Tutorial 6

Lars Abt

Setup: Pakete installieren

```
pacman::p_load(ggplot2, data.table)
```

Wie gewohnt starten wir damit, dass wir unseren package Manager "pacman" laden, und diesen verwenden um die benötigten Pakete, falls notwendig, zu installieren und anschließend zu laden.

Erstellen der zu bearbeitenden Punkte

```
# Punkte definieren
points <- data.table(
  name = c("A1", "A2", "A3", "A4", "A5", "A6", "A7", "A8"),
  x = c(2, 2, 8, 5, 7, 6, 1, 4),
  y = c(10, 5, 4, 8, 5, 4, 2, 9)
)

# Kopie der Punkte erstellen, um die bisherigen Werte nicht zu verwerfen (für später)
basic_points <- copy(points)

# Euklidische Distanz berechnen
dist_matrix <- as.matrix(dist(points, method = "euclidean"))
```

```
Warning in dist(points, method = "euclidean"): NAs durch Umwandlung erzeugt
```

```
print(dist_matrix)
```

```
          1        2         3        4        5        6        7        8
1  0.000000 6.123724 10.392305 4.415880 8.660254 8.831761 9.874209 2.738613
2  6.123724 0.000000  7.449832 5.196152 6.123724 5.049752 3.872983 5.477226
3 10.392305 7.449832  0.000000 6.123724 1.732051 2.449490 8.916277 7.842194
4  4.415880 5.196152  6.123724 0.000000 4.415880 5.049752 8.831761 1.732051
5  8.660254 6.123724  1.732051 4.415880 0.000000 1.732051 8.215838 6.123724
6  8.831761 5.049752  2.449490 5.049752 1.732051 0.000000 6.595453 6.595453
7  9.874209 3.872983  8.916277 8.831761 8.215838 6.595453 0.000000 9.327379
8  2.738613 5.477226  7.842194 1.732051 6.123724 6.595453 9.327379 0.000000
```

Wir haben die Punkte definiert, und können uns die euklidische Distanz zwischen den Punkten in einer Matrix ausgeben lassen. Dies dient zur Visualisierung und dem besseren Verständnis.

Seeds/Startpunkte definieren

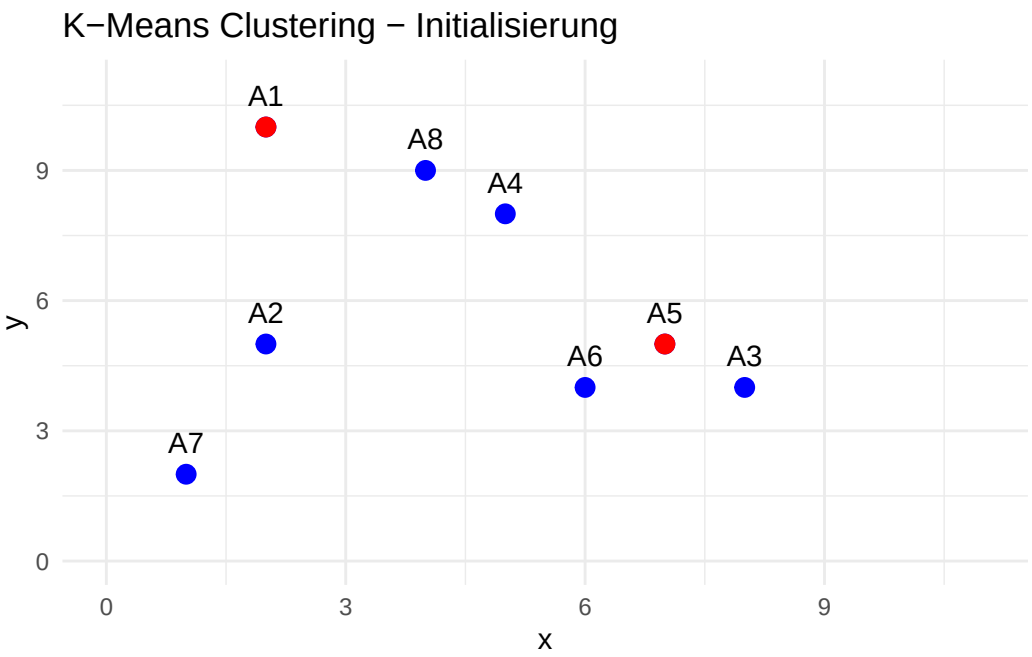
```r
# Funktioniert mit data.table:
seeds <- points[name %in% c("A1", "A5")]

# Funktioniert nicht mit data.table:
# seeds <- points[c("A1", "A5")]
```

Die Startpunkte für die Clusterzentren speichern wir in einem weiteren data.table “seeds” ab. Hierbei handelt es sich um die Punkte A1 und A5.

Plotten unserer Punkte, sodass wir eine bessere Übersicht haben

```r
basic_plot <- ggplot(points, aes(x = x, y = y, label = name)) +
  geom_point(color = "blue", size = 3) +
  geom_text(vjust = -1) +
  geom_point(data = seeds, aes(x = x, y = y), color = "red", size = 3) +
  xlim(0, 11) +
  ylim(0, 11) +
  ggtitle("K-Means Clustering - Initialisierung") +
  theme_minimal()
basic_plot
```



Wir sehen unsere 10 Punkte, und heben dabei unsere beiden Startwerte farblich hervor.

Funktionen definieren

```
# Beispiel: Funktion einer Summe:
summe_ab_v1 = function (a, b) return (a + b)
summe_ab_v2 = function (a, b) { (a + b) }

# Test der Funktionalität
print(summe_ab_v1(8, 15))
```

```
[1] 23
```

```
print(summe_ab_v2(8, 15))
```

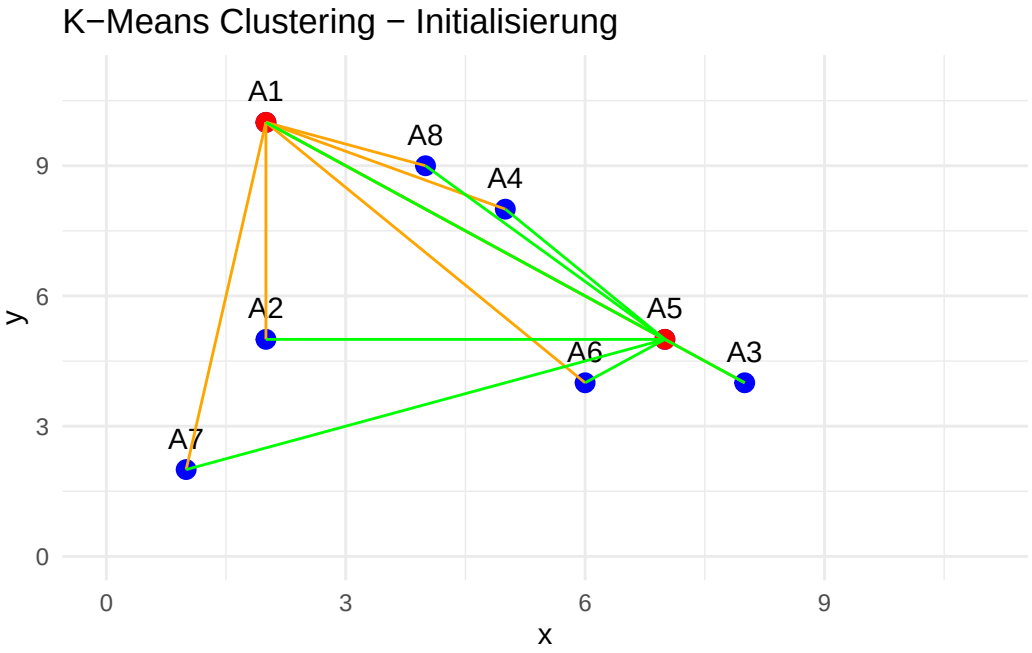
```
[1] 23
```

```
# Funktion zur Berechnung der euklidischen Distanz
euclidian_distance <- function(point, center) {
  sqrt(sum((point - center)^2))
}
```

Wir schreiben uns eine Funktion “euclidian_distance”, die die euklidische Distanz zwischen zwei Punkten berechnet. Diese Funktion wird später benötigt, um die Distanz zwischen den Punkten und den Clusterzentren zu berechnen. Wir geben ihr zwei Argumente mit, “point” und “center”, die die beiden Punkte darstellen, zwischen denen die Distanz berechnet werden soll. Wenn wir diese Funktion verwenden müssen wir darauf achten, dass wir immer beide Argumente mitgeben.

Plotten der Distanzen zu unseren Seeds

```
enhanced_plot <- basic_plot + geom_segment(data = points, aes(x = x, y = y, xend = seeds$x[1]
  geom_segment(data = points, aes(x = x, y = y, xend = seeds$x[2], yend = seeds$y[2]), color
enhanced_plot
```



Wir sehen ein Netz, welches von unseren Startpunkten aufgespannt wird, und erhalten somit eine gute erste Übersicht über die mögliche Lage unserer Clusterzentren.

Distanzen unserer Punkte zu den Seeds/Startpunkten berechnen, mit Hilfe unserer neuen Funktion

```r
dist_to_seeds <- matrix(NA, nrow = nrow(points), ncol = nrow(seeds))
for (i in 1:nrow(points)) {
  for (j in 1:nrow(seeds)) {
    dist_to_seeds[i, j] <- euclidian_distance(points[i, .(x, y)], seeds[j, .(x, y)])
  }
}
colnames(dist_to_seeds) <- paste(seeds$name)
rownames(dist_to_seeds) <- paste(points$name)
print(dist_to_seeds)
```

```
         A1       A5
A1 0.000000 7.071068
A2 5.000000 5.000000
A3 8.485281 1.414214
A4 3.605551 3.605551
A5 7.071068 0.000000
A6 7.211103 1.414214
A7 8.062258 6.708204
A8 2.236068 5.000000
```

Wir sehen, dass unsere Distanzen erfolgreich berechnet werden, und wir für den Punkt A1, sowie den Punkt A5, je einmal den Wert 0 erhalten, und alle anderen Werte positive Distanzen darstellen. Die Funktion erfüllt somit ihren Zweck und wir können die Distanzen zu Punkt A1 (in der ersten Spalte) mit den Distanzen zu Punkt A5 (in der zweiten Spalte) vergleichen.

Zuweisung der Clusterzentren und Berechnung der neuen Zentren und plotten der Ergebnisse

```r
# Zuweisung der Cluster
points[, cluster_assignment := apply(dist_to_seeds, 1, which.min)]
points
```

```
Index: <name>
     name     x     y cluster_assignment
   <char> <num> <num>              <int>
1:     A1     2    10                  1
2:     A2     2     5                  1
3:     A3     8     4                  2
```

```
4:     A4     5     8                 1
5:     A5     7     5                 2
6:     A6     6     4                 2
7:     A7     1     2                 2
8:     A8     4     9                 1
```

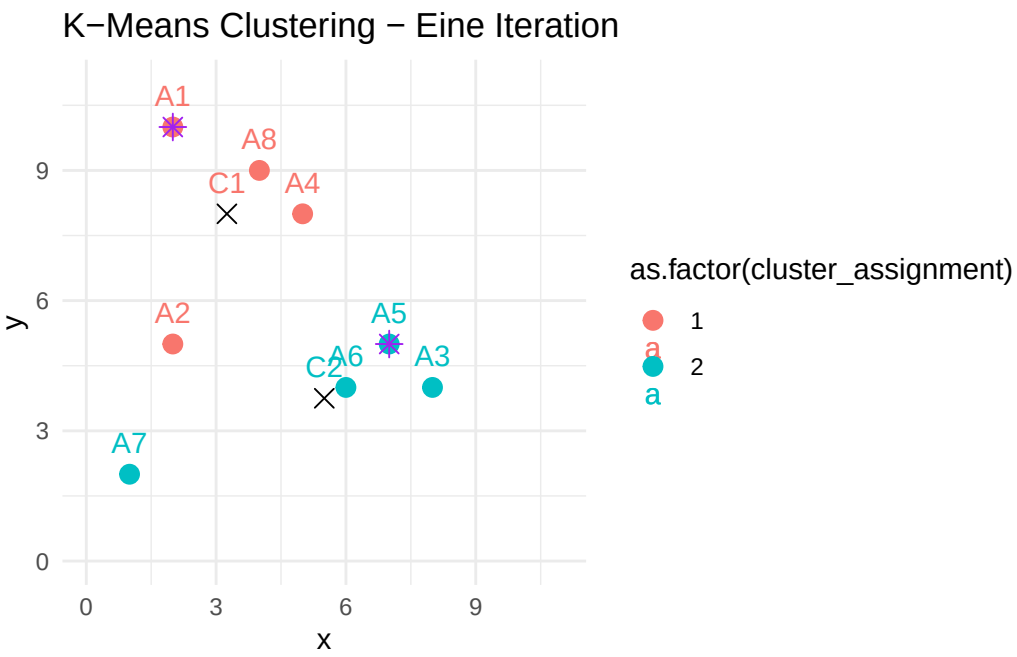
```
# Neue Clusterzentren berechnen
new_centroids <- points[, .(x = mean(x), y = mean(y)), by = cluster_assignment]
new_centroids[, name := paste("C", cluster_assignment, sep = "")]
cat("Neue Clusterzentren:\n")
```

```
Neue Clusterzentren:
```

```
print(new_centroids)
```

```
   cluster_assignment     x     y   name
                <int> <num> <num> <char>
1:                  1  3.25  8.00     C1
2:                  2  5.50  3.75     C2
```

```
# Visualisierung der Cluster und Zentren
ggplot(points, aes(x = x, y = y, color = as.factor(cluster_assignment), label = name)) +
  geom_point(size = 3) +
  geom_text(vjust = -1) +
  geom_point(data = seeds, aes(x = x, y = y), color = "purple", size = 3, shape = 8) +
  geom_point(data = new_centroids, aes(x = x, y = y), color = "black", size = 3, shape = 4) +
  geom_text(data = new_centroids, aes(x = x, y = y), vjust = -1) +
  xlim(0, 11) +
  ylim(0, 11) +
  ggtitle("K-Means Clustering - Eine Iteration") +
  theme_minimal()
```



Wir sehen unsere beiden neuen Zentren, C1 und C2. Insbesondere C1 ist ein schönes anschauliches Beispiel, wie der Punkt ausgewählt wird. C2 hingegen verdeutlicht, dass auch scheinbare Ausreißer, wie A7, in die Berechnung mit einfließen müssen, und die Lage des Zentrums dann ein wenig verschieben können.

Extra: Über das Aufgabenblatt hinaus

Alternatives Vorgehen: Wir nutzen die in R implementierte kmeans-Funktion, welche uns die Arbeit erleichtert

Mit der kmeans Funktion kann man die ganze Arbeit, die wir händisch gemacht haben, sehr schnell von R erledigen lassen. Wir nutzen dafür die Punkte, die wir am Anfang definiert haben, und die Startpunkte, die wir ebenfalls definiert haben. Die Funktion kmeans() erwartet als Argumente die Datenpunkte und die Zentren, und berechnet dann die Clusterzuweisungen und Zentren für uns.

```
# Wir nutzen wieder unsere Punkte vom Anfang
kmeans_points <- copy(basic_points)

# Wir arbeiten mit einem Seed und unserer Kopie von points
set.seed(12345)
kmeans_results <- kmeans(kmeans_points[, .(x, y)], centers = seeds[, .(x, y)])
```

```r
# Extrahieren der Clusterzuweisungen und Zentren
kmeans_points[, cluster_kmeans := kmeans_results$cluster]
center_kmeans <- data.table(kmeans_results$centers)
center_kmeans[, name := c("C1", "C2")]
cat("Neue Clusterzentren mit kmeans-Funktion:\n")
```

```
Neue Clusterzentren mit kmeans-Funktion:
```

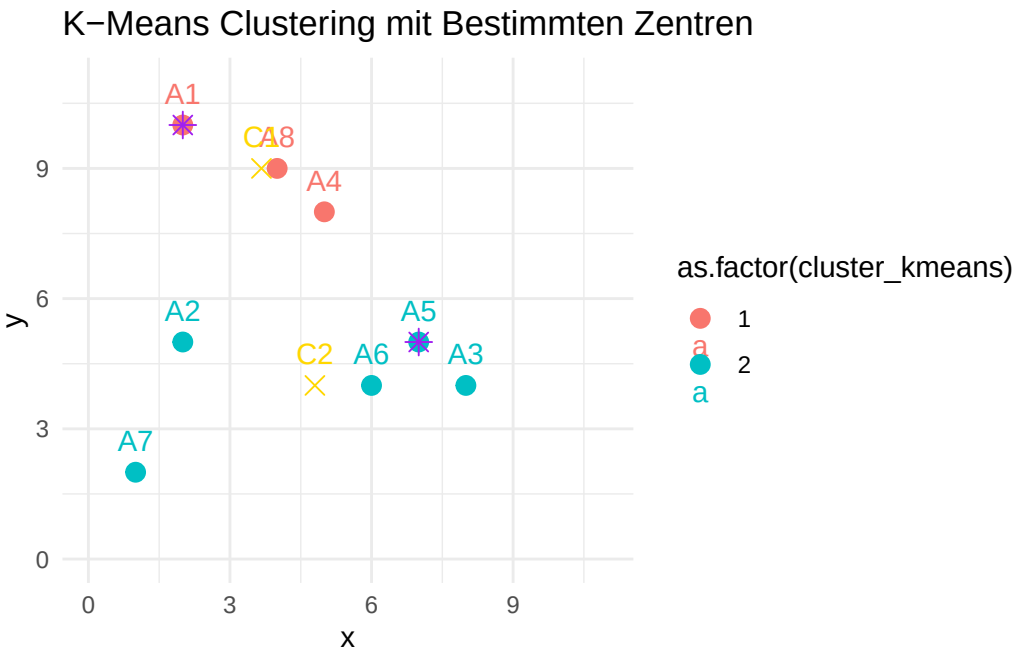
```r
print(center_kmeans)
```

```
          x     y   name
      <num> <num> <char>
1: 3.666667     9     C1
2: 4.800000     4     C2
```

Wir berechnen, in einer Zeile, die Resultate der kmeans-Funktion, und können mit ihrer Hilfe erneut die Zuweisungen der Punkte zu den Clustern, sowie die Zentren berechnen. Wir sehen, dass die Zentren, die wir erhalten, mit unseren eigenen Berechnungen übereinstimmen.

Visualisierung der Ergebnisse

```r
kmeans_plot <- ggplot(kmeans_points, aes(x = x, y = y, color = as.factor(cluster_kmeans), lab
  geom_point(size = 3) +
  geom_text(vjust = -1) +
  geom_point(data = seeds, aes(x = x, y = y), color = "purple", size = 3, shape = 8) +
  geom_point(data = center_kmeans, aes(x = x, y = y), color = "gold", size = 3, shape = 4) +
  geom_text(data = center_kmeans, aes(x = x, y = y), color = "gold", vjust = -1) +
  xlim(0, 11) +
  ylim(0, 11) +
  ggtitle("K-Means Clustering mit Bestimmten Zentren") +
  theme_minimal()
kmeans_plot
```



Wir sehen nun eine etwas abweichende Zuordnung zu unseren beiden Clustern. Dies liegt an der abweichenden Implementierung in R (unsere "per Hand" Methode ist eine Vereinfachung). Zudem werden mit Hilfe der Funktion kmeans() die Clusterzentren solange berechnet, bis wir eine Convergence erreichen, das heißt, sich die Zentren nicht mehr verändern. Dies ist ein weiterer Unterschied zu unserer Methode, bei der wir nur eine Iteration durchführen (vgl. diesbezüglich die Folien und Visualisierung aus der Vorlesung).

Zum Vergleich: Eine Visualisierung, wie die Ergebnisse mit zwei anderen Startpunkten aussehen könnten

```
# Neue Seeds definieren und neue Punkte nutzen
new_points <- copy(basic_points)
new_seeds <- points[name %in% c("A3", "A7")]

# Wir arbeiten wieder mit einem Seed und unserer Kopie von points
set.seed(12345)
new_kmeans_results <- kmeans(new_points[, .(x, y)], centers = new_seeds[, .(x, y)])

# Extrahieren der Clusterzuweisungen und Zentren
new_points[, cluster_kmeans := new_kmeans_results$cluster]
new_center_kmeans <- data.table(new_kmeans_results$centers)
```

```r
new_center_kmeans[, name := c("C1", "C2")]
cat("Neue Clusterzentren mit kmeans-Funktion:\n")
```

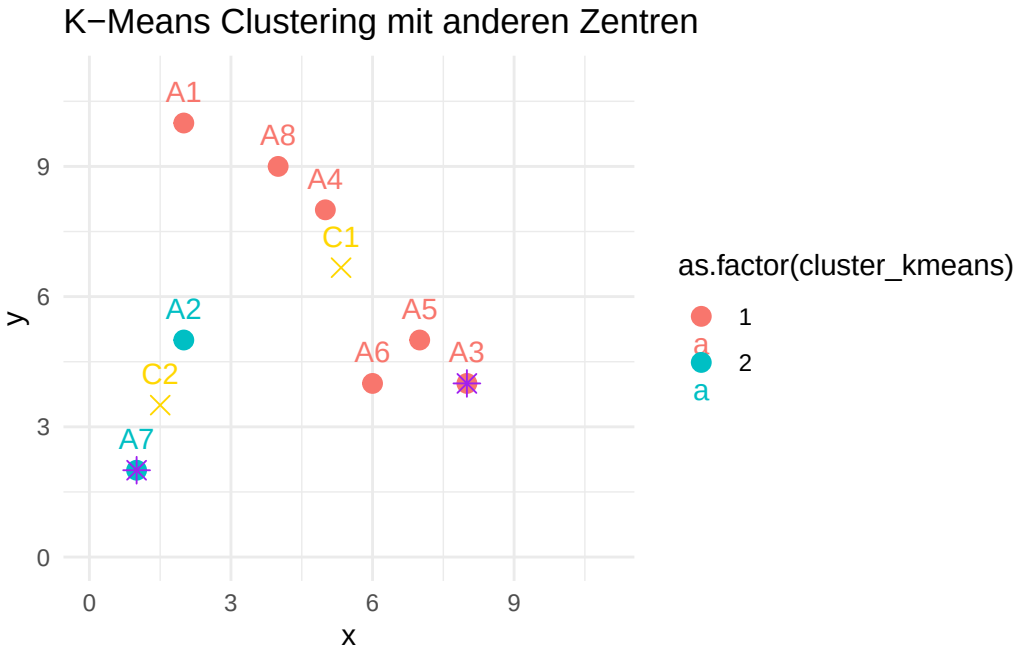
```
Neue Clusterzentren mit kmeans-Funktion:
```

```r
print(new_center_kmeans)
```

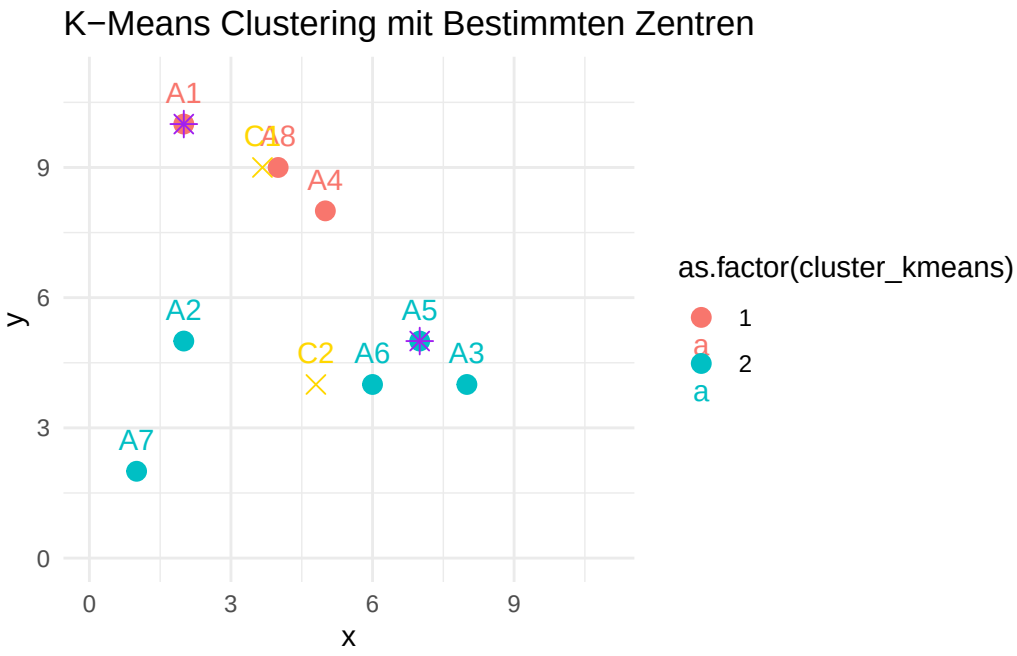
```
          x        y   name
      <num>    <num> <char>
1: 5.333333 6.666667     C1
2: 1.500000 3.500000     C2
```

Jetzt plotten wir das ganze noch einmal:

```r
new_kmeans_plot <- ggplot(new_points, aes(x = x, y = y, color = as.factor(cluster_kmeans), la
  geom_point(size = 3) +
  geom_text(vjust = -1) +
  geom_point(data = new_seeds, aes(x = x, y = y), color = "purple", size = 3, shape = 8) +
  geom_point(data = new_center_kmeans, aes(x = x, y = y), color = "gold", size = 3, shape = 4
  geom_text(data = new_center_kmeans, aes(x = x, y = y), color = "gold", vjust = -1) +
  xlim(0, 11) +
  ylim(0, 11) +
  ggtitle("K-Means Clustering mit anderen Zentren") +
  theme_minimal()
new_kmeans_plot
```




```
# Zum Vergleich noch der alte Plot:
kmeans_plot
```



Abschließend können wir festhalten, dass die Wahl der Startpunkte von großer Bedeutung ist, da sie das Ergebnis entscheidend beeinflussen. Es kann sinnvoll sein, sich mehrere mögliche Startpunkte anzuschauen.