Data Science for Economists

Lecture 14: Unsupervised Learning

Alex Marsh
University of North Carolina | ECON 390

Table of contents

Introduction

Motivation

We've already touched briefly on what unsupervised learning is.

These methods are the methods that y'all have likely not seen.

The idea of these methods is for when there's something unobserved you would like to either reduce or group together.

Unsupervised Learning

Unsupervised Learning

Unsupervised learning is a bit more challenging and less understood than supervised learning.

> We are not interested in prediction, because we do not have an associated response variable Y. Rather, the goal is to discover interesting things about the measurements on $X_1, X_2, \ldots, X_p$.

There is no simple goal and is often more subjective.

Is largely part of exploratory analysis.

There are two common unsupervised learning algorithms:

1. Principal Components Analysis (PCA)
2. Clustering

We are going to skip PCA.

Clustering

Clustering

Clustering refers to a broad set of techniques to find subgroups or "clusters" in data.

With clustering, we often have a reason to believe that there is "heterogeneity" among groups, but don't really know how to label this heterogeneity.

Clustering can be used to find these groups.

Example: Market segmentation.

There are two types of clustering:

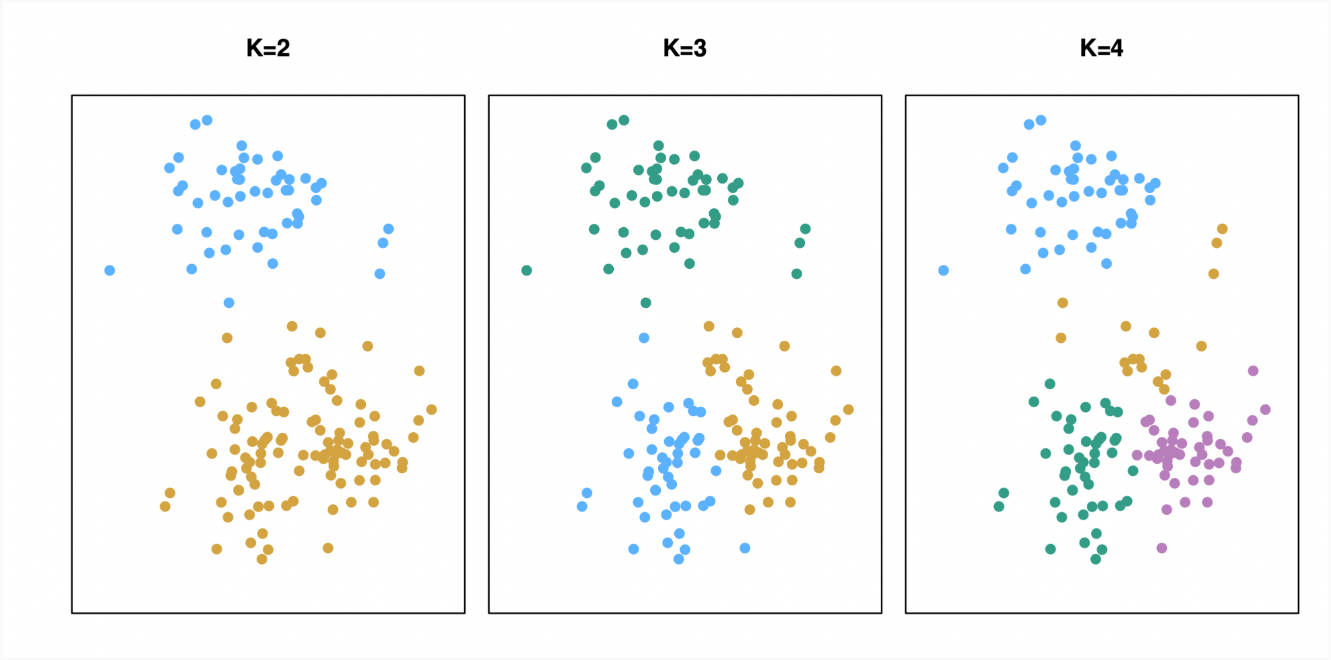
1. Hard Clustering: assign groups to observations with certainty (probability 1).
2. Soft Clustering: assign a non-degenerate probability of belonging to a group to observations.

K-Means Clustering

K-Means

K-means clustering is a simple and elegant approach for partitioning a data set into K distinct, non-overlapping clusters.

K must be specified and then each observation is assigned to one of the K groups.



K-means

Let $C_1, \ldots, C_K$ denote sets containing the indicies of the observations in each cluster.

1. $C_1 \cup C_2 \ldots \cup C_K = \{1, \ldots, n\}$. In other words, each observation belongs to at least one of the K clusters.

2. $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$. In other words, the clusters are non- overlapping: no observation belongs to more than one cluster.

We want to choose $C_1, \ldots, C_K$ in such a way that they solve:

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k)$$

where $W(C_k)$ is some measure of "with-in cluster" variation of cluster k.

The most common choice of $W(C_k)$ is

$$W\left(C_k\right) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2$$

K-Means

Putting this all together, we want to choose $C_1, \ldots, C_K$ s.t. they solve the following optimization problem

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2$$

Now, how do we solve?

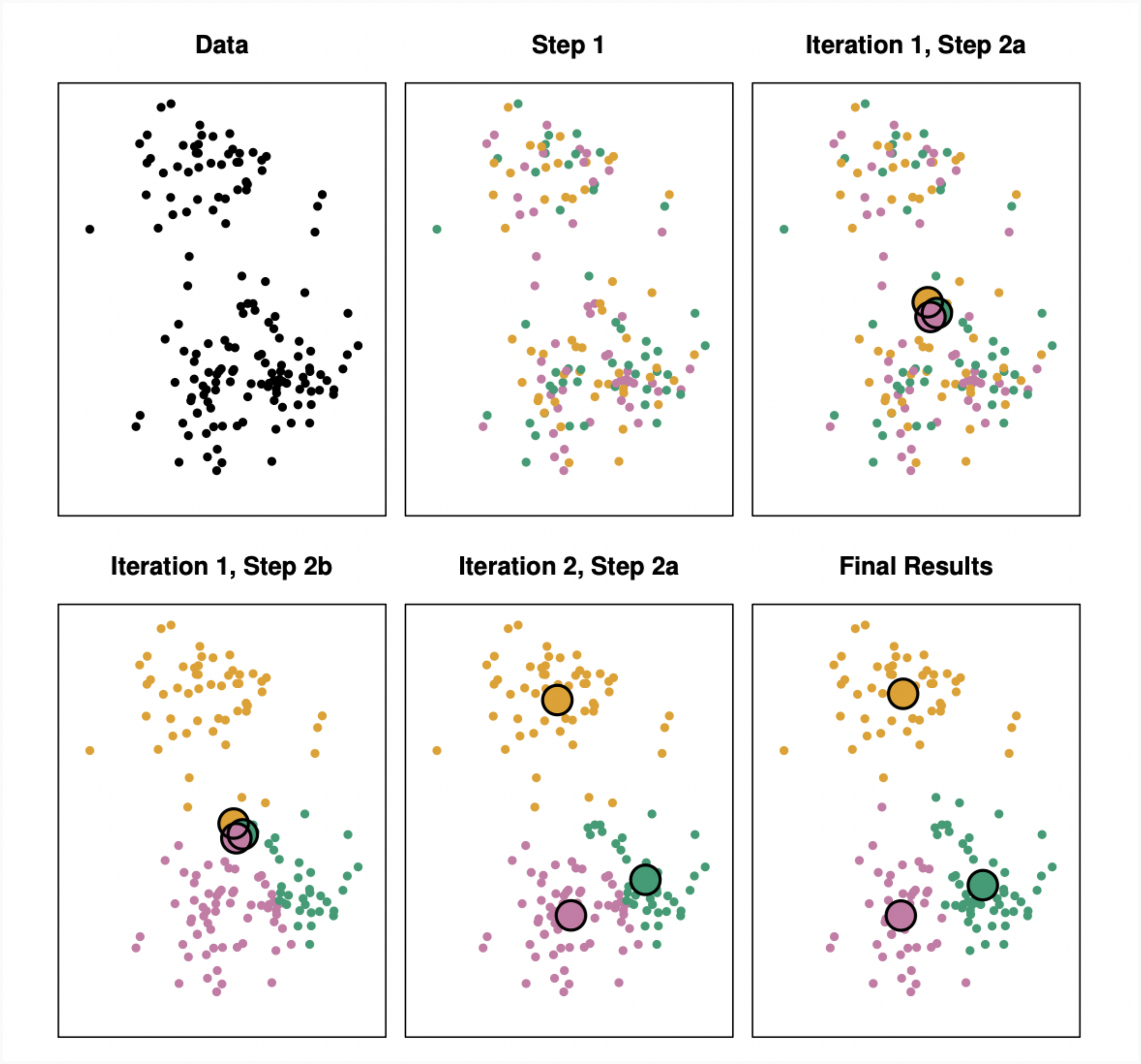
This is actually a hard problem to solve. Fortunately, there is a way to find a local minima - a pretty good solution!

K-Means Algorithm

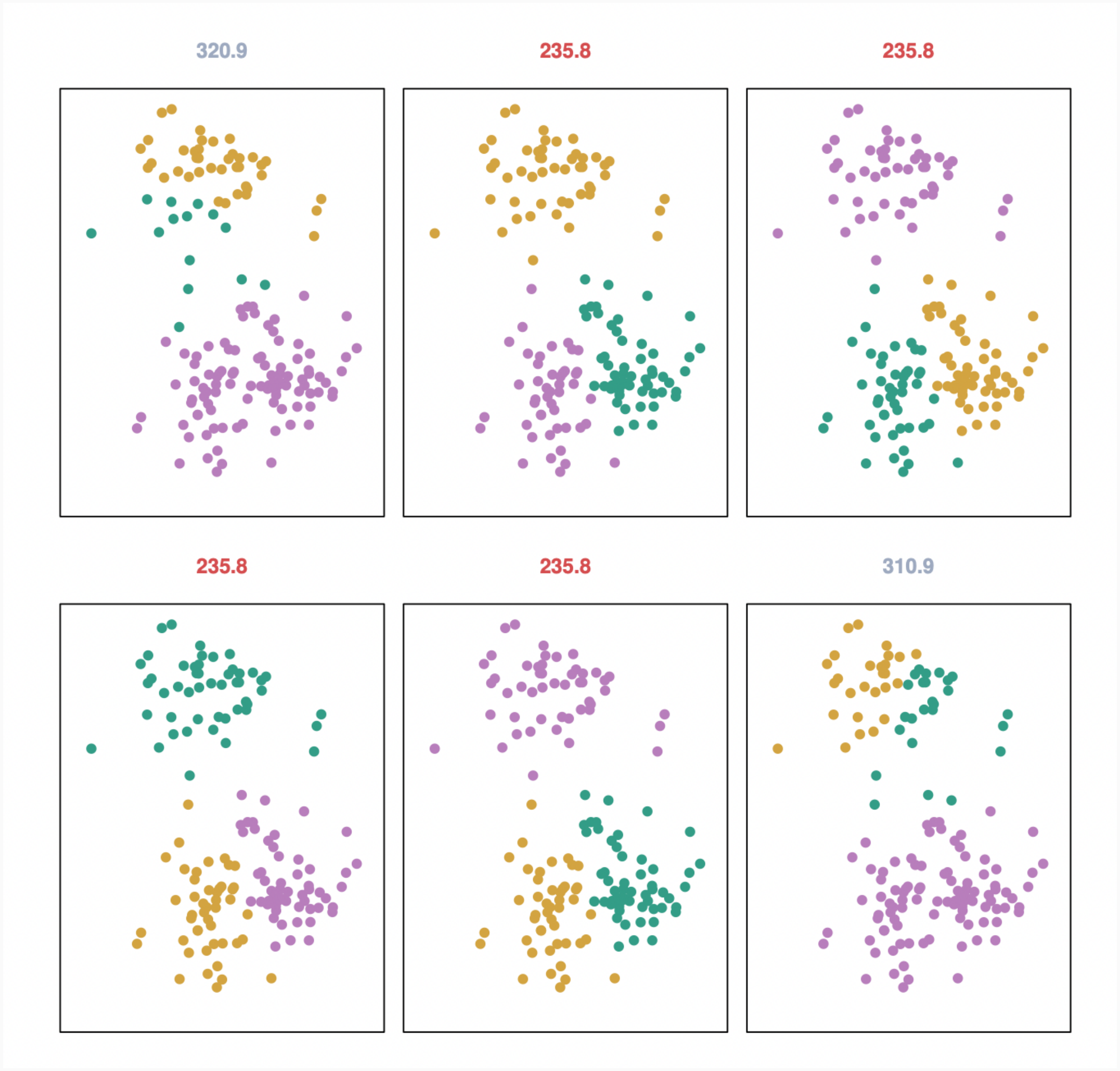
The optimization problem associated with K-means can be solved using the following algorithm.

1. Randomly assign a number, from 1 to K, to each of the observations.These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
    - For each of the K clusters, compute the cluster *centroid*. The k^{th} cluster *centroid* is the vector of the p feature means for the observations in the k^{th} cluster.
    - Assign each observation to the cluster whose *centroid* is closest (where closest is defined using Euclidean distance).

K-Means Algorithm



K-Means Algorithm



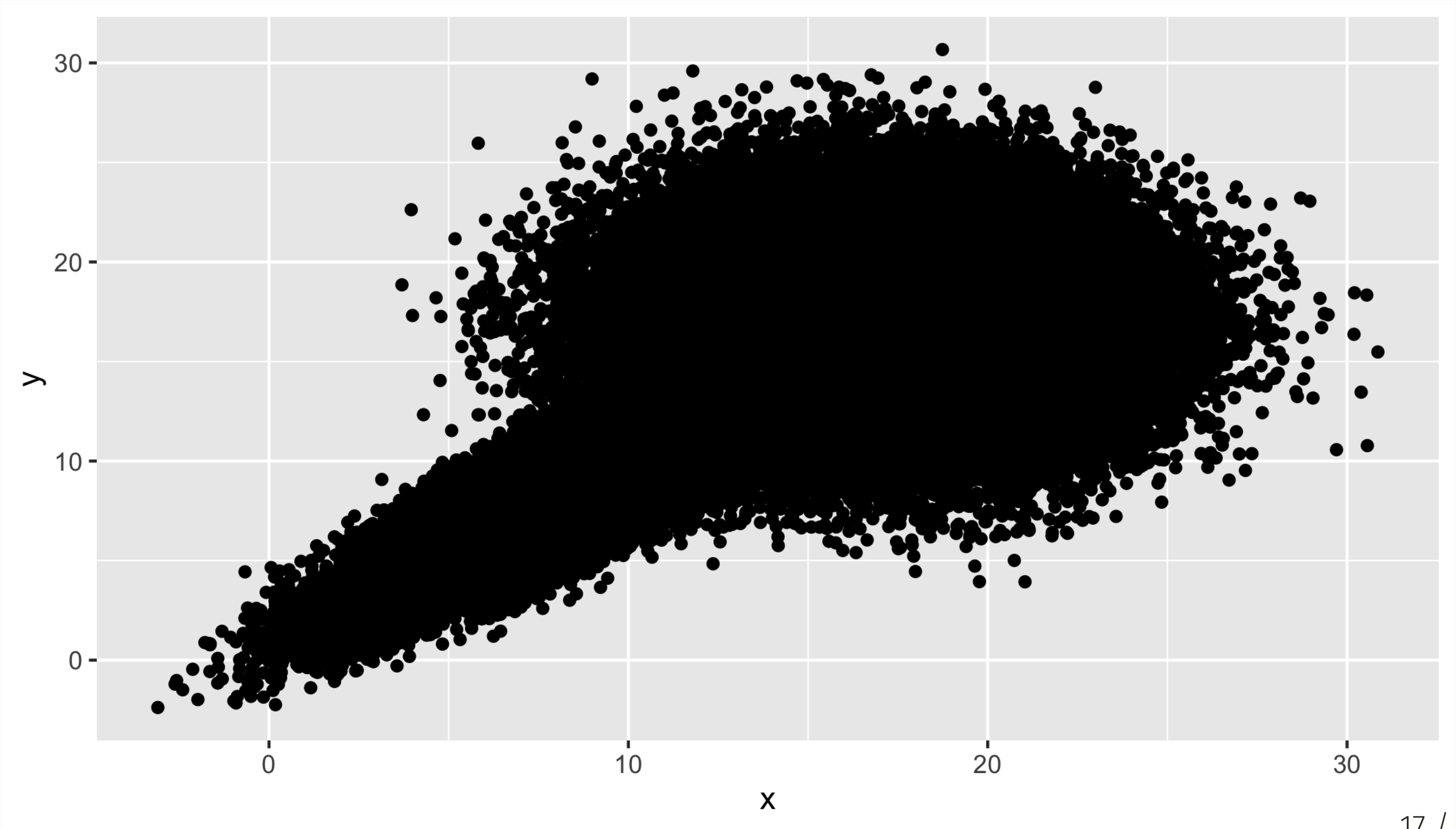
K-Means Example

```r
#requires the following packages: MASS, ggplot2, ggpubr, data.table
set.seed(123)
N = 100000; K = 2; P = 2; NK=N*K

data_MC = mvrnorm(N,c(17,17),matrix(c(10,0,0,10),ncol=2))
data_MC = rbind(data_MC,mvrnorm(N,c(10,10),matrix(c(10,9,9,10),ncol=2)))
data_MC = data.table(x=data_MC[,1],y=data_MC[,2])
data_MC[,group:=c(rep(1,N),rep(2,N))]
data_MC[,group_hat:=sample(1:K,K*N,replace=T)]

data_MC
```

```
##                 x         y group group_hat
##      1: 16.162017 15.227620     1         1
##      2: 11.210668 16.272115     1         2
##      3: 17.187771 21.929068     1         2
##      4: 17.168263 17.222967     1         1
##      5: 15.615225 17.408844     1         1
##     ---
## 199996: 13.704163 13.736442     2         2
## 199997:  1.732098  2.959664     2         2
## 199998:  9.002280  7.461852     2         1
## 199999: 17.288162 15.749094     2         2
## 200000: 13.140113 12.890970     2         1
```

K-Means Example

```
base_plot = ggplot(data=data_MC,aes(x=x,y=y))
base_plot + geom_point()
```



K-Means Example

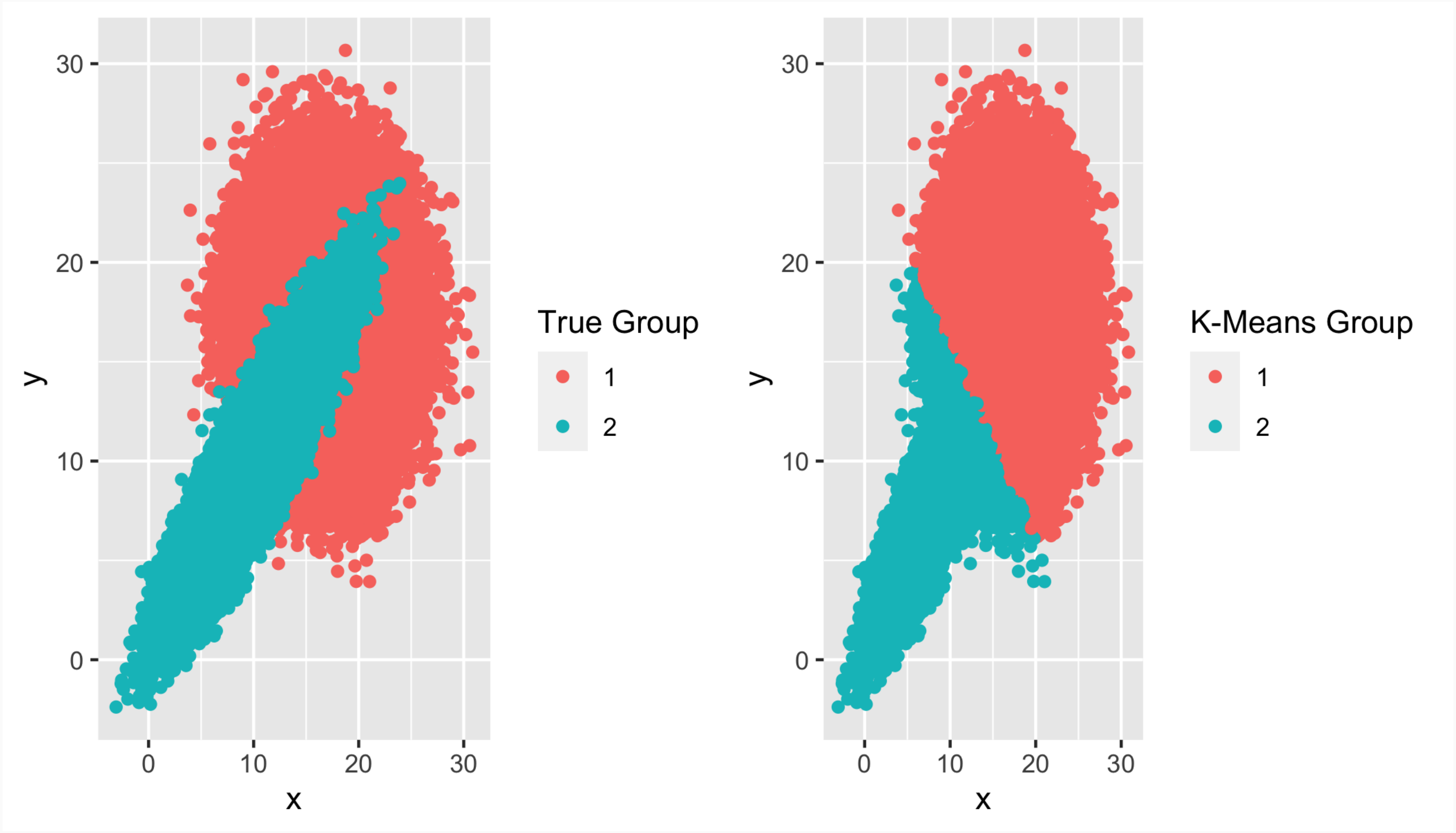
```r
nrow_group = function(i){
  nrow(data_MC[group_hat==i])
}

centroids  = data_MC[,lapply(.SD, mean),by="group_hat"][order(group_hat)]
N_ks       = sapply(1:K,nrow_group)
new_groups = rep(0,NK)

errs = array(0,dim=c(NK,P,K))
MSEs = array(0,dim=c(NK,K))

for(k in 1:K){
  errs[1:NK,1:P,k] = as.matrix((data_MC[,.(x,y)]-centroids[rep(k,NK),.(x,y)])^2)
  MSEs[,k]         = apply(errs[,,k],1,mean)
}
new_groups = apply(MSEs,1,which.min)
```

K-Means Example

```r
while(mean(data_MC$group_hat==new_groups)!=1){
  data_MC[,group_hat:=new_groups]
  centroids   = data_MC[,lapply(.SD, mean),by="group_hat"][order(group_hat)]
  N_ks        = sapply(1:K,nrow_group)
  errs = array(0,dim=c(NK,P,K))
  MSEs = array(0,dim=c(NK,K))

  for(k in 1:K){
    errs[1:NK,1:P,k] = as.matrix((data_MC[,.(x,y)]-centroids[rep(k,NK),.(x,y)])^2)
    MSEs[,k] = apply(errs[,,k],1,mean)
  }
  new_groups = apply(MSEs,1,which.min)
}

data_MC[,group_hat:=new_groups]
centroids   = data_MC[,lapply(.SD, mean),by="group_hat"][order(group_hat)]

plot1 = base_plot+geom_point(data=data_MC,aes(color=factor(group))) +
  labs(color="True Group")
plot2 = base_plot+geom_point(data=data_MC,aes(color=factor(group_hat))) +
  labs(color="K-Means Group")
```

K-Means Example

```
ggarrange(plot1,plot2,ncol=2)
```



Soft Clustering: The EM Algorithm

Soft Clustering

Notice that K-means performs poorly when there's lots of overlap in the data.

Idea: What if we model the clustering and assign probabilities to the group?

Downsides: Must specify a model for the data to be generated from.

Upsides: Can assign probabilities.

The rest of the lecture, we will be using the following example to motivate the method:

- Sex differences in height.

Sex Differences in Height

From the CDC's website, you can download data from the National Health Interview Survey.

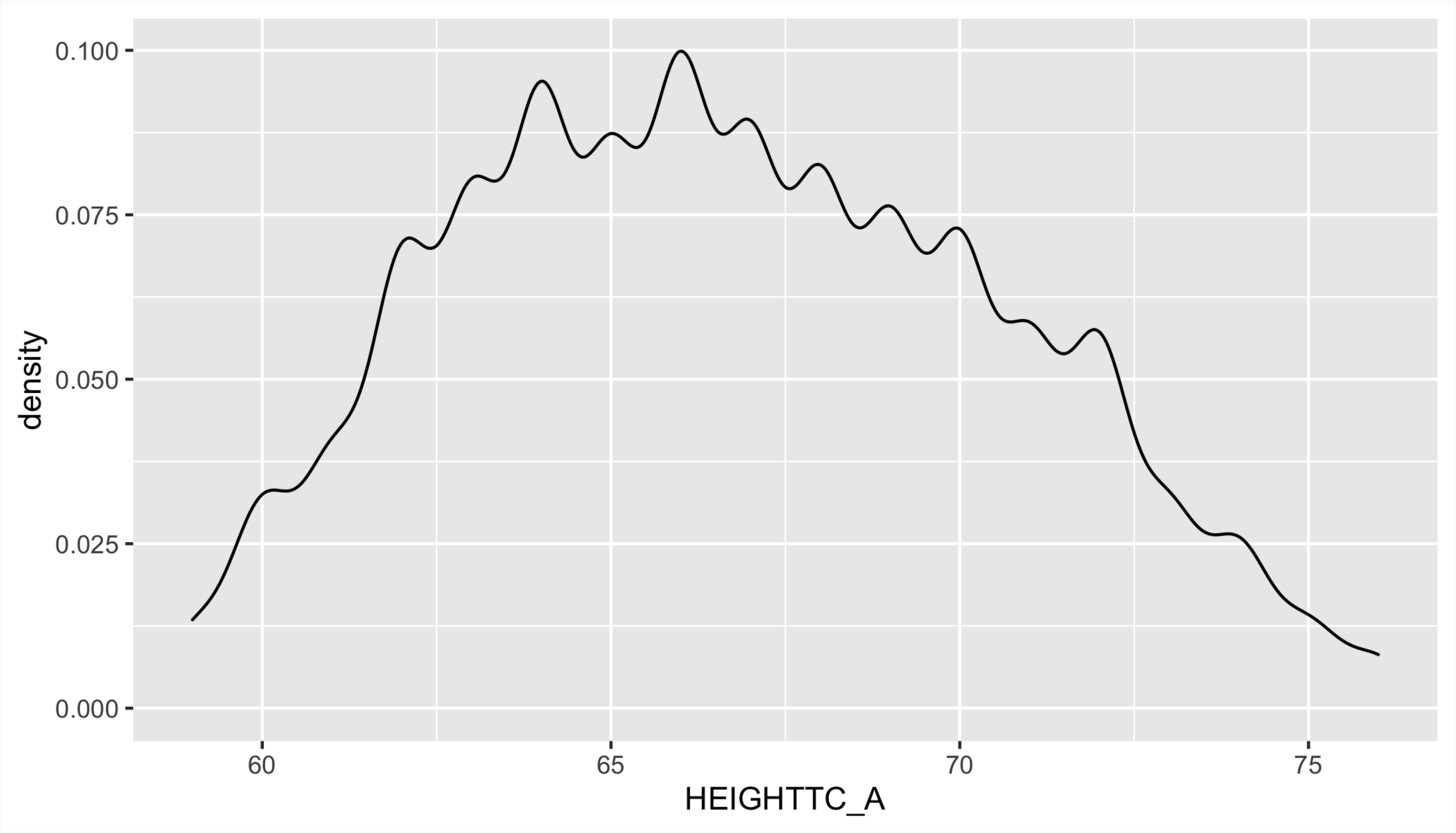
We will be using the adult 2019 data.

This is real data gathered from the US population regarding health.

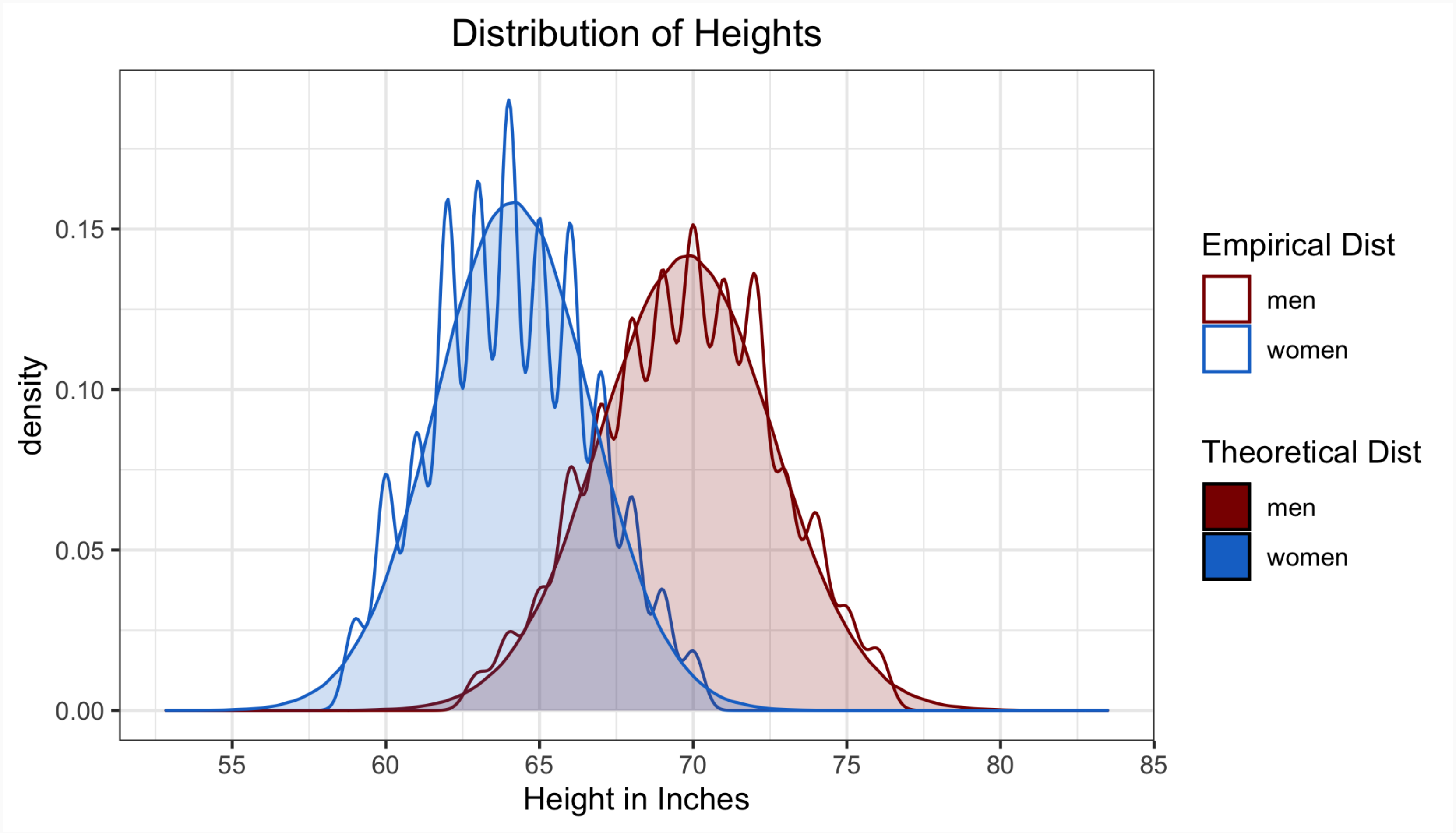
```r
data_path = "/Users/alexmarsh/Documents/School/Teaching/ECON390_local/"
NHIS_data = fread(paste0(data_path,"adult19.csv"))
NHIS_data = NHIS_data[SEX_A<7]       #remove responses not male or female
NHIS_data = NHIS_data[HEIGHTTC_A<96] #remove non-numerical responses

hght_moms         = NHIS_data[,mean(HEIGHTTC_A),by=SEX_A]
names(hght_moms) = c("SEX_A","mu")
temp_data         = NHIS_data[,sd(HEIGHTTC_A),by=SEX_A]
names(temp_data) = c("SEX_A","sigma")
hght_moms         = merge(hght_moms,temp_data,by="SEX_A")
hght_moms
```

```
##    SEX_A       mu    sigma
## 1:     1 69.78983 2.822247
## 2:     2 64.12958 2.516596
```

Sex Differences in Height



Sex Differences in Height



Sex Differences in Height

```r
men_draws     = rnorm(1000000,hght_moms[1,mu],hght_moms[1,sigma])
women_draws   = rnorm(1000000,hght_moms[2,mu],hght_moms[2,sigma])
norm_den_data = data.table(heights=c(men_draws,women_draws),
                           SEX_A = rep(c(1,2),each=length(men_draws)))
sex_vals = c("1"="darkred","2"="dodgerblue3")

ggplot(NHIS_data[HEIGHTTC_A<96],aes(x=HEIGHTTC_A,color=factor(SEX_A)))+geom_density()-
  geom_density(aes(x=heights,fill=factor(SEX_A)),data=norm_den_data,alpha=0.2) +
  scale_color_manual(name="Empirical Dist",labels=c("men","women"),values=sex_vals)+
  scale_fill_manual(name="Theoretical Dist",labels=c("men","women"),values=sex_vals) -
  xlab("Height in Inches") + ggtitle("Distribution of Heights") +
  scale_x_continuous(breaks=seq(50,90,by=5))+
  theme_bw() + theme(plot.title = element_text(hjust = 0.5))
```

Model for Sex Differences: Height

- From the plots, it seems very reasonable to assume that height is normally distributed where men and women have difference distributions.
- If sex is labeled in the data, then this is trivial classification.
- What if it isn't? How can we model this?
- Assume height h_i is drawn from $N(\mu^M, \sigma_M^2)$ with probability α and from $N(\mu^F, \sigma_F^2)$ with probability $(1 - \alpha)$
- This is called a Normal (or Gaussian) Mixture Model
  - The entire distribution is a mixture of two different normals.
- Given a sample of $(h_i)_{i=1}^N$, how can we estimate $\mu^M, \mu^F, \sigma_M^2, \sigma_F^2$, and α?

Expectation-Maximization (EM)

An algorithm called the EM algorithm can be used to estimate the parameters.

There are two steps:

1. The Expectation Step: calculate probability each observation belongs a group.
2. The Maximization Step: re-estimate the parameters given the new probabilities.

This requires something called Bayes' Rule.

Denote $P(M|h_i) = P(\text{sex}_i = M|\text{height} = h_i)$ i.e. the prob i is male *given* i's height h_i.

$$P(M|h_i) = \frac{P(h_i|M)P(M)}{P(h_i|M)P(M) + P(h_i|F)P(F))}$$

Notice $P(M) = \alpha$ and $P(h_i|M)$ is the "probability" (actually density) of height h_i if we knew the individual was male. Since we assume $h_i \sim N(\mu^M, \sigma_M^2)$ if male, we know $P(h_i|M) = \phi(h_i; \mu^M, \sigma_M^2)$

$$P(M|h_i) = \frac{\phi(h_i; \mu^M, \sigma_M^2)\alpha}{\phi(h_i; \mu^M, \sigma_M^2)\alpha + \phi(h_i; \mu^F, \sigma_F^2)(1-\alpha)}$$

EM Algorithm

Denote $\theta_n = (\mu_n^M, \mu_n^F, \sigma^2_{M,n}, \sigma^2_{F,n}, \alpha_n)$ as the parameter estimates after n iterations.

Step 0: Initialize θ_0 somehow. The easiest way to do this is to just run K-means and then use the resulting clusters as the groups to condition on when creating $\mu_0^M, \mu_0^F, \sigma^2_{M,0}, \sigma^2_{F,0}$, and α_0.

Step 1:

- E-Step: Calculate $P(M|h_i;\theta_0) = \pi_1^M(h_i)$ using Bayes' Rule. Note $\pi_1^F(h_i) = 1 - \pi_1^M(h_i)$
- M-Step: Calculate θ_1 given $\pi_1(h_i)$ and the data. Formulas on next slide.

Step n: Continue alternating between the E-Step and M-Step until $||\theta_n - \theta_{n-1}|| < \varepsilon$

Normal Mixture Formulas

$$\mu_n^M = \frac{1}{N_M} \sum_{i=1}^{N} \pi_n^M(h_i) h_i$$

$$\mu_n^F = \frac{1}{N_F} \sum_{i=1}^{N} \pi_n^F(h_i) h_i$$

$$\sigma_{M,n}^2 = \frac{1}{N_M} \sum_{i=1}^{N} \pi_n^M(h_i)(h_i - \mu_n^M)^2$$

$$\sigma_{F,n}^2 = \frac{1}{N_F} \sum_{i=1}^{N} \pi_n^F(h_i)(h_i - \mu_n^F)^2$$

$$\alpha_n = \frac{N_M}{N}$$

$$N_M = \sum_{i=1}^{N} \pi_n^M(h_i)$$

$$N_F = N - N_M$$

EM Algorithm

```r
heights = NHIS_data[,HEIGHTTC_A]
h_kmeans = kmeans(heights,2)
M_id = which.max(h_kmeans$centers)
F_id = which.min(h_kmeans$centers)

mu0    = h_kmeans$centers[M_id:F_id]
sigma0 = sapply(M_id:F_id,function(x){sd(heights[h_kmeans$cluster==x])})
alpha  = mean(h_kmeans$cluster==M_id)
theta0 = c(mu0,sigma0,alpha) #form theta0
theta1 = theta0              #initialize theta1

#E-Step
pMh  = dnorm(heights,mu0[1],sigma0[1])*alpha     #prob data and male
pFh  = dnorm(heights,mu0[2],sigma0[2])*(1-alpha) #prob data and female
piMh = pMh/(pMh+pFh)                             #prob male given data

#M-Step
theta1[1] = sum(piMh*heights)/sum(piMh)                            #male mean
theta1[2] = sum((1-piMh)*heights)/sum(1-piMh)                      #female mean
theta1[3] = sqrt(sum(piMh*(heights-theta0[1])^2)/sum(piMh))        #male std dev
theta1[4] = sqrt(sum((1-piMh)*(heights-theta0[2])^2)/sum(1-piMh)) #female std dev
theta1[5] = sum(piMh)/length(piMh)                                 #prob male
```

EM Algorithm

```r
norm = function(x){sqrt(sum(x^2))}
eps  = 1e-16

while(norm(theta0-theta1)>eps){
  theta0 = theta1 #update theta0

  #E-Step
  pMh  = dnorm(heights,theta0[1],theta0[3])*theta0[5]
  pFh  = dnorm(heights,theta0[2],theta0[4])*(1-theta0[5])
  piMh = pMh/(pMh+pFh)

  #M-Step
  theta1[1] = sum(piMh*heights)/sum(piMh)
  theta1[2] = sum((1-piMh)*heights)/sum(1-piMh)
  theta1[3] = sqrt(sum(piMh*(heights-theta1[1])^2)/sum(piMh))
  theta1[4] = sqrt(sum((1-piMh)*(heights-theta1[2])^2)/sum(1-piMh))
  theta1[5] = sum(piMh)/length(piMh)
}

EM_moms   = data.table(sex=c(1,2),mu=theta1[1:2],sigma = theta1[3:4],
                       prob=c(theta1[5],1-theta1[5]))
hght_moms = cbind(hght_moms,data.table(prob=c(mean(NHIS_data$SEX_A==1),
                                              mean(NHIS_data$SEX_A==2))))
```

EM Algorithm

```
EM_moms   #moments estimated with EM algorithm
```

```
##    sex       mu    sigma      prob
## 1:   1 69.46370 2.887544 0.5220849
## 2:   2 63.75078 2.307772 0.4779151
```

```
hght_moms #moments from the data, observing sex
```

```
##    SEX_A       mu    sigma      prob
## 1:     1 69.78983 2.822247 0.4600201
## 2:     2 64.12958 2.516596 0.5399799
```

mclust Package

The `mclust` package can be used to implement Gaussian Mixture Models.

```
library(mclust)
GMM          = Mclust(heights,G=2)
GMM_mu       = GMM$parameters$mean
GMM_sigma    = sqrt(GMM$parameters$variance$sigmasq)
GMM_probs    = GMM$parameters$pro
mclust_moms = data.table(sex=c(2,1),mu=GMM_mu,sigma=GMM_sigma, prob=GMM_probs)
mclust_moms = mclust_moms[order(sex)]
mclust_moms
```

```
##    sex       mu    sigma    prob
## 1:   1 68.93201 3.098219 0.60549
## 2:   2 63.35903 2.142466 0.39451
```

```
EM_moms
```

```
##    sex       mu    sigma      prob
## 1:   1 69.46370 2.887544 0.5220849
## 2:   2 63.75078 2.307772 0.4779151
```

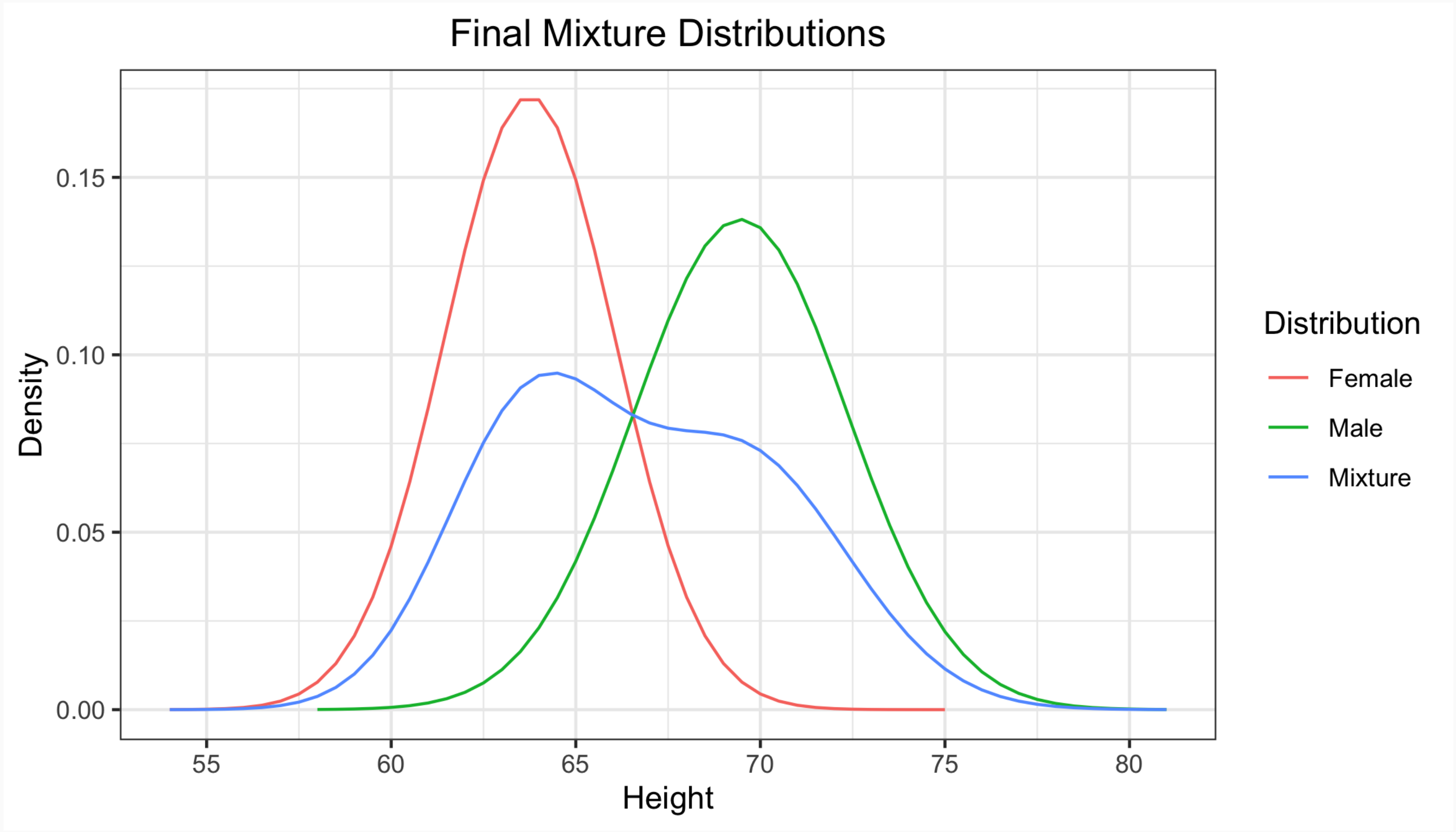
I honestly think my results are better.

EM Algorithm

```r
pMh  = dnorm(heights,theta1[1],theta1[3])*theta0[5]
pFh  = dnorm(heights,theta1[2],theta1[4])*(1-theta0[5])
piMh = pMh/(pMh+pFh)

NHIS_data[,prob_M := piMh]
NHIS_data[,.(HEIGHTTC_A,prob_M,SEX_A)]
```

```
##        HEIGHTTC_A     prob_M SEX_A
##     1:         71 0.99058850     1
##     2:         62 0.03959958     2
##     3:         74 0.99979500     1
##     4:         72 0.99717713     1
##     5:         72 0.99717713     1
##    ---
## 29836:         63 0.06990179     2
## 29837:         71 0.99058850     1
## 29838:         71 0.99058850     1
## 29839:         61 0.02363670     2
## 29840:         64 0.12785762     2
```

Final Distribution



Reading

- Chapter 12 of *An Introduction to Statistical Learning*
- (Optional) Chapter 9 of *Pattern Recognition and Machine Learning* by Christopher Bishop
- (Optional) Economic application of EM:
  - "A Study of Cartel Stability: The Joint Executive Committee, 1880-1886" by Robert Porter (1983 *The Bell Journal of Economics*) (now *The RAND Journal of Economics*)
  - "Finite Mixture Distributions, Sequential Likelihood, and the EM Algorithm" by Arcidiacono and Jones (2003 *Econometrica*)

Next lecture: Supervised Learning