Clustering Goals in the German Bundesliga

Maximilian Klemp

2023-07-15

Examining soccer teams' playing styles by identifying clusters of goals scored in the German Bundesliga.

```
library(tidyverse)
library(flexclust)
options(scipen=999)

path <- '/home/max/drive/dshs/teaching/abschlussarbeiten/Herrmann_Sven/data/Rohdaten.csv'

dat <- read.csv(path)

colnames(dat) <- c("mday", "goalid", "home", "away", "team", "home_score", "away_score", "int
                   "action_leading2goal", "pressure", "preassist", "assist", "scorer", "zone_

df_cluster <- dat %>%
  select(home, away, team, tactical_context, passes_before_goal, previous_action, action_lead
  mutate_all(as.factor) %>%
  mutate(
    tactical_context = factor(tactical_context, levels = c("Freistoß (direkt)", "Eckstoß (ou
                              ),
    action_leading2goal = factor(action_leading2goal, levels = c("Ballkontrolle + Dribbling -
                                 ),
    pressure = factor(pressure, levels = c("No-Pressure", "Pressure"), labels = c("noPressure
    previous_action = factor(previous_action, levels = c("Pass < (10m)", "Shot", "Cross", "Pa
    )
```

Football teams adopt different strategies. The way in which they create their goal opportunities generates a footprint of their playing style. Identifying an opponent's playing style can help to prepare for a match against that team. Since goals can be described by a multitude of features, it is unlikely to observe the same goal twice and to identify styles by looking at the individual goals. However, it is possible to find groups of similar goals. The method from the

realm of Machine Learning that is best suitable for this purpose is **Cluster Analysis**. This notebook demonstrates how to apply this method to a dataset of goals scored in the German Bundesliga season 2022/2023.

The data used in this notebook was created by annotating the videos of all 971 goals scored in the 306 matches of the season. They have been assigned different features, e.g. the tactical context from which the goals were scored or the number of passes played before the goal. From these features of each goal, clusters of goals shall be identified, revealing different prototypical playing styles in the creation of goals. Using these clusters, preferred goal scoring styles are examined per team.

A snippet of the raw dataset is displayed below.

```
knitr::kable(head(df_cluster))
```

home	away	team	tactical_context	passes_before_goal	previous_action	goal_action	leading_2_goal_zone	pressure_assist	assist_assist	assist_goal
SGE	FCB	FCB	directFreekick	0	shortPass	directShot	noPressure	12	15	6
SGE	FCB	FCB	outswingCornerkick	4	shot	directShot	pressure	20	18	9
SGE	FCB	FCB	openPlay	7+	cross	header	noPressure	11	17	9
SGE	FCB	FCB	openPlay	7+	shortPass	directShot	noPressure	18	24	14
SGE	FCB	FCB	counterAttack	5	shortPass	receptionAndShot	noPressure	14	14	9
SGE	FCB	SGE	counterAttack	4	shortPass	directShot	noPressure	11	14	5

Creating numerical features

In order to find different, prototypical categories of goals and to assign them to teams,we will perform **k-means clustering** on the data. The data that we got from our video analysis consists of 8 categorical features for each goal. Since cluster analysis works with numerical data only, we have to transform our categorical variables into numerical ones. This is achieved using a two-step process, by (1) applying **One-Hot-Encoding** on the categorical variables and then performing **dimensionality reduction** on the binary variables. As a side note, technically there are options to perform clustering on categorical variables (like k-modes), but experimenting with both yielded better results for the numerical approach.

One-Hot Encoding

First, our **8 categorical variables** are transformed into a so-called **dummy matrix**. This means that for each variable, a separate column is created for each possible category. Each column then contains only the values 0 or 1. Below is a snippet of the first few rows and columns of this matrix:

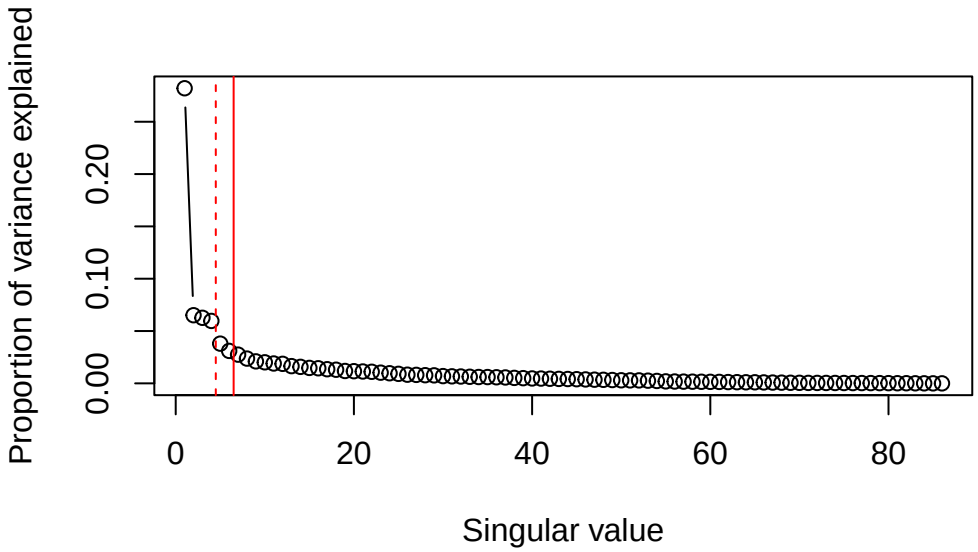
```
df_cluster <- df_cluster %>%
  select(-home, -away, -team)
mat <- model.matrix(~. - 1, data = df_cluster)
df_dummies <- as.data.frame(mat)
df_dummies[,1:10] %>%
  head() %>%
  knitr::kable()
```

tactical_	tactical_	tactical_	tactical_	tactical_	tactical_	tactical_	tactical_	tactical_	goal1
1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0

Dimensionality reduction

Now this matrix contains 86 columns, which are hard to interpret. At the same time, it can be assumed that some of the columns are highly correlated with each other, because they consist of some of the same variables. To address these two problems, a dimensionality reduction technique is now applied. It is assumed that the initial 87 dimensions can be combined in such a way that they still contain most of the input information, but can be reduced to fewer dimensions. For binary datasets, the so-called **Singular Value Decomposition (SVD)** is preferred. This is now applied to the data. A representation of the data in 1 to 86 dimensions is tried out and it is tested how much percent of the variance in the original dataset can be explained by a representation in the respective dimensionality. The gain in information per additional dimension is particularly important. It is therefore examined how much additional variance can be explained by each additional dimension. This can be examined with the following diagram, the so-called **Scree Plot**:

```
svd_result <- svd(df_dummies)

prop_var_explained <- svd_result$d^2 / sum(svd_result$d^2)
cum_var_explained  <- cumsum(prop_var_explained)
plot(prop_var_explained, type = "b", xlab = "Singular value", ylab = "Proportion of variance

#plot(cumsum(prop_var_explained), type = "b", xlab = "Anzahl Dimensionen", ylab = "Kumuliert
abline(v=4.5, col="red", lty = 2)
abline(v=6.5, col="red")
```



Natürlicherweise wird durch jede zusätzliche Dimension auch mehr Varianz erklärt. Allerdings nimmt bei hochdimensionalen Daten ab einer gewissen Dimension meist die zusätzlich erklärte Varianz ab. Ein "Ellenbogenpunkt", welcher eine rapide Änderung der Steigung des Graphen anzeigt, identifiziert die kleinste Dimensionalität, welche noch einen signifikanten Informationsgewinn bedeutet. Nach diesem Punkt flacht die Kurve deutlich ab und jede zusätzliche Dimension verursacht nur noch einen marginalen zusätzlichen Informationsgewinn. Der "Ellenbogenpunkt" ist nur selten eindeutig zu bestimmen. In unserem Diagramm kommen prinzipiell zwei Punkte infrage, welche durch die gestrichelte und die durchgezogene Linie angezeigt werden. Da es sich insgesamt um ein hochdimensionales Problem handelt, wird die durchgezogene Linie gewählt, weil insgesamt bis hierhin noch zusätzliche Varianz erklärt wird und danach der Graph eindeutig abflacht. Diese Linie zeigt eine Dimensionalität von 6 an. Diese wird verwendet, um einen neuen, dimensionsreduzierten Datensatz mit 6 Variablen zu ermitteln. Diese Dimensionen sind nun nicht mehr interpretierbar, können aber für die Clusteranalyse verwendet werden. Eine Übersicht über den neuen Datensatz sieht folgendermaßen aus:

```
## NOTES:
  # based on 80% rule, 22 dimensions should be retained
  # based on visual elbow point, 7 should be retained ( or 29)

# Choose the number of dimensions to retain
num_dims <- 6

# Project the data onto these dimensions
reduced_data <- as.matrix(df_dummies) %*% svd_result$v[, 1:num_dims]
knitr::kable(head(reduced_data))
```

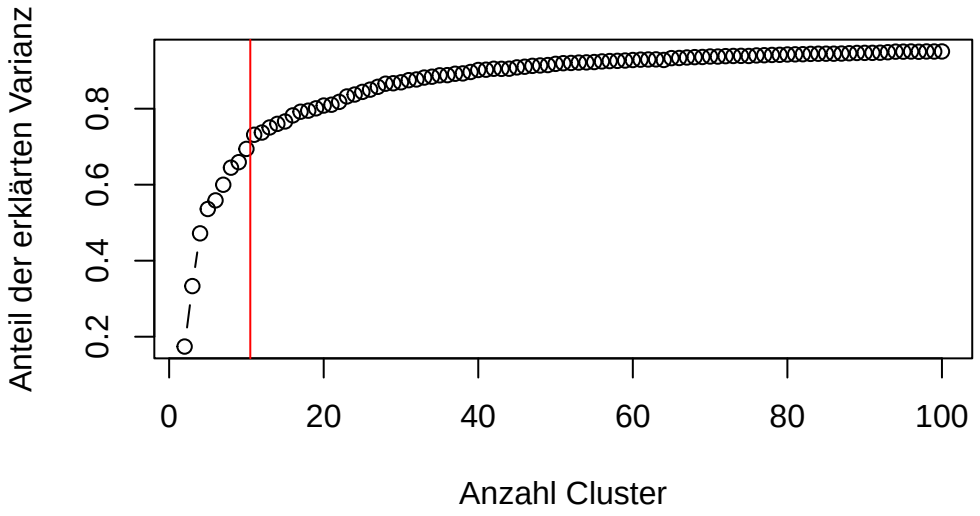
-0.452979	-0.5566330	0.0899807	-0.2644000	0.4703923	-0.2242417
-1.686565	-0.6473897	0.2128024	-0.4694705	-0.0475723	0.3277312
-1.305201	0.4821976	-1.1560128	-0.4364203	-0.4972119	0.0646575
-0.891784	-0.7588065	-0.9605267	0.0877693	0.8343932	0.2202717
-1.127883	-0.0974868	0.6735477	0.7246956	-0.9630310	-0.8314999
-1.004543	-0.5489809	0.8316487	0.6467081	0.4734059	-0.8147062

Clusteranalyse

Auf die dimensionsreduzierte Matrix, welche 6 Variablen und 871 Beobachtungen enthält, wird nun eine Clusteranalyse angewandt. Hierbei wird der übliche **kmeans-Clustering-Algorithmus** angewendet. Bei Verwendung dieses Algorithmus wird eine Anzahl **k** an Clustern festgelegt und der Algorithmus ordnet jede Beobachtung einem der Cluster zu. Dabei werden anhand eines Distanzmaßes (üblicherweise die euklidische Distanz, welche im zweidimensionalen Raum als die räumliche Distanz zwischen zwei Punkten in einem kartesischen Koordinatenssystem dargestellt werden kann) die paarweisen Abstände zwischen Beobachtungen berechnet. Die Beobachtungen werden dann den k Clustern so zugewiesen, dass die durchschnittliche Distanz zu den anderen Beobachtungen im eigenen Cluster **minimiert** und die durchschnittliche Distanz zu den Beobachtungen in den anderen Clustern **maximiert** wird. Die Wahl der optimalen Anzahl von Clustern **k** erfolgt, ähnlich wie bei der Dimensionsreduktion, empirisch. Dazu wird der **kmeans-Algorithmus** unter Wahl unterschiedlicher Ausprägungen von k wiederholt ausgeführt und es werden bestimmte Kennzahlen zwischen den Durchläufen verglichen. Beim **kmeans-Clustering** wird als solche Kennzahl typischerweise der Anteil der Gesamtvarianz, welche durch die Einteilung in die verschiedenen Cluster erklärt werden kann, verwendet. Auch hier wird der Zugewinn an erklärter Varianz pro zusätzlichem Cluster grafisch dargestellt und es wird der Punkt gesucht, an dem die zusätzliche Hinzunahme weiterer Cluster nur noch in marginalem Informationszugewinn resultiert. Nachfolgend wird dieses Diagramm für die Werte $k = 2$ bis $k = 100$ dargestellt:

```r
################### CLUSTERING
# Create an empty vector to store the total within-cluster sum of squares for each k
tot_withinss <- numeric(99)
betweenss    <- numeric(99)
totss        <- numeric(99)

# Loop over different numbers of clusters
for (k in 2:100) {
  set.seed(123) # For reproducibility
  kmeans_result <- kmeans(reduced_data, centers = k)
  tot_withinss[k-1] <- kmeans_result$tot.withinss
  betweenss[k-1] <- kmeans_result$betweenss
```

```
  totss[k-1] <- kmeans_result$totss
}

# Plot total within-cluster sum of squares as a function of the number of clusters
var_explained <- betweenss / totss
plot(2:100, var_explained, type = "b", xlab = "Anzahl Cluster", ylab = "Anteil der erklärten
abline(v=10.5, col = 'red')
```




```
# with new dimensionality in SVD, 10 clusters appear
# NOTES: gain in variance explained suggests 16 clusters
# tot within ss plot suggests 17 clusters

# NOTES: not necessarily interpretable goal types;
#   assigning goals to clusters and then counting per team or per player might also be intere
```

Auch hier ist eine eindeutige Identifikation des Ellenbogenpunkts nicht möglich. Es wird eine Anzahl von $k = 10^*$ Clustern gewählt, weil bis hierhin der letzte große Informationszugewinn erfolgt. Dementsprechend identifiziert die Clusteranalyse **10 verschiedene Arten der Torerzielung**. Nachfolgend werden die 871 Tore im Datensatz ihrem jeweiligen Cluster zugeordnet und die Anzahl der vertretenen Cluster pro Mannschaft analysiert. Außerdem wird für jedes Cluster ein "prototypisches" Tor erstellt, welches anhand der Modi der jeweiligen Variablen pro Cluster ermittelt wird, da die 6 Variablen, welche zur Bestimmung der Cluster verwendet wurden, nicht inhaltlich interpretiert werden können.

```
#### FINAL IMPLEMENTATION
k <- 10
```

```
kmeans_result <- kmeans(reduced_data, centers = k)
dat$cluster <- kmeans_result$cluster
df_cluster$cluster <- kmeans_result$cluster
```

Interpretation der Cluster

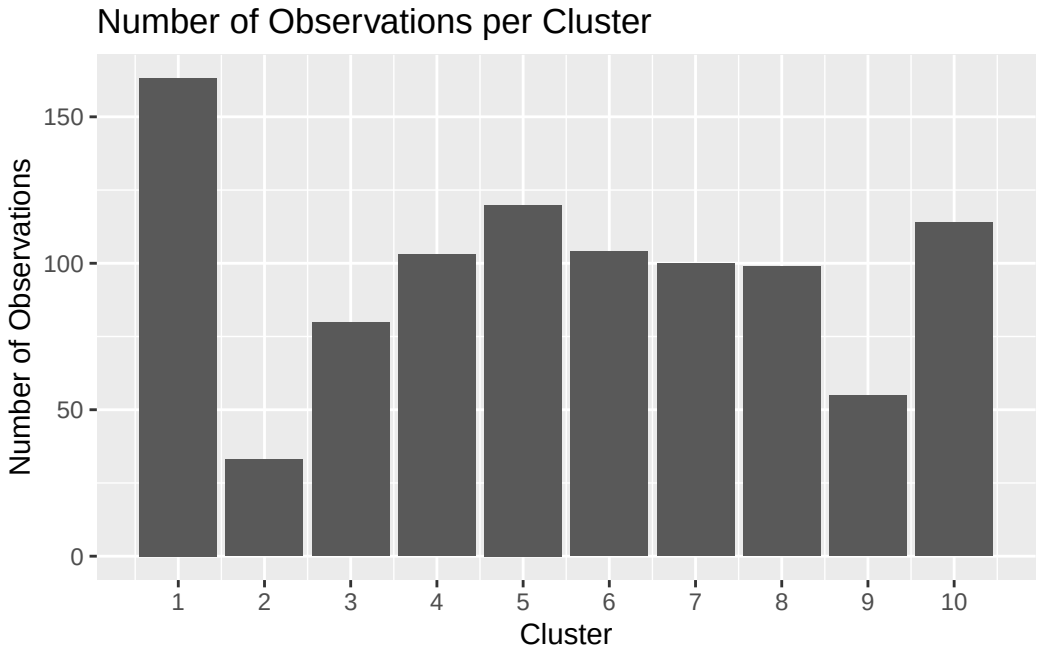
Um zunächst eine Interpretation der 10 verschiedenen Cluster zu ermöglichen, wird nachfolgend für jedes Cluster ein "prototypisches" Tor ermittelt. Dazu werden jeweils alle Beobachtungen pro Cluster gesammelt und für jede der 8 Variablen im Originaldatensatz, welche die Tore semantisch beschreiben, der Modus ermittelt. Der Modus gibt jene Kategorie einer Variable an, welche im Datensatz am häufigsten auftritt. Auf diese Weise kann das "häufigste" Tor im Datensatz ermittelt werden. Es folgt eine Tabelle, welche pro Cluster die häufigste Kombination von Variablen, und somit jeweils ein "prototypisches" Tor zeigt.

```
mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

prototypes <- df_cluster %>%
  group_by(cluster) %>%
  summarise_all(mode)

knitr::kable(prototypes)
```

cluster	tactical_context	passes_before_goal	previous_action	action_leading_2_goal	pressure	zone_preassist	zone_assist	zone_goal
1	openPlay	7+	shortPass	directShot	pressure	14	18	9
2	directFreekick	1	shortPass	directShot	noPressure	n.a	14	5
3	counterAttack	3	shortPass	receptionAndShot	pressure	14	14	9
4	openPlay	1	cross	header	pressure	n.a	16	9
5	openPlay	7+	shortPass	receptionAndShot	pressure	14	14	9
6	counterAttack	2	shortPass	directShot	pressure	16	18	14
7	penalty	0	shortPass	directShot	noPressure	n.a	18	9
8	counterAttack	2	shortPass	dribbleAndShot	pressure	14	14	5
9	inswingCornerkick	1	cross	header	pressure	n.a	16	14
10	counterAttack	4	shortPass	directShot	pressure	14	18	9

Die nachfolgende Grafik zeigt die Anzahl erzielter Tore pro Cluster über die gesamte Stichprobe hinweg:

```r
df_cluster_perteam <- dat %>%
  group_by(team, cluster) %>%
  summarise(count = n(), .groups = "drop") %>%
  group_by(team) %>%
  mutate(proportion = round(count / sum(count), 2))

# Distribution of goals across Clusters
ggplot(df_cluster_perteam, aes(x = cluster, y = count)) +
  geom_bar(stat = "identity") +
  labs(title = "Number of Observations per Cluster",
       x = "Cluster",
       y = "Number of Observations") +
  scale_x_continuous(breaks=1:10)
```



Analyse der Arten erzielter Tore in Abhängigkeit der Mannschaften

Die mithilfe der Clusteranalyse ermittelten Kategorien der Torerzielung sollen nun im Hinblick auf die Mannschaften untersucht werden, welche auf unterschiedliche Weise Tore erzielt haben. Dazu werden die 871 erzielten Tore pro Mannschaft und pro Cluster gezählt. Es soll untersucht werden, welche Mannschaften auf welche Weise die Mehrzahl ihrer Tore schießen.

Die nachfolgende Tabelle zeigt die Verteilung von Toren in die verschiedenen Cluster in Abhängigkeit von den Mannschaften:

```r
df_abs_values <- df_cluster_perteam %>%
  select(-proportion) %>%
  pivot_wider(names_from = cluster, values_from = count, names_prefix = "c") %>%
  replace(is.na(.), 0)

df_rel_values <- df_cluster_perteam %>%
  select(-count) %>%
  pivot_wider(names_from = cluster, values_from = proportion, names_prefix = "c") %>%
  replace(is.na(.), 0)

teams <- df_rel_values$team
df_rel_values$team <- NULL
baseline_probs <- round(table(dat$cluster) / dim(dat)[1], 3)

mat <- as.matrix(df_rel_values)
df_diffs <- round(data.frame(mat - matrix(rep(baseline_probs, 18), nrow=18, ncol=10, byrow =
df_diffs$team <- teams

df_diffs_long <- df_diffs %>%
  pivot_longer(cols = c1:c10, names_to = "cluster", values_to = "proportion_diff") %>%
  mutate(cluster = as.numeric(gsub("c", "", cluster)))

knitr::kable(df_abs_values)
```

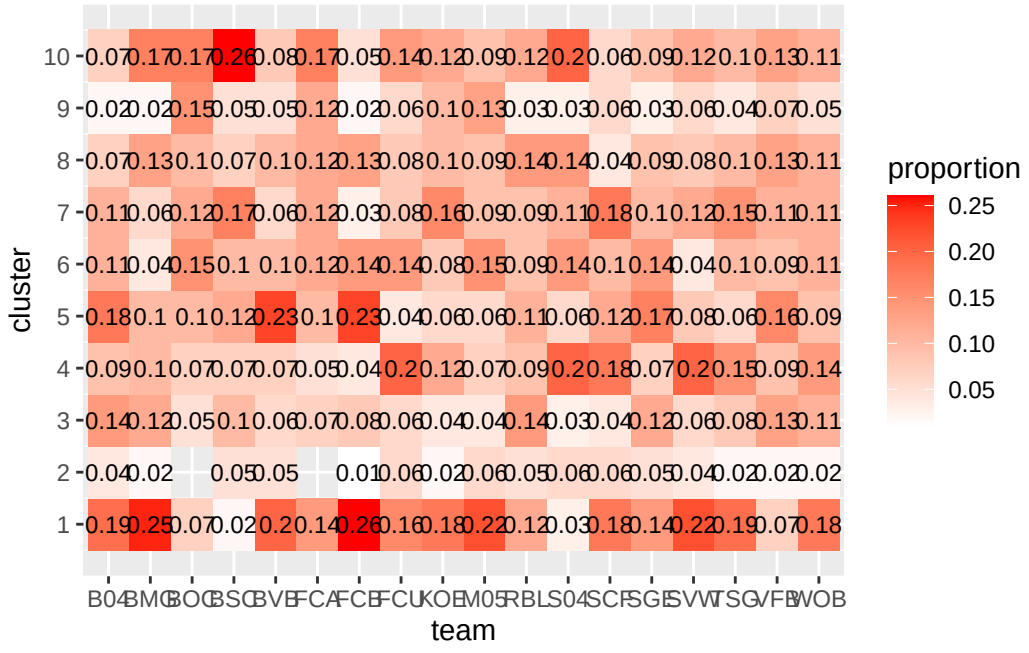
team	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
B04	11	2	8	5	10	6	6	4	1	4
BMG	13	1	6	5	5	2	3	7	1	9
BOC	3	0	2	3	4	6	5	4	6	7
BSC	1	2	4	3	5	4	7	3	2	11
BVB	17	4	5	6	19	8	5	8	4	7
FCA	6	0	3	2	4	5	5	5	5	7
FCB	24	1	7	4	21	13	3	12	2	5
FCU	8	3	3	10	2	7	4	4	3	7
KOE	9	1	2	6	3	4	8	5	5	6
M05	12	3	2	4	3	8	5	5	7	5
RBL	8	3	9	6	7	6	6	9	2	8
S04	1	2	1	7	2	5	4	5	1	7
SCF	9	3	2	9	6	5	9	2	3	3
SGE	8	3	7	4	10	8	6	5	2	5
SVW	11	2	3	10	4	2	6	4	3	6
TSG	9	1	4	7	3	5	7	5	2	5

team	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
VFB	3	1	6	4	7	4	5	6	3	6
WOB	10	1	6	8	5	6	6	6	3	6

Verteilung der Cluster pro Mannschaft

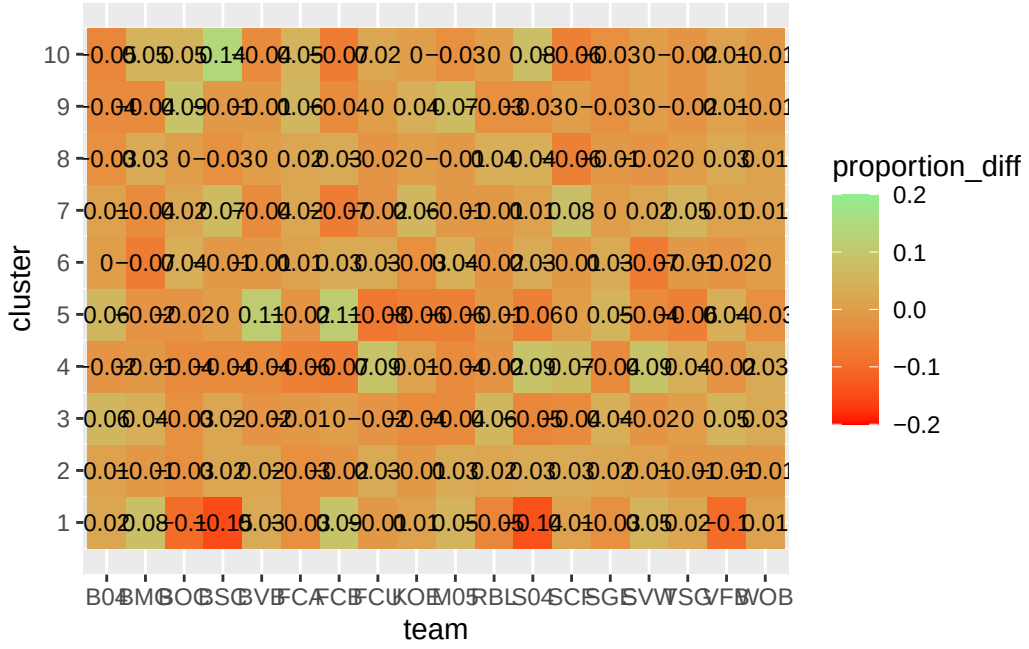
Die absoluten Häufigkeiten der Tore pro Cluster pro Mannschaft werden durch die absolute Anzahl geschossener Tore pro Mannschaft verzerrt. Um ein Bild über die Spielstile der Mannschaften zu zeichnen, müssen die relativen Häufigkeiten von Toren pro Cluster in Abhängigkeit von der Gesamtzahl geschossener Tore pro Mannschaft betrachtet werden. Die nachfolgende Heatmap zeigt daher die relative Häufigkeit von Toren in den 10 Clustern in Abhängigkeit von der Mannschaft. Daraus lassen sich sowohl Rückschlusse auf die grundlegende Verteilung von Toren auf die 10 Cluster ziehen als auch auf Besonderheiten im Spielstil einzelner Mannschaften.

```
## Distribution of clusters per team
ggplot(df_cluster_perteam, aes(team, cluster)) +
  geom_tile(aes(fill=proportion)) +
  geom_text(aes(label=proportion), size = 3) +
  scale_fill_gradient(low = "white", high = "red") +
  scale_y_continuous(breaks = 1:10)
```



In dieser Grafik zeigen sich bereits Muster bestimmter Mannschaften. Allerdings beinhaltet sie auch die Basisrate einzelner Cluster. Daher wird im Folgenden noch die durchschnittliche Häufigkeit von Toren in den 10 Clustern über alle Mannschaften hinweg von den individuellen relativen Häufigkeiten abgezogen, sodass für jede Mannschaft dargestellt wird, wie stark die Häufigkeit der Torerzielung vom Durchschnitt abweicht.

```
## Deviation of teams' distribution from baseline distribution
ggplot(df_diffs_long, aes(team, cluster)) +
  geom_tile(aes(fill=proportion_diff)) +
  geom_text(aes(label=proportion_diff), size = 3) +
  #scale_fill_viridis(limits = c(-.2, .151), oob = scales::squish)
  scale_fill_gradient(low = "red", high = "lightgreen", limits = c(-.2, .2), oob = scales::s
  scale_y_continuous(breaks = 1:10)
```




```
#write.csv(df_cluster, '/home/max/sciebo/local/dshs/teaching/abschlussarbeiten/Herrmann_Sven/
#write.csv(df_abs_values, '/home/max/sciebo/local/dshs/teaching/abschlussarbeiten/Herrmann_Sv
#write.csv(df_rel_values, '/home/max/sciebo/local/dshs/teaching/abschlussarbeiten/Herrmann_Sv
#write.csv(df_diffs, '/home/max/sciebo/local/dshs/teaching/abschlussarbeiten/Herrmann_Sven/da
```