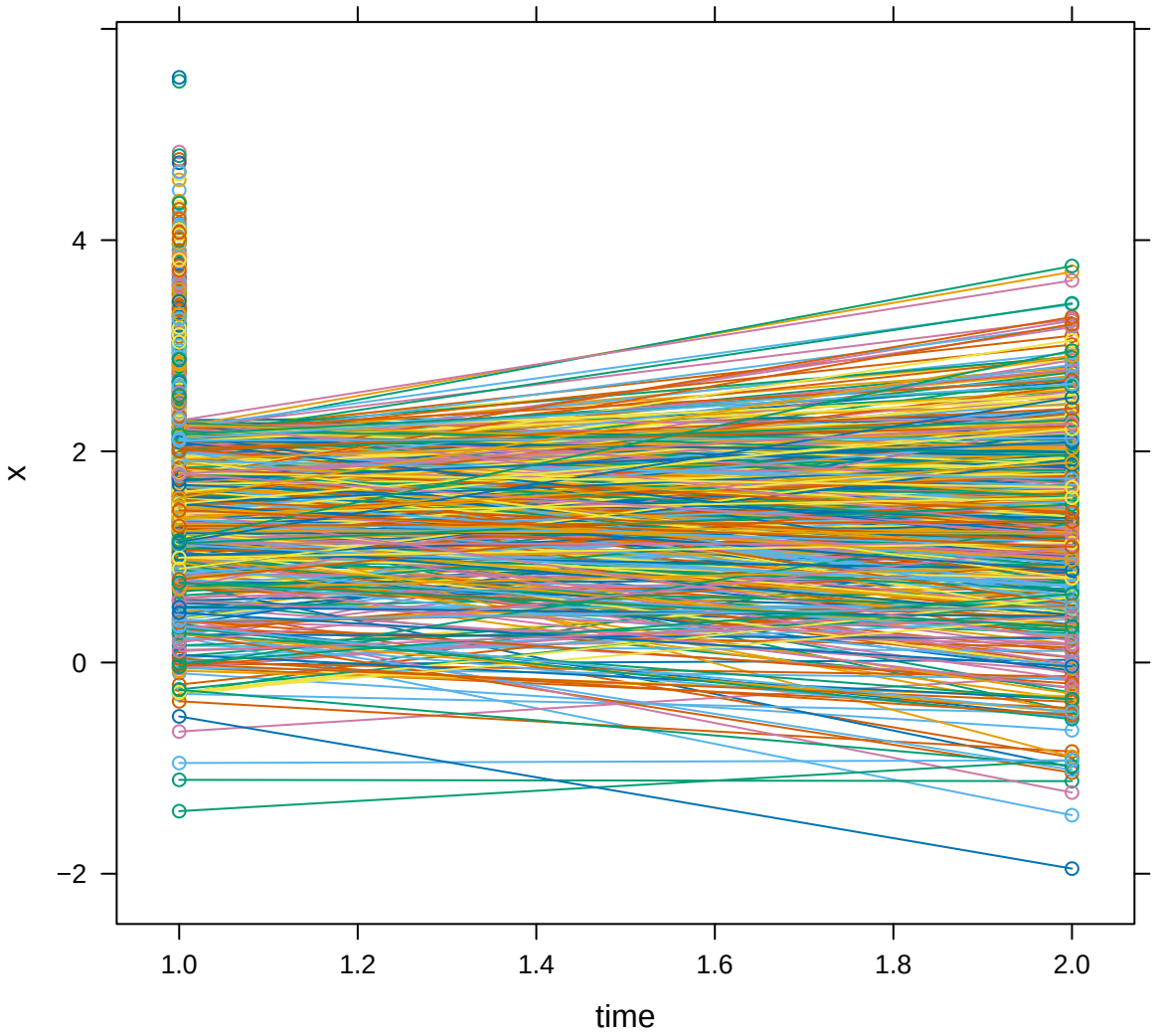
```
dd <- data.frame(x_1 = X[,1], x_2 = X[,2])
dd$id <- 1:nrow(dd)
head(dd)
```

```
##           x_1        x_2 id
## 1  0.36336271 -0.5145991  1
## 2  1.54052607  1.5486942  2
## 3  0.58321658 -0.2397467  3
## 4 -0.02960784  0.5586251  4
## 5  2.22090465  1.3464217  5
## 6  2.04887101  2.0036473  6
```

```
library(spida2)
#
# transform data to long form
#
dl <- tolong(dd)
head(dl)
```

```
##     id time           x
## 1.1  1    1  0.36336271
## 2.1  2    1  1.54052607
## 3.1  3    1  0.58321658
## 4.1  4    1 -0.02960784
## 5.1  5    1  2.22090465
## 6.1  6    1  2.04887101
```

```
library(lattice)
xyplot(x ~ time, dl, groups = id, type = 'b')
```




```
library(nlme)
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:spida2':
##
##     getData
```

```
fit <- gls(x ~ time, dl,
           correlation = corSymm(form = ~ time| id),
           weights = varIdent(form = ~ 1 | time), na.action = na.omit)

summary(fit)
```

```
## Generalized least squares fit by REML
##   Model: x ~ time
##   Data: dl
##       AIC      BIC   logLik
##   4196.98 4223.859 -2093.49
##
## Correlation Structure: General
##  Formula: ~time | id
##  Parameter estimate(s):
##  Correlation:
##   1
## 2 0.8
## Variance function:
##  Structure: Different standard deviations per stratum
##  Formula: ~1 | time
##  Parameter estimates:
##        1        2
## 1.000000 1.091913
##
## Coefficients:
##                  Value  Std.Error  t-value p-value
## (Intercept)  2.2004481 0.04665117 47.16812       0
## time        -0.1386273 0.02832520 -4.89413       0
##
##  Correlation:
##      (Intr)
## time -0.712
##
## Standardized residuals:
##        Min         Q1        Med         Q3        Max
## -3.3872276 -0.8074442 -0.2489592  0.4084822  3.3212358
##
## Residual standard error: 1.047594
## Degrees of freedom: 1599 total; 1597 residual
```

```
L <- rbind(
  mu1 <- c(1,1),
  mu2 <- c(1,2)
)

wald(fit, L)
```

```
##   numDF denDF  F-value p-value
## 1     2  1596 1946.159 <.00001
##      Estimate Std.Error DF   t-value  p-value Lower 0.95 Upper 0.95
```

```
## [1,] 2.061821 0.033128  1596 62.23835 <.00001 1.996842   2.126799
## [2,] 1.923193 0.040289  1596 47.73490 <.00001 1.844168   2.002218
```

```
getV(fit)
```

```
## Marginal variance covariance matrix
##         [,1]    [,2]
## [1,] 1.09750 0.95917
## [2,] 0.95917 1.30850
##   Standard Deviations: 1.0476 1.1439
```

```
#
# Compare with EM algorithm
#
em_bivariate_normal(X, list(c(0,0), diag(2)))
```

```
## [[1]]
## [1] 2.061821 1.921443
##
## [[2]]
##           [,1]      [,2]
## [1,] 1.0963574 0.9561099
## [2,] 0.9561099 1.3031703
```