Fitbit Fitness Tracker Data を用いた健康向上のための分析

2024

目次

- 背景・目的
- 背景の整理・データ概要
- 分析
- 実装付録

目次

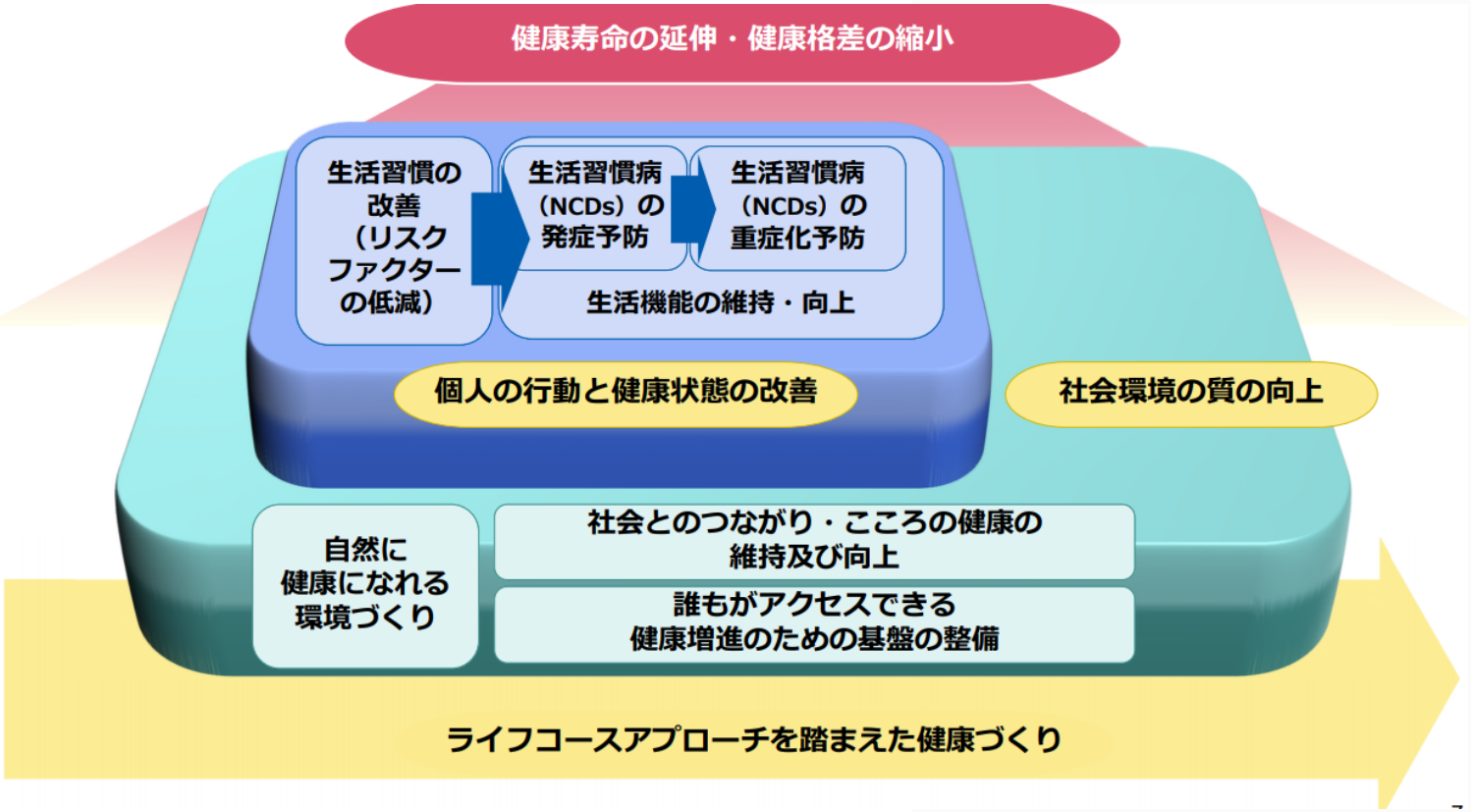
- 背景・目的
- 背景の整理・データ概要
- 分析
- 実装付録

日本における“健康”づくりの活動

*参照：第1回健康日本21(第三次)推進専門委員会 説明資料 資料2より抜粋

- 1978年(昭和53年)より第1次国民健康づくりの活動が開始される。
- 第3次国民健康づくりより、「健康日本21」と題し、健康寿命の延伸・健康格差の縮小などを目標とする活動が本格化された。

健康日本21(第三次)



- 個人の行動と健康状態の改善に、**生活習慣の改善**が目標としてあり、その項目として、**「栄養・食生活」、「身体活動・運動」、「休養・睡眠」**などの個々人の生活習慣改善が挙げられている。
- 健康日本21(第三次)より、新たな視点として、「自然に健康になれる環境づくり」や「個人の健康情報の見える化・利活用」が加わり、**ウェアラブル端末やアプリの利活用**が推進されている。

*参照：第1回健康日本21(第三次)推進専門委員会 説明資料 資料2より抜粋

身体活動・運動

（２）　身体活動・運動目標		
①　日常生活における歩数の増加	１日の歩数の平均値（年齢調整値）	7,100歩（令和14年度）
②　運動習慣者の増加	運動習慣者の割合（年齢調整値）	40%（令和14年度）
③　運動やスポーツを習慣的に行っていないこどもの減少	１週間の総運動時間（体育授業を除く。）が60分未満の児童の割合	第２次成育医療等基本方針に合わせて設定

*参照：第1回健康日本21(第三次)推進専門委員会 説明資料 資料2より抜粋

- 「身体活動・運動」の項目における目標として、
  - ①　日常生活における歩数の増加
  - ②　運動習慣者の増加
  - ③　運動やスポーツを習慣的に行っていない子供の減少　　が列挙される。

歩数・運動習慣の定着の有無・運動時間などが指標として注目されている

分析設計

- 健康向上には身体活動などの生活習慣改善が挙げられ、ウェアラブル端末の利活用が注目されている。

➢ **目的・テーマ**　ウェアラブル端末利用により健康向上につながるのか？

健康とは・・「肉体的、精神的及び社会的に完全に良好な状態」　*参照：世界保健機関（WHO）におけるWHO憲章より抜粋

➢ **仮説**

① 歩数増加及び座位時間減少による体重減少・BMI維持になる

② 睡眠時間増加による抑うつ等の心理的不安状態の軽減　など

➢ **前提・集計条件**　KaggleにおけるFitbit Fitness Tracker Dataを使用

目次

- 背景・目的
- **背景の整理・データ概要**
- 分析
- 実装付録

背景の整理

✓ 背景	Fitbitデータより健康向上に関わる要素を得る
✓ ゴール	Fitbit使用により歩数増加・座位時間減少・活動時間増加・睡眠時間増加等が明確になる、またその可視化を図る
✓ 決定事項	KaggleにあるFitbit Fitness Tracker Dataを使用

使用データ

✓ Fitbit Fitness Tracker Data

- Fitbitは、アメリカのフィットネスの企業が開発した、歩数・心拍数・睡眠の質・階段の昇降の歩数などのフィットネスに関係する指標を測定する、**ウェアラブル技術を用いたスマートウォッチ**である。
- Fitbit Fitness Tracker Dataは、kaggleに投稿された、Amazon Mechanical Turkを介した分散調査への回答者によって生成されたデータである。
- **2016年3月12日～5月12日の2か月間、30人のFitbitユーザー**による心拍数・消費カロリーなどのメトリクスをまとめた合計11のCSVファイルからなる。

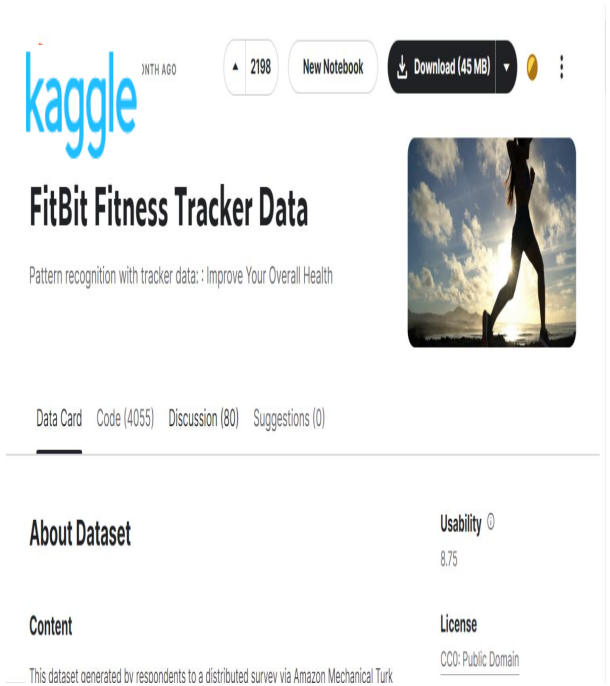
*参照：https://www.kaggle.com/datasets/arashnic/fitbit

使用環境・ツール

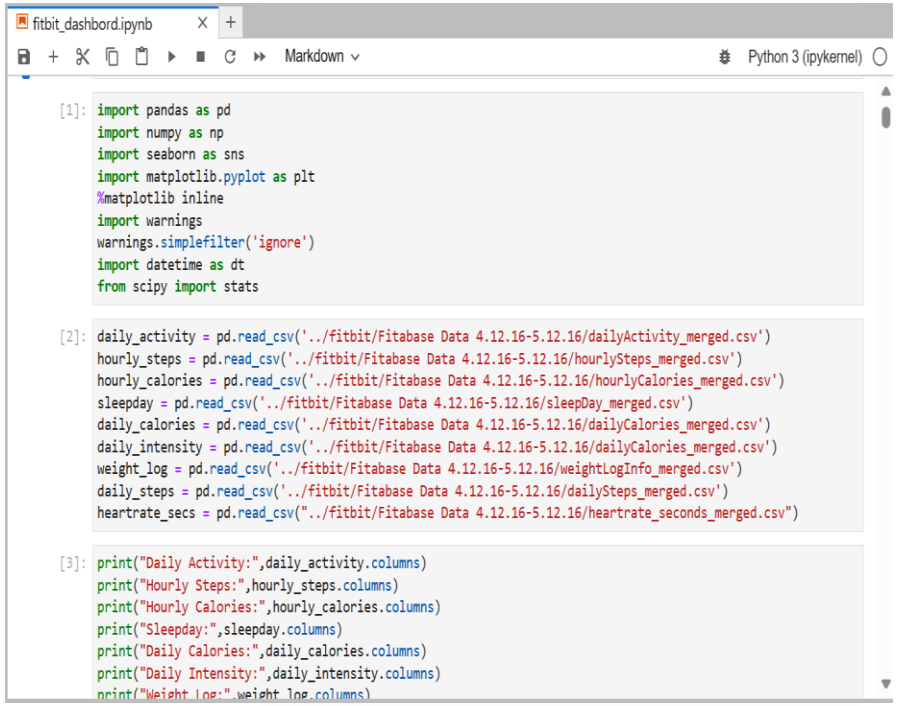
- ベースの環境はJupyter　Labを使用
- 言語はPythonを使用
- データの加工等はpandasを使用
- グラフの描画はseaborn、matplotlibなどを使用

データ分析の流れ

① Kaggleよりデータインポート



② データの前処理



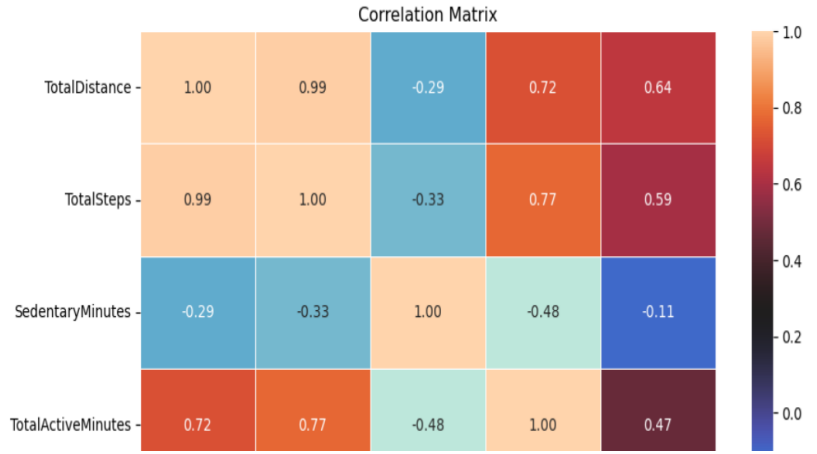
・欠損値、n数確認などデータ探索
・カラムなど要素が重複しているもの、明らかに欠損の多いファイルなどは統合、削除など行う
・同一の内容をまとめるといったデータクレンジングを行う
・分析を行いやすくするため、新規のカラムを追加など行う

③ データの可視化

```
Corr1 = ["TotalDistance", "TotalSteps",
         "SedentaryMinutes", "TotalActiveMinutes", "Calories"]

fig, ax = plt.subplots(figsize=(10, 6))
ax = sns.heatmap(daily_activity[Corr1].corr(),
            annot = True,
            fmt = ".2f",
            linewidths=0.5,
           cmap="icefire")

ax.set_title('Correlation Matrix');
```



・基本統計量、相関行列
・棒グラフ・散布図等

データ概観

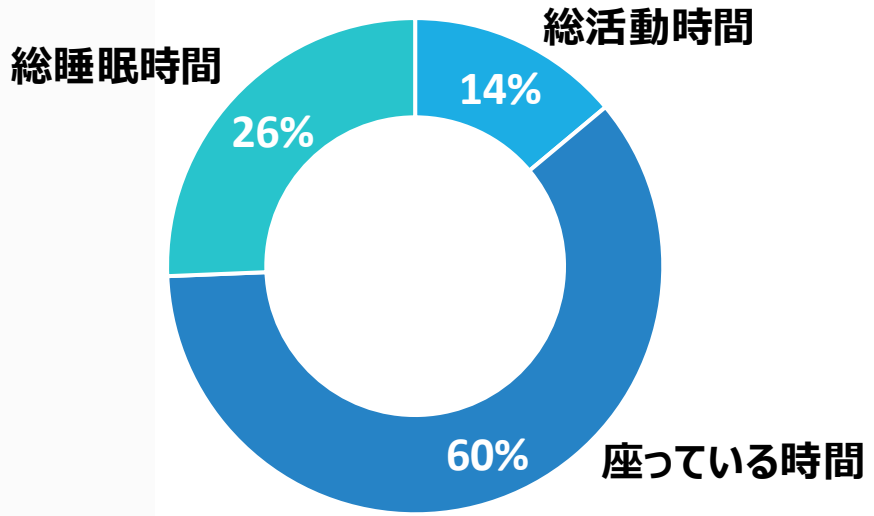
項目No.	項目名	説明	形式
1	Id	Fitbitシステム内の一意のID	整数
2	TotalSteps	歩数の合計(歩)	整数
3	TotalDistance	総歩行距離(m)	整数
4	VeryActiveDistance	非常に活動している歩行距離(m)	整数
5	ModeratelyActiveDistance	適度に活動している歩行距離(m)	整数
6	LightActiveDistance	軽度に活動している歩行距離(m)	整数
7	SedentaryActiveDistance	座っている時間の移動距離(m)	整数
8	VeryActiveMinutes	非常に活動している総時間(分)	整数
9	FairlyActiveMinutes	適度に活動している総時間(分)	整数
10	LightlyActiveMinutes	軽度に活動している総時間(分)	整数
11	SedentaryMinutes	座っている間の総時間(分)	整数
12	TotalActiveMinutes	活動している総時間(分)	整数
13	TotalMinutes	座位活動含めた活動総時間(分)	整数
14	TotalActiveHours	活動している総時間(分)	整数
15	Calories	活動中の推定総エネルギー消費量(cal)	整数
16	StepTotal	最大歩数(歩)	整数
17	DateHour	活動時間(分)	整数
18	TotalSleepRecords	総睡眠記録	整数
19	TotalMinutesAsleep	総睡眠時間(分)	整数
20	TotalTimeInBed	総ベッド臥床時間(分)	整数

- 最終的なデータセットのカラム一覧
- 大きくは歩数、活動時間、座っている時間といった非活動の時間、消費カロリーなどが挙げられた

データ概観

項目	平均値
歩数	7638歩
総歩行距離	5.5km
座っている時間	991分(約16.5時間)
総活動時間 (全ての動いている時間)	228分(約3.8時間)
消費カロリー	2304cal
総睡眠時間	419分(約7時間)

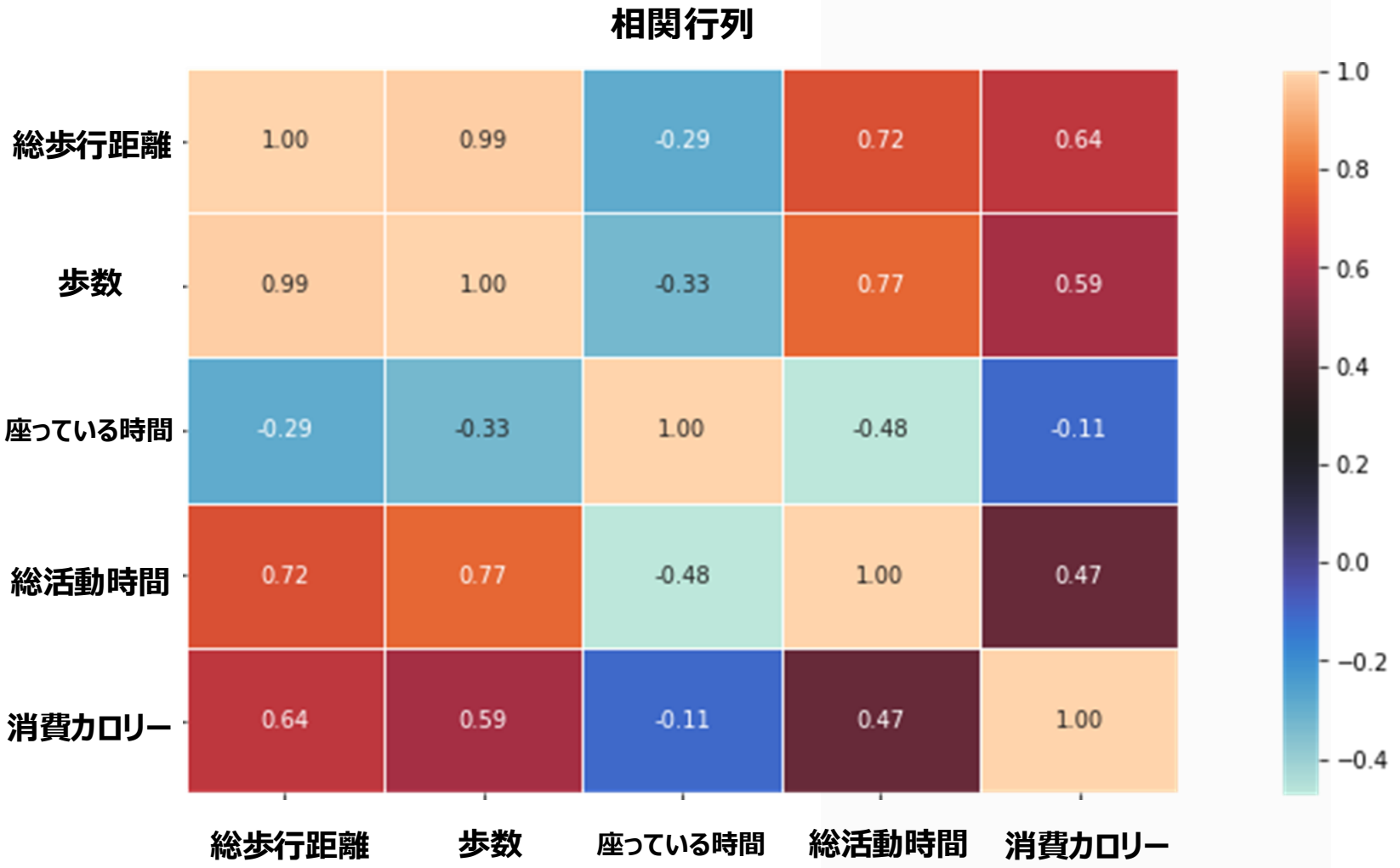
＊n=33、2016年3月12日～5月12日の2か月間
＊総睡眠時間のみn=24にて計算



目次

- 背景・目的
- 背景の整理・データ概要
- **分析**
- 実装付録

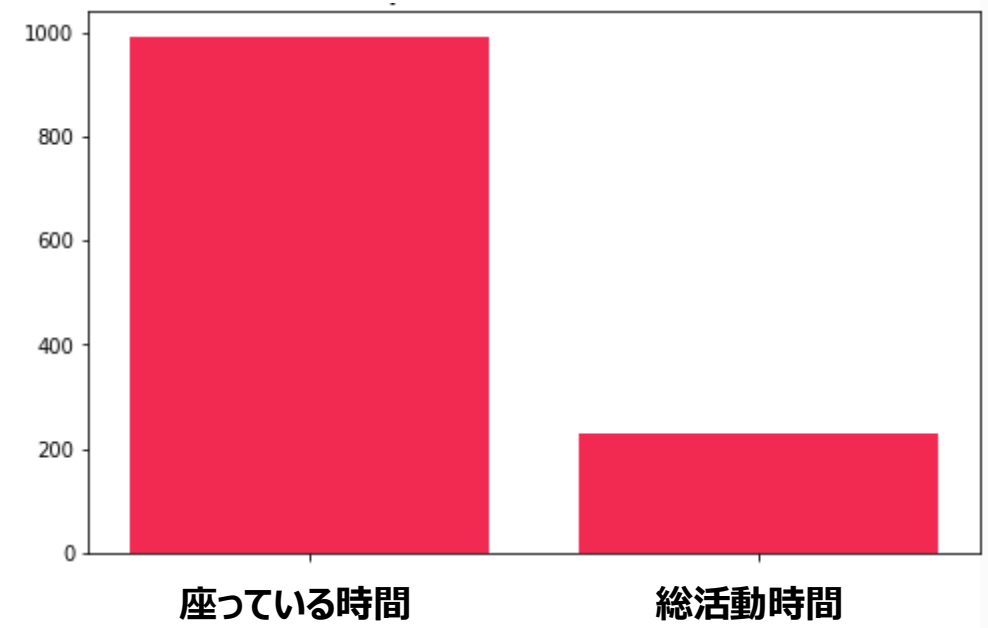
全体の分析



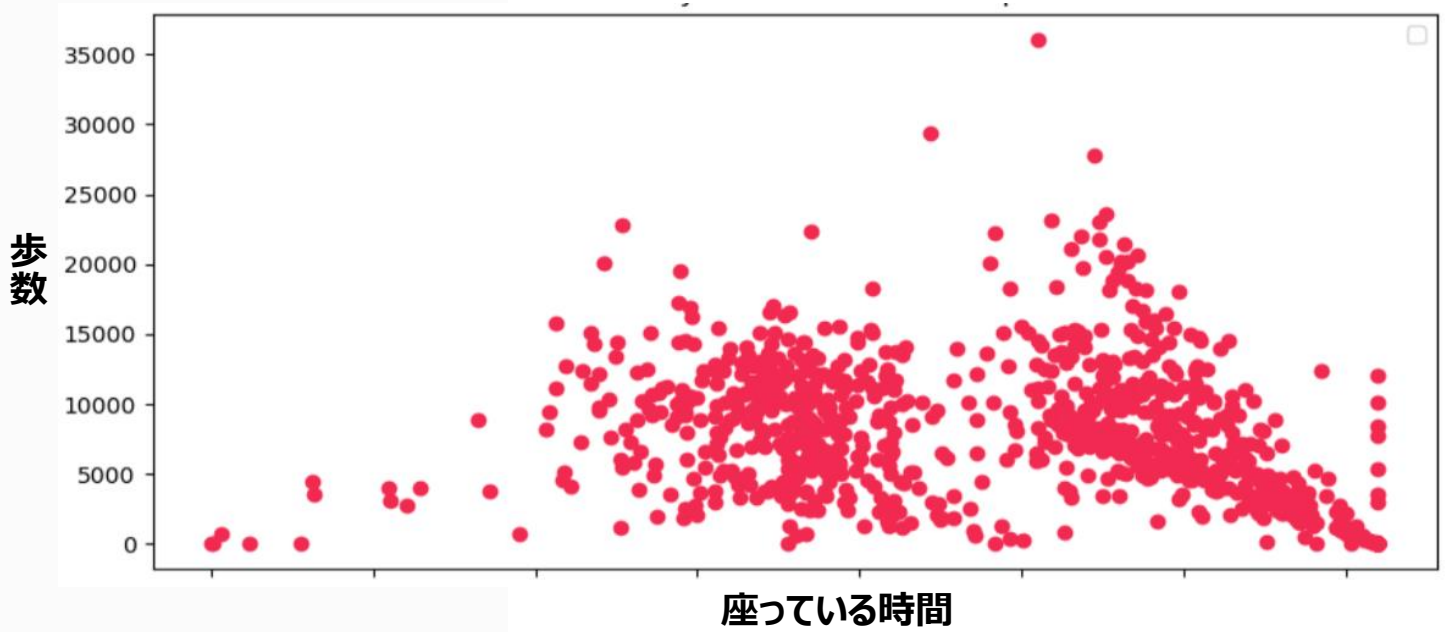
- 「体重」、「睡眠時間」といった項目は欠損あり今回は除外とした
- 「総歩行距離」、「歩数」、「総活動時間」に強い正の相関を認めた。
- また、それらは「消費カロリー」にも正の相関を認めた。
- 「座っている時間」は「総活動時間」、「歩数」に弱～中程度の負の相関を認めた。

全体の分析

座っている時間と総活動時間の比較

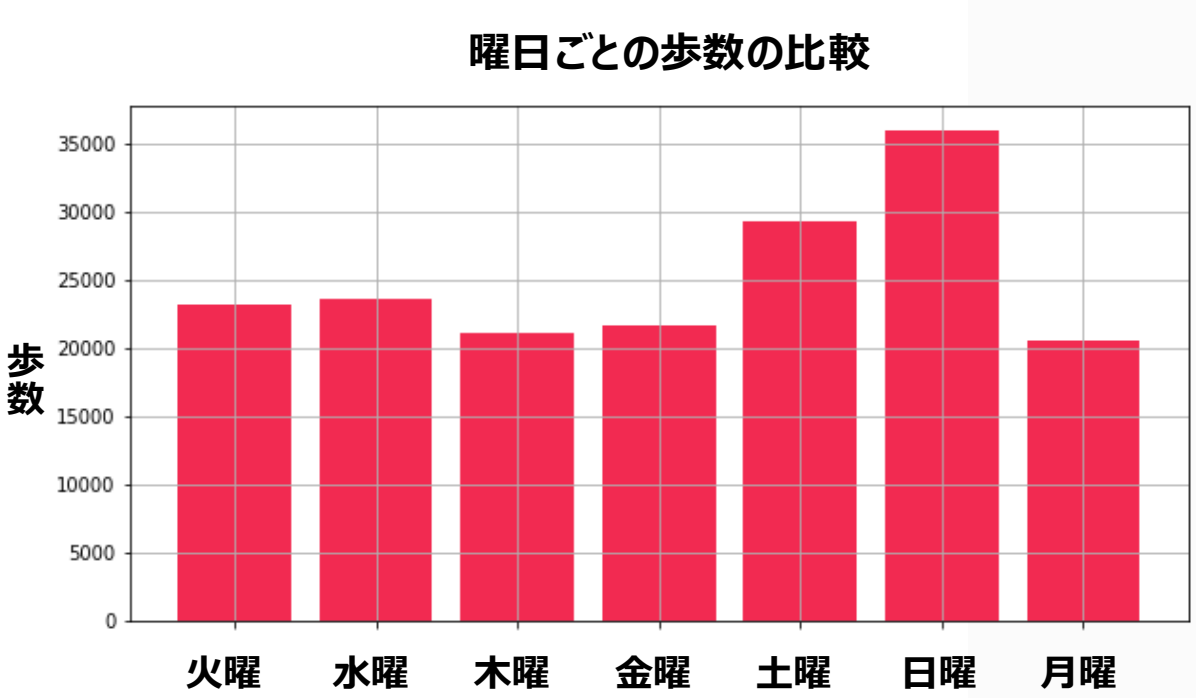


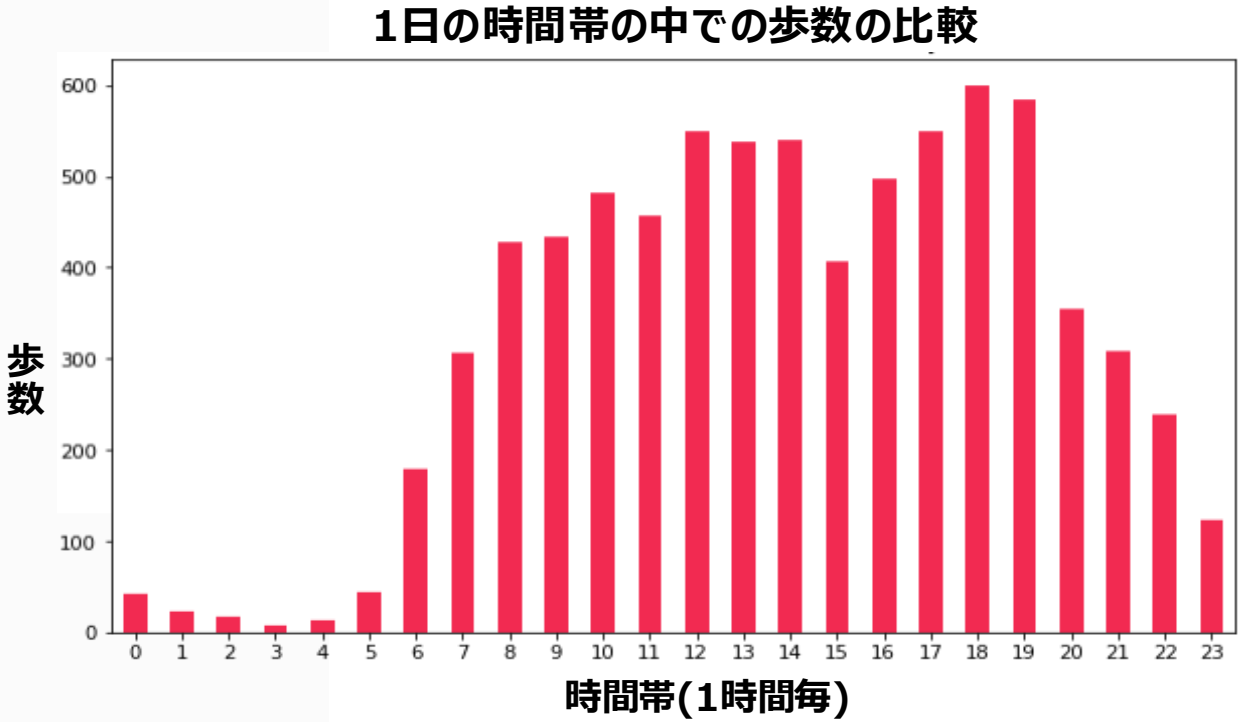
座っている時間と歩数の散布図



- 「座っている時間」と「総活動時間」では、明らかな差がみられた
- 「座っている時間」と「歩数」の散布図では、座っている時間が長い方が歩数も少ない傾向がややみられたが、座っている時間の差異にて2群化されている様子がみられた

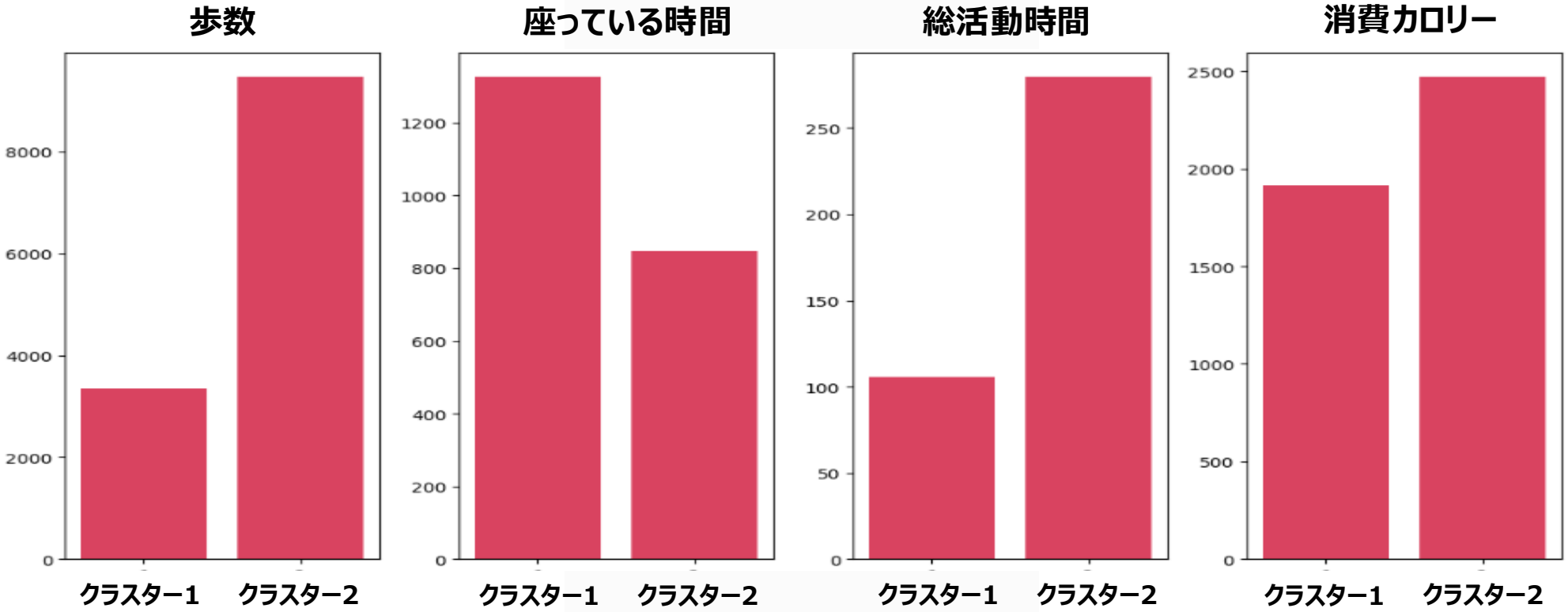
全体の分析





- 曜日別では土日の方が歩数多い傾向、1日の中では午後に、特に17時～19時が歩数多い傾向にあることがみられた

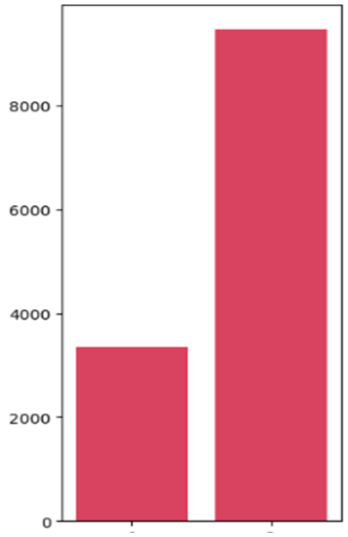
全体の分析



- 階層的クラスタリングにてクラスターを2つに分類
- クラスター間において活動量に伴った活動時間・歩数・消費カロリーの増加、座っている時間の減少が示唆された

2群間の分析

歩数



歩数の基本統計量

項目	歩数
平均値	7637.9
標準偏差	5087.1
最小値	0.0
中央値	7405.5
最大値	36019.0

中央値(7405.5)を基に
上か下かで2群に分類

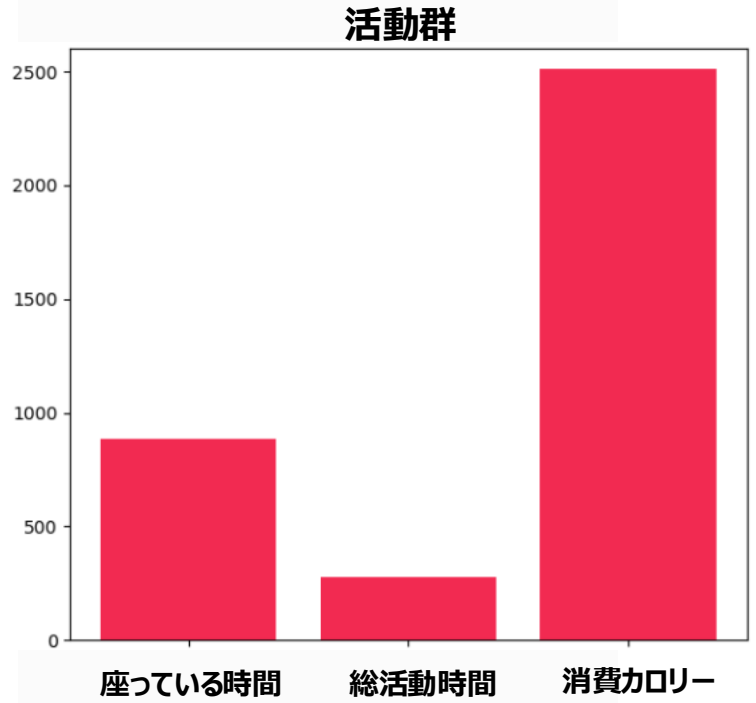
- 活動群(n=16)
- 低活動群(n=17)

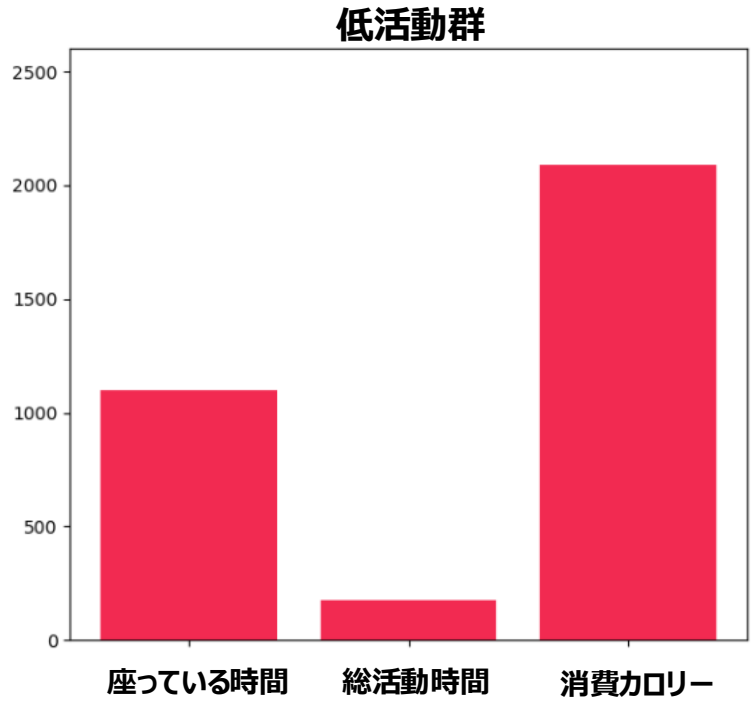
階層的クラスタリングの中で特に2群間で差があった歩数にて
活動量に応じた群間比較を行う

2群間の分析

2群間の平均値の比較

項目	平均値	
	活動群	低活動群
歩数(歩)	10361.9	4843.4
総活動時間(分)	275.9	177.8
座っている時間(分)	886.4	1098.6
消費カロリー(cal)	2513.8	2087.9



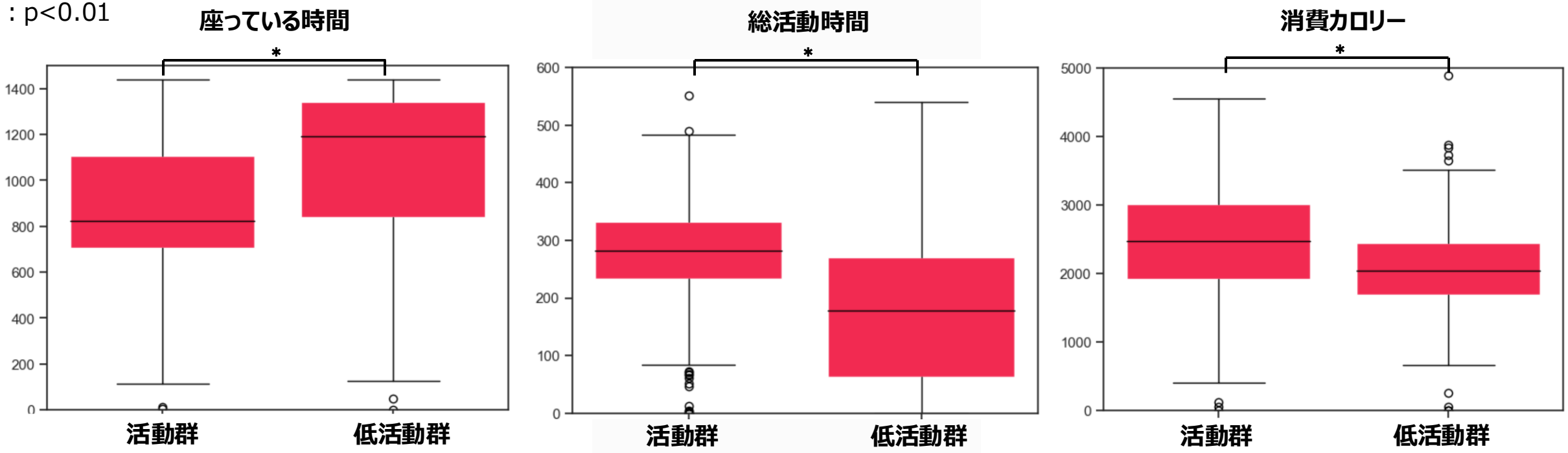


- 歩数を基に2群化したため、歩数には大きな開きがある
- 総活動時間・座っている時間・消費カロリーにおいても2群間で若干の差がみられ、低活動群では座っている時間の増加、総活動時間・消費カロリーの低下がみられる

2群間の分析

2群間の各項目での差の検定

＊：p<0.01



- マンホイットニーのU検定において、2群間の座っている時間・総活動時間・消費カロリーに有意差を認めた

まとめ

- 1日の時間配分としては、活動している時間よりも**座っている時間の方が長い**傾向にある。
- 睡眠時間や体重は欠損多く、今回の分析より除外
- 活動の指標として**歩数と活動時間には正の相関**がみられた
- 歩数が同程度の群でも、**座っている時間には差異**がみられた。
- 活動量に応じた比較では、**座っている時間・消費カロリー・活動時間に有意差**を認めた。

- 活動量の変化には、**活動時間より座っている時間を変化させた方が影響が強い**可能性がある。
- 対象とする群により、座っている時間の**活動内容自体も大きく違う**可能性がある。
- 活動量に応じた比較より、**Fitbit使用に伴って活動量が増加した**可能性がある。

まとめ

背景	Fitbitデータより健康向上に関わる要素を得る
ゴール	Fitbit使用により歩数増加・座位時間減少・活動時間増加・睡眠時間増加等が明確になる、またその可視化を図る
決定事項	KaggleにあるFitbit Fitness Tracker Dataを使用
結果	**①Fitbit使用により歩数などの項目変化が得られた ②歩数増加等の活動量増加に伴う、消費カロリーの増加といった項目毎の変化や、2群間の比較結果などを可視化した**
限界	**体重など欠落した情報が多い、BMI等の基本情報や別途心理学的評価なども含めたアンケートにて評価が行えていない**

提言

✓ For Fitbitユーザー

- 今回の計測では体重や睡眠時間の欠損が多く、これらの値をしっかり計測することで、より健康向上につながる要素を獲得できる可能性があると思われる。
- 相関関係にある項目が多く、まずは座位時間・歩数など自身の関心が強い項目のみを指標として追うことで、総じてその他項目も良くなっていくと思われる。

✓ For Fitbit導入施設など

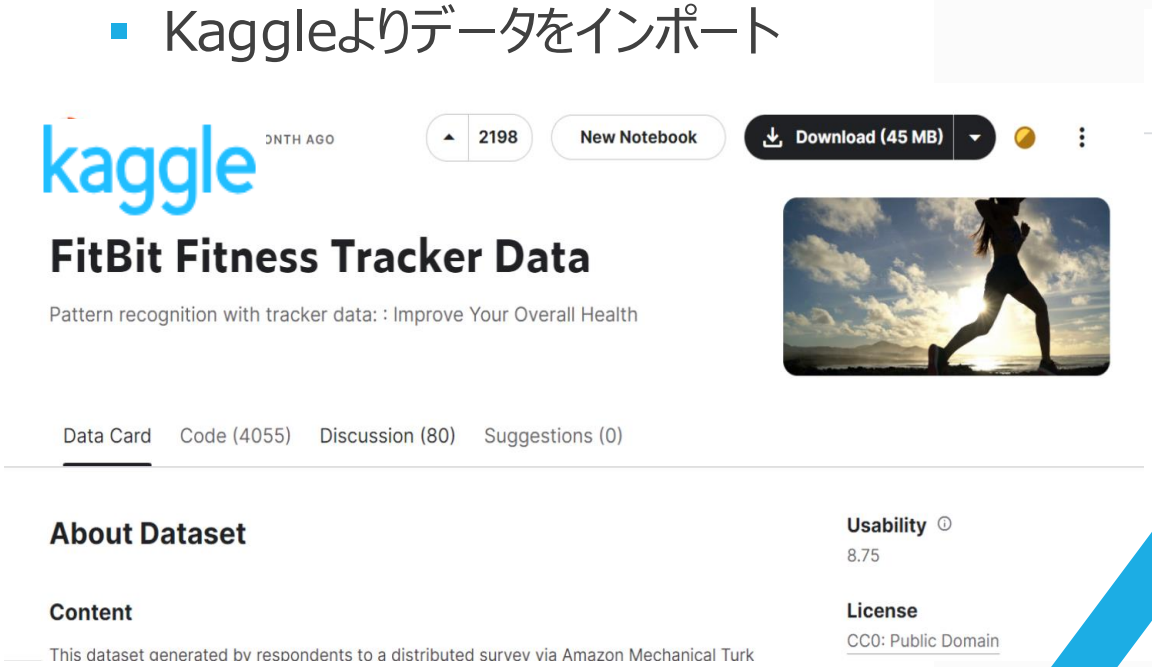
- 活動量としては、土日といった休日や、アフター5の時間（17時～19時）に多い傾向にあり、職務中の活動・運動量は低い傾向にあることが伺えた。社員・職員の健康向上の一つに、座位時間の減少は健康向上に繋がることが示唆されれており、立ちながら作業を行えるといった作業姿勢など働くスタイルの見直し・指標の一つに良いと思われる。

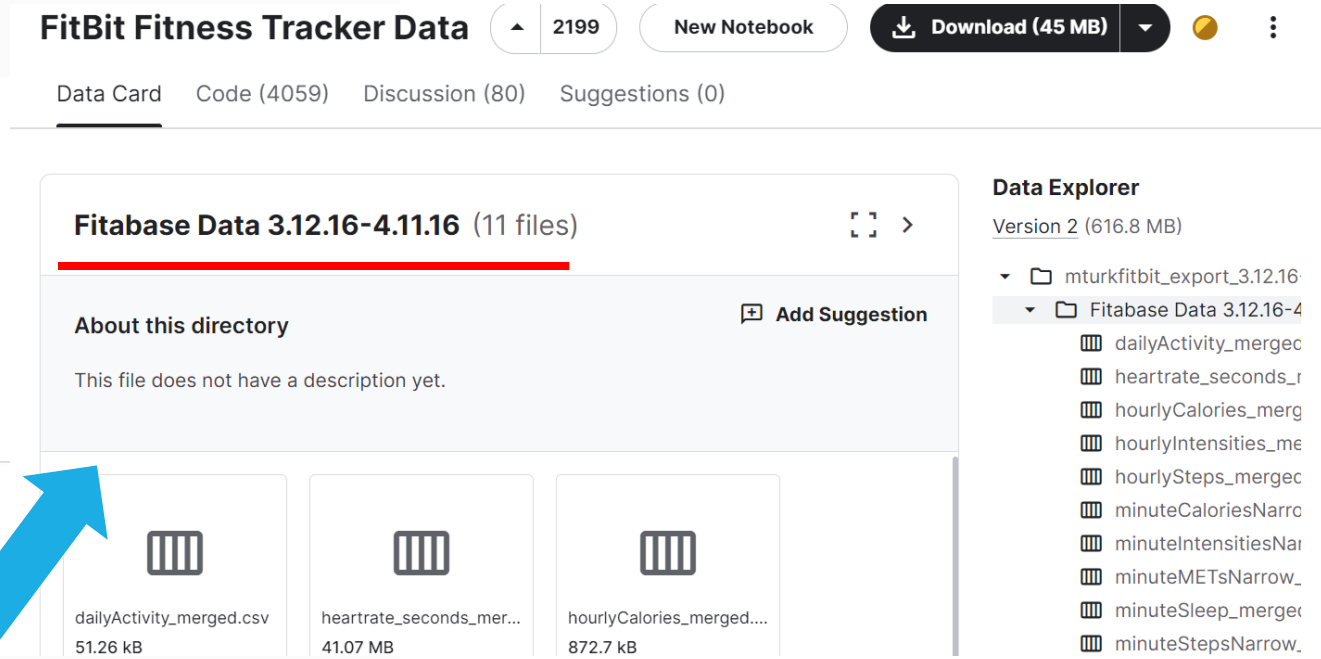
目次

- 背景・目的
- 背景の整理・データ概要
- 分析
- **実装付録**

実装の詳細を付録

- Kaggleよりデータをインポート





- 「Fitbit Fitness Tracker Data」は11のCSVファイルがあり、全てをインポート

各種ライブラリをインポート、11個のCSVファイルを読み込み

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.simplefilter('ignore')
import datetime as dt
from scipy import stats
```

```
daily_activity = pd.read_csv('../fitbit/Fitabase Data 4.12.16-5.12.16/dailyActivity_merged.csv')
hourly_steps = pd.read_csv('../fitbit/Fitabase Data 4.12.16-5.12.16/hourlySteps_merged.csv')
hourly_calories = pd.read_csv('../fitbit/Fitabase Data 4.12.16-5.12.16/hourlyCalories_merged.csv')
sleepday = pd.read_csv('../fitbit/Fitabase Data 4.12.16-5.12.16/sleepDay_merged.csv')
daily_calories = pd.read_csv('../fitbit/Fitabase Data 4.12.16-5.12.16/dailyCalories_merged.csv')
daily_intensity = pd.read_csv('../fitbit/Fitabase Data 4.12.16-5.12.16/dailyCalories_merged.csv')
weight_log = pd.read_csv('../fitbit/Fitabase Data 4.12.16-5.12.16/weightLogInfo_merged.csv')
daily_steps = pd.read_csv('../fitbit/Fitabase Data 4.12.16-5.12.16/dailySteps_merged.csv')
heartrate_secs = pd.read_csv("../fitbit/Fitabase Data 4.12.16-5.12.16/heartrate_seconds_merged.csv")
```

各ファイル毎のID（一意の値）を取得

```
print("Daily Activity Dataset:",daily_activity.Id.nunique(), "unique users")
print("Hourly Steps Dataset:",hourly_steps.Id.nunique(), "unique users")
print("Hourly Calories Dataset:",hourly_calories.Id.nunique(), "unique users")
print("SleepDay Dataset:",sleepday.Id.nunique(), "unique users")
print("Daily Calories Dataset:",daily_calories.Id.nunique(), "unique users")
print("Daily Intensity Dataset:",daily_intensity.Id.nunique(), "unique users")
print("Weight Logged Dataset:",weight_log.Id.nunique(), "unique users")
print("Daily Steps Dataset:",daily_steps.Id.nunique(), "unique users")
print("Heart Rate Dataset:",heartrate_secs.Id.nunique(), "unique users")
```

```
Daily Activity Dataset: 33 unique users
Hourly Steps Dataset: 33 unique users
Hourly Calories Dataset: 33 unique users
SleepDay Dataset: 24 unique users
Daily Calories Dataset: 33 unique users
Daily Intensity Dataset: 33 unique users
Weight Logged Dataset: 8 unique users
Daily Steps Dataset: 33 unique users
Heart Rate Dataset: 14 unique users
```

各要素(columns)を取得

```
print("Daily Activity:",daily_activity.columns)
print("Hourly Steps:",hourly_steps.columns)
print("Hourly Calories:",hourly_calories.columns)
print("Sleepday:",sleepday.columns)
print("Daily Calories:",daily_calories.columns)
print("Daily Intensity:",daily_intensity.columns)
print("Weight Log:",weight_log.columns)
print("Daily Steps:",daily_steps.columns)
print("Heart Rate:",heartrate_secs.columns)
```

```
Daily Activity: Index(['Id', 'ActivityDate', 'TotalSteps', 'TotalDistance', 'TrackerDistance',
       'LoggedActivitiesDistance', 'VeryActiveDistance',
       'ModeratelyActiveDistance', 'LightActiveDistance',
       'SedentaryActiveDistance', 'VeryActiveMinutes', 'FairlyActiveMinutes',
       'LightlyActiveMinutes', 'SedentaryMinutes', 'Calories'],
      dtype='object')
Hourly Steps: Index(['Id', 'ActivityHour', 'StepTotal'], dtype='object')
Hourly Calories: Index(['Id', 'ActivityHour', 'Calories'], dtype='object')
Sleepday: Index(['Id', 'SleepDay', 'TotalSleepRecords', 'TotalMinutesAsleep',
       'TotalTimeInBed'],
      dtype='object')
Daily Calories: Index(['Id', 'ActivityDay', 'Calories'], dtype='object')
Daily Intensity: Index(['Id', 'ActivityDay', 'Calories'], dtype='object')
Weight Log: Index(['Id', 'Date', 'WeightKg', 'WeightPounds', 'Fat', 'BMI',
       'IsManualReport', 'LogId'],
      dtype='object')
Daily Steps: Index(['Id', 'ActivityDay', 'StepTotal'], dtype='object')
Heart Rate: Index(['Id', 'Time', 'Value'], dtype='object')
```

各ファイル毎の文字型・行列要素の取得、欠損・重複の確認

```
daily_activity.dtypes
```

```
Id                          int64
ActivityDate               object
TotalSteps                  int64
TotalDistance             float64
TrackerDistance           float64
LoggedActivitiesDistance  float64
VeryActiveDistance        float64
ModeratelyActiveDistance  float64
LightActiveDistance       float64
SedentaryActiveDistance   float64
VeryActiveMinutes           int64
FairlyActiveMinutes         int64
LightlyActiveMinutes        int64
SedentaryMinutes            int64
Calories                    int64
dtype: object
```

```
print("Shape of Daily Activity:",daily_activity.shape)
```

```
Shape of Daily Activity: (940, 15)
```

```
print("Daily Activity Dataset has",daily_activity.isnull().values.sum(), "missing values")
```

```
Daily Activity Dataset has 0 missing values
```

```
print("Duplicates in Daily Acitivity: ", daily_activity.duplicated().head())
```

```
Duplicates in Daily Acitivity:  0    False
1    False
2    False
3    False
4    False
dtype: bool
```

- datetime型に変換、「hourly_steps」・「hourly_calories」のデータフレームを結合

```
daily_activity["ActivityDate"] = pd.to_datetime(daily_activity["ActivityDate"])
hourly_steps["ActivityHour"] = pd.to_datetime(hourly_calories["ActivityHour"])
hourly_calories["ActivityHour"] = pd.to_datetime(hourly_calories["ActivityHour"])
sleepday["SleepDay"] = pd.to_datetime(sleepday["SleepDay"])

print("Daily Acitivity data type is", daily_activity["ActivityDate"].dtypes, "data type")
print("Hourly Steps data type is", hourly_steps["ActivityHour"].dtypes, "data type")
print("Hourly Calories data type is", hourly_calories["ActivityHour"].dtypes, "data type")
print("Sleepday data type is", sleepday["SleepDay"].dtypes, "data type")
```

```
Daily Acitivity data type is datetime64[ns] data type
Hourly Steps data type is datetime64[ns] data type
Hourly Calories data type is datetime64[ns] data type
Sleepday data type is datetime64[ns] data type
```

```
for col in hourly_steps.columns, hourly_calories.columns:
    print (col)
merge_df = pd.merge(hourly_steps, hourly_calories,
                    on=['Id', 'ActivityHour'], how='inner')
merge_df.head()
```

```
Index(['Id', 'ActivityHour', 'StepTotal'], dtype='object')
Index(['Id', 'ActivityHour', 'Calories'], dtype='object')
```

	Id	ActivityHour	StepTotal	Calories
0	1503960366	2016-04-12 00:00:00	373	81
1	1503960366	2016-04-12 01:00:00	160	61
2	1503960366	2016-04-12 02:00:00	151	59

- 新たなデータフレームとして消費カロリーの項目を追加するべく、文字型の変換・列名の変更、データフレームの結合を行う

```
merge_df["ActivityDay"] = pd.to_datetime(merge_df["ActivityDay"])
sleepday = sleepday.rename(columns={'SleepDay': 'ActivityDay'})
sleepday.head()
```

	Id	ActivityDay	TotalSleepRecords	TotalMinutesAsleep	TotalTimeInBed	WeekDay
0	1503960366	2016-04-12	1	327	346	Tuesday
1	1503960366	2016-04-13	2	384	407	Wednesday
2	1503960366	2016-04-15	1	412	442	Friday
3	1503960366	2016-04-16	2	340	367	Saturday
4	1503960366	2016-04-17	1	700	712	Sunday

```
for col in merge_df.columns, sleepday.columns:
    print (col)
```

```
Index(['Id', 'StepTotal', 'Calories', 'WeekDay', 'DateHour', 'ActivityDay'], dtype='object')
Index(['Id', 'ActivityDay', 'TotalSleepRecords', 'TotalMinutesAsleep',
       'TotalTimeInBed', 'WeekDay'],
      dtype='object')
```

```
merge_df2 = pd.merge(merge_df, sleepday,
                     on=['Id', 'ActivityDay', 'WeekDay'], how='inner')
merge_df2.head(5)
```

	Id	StepTotal	Calories	WeekDay	DateHour	ActivityDay	TotalSleepRecords	TotalMinutesAsleep	TotalTimeInBed
0	1503960366	373	81	Tuesday	0	2016-04-12	1	327	346
1	1503960366	160	61	Tuesday	1	2016-04-12	1	327	346
2	1503960366	151	59	Tuesday	2	2016-04-12	1	327	346

- 「WeekDay」という新しい列を作成

```
daily_activity["WeekDay"] = daily_activity["ActivityDate"].dt.day_name()
merge_df["WeekDay"] = merge_df["ActivityHour"].dt.day_name()
sleepday["WeekDay"] = sleepday["SleepDay"].dt.day_name()

merge_df["DateHour"] = merge_df["ActivityHour"].dt.hour
merge_df["ActivityDay"] = merge_df["ActivityHour"].dt.date
merge_df = merge_df.drop("ActivityHour", axis = 1)
merge_df.head()
```

	Id	StepTotal	Calories	WeekDay	DateHour	ActivityDay
0	1503960366	373	81	Tuesday	0	2016-04-12
1	1503960366	160	61	Tuesday	1	2016-04-12
2	1503960366	151	59	Tuesday	2	2016-04-12
3	1503960366	0	47	Tuesday	3	2016-04-12
4	1503960366	0	48	Tuesday	4	2016-04-12

- 既存のデータフレームに新規の列を作成し追加

```
new_cols = ["Id", "ActivityDate",
            "WeekDay", "TotalSteps",
            "TotalDistance", "VeryActiveDistance",
            "ModeratelyActiveDistance", "LightActiveDistance",
            "SedentaryActiveDistance", "VeryActiveMinutes",
            "FairlyActiveMinutes", "LightlyActiveMinutes",
            "SedentaryMinutes", "TotalActiveMinutes",
            "TotalMinutes", "TotalActiveHours", "Calories"]

daily_activity = daily_activity.reindex(columns=new_cols)

daily_activity[["TotalActiveMinutes",
                "TotalMinutes", "TotalActiveHours"]].head()
```

	TotalActiveMinutes	TotalMinutes	TotalActiveHours
0	NaN	NaN	NaN
1	NaN	NaN	NaN

新規の列に値を入力

```
daily_activity["TotalActiveMinutes"] = daily_activity["VeryActiveMinutes"] + daily_activity["FairlyActiveMinutes"] \
                                     + daily_activity["LightlyActiveMinutes"]
daily_activity["TotalMinutes"] = daily_activity["TotalActiveMinutes"] + daily_activity["SedentaryMinutes"]
daily_activity["TotalActiveHours"] = round(daily_activity["TotalActiveMinutes"] / 60)
daily_activity[["TotalActiveMinutes", "TotalMinutes", "TotalActiveHours"]].head()
```

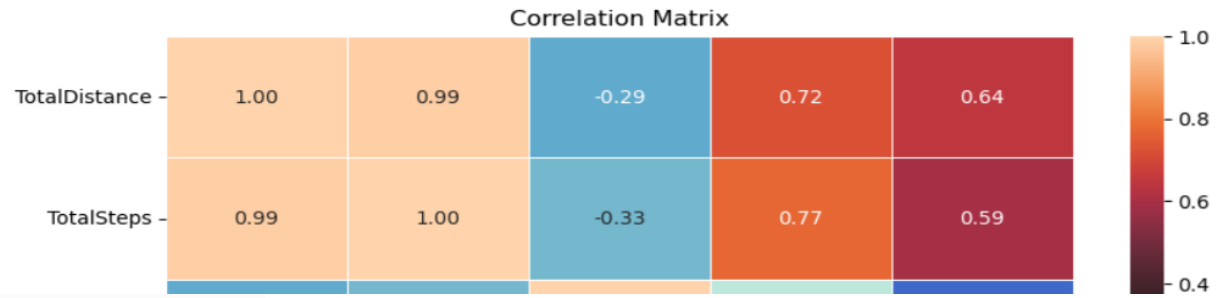
	TotalActiveMinutes	TotalMinutes	TotalActiveHours
0	366	1094	6.0
1	257	1033	4.0
2	222	1440	4.0

主要と思われる項目の相関行列を作成

```
Corr1 = ["TotalDistance", "TotalSteps",
         "SedentaryMinutes", "TotalActiveMinutes", "Calories"]

fig, ax = plt.subplots(figsize=(10, 6))
ax = sns.heatmap(daily_activity[Corr1].corr(),
            annot = True,
            fmt = ".2f",
            linewidths=0.5,
           cmap="icefire")

ax.set_title('Correlation Matrix');
```



作成したデータフレームの基本統計量を確認

```
daily_activity.describe()
```

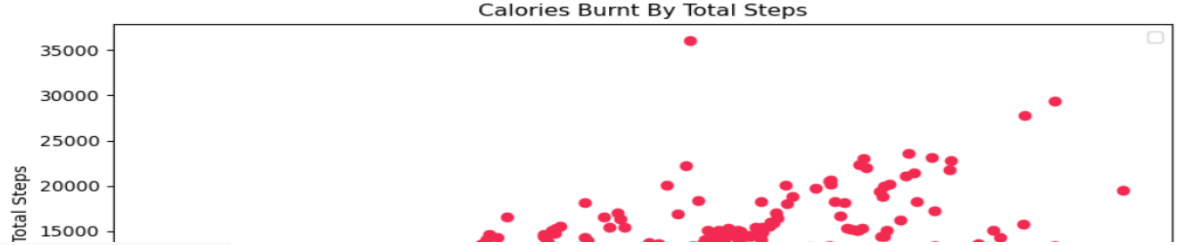
	Id	TotalSteps	TotalDistance	VeryActiveDistance	ModeratelyActiveDistance	LightActiveDistance	Sed
count	9.400000e+02	940.000000	940.000000	940.000000	940.000000	940.000000	
mean	4.855407e+09	7637.910638	5.489702	1.502681	0.567543	3.340819	
std	2.424805e+09	5087.150742	3.924606	2.658941	0.883580	2.040655	
min	1.503960e+09	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	2.320127e+09	3789.750000	2.620000	0.000000	0.000000	1.945000	
50%	4.445115e+09	7405.500000	5.245000	0.210000	0.240000	3.365000	
75%	6.962181e+09	10727.000000	7.712500	2.052500	0.800000	4.782500	
max	8.877689e+09	36019.000000	28.030001	21.920000	6.480000	10.710000	

```
merge_df.describe()
```

	Id	StepTotal	Calories	DateHour
count	2.209900e+04	22099.000000	22099.000000	22099.000000
mean	4.848235e+09	320.166342	97.386760	11.415765
std	2.422500e+09	690.384228	60.702622	6.915140
min	1.503960e+09	0.000000	42.000000	0.000000

「歩数と消費カロリー」など散布図を作成

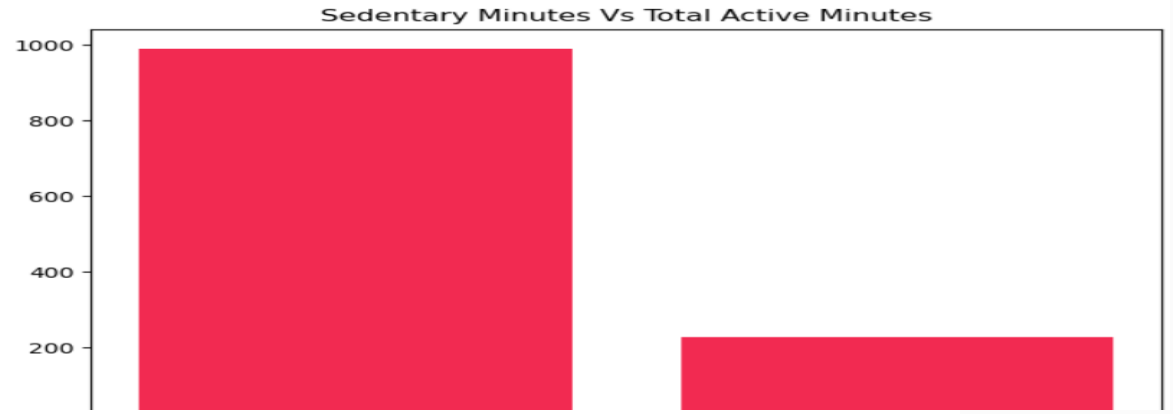
```
fig, (ax0, ax1) = plt.subplots(nrows=2,
                               ncols=1,
                               figsize=(10, 10),
                                 sharex=True)
scatter = ax0.scatter(x=daily_activity["Calories"],
                      y=daily_activity["TotalSteps"],
                      color="#f22a51")
ax0.set(title="Calories Burnt By Total Steps",
        ylabel="Total Steps")
ax0.legend(*scatter.legend_elements(), title=None)
scatter = ax1.scatter(x=daily_activity["Calories"],
                      y=daily_activity["SedentaryMinutes"],
                      color="#f22a51")
ax1.set(title="Sedentary Minutes and Calories",
        ylabel="Sedentary Minutes")
ax1.legend(*scatter.legend_elements(), title=None);
```



「座っている時間と活動時間」など棒グラフを作成

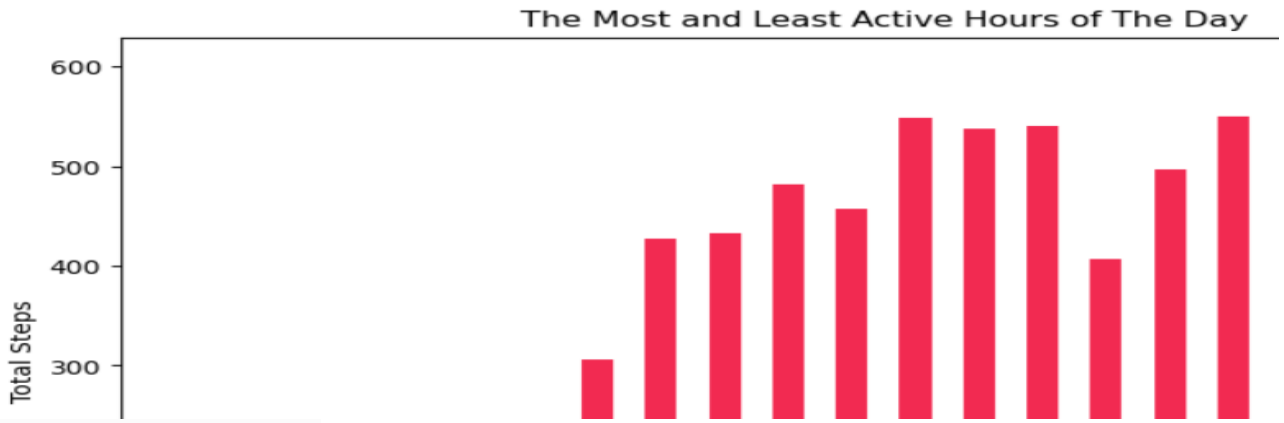
```
fig, ax = plt.subplots(figsize=(8,5))
x = np.array(["Sedentary Minutes", "TotalActiveMinutes"])
y = np.array([daily_activity["SedentaryMinutes"].mean(),
              daily_activity["TotalActiveMinutes"].mean()])

plt.title("Sedentary Minutes Vs Total Active Minutes")
plt.bar(x, y, width = 0.8, color = "#f22a51")
plt.show()
```



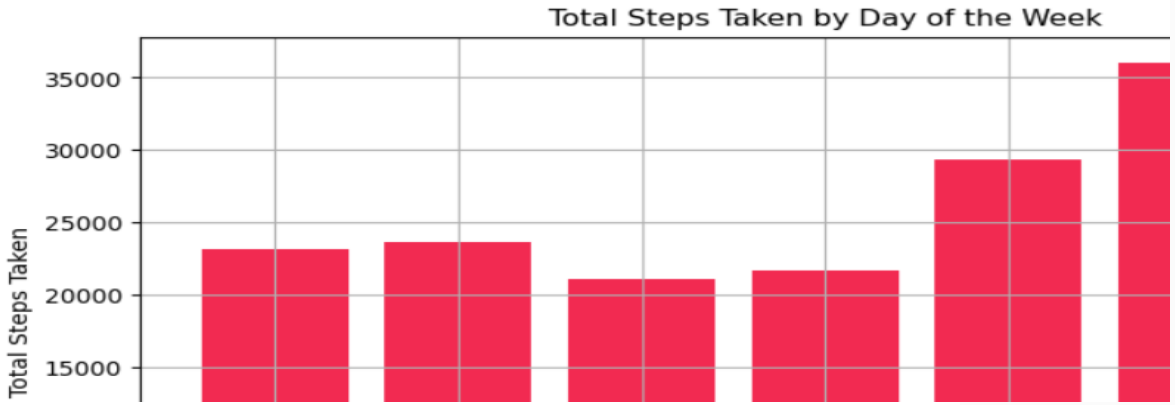
「1日の時間帯ごとの歩数平均」の棒グラフ作成

```
fig, axs = plt.subplots(figsize=(10, 6))
merge_df.groupby(merge_df['DateHour'])['StepTotal'].mean().plot(kind='bar', rot=0, ax=axs,
plt.xlabel("Hour of the Day")
plt.ylabel("Total Steps")
plt.show()
```



「曜日ごとの歩数」の棒グラフ作成

```
fig, ax = plt.subplots(figsize=(10,5))
plt.bar(daily_activity.WeekDay, daily_activity.TotalSteps,
        color = "#f22a51")
plt.xlabel("Week Day")
plt.ylabel("Total Steps Taken")
plt.title("Total Steps Taken by Day of the Week")
plt.grid()
plt.show()
```



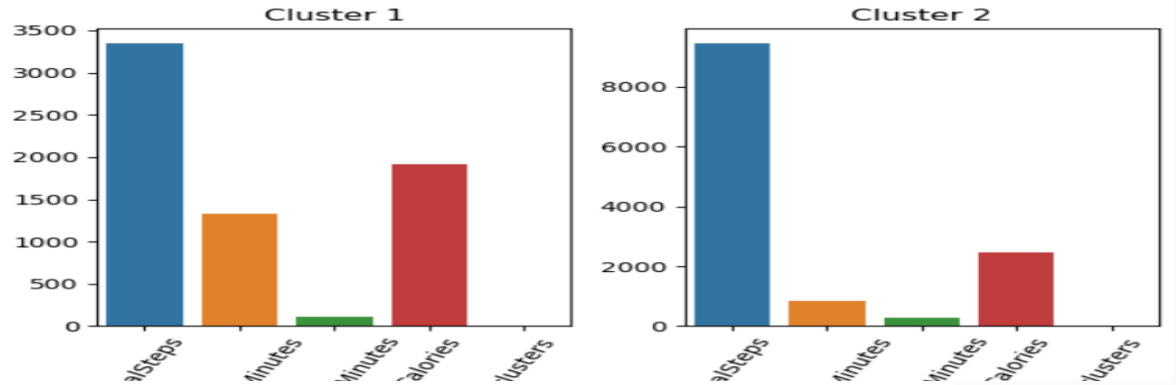
階層的クラスタリング手法にてクラスターを2つに分類、散布図を作成

```
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
dendro_df = daily_activity.copy()
dendro_df = dendro_df.drop(["Id","ActivityDate","WeekDay","TotalDistance","VeryActiveDistance",
                            "ModeratelyActiveDistance","LightActiveDistance","SedentaryActiveDistance",
                            "VeryActiveMinutes","FairlyActiveMinutes","LightlyActiveMinutes",
                            "TotalMinutes","TotalActiveHours"],axis=1)
scaled = StandardScaler().fit_transform(dendro_df)
Z = linkage(scaled, method='ward')
clusters = fcluster(Z, criterion="maxclust", t=2)
d_result_df = pd.concat([dendro_df, pd.DataFrame({"clusters":clusters})], axis=1)
```

```
plt.scatter(d_result_df[d_result_df["clusters"]==1]["SedentaryMinutes"], d_result_df[d_result_df["clusters"]==1]["TotalSteps"],
            marker='o', s=25, c='r', alpha=0.5, label="1")
plt.scatter(d_result_df[d_result_df["clusters"]==2]["SedentaryMinutes"], d_result_df[d_result_df["clusters"]==2]["TotalSteps"],
            marker='o', s=25, c='b', alpha=0.5, label="2")
plt.xlabel("SedentaryMinutes")
plt.ylabel("TotalSteps")
plt.legend(loc="upper right")
plt.title('TotalSteps VS SedentaryMinutes')
plt.show()
```

２つのクラスターで各項目の棒グラフ作成し比較

```
mean = groups.mean()
for i in range(1,3):
    plt.subplot(1, 2, i)
    plt.title(f"Cluster {i}")
    sns.barplot(x=mean.columns, y=mean.loc[i])
    plt.xticks(rotation=60)
    plt.xlabel(None)
    plt.ylabel(None)
    plt.tight_layout()
```

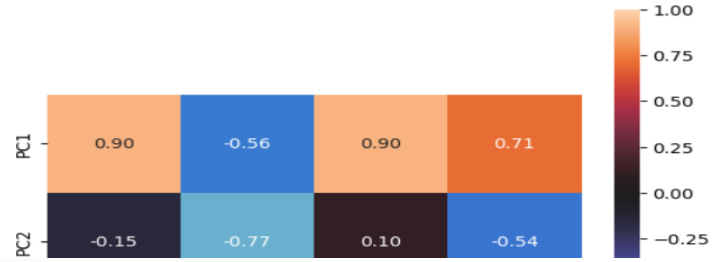


主成分分析にて寄与率を確認

```
factor_loading = pca.components_ * np.c_[np.sqrt(pca.explained_variance_)]
df_factor_loading = pd.DataFrame(factor_loading, index=[f"PC{i+1}"for i in range(len(factor_loading))], columns=kmeans_df.columns)
df_factor_loading.head()
```

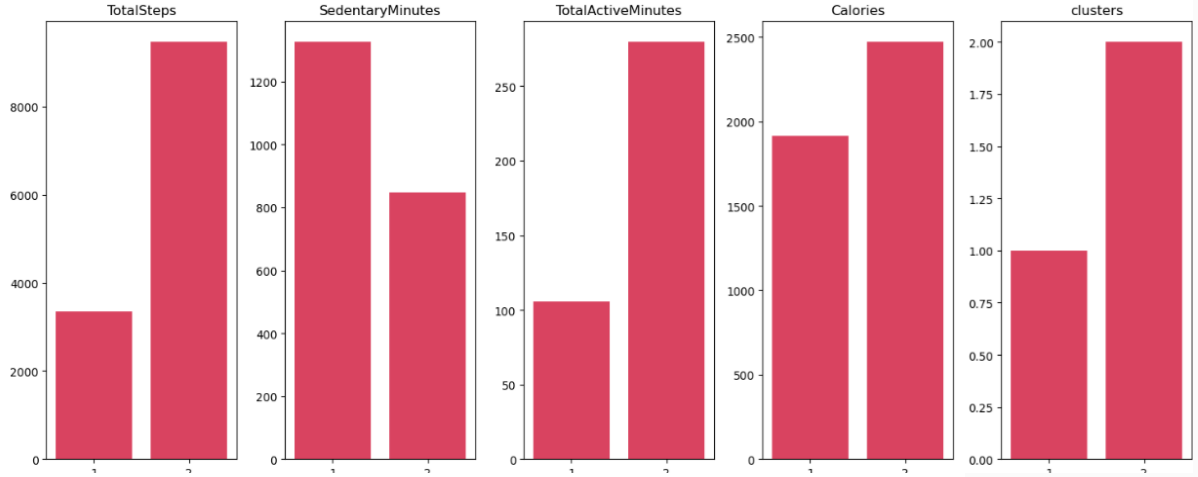
	TotalSteps	SedentaryMinutes	TotalActiveMinutes	Calories
PC1	0.902849	-0.556447	0.901702	0.708892
PC2	-0.152820	-0.771540	0.104228	-0.543567
PC3	0.249040	0.304183	0.289235	-0.446313
PC4	0.317103	-0.060250	-0.305741	-0.062258

```
sns.heatmap(df_factor_loading.head(2), vmin=-1, vmax=1, center=0, square=True, annot=True, fmt='.2F')
plt.show()
```



各項目ごとの２つのクラスターを棒グラフ作成し比較

```
plt.figure(figsize=(15,6))
for i,col in enumerate(mean.columns):
    plt.subplot(1,5,i+1)
    plt.title(f"{col}")
    sns.barplot(x=mean.index, y=mean[col], color = "#f22a51")
    plt.ylabel(None)
    plt.tight_layout()
```



中央値より大小を真偽としたbool値を作成しデータフレームに追加、その値をもとに2群に群分けを行う

```
df_totalsteps=pd.DataFrame(daily_activity.groupby("Id")["TotalSteps"].mean() > 7405.5)
df_totalsteps=df_totalsteps.rename(columns={'TotalSteps':'AverageSteps'})
df_totalsteps = pd.merge(daily_activity,df_totalsteps,on="Id",how="outer")
df_totalsteps.head()
```

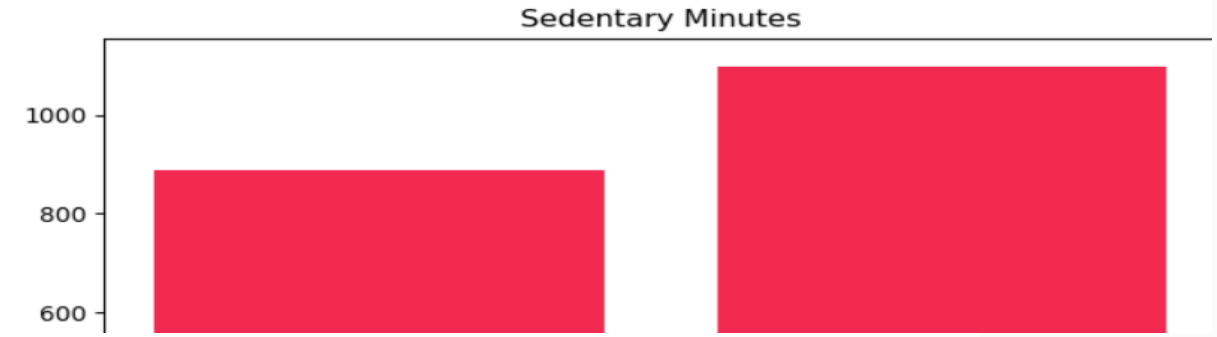
	Id	ActivityDate	WeekDay	TotalSteps	TotalDistance	VeryActiveDistance	ModeratelyActiveDistance
0	1503960366	2016-04-12	Tuesday	13162	8.50	1.88	0.55
1	1503960366	2016-04-13	Wednesday	10735	6.97	1.57	0.69
2	1503960366	2016-04-14	Thursday	10460	6.74	2.44	0.40
3	1503960366	2016-04-15	Friday	9762	6.28	2.14	1.26
4	1503960366	2016-04-16	Saturday	12669	8.16	2.71	0.41

```
activegroup = df_totalsteps[df_totalsteps["AverageSteps"]==True]
low_activegroup = df_totalsteps[df_totalsteps["AverageSteps"]==False]
```

- 2群間での座っている時間の比較の棒グラフ作成

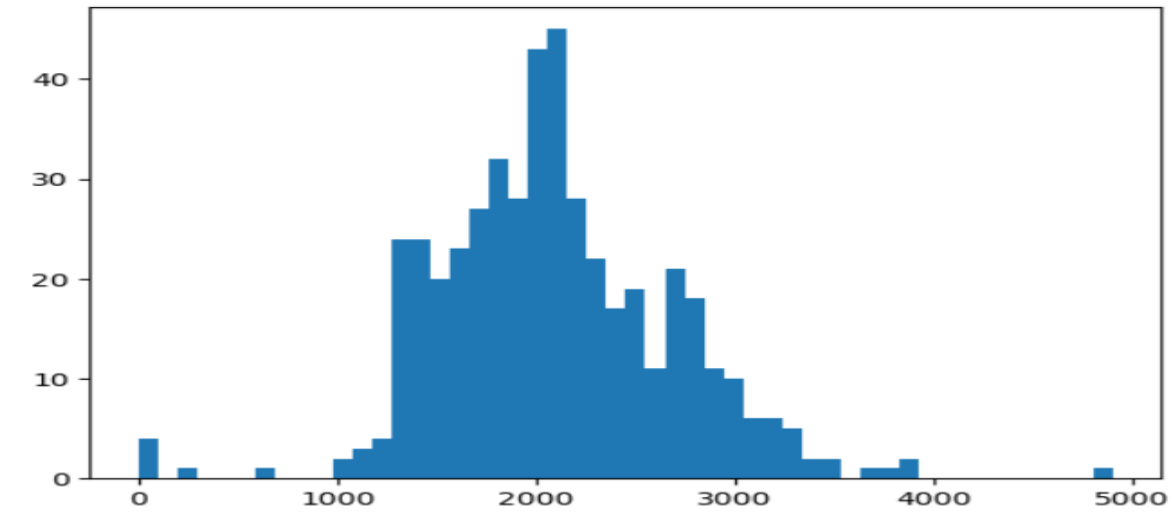
```
fig, ax = plt.subplots(figsize=(8,5))
x = np.array(["Active Group", "Low Active Group"])
y = np.array([activegroup["SedentaryMinutes"].mean(),
              low_activegroup["SedentaryMinutes"].mean()])

plt.title("Sedentary Minutes")
plt.bar(x, y, width = 0.8, color = "#f22a51")
plt.show()
```



- 2群間の差の検定前に正規性の有無を確認するため、ヒストグラムを作成

```
plt.hist(x=low_activegroup["Calories"], bins=50)
plt.show()
```



- 正規性の検定を行う

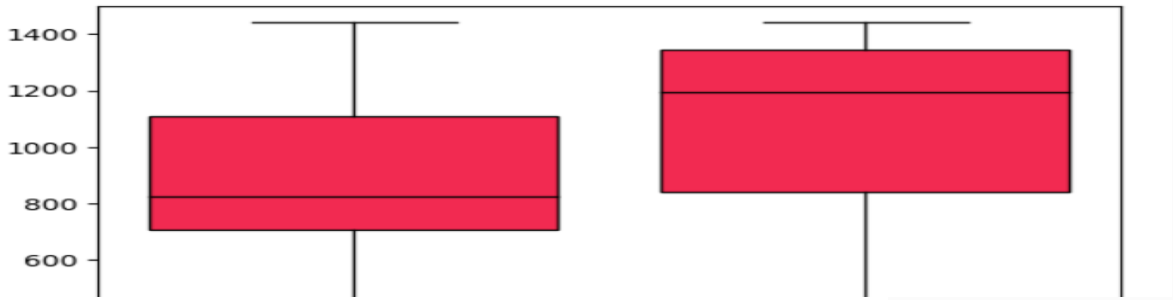
```
W, shapiro_p_value = stats.shapiro(activegroup["TotalActiveMinutes"])
print(f'Shapiro-Wilk test statistic: {W}, p-value: {shapiro_p_value}')
```

```
Shapiro-Wilk test statistic: 0.9644242525100708, p-value: 2.49777132310669e-09
```

- ノンパラメトリックでの2群における差の検定、マンホイットニーのU検定を行う

```
from scipy.stats import mannwhitneyu
result_sendentary = mannwhitneyu(activegroup["SedentaryMinutes"], low_activegroup["SedentaryMinutes"],
            alternative='less')
result_acitive = mannwhitneyu(activegroup["TotalActiveMinutes"], low_activegroup["TotalActiveMinutes"],
            alternative='greater')
result_calories = mannwhitneyu(activegroup["Calories"], low_activegroup["Calories"],
            alternative='greater')
print(f"SedentaryMinutes : {result_sendentary}")
print(f"TotalActiveMinutes : {result_acitive}")
print(f"Calories : {result_calories}")
```

```
SedentaryMinutes : MannwhitneyuResult(statistic=61980.0, pvalue=1.2047589694051304e-31)
TotalActiveMinutes : MannwhitneyuResult(statistic=161058.0, pvalue=2.272739410573179e-34)
Calories : MannwhitneyuResult(statistic=146327.0, pvalue=3.202429039576977e-18)
```

- 箱ひげ図にて可視化

```
x1 = activegroup_df["SedentaryMinutes"]
x2 = low_activegroup_df["SedentaryMinutes"]
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.boxplot([x1, x2],
           labels=['Active Group', 'Low Active Group'],
           patch_artist=True,
           widths=0.8,
          boxprops=dict(facecolor='#f22a51',
                        color=None),
          medianprops=dict(color='black', linewidth=1)
          )
ax.set_ylim(0, 1500)
plt.show()
```