Clustering

Hierarchical clustering

June 15th, 2022

Prep NBA player dataset

Created dataset of NBA player statistics per 100 possessions using `ballr`

```r
library(tidyverse)
nba_pos_stats <-
  read_csv("http://www.stat.cmu.edu/cmsac/sure/2022/materials/data/sports/clustering/nba_2022_pla
# Find rows for players indicating a full season worth of stats
tot_players <- nba_pos_stats %>% filter(tm == "TOT")
# Stack this dataset with players that played on just one team
nba_player_stats <- nba_pos_stats %>% filter(!(player %in% tot_players$player)) %>%
  bind_rows(tot_players)
# Filter to only players with at least 125 minutes played
nba_filtered_stats <- nba_player_stats %>% filter(mp >= 125)
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 … C        22 TOR      73    28  1725   7.7  17.5   0.439   1.6   4.5
## 2 Steven Ad… C        28 MEM      76    75  1999   5     9.2   0.547   0     0
## 3 Bam Adeba… C        24 MIA      56    56  1825  11.1  20     0.557   0     0.2
## 4 Santi Ald… PF       21 MEM      32     0   360   7    17.5   0.402   0.8   6.4
## 5 LaMarcus … C        36 BRK      47    12  1050  11.6  21.1   0.55    0.6   2.1
## 6 Grayson A… SG       26 MIL      66    61  1805   6.8  15.1   0.448   4.2  10.4
```

Let's work from the bottom-up...

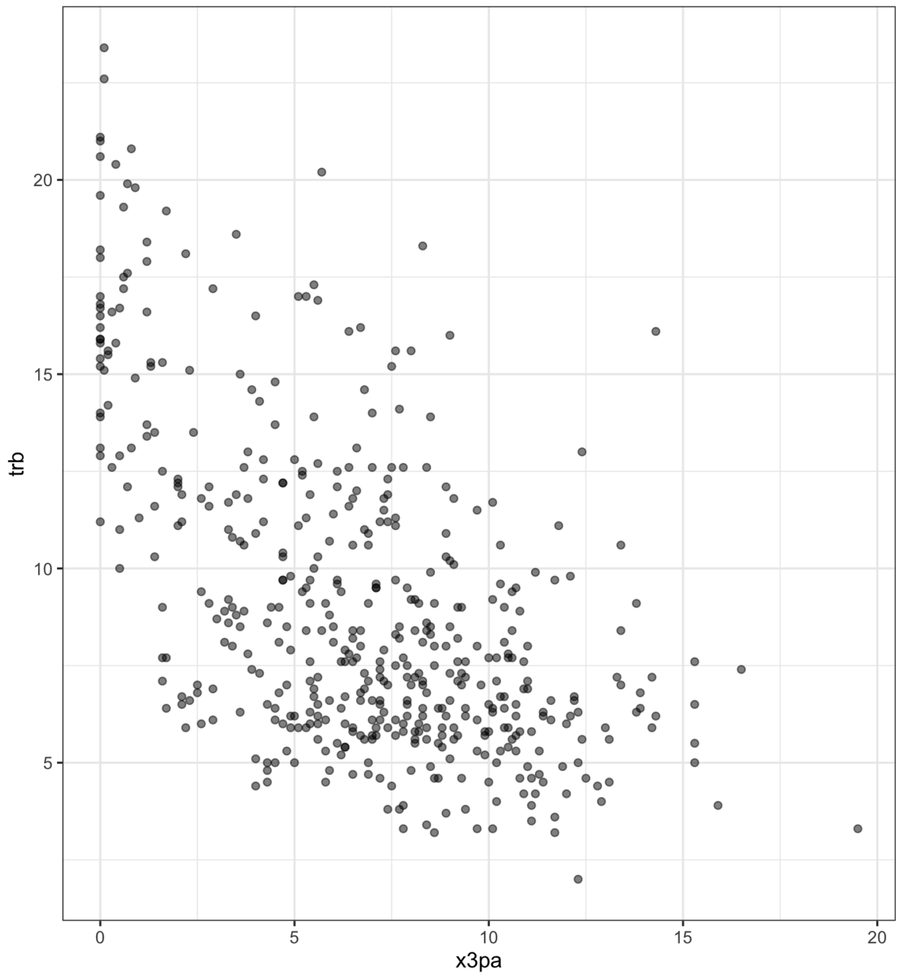
- **Review**: We have p variables for n observations $x_1, \ldots, x_n$,

- Compute the **distance / dissimilarity** between observations

- e.g. **Euclidean distance** between observations i and j

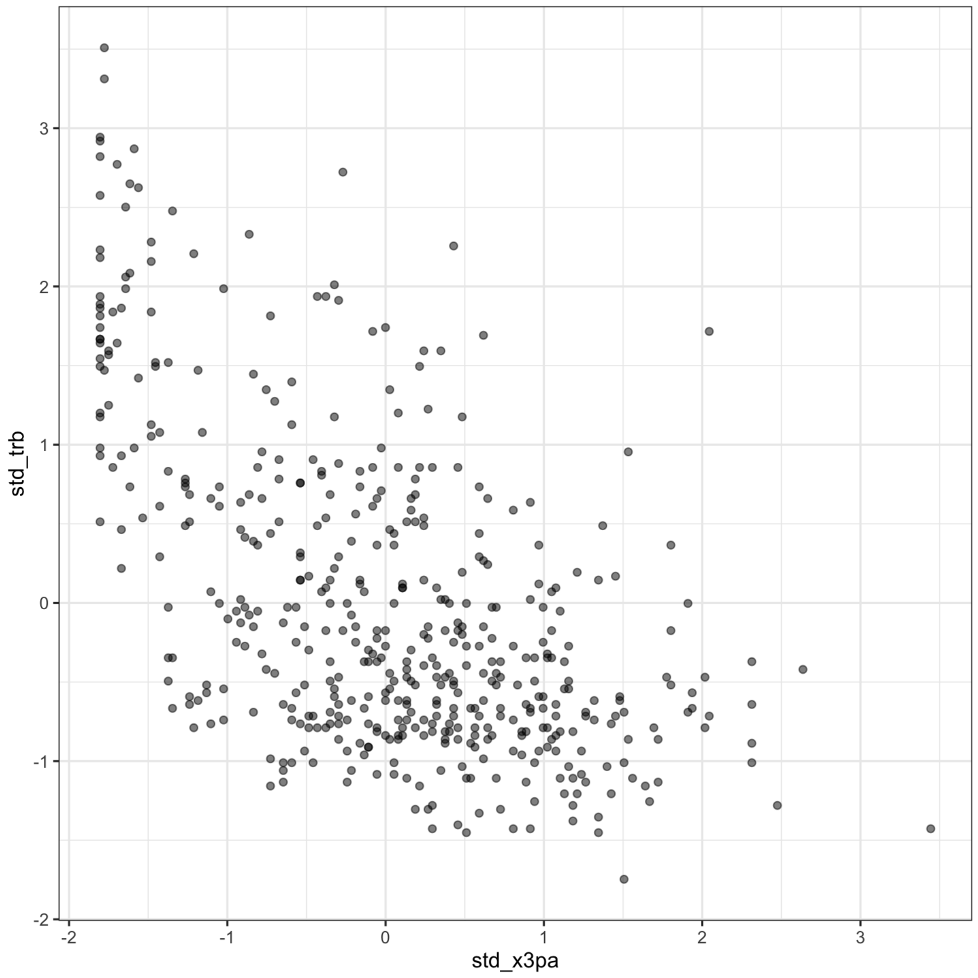
$$d(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + \cdots + (x_{ip} - x_{jp})^2}$$

What are the distances between these NBA players using `x3pa` and `trb`?



Remember to standardize!

```
nba_filtered_stats <- nba_filtered_stats %>%
  mutate(std_x3pa = as.numeric(scale(x3pa)),
         std_trb = as.numeric(scale(trb)))
nba_filtered_stats %>%
  ggplot(aes(x = std_x3pa, y = std_trb)) +
  geom_point(alpha = 0.5) +
  theme_bw() +
  coord_fixed()
```



Compute the distance matrix using `dist()`

- Compute pairwise Euclidean distance

```
player_dist <- dist(dplyr::select(nba_filtere
                                  std_x3pa, s
```

- Returns an object of `dist` class - i.e., not a matrix
- Can convert to a matrix, then set the row and column names:

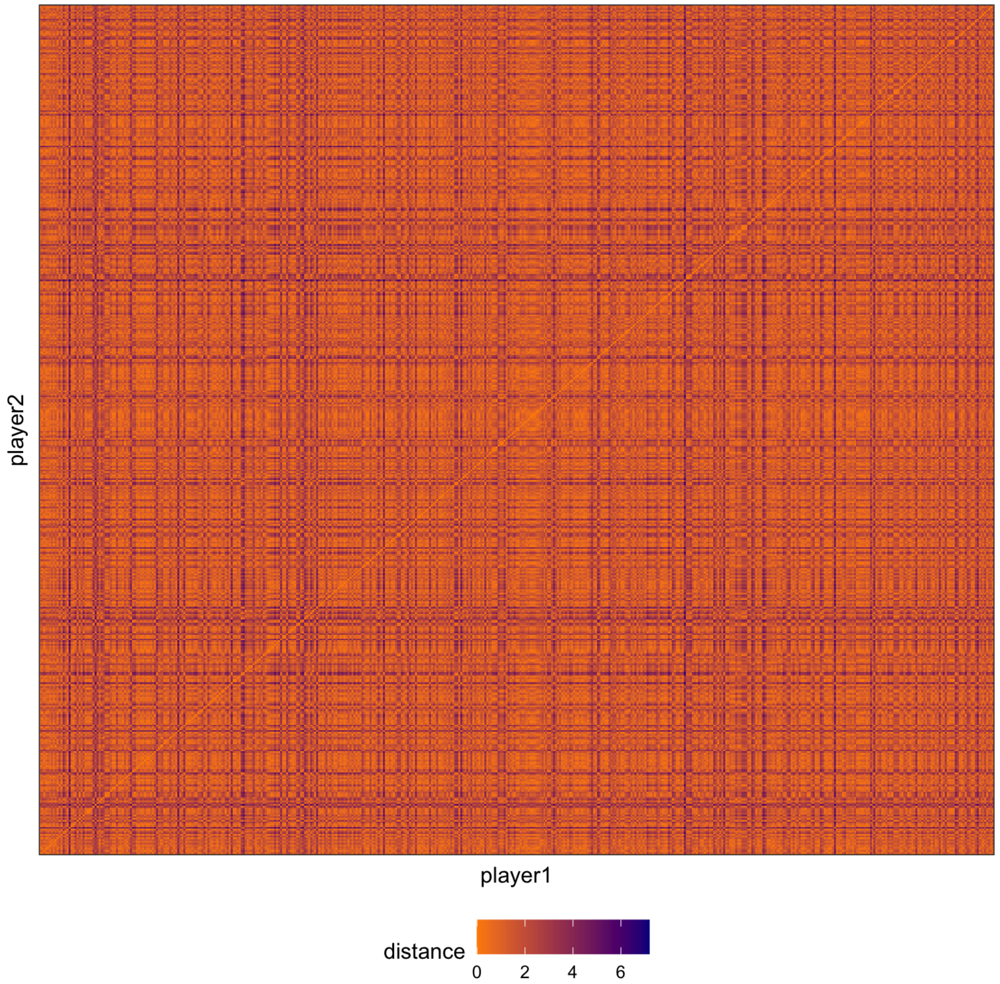
```
player_dist_matrix <- as.matrix(player_dist)
rownames(player_dist_matrix) <- nba_filtered_
colnames(player_dist_matrix) <- nba_filtered_
head(player_dist_matrix[1:3, 1:3])
```

```
##                      Precious Achiuwa Steven Adams Ba
## Precious Achiuwa             0.000000    1.6394586
## Steven Adams                 1.639459    0.0000000
## Bam Adebayo                  1.238740    0.6652539    0.0000000
```

Can convert to a long table for plotting with `ggplot`:

```
long_dist_matrix <-
  as_tibble(player_dist_matrix) %>%
  mutate(player1 = rownames(player_dist_matri
  pivot_longer(cols = -player1,
               names_to = "player2",
               values_to = "distance")
long_dist_matrix %>%
  ggplot(aes(x = player1, y = player2,
             fill = distance)) +
  geom_tile() +
  theme_bw() +
  theme(axis.text = element_blank(),
        axis.ticks = element_blank(),
        legend.position = "bottom") +
  scale_fill_gradient(low = "darkorange",
                      high = "darkblue")
```

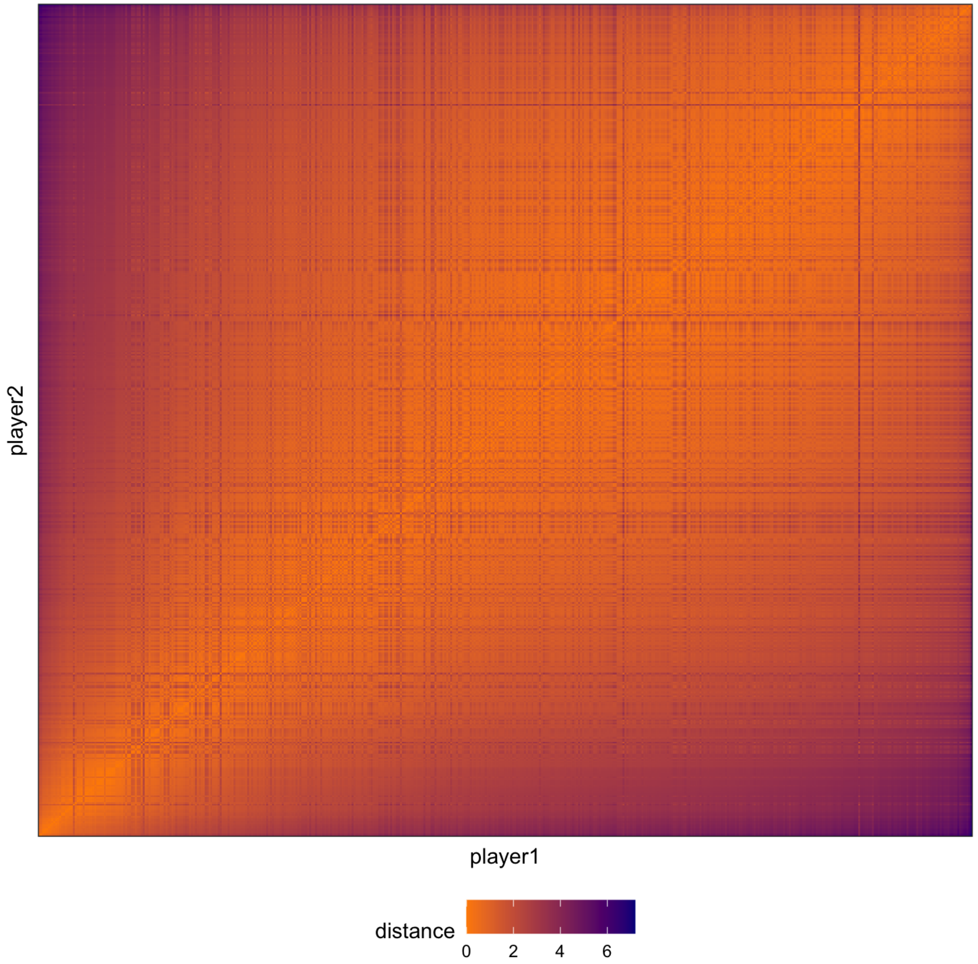
This is useless...



Code interlude: arrange your heatmap with `seriation`

```r
library(seriation)
player_dist_seriate <- seriate(player_dist)
player_order <- get_order(player_dist_seriate
player_names_order <-
  nba_filtered_stats$player[player_order]

long_dist_matrix %>%
  mutate(player1 =
           fct_relevel(player1,
                       player_names_order),
         player2 =
           fct_relevel(player2,
                       player_names_order)) %
  ggplot(aes(x = player1, y = player2,
             fill = distance)) +
  geom_tile() + theme_bw() +
  theme(axis.text = element_blank(),
        axis.ticks = element_blank(),
        legend.position = "bottom") +
  scale_fill_gradient(low = "darkorange",
                      high = "darkblue")
```



(Agglomerative) Hierarchical clustering

Let's pretend all n observations are in their own cluster

- Step 1: Compute the pairwise dissimilarities between each cluster
  - e.g., distance matrix on previous slides
- Step 2: Identify the pair of clusters that are **least dissimilar**
- Step 3: Fuse these two clusters into a new cluster!
- **Repeat Steps 1 to 3 until all observations are in the same cluster**

"Bottom-up", agglomerative clustering that forms a **tree / hierarchy** of merging

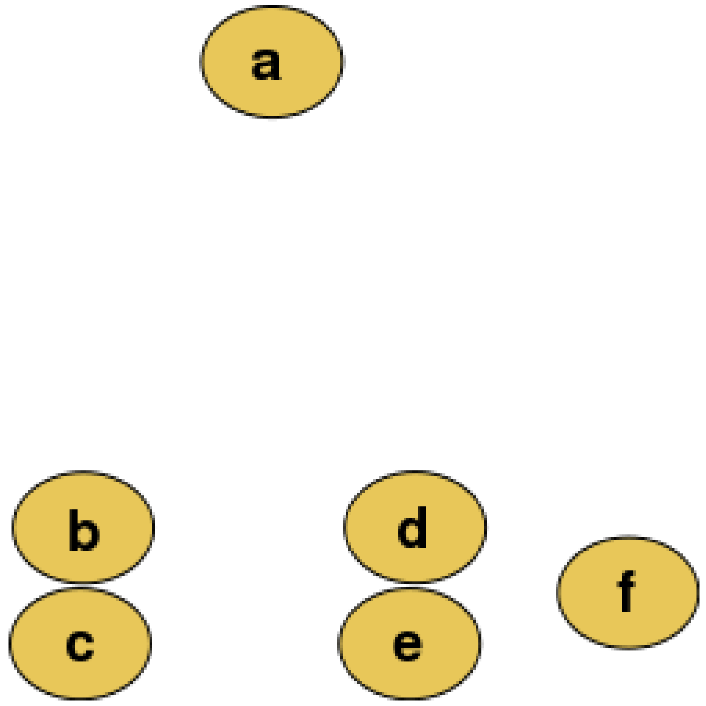
No mention of any randomness!

No mention of the number of clusters K!

(Agglomerative) Hierarchical clustering

Start with all observations in their own cluster

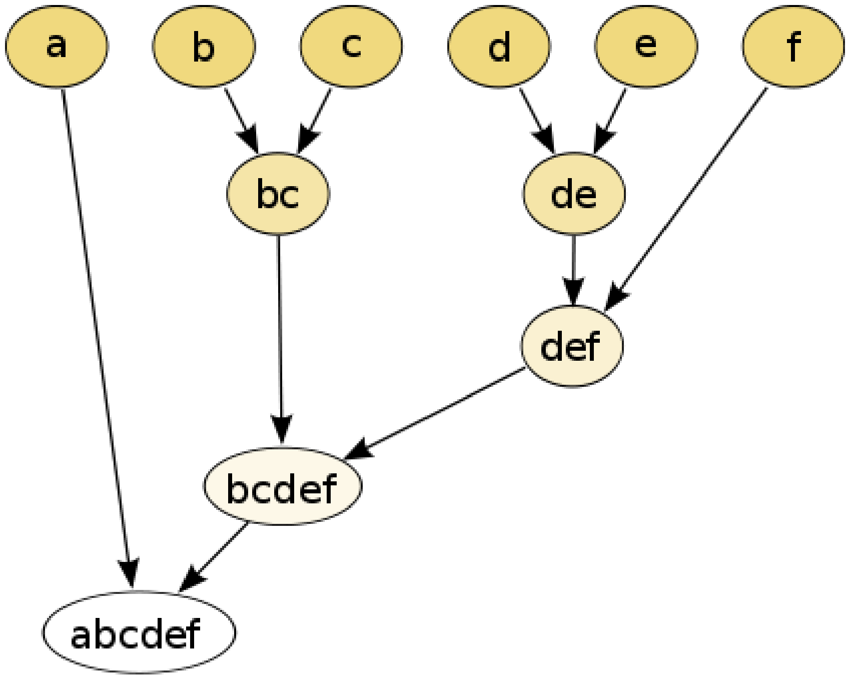
- Step 1: Compute the pairwise dissimilarities between each cluster
- Step 2: Identify the pair of clusters that are **least dissimilar**
- Step 3: Fuse these two clusters into a new cluster!
- **Repeat Steps 1 to 3 until all observations are in the same cluster**



(Agglomerative) Hierarchical clustering

Start with all observations in their own cluster

- Step 1: Compute the pairwise dissimilarities between each cluster
- Step 2: Identify the pair of clusters that are **least dissimilar**
- Step 3: Fuse these two clusters into a new cluster!
- **Repeat Steps 1 to 3 until all observations are in the same cluster**



Forms a **dendrogram** (typically displayed from bottom-up)

How do we define dissimilarity between clusters?

We know how to compute distance / dissimilarity between two observations

But how do we handle clusters?

- Dissimilarity between a cluster and an observation, or between two clusters

We need to choose a **linkage function**! Clusters are built up by **linking them together**

Compute all pairwise dissimilarities between observations in cluster 1 with observations in cluster 2

i.e. Compute the distance matrix between observations, $d(x_i, x_j)$ for $i \in C_1$ and $j \in C_2$

- **Complete linkage**: Use the **maximum** value of these dissimilarities: $\max\limits_{i \in C_1, j \in C_2} d(x_i, x_j)$
- **Single linkage**: Use the **minimum** value: $\min\limits_{i \in C_1, j \in C_2} d(x_i, x_j)$
- **Average linkage**: Use the **average** value: $\frac{1}{|C_1| \cdot |C_2|} \sum_{i \in C_1} \sum_{j \in C_2} d(x_i, x_j)$

Define dissimilarity between two clusters **based on our initial dissimilarity matrix between observations**

Complete linkage example

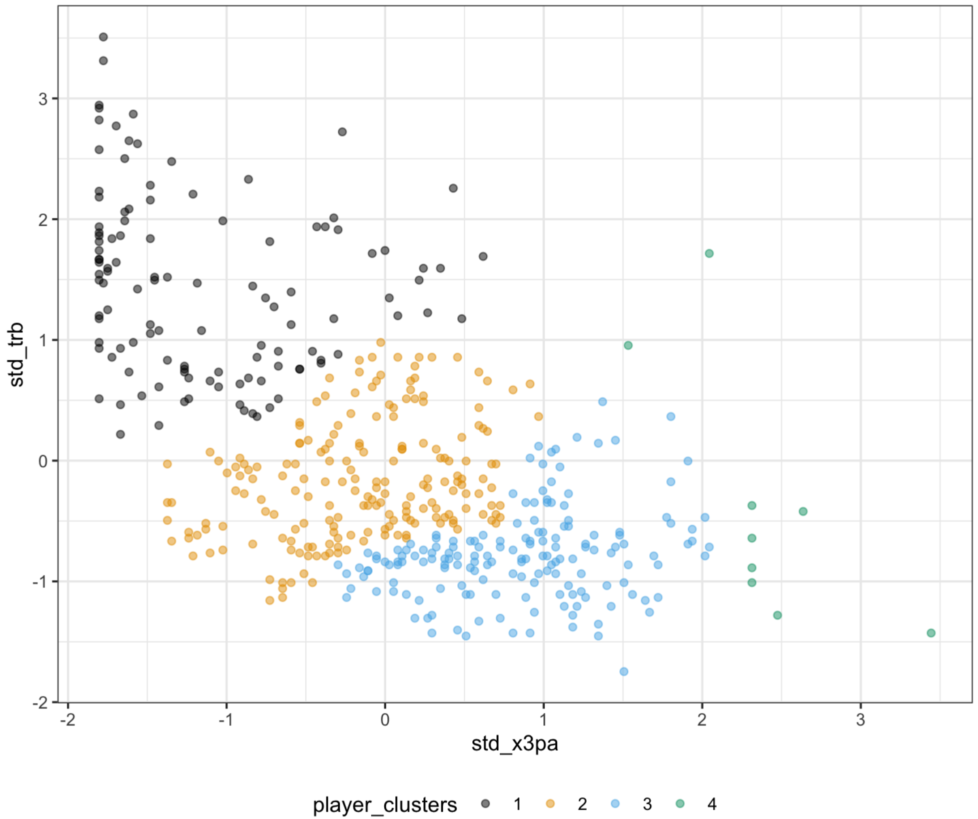
- Use the `hclust` function with a `dist()` object
- Uses `complete` linkage by default

```
nba_complete_hclust <-
  hclust(player_dist, method = "complete")
```

- Need to use `cutree()` to return cluster labels:

```
nba_filtered_stats %>%
  mutate(player_clusters =
           as.factor(cutree(nba_complete_hclu
                            k = 4))) %>%
  ggplot(aes(x = std_x3pa, y = std_trb,
             color = player_clusters)) +
  geom_point(alpha = 0.5) +
  ggthemes::scale_color_colorblind() +
  theme_bw() +
  theme(legend.position = "bottom")
```

Returns *compact* clusters, similar to K-means

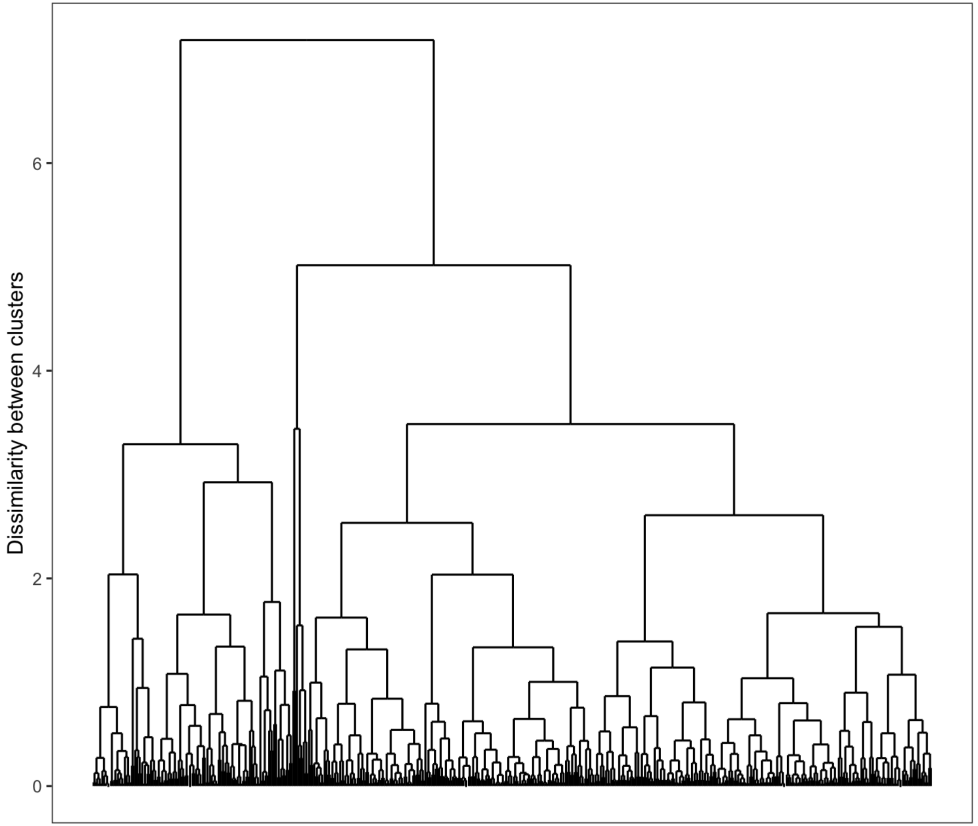


What are we cutting? Dendrograms

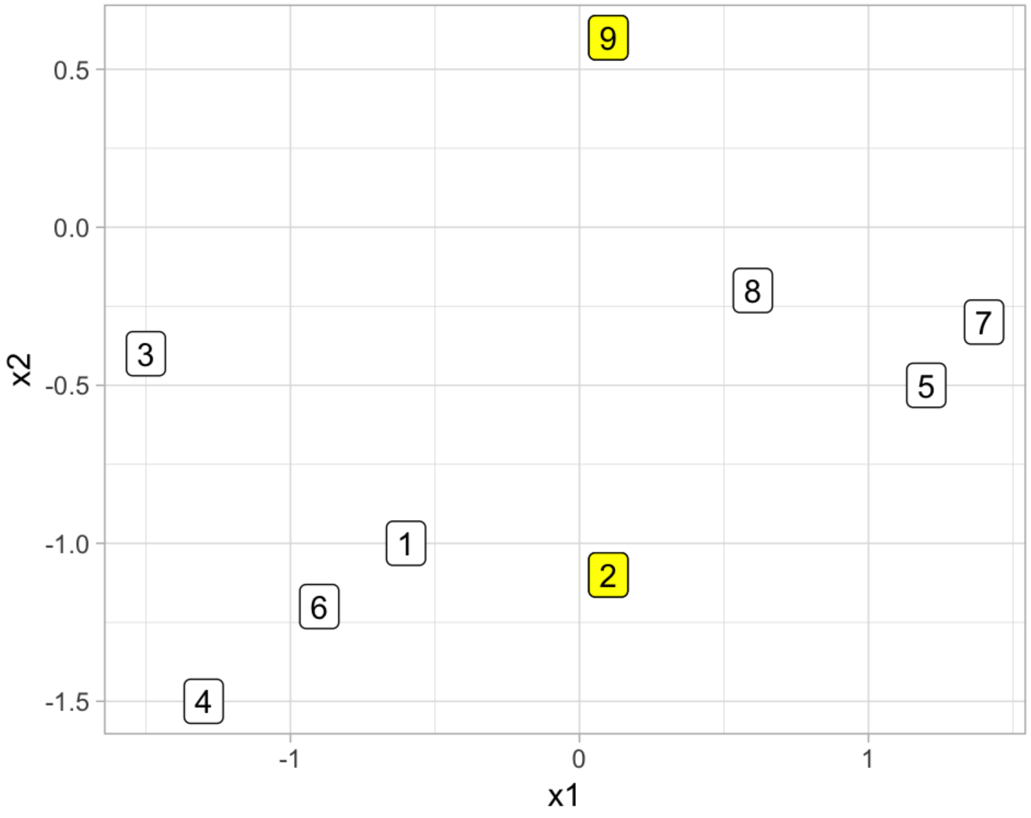
Use the ggdendro package (instead of `plot()`)

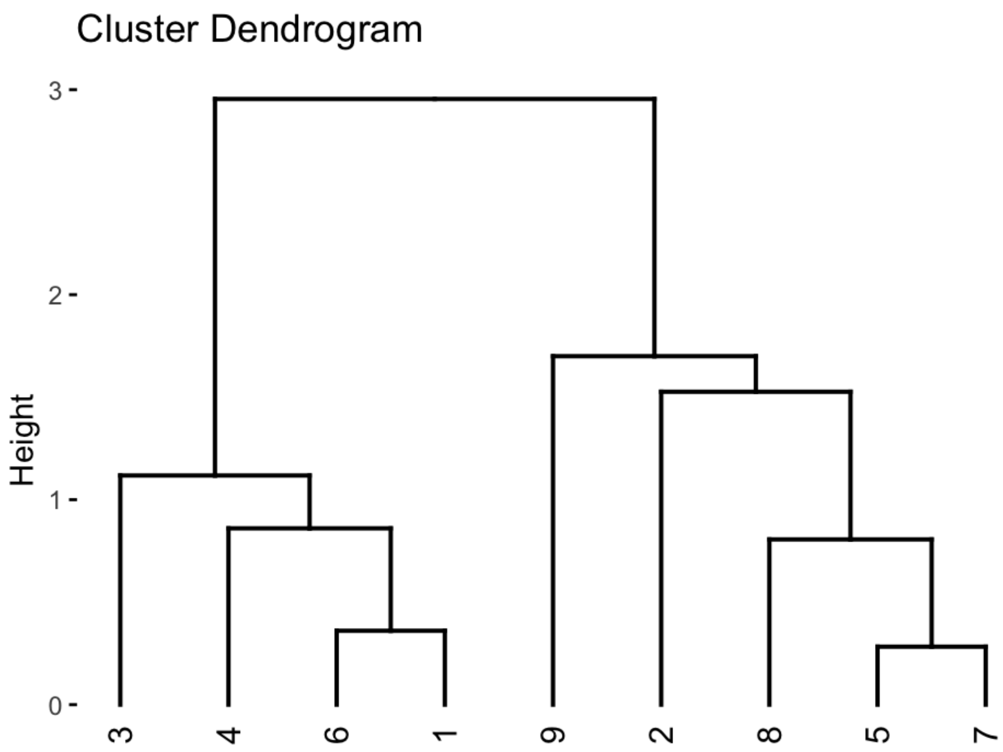
```
library(ggdendro)
ggdendrogram(nba_complete_hclust, theme_dendr
             labels = FALSE, leaf_labels = FA
  labs(y = "Dissimilarity between clusters")
  theme_bw() +
  theme(axis.text.x = element_blank(),
        axis.title.x = element_blank(),
        axis.ticks.x = element_blank(),
        panel.grid = element_blank())
```

- Each **leaf** is one observation
- **Height of branch indicates dissimilarity between clusters**
    - (After first step) Horizontal position along x-axis means nothing



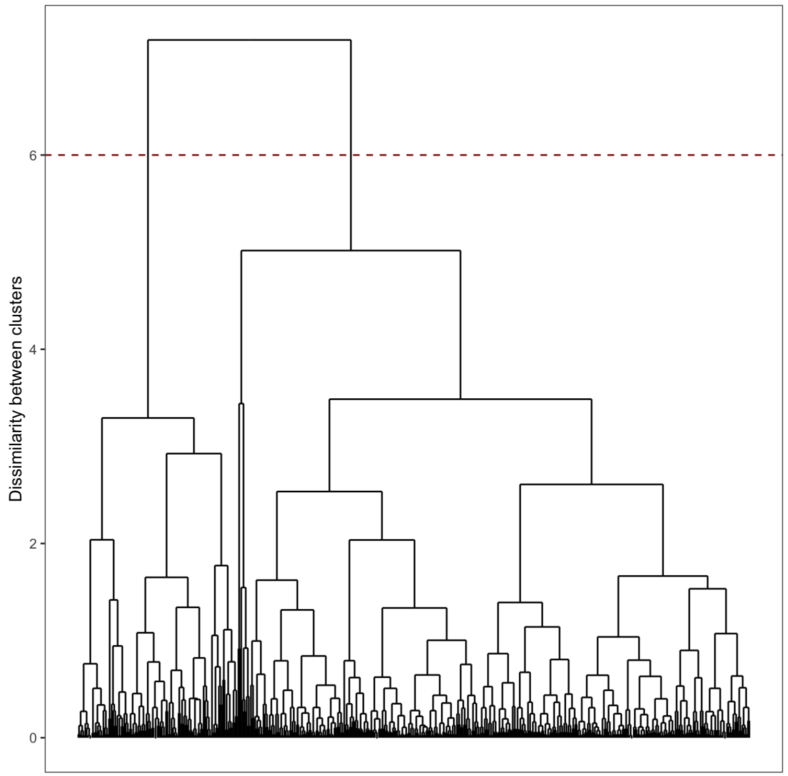
Textbook example



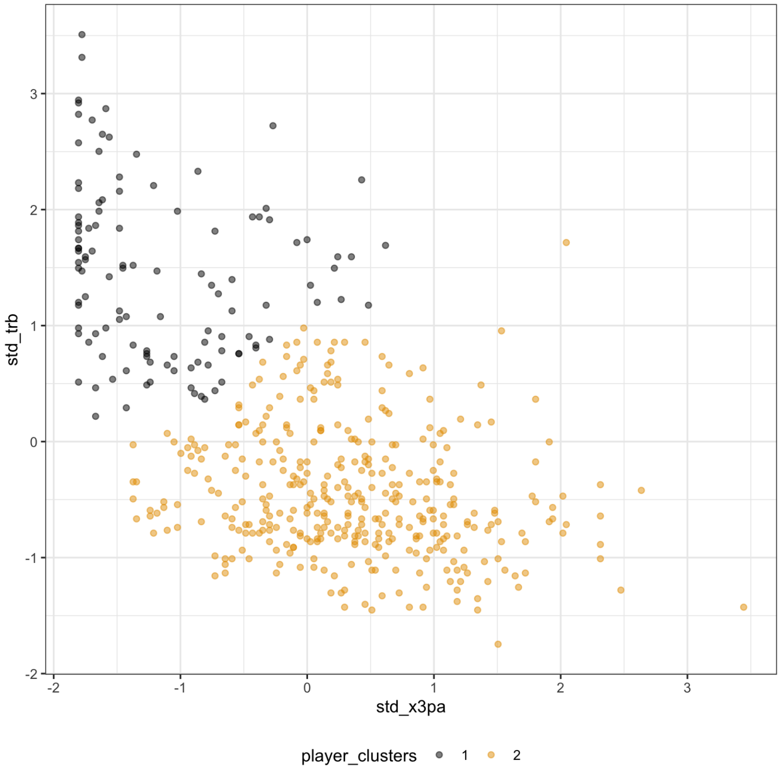


Cut dendrograms to obtain cluster labels

Specify the height to cut with `h` instead of `k`

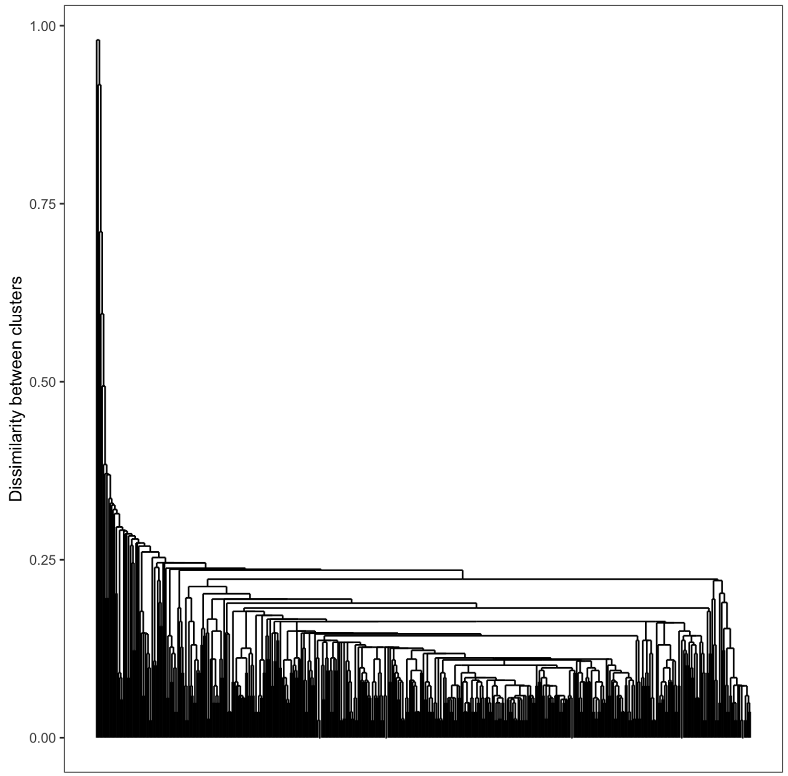



```
cutree(nba_complete_hclust, h = 6)
```

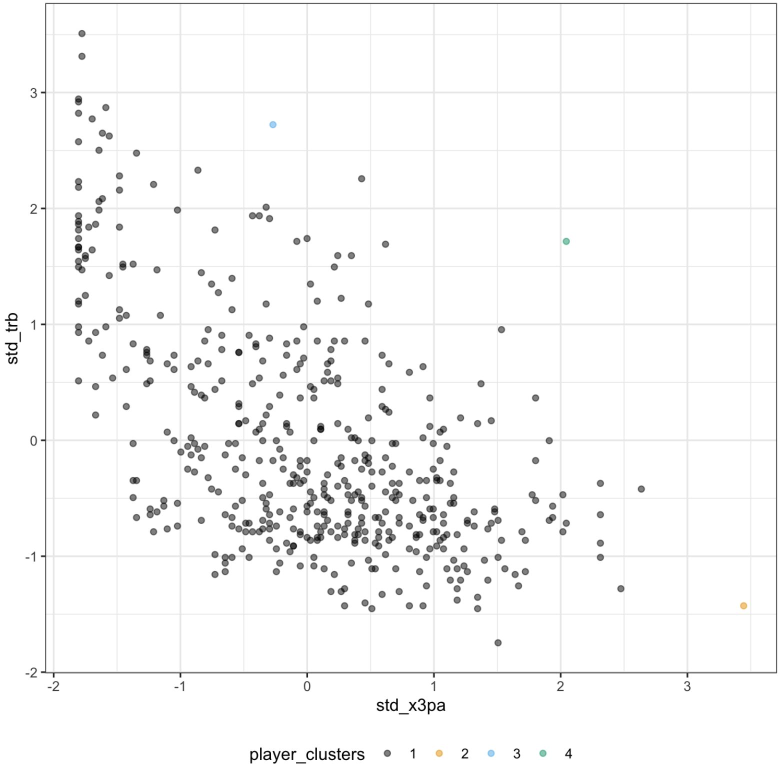


Single linkage example

Change the `method` argument to `single`

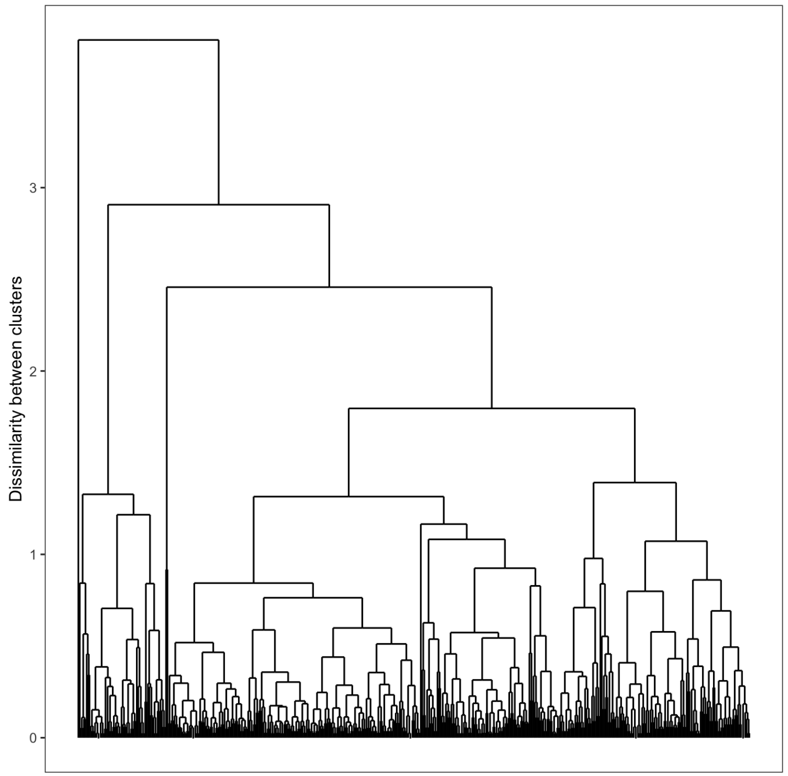


Results in a **chaining** effect

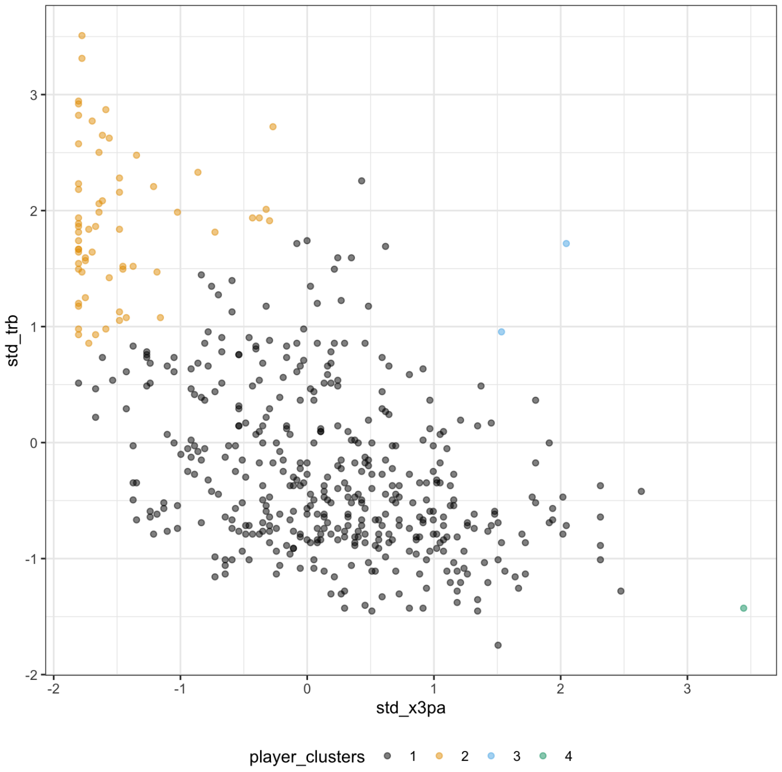


Average linkage example

Change the `method` argument to `average`



Closer to `complete` but varies in compactness

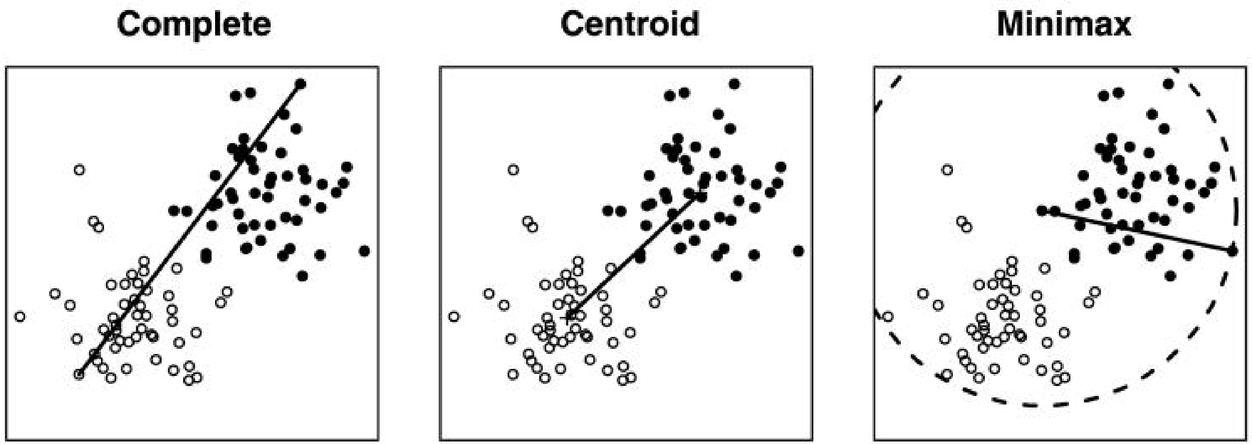


More linkage functions

- **Centroid linkage**: Computes the dissimilarity between the centroid for cluster 1 and the centroid for cluster 2
  - i.e. distance between the averages of the two clusters
  - use `method = centroid`
- **Ward's linkage**: Merges a pair of clusters to minimize the within-cluster variance
  - i.e. aim is to minimize the objection function from K-means
  - can use `ward.D` or `ward.D2` (different algorithms)

Minimax linkage

- Each cluster is defined **by a prototype** observation (most representative)
- **Identify the point whose farthest point is closest** (hence the minimax)



- Use this minimum-maximum distance as the measure of cluster dissimilarity
- Dendogram interpretation: each point point is $\leq h$ in dissimilarity to the **prototype** of cluster
- **Cluster centers are chosen among the observations themselves - hence prototype**

Minimax linkage example

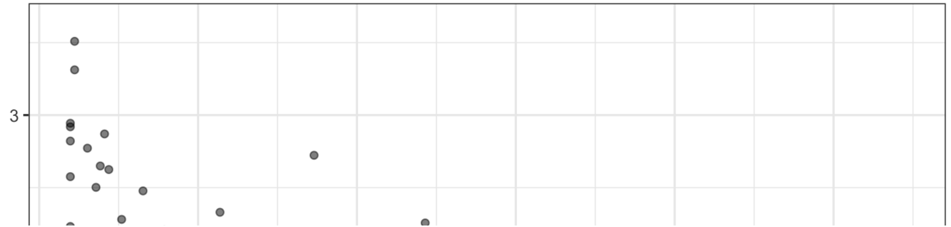
- Easily done in `R` via the `protoclust` package
- Use the `protoclust()` function to apply the clustering to the `dist()` object

```
library(protoclust)
nba_minimax <- protoclust(player_dist)
# ggdendrogram(nba_minimax,
#              theme_dendro = FALSE,
#              labels = FALSE,
#              leaf_labels = FALSE) +
#   labs(y = "Maximum dissimilarity from prot
#   theme_bw() +
#   theme(axis.text.x = element_blank(),
#         axis.title.x = element_blank(),
#         axis.ticks.x = element_blank(),
#         panel.grid = element_blank())
```

Minimax linkage example

- Use the `protocut()` function to make the cut
- But then access the cluster labels `cl`

```
library(protoclust)
minimax_player_clusters <-
  protocut(nba_minimax, k = 4)
nba_filtered_stats %>%
  mutate(player_clusters =
           as.factor(minimax_player_clusters$
  ggplot(aes(x = std_x3pa, y = std_trb,
             color = player_clusters)) +
  geom_point(alpha = 0.5) +
  ggthemes::scale_color_colorblind() +
  theme_bw() +
  theme(legend.position = "bottom")
```



Minimax linkage example

- Want to check out the prototypes for the three clusters
- `protocut` returns the indices of the prototypes (in order of the cluster labels)

```
minimax_player_clusters$protos
```

```
## [1] 468 347 103 251
```

- View these player rows using `slice`:

```
nba_filtered_stats %>%
  dplyr::select(player, pos, age, std_x3pa, std_trb) %>%
  slice(minimax_player_clusters$protos)
```

```
## # A tibble: 4 × 5
##   player            pos     age std_x3pa std_trb
##   <chr>             <chr> <dbl>    <dbl>   <dbl>
## 1 Domantas Sabonis C-PF     25   -1.02    1.99
## 2 Jalen Suggs      PG       20    0.161  -0.691
## 3 Luka Dončić      PG       22    1.53    0.955
## 4 Ben McLemore     SG       28    2.47   -1.28
```

Wrapping up...

- For context, how does player position (`pos`) relate to our clustering results?

```
table("Clusters" = minimax_player_clusters$cl, "Positions" = nba_filtered_stats$pos)
```

```
##           Positions
## Clusters   C C-PF PF PF-SF PG PG-SG SF SF-SG SG SG-PG SG-SF
##        1 71    2 34     0  0     0  8     0  3     0     0
##        2 13    0 54     1 88     1 76     5 90     2     4
##        3  1    0  4     0  2     0  6     0  3     0     0
##        4  0    0  1     0  2     0  2     0  9     1     0
```

- Can see positions tend to fall within particular clusters...
- *What's the way to visually compare the two labels?*
- **We can easily include more variables** - just changes our distance matrix
- But we might want to explore **soft** assignments instead...