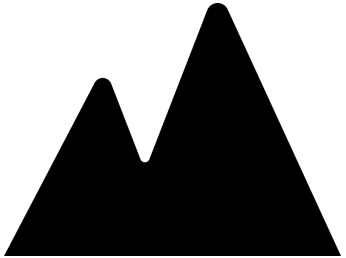
クラスタリングと
レコメンデーション

Integration Step Project
2017年12月23日
下角 康子

何のデータでしょう？

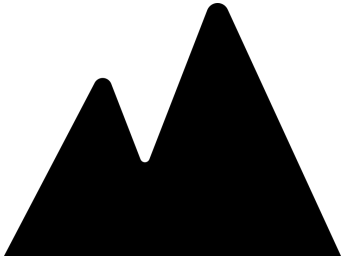
No	名前	時期	累積標高(公式)	ピーク数	距離(公式)	ペース	体感的辛さ
1		2016/2/6	982m		13.25km	13:01/km	5
		2016/6/25	388m		12.20km	11:48/km	1
		2016/8/11	1209m		18.02km	13:13/km	8
		2016/9/4	1263m		16.38km	15:33/km	6
		2016/10/2	1186m		17.13km	12:52/km	7
		2016/10/10	1058m(1200m)		19.30km(16km)	8:33/km	5
		2016/12/29	739m		12.97km	8:13/km	4
		2017/2/12	2091m(1866m)	37個	34.90km(30km)	8:51/km	9



とある日の山行データです

No	名前	時期	累積標高(公式)	ピーク数	距離(公式)	ペース	体感的辛さ
1	高尾トレイル	2016/2/6	982m		13.25km	13:01/km	5
2	逗子ゆるトレイル	2016/6/25	388m		12.20km	11:48/km	1
3	塩山やまの日トレイル	2016/8/11	1209m		18.02km	13:13/km	8
4	高尾トレイル	2016/9/4	1263m		16.38km	15:33/km	6
5	箱根金時山トレイル	2016/10/2	1186m		17.13km	12:52/km	7
6	五箇山レース	2016/10/10	1058m(1200m)		19.30km(16km)	8:33/km	5
7	武田の杜@甲府トレイル	2016/12/29	739m		12.97km	8:13/km	4
8	くだまつ笠戸島レース	2017/2/12	2091m(1866m)	37個	34.90km(30km)	8:51/km	9

昨年から、趣味でトレイルランニングをしています。



トレイルランニングとは

トレイルランニング（英: Trail running）は、陸上競技の中長距離走の一種で、**舗装路以外の山野を走るものをさす。トレラン**や**トレイルラン**と略される。**山岳レース**とも呼ばれる。

マラソンと同様にほとんど装備を持たずに走るクロスカントリーとは異なり、トレイルランニングで**専用の小型リュックサックに必要な装備を入れて走ることが普通である。**

黎明期では「ランニング登山」と称して、通常登山靴や、登山用の杖などを装備して登るような山を、Tシャツに短パン、スパッツ、ランニング・シューズといったランニングのスタイルで入山して走っていた。

しかし近年のトレイルランニングの普及によって専用の装備も開発されるようになり**トレイルランニング製品の市場も拡大しつつある。**例えば、シューズでは、登山靴でもランニングシューズでもない、軽く走りやすくグリップの良い**トレイルランニング用シューズ**を使うことが認知されてきた。また、水筒の代わりにチューブを使って給水できるハイドレーションシステム、走りをサポートするストックなども、使われるようになってきている。

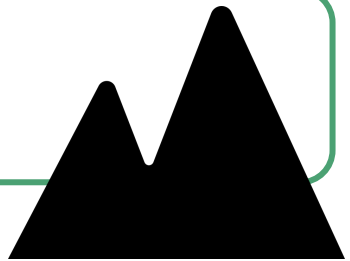
出典：https://ja.wikipedia.org/wiki/トレイルランニング

補足

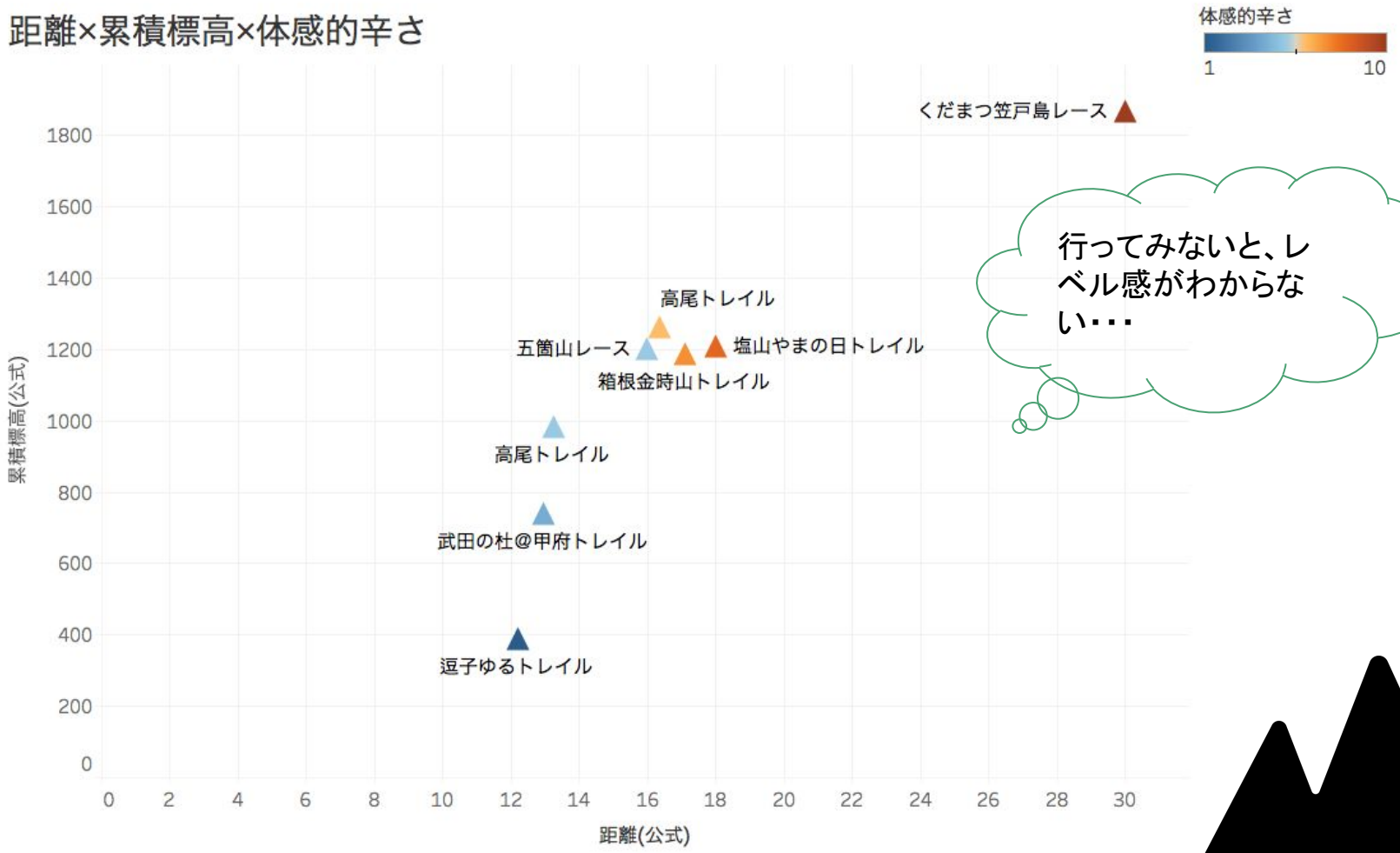
魅力は？
➔岩や木の根などの障害物を避けるため、複雑な足運びとペースコントロールが要求される。より路面への意識を集中させる必要あり。

醍醐味
➔集中力MAXで下りを駆け抜ける！
ロードよりも、景色や地形の変化が刺激的。

難しさ
➔登りのペース配分。登りでの消耗が命取りに…



距離×累積標高×体感的辛さ



行ってみないと、レベル感がわからない・・・

テーマ

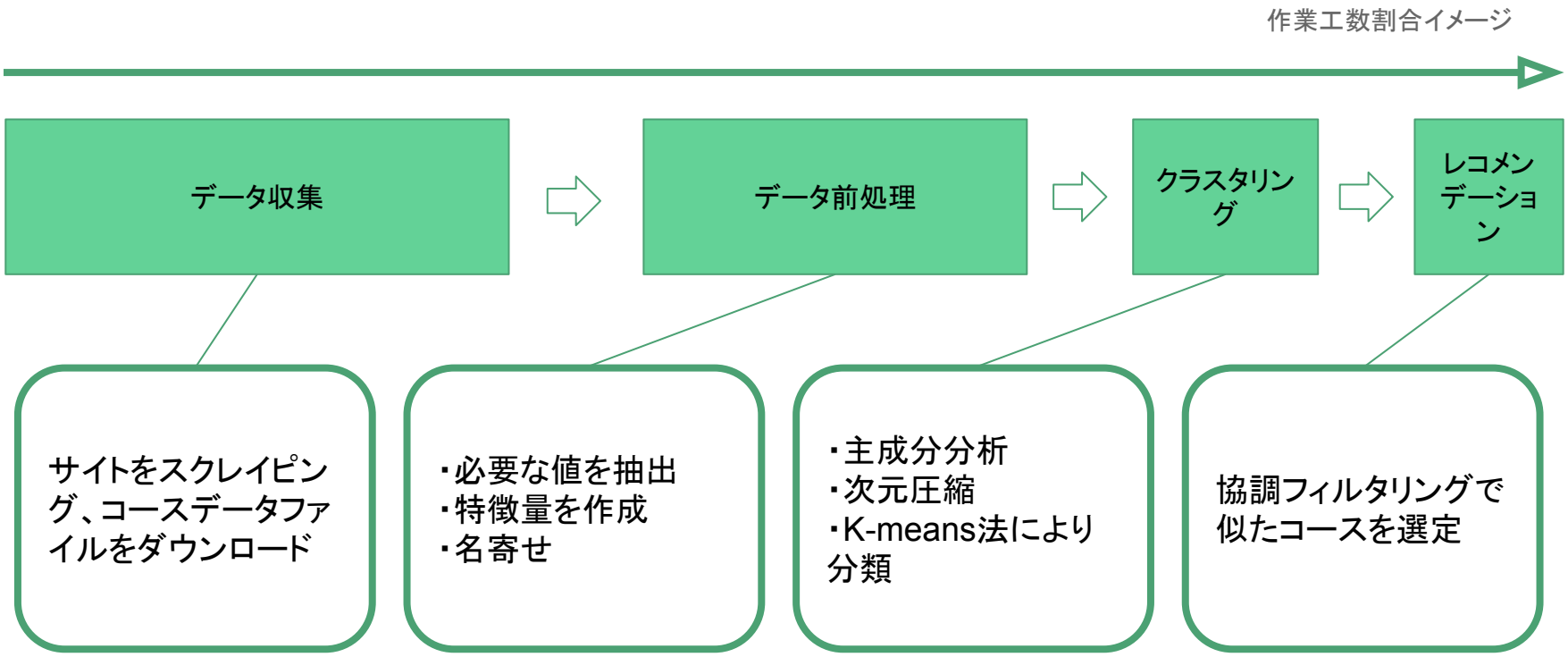
どの山、どのコースが似ているのか？

①日本全国のヤマレコユーザーのデータを使って
トレイルコースを分類

②行ったコースに近いコースをレコメンド

コース選定がしやすくなる！

作業流れ


作業工数割合イメージ
データ収集
データ前処理
クラスタリング
レコメンデーション
サイトをスクレイピング、コースデータファイルをダウンロード
・必要な値を抽出
・特徴量を作成
・名寄せ
・主成分分析
・次元圧縮
・K-means法により分類
協調フィルタリングで似たコースを選定

<1>データ収集

GPX

収集元：https://www.yamareco.com/

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" x
gpx" version="1.1" creator="Yamatabi logger for Android" xsi:sche
gpx.xsd  http://www.chizroid.info/ns/gpx http://www.chizroid.info/ns
<trk>
<name><![CDATA[塔ノ岳/2017-11-19 05:54:37]]></name>
<trkseg>
<trkpt lat="35.405045" lon="139.16858"><ele>315</ele><time>2017-11-
<trkpt lat="35.405075" lon="139.1686"><ele>297</ele><time>2017-11-18T20:54:42Z</time></trkpt>
<trkpt lat="35.405003" lon="139.16844"><ele>297</ele><time>2017-11-18T20:55:02Z</time></trkpt>
<trkpt lat="35.405025" lon="139.16843"><ele>297</ele><time>2017-11-18T20:55:15Z</time></trkpt>
<trkpt lat="35.40518" lon="139.1684"><ele>298</ele><time>2017-11-18T20:55:26Z</time></trkpt>
<trkpt lat="35.405243" lon="139.16837"><ele>298</ele><time>2017-11-18T20:55:31Z</time></trkpt>
<trkpt lat="35.405296" lon="139.16832"><ele>298</ele><time>2017-11-18T20:55:36Z</time></trkpt>
<trkpt lat="35.405354" lon="139.16829"><ele>299</ele><time>2017-11-18T20:55:41Z</time></trkpt>
<trkpt lat="35.405415" lon="139.16826"><ele>299</ele><time>2017-11-18T20:55:46Z</time></trkpt>
<trkpt lat="35.40548" lon="139.16823"><ele>299</ele><time>2017-11-18T20:55:51Z</time></trkpt>
<trkpt lat="35.40554" lon="139.16818"><ele>300</ele><time>2017-11-18T20:55:56Z</time></trkpt>
```

lat="35.XXXX"
lon="139.XXXXX"
ele="478"
1レコード間は1秒～

＜1＞データ収集

使用ツール：Python + Selenium + ChromeDriver

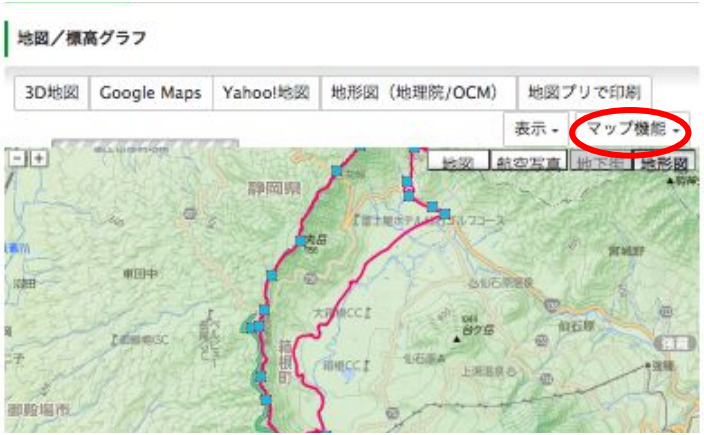
会員サイト
ログイン

山行記録ページ

トレイル
カテゴリ選択

詳細ページ
一覧取得

各ページにて
GPXをDL



＜2＞データ前処理

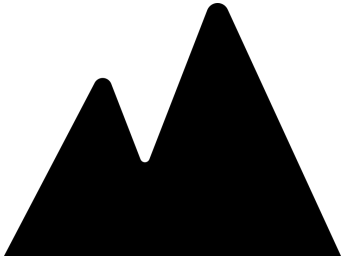
①GPXファイルの形式から以下の形に抽出

Course ID	lat	lon	ele
00001	35.9876	139.28746	321
00001	35.9584	139.28750	322
00001	35.8875	139.28800	322
00001	35.8865	139.29000	334

②レコード間の差分から、距離や累積標高を算出

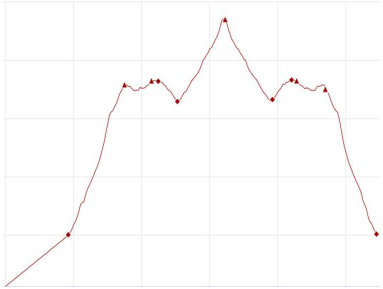
③分類に効きそうな特徴量を作成

④名寄せ：距離や緯度経度が近しいコースは同一コースとする

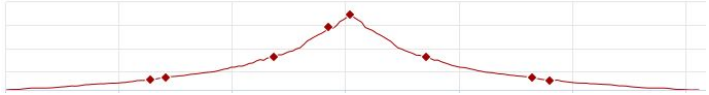


＜2＞データ前処理：特徴量作成

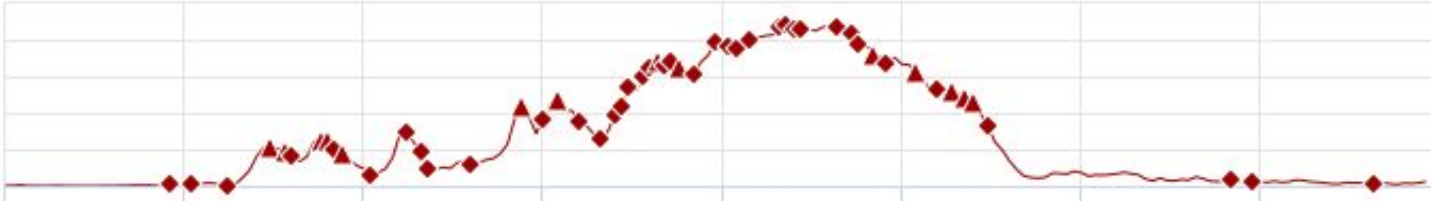
知りたいこと：どんなタイプの山か？　➔どんなタイプがあり得るか



短距離だが、急勾配



中距離、緩やかな勾配



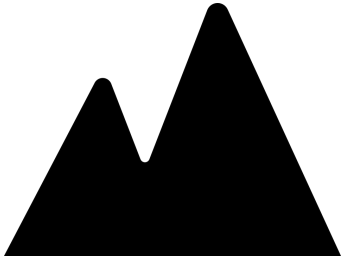
長く、ある程度の上り、高山

距離

累積標高

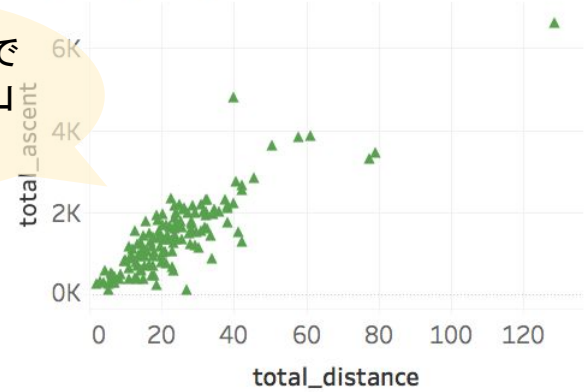
勾配

標高差

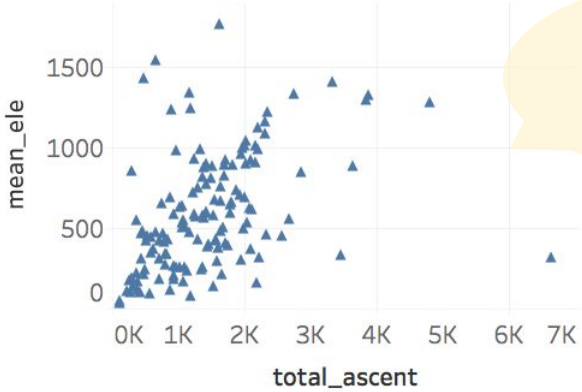


累積標高だけでは、高山か低山か不明

距離×累積標高

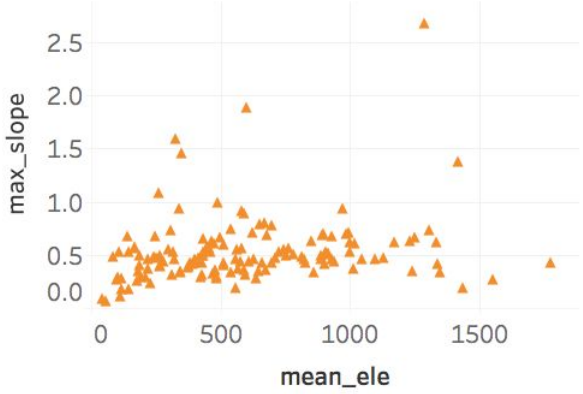


累積標高×平均標高

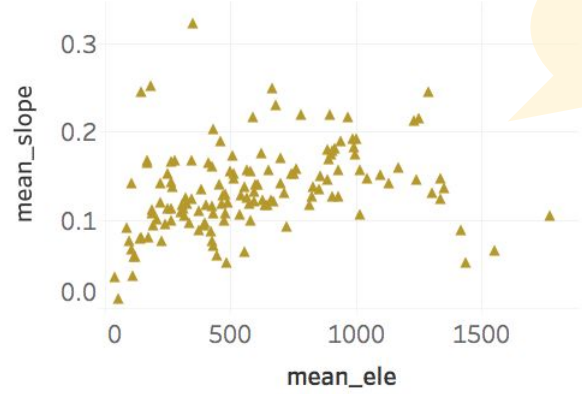


平均の標高をみると、ばらつきが

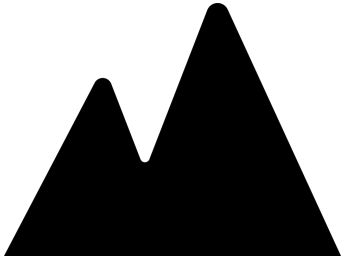
平均標高×最高勾配



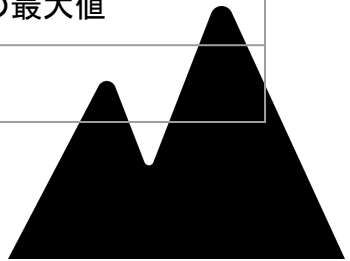
平均標高×平均勾配



急勾配が多いほど厳しいはず



特徴量	変数名	算出方法
全長距離	total_distance	レコード間の距離を緯度経度から計算して足し上げる。
累積標高（上り）	total_ascent	距離100m毎のレコード間の標高差が＋の場合を足しあげる。
累積標高（下り）	total_descent	距離100m毎のレコード間の標高差がーの場合を足しあげる。
平均標高	mean_ele	全レコードの標高の平均値
標高中央値	median_ele	全レコードの標高の中央値
最高標高	max_ele	全レコード中、標高の最大値
平均勾配	mean_slope	標高差／距離(%)　100m毎のデータに間引いたレコードの平均値
勾配中央値	median_slope	標高差／距離(%)　100m毎のデータに間引いたレコードの中央値
最大勾配	max_slope	標高差／距離(%)　100m毎のデータに間引いたレコードの最大値
高低差	dif_ele	最高標高と最低標高の差分



地獄のレース ～前処理トラップランキング～

UTMB級	累積標高の算出
MDS級	データ抽出高速化
UTMF級	スクレイピング(データ収集)
STY級	名寄せ(コース定義)

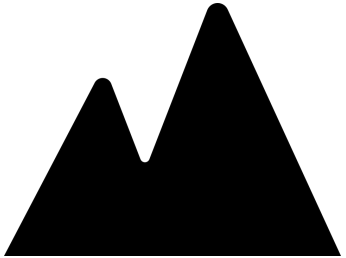
STY級：名寄せ（コース定義）

静岡 TO 山梨　92km　4100m　制限20時間　富士山の周囲を半周

単純にコース毎の数値の比較で、同じコースデータを省く。

名寄せ条件：
①スタート／ゴール地点がほぼ同じ（半径約500m以内）
②全長距離がほぼ同一（1km以内）
③最高標高がほぼ同一（100m以内）

結果、約150➡140コースに
・想定以上に同じコースが入っていなかった
・実質同じコースでも、スタート／ゴール地点は、様々
・使用機種による誤差も
➡レコメンデーションの際には、近すぎるコースを省くことで対応



UTMF級：スクレイピング（データ収集）

ULTRA-TRAIL Mt.FUJI　170km　8000m　制限46時間　富士山の周囲を一周

サイト
ログイン
➡
ページ遷移
➡
詳細一覧
➡
詳細ページ
でDL

計画　➡　Python + Urllib + BeautifulSoup

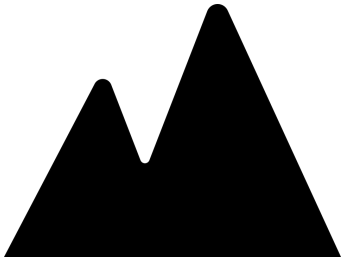
　┗ログインに必要な情報がわからず、やり方変更

・Selenium: Webブラウザを操作してくれるテストツール

・ChromeDriver:PhantomJSを利用したが、ファイルのダウンロードに対応していないため、Chromeに変更

・初めてのCSSセレクタ:右クリック+プルダウンメニューをクリックしてDL

感想：便利！！！



MDS級：データ抽出高速化

サハラマラソン　南モロッコ　230km　6ステージ7日間

当初、約300ファイルを正規表現で取得 → 一晩以上待って1/4くらい…
└データによっては、数万行あるため

正規表現re → in演算子+splitで対処 → 20%高速化（データによる）

結果、
・150ファイルに絞り、
・パラレル(物理的に)で実行
→2～3時間で完了！

UTMB級：累積標高の算出

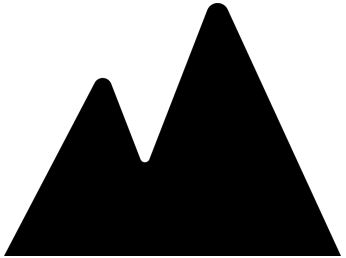
ウルトラトレイル・デュ・モンブラン　169km　9800m　フランス・スイス・イタリアに跨る

想定 ➡ 距離同様、＜ele＞の値をレコード間で足し上げ
　　レコード間で＋の値のみを足していく＝累積標高(上り)

想定外 ➡ GPS計測が正確でない
　　レコード間が距離1mに対し、標高が整数のデータもある

そもそも、累積標高の計算には諸説ある・・・
➡サイト上の値＋実体験より、
距離100m間隔の標高差であれば、ある程度妥当な数値が得られると判断

➡100m毎に間引いたデータで計算した場合誤差がかなり縮まった



＜3＞クラスタリング

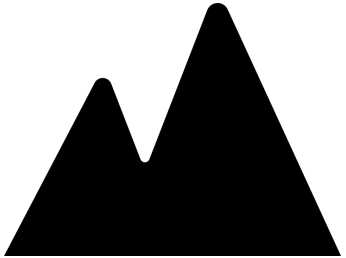
…の前に

緯度・経度、標高のデータから、10の特徴量を作成

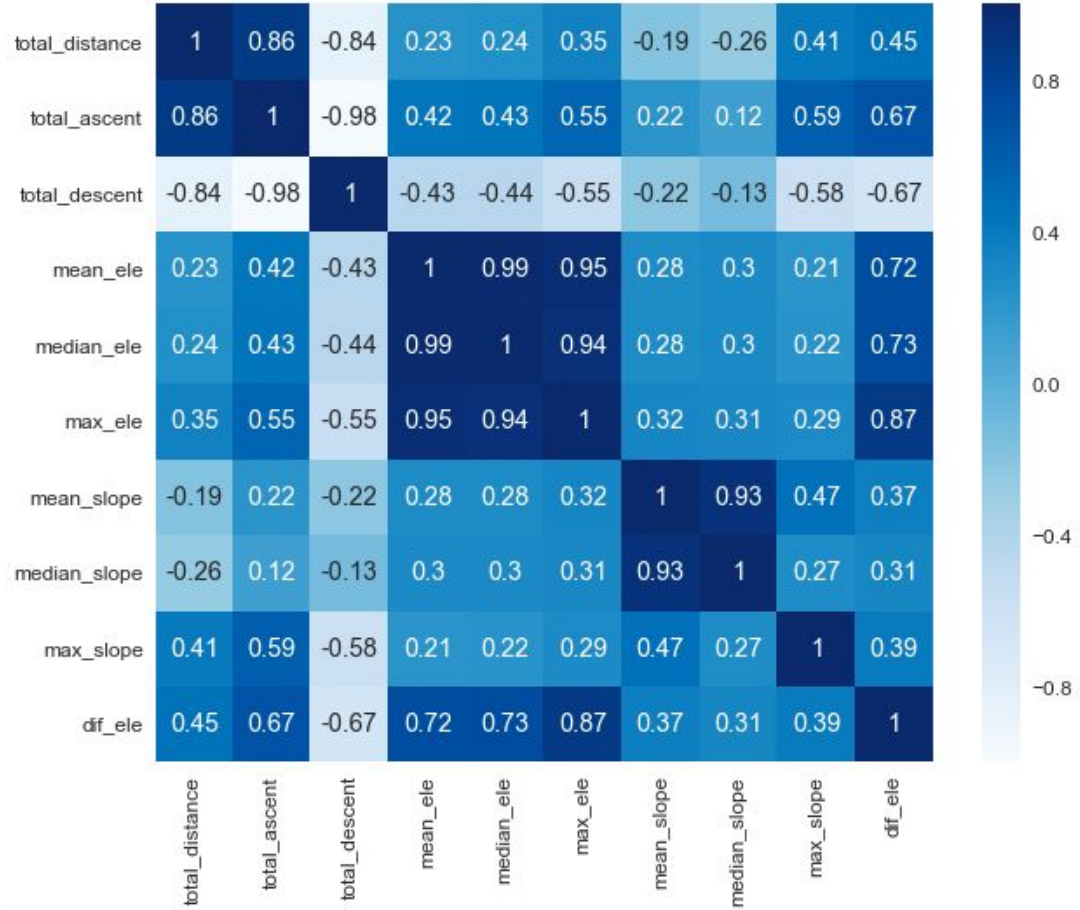
➡相関が高そうなものもある

①相関行列を確認

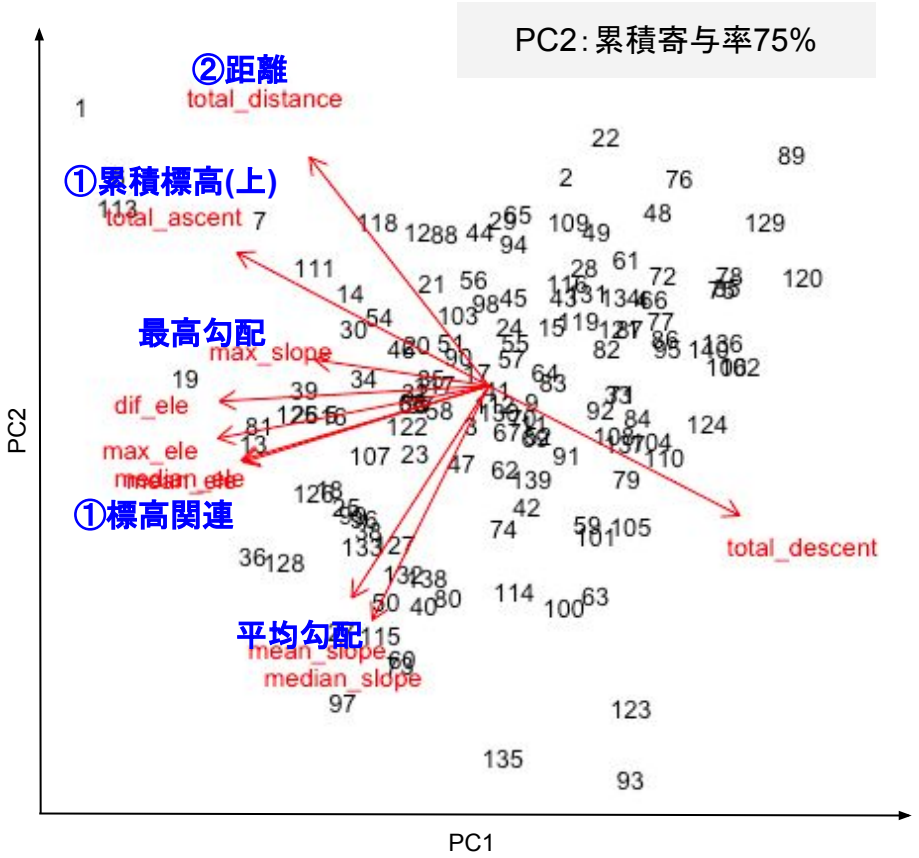
②主成分分析によって、2次元で各特徴量の関係性を可視化

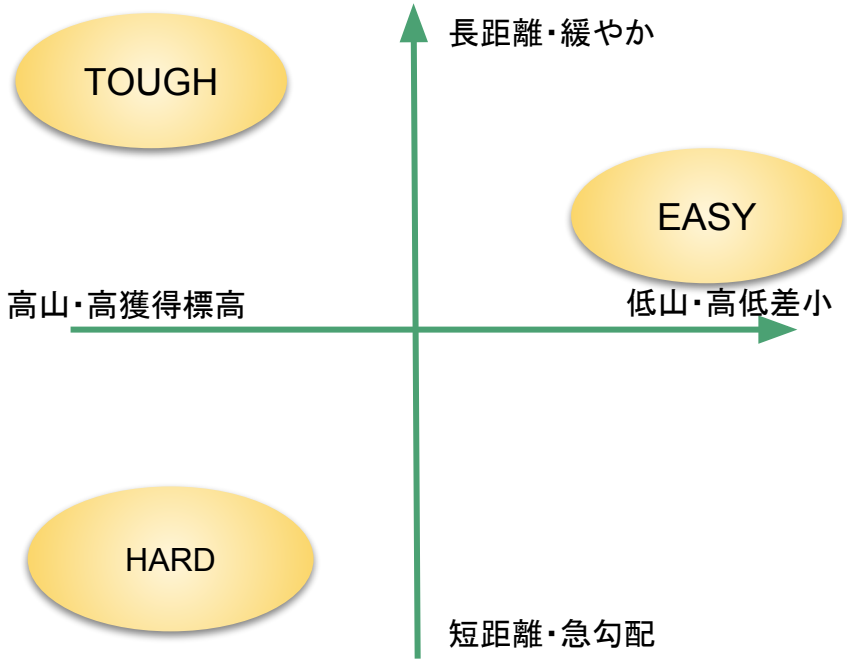


①相関行列を確認


total_distance 1 0.86 -0.84 0.23 0.24 0.35 -0.19 -0.26 0.41 0.45
total_ascent 0.86 1 -0.98 0.42 0.43 0.55 0.22 0.12 0.59 0.67
total_descent -0.84 -0.98 1 -0.43 -0.44 -0.55 -0.22 -0.13 -0.58 -0.67
mean_ele 0.23 0.42 -0.43 1 0.99 0.95 0.28 0.3 0.21 0.72
median_ele 0.24 0.43 -0.44 0.99 1 0.94 0.28 0.3 0.22 0.73
max_ele 0.35 0.55 -0.55 0.95 0.94 1 0.32 0.31 0.29 0.87
mean_slope -0.19 0.22 -0.22 0.28 0.28 0.32 1 0.93 0.47 0.37
median_slope -0.26 0.12 -0.13 0.3 0.3 0.31 0.93 1 0.27 0.31
max_slope 0.41 0.59 -0.58 0.21 0.22 0.29 0.47 0.27 1 0.39
dif_ele 0.45 0.67 -0.67 0.72 0.73 0.87 0.37 0.31 0.39 1
total_distance
total_ascent
total_descent
mean_ele
median_ele
max_ele
mean_slope
median_slope
max_slope
dif_ele
0.8
0.4
0.0
−0.4
−0.8

②主成分分析で可視化

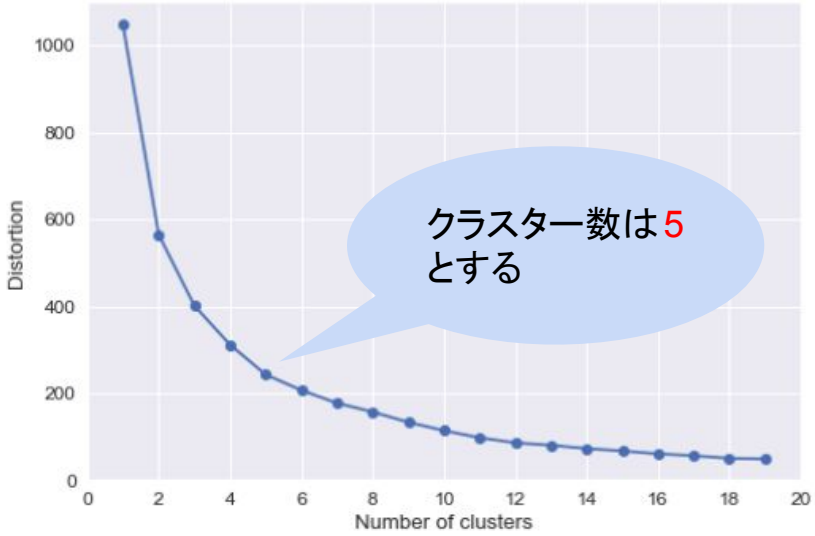

PC2：累積寄与率75%
②距離
total_distance
①累積標高(上)
total_ascent
最高勾配
max_slope
dif_ele
max_ele
①標高関連
total_descent
平均勾配
mean_slope
median_slope
PC2
PC1


TOUGH
EASY
HARD
長距離・緩やか
高山・高獲得標高
低山・高低差小
短距離・急勾配

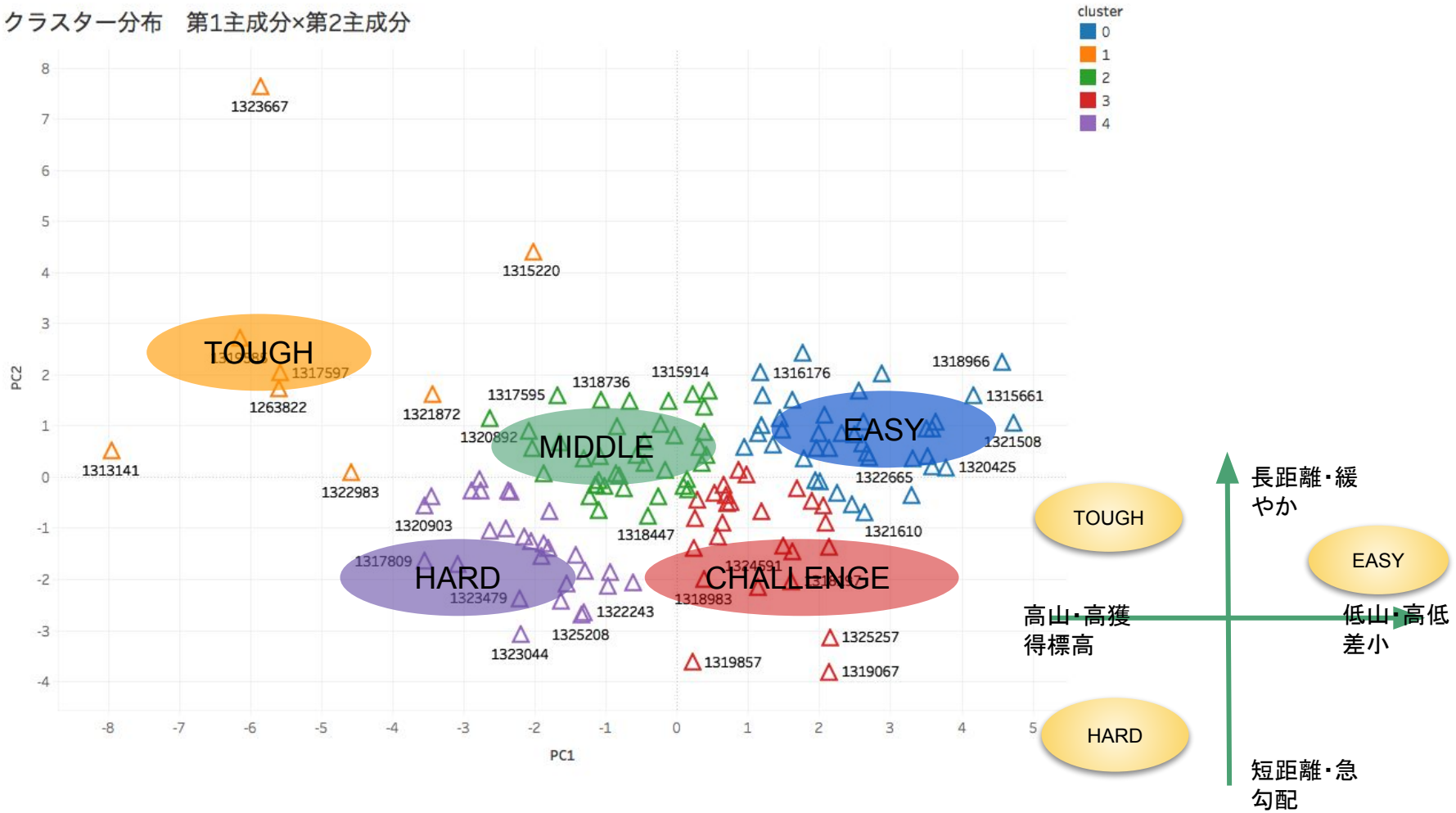
＜3＞クラスタリング：K-means法

PCAで次元圧縮した第1主成分、第2主成分を使って
K-meansでクラスリングを行う。

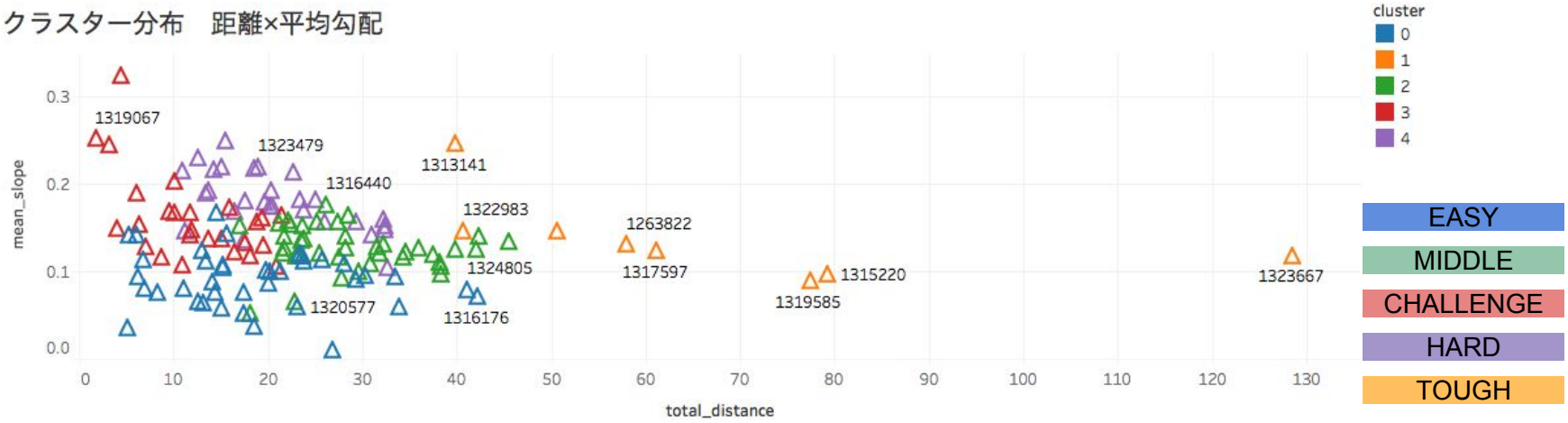
■最適なクラスター数：Elbow Methodで決定



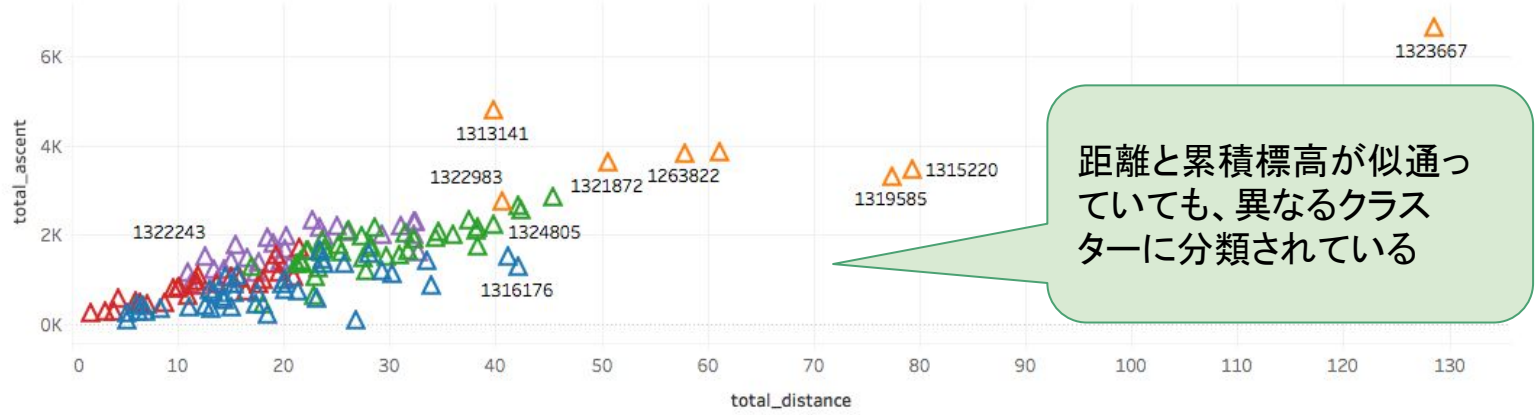
※距離の違いに幅があるため、やや細かく分類したい


クラスター分布　第1主成分×第2主成分
cluster
0
1
2
3
4
PC1
PC2
TOUGH
MIDDLE
EASY
HARD
CHALLENGE
1323667
1315220
1317597
1263822
1313141
1321872
1317595
1318736
1315914
1316176
1318966
1315661
1320892
1321508
1322983
1320425
1322665
1320903
1321610
1318447
1317809
1324591
1323479
1318983
1322243
1325208
1325257
1323044
1319857
1319067
長距離・緩やか
TOUGH
EASY
高山・高獲得標高
低山・高低差小
HARD
短距離・急勾配

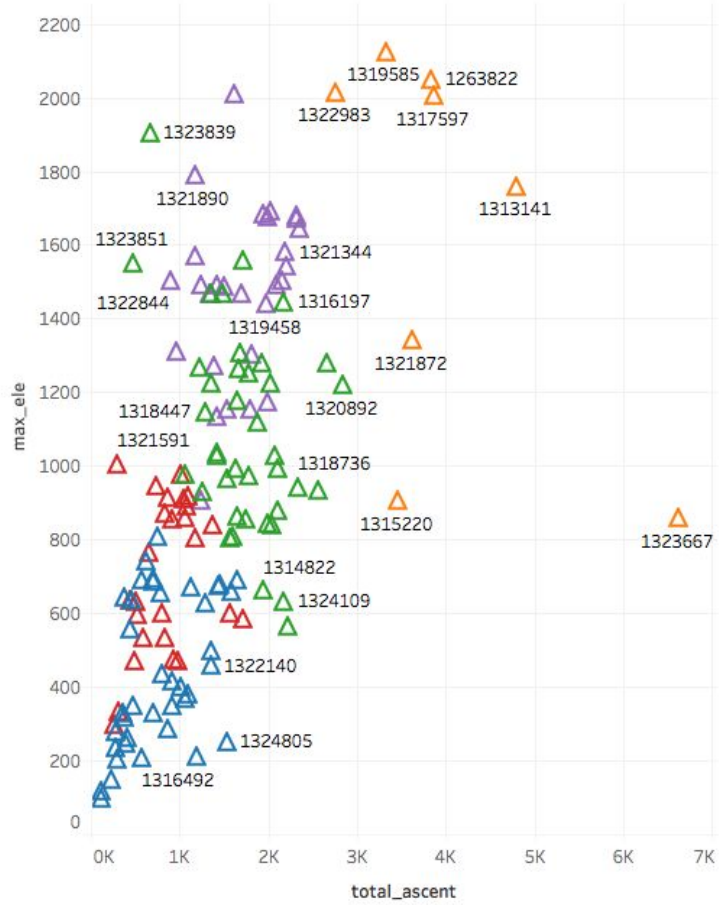
クラスター分布　距離×平均勾配



クラスター分布　距離×累積標高



クラスター分布　累積標高×最高標高



クラスター分布　累積標高×平均勾配

cluster
0
1
2
3
4

EASY
MIDDLE
CHALLENGE
HARD
TOUGH

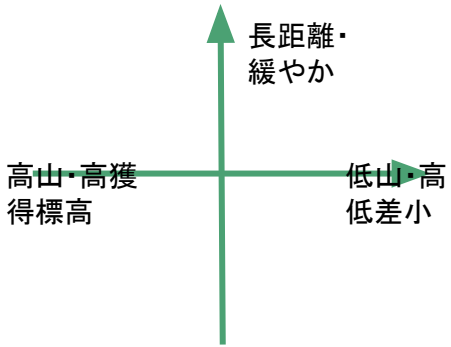
MIDDLEは高山はあるが
勾配は比較的緩やか

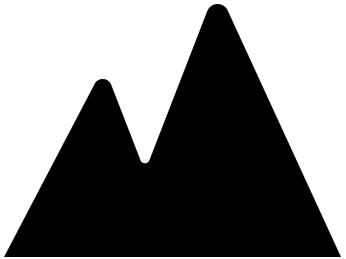
＜4＞レコメンデーション

第1主成分、第2主成分の類似度で協調フィルタリングを実施

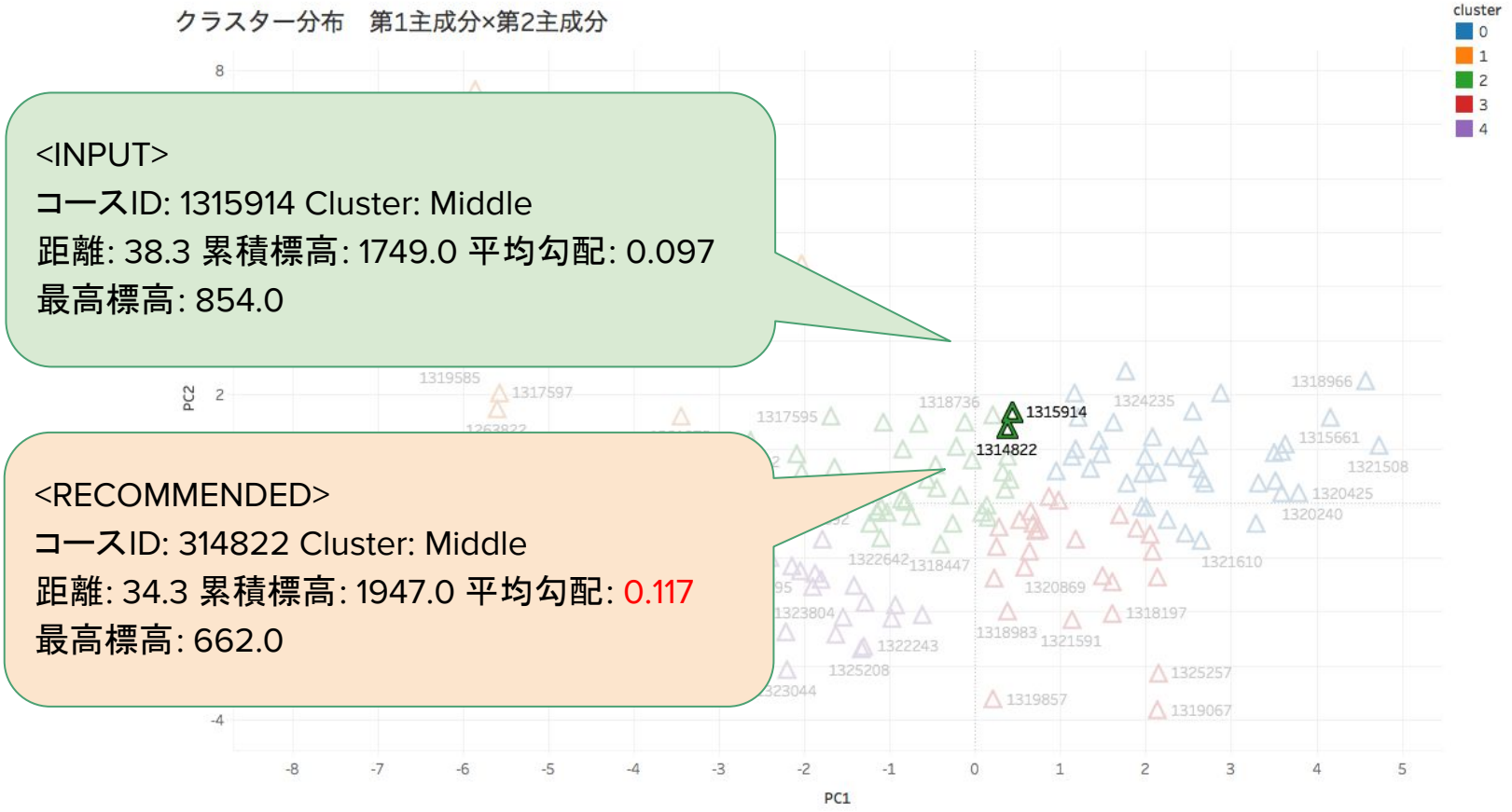
類似度の高いコースから

①場所が近すぎるコースは排除（名寄せしきれていない可能性がある、かつ目新しいコースを紹介するため）
②よりレベルアップした(と想定される)コースを選定する
　└第1主成分は、より一(高山)
　　第2主成分は、>0ならより＋(長距離)
　　　<0ならより一(急勾配）





コースID:1315914と類似したコースのレコメンド例



<INPUT>
コースID: 1315914 Cluster: Middle
距離: 38.3 累積標高: 1749.0 平均勾配: 0.097
最高標高: 854.0

<RECOMMENDED>
コースID: 314822 Cluster: Middle
距離: 34.3 累積標高: 1947.0 平均勾配: 0.117
最高標高: 662.0

活用可能性

・勾配視点のコース紹介は、意外とない

・より急勾配か、長距離か、選択してコースレコメンド

・エリア指定

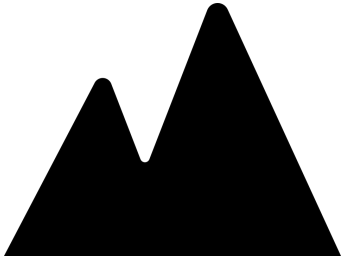
より安心・安全な山行計画を☀

Appendix

＜出典＞

①緯度経度から距離を求める計算式は、ヒュベニの距離計算式（世界測地系）を使用
http://yamadarake.jp/trdi/report000001.html
https://blogs.yahoo.co.jp/qga03052/33991636.html ->Pythonのコードを参考にさせていただきました
http://www.kashmir3d.com/kash/manual/std_siki.htm（参考）

②勾配計算参考
http://tomari.org/main/java/koubai_keisan.html



Appendix

■Rによる PCA結果

```
> summary(result)
Importance of components%s:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC8
Standard deviation     2.3079 1.4842 1.2578 0.68163 0.50616 0.27913 0.20616 0.16215
Proportion of Variance 0.5327 0.2203 0.1582 0.04646 0.02562 0.00779 0.00425 0.00263
Cumulative Proportion  0.5327 0.7529 0.9111 0.95759 0.98321 0.99100 0.99525 0.99788
                           PC9    PC10
Standard deviation     0.13688 0.04945
Proportion of Variance 0.00187 0.00024
Cumulative Proportion  0.99976 1.00000
```