問題1:

Rの中に車の特性に関するデータセットmtcarsがある。help(mtcars)でデータの説明を読んでください。 データmtcarsを標準化し、バイクラスタ(heatmap)分析を行い、その結果を簡潔に述べよ。

In [1]:
```
# libraryのインストール
# !pip install pydataset
# !pip install seaborn
```

In [3]:
```
import pandas as pd #データ管理
from pydataset import data # Rのデータセットを取得
import seaborn as sns # 解析結果の可視化
import matplotlib as mpl # Python用可視化
import matplotlib.pyplot as plt # Python用可視化
from sklearn.cluster import AgglomerativeClustering # 階層的クラスタリング
from sklearn.cluster import KMeans # K-means
from sklearn.manifold import TSNE # t-SNE
from sklearn.metrics import confusion_matrix # 混同行列の作成
from sklearn.metrics import accuracy_score # 正解率を計算
```

In [3]:
```
# mtcarsを取得
data = data("mtcars")

#データ標準化: 特徴量ごとに単位が異なるため
df_cars_std = data.apply(lambda x: (x-x.mean())/x.std(), axis=0)
# df_cars_std.head()
```

In [4]:
```
# heatmapによる可視化
sns.set(rc={'axes.facecolor':'white', 'figure.facecolor':'white'}) # 背景色の設定
sns.clustermap(df_cars_std,  method = "ward", metric='euclidean', cmap = "RdYlBu",
               facecolor = "coral").fig.suptitle('図1. mtcarsのheatmap', y = -0.01,
fontname="MS Gothic")  # ユークリッド距離, ウォード法
```

Out[4]: Text(0.5, -0.01, '図1. mtcarsのheatmap')

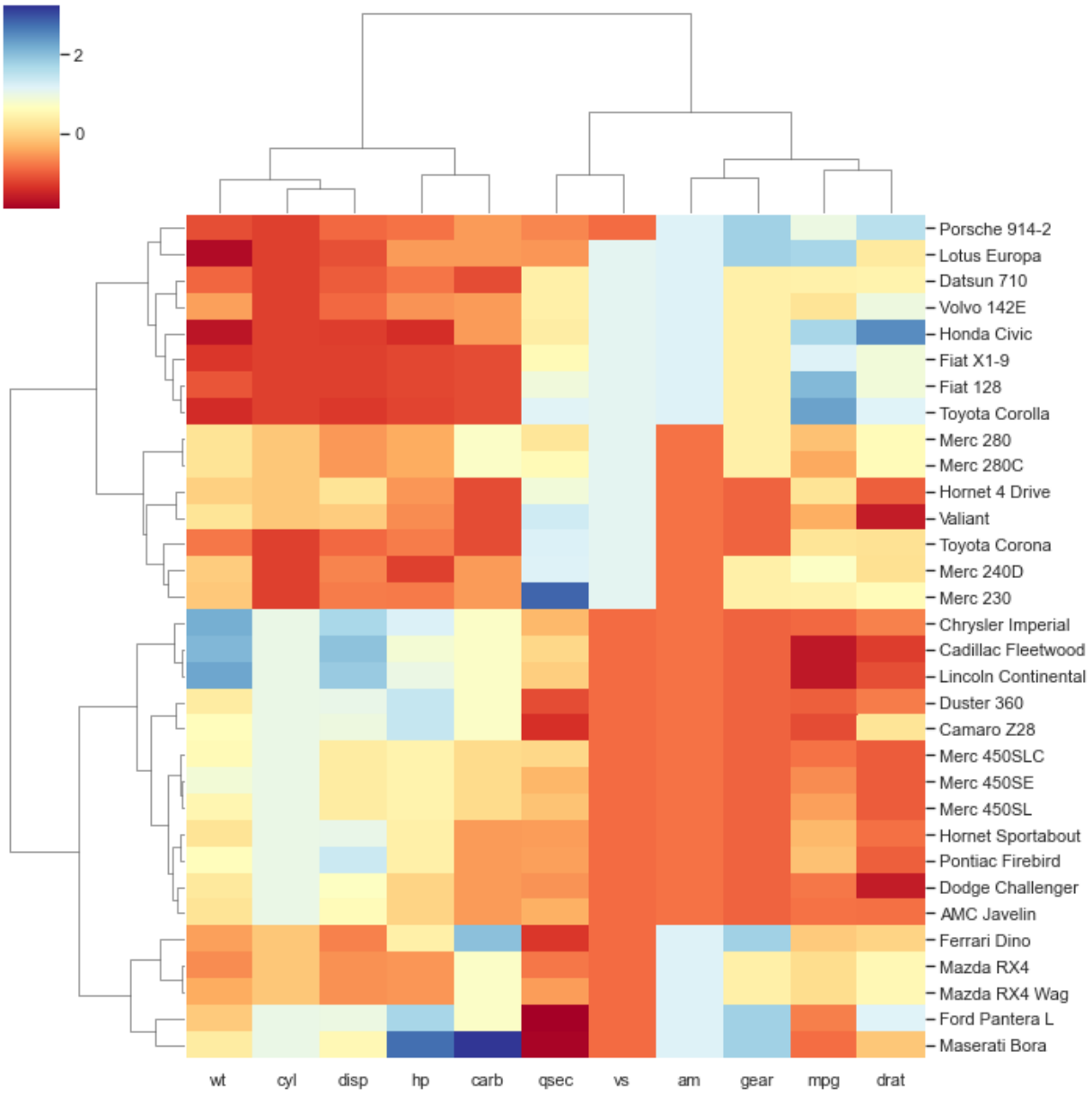

図1. mtcarsのheatmap

考察:

クラスタリングのグラフにより、行(車種)および列（変数）はそれぞれ概ね二つのクラスターに分けられると考察できる。

左上の値の範囲表示より、青に近づく方が数値が高い。

上半分のクラスター　（**Porsche 914-2, Lotus Europa, ~, merc 230**）は、**drat, gear, am,mpg, vs, qsec**の値が相対的に大きいことから、

直列型エンジンで順送りギア数が多く、 燃費がよいなどの特徴を持っている。

下半分のクラスター**(chrysler Imperial, Cadillac Fleetwood, ~, Maserati Bora)** は、**disp, cyl, hp, wt, carb**の値が相対的に大きいことから、

重くかつ大馬力で排気量が大きく、シリンダー数が多いなどの特徴がある。

問題2:

zip.trainは**7291**個の手書きの数字**(0～9)**をデータ化したものである．第**1**列は行のラベルである． **3**より小さい数字のデータを用いて以下の問いに答えよ。

1. 数字のラベルを除いて，階層的クラスタリング (ユークリッド距離,ward.D2) の結果を3クラスに分け、誤り率を求めよ。
2. 同じのデータを用いて、set.seed(0)でk-means法で3クラスに分け,その誤り率を求めたうえで問(1)の結果と比べて所見を述べよ。
3. 同じデータをt-SNEの結果の散布図を作成し、数値ラベルをプロットせよ。また、上記の2種類の方法のうち、 正解率が #高い方のクラスタリングの結果を用いて散布図を色付け、t-SNEの結果とクラスタリングの結果を考察せよ。

In [5]:
```
#zipTrainの読み込み
df_ziptrain = pd.read_csv("https://mjin.doshisha.ac.jp//data//zipTrain.csv")

# 0~3の数字のデータのみを取り出す。
df_ziptrain2 = df_ziptrain[df_ziptrain.V1<3]
```

In [6]:
```
# 問1
# 数値ラベル以外の特徴量 (変数) を取得
df_ziptrain_features = df_ziptrain2[df_ziptrain2.columns[df_ziptrain2.columns != "V1"]]

# 階層的クラスタリングの実行 (ユークリッド距離, ウォード法)
clus   = AgglomerativeClustering(n_clusters = 3, affinity = 'euclidean', linkage = 'ward')
zip_clus = clus.fit_predict(df_ziptrain_features)

# 正解率の計算
cm_hir = confusion_matrix(df_ziptrain2["V1"], zip_clus) # 混同行列の作成
# print(cm_hir)
acc_hir = accuracy_score(df_ziptrain2["V1"], zip_clus) # 正解率の算出
print(f"正解率: {acc_hir}")
print(f"誤り値: {round((1-acc_hir), 3)}")
```

```
正解率: 0.9935153583617747
誤り値: 0.006
```

正解率: **0.9935153583617747**

誤り値: **0.006**

結果: 誤り率は比較的小さい

In [7]:
```
# 問2
# K-meansの実行
kmeans_model = KMeans(n_clusters = 3, random_state = 0).fit(df_ziptrain_features)
# 分類結果のラベルを取得する
labels = kmeans_model.labels_

# 正解率の計算
cm_km = confusion_matrix(df_ziptrain2["V1"], labels) # 混同行列の作成
# print(cm_km)
acc_km = accuracy_score(df_ziptrain2["V1"], labels) # 正解率の算出
```

```
print(f"正解率: {acc_km}")
print(f"誤り値: {round((1-acc_km), 3)}")
```

```
正解率: 0.005119453924914676
誤り値: 0.995
```

正解率: **0.9201365187713311**

誤り値: **0.08**

比較: 誤り率は、約8%であり、階層的クラスタリングのward.D2法より誤り率が約13倍高い。

In [8]:

```
# # 問3
# t-SNEの実行
tsne = TSNE(n_components = 2, random_state = 0, perplexity = 30, n_iter = 1000) #
perplexity: 5~50の範囲で調整
zip_tsne = tsne.fit_transform(df_ziptrain_features)
```

In [11]:

```
# 散布図の描画
plt.figure(figsize=(10, 10), facecolor = "silver", dpi = 100)
plt.xlabel('t-SNE 1次元目', fontname="MS Gothic") # x軸ラベルに名前付け
plt.ylabel('t-SNE 2次元目', fontname="MS Gothic") # y軸ラベルに名前付け
plt.scatter(zip_tsne[:, 0], zip_tsne[:, 1],
            c=df_ziptrain2["V1"], s=30, alpha=1, cmap = "tab10") # s: プロットされる個
体の大きさ, alpha: 色の透明度
plt.title('図2. 真のラベルで着色', fontname="MS Gothic", y = -0.1) # タイトルの作成,
y: 文字の配置位置
plt.show()
```

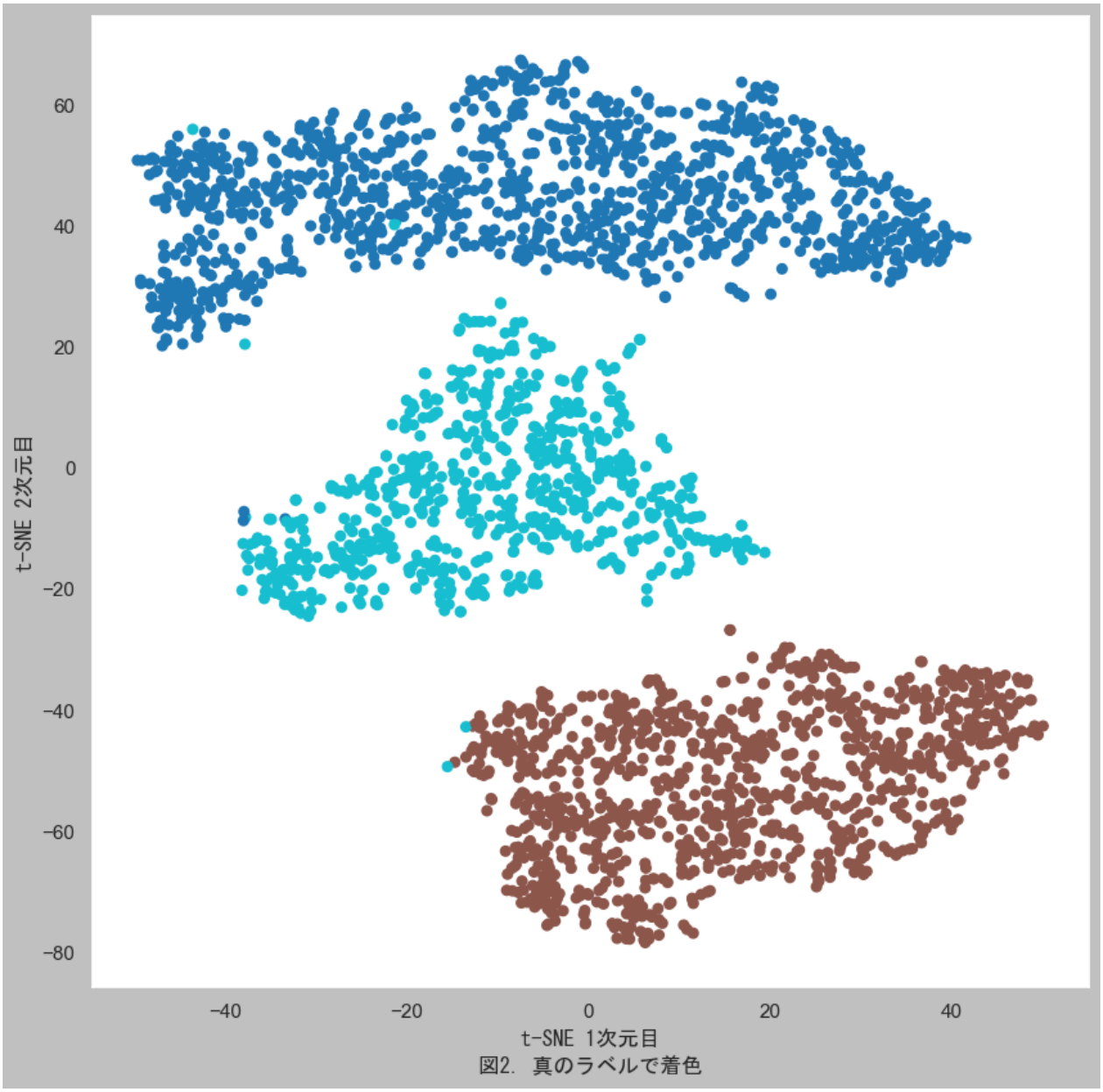

図2. 真のラベルで着色

In [12]:

```
# 散布図の作成（階層的クラスタリングの結果で着色）
plt.figure(figsize=(10, 10), facecolor = "silver", dpi = 100)
plt.xlabel('t-SNE 1次元目', fontname="MS Gothic") # x軸ラベルに名前付け
plt.ylabel('t-SNE 2次元目', fontname="MS Gothic") # y軸ラベルに名前付け
plt.scatter(zip_tsne[:, 0], zip_tsne[:, 1],
            c=zip_clus, s=30, alpha=1, cmap = "tab10") # s: プロットされる個体の大きさ, alpha: 色の透明度
plt.title('図3. 階層的クラスタリングの結果で着色', fontname="MS Gothic", y = -0.1) # タイトルの作成, y: 文字の配置位置
plt.show()
```

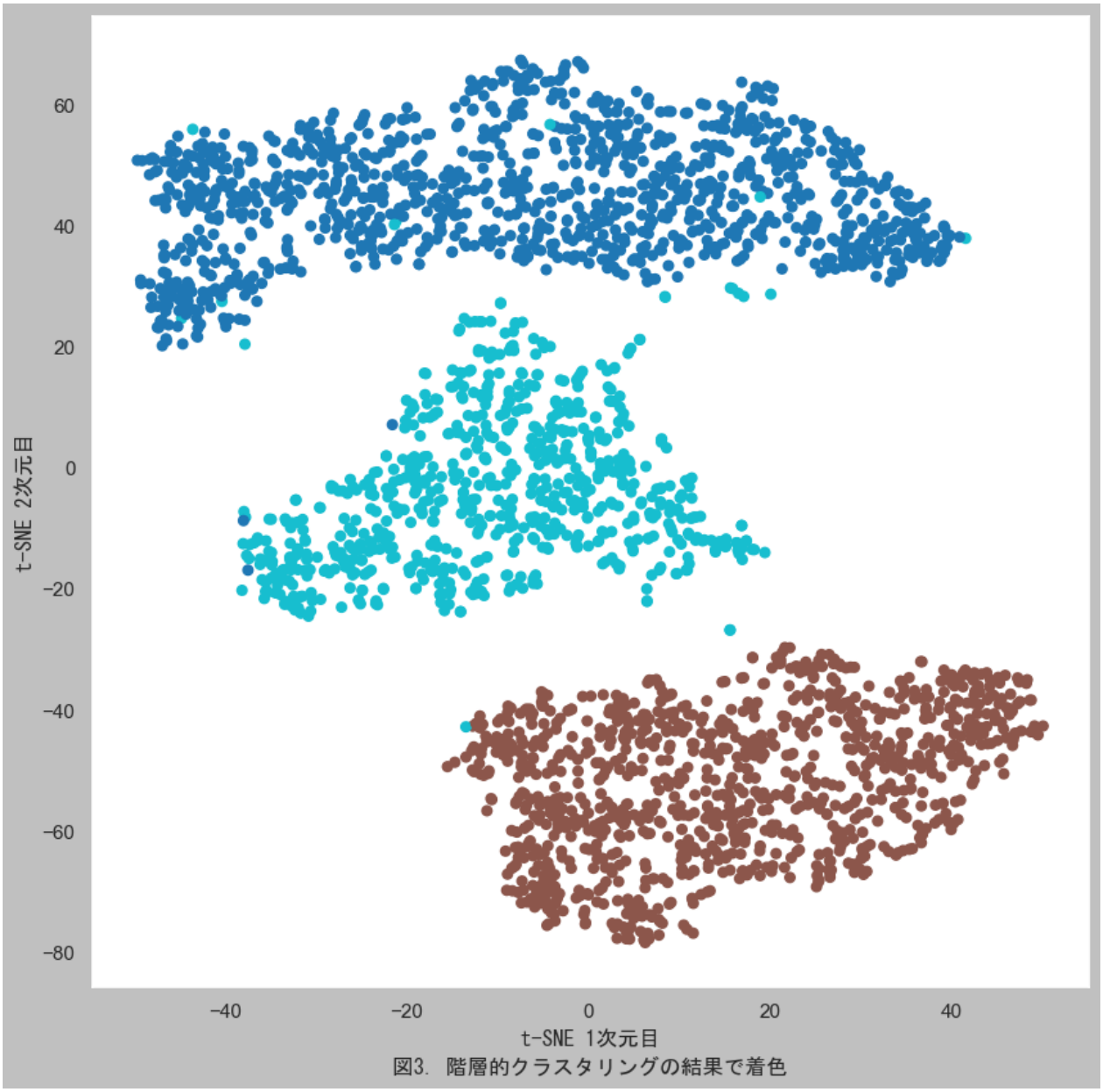

図3. 階層的クラスタリングの結果で着色

考察: 図2, 図3より、t-SNEの三つの塊りは、ward.D2法の結果に近いことが分かった