Arbeitsblatt 2

Aufgabe 1: Distanzen

Wir betrachten die synthetischen Daten mit den beiden Variablen x_1 und x_2.

Tabelle 1: Datensatz

	A	B	C	D	E	F	G	H	I	J	K	L
x1	10.0	8.0	11.0	9	12.0	45.0	55.0	59.0	52.0	56	58.0	57.0
x2	2.8	5.2	4.1	10	6.5	5.2	5.6	1.2	9.5	11	8.5	4.9

a) Lesen Sie die Beschreibung der R-Funktionen `dist(...)` und `daisy(...)` aus dem R-Package `cluster` durch. Bestimmen Sie die euklidischen Distanzen zwischen dem Objekt A und den Objekten D, F und G mit Hilfe von `dist(...)` oder `daisy(...)`.

Sie müssen dazu zuerst das Paket `cluster` installieren und laden (`library(cluster)`). Den Datensatz können Sie wie folgt erzeugen.

```
D.S <- data.frame(x1=c(10, 8, 11, 9, 12, 45, 55, 59, 52, 56, 58, 57),
                  x2=c(2.8, 5.2, 4.1, 10, 6.5,5.2,5.6,1.2, 9.5,11, 8.5, 4.9))
rownames(D.S) <- LETTERS[1:nrow(D.S)]
```

```
library(cluster)
round(daisy(D.S),1)
```

```
Dissimilarities :
     A    B    C    D    E    F    G    H    I    J    K
B  3.1
C  1.6  3.2
D  7.3  4.9  6.2
E  4.2  4.2  2.6  4.6
F 35.1 37.0 34.0 36.3 33.0
G 45.1 47.0 44.0 46.2 43.0 10.0
H 49.0 51.2 48.1 50.8 47.3 14.6  5.9
I 42.5 44.2 41.4 43.0 40.1  8.2  4.9 10.9
J 46.7 48.3 45.5 47.0 44.2 12.4  5.5 10.2  4.3
K 48.3 50.1 47.2 49.0 46.0 13.4  4.2  7.4  6.1  3.2
L 47.0 49.0 46.0 48.3 45.0 12.0  2.1  4.2  6.8  6.2  3.7

Metric :  euclidean
Number of objects : 12
```

```
round(dist(D.S),1)
```

```
     A    B    C    D    E    F    G    H    I    J    K
B  3.1
C  1.6  3.2
D  7.3  4.9  6.2
E  4.2  4.2  2.6  4.6
F 35.1 37.0 34.0 36.3 33.0
```

```
G 45.1 47.0 44.0 46.2 43.0 10.0
H 49.0 51.2 48.1 50.8 47.3 14.6  5.9
I 42.5 44.2 41.4 43.0 40.1  8.2  4.9 10.9
J 46.7 48.3 45.5 47.0 44.2 12.4  5.5 10.2  4.3
K 48.3 50.1 47.2 49.0 46.0 13.4  4.2  7.4  6.1  3.2
L 47.0 49.0 46.0 48.3 45.0 12.0  2.1  4.2  6.8  6.2  3.7
```

```
## Damit wir einfach die gewünschten Distanzen sehen:
round(as.matrix(daisy(D.S))[c("D", "F", "G"), "A"],1)
```

```
   D    F    G
 7.3 35.1 45.1
```

```
# manuelle Berechnung
sqrt(sum((D.S[c("A"),] - D.S[c("D"),])^2))
```

```
[1] 7.269113
```

b) Bestimmen Sie die mit der Standardabweichung skalierten euklidischen Distanzen zwischen dem Objekt A und den Objekten D, F und G mit Hilfe von `daisy(...)`.

```
round(as.matrix(daisy(D.S, stand=TRUE))[c("D", "F", "G"), "A"],1)
```

```
  D   F   G
3.0 1.9 2.4
```

```
# Standardisierung in daisy
x <- scale(D.S, center = TRUE, scale = FALSE)
sx <- colMeans(abs(x), na.rm = TRUE)
x <- scale(x, center = FALSE, scale = sx)
round(as.matrix(daisy(x))[c("D", "F", "G"), "A"],1)
```

```
  D   F   G
3.0 1.9 2.4
```

Die skalierten Distanzen sind deutlich unterschiedlich. Die Distanz AF ist jetzt am kleinsten! ACHTUNG! Die Skalierung in daisy(..., stand=TRUE) geschieht durch den Mittelwert der absoluten Abweichung vom Zentrum!!

c) Ergänzen Sie den ursprünglichen Datensatz um eine Faktorvariable und berechnen Sie danach erneut die Distanzen zwischen dem Objekt A und den Objekten D, F und G mit Hilfe von `daisy(...)`.

```
D.S2 <- data.frame(x1=c(10, 8, 11, 9, 12, 45, 55, 59, 52, 56, 58, 57),
                   x2=c(2.8, 5.2, 4.1, 10, 6.5,5.2,5.6,1.2, 9.5,11, 8.5, 4.9),
                   x3=c("a", "a", "a", "b", "b", "b", "c", "c", "c", "d", "d", "d"))
rownames(D.S2) <- LETTERS[1:nrow(D.S2)]
```

```
round(as.matrix(daisy(D.S2, metric="gower"))[c("D", "F", "G"), "A"],2)
```

```
   D    F    G
0.58 0.64 0.72
```

```
round(as.matrix(daisy(D.S2))[c("D", "F", "G"), "A"],2)
```

```
   D    F    G
0.58 0.64 0.72
```

```
# manuelle Berechnung
1/3*(sum(abs(D.S2[c("A"),1:2] - D.S2[c("D"),1:2])/apply(D.S,2, FUN=function(x) diff(range(x)))) +
       (1-sum(D.S2[c("A"),3] == D.S2[c("D"),3])))
```

```
[1] 0.5847672
```

Falls verschiedene Variablentypen vorhanden sind, nimmt daisy() automatisch 'metric="gower".

d) Behandeln Sie die Variable x1 wie eine ordinale Variable und berechnen Sie danach noch einmal die Distanzen zwischen dem Objekt A und den Objekten D, F und G (`daisy(D.S2, metric="gower", type=list(ordratio=1))`). Was geschieht hier?

```
round(as.matrix(daisy(D.S2, metric="gower", type=list(ordratio=1)))[c("D", "F", "G"), "A"],2)
```

```
   D    F    G
0.61 0.51 0.58
```

```
#manuelle Umwandlung
D.S2$x1 <- as.ordered(D.S2$x1)
round(as.matrix(daisy(D.S2, metric="gower"))[c("D", "F", "G"), "A"],2)
```

```
   D    F    G
0.61 0.51 0.58
```

Wie beabsichtigt, wird die Variable x1 in eine ordinale Variable transformiert.

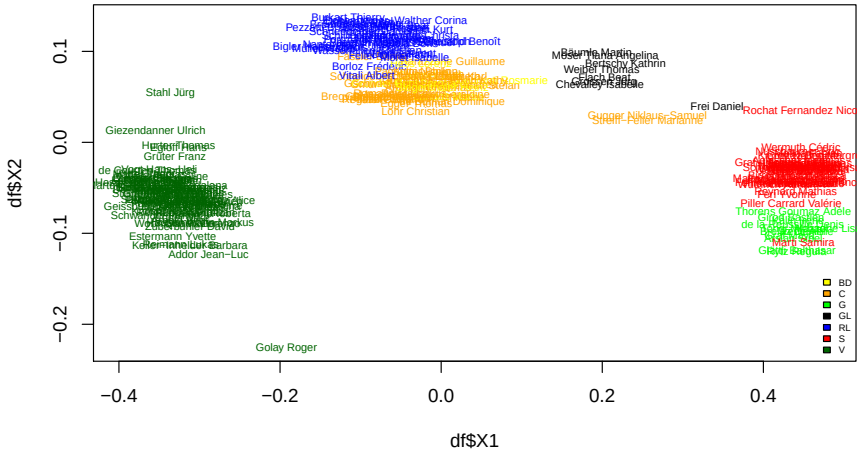
Aufgabe 2: Metrische MDS

Laden Sie den Datensatz `voting_NR.rdata`. Es enthält zwei Datensätze: `NR_voting` ist eine "Distanz"-Matrix, welche das Abstimmungsverhalten des Schweizer Parlaments (Nationalrats) beschreibt. Die Elemente in der Distanzmatrix zeigen, wie ähnlich bzw. unähnlich (Distanz) die Parlamentarierinnen und Parlamentarier abstimmen. Der Datensatz enthält Informationen zu 199 Leuten. Die Distanzmatrix wurde basierend auf 921 Abstimmungen im Jahr 2019 erstellt (vor den Erneuerungswahlen). Die Distanz zwischen zwei Nationalräten entspricht jeweils der mittleren absoluten Differenz in allen Abstimmungen bei welchen beide anwesend waren. Ja-Stimmen wurden dabei mit 1 codiert, Nein-Stimmen mit 0, Enthaltungen mit 0.5. Der zweite Datensatz `NR_meta` enthält Metainformationen (Fraktion und Kanton) aller Nationalräte. Die Daten sind gleich sortiert.

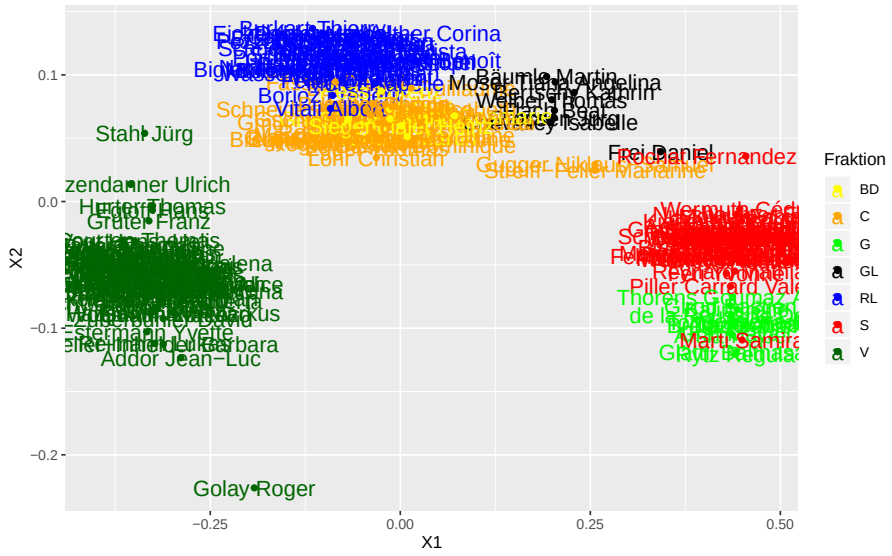
a) Checken Sie mit einer 2-dimensionalen Visualisierung, ob Mitglieder der gleichen Fraktion ähnlich abstimmen. Verwenden Sie dazu die metrische MDS-Methode (`cmdscale`). Um die Fraktionszugehörigkeit farblich zu visualisieren, können Sie in der Plot-Funktion das Argument `col` z.B. wie folgt parametrisieren:

```
col=c("yellow", "orange", "green", "black", "blue", "red", "darkgreen")[NR_meta$Fraktion]`
```

```
load("Daten/voting_NR.Rdata")
res <- cmdscale(NR_voting)
df <- data.frame(res)
plot(df$X1, df$X2, type = 'n')
text(df$X1, df$X2, label=rownames(df), cex=0.6,
     col=c("yellow", "orange", "green", "black", "blue", "red", "darkgreen")[NR_meta$Fraktion])
legend("bottomright", legend=levels(NR_meta$Fraktion), bty="n", cex=0.5,
       fill=c("yellow", "orange", "green", "black", "blue", "red", "darkgreen"))
```




```
# Alternative Visualisierung mit ggplot
plot_data <- cbind(df, NR_meta)
library(ggplot2)
ggplot(plot_data, aes(x=X1, y=X2, label=Name, colour=Fraktion))+
  geom_text(size=5) +
  geom_point() +
  scale_color_manual(values=c("yellow", "orange", "green", "black", "blue", "red", "darkgreen"))
```



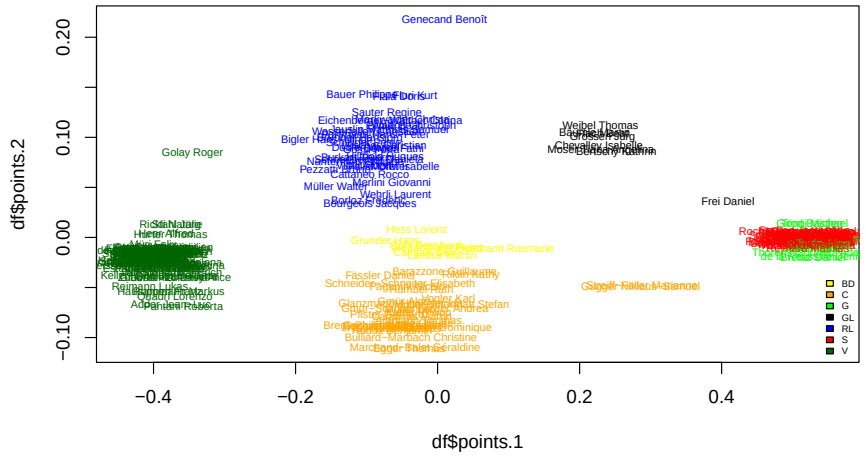
Wir können die Unterschiede zwischen den Parteien sehr gut erkennen. Wir sehen auch welche Parteien ähnlich abstimmen und welche Parlamentarie ein etwas anderes Stimmverhalten zu haben als ihre Fraktionskollegen (z.B. Roger Golay (SVP) oder Pernand Rochat (SP)).

b) Wiederholen Sie Ihre Analyse mit der nicht-metrischen MDS (Funktion `isoMDS` aus dem Paket `MASS`). Gibt es grosse Unterschied in der Visualisierung?

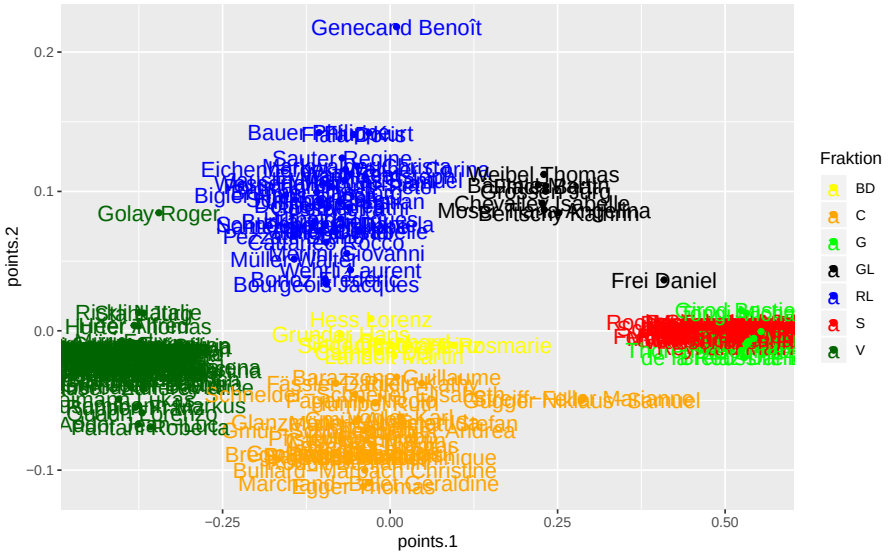
```
library(MASS)
res <- isoMDS(NR_voting)
```

```
initial  value 5.241492
iter   5 value 4.077510
iter  10 value 3.103541
iter  15 value 2.940117
iter  20 value 2.851489
final  value 2.816272
converged
```

```
df <- data.frame(res)
plot(df$points.1, df$points.2, type = 'n')
text(df$points.1, df$points.2, label=rownames(df), cex=0.6,
       col=c("yellow", "orange", "green", "black", "blue", "red", "darkgreen")[NR_meta$Fraktion])
legend("bottomright", legend=levels(NR_meta$Fraktion), bty="n", cex=0.5,
       fill=c("yellow", "orange", "green", "black", "blue", "red", "darkgreen"))
```




```
# Alternative Visualisierung mit ggplot
plot_data <- cbind(df, NR_meta)
library(ggplot2)
ggplot(plot_data, aes(x=points.1, y=points.2, label=Name, colour=Fraktion))+
  geom_text(size=5) +
  geom_point() +
  scale_color_manual(values=c("yellow", "orange", "green", "black", "blue", "red", "darkgreen"))
```



Aufgabe 3: MNIST

Wir betrachten noch einmal den MNIST-Datensatz. Dieses Mal mit 2000 zufällig ausgewählten handgeschriebenen Nummern von 0 bis 9. Nach dem Laden des Datensatzes `mnist_2k.Rdata` steht das Dataframe `x` zur Verfügung. Jede Zeile des Dataframes enthält die linearisierten Pixel-Grauwerte eines Bildes `pixel0` bis `pixel783`. Zudem gibt es eine Spalte `label` mit den Labeln.

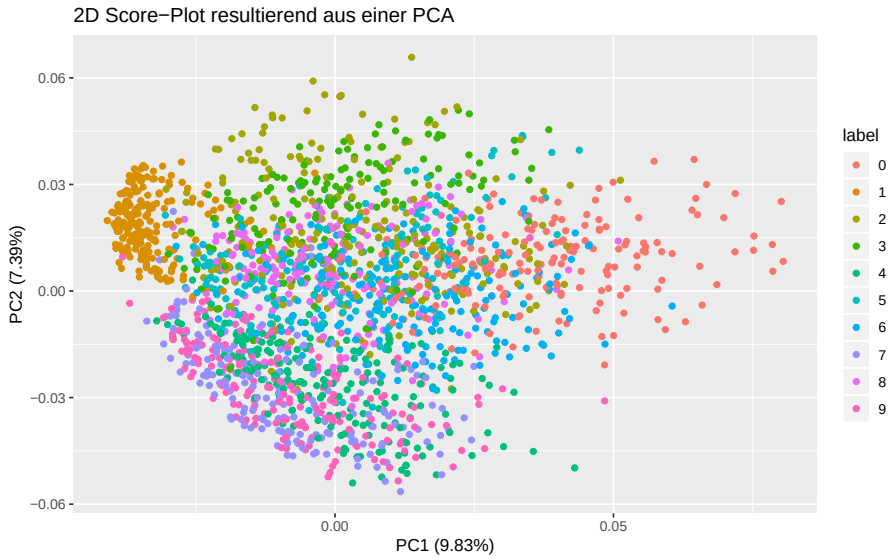
```
load('Daten/mnist_2k.Rdata')
```

Machen Sie sich mit dem Datensatz vertraut. Folgender Code hilft Ihnen dabei:

```
dim(x)
x$label <- as.factor(x$label)
table(x$label)
image_nummer <- 7 # hier können Sie ein Bild von 1 bis 2000 auswählen
bild <- matrix(as.numeric(x[image_nummer, 2:785]), ncol=28, byrow = TRUE)
image(t(bild[28:1,1:28]))
x$label[image_nummer]
```

a) Führen Sie eine Hauptkomponentenanalyse durch und visualisieren Sie das Ergebnis in einem 2D Score-Plot (beachten Sie, dass Sie die Spalte mit den Labels nicht für die PCA verwenden). Verwenden Sie Farben, um die verschiedenen Nummern zu visualisieren. Sehr einfach geht das mit der Funktion `autoplot(res_pca, data=x, colour= 'label')` aus dem Paket `ggfortify`.

```
library(ggfortify)
x$label <- as.factor(x$label)
res_pca <- prcomp(x[, -1])
autoplot(res_pca, data = x, colour = 'label', label.size = 3) +
  ggtitle("2D Score-Plot resultierend aus einer PCA")
```

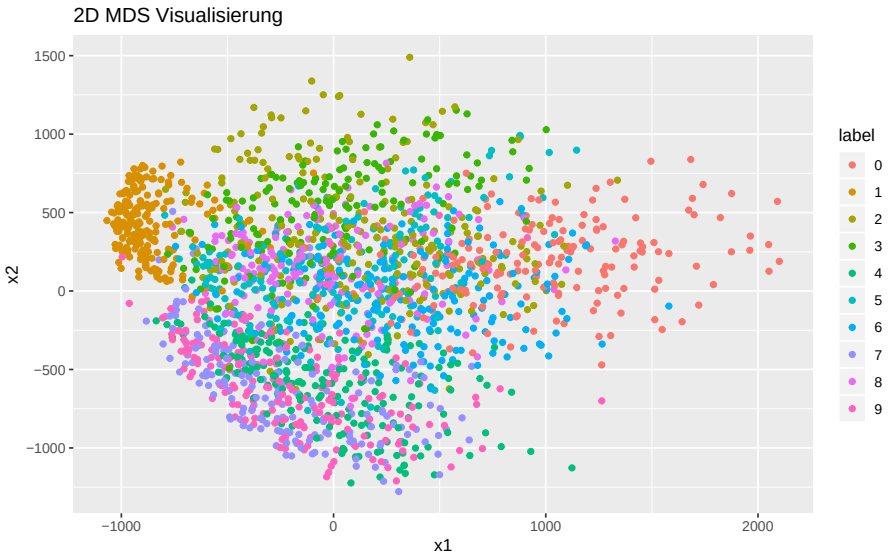


Im 2D Score-Plot sind die 10 Nummern nicht sehr schön getrennt, einige Separationen können aber schon festgestellt werden (zum Beispiel die 1 sind stark geclustert). Beachte, dass die ersten beiden Hauptkomponenten hier auch nur 26% der Variation erklären.

b) Visualisieren Sie die Daten auch mittels metrischer MDS. Achtung die Distanzberechnung braucht etwas Zeit. Vergleichen Sie das Ergebnis mit den Erkenntnissen aus a). Verwenden Sie zur Visualisierung folgenden Code:

```
d_cmd <- as.data.frame(res.cmd$Y) #res_cmd -> Ergebnis MDS
names(d_tsne) <- c("x1","x2")
d_cmd$label <- x$label
library(ggplot2)
ggplot(data = d_cmd, aes(x = x1, y = x2, col = label)) +
  geom_point() +
  ggtitle("2D MDS Visualisierung")
```

```
res.cmd <- cmdscale(dist(x[, -1]), k=2)
d_cmd <- as.data.frame(res.cmd)
names(d_cmd) <- c("x1","x2")
d_cmd$label <- x$label
library(ggplot2)
ggplot(data = d_cmd, aes(x = x1, y = x2, col = label)) +
  geom_point() +
  ggtitle("2D MDS Visualisierung")
```

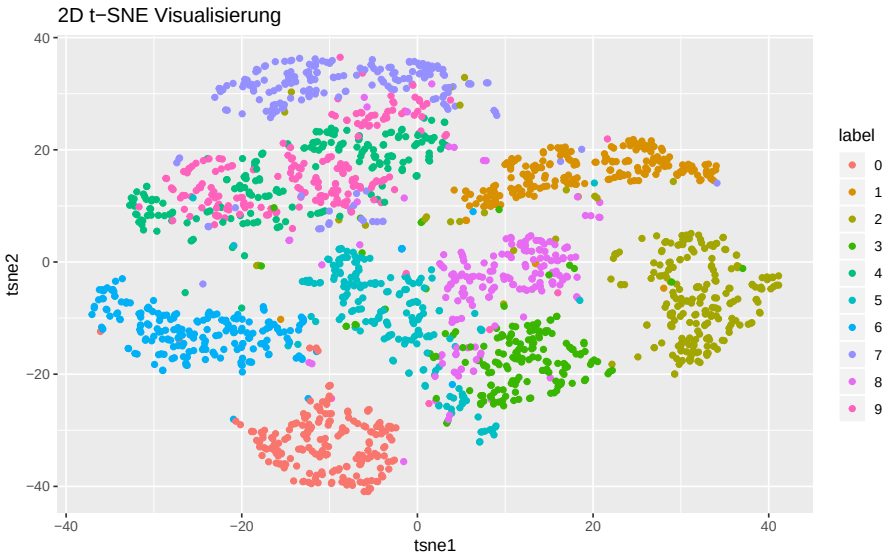


Identisch zur PCA Visualisierung.

c) Visualisieren Sie die Daten auch noch mit t-SNE (Funktion `Rtsne` aus Paket `Rtsne`). Vergleichen Sie das Ergebnis mit den Erkenntnissen aus a) und b). Verwenden Sie zur Visualisierung hier folgenden Code:

```
library(ggplot2)
d_tsne <- as.data.frame(res_tSNE$Y) #res_tSNE -> Ergebnis Rtsne
names(d_tsne) <- c("tsne1","tsne2")
d_tsne$label <- x$label
ggplot(data = , aes(x = tsne1, y = tsne2, col = label)) +
  geom_point()
```

```
library(Rtsne)
set.seed(1)
restSNE <- Rtsne(x[, -1], perplexity = 20)
d_tsne <- as.data.frame(restSNE$Y)
names(d_tsne) <- c("tsne1","tsne2")
d_tsne$label <- x$label
library(ggplot2)
ggplot(data = d_tsne, aes(x = tsne1, y = tsne2, col = label)) +
  geom_point() +
  ggtitle("2D t-SNE Visualisierung")
```



In der t-SNE 2D Visualisierung sind die 10 Nummern ziemlich gut separiert. Viel deutlicher als im 2D Score-Plot der PCA. Ein Grund dafür kann sein, dass die Daten nicht wirklich in einer Hyperebene liegen, sondern dass die Struktur komplexer ist. Je nach Wahl des perplexity-Parameters erhalten Sie ein etwas anderes Ergebnis.

Aufgabe 4: Leistungsnachweis

- Visualisieren Sie Ihren Datensatz (oder Teile davon) mittels eines geeigneten Dimensionsreduktionsansatzes Ihrer Wahl.
- Interpretieren Sie Ihr Ergebnis.