Latent class analysis

Latent profile analysis and the EM algorithm

L Boeschoten & DL Oberski

Learning goals

- Explain how the EM algorithm works
- Understand and run R code implementing EM for a two-component mixture of univariate Gaussians
- Explain the key assumption that makes LPA possible

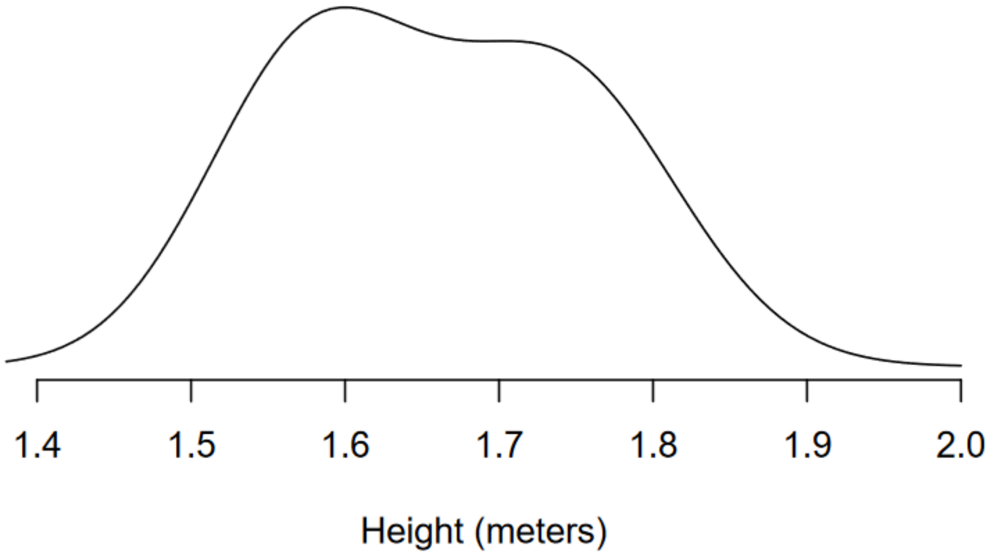
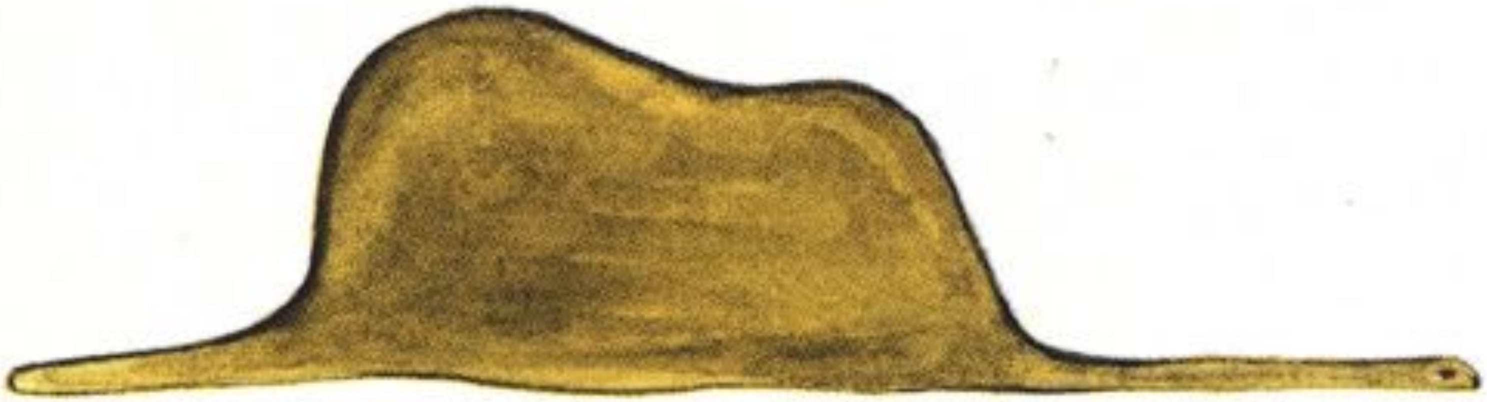
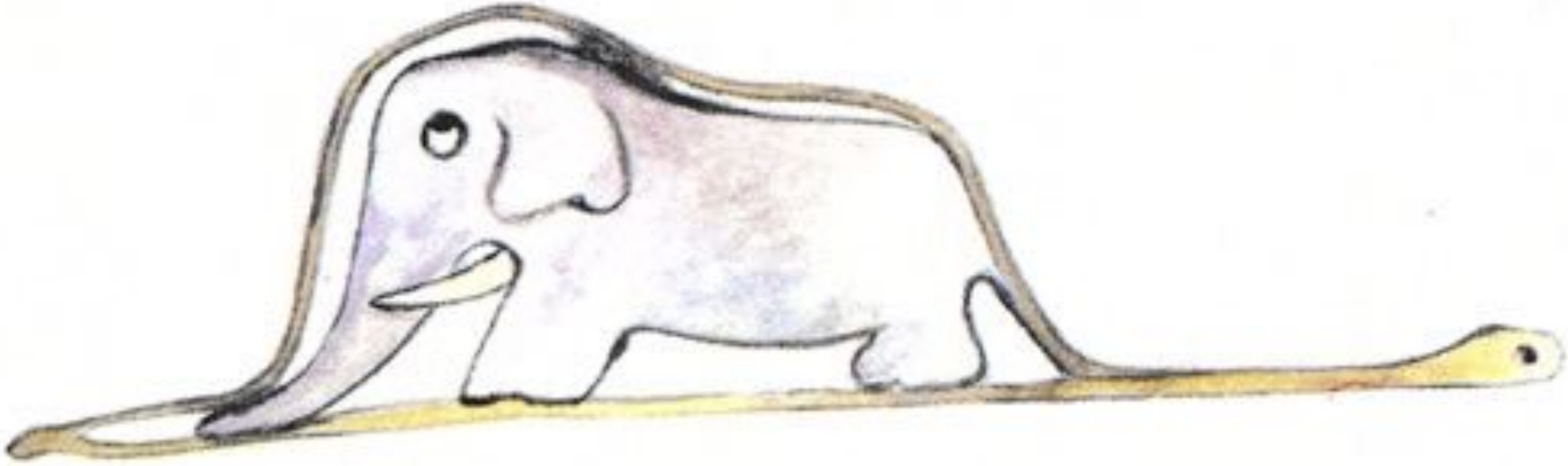


Fig. 1 Peoples' height.

Le Petit Prince



Mon dessin ne représentait pas un chapeau. Il représentait un serpent boa qui digérait un éléphant



J'ai alors dessiné
l'intérieur du serpent boa, afin que les grandes personnes puissent comprendre. Elles ont toujours besoin d'explications

There is a much quoted story about David Hilbert, who one day noticed that a certain student had stopped attending class. When told that the student had decided to drop mathematics to become a poet, Hilbert replied,

> "Good — he did not have enough imagination to become a mathematician."

— Robert Osserman (US mathematician, 1926 – 2011)

Gaussian mixture modeling

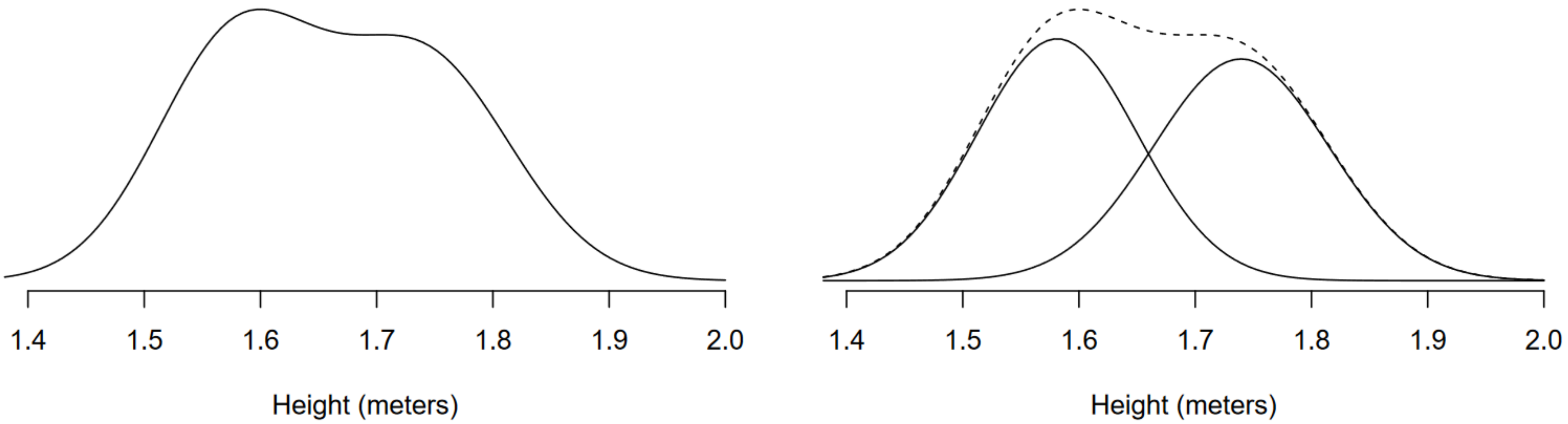


Fig. 1 Peoples' height. Left: observed distribution. Right: men and women separate, with the total shown as a dotted line.

How data were generated

```
n <- 1000

true_mean_men     <- 1.74#m
true_mean_women   <- 1.58#m
true_sd_men       <- 0.08#m
true_sd_women     <- 0.07#m

true_sex <- rbinom(n, size = 1, prob = 0.5) + 1
```

How data were generated

```
means_mix <- c(true_mean_men,
                          true_mean_women)[true_sex]

sds_mix <- c(true_sd_men, true_sd_women)[true_sex]

height <- rnorm(n, means_mix, sds_mix)
```

Gaussian mixture modeling

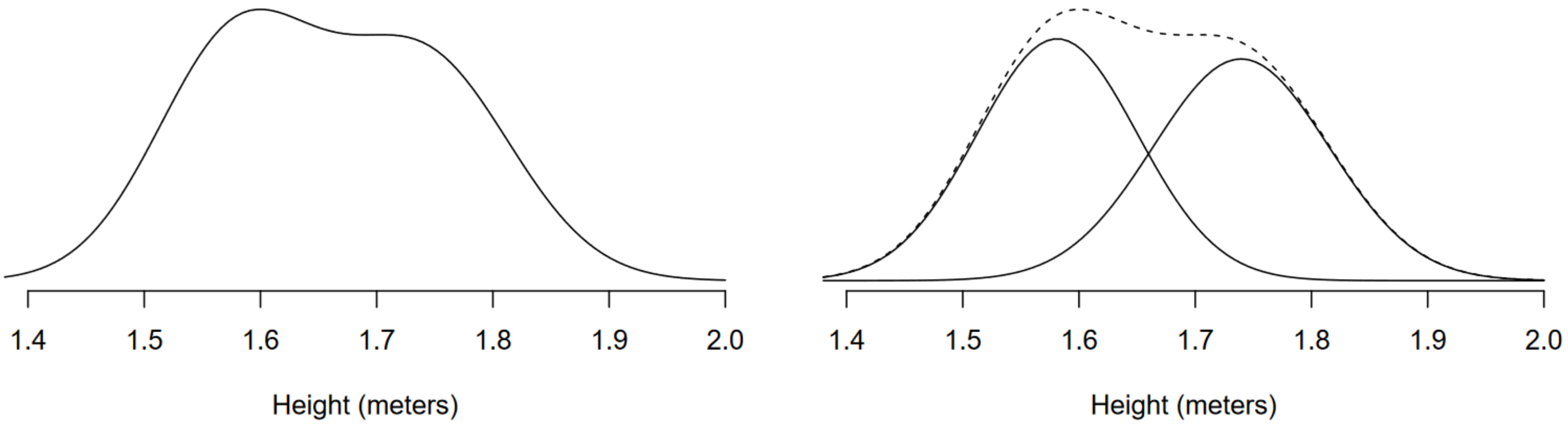


Fig. 1 Peoples' height. Left: observed distribution. Right: men and women separate, with the total shown as a dotted line.

What mixture modeling does

- Given only the **observed data** (of course!),
- can we recover the values:

```
true_mean_men     <- 1.74#m
true_mean_women   <- 1.58#m
true_sd_men       <- 0.08#m
true_sd_women     <- 0.07#m
```

?

"Maximum likelihood"

- Statistical model = **assumptions** defines a likelihood

$$p(\text{data} \mid \text{parameters}) = p(y \mid \theta)$$

- Maximum likelihood estimation: find the parameters θ that make it most likely to observe the data we actually observed, y

- The above procedure automatically gives algorithm for computing clusters from data, given the model.

Univariate mixture of Gaussians

Likelihood (*density*) for height data:

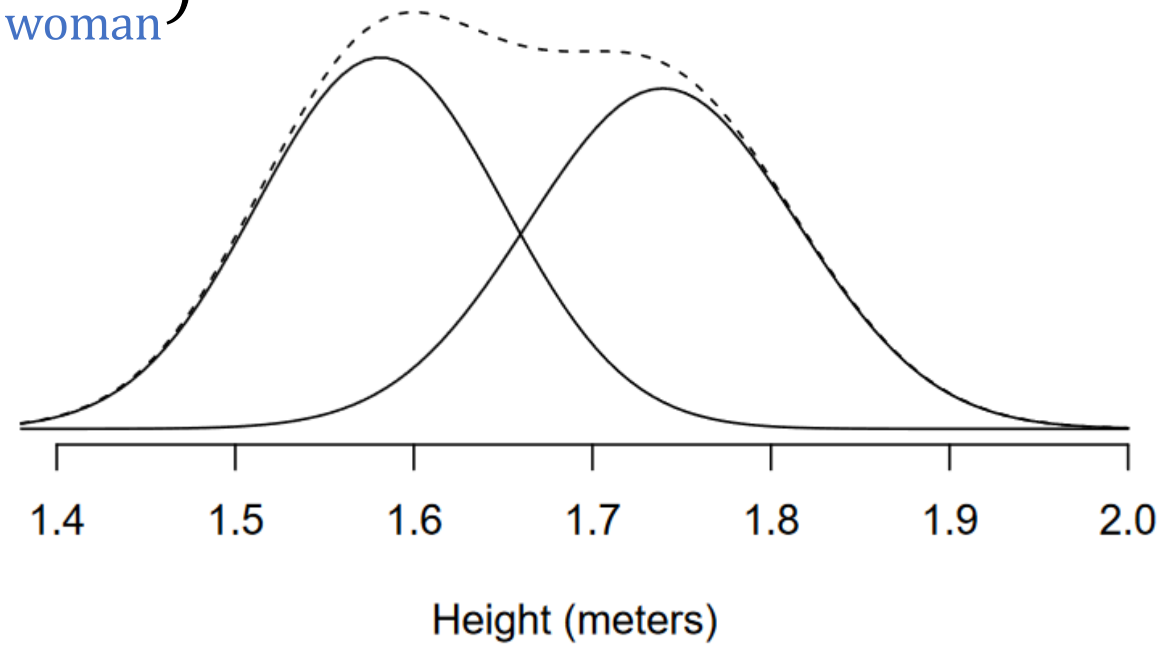
$p(\text{height} \mid \theta) =$

$$\Pr(\text{man}) \cdot \text{Normal}(\mu_{\text{man}}, \sigma_{\text{man}}) +$$
$$\Pr(\text{woman}) \cdot \text{Normal}(\mu_{\text{woman}}, \sigma_{\text{woman}})$$

Or, more concise notation:

$p(\text{height} \mid \theta) =$

$$\pi_1^X \text{Normal}(\mu_1, \sigma_1) +$$
$$(1 - \pi_1^X)\text{Normal}(\mu_2, \sigma_2)$$



Mixture model

Gaussian mixture model **parameters**:

- π_1^X determines the relative cluster sizes
  - Proportion of observations to be expected in each cluster
- μ_1 and μ_2 determine the locations of the clusters
  - Like centroids in K-means clustering
- σ_1 and σ_2 determine the volume of the clusters
  - how large / spread out the are clusters are in data space

Together, these 5 unknown parameters describe our model of how the data is generated.

Plan of attack: direct maximum likelihood

- Write down how we think the data were generated (*imagination step*)
  → ***model***
- This involves **parameters** (the means, sds, and class sizes)
- Try out some values for the parameters (*exploration step)*
- Work out what data *would* look like, if our model were right, and the parameters had the values we are currently guessing
- Compare to the real data (*evaluation step*)
- Pick the guessed parameter values that would give data closest to the observed data (*optimization step*)

Why not this plan of attack?

- It gives a general solution to the problem, BUT
- The exploration space is large
- **Difficult** to find a way that always gives the right answer in the end
- Model can be complicated, no ready-made solutions

Estimation: the EM algorithm

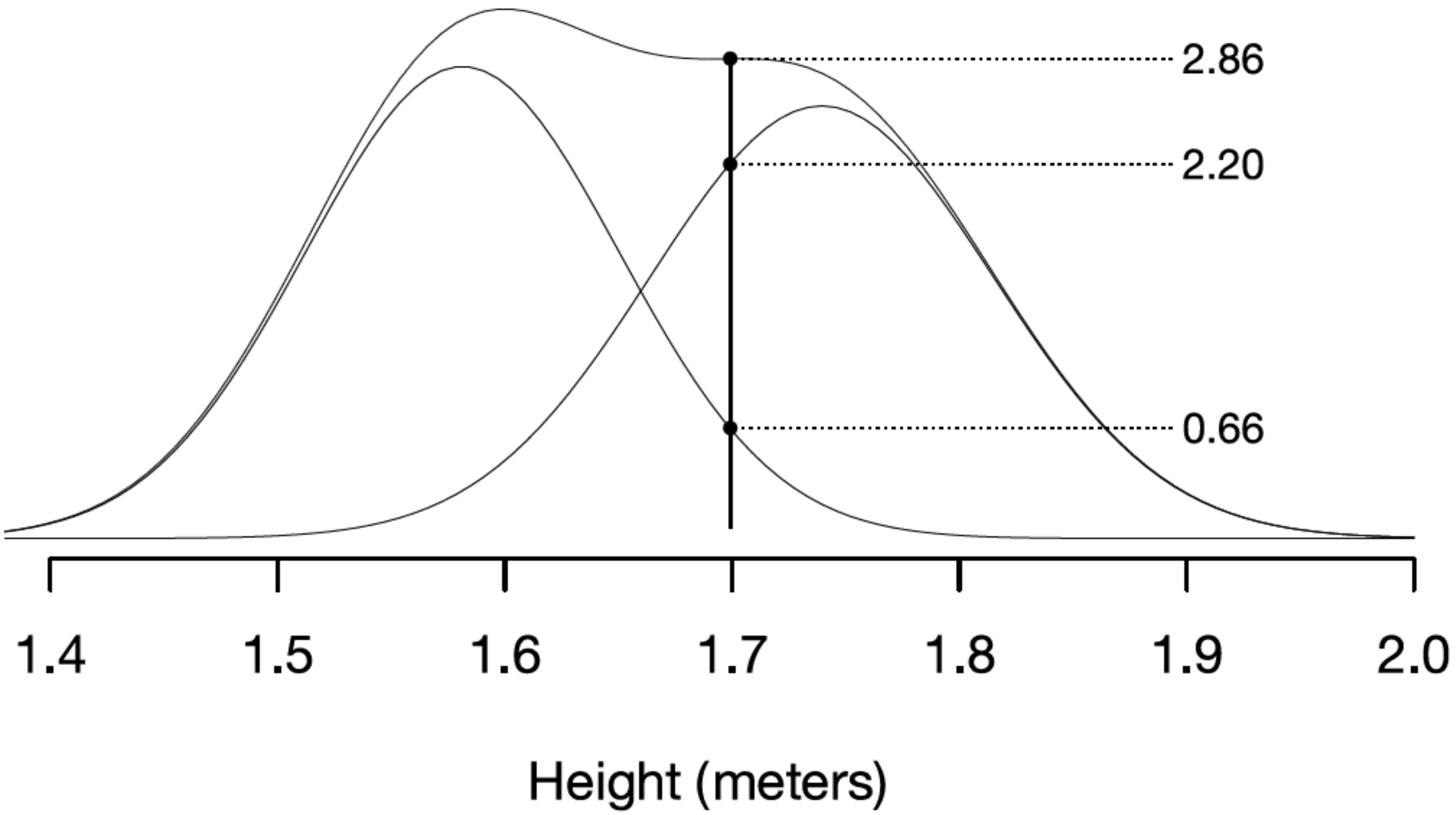
- If we knew in advance who is a man and who is a woman, it would have been easy to find the estimates for μ and σ:

$$\hat{\mu}_1 = \frac{\sum_{i=1}^{N_1} \mathtt{height}_i}{N_1}, \qquad \hat{\sigma}_1 = \sqrt{\frac{\sum_{i=1}^{N_1} (\mathtt{height}_i - \hat{\mu}_1)^2}{N_1}}$$

(and same for $\hat{\mu}_2$ and $\hat{\sigma}_2$.)

- But we don't know this!

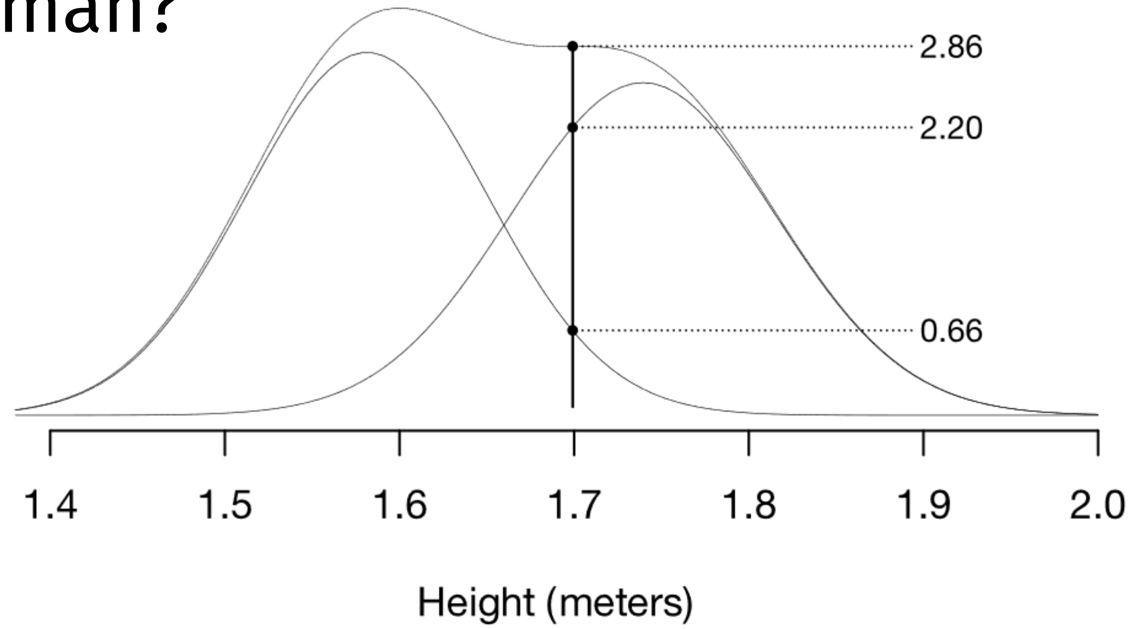
-> **Assignments need to be estimated** too.


2.86
2.20
0.66
1.4
1.5
1.6
1.7
1.8
1.9
2.0
Height (meters)

Estimation: the EM algorithm

- Solution: Figure out the posterior probability of being a man/woman, given the current estimates of the means and sds
- If we know cluster locations and shapes, how likely is it that a 1.7m person is a man or a woman?

$$\pi^X_{man} = \frac{2.20}{2.86} \approx 0.77$$



Estimation: the Expectation-Maximization (EM) algorithm

- Now we have some class *assignments* (probabilities);
- So we can go back to the parameters and update them using our easy rule (M-step)
- Then, we can compute new posterior probabilities (E-step)

Estimation: the EM algorithm

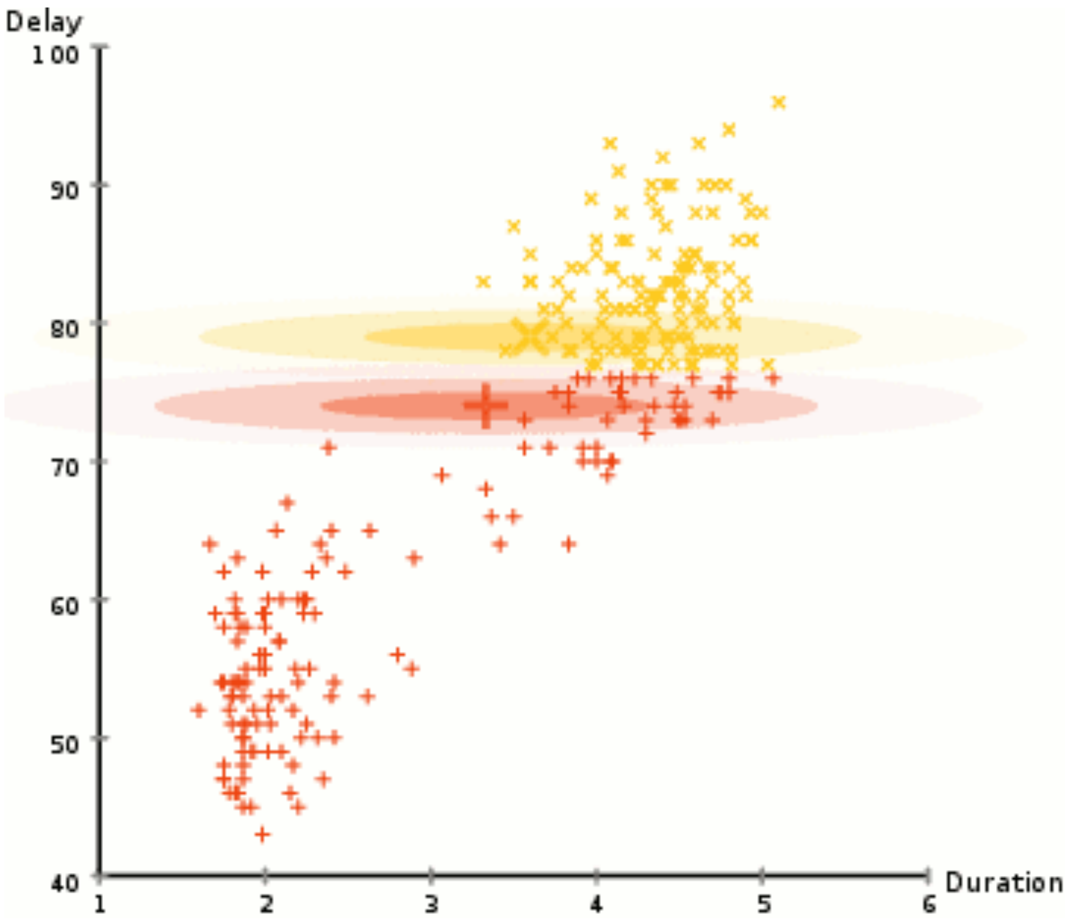
(0) Guess the parameters

"E-step" (1) Work out posterior of being M/F (assuming normality)

"M-step" (2) Update the parameters

Stop when parameters stop changing

Estimation: the EM algorithm



R code: starting values

```
guess_mean_men <- mean(height) + sep_start
guess_mean_wom <- mean(height) - sep_start

guess_sd_men <- sd(height)
guess_sd_wom <- sd(height)
```

R code: iteration between E and M steps

```
for(it in 1:maxit) {
    # E-step

    # M-step
}
```

R code: E-step

```
pman <- dnorm(height, mean = guess_mean_men,
                      sd = guess_sd_men)

pwom <- dnorm(height, mean = guess_mean_wom,
                      sd = guess_sd_wom)

post <- pman / (pman + pwom)
```

R code: M-step for means

```
guess_mean_men <-
    weighted.mean(height, w = post)

guess_mean_wom <-
    weighted.mean(height, w = 1-post)
```

R code: M-step for sd's

```
guess_sd_men <- sqrt(weighted.mean(
    (height - guess_mean_men)^2, w = post))

guess_sd_wom <- sqrt(weighted.mean(
    (height - guess_mean_wom)^2, w = 1-post))
```

… that’s it!

Our simple code works

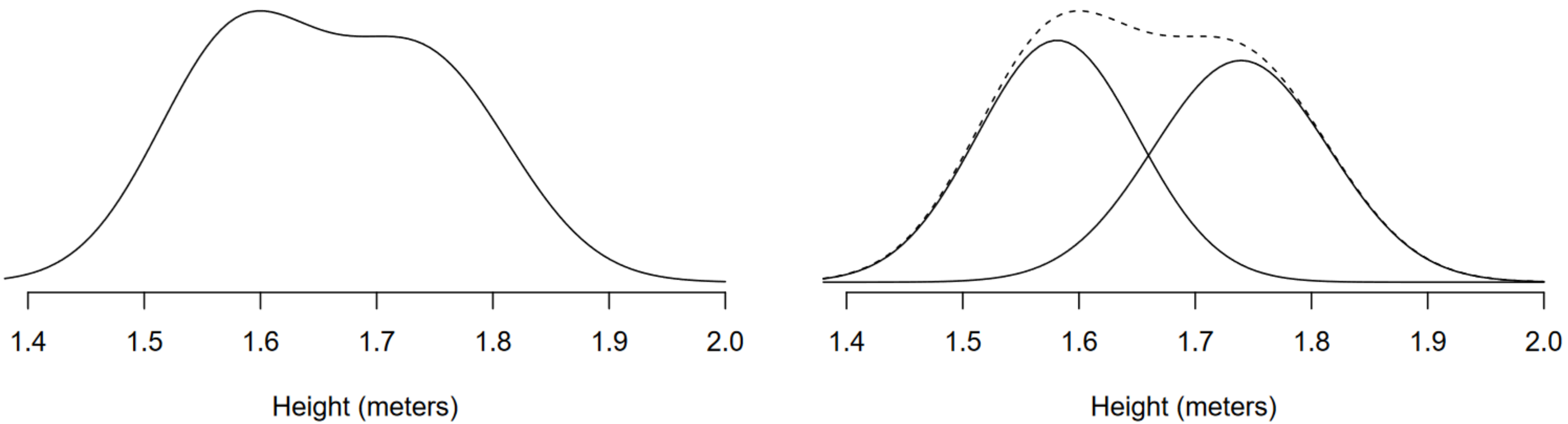
```
> runit()

Iter:    M:       F:       sd(M):   sd(F):
---------------------------------------
Start    1.86     1.46     0.107    0.107
1        1.74     1.58     0.072    0.066
2        1.74     1.58     0.072    0.066
3        1.74     1.58     0.073    0.066
```

What just happened?

Went from left to right.

MAGIC???



What just happened?

- We were able to achieve our goal: guess the parameters of the imagined model, using only observed data
- The price we paid was an **assumption**:

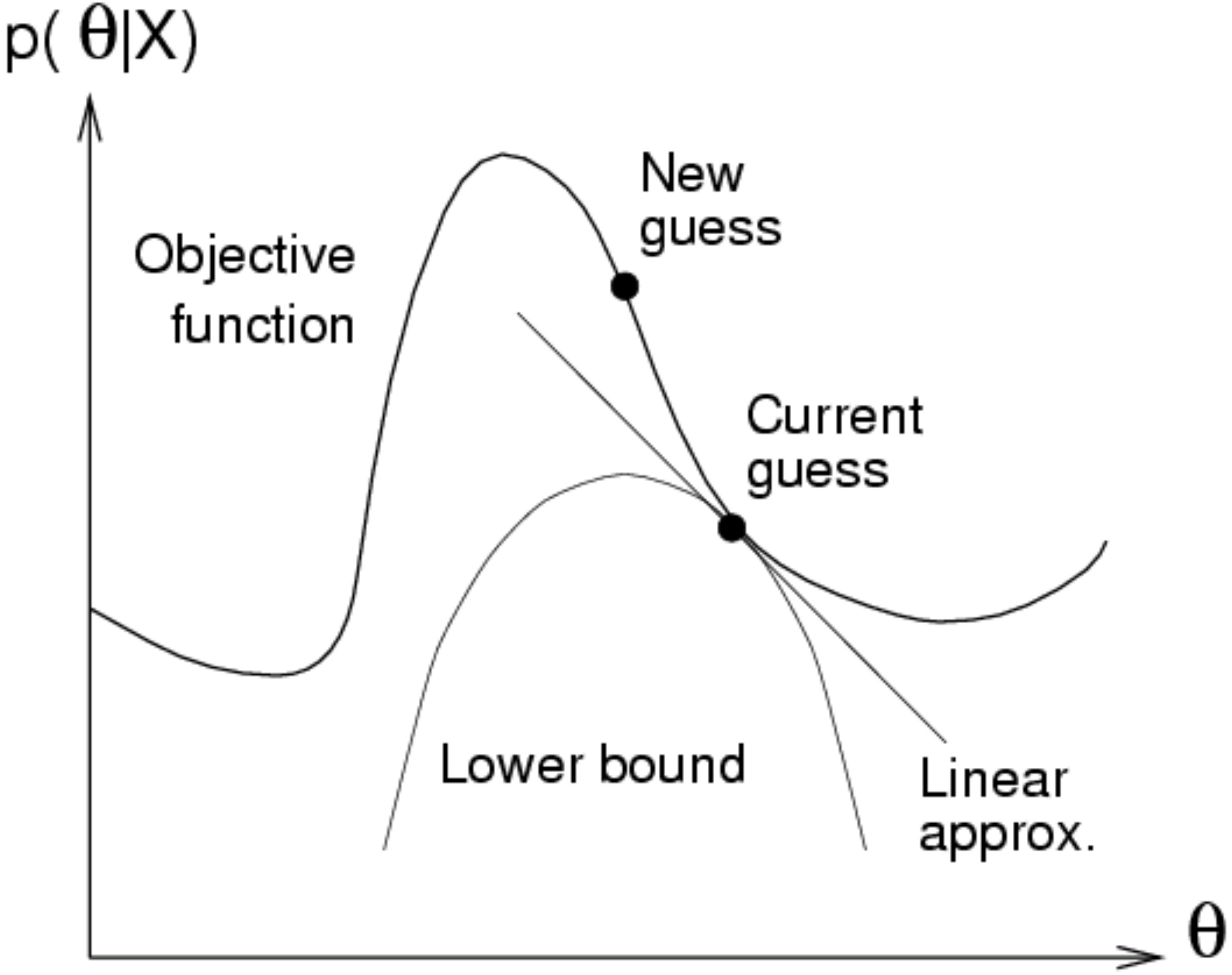
  Within each group, the distribution of heights is exactly Normal (Gaussian)
- By making this assumption, we could perform our E and M steps

(Mixture) modeling is not based on MAGIC but on ASSUMPTIONS

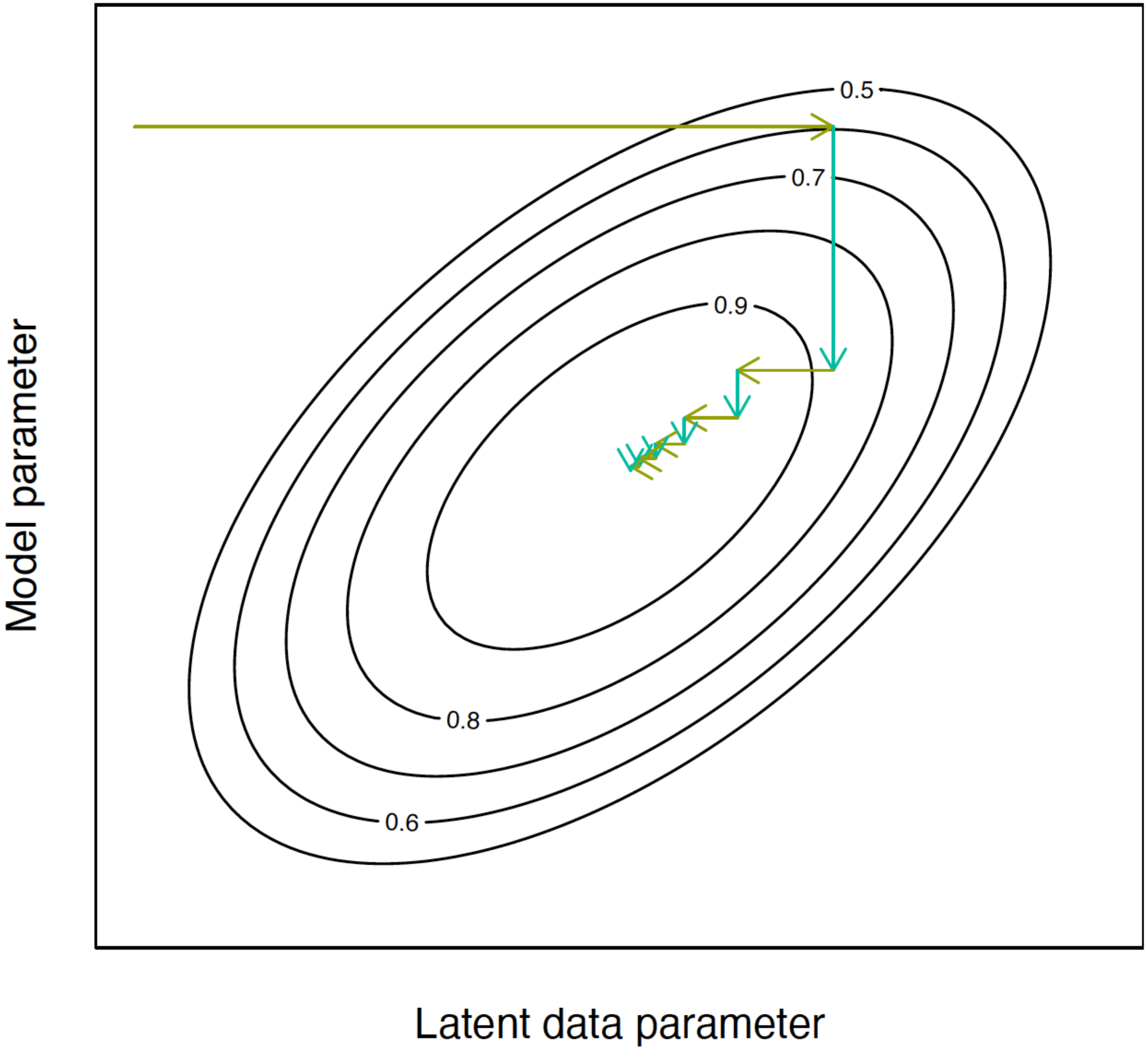
EM in general

- There is some “missing data” (in our case *class membership*)
- If we had the missing data, finding the parameter estimates would be easy
  - This easy rule is our “maximization” (M) step
- If we had the parameters, then estimating the missing data would be easy
  - This easy rule is our “expectation” (E) step

- Now we just alternate between easy rules – easy!



Source: Fatih Gelgi



Source: Bettina Grün

EM algorithm *Source*: Bettina Grün

Advantages:

- The likelihood is increased in each step → converges
- Relatively easy to implement:
- Different mixture models require only different M-steps.
- Weighted ML estimation of the component specific model is sometimes already available in standard software.

Disadvantages:

- Standard errors have to be determined separately
- Slow convergence.
- **Convergence only to a *local optimum*.**

Convergence to local optimum ☹

```
> library(flexmix)
> height_fit_fm <- flexmix(height ~ 1, k = 2)
> parameters(height_fit_fm)

                      Comp.1    Comp.2
coef.(Intercept) 1.6587925 1.6610152
sigma            0.1074166 0.1075208
```

EM algorithm

- The big disadvantage of EM is that does not always find the best solution
- Currently, the advice to avoid this problem is to try many different random starting values
- The solution with the best likelihood should be chosen
- And, ideally, should be found several times from different starting points

Multiple random starts solve problem ^^

```
> library(flexmix)
> height_fit_fm <-
    stepFlexmix(height ~ 1, k = 2, nrep = 50)
2 : * * * * * * * * * * * * * * * * * * * * *
> parameters(height_fit_flexmix)

                     Comp.1   Comp.2
coef.(Intercept) 1.73240449 1.569787
sigma            0.07717027 0.061939
```

The classic

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 1–38.

WILEY

The EM Algorithm and Extensions

Second Edition

Geoffrey J. McLachlan
Thriyambakam Krishnan

WWW.

WILEY SERIES IN PROBABILITY AND STATISTICS