

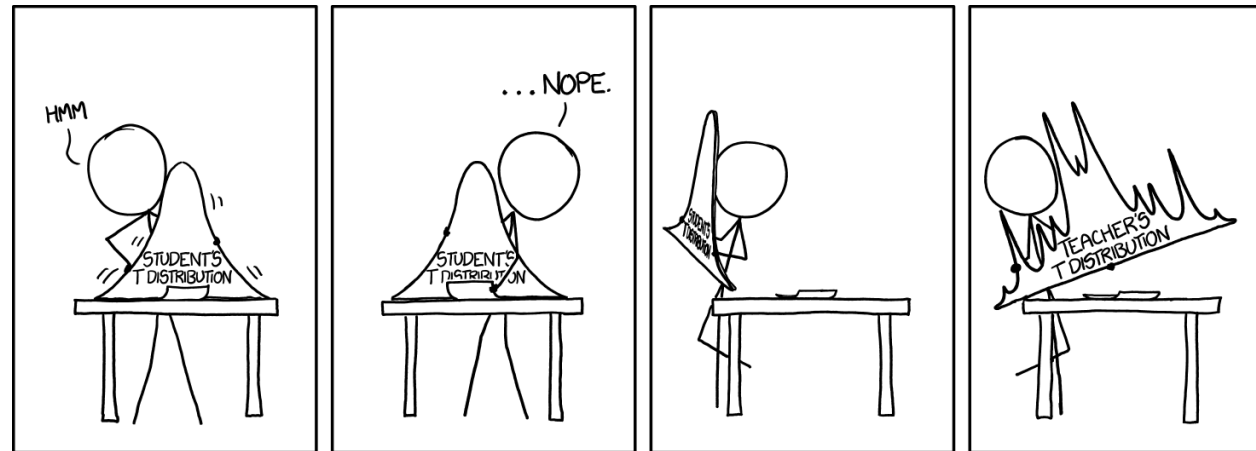
Model-based clustering

Gaussian mixture models

June 17th, 2021

Previously...

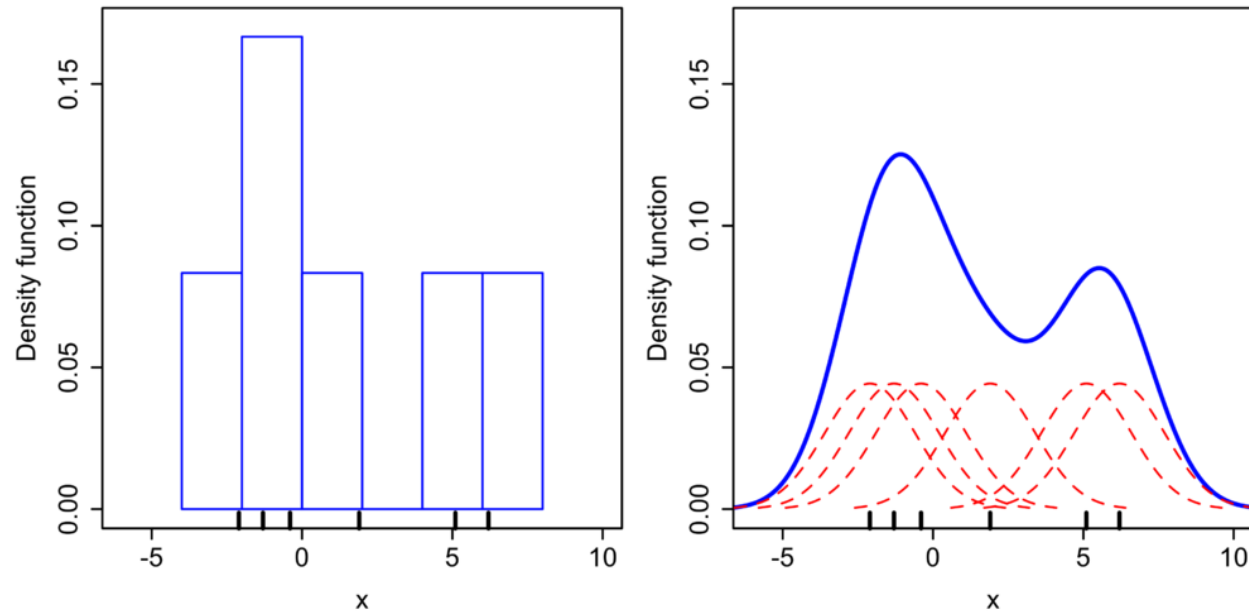
- We explored the use of **K-means** and **hierarchical clustering** for clustering NBA players
- These methods yield **hard** assignments, strictly assigning observations to only one cluster
- What about **soft** assignments? Allow for some **uncertainty** in the clustering results
- Welcome to the wonderful world of **mixture models**



Previously in kernel density estimation...

$$\text{Kernel density estimate: } \hat{f}(x) = \frac{1}{n} \sum_{i=1}^n \frac{1}{h} K_h(x - x_i)$$

- We have to use **every observation** when estimating the density for new points



- Instead we can make **assumptions** to "simplify" the problem

Mixture models

We assume the distribution $f(x)$ is a **mixture** of K **component** distributions:

$$f(x) = \sum_{k=1}^K \pi_k f_k(x)$$

- $\pi_k =$ **mixing proportions (or weights)**, where $\pi_k > 0$, and $\sum_k \pi_k = 1$

This is a **data generating process**, meaning to generate a new point:

1. **pick a distribution / component** among our K options, by introducing a new variable:
 - $z \sim \text{Multinomial}(\pi_1, \pi_2, \dots, \pi_K)$, i.e. categorical variable saying which group the new point is from
2. **generate an observation with that distribution / component**, i.e. $x|z \sim f_z$

So what do we use for each f_k ?

Gaussian mixture models (GMMs)

Assume a **parametric mixture model**, with **parameters** θ_k for the k th component

$$f(x) = \sum_{k=1}^K \pi_k f_k(x; \theta_k)$$

Assume each component is **Gaussian / Normal** where for 1D case:

$$f_k(x; \theta_k) = N(x; \mu_k, \sigma_k^2) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x - \mu_k)^2}{2\sigma_k^2}\right)$$

We need to estimate each $\pi_1, \dots, \pi_k, \mu_1, \dots, \mu_k, \sigma_1, \dots, \sigma_k$!

Let's pretend we only have one component...

If we have n observations from a single Normal distribution, we estimate the distribution parameters using the **likelihood function**, the probability / density of observing the data given the parameters

$$\mathcal{L}(\mu, \sigma | x_1, \dots, x_n) = f(x_1, \dots, x_n | \mu, \sigma) = \prod_i^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{(x_i - \mu)^2}{2\sigma^2}$$

We can compute the **maximum likelihood estimates (MLEs)** for μ and σ

You already know these values!

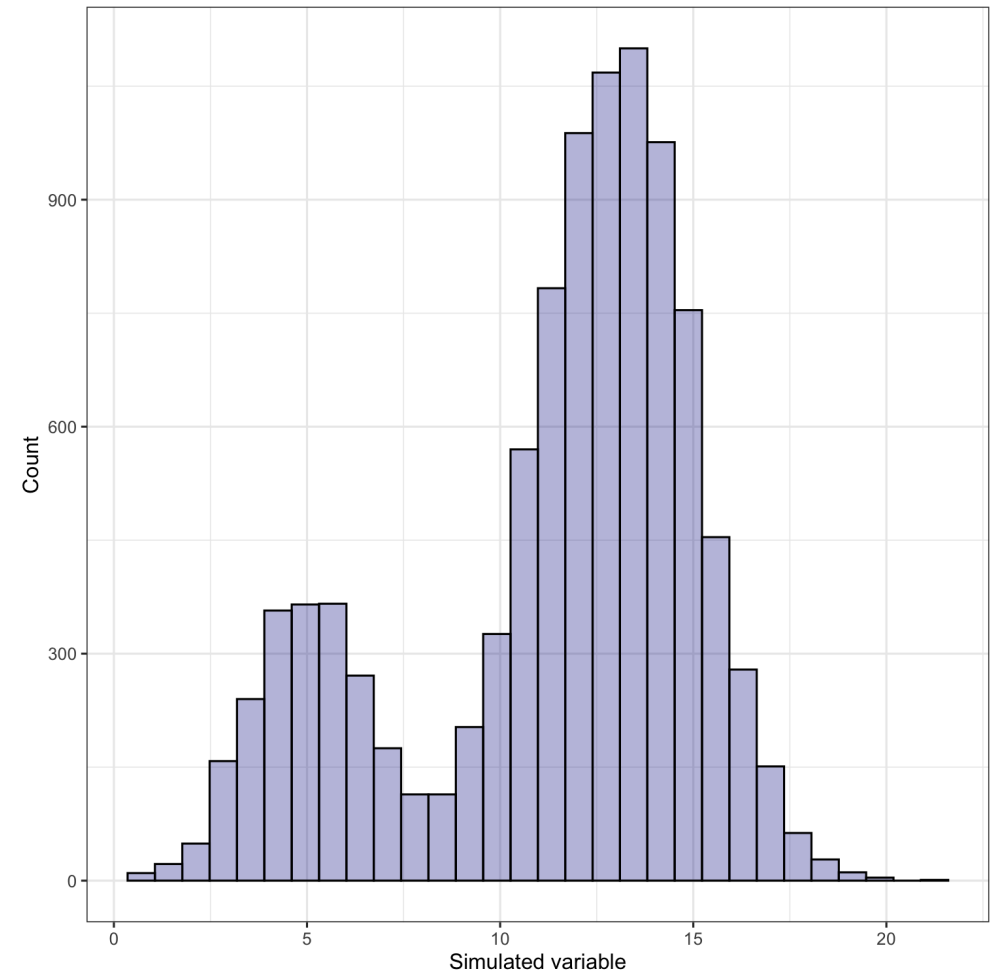
- $\hat{\mu}_{MLE} = \frac{1}{n} \sum_i^n x_i$, sample mean
- $\hat{\sigma}_{MLE} = \sqrt{\frac{1}{n} \sum_i^n (x_i - \mu)^2}$, sample standard deviation (plug in $\hat{\mu}_{MLE}$)

The problem with more than one component...

- We don't know which component an observation belongs to
- IF WE DID KNOW, then we could compute each component's MLEs as before
- But we don't know because z is a **latent variable**! So what about its distribution given the data?

$$P(z_i = k | x_i) = \frac{P(x_i | z_i = k)P(z_i = k)}{P(x_i)}$$
$$= \frac{\pi_k N(\mu_k, \sigma_k^2)}{\sum_{k=1}^K \pi_k N(\mu_k, \sigma_k^2)}$$

- But we do NOT know these parameters!
- This leads to a very useful algorithm in statistics...



Expectation-maximization (EM) algorithm

We alternate between the following:

- *pretending* to know the probability each observation belongs to each group, to estimate the parameters of the components
- *pretending* to know the parameters of the components, to estimate the probability each observation belong to each group

Where have you seen this before? K-means algorithm!

1. Start with initial guesses about $\pi_1, \dots, \pi_k, \mu_1, \dots, \mu_k, \sigma_1, \dots, \sigma_k$
2. Repeat until nothing changes:
 - **Expectation** step: calculate \hat{z}_{ik} = expected membership of observation i in cluster k
 - **Maximization** step: update parameter estimates with **weighted** MLE using \hat{z}_{ik}

How does this relate back to clustering?

From the EM algorithm: \hat{z}_{ik} is a **soft membership** of observation i in cluster k

- you can assign observation i to a cluster with the largest \hat{z}_{ik}
- measure cluster assignment **uncertainty** $= 1 - \max_k \hat{z}_{ik}$

Our parameters determine the type of clusters

In 1D we only have two options:

1. each cluster **is assumed to have equal variance** (spread): $\sigma_1^2 = \sigma_2^2 = \dots = \sigma_k^2$
2. each cluster **is allowed to have a different variance**

But that is only 1D... what happens in multiple dimensions?

Multivariate GMMs

$$f(x) = \sum_{k=1}^K \pi_k f_k(x; \theta_k)$$

where $f_k(x; \theta_k) \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

Each component is a **multivariate normal distribution**:

- $\boldsymbol{\mu}_k$ is a *vector* of means in p dimensions
- $\boldsymbol{\Sigma}_k$ is the $p \times p$ **covariance** matrix - describes the joint variability between pairs of variables

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} & \cdots & \sigma_{1,p} \\ \sigma_{2,1} & \sigma_2^2 & \cdots & \sigma_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p,1} & \sigma_{p,2} & \cdots & \sigma_p^2 \end{bmatrix}$$

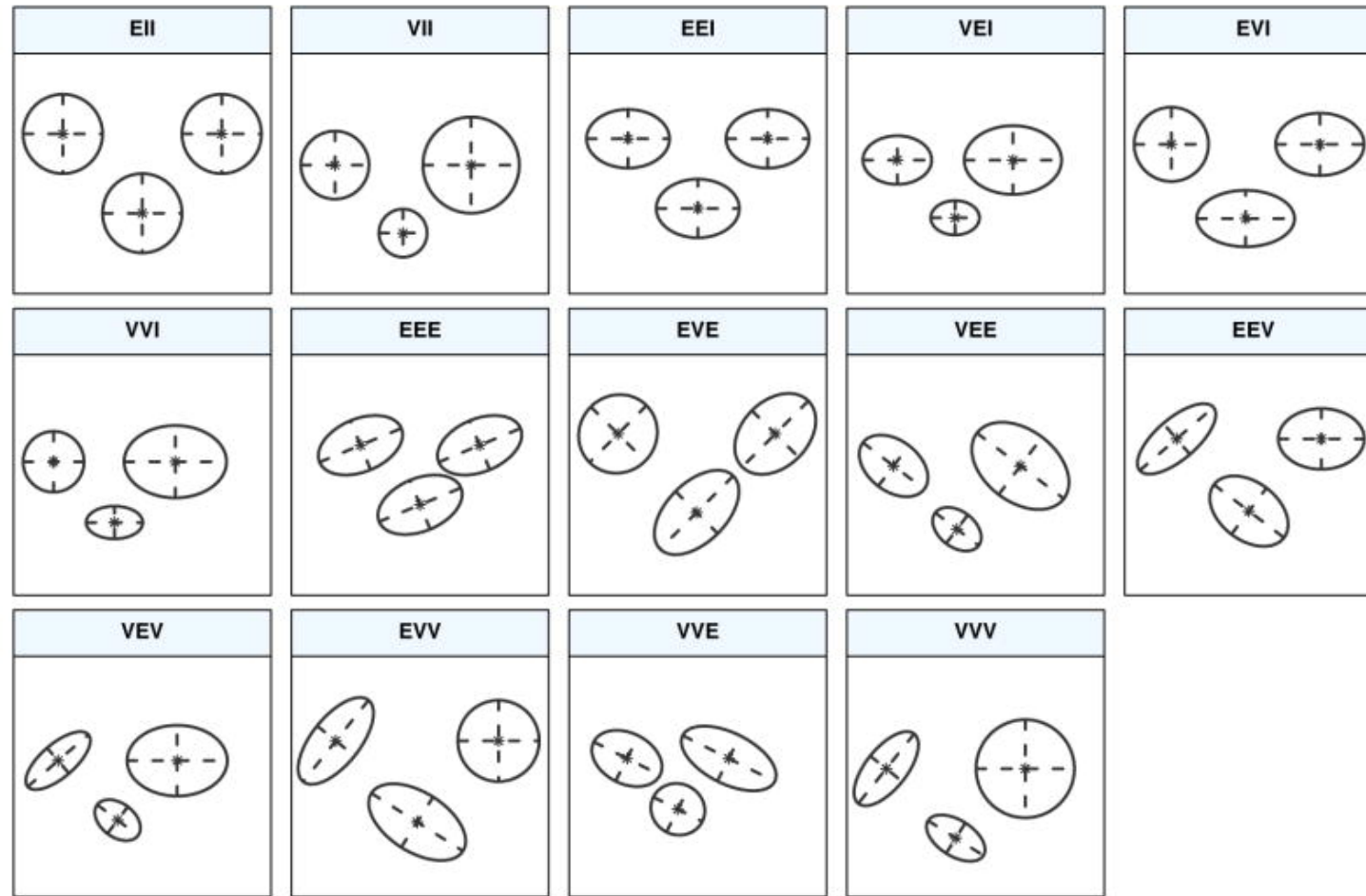
Covariance constraints

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} & \cdots & \sigma_{1,p} \\ \sigma_{2,1} & \sigma_2^2 & \cdots & \sigma_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p,1} & \sigma_{p,2} & \cdots & \sigma_p^2 \end{bmatrix}$$

As we increase the number of dimensions, model fitting and estimation becomes increasingly difficult

We can use **constraints** on multiple aspects of the k covariance matrices:

- **volume**: size of the clusters, i.e., number of observations,
- **shape**: direction of variance, i.e. which variables display more variance
- **orientation**: aligned with axes (low covariance) versus tilted (due to relationships between variables)



- Control volume, shape, orientation
- **E** means equal and **V** means variable (VVV is the most flexible, but has the most parameters)
- Two II is **spherical**, one I is **diagonal**, and the remaining are **general**

So many options! How do we know what to do?



Bayesian information criterion (BIC)

This is a statistical model

$$f(x) = \sum_{k=1}^K \pi_k f_k(x; \theta_k)$$

where $f_k(x; \theta_k) \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

Meaning we can use a **model selection** procedure for determining which best characterizes the data

Specifically - we will use a **penalized likelihood** measure

$$BIC = 2 \log \mathcal{L} - m \log n$$

- $\log \mathcal{L}$ is the log-likelihood of the considered model
- with m parameters (VWV has the most parameters) and n observations
- **penalizes** large models with **many clusters without constraints**
- **we can use BIC to choose the covariance constraints AND number of clusters K !**

The above BIC is really the -BIC of what you typically see, this sign flip is just for ease

How do we implement this? Back to our NBA data... enter `mcclust`

```
library(tidyverse)
nba_pos_stats <-
  read_csv("http://www.stat.cmu.edu/cmsac/sure/2021/materials/data/clustering/nba_2021_player_per
tot_players <- nba_pos_stats %>% filter(tm == "TOT")
nba_player_stats <- nba_pos_stats %>% filter(!(player %in% tot_players$player)) %>%
  bind_rows(tot_players)
nba_filtered_stats <- nba_player_stats %>% filter(mp >= 250)
head(nba_filtered_stats)
```

```
## # A tibble: 6 × 31
##   player      pos  age tm      g    gs    mp    fg    fga fgper...1  x3p  x3pa
##   <chr>      <chr> <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl>
## 1 Precious ... PF      21 MIA      61     4   737    8.4   15.4    0.544    0     0.1
## 2 Steven Ad... C       27 NOP      58    58  1605    5.6    9.2    0.614    0     0.1
## 3 Bam Adeb... C       23 MIA      64    64  2143   10.6   18.5    0.57     0     0.2
## 4 Nickeil A... SG      22 NOP      46    13  1007    9.1   21.8    0.419    3.6   10.4
## 5 Grayson A... SG      25 MEM      50    38  1259    6.6   15.7    0.418    4.1   10.4
## 6 Kyle Ande... PF      27 MEM      69    69  1887    7.8   16.7    0.468    2.4    6.6
## # ... with 19 more variables: x3ppercent <dbl>, x2p <dbl>, x2pa <dbl>,
## #   x2ppercent <dbl>, ft <dbl>, fta <dbl>, ftpercent <dbl>, orb <dbl>,
## #   drb <dbl>, trb <dbl>, ast <dbl>, stl <dbl>, blk <dbl>, tov <dbl>, pf <dbl>,
## #   pts <dbl>, x <lgl>, ortg <dbl>, drtg <dbl>, and abbreviated variable name
```

Selecting the model and number of clusters

Use the `Mclust` function to search over 1 to 9 clusters ($K = G$) and the different covariance constraints (i.e. models)

```
nba_mclust <- Mclust(dplyr::select(nba_filtered_stats, x3pa, trb))
```

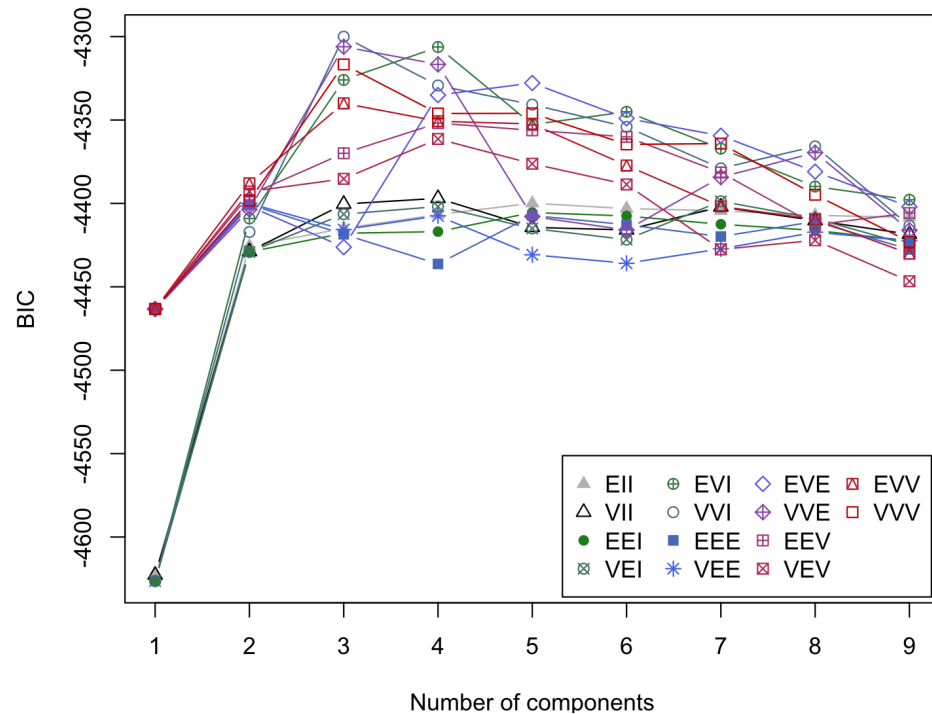
We can use the `summary()` function to display the selection and resulting table of assignments:

```
summary(nba_mclust)
```

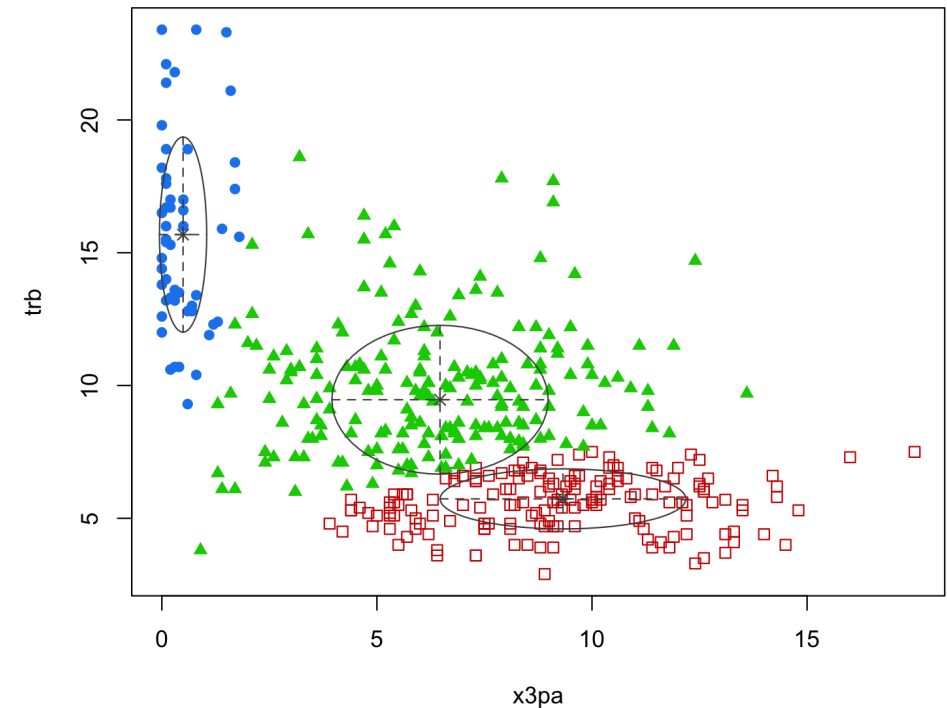
```
## -----  
## Gaussian finite mixture model fitted by EM algorithm  
## -----  
##  
## Mclust VVI (diagonal, varying volume and shape) model with 3 components:  
##  
##   log-likelihood   n df       BIC       ICL  
##      -2107.77 418 14 -4300.038 -4429.194  
##  
## Clustering table:  
##    1    2    3  
##  52 165 201
```


Display the BIC for each model and number of clusters

```
plot(nba_mclust, what = 'BIC',  
     legendArgs = list(x = "bottomright", nco
```



```
plot(nba_mclust, what = 'classification')
```



How do the cluster assignments compare to the positions?

We can again compare the clustering assignments with player positions:

```
table("Clusters" = nba_mclust$classification, "Positions" = nba_filtered_stats$pos)
```

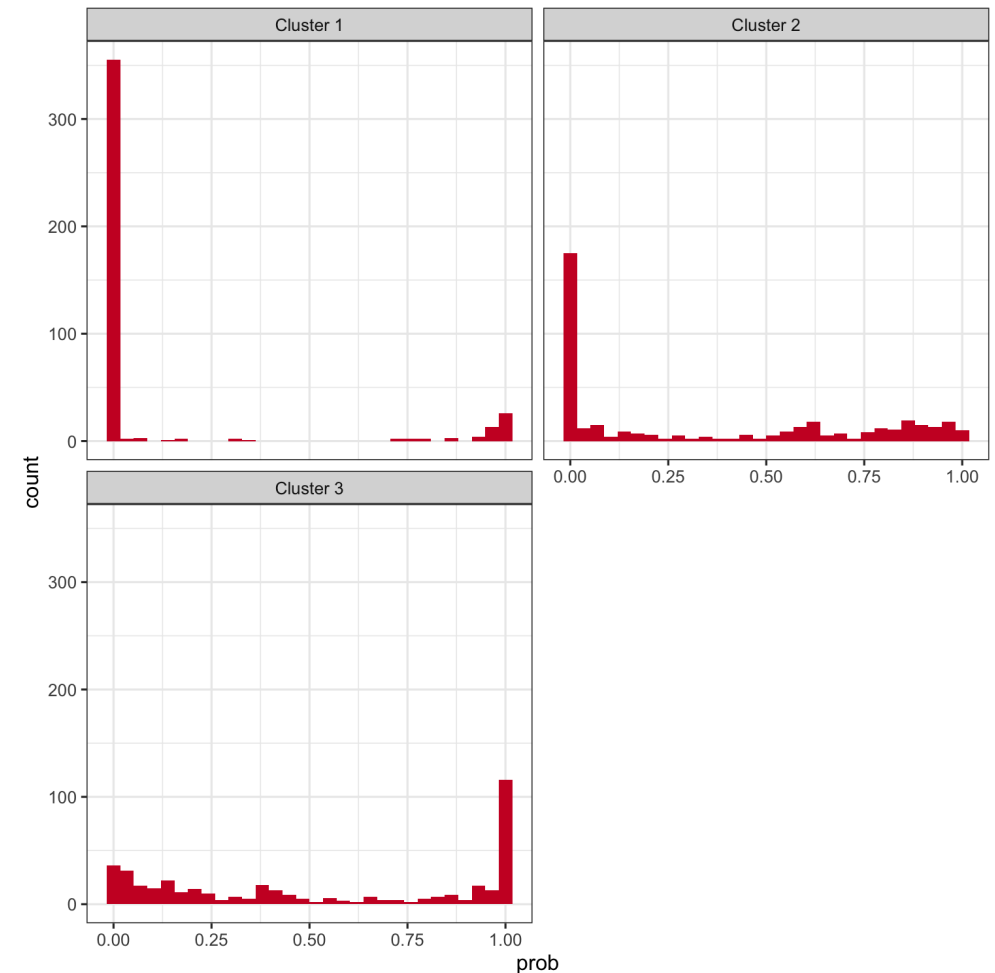
##		Positions												
##	Clusters	C	C-PF	PF	PF-C	PF-SF	PG	PG-SG	SF	SF-PF	SF-SG	SG	SG-PG	SG-SF
##	1	42	0	7	1	0	1	0	0	1	0	0	0	0
##	2	0	0	8	0	1	56	0	23	0	2	74	0	1
##	3	34	2	66	0	0	18	1	49	2	0	25	3	1

What about the cluster probabilities?

```
nba_player_probs <- nba_mclust$z
colnames(nba_player_probs) <-
  paste0('Cluster ', 1:3)

nba_player_probs <- nba_player_probs %>%
  as_tibble() %>%
  mutate(player =
    nba_filtered_stats$player) %>%
  pivot_longer(contains("Cluster"),
    names_to = "cluster",
    values_to = "prob")

nba_player_probs %>%
  ggplot(aes(prob)) +
  geom_histogram() +
  theme_bw() +
  facet_wrap(~ cluster, nrow = 2)
```



Which players have the highest uncertainty?

```
nba_filtered_stats %>%  
  mutate(cluster =  
    nba_mclust$classification,  
    uncertainty =  
    nba_mclust$uncertainty) %>%  
  group_by(cluster) %>%  
  arrange(desc(uncertainty)) %>%  
  slice(1:5) %>%  
  ggplot(aes(y = uncertainty,  
             x = reorder(player,  
                          uncertainty))) +  
  geom_point() +  
  coord_flip() +  
  theme_bw() +  
  facet_wrap(~ cluster,  
             scales = 'free_y', nrow = 3)
```

