Tableau + Python

・使用するデータ

種類: 自分のfitbitの毎分単位の心拍数のデータ

期間: 2016/04/01 ~ 2017/03/31

・Tableau + Pythonの環境

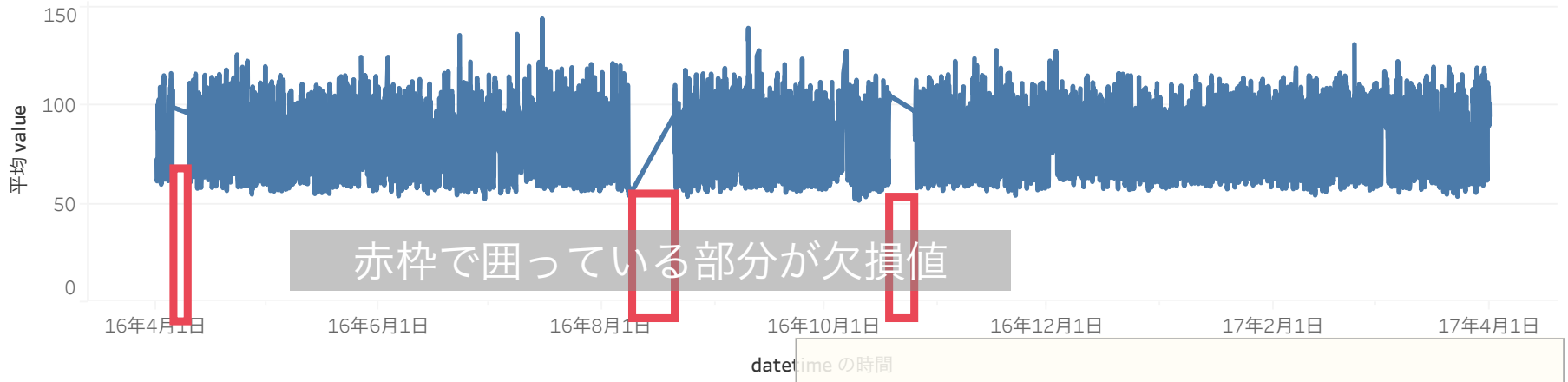
Python 3.6.1(pyenv anaconda3-4.2.0)

Tableau v10.3(Preview)

Tableau + Python

データ・環境の説明	データの確認	Tabpy	Tabpy(書き方)	変化点検知の説明	Tabpy(function deploy)	Tabpy: 変化点検知

データの確認



「ダッシュボード」の「アクション」を使って、
欠損が起きている部分のデータを確認する。

欠損値の処理

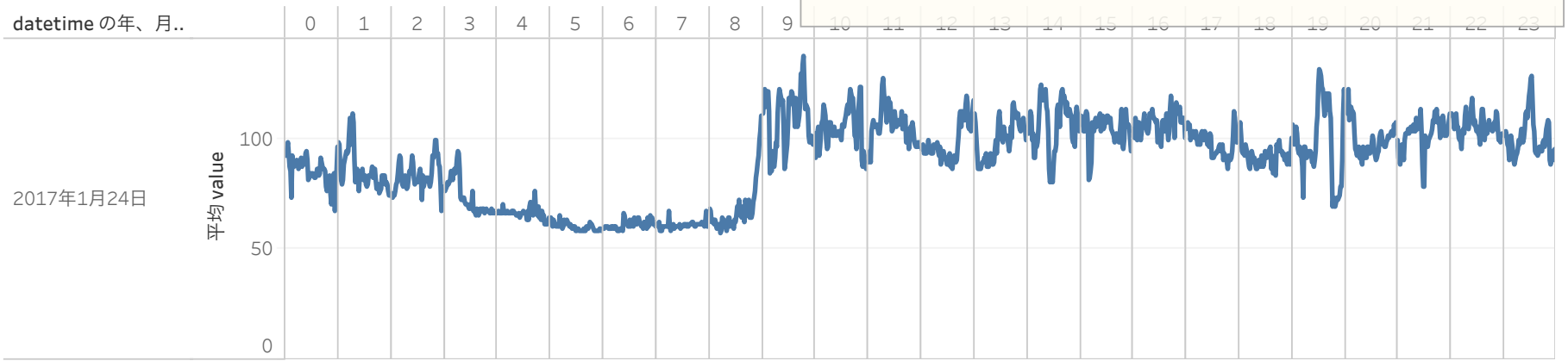


Tableau + Python

・Tabpy

Tableau v10.2から正式サポートされたPython連携の機能

※環境構築は、下記を参考にしてください。

https://github.com/tableau/TabPy

・TabpyでのScriptの書き方

1. Tableauの計算フィールドに直接書く。

2. tabpy-clientから関数として、Deployする。

Tableau + Python

・計算フィールドでの記述

・tabpy-clientでの記述

Tableau + Python

データの確認	Tabpy	Tabpy(書き方)	変化点検知の説明	Tabpy(function deploy)	Tabpy:変化点検知	クラスター分析: 導入

手頃な時系列のデータがあるので、
まずは変化点検知でTabpyを試してみる。

使用したパッケージ: changefinder

詳細は、こちら
http://argmax.jp/index.php?changefinder

Tableau + Python

Tabpy	Tabpy(書き方)	変化点検知の説明	Tabpy(function deploy)	Tabpy:変化点検知	クラスター分析: 導入	クラスター分析: データ準備 データ作成

・Jupyter notebookで作成

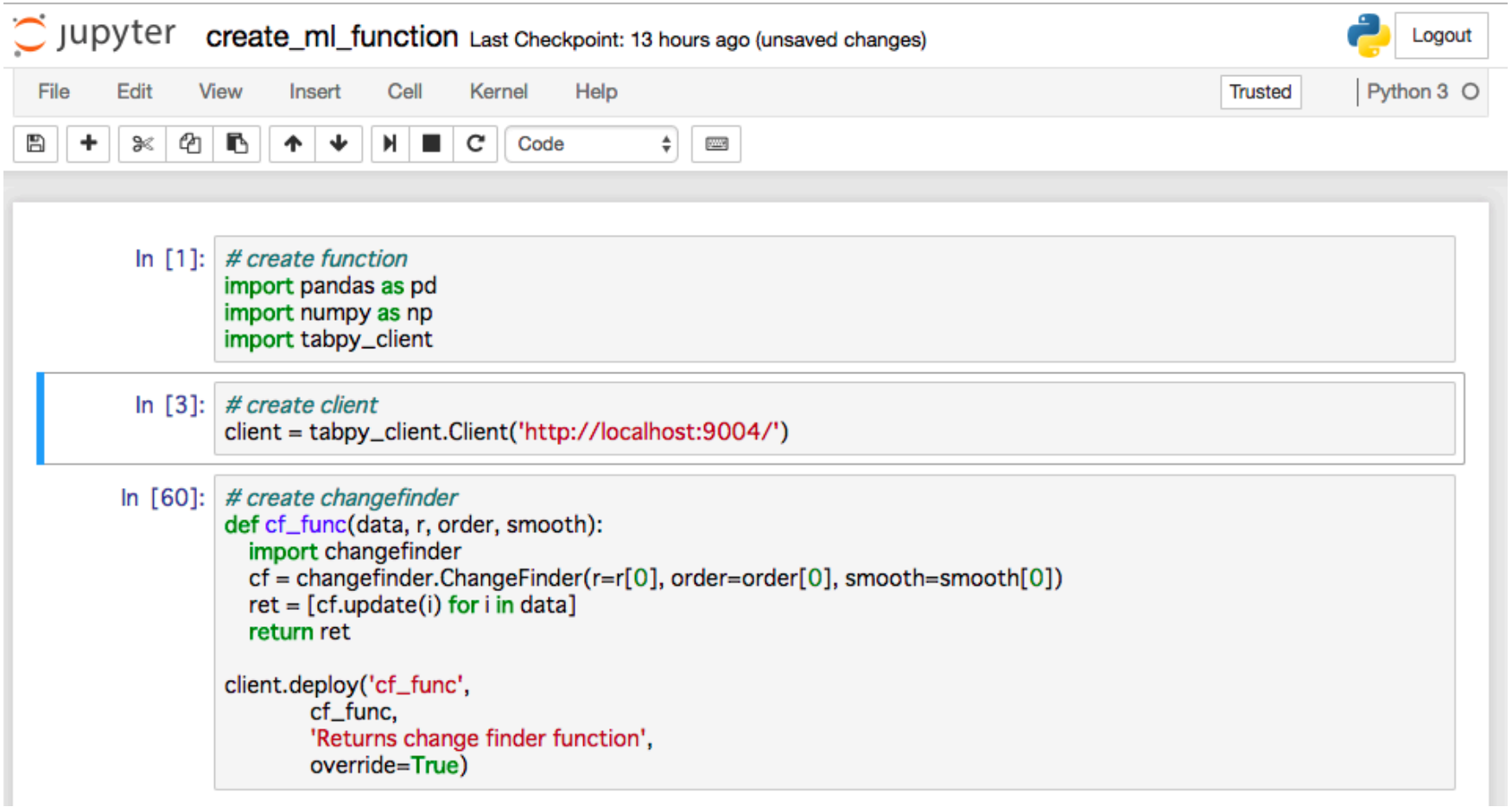
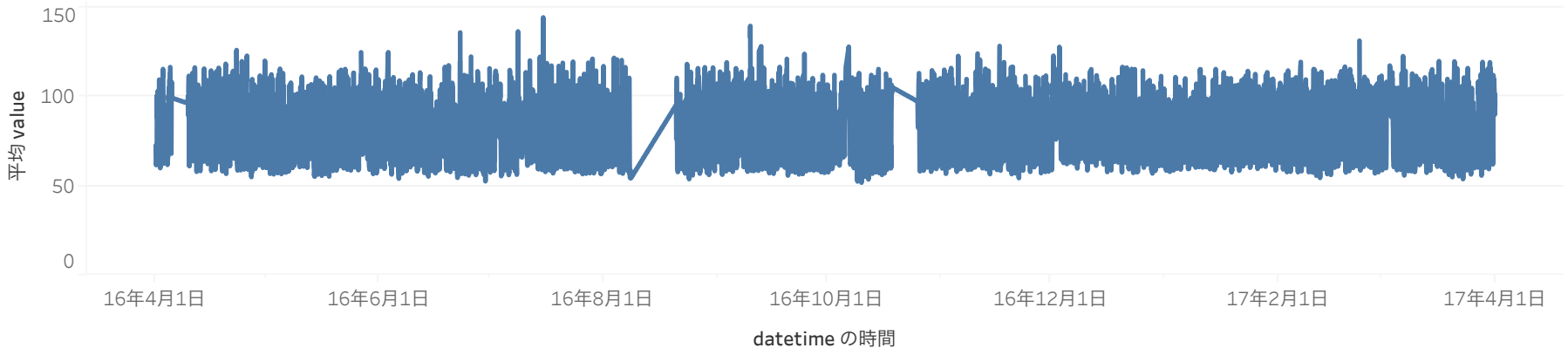


Tableau + Python

Tabpy(書き方)	変化点検知の説明	Tabpy(function deploy)	Tabpy:変化点検知	クラスター分析: 導入	クラスター分析: データ準備 データ作成	クラスター分析: データ準備 データコピー

データの確認



order 1 r 0.02 smooth 7

Tabpy_変化点検知(deploy)

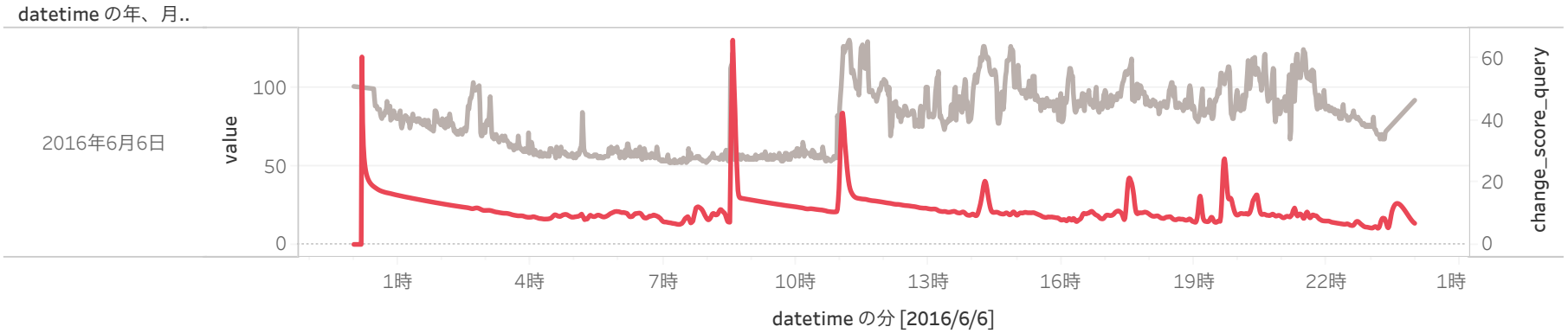


Tableau + Python

変化点検知の説明	Tabpy(function deploy)	Tabpy:変化点検知	クラスター分析: 導入	クラスター分析: データ準備 データ作成	クラスター分析: データ準備 データコピー	Kmeans Function

活動の特徴を把握するために、kmeansでセグメントを付与する。
TabpyでKmeansを動的に行う。

Tableau + Python

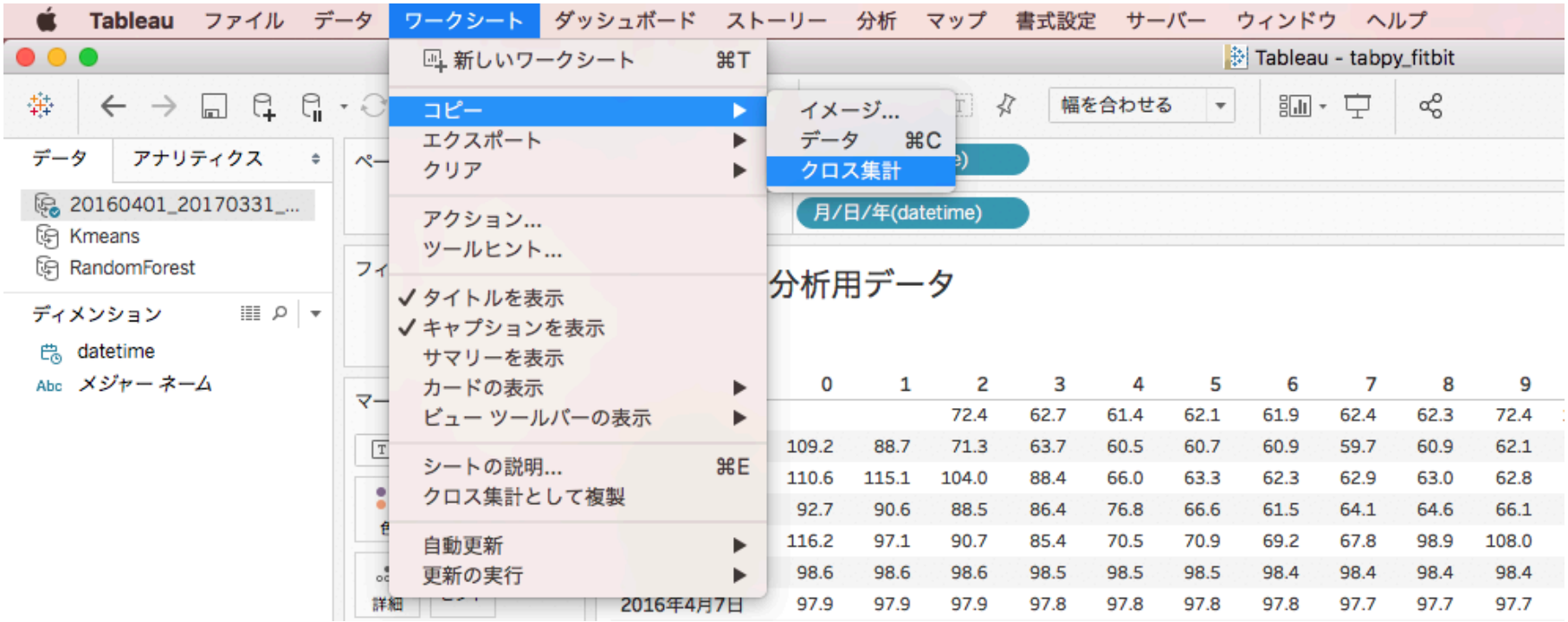
クラスター分析用データ

	datetime																							
datetime の年..	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
2016年4月1日			72.4	62.7	61.4	62.1	61.9	62.4	62.3	72.4	###	87.6	96.7	89.5	###	97.5	92.3	85.8	97.5	84.0	83.8	87.6	###	97.1
2016年4月2日	###	88.7	71.3	63.7	60.5	60.7	60.9	59.7	60.9	62.1	98.7	84.7	97.9	###	###	84.8	77.2	70.1	72.5	97.0	###	94.4	###	92.9
2016年4月3日	###	###	###	88.4	66.0	63.3	62.3	62.9	63.0	62.8	72.0	87.4	89.2	91.9	89.9	83.9	67.8	78.9	89.9	94.5	89.6	93.0	94.9	94.7
2016年4月4日	92.7	90.6	88.5	86.4	76.8	66.6	61.5	64.1	64.6	66.1	66.0	85.2	95.1	###	98.1	###	###	97.0	93.9	94.7	###	98.3	###	###
2016年4月5日	###	97.1	90.7	85.4	70.5	70.9	69.2	67.8	98.9	###	98.7	99.0	99.0	98.9	98.9	98.9	98.8	98.8	98.8	98.8	98.7	98.7	98.7	98.6
2016年4月6日	98.6	98.6	98.6	98.5	98.5	98.5	98.4	98.4	98.4	98.4	98.3	98.3	98.3	98.2	98.2	98.2	98.2	98.1	98.1	98.1	98.0	98.0	98.0	98.0
2016年4月7日	97.9	97.9	97.9	97.8	97.8	97.8	97.8	97.7	97.7	97.7	97.6	97.6	97.6	97.6	97.5	97.5	97.5	97.4	97.4	97.4	97.4	97.3	97.3	97.3
2016年4月8日	97.2	97.2	97.2	97.2	97.1	97.1	97.1	97.0	97.0	97.0	97.0	96.9	96.9	96.9	96.8	96.8	96.8	96.8	96.7	96.7	96.7	96.6	96.6	96.6
2016年4月9日	96.6	96.5	96.5	96.5	96.4	96.4	96.4	96.4	96.3	96.3	96.3	96.2	96.2	96.2	96.2	96.1	96.1	96.1	96.0	96.3	98.4	###	89.2	95.7
2016年4月10日	###	###	###	81.1	61.2	62.6	60.8	63.6	87.4	98.3	90.5	95.1	88.5	95.9	###	96.8	85.4	94.2	###	83.9	75.3	73.7	###	###
2016年4月11日	###	###	###	84.1	83.6	69.7	66.8	63.6	67.2	65.6	63.8	99.2	95.2	97.6	98.9	###	###	###	###	###	95.2	93.9	91.2	83.6
2016年4月12日	92.7	84.9	80.9	67.8	60.2	57.7	61.3	59.9	92.3	###	99.3	98.7	###	###	97.1	###	98.8	91.9	87.0	95.5	88.6	###	###	91.8
2016年4月13日	92.0	85.6	79.6	59.1	58.4	58.1	[illegible]	[illegible]	[illegible]	###	[illegible]	###	[illegible]	[illegible]	[illegible]	96.6	88.0	[illegible]	###	97.8	97.5	###	###	###
2016年4月14日	92.3	83.2	72.5	63.4	66.7	92.2	###	###	96.3	###	89.5	86.6	89.9	99.3	95.2	86.5	89.5	87.6	84.0	90.4	96.4	###	85.2	73.1
2016年4月15日	70.3	80.5	###	91.3	84.2	71.6	62.0	62.9	81.8	###	89.5	###	99.9	96.0	91.2	###	###	92.5	74.9	66.7	65.7	67.7	81.8	86.7
2016年4月16日	94.4	###	###	###	82.4	58.1	59.0	61.8	65.1	64.9	79.2	96.5	93.8	93.7	84.8	89.2	91.6	93.4	86.9	68.5	68.6	64.8	72.6	96.7
2016年4月17日	###	###	81.7	79.9	72.1	62.6	60.4	58.8	61.1	62.4	61.1	75.4	###	92.9	###	###	88.4	82.6	90.2	93.6	87.7	95.6	94.7	###
2016年4月18日	84.2	89.4	96.0	###	###	###	93.6	64.4	63.2	64.1	75.7	###	98.3	###	###	###	98.0	###	93.2	###	97.9	###	###	94.3
2016年4月19日	91.4	76.5	66.1	59.8	59.0	58.0	59.0	59.8	###	###	###	99.4	95.1	###	97.3	99.5	97.8	84.3	###	90.2	93.7	###	93.4	###
2016年4月20日	###	###	###	###	59.7	58.8	56.5	58.3	90.6	###	94.4	87.3	83.7	90.9	89.3	93.8	98.4	93.0	94.1	91.0	95.2	###	88.4	88.7
2016年4月21日	96.5	###	###	71.2	66.4	63.6	60.8	61.2	81.9	###	91.7	90.2	95.3	###	99.6	94.4	92.9	92.7	91.3	99.7	###	###	81.1	76.4
2016年4月22日	68.9	85.3	92.5	###	88.3	78.6	65.6	65.1	67.5	89.8	87.7	92.2	###	92.8	90.4	93.0	86.0	89.9	86.1	82.0	91.3	99.1	###	94.5
2016年4月23日	73.8	85.5	###	###	75.8	62.6	61.4	60.0	58.9	58.4	59.5	65.8	75.4	###	###	63.3	62.3	69.5	81.4	66.9	67.3	90.3	89.3	93.0
2016年4月24日	###	###	89.4	84.9	79.0	62.9	58.6	59.3	59.7	62.3	66.7	69.9	###	###	###	89.2	74.2	###	99.3	92.5	96.8	94.8	81.5	###
2016年4月25日	###	86.3	76.8	62.2	59.0	60.2	59.3	62.1	59.4	97.4	###	96.6	96.4	84.2	89.3	97.3	91.1	89.0	83.2	86.6	###	###	###	###

1日の心拍数を毎時で平均したクロス表

Tableau + Python

Tabpy:変化点検知	クラスター分析: 導入	クラスター分析: データ準備 データ作成	クラスター分析: データ準備 データコピー	Kmeans Function	Kmeans結果	RandomForest: 導入



クラスター分析用データのシートを選択し、
メニューから、「ワークシート」 => 「コピー」=> 「クロス表」

Tableau + Python

クラスター分析: 導入	クラスター分析: データ準備 データ作成	クラスター分析: データ準備 データコピー	Kmeans Function	Kmeans結果	RandomForest: 導入	RandomForest: データ準備

Tabpy: Kmeans Function

In [71]:
```
# create kmenas function
def kmeans_func(n_clus, rstat, *args):
    import pandas as pd
    import numpy as np
    import numpy as np
    from sklearn.cluster import KMeans

    df = pd.DataFrame()
    for arg in args:
        df = pd.concat([df, pd.DataFrame(arg)] , axis=1)
    X = df.as_matrix()
    kmeans_model = KMeans(n_clusters=n_clus[0], random_state=rstat[0])
    db = kmeans_model.fit(X)
    return db.labels_.tolist()

client.deploy('kmeans',
              kmeans_func,
              'Returns kmeans label',
              override=True)
```

Tableau + Python

クラスター分析:
データ準備
データ作成

クラスター分析:
データ準備
データコピー

Kmeans Function

Kmeans結果

RandomForest: 導入

RandomForest:
データ準備

RandomForest:
function

Cluster1: 休日
Cluster2: 欠損値補完？
Cluster3: 遅刻
Cluster4: 通常勤務

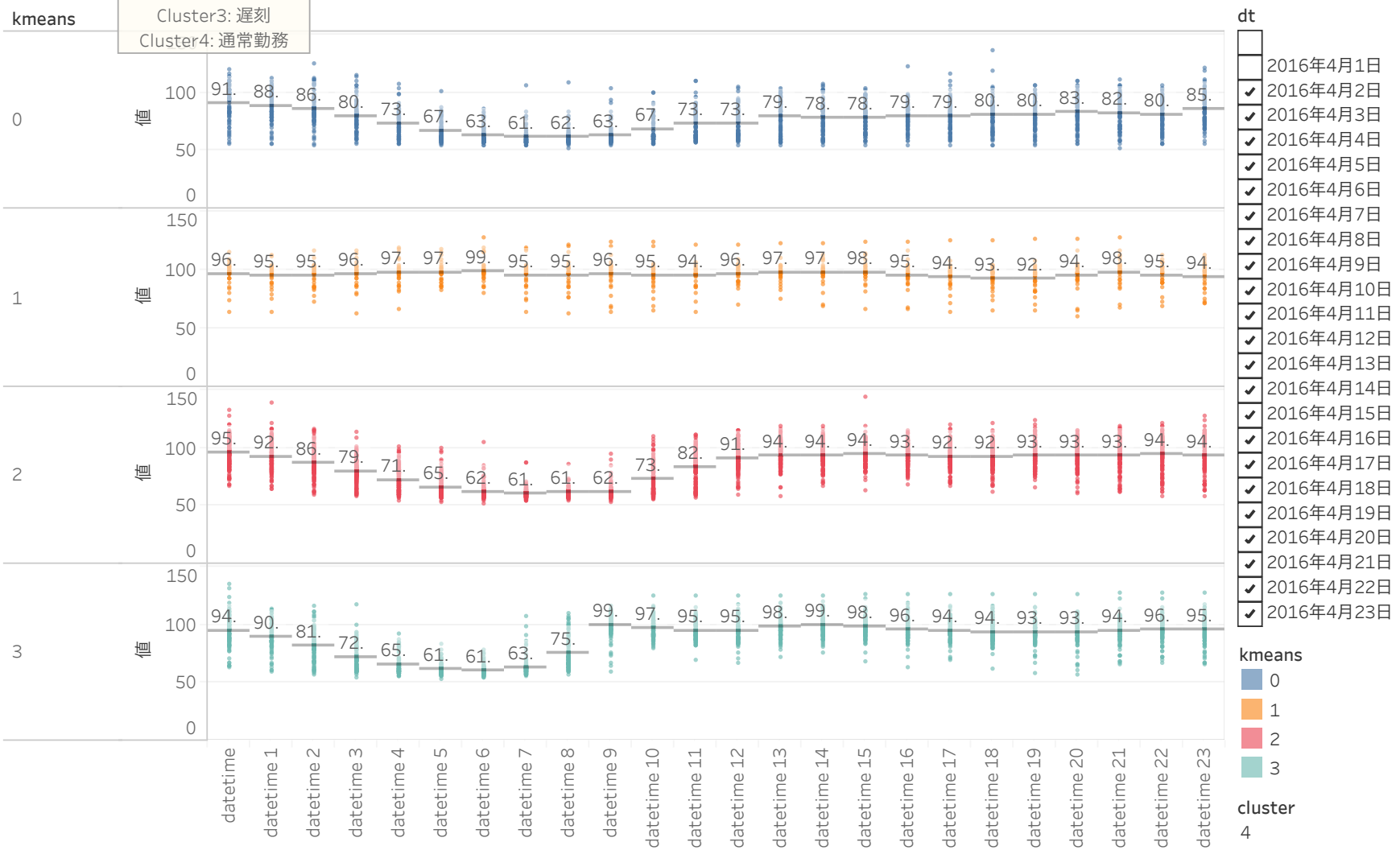


Tableau + Python

クラスター分析: データ準備 データコピー	Kmeans Function	Kmeans結果	RandomForest: 導入	RandomForest: データ準備	RandomForest: function	RandomForest: 結果確認

Kmeansでセグメントの付与ができたら、
今度は未知(将来)のデータに対して、
そのセグメントの判別モデルを作成したくなる。

RandomForestでサクッと作成

Tableau + Python

クラスター分析: データ準備 データコピー	Kmeans Function	Kmeans結果	RandomForest: 導入	RandomForest: データ準備	RandomForest: function	RandomForest: 結果確認

クラスター分析と同じ手順を
クラスター番号を追加して、クロス表をコピーする。

Tableau + Python

クラスター分析: データ準備 データコピー	Kmeans Function	Kmeans結果	RandomForest: 導入	RandomForest: データ準備	RandomForest: function	RandomForest: 結果確認

Random Forest Function

In [14]:
```
def randomforest_func(n_estimators, label, test_size, *args):
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import train_test_split
    import pandas as pd
    import numpy as np

    # 引数をDataFrameにする
    df = pd.DataFrame()
    for arg in args:
        df = pd.concat([df, pd.DataFrame(arg)] , axis=1)
    X = df.as_matrix()

    # Split Train and Test data
    X_train, X_test, y_train, y_test = train_test_split(X, label, test_size=test_size[0], random_state=71)

    clf = RandomForestClassifier(n_estimators=n_estimators[0])
    clf.fit(X_train,y_train)
    yhat_train = clf.predict(X_train)
    print("Train Accuracy: {0}".format(sum(yhat_train == y_train) / len(y_train)))

    yhat_test = clf.predict(X_test)
    print("Test Accuracy: {0}".format(sum(yhat_test == y_test) / len(y_test)))

    result = clf.predict(X)
    return result.tolist()

client.deploy('randomforest',
              randomforest_func,
              'Returns predict label',
              override=True)
```

Tableau + Python

クラスター分析:
データ準備
データコピー | Kmeans Function | Kmeans結果 | RandomForest: 導入 | RandomForest:
データ準備 | RandomForest:
function | **RandomForest:
結果確認**