

卒業論文

環境認識ライフログからの行動パターン 解析による類似性・イベント検出

Similarity and event detection by behavior pattern analysis from
environment recognition life log

富山県立大学 電子・情報工学科

1415048 福嶋 瑞希

指導教員 奥原 浩之 教授

平成30年2月6日

記号一覧

以下に本論文において用いられる用語と記号の対応表を示す.

用語	記号
クラスター	X, Y
クラスター内での重心とサンプルとの距離の 2 乗和	$L(X), L(Y)$
クラスターの重心とクラスター内の各サンプルとの距離の 2 乗和	$L(X \cup Y)$
入力データベクトル	x
競合層のニューロンの番号	i
参照ベクトル	m_i
勝者ニューロン	c
勝者ニューロンとの距離によりガウス関数で減衰する係数	h_{ci}
i 番目のニューロンの競合層上での位置	r_i
勝者ニューロンの競合層上での位置	r_c
学習回数	t
学習率係数	$\alpha(t)$
学習半径	$\sigma^2(t)$

目次

記号一覧	1
第1章 序論	5
§ 1.1 本研究の概要	5
§ 1.2 本研究の目的	5
§ 1.3 本論文のあらまし	6
第2章 ライフログとスマートグラス	7
§ 2.1 現状のライフログ	7
§ 2.2 スマートグラス	8
§ 2.3 画像認識 API	10
第3章 行動識別	13
§ 3.1 行動識別	13
§ 3.2 行動識別のための分析手法	14
§ 3.3 類似性・イベント性	19
第4章 提案手法	23
§ 4.1 開発システムの概要	23
§ 4.2 省電力化	24
§ 4.3 周期性の検出	25
第5章 シミュレーション結果ならびに考察	27
第6章 結論と今後の課題	35
謝辞	36
参考文献	37
付録	40
A. 1 ライフログデータ取得アプリケーションのソースコード	40
A. 2 クラスター分析を作成するソースコード	41
A. 3 多次元尺度法を作成するソースコード	46

A. 4 対応分析を作成するソースコード	51
A. 5 共起ネットワークを作成するソースコード	59
A. 6 時系列 SOM を作成するソースコード	73

序論

§ 1.1 本研究の概要

現代、多くの人がスマートフォンやウェアラブルデバイスを持ち歩くことが一般的であり、急速な情報技術の発達から、個人の生活や行動をデータとして取得、記録することが可能となっている。スマートフォンやウェアラブルデバイスを使用して取得したライフログデータは、個人の生活に生かしたり、社会に生かしたりできると考えられている。

スマートフォンやウェアラブルデバイスの全地球測位システム (Global Positioning System, Global Positioning Satellite : GPS) をライフログとして取得、解析するアプリケーションが多く存在し、受容性の高いライフログのための研究が行われている [1]。しかし GPS はライフログデータのなかでも精密な個人情報が含まれるため、不安や嫌悪感が大きく、情報漏えいへのリスクに対する警戒心が強いのが現状であり、技術面とは異なる課題となっている [2]。また、手動でライフログデータを取得するアプリケーションも多く、未だライフログの受容性は改善の余地がある。

個人情報に対する心理的不安、ライフログデータを取得するという物理的負担が少ないライフログは多くの人に広く受け入れられると考える。ライフログ自体が多くの人に広く受け入れられることで、取得するデータ量を増やすことができるため、より個人や社会に生かすことができると考えられる。したがって、ライフログの在り方は改善すべきであると考えられる。

§ 1.2 本研究の目的

本研究は、多くの人に広く受け入れられるライフログとして、個人情報保護に着目し、手間がかからず自動的にライフログデータの取得を行い、取得したデータから類似性やイベント性を考察できることを目的とする。この目的のため、スマートグラスと画像認識を用いたリアルタイム視界情報取得アプリケーションを提案する。このアプリケーションを使用したビッグデータ構築、データ解析を行い、行動パターンの類似性・イベント検出を行

う。また、このアプリケーションの開発には画像認識 API を使用し、デバイスは EPSON MOVERIOBT-300 を使用する。データの解析は、自己組織化マップ (Self-organizing maps: SOM)、階層的クラスター分析、多次元尺度構成法 (Multi Dimensional Scaling: MDS)、対応分析、共起ネットワークを行い、読み取り・比較を行うことでライフログデータの類似性やイベント性を考察する。

§ 1.3 本論文のあらまし

本論文は次のように構成される。

第 1 章：本章 第 1 章では、本研究の概要と目的について説明した。

第 2 章 第 2 章では、現状のライフログ・ライフログアプリケーションの問題や特徴について説明する。また、ウェアラブルデバイスであるスマートグラスの種類や本研究で使用するスマートグラスについての説明、視界情報を取得するための画像認識 API について述べる。

第 3 章 第 3 章では、行動識別についての研究や、行動識別のための分析手法について説明する。また、分析から考えられる類似性やイベント性に説明する。

第 4 章 第 4 章では、本研究の提案手法として開発した視界情報取得アプリケーションについて述べる。また、このアプリケーションを開発する上で、省電力化を行うことについて説明する。開発したアプリケーションを用いて取得したライフログデータの SOM をよりわかりやすくするための工夫についても述べる。

第 5 章 第 5 章では、開発したアプリケーションで取得したライフログデータから多変量解析を行ったうえでの行動パターンの類似性・イベント検出について考察を述べる。

第 6 章 第 6 章では、まとめと今後の課題を述べる。

ライフログとスマートグラス

§ 2.1 現状のライフログ

ライフログ (lifelog) とは、人間の活動 (life) の記録 (log) であり、センサーなどで個人の活動に関するログを取得する行為が、元来のライフログの語源と考えられている。本研究では、この行為をライフログとし、個人の行動履歴に基づいて生み出されるビッグデータのことをライフログデータと呼ぶこととする。また、ライフログに関して、長時間の記録や膨大なデータが必要という定義はない。

ライフログデータを取得・活用できるアプリケーションとして、ソニーモバイルの Xperia 専用アプリ「Lifelog¹」がある (図 2.1 参照)。このアプリケーションはスマートウェア「SmartBand 2²」 (図 2.2 参照) と連携することでどれほど歩いた、走ったかという歩数や心拍数等のライフログデータを取得し、ユーザ自身が健康管理に生かすことができる。また、自動で位置情報をマップにマッピングできる「マッピング - GPS ログまとめて全部記録³」 (図 2.3 参照) や、手動でマッピングする「Swarm⁴」 (図 2.4 参照) というアプリケーションがある。このアプリケーションは行動の記録を取ることができるため、日々の生活や旅行の記録として使用できる。上記のアプリケーションは GPS のアクセス許可が必要であり、上記以外のライフログアプリケーションも GPS を必要とすることが多い。

ライフログに関する既存研究として、スマートフォンから得られる位置情報履歴や写真撮影履歴、ツイートを使用したライフログデータから行動特徴抽出・イベント検出を行う研究が行われている [3] [4]。取得したライフログデータの解析を行うことで、ユーザー自身の健康管理や学習 [5] に生かすだけではなく、ビジネスとしてターゲティング広告に生かすこともできる。ライフログは、様々な視点からライフログデータの比較を行うことで、個人や社会に利用できるという価値があると考えられる。

しかい、現状のライフログには、大きくわけて二つの問題があると考えられる。一つ目

¹<http://www.sonymobile.co.jp/myxperia/app/lifelog/>

²<http://www.sonymobile.co.jp/product/smartproducts/swr12/>

³<https://play.google.com/store/apps/details?id=org.liteapp.mat2>

⁴<https://play.google.com/store/apps/details?id=com.foursquare.robin>

はライフログの個人情報問題，二つ目はライフログの煩雑問題であると考えられる，

ライフログの個人情報問題

ライフログデータとして主に用いられることが多いのは，GPS であると考えられる．GPS を用いることで，正確な位置情報を取得することができるため，いつ，どこに，どれくらいいるのかという情報をライフログデータに含むことができる．また，同時にツイートやその場で撮影した写真を取得することで，どんな行動を行っているか推測することができる．正確なライフログデータを取得できる反面，GPS の情報がネット上でどのように扱われているかユーザーは把握できず，一度情報が漏えいしてしまうと個人が特定されてしまうというリスクがある [6]．このような，GPS の含まれたライフログデータという個人の活動に関するログは，個人を特定することが安易であるため，個人情報の取り扱いに伴う義務が生じプライバシーの侵害という問題を引き起こす [7]．

ライフログの煩雑問題

ライフログアプリケーションの中には，意識的にライフログデータを取得しなければならないアプリケーションが存在する．このようなアプリケーションはライフログのために，ユーザーが自ら位置情報をマッピングしたり，食事風景の写真をとることを意識しなくてはならない [8]．ユーザーの主観的なライフログデータを取得できるが，ライフログデータを取得するのに手間がかかってしまうという問題を引き起こす．

また，ライフログの個人情報問題に関して，株式会社N T Tデータ経営研究所が2016年に10代から60代の男女1059人を対象として実施した「パーソナルデータに関する一般消費者の意識調査 [9]」という調査がある．この調査において，「企業のマーケティング等の利用目的にて，パーソナルデータを企業に提供しても良いと思うデータの条件」において，金銭や商品を受け取ることができたり，個人が特定できない状態でも，どのような条件であっても位置情報は提供したくないという人が66.2%であり，過半数以上を占めていることがわかっている．

ライフログの個人情報問題，煩雑問題という二つの問題に対し，ライフログデータを収集する上で重要であることは，可能な限り不安要素を取り除くことと，手間をかけず無意識でライフログデータを残すことであると考えられる．GPSを使用せず，自動でライフログデータを残すことを可能にすることにより，誰でもプライバシー侵害の不安や負担のないライフログを可能にする．

§ 2.2 スマートグラス

近年，スマートフォンやタブレットなどのスマートデバイスが急速に普及している．その次のデバイスとして期待されているものがウェアラブルデバイスである．ウェアラブルデバイスとは，体に装着して利用するコンピュータデバイスの総称であり，スマートグラスはメガネ型のウェアラブルデバイスである．代表的なものとして，Glass Enterprise Edition⁵」

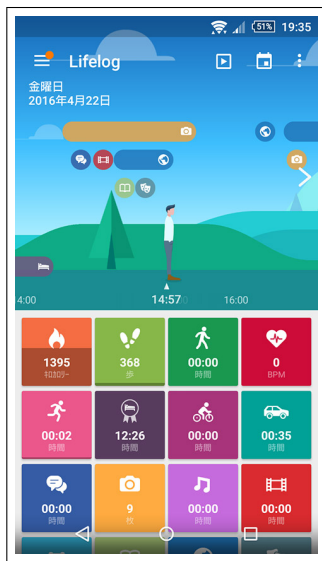


図 2.1: Lifelog



図 2.2: SmartBand 2

(図 2.5 参照), SmartEyeglass⁶ (図 2.6 参照) などが挙げられる。

このようなスマートグラスは把持の必要がなく、常に目の前に仮想画面を表示可能である [10] ため、両手を常に開けておくことが可能である。手をあまり使うことなく欲しい情報を提示することができ、また、外部から見ているものを知られることなく情報を活用できる [11]。さらに、ユーザーが見ている実際の景色に必要な情報を重ねて表示することができるため、拡張現実技術との親和性も高い。

このように、スマートグラスを使用すると、ユーザーがあまり意識する事なく、常時画像の取得を行える。この利点を生かし、ユーザーに負担をかけることなくライフログデータを取得できる。

本研究では、スマートグラスとしてセイコーエプソン社のシースルーモバイルビューアー MOVERIO BT-300 を使用する (図 2.7)。使用する理由として、スマートグラスの中でも比較的安価であり、Android アプリケーションを作成して動作できるためである。MOVERIO はユーザーが見ている現実空間に対してコンピュータが生成した仮想オブジェクトを重畳表示するというシースルー表示を行う。シースルー表示の実現にはビデオシースルー方式と光学シースルー方式の 2 通りがあり、MOVERIO は光学シースルー方式を使用することが可能である [10]。ビデオシースルー方式は、カメラ画像とコンピューターグラフィックス (CG) を合成した画像を表示する方式である。一方、光学シースルー方式は肉眼の視界に対して CG を重畳する方式であるため、視界が広く、移動中の使用や現実の物体を用いた作業時の使用に適している。

⁵<https://www.x.company/glass/>

⁶<https://developer.sony.com/ja/develop/smarteyeglass-sed-e1/>

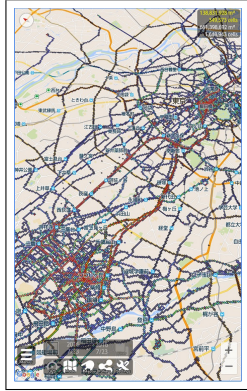


図 2.3: マッピング-GPS ログまとめて全部記録

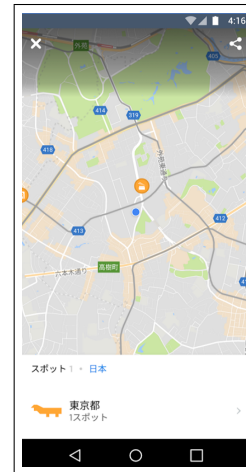


図 2.4: Swarm



図 2.5: Glass Enterprise Edition



図 2.6: SmartEyeglass

§ 2.3 画像認識 API

画像認識技術とは、コンピュータに画像を理解をさせる技術である。画像内のピクセル信号のパターンから意味を抽出するパターン認識により、人間の視覚機能をコンピュータに処理させることができる。画像認識技術を用いることで、画像が持つ情報をテキストで取得することができる。代表的な画像認識 API に、Google Cloud Vision API と、Computer Vision API, IBM Bluemix Alchemy API という三種類がある。画像認識 API を使用することで、個人だけでは取得が難しい大量のデータを利用することができるため、スマートフォンやウェアラブルデバイスで取得したカメラ画像を認識するアプリケーションの開発が可能となる。



図 2.7: MOVERIO BT-300

Google Cloud Vision API

2016 年に一般公開された画像認識クラウドサービス Google Cloud Vision API⁷は、写真の被写体を機械判定し、ラベリングする機能をもっている。Google Cloud Vision API を用いて個々の写真の情報（ラベル）を取得できるが、撮影内容が不明瞭のときは、1 つも得られないこともあり [12]、同じ単語を複数個返して来る場合もある。初年度無料である。

Computer Vision API

Microsoft 社が提供している API の一つである Computer Vision API⁸は、写真画像の被写体を機械判読し、ラベリングや画像内のテキストの判読など多様な機能を有している。ラベリングだけでも tags や captions という機能を有する。tags は、画像内の要素を、2,000 以上の認識要素、生物、風景などに基づいて、タグ情報を算出する [13]。captions は文章で人間が読める言語として要約を表示する。初年度無料である。

IBM Bluemix Alchemy API

Watson が提供している API の一つである IBM Bluemix Alchemy API⁹は、画像に対してタグ付けを行うことができる。キャプションは出力できないが、分類結果の階層構造に強く、食べ物の分類などに特化している。Lite コースは無料である

⁷<https://cloud.google.com/vision/>

⁸<https://azure.microsoft.com/ja-jp/services/cognitive-services/computer-vision/>

⁹<https://www.ibm.com/watson/jp-ja/developercloud/visual-recognition.html>

行動識別

§ 3.1 行動識別

携帯電話やウェアラブルデバイスを用いて、ユーザーが今何を行っているかという行動をライフログデータとして取得し、取得したライフログデータの解析から行動を認識することを行動認識や行動識別という。本研究では、行動識別と呼ぶことにする。

既存研究には、携帯電話の加速度センサやGPSを用いてライフログデータを取得し、走行や歩行しているなどの行動識別を行う研究 [14] や、ウェアラブルデバイスの加速度センサやGPSを利用する事で人の行うさまざまな行動を取得し、行動識別を行う [15] 研究がある。しかし、本を読んでいることであったり、料理をしていることなどの細かい動作をライフログデータとして取得することは難しい。細かい動作をライフログに組み合わせるため、手動で動作の開始・終了を記録するアプリケーション「行動の記録 (LifeLog)¹⁰(図 3.1 参照)」や、机上に設置した KinectTM(図 3.2 参照) を用いて机上の細かい動作を認識する研究がある。しかし、この研究は机上に限っているため屋外や机上以外の行動は認識できない [16]。なお、KinectTM は 2017 年 10 月 25 日に生産終了が公表されている。

本研究ではGPSやKinectTMを使用せず、細かい行動をライフログデータとして取得する手法として、ユーザーの視界情報を取得する。視界情報を取得することで、視界上にある物体からどのような作業を行っているのか、視界の風景から室内か室外にいることなどを判断できるため細かい行動をライフログデータとして取得することが可能となる。しかし、ユーザーの視界情報を取得すると、GPSを使用しなくても、どこで何をしているのか、誰と会っているのかなど必要以上のライフログデータを取得してしまう。この結果、ユーザーやユーザーの視界にいる第三者のプライバシーを侵害してしまう。さらに、視界情報を画像や動画で蓄積するとデータ量が多くなるという問題が生じる。この問題に対し、ウェアラブルデバイスのカメラ画像を取得し、減色処理を施しデータの蓄積・解析を行う研究が行われている [17]。本研究では MOVERIO のカメラ画像を取得し、画像認識によりカメラ画像をテキストに変換することで、データ量を削減、かつプライバシーに配慮したライ

¹⁰<https://play.google.com/store/apps/details?id=com.yoko.tama.workLog&hl=ja>



図 3.1: 行動の記録 (LifeLog)

フログデータの取得を検討する。



図 3.2: Kinect™

§ 3.2 行動識別のための分析手法

本研究ではいくつかの解析手法を用いて、一定時間内の取得データを視覚的に表し、行動識別を行う。そのために、多変量解析である SOM, 階層的クラスター分析, MDS, 対応分析, 共起ネットワークを用いてテキストデータの可視化を行う。

多変量解析を行うツールとして KH Coder¹¹を使用する。KH Coder とは、テキスト型データの計量的な内容分析、もしくはテキストマイニングのためのフリーソフトウェアである [18]。無償でウェブサイトから入手でき、すべての機能をマウス操作で利用できる。また、どんな言葉が多く出現していたのかを頻度表から見るができたり、SOM, 共起ネットワークなどの多変量解析を行ったりできる [19]。KH Coder を用いて行われた研究としては、アンケートの自由回答項目・新聞記事・インタビューデータなどさまざまなデータを分析した事例がある [20]。本研究では Version 3.Alpha.11 を使用する。また本章ではチュートリアル用に KH Coder から提供される「坊ちゃん」英語版テキストデータを用いて解析を行う (図 3.3 参照)。この「坊ちゃん」英語版テキストデータは KH Coder をダウンロードする際に同時に取得することが可能であり、KH Coder のホームページよりダウンロード¹²後 C:\khcoder3\tutorial_en にテキストファイルとして保存されている。

KH Coder をダウンロードする際に取得できる KH Coder3 リファレンス・マニュアルによると、KH Coder を使用した多変量解析には、まず対象のテキストファイル (もしくはエクセルファイル) を読み込む事から始める。読み込むテキストファイルに事前に手動で h1 タグや h2 タグという見出しタグを設定することで、見出しごとの解析も可能である。この時 Stop words として Be 動詞のような一般的な語を指定しておく、分析対象から外すこ

¹¹<http://khc.sourceforge.net/>

¹²<http://khc.sourceforge.net/dl3.html>

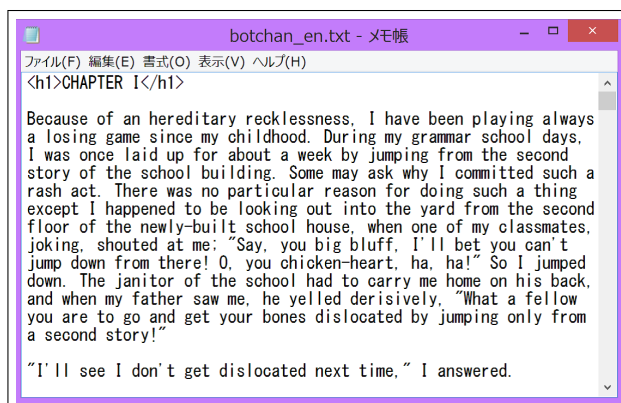


図 3.3: 「坊ちゃん」英語版テキストデータの一部

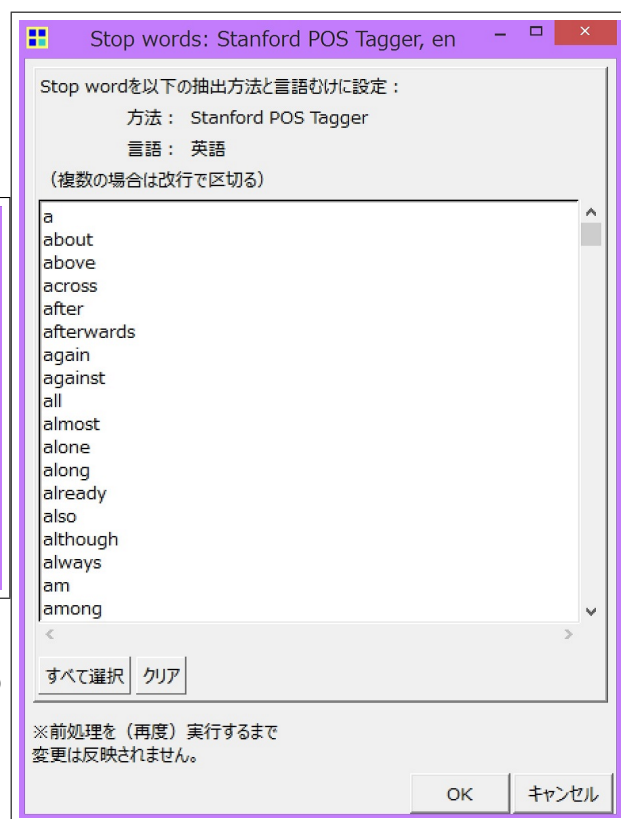


図 3.4: Stop words の一部

とができる（図 3.4 参照）。この Stop words 一覧テキストファイルは、KH Coder をダウンロードする際に同時に取得することが可能であり、解析を行う際に必要でない単語を自由に追加できる。

次に、前処理として、POS Tagger¹³を使用して自然言語処理を行う（表 3.1）。前処理を行うことで、多変量解析に使用する「各文書に、それぞれの語が何度出現していたのか」という集計表である「文書×抽出語」表 [21](表 3.2 参照)を出力することができる。h1 から h5 というのは見出し番号であり、h1 タグや h2 タグが存在すると 1 増加する。dan は段落番号で、bun は文番号である。id は文書の通し番号で、リセットされることは無い。length_c は文書の長さを文字数で表し、length_w は文書の長さを語数で表したものである。

まず、ライフログデータの内容解析のため、階層的クラスター分析、MDS、対応分析、共起ネットワークを行う。この時テキストデータは、抽出語の中でも、50 回以上出現する 35 語の抽出語を用いる。理由として、出力される抽出語が多すぎると解析結果の読み取りが難しくなるためである。

階層的クラスター分析は、抽出語の最も似ている組み合わせから順番にクラスターにしていく方法であり、デンドログラムを表示する [22]。指定されたクラスター数に全体を分割し、その結果を色分けによって表示する。なお、KH Coder では、デフォルトの Auto では、抽出語数の平方根を四捨五入したものをを用いている [23]。1つのクラスターには関連性が高い抽出語が集まっているため、クラスターごとに集まっている抽出語を調べることで

¹³<https://nlp.stanford.edu/software/tagger.html>

表 3.1: 「坊ちゃん」データを用いて KH Coder で出力した抽出語の一部

Noun		ProperNoun	
school	130	Red	171
room	119	Shirt	163
teacher	119	Porcupine	128
house	108	Clown	85
time	95	Kiyo	73
fellow	88	Tokyo	47
day	84	Hubbard	46
student	84	Squash	46
way	78	Badger	32
night	70	Madonna	28
head	65	Koga	23
face	64	Sir	21

表 3.2: 「坊ちゃん」データを用いて KH Coder で出力した「文書×抽出語」表の一部

h1	h2	h3	h4	h5	dan	id	length_c	length_w	school	room	teacher	house	time
1	0	0	0	0	1	1	650	177	4	0	0	1	0
1	0	0	0	0	2	2	49	16	0	0	0	0	1
1	0	0	0	0	3	3	203	51	0	0	0	0	0
1	0	0	0	0	4	4	43	15	0	0	0	0	0
1	0	0	0	0	5	5	207	58	0	0	0	0	0
1	0	0	0	0	6	6	577	147	1	0	0	1	0
1	0	0	0	0	7	7	853	216	0	0	0	0	0
1	0	0	0	0	8	8	800	193	0	0	0	1	1
1	0	0	0	0	9	9	155	42	0	0	0	0	0

テキストデータ全体における文書の傾向や特徴を知ることができる。階層的クラスター分析の作成方法は、まず抽出語として A,B,C,D があったとする。この時抽出語の中で最も距離の近い組み合わせを A と B とし、A と B をくくり、2 点の代表点を求める。次に、AB の重心、C、D の 3 点で、最も距離の近い組み合わせを見つける。このとき C と D が最も近いとすると、C と D をくくる。このように繰り返していくことで、デンドログラムを作成する [22] [24]。

KH Coder3 リファレンス・マニュアルによると、クラスター間の距離測定方法として、KH Coder ではワード法を使用している。2つのクラスター X,Y を結合したと仮定したとき、それにより移動したクラスターの重心とクラスター内の各サンプルとの距離の 2 乗和 $L(X \cup Y)$ と、もともとの 2つのクラスター内での重心とそれぞれのサンプルとの距離の 2 乗和 $L(X)$, $L(Y)$ の差が最小となるようなクラスターどうしを結合する手法である。ワード法は、計算量は多いが分類感度が高いため用いられることが多く、ワード法は一つのクラスターに抽出語が順に吸収され類似するクラスターが形成される鏡効果が起こりにく

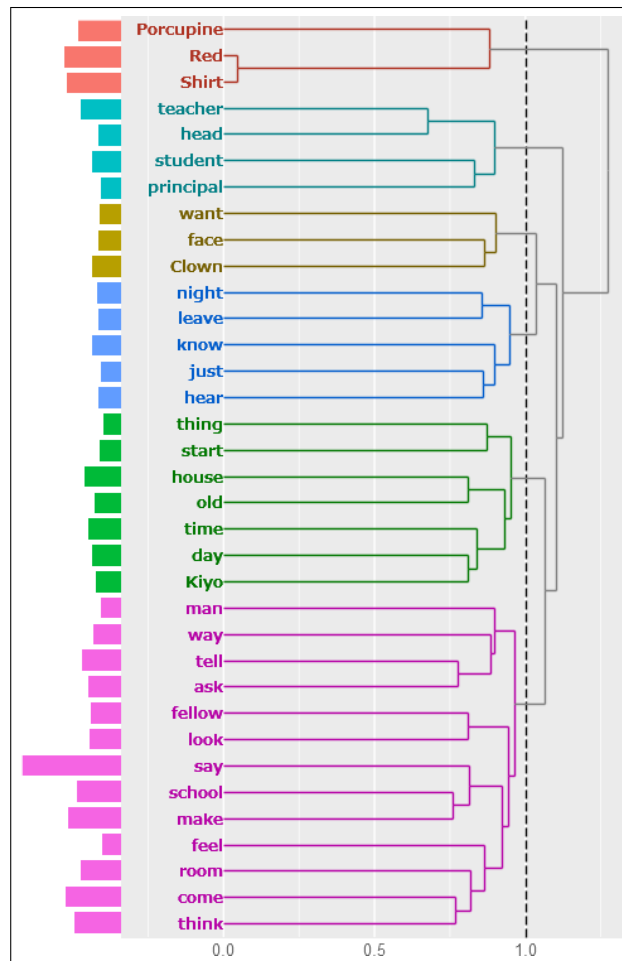


図 3.5: 「坊ちゃん」データから作成したクラスター分析

いという強みがある [25] [26].

$$\Delta = L(X \cup Y) - L(X) - L(Y) \quad (3.1)$$

クラスター分析の結果を図 3.5 に示す. この時クラスター数を Auto にしたためクラスター数は 6 となる. 併合標準 (図 3.6 参照) からクラスター数が 6 であることは妥当だと考えられるためクラスター数は 6 とした. クラスター分析から, 最も多く出現している say という単語があるクラスターに school という単語がある点から, 学校で何かを話す場面が多いのではないかと推測できる. また, teacher と student が同じクラスターにある点からやはり学校が重要ではないかと推測できる. クラスター内やクラスターどうしの比較からデータ内で重要な語の関係を推測できる.

MDS は, 抽出語間の関連性や類似性の強さをマップ上の点と点の距離に置き換えて, 相対的な関係性を視覚化する手法である [27]. KH Coder では MDS の中でも最も広く利用されてきた Kruskal の非計量 MDS を使用している [28]. また, 語と語の関連を見るために Jaccard 係数を使用している. Jaccard 係数とは二文章間の類似度であり, 「語 A を含む」かつ「語 B を含む」文書の数, 「語 A を含む」または「語 B を含む」どちらかでも当てはまる文書の数で割った係数である [29]. MDS の結果は, 相対的な位置関係だけを表わしてい

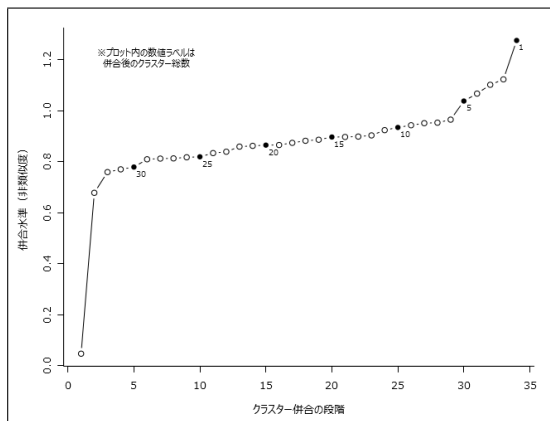


図 3.6: クラスター分析の併合標準

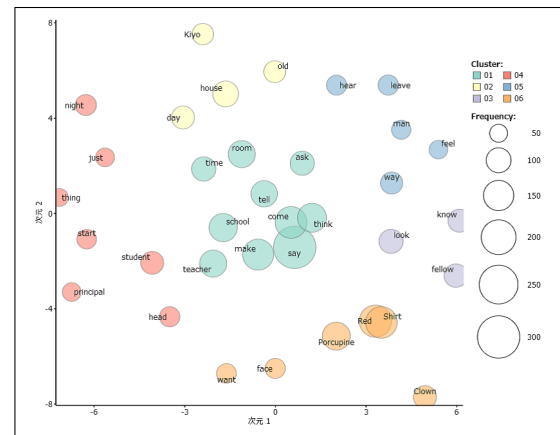


図 3.7: 「坊ちゃん」データから作成した MDS

るため軸の方向性に意味はない。

$$\text{Jaccard 係数} = \frac{\text{語 A と語 B を含む文書}}{\text{語 A もしくは語 B を含む文書}} \quad (3.2)$$

図 3.8 は坊ちゃんデータから出力した MDS である。クラスター分析を参考にし、クラスター数は 6 に設定した。この結果、目立つものはクラスター 01 であり、どのクラスターからも同じような距離であることから、データの中で中心的なクラスター・抽出語であることがわかる。クラスター 02 と 06 のように離れて配置されるクラスターもあり、関係性の低いクラスター・抽出語であることがわかる。

対応分析は、単純な 2 次元表や多重表の行と列間の対応する測定値を分析する探索的データ解析の手法であり、分析結果として、2 次元のマップが表示される [30]。このマップで近くに位置しているものは、相対的に関連が強いということを示し、遠くに位置しているものは関連が弱いということになる。また、対応分析では、これといって特徴のない語が原点付近に密集することが多い。

図 3.9 は坊ちゃんデータより出力した対応分析である。この対応分析から、think や say という行動は特徴的ではなく、データ全体によく出現することがわかる。一方で、Red Shirt や Clown は原点から遠く離れているため、特徴的であることがわかる。

共起ネットワークはある語が語られる状況の断面を多角的に把握するのに強力な解析手法であり、線がつながっている語が共起関係にあり、その繋がりにのみ着目する [31]。抽出語の中で出現パターンの似通ったものを線で結ぶネットワークであり、すなわち共起関係を線で表したネットワークである。MDS と異なっている点は、プロットされた位置ではなく線で結ばれているかどうかということに意味がある点である。

また、KH Coder3 リファレンス・マニュアルによると、KH Coder では、共起ネットワーク図の表し方として複数用意されており、それらの中から選択できる。種類は、中心性（媒介）、中心性（次数）、中心性（固有ベクトル）、サブグラフ検出（媒介）、サブグラフ検出（random walks）、サブグラフ検出（modularity）の 6 種類がある。この時の中心性が高い語とは、語がデータ中で重要な役割を果たしている可能性がある語である [32]。サブグラフ検出とは、比較的強くお互いに結びついている部分を自動的に検出してグループ分けを

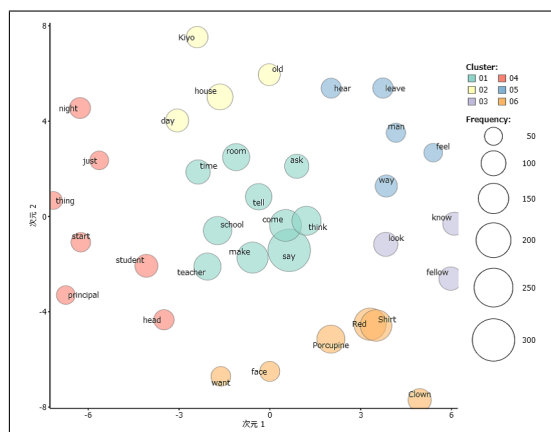


図 3.8: 「坊ちゃん」データから作成した MDS

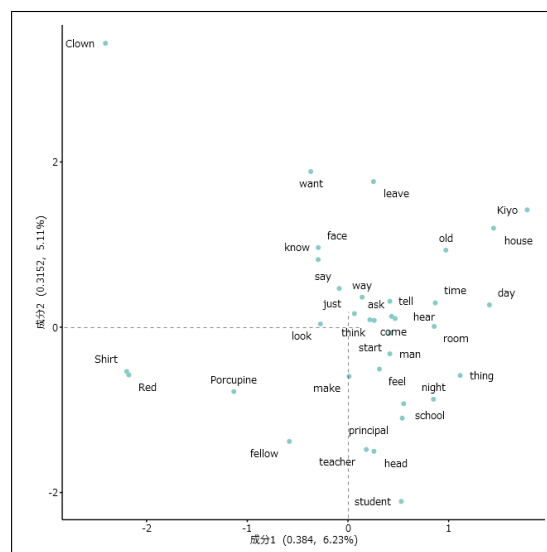


図 3.9: 「坊ちゃん」データから作成した対応分析

行い、その結果を色分けによって示す方法 [33] であり、抽出語同士の関係性が強い、つまり Jaccard 係数の値が高い抽出語の集まりである。本分析では、各抽出語の関係性を確認するため、KH Coder の出力する共起ネットワーク図の中から「サブグラフ検出（媒介）」を使用する [34]。

坊ちゃんデータを使用して出力した共起ネットワークが図 3.10 になる。この時、Jaccard 係数が 0.2 以上の共起関係を描画している。Jaccard 係数が小さいほど類似度が低いものも含まれ、大きいと類似度が大きいものしか描画されないため状況に応じて検討すべきである。共起ネットワークより、school は teacher や student と共起関係があり、make や say という動詞とも関係性があることがわかる。room は come や think などの動詞と関係性があり、Red と Shirt は関係性があることが、視覚的に理解しやすくなっていると考えられる。

§ 3.3 類似性・イベント性

本研究では、ライフログデータの内容解析のため、階層的クラスター分析、MDS、対応分析、共起ネットワークを行った後、データの時系列を SOM を用いて解析を行う。SOM を用いて、テキストデータの時系列を可視化することでテキストデータの類似性を検出する。

SOM とは、ヘルシンキ大学のコホーネン教授により 1981 年頃に発表された、教師なし学習を行なうニューラルネットワークの代表例と言える解析手法である [35]。ニューラルネットワークとは、脳機能に見られるいくつかの特性を計算機上のシミュレーションによって表現することを目指した数学モデルである。つまり、人間が無意識にやっていることを機械にやらせるということである。

SOM は図 3.11 に示すように入力層と出力層の 2 つに分かれて競合学習を行う [36] [37]。図 3.11 には、入力層のニューロンが複数個あるが、各々のニューロンがそれぞれの次元に対応した出力を行っていると考えられる。入力データベクトルと呼ばれる入力層から出力層へ

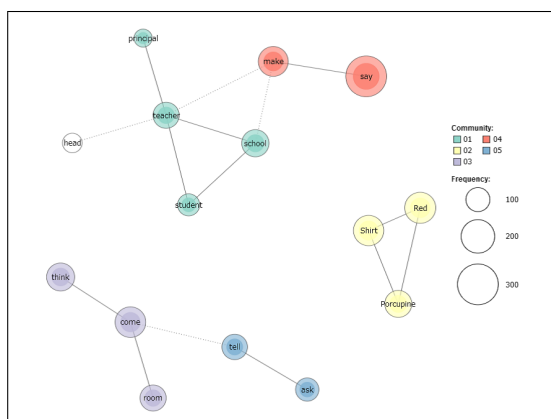


図 3.10: 「坊ちゃん」データから作成した共起ネットワーク

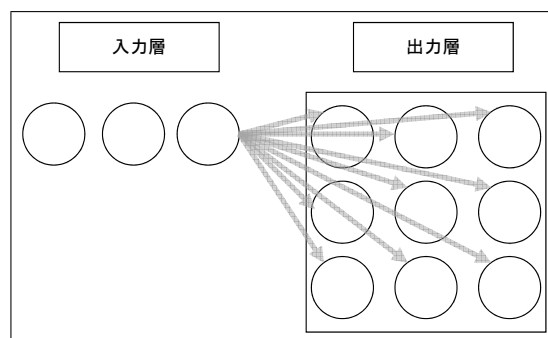


図 3.11: 入力層と出力層

の入力を x と定義し，出力層のニューロンと入力層のそれぞれのニューロンとの結合強度は総称して参照ベクトルと呼ばれ， i を出力層のニューロンの番号とすると， m_i で表される．まず初めに， m_i の初期化を行い，入力データベクトル x を選び，入力データベクトルと各ニューロンの参照ベクトルとのユークリッド距離で出力層のニューロンを競合させる．勝者ニューロンを c とすると，式 3.3 で表される． $\arg \min f(a)$ は $f(a)$ を最小にする a の集合であり，下側に変数がつく値の範囲を書くことが多い．

$$c = \arg \min_i \{ \|x - m_i\| \} \quad (3.3)$$

次に，勝者ニューロンと勝者ニューロンに近いニューロンは自らの参照ベクトルと入力データベクトルを近づける学習を行うため，参照ベクトルを同様に更新させる．この時， h_{ci} は勝者ニューロンとの距離によりガウス関数で減衰する係数である．

$$m_i(t+1) = m_i(t) + h_{ci}(t) \cdot \{x(t) - m_i(t)\} \quad (3.4)$$

$$h_{ci} = \alpha(t) \cdot \exp \frac{-\|r_c - r_i\|^2}{2\sigma^2(t)} \quad (3.5)$$

また，SOM 作成過程ではユークリッド距離を利用している．また，KH Coder3 リファレンス・マニュアルによると，文書の長さのばらつきに左右されない形で計算を行うために，文書中における語の出現回数をそのまま使うのではなく，1,000 語あたりの出現回数に調整したものを計算に使用している．

KH Coder3 リファレンス・マニュアルによると，KH Coder の SOM の学習は，大まかな順序づけを行う段階と，微調整を行う収束段階の 2 段階で行われる．KH Coder では，1 段階目が 1,000，2 段階目が「全体のノード数を 500 倍した数値」に設定されており，全体のノード数が 40 の場合は 200,000 回である．また，各ノードがもつベクトルをワード法で分類してクラスター化する，クラスター数は任意に決定できるため，クラスター分析などの結果からクラスター数を調整する．

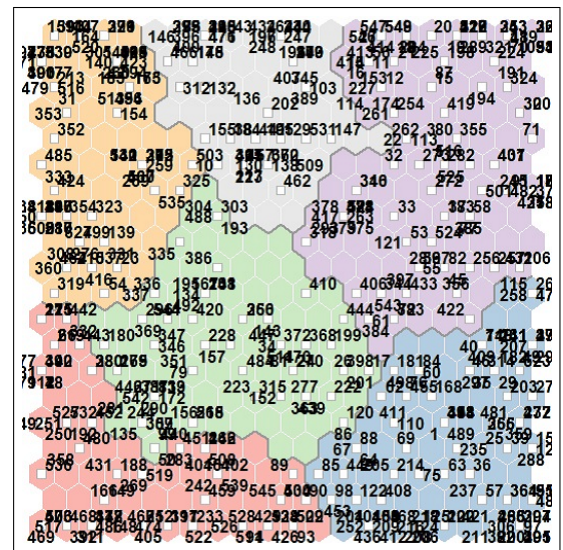
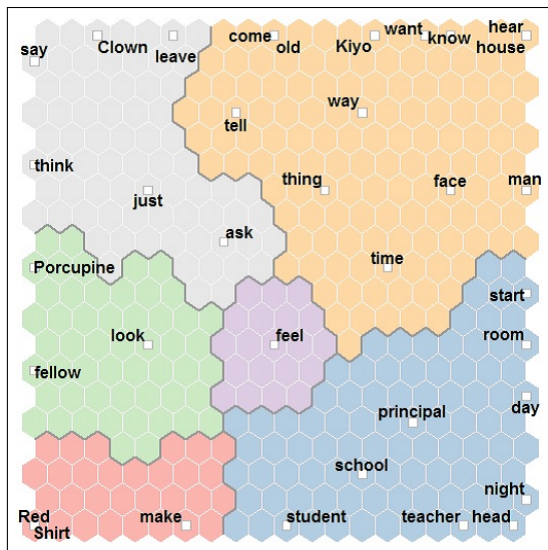


図 3.12: 「坊ちゃん」データから KH Coder 図 3.13: 「坊ちゃん」データから R で作成した SOM
で作成した SOM

データA	データB	データC
- 勉強する - 食事する - 散歩する	- 勉強する - 食事する - 読書する	- 運動する - 運転する - 掃除する

図 3.14: データ A, データ B, データ C の例

本研究ではこの KH Coder はデータの前処理段階に使用し、実際の SOM 作成には、データ解析・グラフィックス環境を備えたオープンソースのソフトウェアである R を使用する。KH Coder で出力できる SOM（図 3.12 参照）は抽出語どうしの関係を示すものとなっているため、今回のライフログデータの解析に用いることには向かないためである。

KH Coder で出力できる SOM の R ファイルを基にソースコードを書き換え、R で出力を行う。出力した SOM が図 3.13 になる。この時 SOM 上の数字は id であり、文書同士の関係が表されている。学習回数は 1 段階目が 1,000、2 段階目が 200,000 となっている。また、クラスター数は 6 とした。

図 3.13 より、文書どうしにまとまりは少ないことがわかる。これは文書数が多いことと、対象としているデータが文学作品であることから似たような文章が並ぶことが少ないためであると考えられる。

KH Coder と R を用いてライフログデータであるテキストデータの変量解析を行い、解析結果の読み取り・比較を行うことで一定時間内の行動識別を行い、ライフログデータの類似性やイベント性を検出できると考える。

本研究では、ライフログデータの類似性とは、多変量解析によるクラスターの分かれ方やクラスターを構成する抽出語から導き出せる行動や、プロットの関係性、SOMであらわされる時系列が類似している場合類似性があると考えられる。また、ユーザー自身の複数のライフログデータの中で類似性のあるライフログデータが多ければ、そのライフログデータは平常日を表していると考えられる。一方でライフログデータの類似性ではなく、ライフログデータから特徴的なイベント性を検出した場合、イベント日であることを検出できたり、平常日とは違うという危険を察知したりできる。

階層的クラスター分析、共起ネットワークはクラスターの分かれ方やデータを構成する単語の関係性からどのような行動があるのかがわかり、このとき予想できるクラスター数はSOMにも利用できる。MDS、対応分析はプロット点やプロット間隔から類似する行動やイベント性のある行動がわかる。

SOMはクラスターの分かれ方や時系列を表すプロット順を追っていくことで行動パターンを識別し、多くのライフログの中でも類似したライフログか、特徴的でイベント性のあるライフログかわかる。

図3.14はライフログデータとして、データA、データB、データCがあり、行動識別により各データに三つの行動が存在していることが検出できた場合を表している。この時、データAとデータBは類似性があるといえる。一方でデータA、データBと、データCは類似性がないといえる。この三つのデータが一人のユーザーのライフログデータであれば、平常日とイベント日の比較に利用できる。もし、三つのデータがバラバラのユーザーである場合、ライフログの類似性があるユーザーどうしでコミュニケーションを促進することができたり、ライフログの類似性がないユーザーどうしの比較を行うことで行動の中で改善すべき行動を検出できたりする [38]。このようにライフログデータの類似性やイベント性の検出は様々な応用が可能であると考えられる。

提案手法

§ 4.1 開発システムの概要

本研究で開発するシステムは，アプリケーションを用いたデータ取得部と，多変量解析によるデータ解析を用いた行動識別部で構成される．図 4.1 はシステムの全体図である．まず，データ取得部について提案をする．

本研究では個人情報保護に着目したライフログのため，MOVERIO と画像認識 API を用いたリアルタイム視界情報テキスト変換アプリケーションの開発を行う．開発エンジンは，Unity Technologies が提供するゲーム制作向け開発エンジン Unity5 を使用する．Unity5 は 3D オブジェクトを主として扱い，モバイル端末への出力にも対応している．画像認識 API は Computer Vision API を使用し開発を行う．MOVERIO は Android5.1 であるため API level22 で Android アプリケーションを作成する．

図 4.2 はライフログデータ取得アプリケーションのフローチャートである．起動した際画面は真っ暗であり，カメラを起動しても画面に何も表示を行わないようにしている．MOVERIO は黒い画面は透過する性質があるため，視界を妨げずにライフログデータ取得を行える．本研究では，データが取得できているか常時確認を行うため，取得したタグを邪魔にならない程度の大きさで表示を行うプログラム（ソースコード A.1 参照）を使用している．

カメラ画像を取得すると，画像認識 API へ送信する．画像認識 API を通じて，カメラ画像の情報が JSON データで取得できる．この JSON データに取得時の年月日と時刻を追加して，テキストデータとして MOVERIO 内に保存される．テキストファイルは「long_report_yyyymmdd.txt」という名前で保存され，MOVERIO 内に同じ名前のテキストファイルがなければ新しくテキストファイルを作成し，テキストデータを保存する．同じ名前のテキストファイルがある場合，そのテキストファイルの最後の行にテキストデータを追加し保存する．

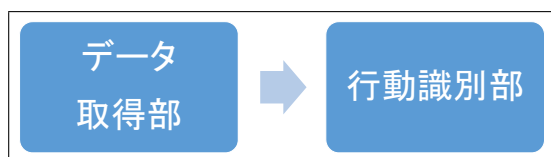


図 4.1: システム全体図

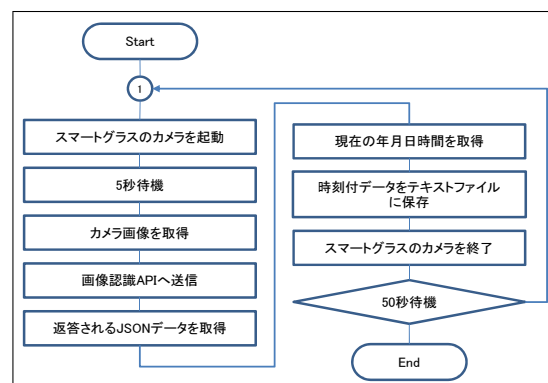


図 4.2: アプリケーションのフローチャート

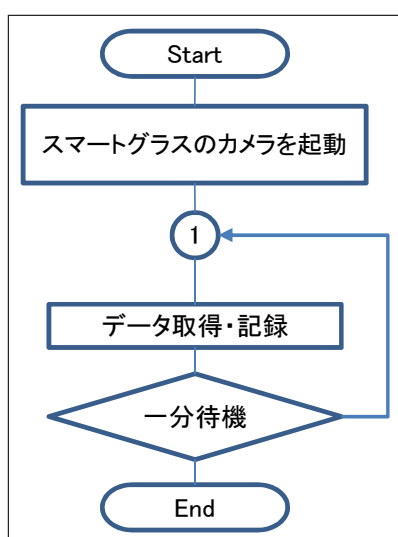


図 4.3: 省電力化前

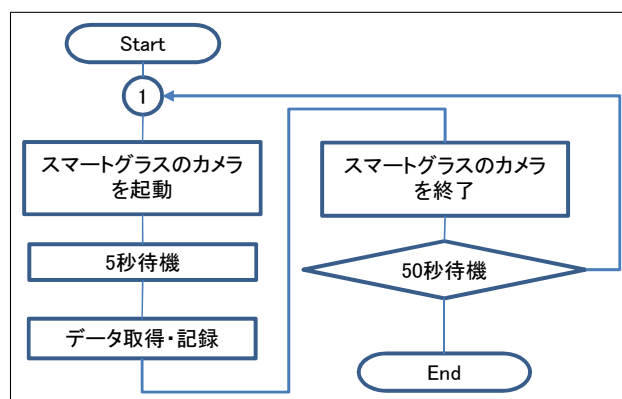


図 4.4: 省電力化後

§ 4.2 省電力化

開発したアプリケーションのバッテリー消費に関しての工夫について述べる．図 4.3 は、スマートグラスのカメラ機能を起動させたまま 1 分ごとにデータ取得を続けるアプリケーションのフローチャートである．この時、カメラを起動させたままだと 2 時間程度でスマートグラスのバッテリーがなくなってしまう．なお、MOVERIO の標準的な駆動時間は約 6 時間¹⁴となっている．充電不可能な外出先でのデータ取得のため、少しでも稼働時間を伸ばす必要がある．

この問題に対し、本研究ではカメラの起動・終了にかかるバッテリー消費よりも、連続起動の方がバッテリー消費が大きいと考え、データ取得後にカメラ機能を終了するようにプログラムに組み込んでいる（図 4.4 参照）．カメラ終了をプログラムに組み込むことにより、3 時間から 3 時間半程度稼働することができた．

約一分おきにデータを取得するために、カメラの休止時間は 50 秒とし、カメラを立ち上

¹⁴<http://www.epson.jp/products/moverio/bt300/spec.htm>

げてから5秒後に撮影を行う。理由として、カメラを起動するのに少なからず時間がかかるため、起動後すぐに撮影を行いカメラ画像を取得することは難しいためである。また、その後画像認識 API の応答を得るまでおよそ5秒程度かかるため、約一分ごとにデータを取得できるようにしている。

§ 4.3 周期性の検出

データ取得部の次に行う、行動識別部について述べる。行動識別部では、データ取得部で得たデータを整理し、KH Coder と R を用いて多変量解析を行い、ライフログデータの周期性を検出する。

開発したアプリケーションは、MOVERIO のカメラ画像を画像認識 API に送信し、約一分ごとに以下のテキストデータを取得、記録する。

```
[2018-01-28 10:30:12]{ "description":{ "tags":["indoor","laptop","table","computer","sitting","top","open","desk","white","keyboard","room","man","mouse","plate","laying","bed","playing"], "captions":[{"text":"an open laptop computer sitting on a table","confidence":0.95249546527278339}]},"requestId":"21b3c022-3d1b-4bc1-9cc8-df210d1f2094","metadata":{"height":720,"width":1280,"format":"Jpeg"}}
```

多変量解析を行う前に、前処理としてテキストデータのうち多変量解析に必要なデータのみを抽出する。Computer Vision API はタグとキャプションをライフログデータとして取得できるが、一度に取得するデータが多すぎるとライフログデータとしてノイズとなってしまう。なお、この時の視界は、机の上にノート PC がある状態であり、キャプションの精度は高く、机にあるノート PC を認識できていることがわかる。キャプションだけでは取得できない、indoor などの情報はタグの上位5個に現れていると考え、本研究では tags は confidence の高い順に5個、caption は confidence の最も高いキャプションを使用する。抽出した下記のテキストデータを解析を行いたい時間分テキストファイルに保存する。

```
an open laptop computer sitting on a table,indoor,laptop,table,computer,sitting
```

保存したテキストファイルを KH Coder を使用して、多変量解析する。この時、テキストデータの時系列から周期性を検出するために SOM を使用する。まず、KH Coder で SOM を作成する。このときクラスター数はクラスター解析などの結果から決定できる。KH Coder で SOM を作成した際に取得できる R ファイルの内容を、ライフログデータの時系列関係がわかるように書き換える必要がある。

R ファイルを読み込み、som パッケージを用いて SOM を出力する前に、以下のコマンドを追加する。追加することで、抽出語どうしの関係性を示す SOM ではなく、文章どうし、本研究では一分ごとのライフログデータどうしの関係性を示す SOM を出力できる。

```
d <- t(d)
rownames(d) <- 1:nrow(d)
```

また、本研究ではライフログデータの時系列関係をより視覚的に理解するため、R ファイルの最後に以下のコマンドを追加する。以下のコマンドを追加することで、出力された SOM データが格納されたデータフレーム points がプロットされた id 上に線分のみを上書きできる。

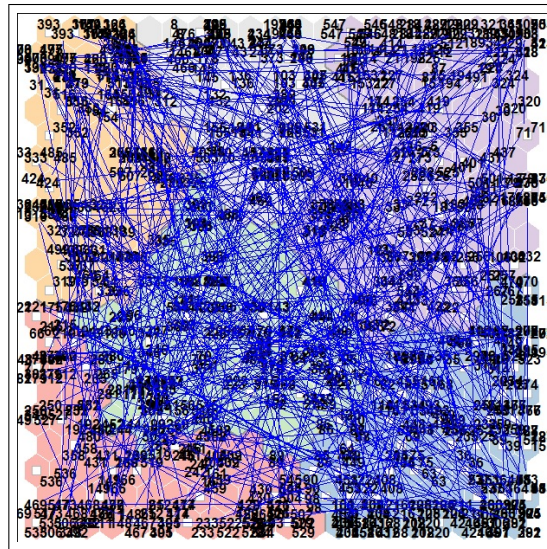


図 4.5: 「坊ちゃん」データから作成した SOM に線分を追加

```
par(new=T)
plot(points[,1],points[,2],type="c",col="色指定")
```

なお、二種類のデータを比較する SOM を作成する場合は、一つのテキストファイルに二種類のテキストファイルをまとめ以下のコマンドを追加する。以下のコマンドを追加することで、データフレーム `points` を二種類のデータに戻し、各々の色で線分を上書きできる。

```
points1<-head(points,n=総ライフログデータ/2)
par(new=T)
plot(points1[,1],points1[,2],type="c",col="色指定1")

points2<-tail(points,n=総ライフログデータ/2)
par(new=T)
plot(points2[,1],points2[,2],type="c",col="色指定2")
```

これらの R コマンドを使用して、ライフログデータの時系列を表示できる SOM を出力する。図 4.5 は 3.13 に線分を追加した SOM である。出力された SOM から行動パターンの類似性やイベント性を検出する。

シミュレーション結果ならびに考察

開発したアプリケーションを実際に使用して、ライフログデータを取得する。また、取得したデータを多変量解析を用いて、行動パターンの類似性やイベント性を検出する。

ライフログデータの取得日は2018年1月27日と28日の10時30分から13時30分の180分である。デバイスの充電が100%である状態から充電が切れるまで取得を行ったため取得時間は180分となっている。なお、おおよそ一分に一回データを取得したが、デバイスの処理能力に波があることからデータ数は190となっている。27日に取得したデータをデータ1、28日に取得したデータをデータ2とする。

図5.1にデータ1とデータ2のタイムスケジュールを示す。データ1は学校でデスクトップPCで作業を行い、外出するというライフログデータである。データ2は自宅でノートPCでの作業と食事を行うというライフログデータである。データ1、データ2共に、取得したテキストデータの中から confidence の高いタグ上位5個とキャプションを一行とした190行のテキストファイルを解析に使用する。

データ1とデータ2の比較を行いやすくするため、データ1とデータ2を一つのcsvファイルにしたものをデータ3とする（表5.1参照）。この時label列はデータ1とデータ2の区切りを格納してある。行番号1から190がデータ1であり、191から380がデータ2がとなっている。データ3を用いることで、データ1とデータ2の多変量解析結果を一つの結果上で確認できる。

KH Coder を用いて、取得したテキストファイルの前処理として自然言語処理を行い単語を抽出する。この時、抽出語の中でも、3回以上出現する抽出語を用いる。理由として、データ1、データ2共に抽出語を20個前後にし、プロットされる抽出語を減らすことで解析結果を読み取りやすくするためである。また、解析に使用する品詞は名詞と形容詞に絞る。理由として、動詞は取得したデータ内のキャプションに現れることが多く、コンピューターが置いてある、という状態を an open laptop computer sitting on a table というように画像認識APIが返答してしまうため、本研究ではsittingという状態がノイズになってしまう。そのため品詞を絞り、ノイズを減らすこととした。取得したデータからKH Coderで階層的クラスター分析、MDS、対応分析、共起ネットワーク、SOMを出力する。

まず、データ1とデータ2のクラスター分析を行う。図5.2はデータ1から作成したクラ

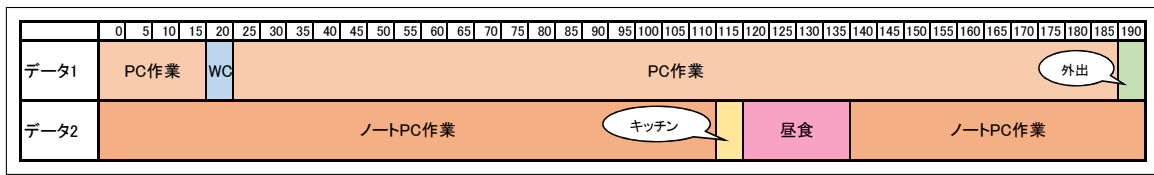


図 5.1: データ 1 とデータ 2 のタイムスケジュール

表 5.1: データ 3

	label	textdata
1	data1	a desk with a computer monitor,indoor,electronics,computer,monitor,table
2	data1	a desk with a computer monitor,indoor,monitor,computer,table,desk
3	data1	a desk with a computer monitor,indoor,computer,table,monitor,desk
4	data1	a desk with a computer monitor,indoor,monitor,table,computer,desk
⋮	⋮	⋮
377	data2	a stack of flyers on a table,indoor,table,top,sitting,desk
378	data2	a stack of flyers on a table,indoor,table,top,sitting,desk
379	data2	a stack of flyers on a table,indoor,table,top,sitting,desk
380	data2	a stack of flyers on a table,indoor,table,top,sitting,desk

スター分析である。この時クラスター数は Auto では 4 となり、図 5.3 の併合標準よりクラスター数 4 前後の傾きに大きな変化がないためクラスター数は 4 のままとした。データ 1 のクラスター分析より、赤のクラスターは computer や keyboard が含まれることから PC 作業であり出現回数もその他のクラスターに比べると最も多いことがわかる。青のクラスターは car や snow が含まれていること、outdoor という単語が含まれていることから外出時のことを表していると考えられる。紫のクラスターは女子トイレの壁の色である white が出現していることからトイレに行くことを示しているように考えた。緑のクラスターは screenshot という単語だけで構成されている、これは視界に画面が大きく含まれている状態ではないかと推測する。

図 5.4 はデータ 2 から作成したクラスター分析である。この時クラスター数は Auto では 5 となり、図 5.5 の併合標準より、クラスター数は 4 よりも 5 が適切であることは明らかのためクラスター数は 5 のままとした。最も多いのは青のクラスターの PC 作業であることが分かる。データ 1 と比較すると、PC 作業を表す単語の中に desktop や keyboard は含まれず、laptop が含まれていることからノート PC での作業を確認できる。ピンクのクラスターは food や plate から食事を表し、緑のクラスターはキッチンを表していると考えますが、PC で動画を見ながら食事を行っていたため、食べ物以外の物体との関係性が強く表され、自室の私物である flyer や bottle など含まれてしまっている。食事していることをより正確にライフログデータとして取得するには、食べ物をなるべく視界に入れてデータを取得しなくてはいけないのではないかと考えた。また、図 5.2 と図 5.4 を比較すると、データ 2 の自宅の方が視界に入る物体が多いことから bottle や flyer などライフログデータに関係のないものまで取得されていることがわかる。

次に、データ 1 とデータ 2 の MDS を行う。図 5.6 はデータ 1 から作成した MDS である。

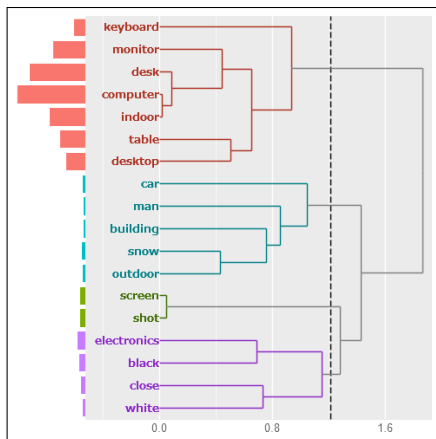


図 5.2: データ 1 のクラスター分析

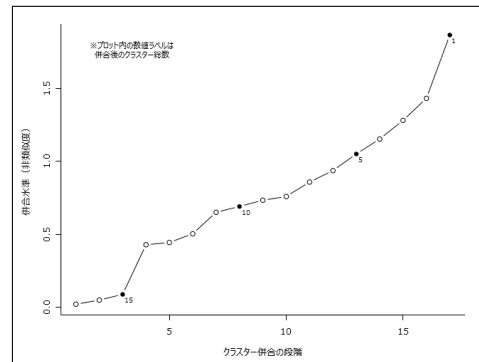


図 5.3: データ 1 の併合標準

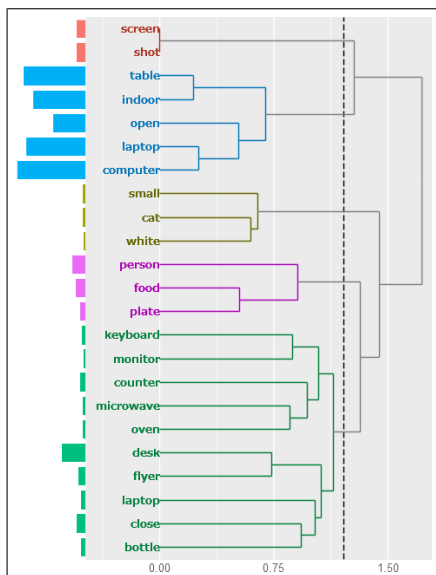


図 5.4: データ 2 のクラスター分析

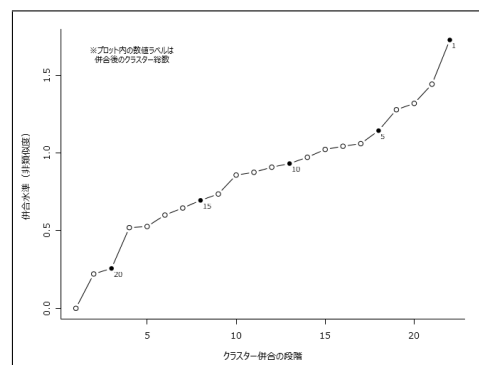


図 5.5: データ 2 の併合標準

クラスター分析より、クラスター数は4とした。PC 作業に関係する computer や desk という抽出語から構成されている一番大きいクラスター 01 と、クラスター 02,03,04 は距離が離れていることとクラスター分けから行動がはっきり分かれていることがわかる。なお、クラスター数を3にして出力を行うとクラスター 03 と 04 が一つのクラスターになったため、クラスター 03 は 02 よりも 04 に近いものとする。

図 5.7 はデータ 2 から作成した MDS である。クラスター分析より、クラスター数は5とした。computer や screen から構成されるクラスター 01 と desk や flyer から構成される 02 は PC 作業という行動を示すため、近い位置に配置されていることがわかる、クラスター 01 と 03 が近いのは、食事の際視界に PC が入り込んでいた影響だと考える。したがって、作業を行いながら食事を行っているのではないかと推測できるプロットとなっている。01 から少し離れた 04 に oven や microwave という抽出語があるため、オーブンや電子レンジを使用したことなどが考えられる。また、クラスター 05 は 01~04 より少し離れているた

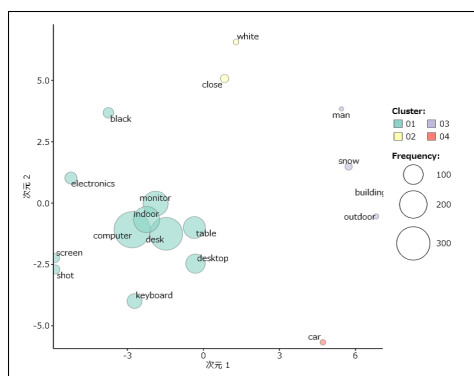


図 5.6: データ 1 の MDS

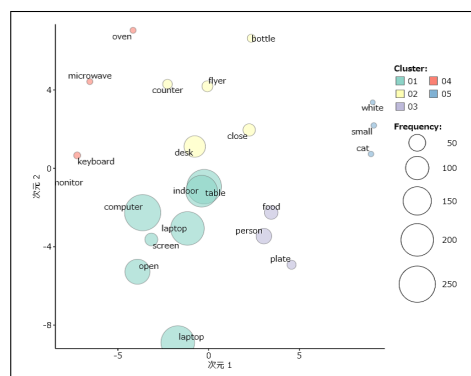


図 5.7: データ 2 の MDS

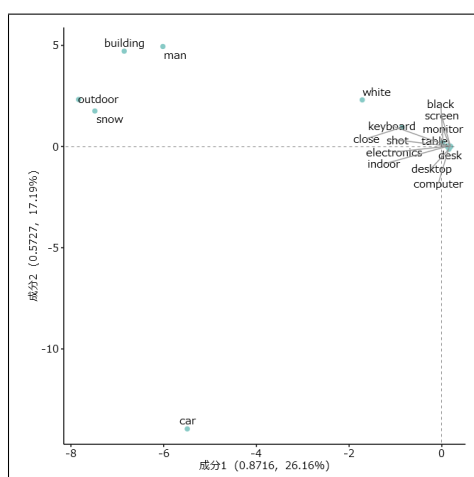


図 5.8: データ 1 の対応分析

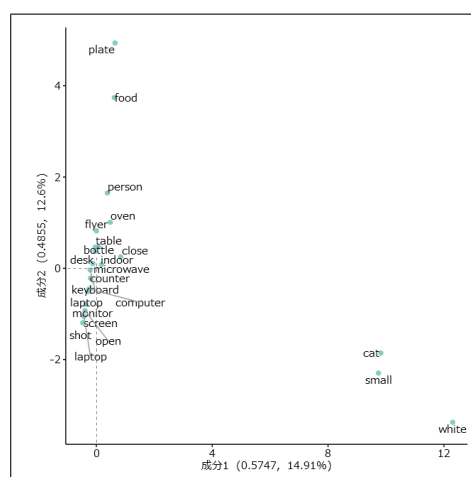


図 5.9: データ 2 の対応分析

めノイズではないかと考えられる．図 5.6 と図 5.7 を比較すると，データ 2 のほうがクラスター間のプロットの距離が近いことがわかる．このことから，実際に似たような行動を複数行っているか，行動を行っている際の視界情報が多いのではないかと考えられる．データ 1 もデータ 2 もクラスター 01 は PC 作業に関する抽出語で構成されているため，PC 作業は類似した行動ではないかと考えられる．クラスター 01 が一番大きく，他のクラスターと大きく差がある点は類似しているが，他のクラスターを構成する抽出語の相違からイベント性を検出できる．

次に，データ 1 とデータ 2 の対応分析を行う．図 5.8 はデータ 1 から作成した対応分析である．図 5.8 より，computer や keyboard で構成される行動である PC 作業を行っていることが最も多く，データ 1 内で特徴的でないことがわかる．また，white や outdoor は原点より離れているため特徴的な行動を構成する抽出語であると考えられる．また，building や outdoor から室外での行動であることが考えられる．

図 5.9 はデータ 2 から作成した対応分析である．図 5.8 と図 5.9 の比較すると，データ 1 もデータ 2 も PC 作業を中心に行っているが，データ 2 は原点より少し離れたところに oven があり，food や white はより特徴的になっていることがわかる．このことから食事をとったことが推測できる．

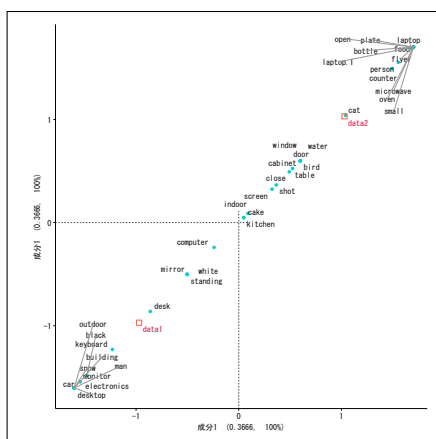


図 5.10: データ 3 の対応分析

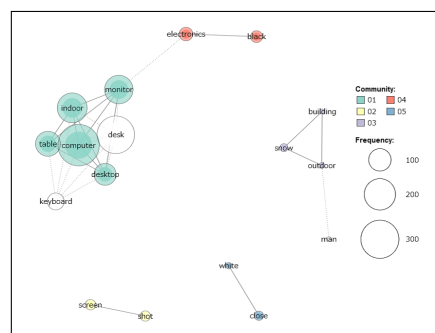


図 5.11: データ 1 の共起ネットワーク

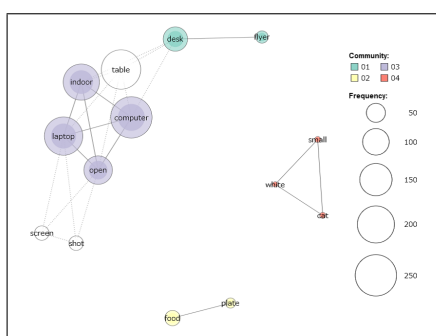


図 5.12: データ 2 の共起ネットワーク

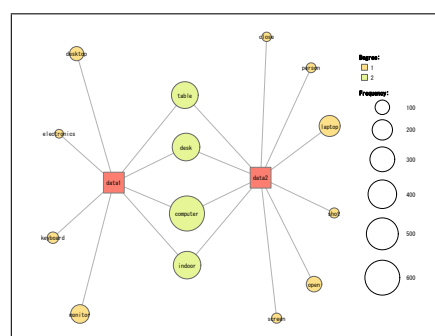


図 5.13: データ 3 の共起ネットワーク

さらに、データ 1 とデータ 2 の関係性を同時に出力することができるため、データ 3 の対応分析を行う。5.10 より、データ 1、データ 2 共に同じくらい出現している抽出語、つまり特徴的ではない抽出語として indoor や computer が出現している。データ 1 からみて、データ 2 に含まれる table 等は関係性が近いが、food 等は関係性がないため特徴的であるように出力されている。同じようにデータ 2 からみて、データ 1 に含まれる snow 等は特徴的な語となっている。この比較より、データ 1 とデータ 2 には類似する行動もあることがわかる。

次に、データ 1 とデータ 2 の共起ネットワークを行う。図 5.11 はデータ 1 から作成した共起ネットワークである。この時、Jaccard 係数が 0.2 以上の共起関係を描画している。図 5.11 より、computer や monitor など PC 作業を表す抽出語どうしは線で結ばれているため、共起関係があることがわかる。また、electronics と black とも、クラスターは違っているが共起関係があることがわかる。図 5.12 はデータ 2 から作成した共起ネットワークであり、図 5.11 と比較すると、Jaccard 係数が 0.2 以上の強い共起関係を持つ抽出語が少なく、クラスターも少なくなっていることがわかる。

さらに、データ 1 とデータ 2 の共起関係性を同時に出力するため、データ 3 の共起ネットワークを行う。5.13 より、データ 1 とデータ 2 はともに table, desk, computer, indoor という抽出語と共起関係があり、両方とも Jaccard 係数が 0.2 以上の強い共起関係をもつのは PC 作業を表す抽出語であり、類似性のある行動が確認できる。

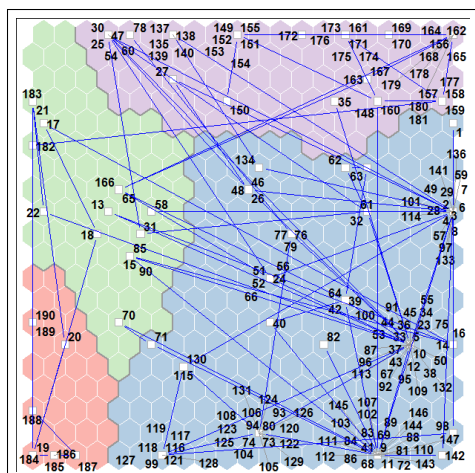


図 5.14: データ 1 の SOM

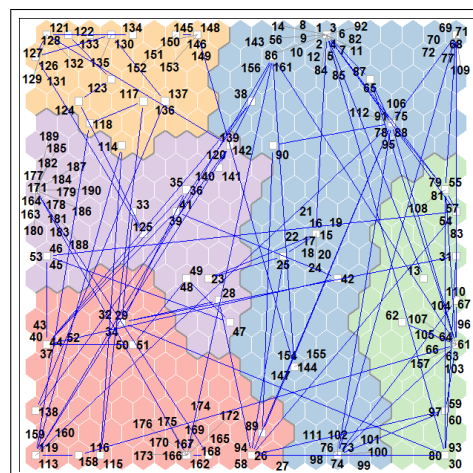


図 5.15: データ 2 の SOM

最後に今までの解析を踏まえてライフログデータの時系列を可視化するため、SOMを作成する。クラスター数はデータ 1 は 4，データ 2 は 5 とした。

図 5.14 はデータ 1 の SOM である。この SOM とテキストデータを照らし合わせると、青のクラスターは PC 作業を表し、緑のクラスターはトイレ、赤のクラスターは外出を表していると考えることができた。なお、外出時もトイレにいる際も視界は白色が多く、white という単語が共通するため、プロットが近いのだと考えた。また、紫のクラスターは a screen shot of a computer や a close up of a computer などの PC 作業の中で出現したノイズのようなテキストから構成されていたが、これはコンピュータースクリーンに近づいて作業を行っている際に出現するテキストであり、青と紫のクラスターは同じ PC 作業を表している。

図 5.15 はデータ 2 の SOM である。この SOM とテキストデータを同じく照らし合わせると、黄色のクラスターは食事、紫のクラスターは書類が置いてあること、赤のクラスターはキッチンや部屋においてある家具を表していた。青のクラスターは PC 作業をあらわし、緑のクラスターはデータ 1 と同じくコンピュータースクリーンに近づいて作業を行っている際に出現するテキストから構成されているため、青と緑のクラスターは同じ PC 作業を表している。また、クラスター間を大きくまたぐ線などもあり、常に同じ行動を行っていても、視界に入る物体の変化から線が乱雑になっていると感じた。

データ 3 の SOM を作成し、データ 1 とデータ 2 の時系列の類似性を比較する。データは赤色の線分、データ 2 は青色の線分で示す。クラスター数はデータ 1 とデータ 2 のクラスター数を合わせ 9 とした。また、データ 3 の SOM を作成するソースコードはソースコード A.?? に示す。図 5.16 の、クラスターの分かれ方を 5.17 にしめす。

これより、データ 1 とデータ 2 の時系列は類似性が低いことがわかる。理由として、同じ PC 作業であってもデスクトップ PC とノート PC という別の PC を使用した作業であるため同じ行動の中でも視界に写る物体が違いから行動が区別されているからであると考えられる。また、赤い線と青い線が両方ともつながっている時間のテキストを確認すると、a screen shot of a computer や a close up of a computer などのコンピュータースクリーンに近づいて作業を行っていることや person という単語が入るテキストが含まれていた。これは PC 画面に映った人の画像や、自宅のポスターを認識していると考えられる。

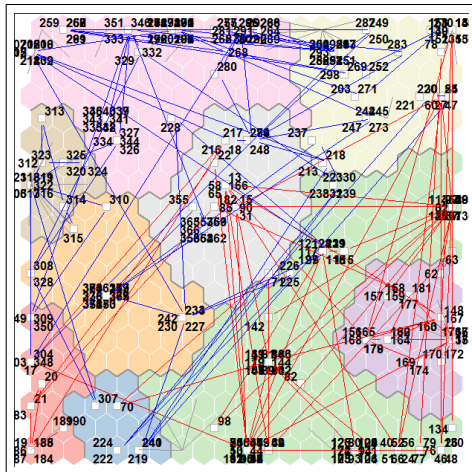


図 5.16: データ 3 の SOM

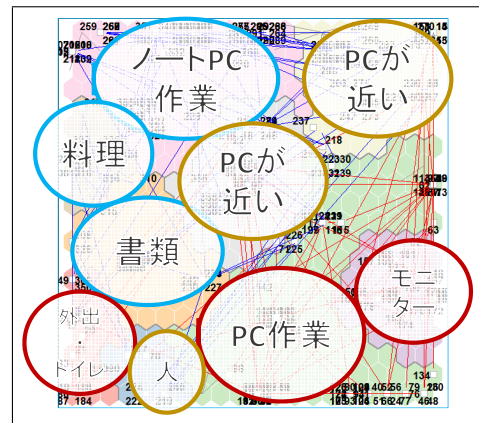


図 5.17: データ 3 の SOM に行動を追記した SOM

階層的クラスター分析，MDS，対応分析，共起ネットワークの解析から，データを構成する行動の検出，類似する行動とそうではないイベント性のある行動を検出することができた．これによって，どのような行動から行っているかという行動識別が可能となっていると考える．また，SOMの解析から，同じ行動でも視界に写る物体の違いから行動の類似性やイベント性を検出できた．この結果から，同じ行動でも使用する場所や物体の変化によって別行動として認識させることができるため，GPSを使用せず，ライフログデータに位置情報を付加できると考えられる．よって，個人情報保護に着目し取得したライフログデータから類似性やイベント性を検出できたと考える．

結論と今後の課題

本研究の目的は、多くの人に広く受け入れられるライフログとして、個人情報保護に着目し、手間がかからず自動的にライフログデータの取得を行い、取得したデータから類似性やイベント性を考察できることである。開発したライフログデータ取得アプリケーションを使用したビッグデータ構築・データ解析を行い、行動パターンの類似性・イベント検出を行った。

結論として、個人情報保護に着目したライフログデータ取得アプリケーションの開発ができ、多変量解析を用いることでライフログの可視化を行い行動パターンの類似性やイベント性を視覚的に検出するという目標は達成できた。特に、SOMの解析結果より、同じ行動でも視界に写る物体の違いから行動の類似性やイベント性を検出できた。同じ行動でも使用する場所や物体の変化によって別行動として認識させることができるため、ライフログデータに位置情報を付加できると考えられる。よって、個人情報保護に着目し取得したライフログデータから類似性やイベント性を検出できたと考える。本研究の研究成果は、テキストによるライフログデータ取得、解析を行い新たなビジネスプランの検討やユーザー自身の生活の見直しなどに使用できるため、より高度なアプリケーション開発を目指す開発者、研究者の方々の参考になれば幸いである。解明できた点は必ずしも多くはないが、若干なりとも寄与できたと思われる。

今後の課題として、開発したアプリケーションの改善点を上げる。開発したアプリケーションは自動的にライフログデータを取得する点が利点として挙げられるが、一方で客観的なライフログデータしか取得できないという弱点もある。ユーザーが興味を持った瞬間や、データを取得したい瞬間のライフログデータは現状のアプリケーションには含まれていないためである。この弱点に対し、ユーザーが取得したいタイミングでライフログデータを取得する方法をアプリケーションに組み込む必要がある。組み込むため、取得したいタイミングを MOVERIO に伝える方法の検討も必要となる。

謝辞

本研究を遂行するにあたり，多大なご指導と終始懇切丁寧なご鞭撻を賜った富山県立大学電子・情報工学科の奥原浩之教授に深甚な謝意を表します．最後になりましたが，多大な協力をして頂いた研究室の同輩諸氏に感謝致します．

2018 年 2 月

福島 瑞希

参考文献

- [1] 相澤清晴, “ライフログ”, 映像情報メディア学会誌, Vol. 63, No. 4, pp. 445–448, 2009.
- [2] 芳竹宣裕, 伊藤慎, “ユビキタス環境が生み出す大量情報「ライフログ」の活用と実装技術”, NEC 技報, Vol. 62, No. 4, p. 77, 2009.
- [3] 角田宏貴, Hiroki SUMIDA, “ライフログ分析による行動特徴抽出及びイベント検出”, 法政大学大学院紀要 (情報科学研究科編), Vol. 9, pp. 119–124, 2014.
- [4] 矢野裕司, 横井健, 橋山智訓, “行動辞書を利用した Twitter からの行動抽出”, 情報科学技術フォーラム講演論文集, Vol. 11, No. 4, pp. 51–56, 2012.
- [5] 緒方広明, “日本語学習を支援するユビキタス学習環境に関する研究”, <http://www.taf.or.jp/files/items/542/File/P212.pdf>, 閲覧日 2018,1,30.
- [6] 啓之田中, “位置情報の規律のあり方: スマートフォン時代の利便性とプライバシー”, 人間社会研究, Vol. 11, pp. 75–85, 2014.
- [7] 新保史生, “ライフログの定義と法的責任 個人の行動履歴を営利目的で利用することの妥当性”, 情報管理, Vol. 53, No. 6, pp. 295–310, 2010.
- [8] 北村圭吾, 山崎俊彦, 相澤清晴, “食事ログの取得と処理ー画像処理による食事記録ー”, 映像情報メディア学会誌, Vol. 63, No. 3, pp. 376–379, 2009.
- [9] 株式会社 N T T データ経営研究所, “日本語学習を支援するユビキタス学習環境に関する研究”, <http://www.keieiken.co.jp/aboutus/newsrelease/161122/>, 閲覧日 2018,2,5.
- [10] 入江英嗣, 森田光貴, 岩崎央, 千竈航平, 放地宏佳, 小木真人, 檜原裕大, 芝星帆, 眞島一貴, 努吉永, “AirTarget: 光学シースルー方式 HMD とマーカレス画像認識による高可搬性実世界志向インタフェース”, 情報処理学会論文誌, Vol. 55, No. 4, pp. 1415–1427, 2014.
- [11] 川上晃平, “スマートグラスを利用した授業支援システムの開発”, 2017.
- [12] 倉田陽平, 真田風, 鈴木祥平, 石川博, “Flickr と Google Cloud Vision API によりテーマ別観光マップを作る試み”, <http://db-event.jpn.org/deim2017/papers/321.pdf>, 閲覧日 2018,1,4.
- [13] 大雄治, 吉川眞, 田中一成, “ソーシャルメディアを活用した景観の分析と評価”, 日本都市計画学会関西支部研究発表会講演概要集, Vol. 15, pp. 13–16, 2017.
- [14] 小林亜令, 岩本健嗣, 西山智, “釈迦: 携帯電話を用いたユーザ移動状態推定・共有方式-モバイルコンピューティング, モバイルアプリケーション, ユビキタス通信, モバイルマルチメディア通信-”, 電子情報通信学会技術研究報告. MoMuC, モバイルマルチメディア通信, Vol. 108, No. 44, pp. 115–120, 2008.

- [15] 寺田努, “ウェアラブルセンサを用いた行動認識技術の現状と課題”, コンピュータ ソフトウェア, Vol. 28, No. 2, pp. 43–54, 2011.
- [16] 貴志一樹, 山崎俊彦, 相澤清晴, “机上行動のライフログのための行動認識”, 映像情報メディア学会年次大会講演予稿集, Vol. 2014, , 2014.
- [17] 前川卓也, 柳沢豊, 岸野泰恵, 石黒勝彦, 亀井剛次, 櫻井保志, 岡留剛, “ウェアラブルセンサによるモノを用いた行動の認識について”, Technical Report 57, 研究報告ユビキタスコンピューティングシステム (UBI) , 2010.
- [18] 樋口耕一, “テキスト型データの計量的分析: 2つのアプローチの峻別と統合”, 理論と方法, Vol. 19, No. 1, pp. 101–115, 2004.
- [19] 佐野香織, 李在鎬, “KH Coder で何ができるか: 日本語習得・日本語教育研究利用への示唆”, 言語文化と日本語教育, Vol. 33, pp. 94–95, 2007.
- [20] “KH Coder を用いた研究事例のリスト”, <http://khc.sourceforge.net/bib.html>, 閲覧日 2018,1,7.
- [21] 二宮隆次, 小野浩幸, 高橋幸司, 野田博行, “新聞記事を基にしたテキストマイニング手法による産学官連携活動分析”, 科学・技術研究, Vol. 5, No. 1, pp. 93–104, 2016.
- [22] “クラスター分析の手法 (階層クラスター分析) — データ分析基礎知識”, https://www.albert2005.co.jp/knowledge/data_mining/cluster/hierarchical_clustering, 閲覧日 2018,1,24.
- [23] “階層的クラスター分析について”, http://koichi.nihon.to/cgi-bin/bbs_khn/khcf.cgi?list=&no=977&mode=allread&page=0, 閲覧日 2018,1,24.
- [24] 吉原一紘, 徳高平蔵, “クラスター分析の概要”, *Journal of Surface Analysis*, Vol. 21, No. 1, pp. 10–17, 2014.
- [25] 李美龍, 田中恒也, 成田吉弘, “画像を用いた製品の「飽き」に関する感性評価: デザインの視覚的要素を中心に—”, 日本感性工学会論文誌, Vol. 11, No. 3, pp. 407–417, 2012.
- [26] 小峯敦・下平裕之, “ベヴァリッジ『自由社会における完全雇用』のケインズの要素-テキストマイニングを加味した量的・質的分析-”, 2017.
- [27] 齋藤堯幸, “多次元尺度構成法”, 計測と制御, Vol. 22, No. 1, pp. 126–131, 1983.
- [28] Joseph B Kruskal, “Multidimensional scaling by optimizing goodness of fit to a non-metric hypothesis”, *Psychometrika*, Vol. 29, No. 1, pp. 1–27, 1964.
- [29] “Jaccard 係数の計算式と特徴 (1) ”, <https://www.slideshare.net/khcoder/jaccard1>, 閲覧日 2018,2,3.

- [30] 中山慶一郎, “< 研究ノート > 対応分析によるデータ解析”, 関西学院大学社会学部紀要, No. 108, pp. 133–145, 2009.
- [31] 田中京子, “KH Coder と R を用いたネットワーク分析”, 久留米大学コンピュータジャーナル, Vol. 28, pp. 37–52, 2014.
- [32] “共起ネットワークにおける中心性の解釈について”, http://www.koichi.nihon.to/cgi-bin/bbs_khn/khcf.cgi?no=2493&mode=allread#2496, 閲覧日 2018,2,2.
- [33] 横田尚己, 山田圭二郎, “熊本地震のつぶやきに見る感情極性値の時空間解析”, 都市計画論文集, Vol. 52, No. 3, pp. 1081–1087, 2017.
- [34] 増田正, Masuda Tadashi, 高崎経済大学地域政策学部, “地方議会の会議録に関するテキストマイニング分析：高崎市議会を事例として”, 地域政策研究 = Studies of regional policy, Vol. 15, No. 1, pp. 17–31, 2012.
- [35] T. KOHONEN, “Self-organized formation of topologically correct feature map”, *Biol. Cybern.*, Vol. 43, pp. 59–69, 1982.
- [36] 岡晋之介, “自己組織化マップを用いた気象要素の分類と予測”, <http://www.gifu-nct.ac.jp/elec/deguchi/sotsuron/oka/oka.html>, 閲覧日 2018,1,7.
- [37] “自己組織化特徴マップ (SOM) ”, <http://www.sist.ac.jp/kanakubo/research/neuro/selforganizingmap.html>, 閲覧日 2018,1,31.
- [38] 勝治宏基, 米澤拓郎, 中澤仁, 高汐一紀, 徳田英幸ほか, “Synchrometer: ライフログを利用した日常行動における他者との類似度生成”, 研究報告ユビキタスコンピューティングシステム (UBI), Vol. 2013, No. 17, pp. 1–7, 2013.

付録

A. 1 ライフログデータ取得アプリケーションのソースコード

ライフログデータ取得アプリケーションのソースコード A.1 をしめす.

ソースコード A. 1: app.cs

```
1 using UnityEngine;
2 using System.Collections;
3 using System.Collections.Generic;
4 using UnityEngine.UI;
5 using System.IO;
6 using System;
7 using System.Text;
8 using System.Linq;
9
10 public class app : MonoBehaviour
11 {
12     private float captureIntervalSeconds = 50.0f;
13     private float captureIntervalSeconds2 = 5.0f;
14     public Text gtext;
15     Dictionary<string, string> headers;
16     private int Width = 1280;
17     private int Height = 720;
18     private int FPS = 30;
19     private WebCamTexture webcamTexture;
20     private Color32[] color32;
21     string responseData;
22     private string reportFileName2 = "long_report.txt";
23     public bool addDateTime = true;
24
25     void Start ()
26     {
27         Screen.sleepTimeout = SleepTimeout.NeverSleep;
28         StartCoroutine ("Sample");
29     }
30
31     public IEnumerator Sample ()
32     {
33         WebCamDevice[] devices = WebCamTexture.devices;
34         WebCamDevice userCameraDevice = WebCamTexture.devices [0];
35         webcamTexture = new WebCamTexture (userCameraDevice.name, Width, Height,
36             FPS);
37         webcamTexture.Play ();
38         Debug.Log ("webcamTexture");
39
40         yield return new WaitForSeconds (captureIntervalSeconds2);
41         color32 = webcamTexture.GetPixels32 ();
42         Texture2D texture = new Texture2D (webcamTexture.width, webcamTexture.height
43             );
44         texture.SetPixels32 (color32);
45         texture.Apply ();
46         byte[] jpg = texture.EncodeToJPG ();
47         string VISIONKEY = "ba7982e18b4943d18024749aca8031fb";
```

```

46     var uri = "https://westus.api.cognitive.microsoft.com/vision/v1.0/
        describe";
47
48
49     var headers = new Dictionary<string, string> () {
50         { "Ocp-Apim-Subscription-Key", VISIONKEY },
51         { "Content-Type", "application/octet-stream" }
52     };
53
54     WWW www = new WWW (uri, jpg, headers);
55     yield return www;
56     responseData = www.text;
57     gtext.text = responseData;
58     DateTime dt = DateTime.Now;
59     string text2 = dt.ToString (" [yyyy-MM-dd_HH:mm:ss] ") + responseData.ToString
        () + "\n";
60     string outfile2 = reportFileName2;
61
62     if (addDateTime) {
63         string file2 = Path.GetFileNameWithoutExtension (reportFileName2);
64         string ext2 = Path.GetExtension (reportFileName2);
65         outfile2 = file2 + "_" + dt.ToString ("yyyyMMdd") + ext2;
66     }
67
68     SaveText (text2, Path.Combine (Application.persistentDataPath, outfile2));
69     color32 = null;
70     StartCoroutine ("StopRunTimeTemp");
71 }
72
73 public IEnumerator StopRunTimeTemp ()
74 {
75     webcamTexture.Stop ();
76     yield return new WaitForSeconds (captureIntervalSeconds);
77     StartCoroutine ("Sample");
78 }
79
80 public static bool SaveText (string text, string path)
81 {
82     try {
83         using (StreamWriter writer = new StreamWriter (path, true)) {
84             writer.Write (text);
85             writer.Flush ();
86             writer.Close ();
87         }
88     } catch (Exception e) {
89         Debug.Log (e.Message);
90         return false;
91     }
92     return true;
93 }
94 }

```

A. 2 クラスター分析を作成するソースコード

クラスター分析を作成するソースコード A.2 をしめす。

ソースコード A. 2: kura.r

```

1 d <- NULL
2 d <- matrix( c(1,...省略...,0), byrow
               =T, nrow=190, ncol=19 )
3 d <- d[,-1]
4 colnames(d) <- c("desk",...省略...,"
                   outdoor")
5 doc_length_mtr <- matrix( c(
    70,18,...省略...45,17), ncol=2,
    byrow=T)
6 colnames(doc_length_mtr) <- c("
    length_c", "length_w")
7 color_universal_design <- 1
8
9 d <- t(d)
10 # END: DATA
11 n_cls <- 4
12 font_size <- 1
13 labels <- rownames(d)
14 rownames(d) <- NULL
15 freq <- NULL
16 for (i in 1:nrow(d)) {
17   freq[i] = sum( d[i,] )
18 }
19 method_dist <- "binary"
20 method_clst <- "ward"
21
22 library(amen)
23 dj <- Dist(d,method=method_dist)
24
25 if (
26   ( as.numeric( R.Version())$
        major ) >= 3 )
27   && ( as.numeric( R.Version())$
        minor ) >= 1.0)
28 ){ # >= R 3.1.0
29   if (method_clst == "ward"){
30     method_clst <- "ward.D2"
31   }
32   hcl <- hclust(dj,method=method_
    clst)
33 } else { # <= R 3.0
34   if (method_clst == "ward"){
35     dj <- dj^2
36     hcl <- hclust(dj,method=method
    _clst)
37     hcl$height <- sqrt( hcl$height )
38   } else {
39     hcl <- hclust(dj,method=method
    _clst)
40   }
41 }
42 par(
43   mai=c(0,0,0,0),
44   mar=c(1,2,1,0),
45   oma=c(0,0,0,0),
46   oma=c(0,0,0,0)
47 )
48
49
50 library(grid)
51 library(ggplot2)
52 library(ggdendro)
53
54 ddata <- dendro_data(as.
    dendrogram(hcl), type="
    rectangle")
55
56 p <- NULL
57 p <- ggplot()
58
59 font_family <- "Meiryo UI"
60 if ( exists("PERL_font_family") ){
61   font_family <- PERL_font_family
62 }
63
64 if (n_cls > 1){
65   memb <- cutree(hcl,k=n_cls)
66
67   p <- p + scale_colour_hue(l=40, c
    =100)
68
69   cutpoint <- mean(
70     c(
71       rev(hcl$height)[n_cls-1],
72       rev(hcl$height)[n_cls]
73     )
74   )
75
76   n <- length( unique(memb[hcl$
    order]) )
77   new_col <- NULL
78   for (i in 1:ceiling(n / 2) ){
79     new_col <- c(new_col, i)
80     if (i + ceiling(n / 2) <= n){
81       new_col <- c(new_col, i +
    ceiling(n / 2))
82     }
83   }
84
85   col_tab <- cbind(
86     unique(memb[hcl$order]),
87     new_col
88   )
89   colnames(col_tab) <- c("org","
    new")
90   col_vec <- NULL
91   for (i in col_tab[order(col_tab[,1])
    ,2]){
92     c <- as.character(i)
93     while (nchar(c) < 3){
94       c <- paste("0",c,sep="")
95     }
96     col_vec <- c(col_vec, c)
97   }
98
99   seg_bl <- NULL
100   seg_cl <- NULL
101   colnames(ddata$segment) <- c(
102     "x0",
103     "y0",

```

```

104     "x1",
105     "y1"
106 )
107 colnames(ddata$labels) <- c(
108     "x",
109     "y",
110     "text"
111 )
112 for ( i in 1:nrow( ddata$segment ) )
113 {
114     if (
115         ddata$segment$y0[i] >
116         cutpoint
117         || ddata$segment$y1[i] >
118         cutpoint
119         || (
120             ddata$segment$y0[i] >=
121             cutpoint
122             && ddata$segment$y1[i] >=
123             cutpoint
124         )
125     ) {
126         seg_bl <- c(
127             seg_bl,
128             ddata$segment$x0[i],
129             ddata$segment$y0[i],
130             ddata$segment$x1[i],
131             ddata$segment$y1[i]
132         )
133     } else {
134         seg_cl <- c(
135             seg_cl,
136             ddata$segment$x0[i],
137             ddata$segment$y0[i],
138             ddata$segment$x1[i],
139             ddata$segment$y1[i],
140             #col_vec[
141             memb[hcl$order][
142             floor(
143                 mean(
144                     ddata$segment$x0[i],
145                     ddata$segment$x1[i])
146                 )
147             ]
148             #]
149         )
150     }
151 }
152 seg_bl = matrix(seg_bl, byrow=T,
153                 ncol=4 )
154 seg_cl = matrix(seg_cl, byrow=T,
155                 ncol=5 )
156
157 if (is.null(seg_bl) == F){
158     colnames(seg_bl) <- c("x0", "y0",
159                           "x1", "y1")
160     seg_bl <- as.data.frame(seg_bl)
161
162     if ( max(seg_bl$y1) > cutpoint ){
163         p <- p + geom_hline(
164             yintercept = cutpoint,
165             colour="black",
166             linetype=5,
167             size=0.5
168         )
169     }
170 }
171
172 colnames(seg_cl) <- c("x0", "y0",
173                       "x1", "y1", "c")
174 seg_cl <- as.data.frame(seg_cl)
175 seg_cl$c <- col_vec[seg_cl$c]
176
177 p <- p + geom_text(
178     data=data.frame(
179         x=label(ddata)$x,
180         y=label(ddata)$y,
181         text=labels[ as.numeric( as.
182             vector( ddata$labels$text
183             ) ) ],
184         cols= col_vec[ memb[ as.
185             numeric( as.vector( ddata
186             $labels$text ) ) ] ]
187     ),
188     aes_string(
189         x="x",
190         y="y",
191         label="text",
192         colour="cols"
193     ),
194     hjust=1,
195     angle=0,
196     family = font_family,
197     fontface = "bold",
198     size = 5 * 0.85 * font_size
199 )
200
201 p <- p + geom_segment(
202     data=seg_cl,
203     aes_string(x="x0", y="y0", xend
204               ="x1", yend="y1", colour="c"
205               ),
206     size=0.5
207 )
208 } else {
209     memb <- rep( c("a"), length(
210         labels) )
211     p <- p + scale_colour_manual(
212         values=c("black"))
213     seg_bl <- ddata$segment
214     col_vec <- c("001")
215     p <- p + geom_text(
216         data=data.frame(
217             x=label(ddata)$x,
218             y=label(ddata)$y,
219             text=labels[ as.numeric( as.
220                 vector( ddata$labels$text
221                 ) ) ],
222             cols= col_vec[ memb[ as.
223                 numeric( as.vector( ddata
224                 $labels$text ) ) ] ]
225         ),
226         aes_string(

```

```

205     x="x",
206     y="y",
207     label="text",
208     colour="cols"
209   ),
210   hjust=1,
211   angle=0,
212   family = font_family,
213   fontface = "bold",
214   size = 5 * 0.85 * font_size
215 )
216 }
217
218 if (is.null(seg_bl) == F){
219   p <- p + geom_segment(
220     data=seg_bl,
221     aes_string(x="x0", y="y0", xend
222               ="x1", yend="y1"),
223     color="gray50",
224     linetype=1,
225   )
226
227   p <- p + geom_text(
228     data=data.frame(
229       x=label(ddata)$x,
230       y=label(ddata)$y,
231       text=labels[ as.numeric( as.
232                             vector( ddata$labels$text )
233                             ) ],
234       cols= col_vec[ memb[ as.numeric
235                           ( as.vector( ddata$labels$
236                             text ) ) ] ]
237     ),
238     aes_string(
239       x="x",
240       y="y",
241       label="text",
242       colour="cols"
243     ),
244     hjust=1,
245     angle=0,
246     family = font_family,
247     fontface = "bold",
248     size = 5 * 0.85 * font_size
249   )
250   # "strwidth" crashes if the device is
251   # cairo_pdf or cairo_ps
252   if (
253     is.na(dev.list()[ "cairo_pdf" ])
254     && is.na(dev.list()[ "cairo_ps" ])
255   ){
256     y_min <- max(
257       strwidth(
258         labels[ as.numeric( as.vector(

```

```

259         font = 2
260       )
261     )
262   }
263   y_min <- ( 6 * y_max * y_min ) / (
264     5 - 6 * y_min )
265   y_min <- y_min * 1.1
266   if (y_min > y_max * 2){
267     y_min <- y_max * 2
268   }
269   y_min <- y_min * -1
270
271   b1 <- 0
272   for (i in 1:1000){
273     b1 <- signif(y_max * 0.875, i)
274     if (b1 < y_max){
275       break
276     }
277   }
278
279   p <- p + coord_flip()
280   p <- p + scale_x_reverse(
281     expand = c(0,0),
282     breaks = NULL,
283     limits=c( length(ddata$labels$
284               text) + 0.5 , 1 - 0.5 )
285   )
286   p <- p + scale_y_continuous(
287     limits=c(y_min,y_max),
288     breaks=c(0,b1/2,b1),
289     expand = c(0.02,0.02)
290   )
291   p <- p + theme(
292     axis.title.y = element_blank(),
293     axis.title.x = element_blank(),
294     axis.ticks = element_line(colour="
295       gray60"),
296     axis.text.y = element_text(size=12,
297       colour="gray40"),
298     axis.text.x = element_text(size=12,
299       colour="gray40"),
300     legend.position="none"
301   )
302   if (n_cls <= 1){
303     p <- p + theme(
304       axis.text.y = element_blank(),
305       axis.text.x = element_text(size
306         =12,colour="black"),
307       axis.ticks = element_line(colour="
308         black"),
309       #panel.grid.major = theme_blank
310       (),
311       #panel.grid.minor = theme_blank
312       (),
313       #panel.background = theme_blank
314       (),
315       axis.line = element_line(colour =
316         "black")
317     )
318   }

```

```

309   )
310 }
311
312 show_bar <- 1
313
314 if (show_bar == 1){
315   p <- p + theme(
316     axis.ticks = element_blank(),
317     axis.text.y = element_blank()
318   )
319   p <- p + theme(
320     plot.margin = unit(c(0,0,0,0), "
321       lines")
322   )
323   bard <- data.frame(
324     nm <- labels[ as.numeric( as.
325       vector( ddata$labels$text )
326     ) ],
327     ht <- freq[ as.numeric( as.
328       vector( ddata$labels$text )
329     ) ],
330     cl <- col_vec[ memb[ as.
331       numeric( as.vector( ddata$
332         labels$text ) ) ] ],
333     od <- nrow(d):1
334   )
335   if (n_cls <= 1){
336     bard$cl <- "001"
337   }
338
339   p2 <- NULL
340   p2 <- ggplot()
341   p2 <- p2 + geom_bar(
342     stat="identity",
343     position = "identity",
344     width=0.75,
345     data=bard,
346     aes(
347       x=reorder(od,od),
348       y=ht,
349       fill=cl
350     )
351   )
352   p2 <- p2 + coord_flip()
353   p2 <- p2 + scale_y_reverse(
354     expand = c(0,0))
355   p2 <- p2 + scale_x_discrete(
356     expand = c(0,0) )
357   p2 <- p2 + theme(
358     axis.title.y = element_blank(),
359     axis.title.x = element_blank(),
360     axis.ticks = element_blank(),
361     axis.text.y = element_blank(),
362     axis.text.x = element_text(size
363       =12,colour="white"),
364     legend.position = "none",
365     panel.background = element_rect
366       (fill="white", colour="white
367
368   ),
369   panel.grid.major = element_
370     blank(),
371   panel.grid.minor = element_
372     blank()
373 )
374
375 margin <- 0.002 * nrow(d) +
376   0.00001 * nrow(d)^2 - 0.12
377 p2 <- p2 + theme(
378   plot.margin = unit(c
379     (0.25,0,0.25,0), "lines") # r:
380     -0.75
381 )
382
383 grid.newpage()
384 pushViewport(viewport(layout=
385   grid.layout(1,2, width=c(1,5))
386 ))
387 print(p, vp= viewport(layout.pos.
388   row=1, layout.pos.col=2) )
389 print(p2, vp= viewport(layout.pos.
390   row=1, layout.pos.col=1) )
391 } else {
392   print(p)
393 }
394
395 if (
396   is.na(dev.list()["pdf"])
397   && is.na(dev.list()["postscript"
398     ])
399   && is.na(dev.list()["cairo-pdf"])
400   && is.na(dev.list()["cairo-ps"])
401 ){
402   if ( grepl("darwin", R.version$
403     platform) ){
404     quartzFonts(HiraKaku=quartzFont
405       (rep("Meiryō UI",4)))
406     grid.gedit("GRID.text", grep=
407       TRUE, global=TRUE, gp=
408       gpar(fontfamily="HiraKaku"))
409   }
410 }
411
412 detach("package:ggdendro", unload
413   =T)
414
415 # for clickable image map
416 exp <- (y_max - y_min ) * 0.02
417 coord <- cbind(
418   (1 / 6 + 5 / 6 * -1 * (y_min -
419     exp) / ( (y_max + exp) - (y_
420     min - exp) )) * 1.03,
421   1:length(ddata$labels$text) /
422     length(ddata$labels$text)
423 )
424
425 rownames(coord) <-
426   labels[ as.numeric( as.vector(
427     ddata$labels$text ) ) ]

```

A. 3 多次元尺度法を作成するソースコード

多次元尺度法を作成するソースコード A.3 をしめす。

```

ソースコード A. 3: taji.r
1 d <- NULL
2 d <- matrix( c(1,...Abbreviated...,0)
, byrow=T, nrow=190, ncol=19 )
3 d <- d[,-1]
4 colnames(d) <- c("desk",...省略..., "
outdoor")
5 doc.length_mtr <- matrix( c(
70,18,...省略...,45,17), ncol=2,
byrow=T)
6 colnames(doc.length_mtr) <- c("
length.c", "length.w")
7 color_universal_design <- 1
8 d <- t(d)
9 # END: DATA
10
11 library(amen)
12 check4mds <- function(d){
13   jm <- as.matrix(Dist(d, method=
"binary"))
14   jm[upper.tri(jm,diag=TRUE)] <-
NA
15   if ( length( which(jm==0, arr.ind
=TRUE) ) ){
16     return( which(jm==0, arr.ind=
TRUE)[,1][1] )
17   } else {
18     return( NA )
19   }
20 }
21
22 while ( is.na(check4mds(d)) == 0 ){
23   n <- check4mds(d)
24   print( paste( "Dropped object:",
row.names(d)[n] ) )
25   d <- d[-n,]
26 }
27 dj <- Dist(d,method="binary")
28 random_starts <- 1
29 dim_n <- 2
30 method_mds <- "K"
31
32 if (method_mds == "K"){
33   # Kruskal
34   library(MASS)
35   c <- isoMDS(dj, k=dim_n, maxit
=3000, tol=0.000001)
36   if (random_starts == 1){
37     print("Running random starts
...")
38     set.seed(123)
39     for (i in 1:1000){ # 200sec
40       if (dim_n == 1){
41         init <- cbind( rnorm(
nrow(d)) )
42       } else if (dim_n == 2){
43         init <- cbind( rnorm(
nrow(d)), rnorm(nrow
(d)) )
44       } else if (dim_n == 3){
45         init <- cbind( rnorm(
nrow(d)), rnorm(nrow
(d)), rnorm(nrow(d)) )
46       } else {
47         warn("Error: invalid
dimesion number!")
48       }
49       ct <- isoMDS(dj, y=init, k=
dim_n, maxit=3000, tol
=0.000001, trace=F)
50       if (ct$stress < c$stress){
51         c <- ct
52         print( paste("random
start #", i, ":", c$
stress, sep=""))
53       }
54     }
55   }
56   cl <- c$points
57 } else if (method_mds == "S"){
58   #Sammon
59   library(MASS)
60   c <- sammon(dj, k=dim_n, niter
=3000, tol=0.000001)
61   if (random_starts == 1){
62     print("Running random starts
...")
63     set.seed(123)
64     for (i in 1:1000){ # 200sec
65       if (dim_n == 1){
66         init <- cbind( rnorm(
nrow(d)) )
67       } else if (dim_n == 2){
68         init <- cbind( rnorm(
nrow(d)), rnorm(nrow
(d)) )
69       } else if (dim_n == 3){
70         init <- cbind( rnorm(
nrow(d)), rnorm(nrow
(d)), rnorm(nrow(d)) )
71       } else {
72         warn("Error: invalid
dimesion number!")
73       }
74       ct <- sammon(dj, y=init, k=
dim_n, niter=3000, tol
=0.000001, trace=F)
75       if (ct$stress < c$stress){
76         c <- ct
77         print( paste("random

```

```

      start #", i, ": ", c$
      stress, sep=""))
78   }
79   }
80 }
81 cl <- c$points
82 } else if (method_mds == "C"){
83   # Classical
84   c <- cmdscale(dj, k=dim_n)
85   cl <- c
86 } else if (method_mds == "SM"){
87   # SMACOF
88   library(smacof)
89   c <- mds(dj, ndim=dim_n, type="
      ordinal", itmax=3000)
90   if (random_starts == 1){
91     print("Running random starts
      ...")
92     set.seed(123)
93     for (i in 1:200){ # 200 -> 246sec
94       run <- mds(dj, ndim=dim_n,
        type="ordinal", init = "
        random", itmax=3000)
95       if (run$stress < c$stress){
96         c <- run
97         print( paste("random start
          #", i, ": ", c$stress, sep=
            ""))
98       }
99     }
100   }
101   cl <- c$conf
102 }
103 save(d,cl,dim_n, file="C:/
      khcoder3/config/R-bridge/
      khc6_word_mds" )
104
105 use_alpha <- 1
106
107 if ( exists("saving_emf") ||
      exists("saving_eps" ) ){
108   use_alpha <- 0
109 }
110 plot_mode <- "color"
111 font_size <- 1
112 n_cls <- 4
113 cls_raw <- 0
114 name_dim <- '\u6b21\u5143'
115 name_dim1 <- paste(name_dim,'1')
116 name_dim2 <- paste(name_dim,'2')
117 name_dim3 <- paste(name_dim,'3')
118 fix_asp <- 0
119 name_dim <- '\u6b21\u5143'
120 text_font <- 1
121 bubble <- 1
122 bubble_size <- 100
123
124 ylab_text <- ""
125 if ( dim_n == 1 ){
126   name_dim2 <- name_dim1
127   cl <- cbind(cl[,1],cl[,1])
128 }
129
130 col_base <- "mediumaquamarine"
131 bty <- "1"
132
133 if ( exists("bubble_size") == F ) {
134   bubble_size <- 100
135 }
136 if ( exists("bs_fixed") == F ) {
137   bubble_size <- bubble_size / 1
138   bs_fixed <- 1
139 }
140
141 if (n_cls > 0){
142   if (nrow(d) < n_cls){
143     n_cls <- nrow(d)
144   }
145
146   if (cls_raw == 1){
147     djj <- dj
148   } else {
149     djj <- dist(cl,method="euclid")
150   }
151
152   if (
153     ( as.numeric( R.Version())$
154       major ) >= 3 )
155     && ( as.numeric( R.Version())$
156       minor ) >= 1.0)
157   ){ # >= R 3.1.0
158     hcl <- hclust(djj,method="ward.
159       D2")
160   } else { # <= R 3.0
161     djj <- djj^2
162     hcl <- hclust(djj,method="ward"
163       )
164     #hcl$height <- sqrt( hcl$height )
165   }
166 }
167 b_size <- NULL
168
169 for (i in rownames(cl)){
170   if ( is.na(i) || is.null(i) || is.nan(i) )
171     {
172       b_size <- c( b_size, 1 )
173     } else {
174       b_size <- c( b_size, sum( d[i,] ) )
175     }
176 }
177
178 if (plot_mode == "color") {
179   png_width <- 822
180   png_height <- 640
181   if ( png_width > png_height ){
182     png_width <- png_width - 0.16
183       * 1 * bubble_size / 100 * png_
184       width
185   }
186 }

```

```

182 dpi <- 72 * min(png_width, png_
      width) / 640 * 1
183 p_size <- 12 * dpi / 72;
184 png("temp.png", width=png_width,
      height=png_height, unit="px",
      pointsize=p_size)
185
186 #if ( exists("PERL_font_family") ){
187 # par(family=PERL_font_family)
188 #}
189
190 plot(cl)
191 library(maptools)
192 labcd <- pointLabel(
193   x=cl[,1],
194   y=cl[,2],
195   labels=rownames(cl),
196   cex=font_size,
197   offset=0,
198   doPlot=F
199 )
200
201 xorg <- cl[,1]
202 yorg <- cl[,2]
203 cex <- font_size
204 segs <- NULL
205
206 if ( length(xorg) < 300 ) {
207   library(wordcloud)
208
209   # fix for "wordlayout" function
210   filename <- tempfile()
211   writeLines("wordlayout <-
      function (x, y, words, cex =
        1, rotate90 = FALSE, xlim = c
        (-Inf,
212   Inf), ylim = c(-Inf, Inf),
      tstep = 0.1, rstep = 0.1,
      ...)
213 {
214   tails <- \"g|j|p|q|y\"
215   n <- length(words)
216   sdx <- sd(x, na.rm = TRUE)
217   sdy <- sd(y, na.rm = TRUE)
218   iterations <- 0
219   if (sdx == 0)
220     sdx <- 1
221   if (sdy == 0)
222     sdy <- 1
223   if (length(cex) == 1)
224     cex <- rep(cex, n)
225   if (length(rotate90) == 1)
226     rotate90 <- rep(rotate90, n)
227   boxes <- list()
228   for (i in 1:length(words)) {
229     rotWord <- rotate90[i]
230     r <- 0
231     theta <- runif(1, 0, 2 * pi)
232     x1 <- xo <- x[i]
233     y1 <- yo <- y[i]
234     wid <- strwidth(words[i],

```

```

      cex = cex[i], ...)
235   ht <- strheight(words[i],
      cex = cex[i], ...)
236   if (grepl(tails, words[i]))
237     ht <- ht + ht * 0.2
238   if (rotWord) {
239     tmp <- ht
240     ht <- wid
241     wid <- tmp
242   }
243   isOverlaped <- TRUE
244   while (isOverlaped) {
245     if (!.overlap(x1 - 0.5 *
246       wid, y1 - 0.5 * ht, wid,
247       ht, boxes) && x1 - 0.5 *
248       wid > xlim[1] && y1
249       -
250       0.5 * ht > ylim[1] && x1
251       + 0.5 * wid < xlim[2]
252       &&
253       y1 + 0.5 * ht < ylim[2])
254     {
255       boxes[[length(boxes) +
256         1]] <- c(x1 - 0.5 *
257         wid,
258         y1 - 0.5 * ht, wid, ht)
259       isOverlaped <- FALSE
260     }
261   else {
262     theta <- theta + tstep
263     r <- r + rstep * tstep /
264       (2 * pi)
265     x1 <- xo + sdx * r * cos
266       (theta)
267     y1 <- yo + sdy * r * sin
268       (theta)
269     iterations <- iterations
270       + 1
271     if (iterations > 500000){
272       boxes[[length(boxes) +
273         1]] <- c(x1 - 0.5 *
274         wid,
275         y1 - 0.5 * ht, wid, ht)
276       isOverlaped = FALSE
277     }
278   }
279 }
280
281 print( paste(\"iterations: \",
282   iterations) )
283 result <- do.call(rbind,
284   boxes)
285 colnames(result) <- c(\"x\",
286   \"y\", \"width\", \"ht\")
287 rownames(result) <- words
288 result
289 }
290
291 ", filename)
292 insertSource(filename, package="
293   wordcloud", force=FALSE)
294
295

```

```

276
277   nc <- wordlayout(
278     labcd$x,
279     labcd$y,
280     rownames(cl),
281     cex=cex * 1.25,
282     xlim=c( par( "usr" )[1], par( "
283       usr" )[2] ),
284     ylim=c( par( "usr" )[3], par( "
285       usr" )[4] )
286   )
287   xlen <- par("usr")[2] - par("
288     usr")[1]
289   ylen <- par("usr")[4] - par("
290     usr")[3]
291   for (i in 1:length(rownames(cl))
292     ){
293     x <- ( nc[i,1] + .5 * nc[i,3] -
294       labcd$x[i] ) / xlen
295     y <- ( nc[i,2] + .5 * nc[i,4] -
296       labcd$y[i] ) / ylen
297     dst <- sqrt( x^2 + y^2 )
298     if ( dst > 0.05 ){
299       segs <- rbind(
300         segs,
301         c(
302           nc[i,1] + .5 * nc[i,3],
303           nc[i,2] + .5 * nc[i,4],
304           xorg[i],
305           yorg[i]
306         )
307       )
308     }
309   }
310   xorg <- labcd$x
311   yorg <- labcd$y
312   labcd$x <- nc[,1] + .5 * nc[,3]
313   labcd$y <- nc[,2] + .5 * nc[,4]
314 }
315 dev.off()
316 }
317
318 library(grid)
319 library(ggplot2)
320
321 font_family <- "Meiryo UI"
322
323 if ( exists("PERL_font_family") ){
324   font_family <- PERL_font_family
325 }
326
327 if (use_alpha == 1){
328   alpha_value = 0.6
329 } else {
330   alpha_value = 1
331 }
332
333 if ( n_cls > 0 ){
334   cls_labels <- cutree(hcl, k=n_cls)

```

```

335   cls_labels <- formatC(cls_labels,
336     width=2,flag="0")
337
338   cls_labels <- paste(cls_labels, " "
339     )
340 } else {
341   cls_labels <- "cluster 1"
342 }
343
344 cl2 <- data.frame(
345   d1 = cl[,1],
346   d2 = cl[,2],
347   s = b_size,
348   col_f = cls_labels,
349   lx = labcd$x,
350   ly = labcd$y,
351   labels = rownames(cl),
352   stringsAsFactors = F
353 )
354
355 g <- ggplot()
356
357 # Plot
358 if ( bubble == 1 ){
359   g <- g + geom_point(
360     data=cl2,
361     aes(x=d1, y=d2, size=s, colour=
362       col_f, fill=col_f),
363     shape=21,
364     colour="gray40",
365     alpha=alpha_value
366   )
367   g <- g + scale_size_area(
368     max_size= 30 * bubble_size / 100,
369     guide = guide_legend(
370       title = "Frequency:",
371       override.aes = list(colour="
372         black", alpha=1),
373       label.hjust = 1,
374       order = 2
375     )
376   )
377 } else {
378   if ( n_cls > 0 ){
379     g <- g + geom_point(
380       data=cl2,
381       aes(x=d1, y=d2, colour=col_f,
382         fill=col_f),
383       size=5.5,
384       shape=21,
385       colour="gray40",
386       alpha=alpha_value
387     )
388   } else {
389     g <- g + geom_point(
390       data=cl2,
391       aes(x=d1, y=d2),
392       size=2,
393       shape=16,
394       colour="mediumaquamarine"
395     )

```

```

386   }
387 }
388
389 if ( n_cls > 0 ){
390   g <- g + scale_fill_brewer(
391     palette = "Set3",
392     guide = guide_legend(
393       title = "Cluster:",
394       override.aes = list(size=5.5,
395         alpha=1, shape=22),
396       keyheight = unit(1.25,"line"),
397       ncol=2,
398       order = 1
399     )
400   } else {
401     g <- g + scale_fill_brewer(
402       palette = "Set3",
403       guide = "none"
404     )
405   }
406
407   # Text
408   if (plot_mode == "color") {
409     if (text_font == 1){
410       face <- "plain"
411     } else {
412       face <- "bold"
413     }
414     g <- g + geom_text(
415       data=cl2,
416       aes(x=lx,y=ly,label=labels),
417       size=4,
418       colour="black",
419       family=font_family,
420       fontface=face
421     )
422     if (length(segs) > 0){
423       colnames(segs) <- c("x1", "y1",
424         "x2", "y2")
425       segs <- as.data.frame(segs)
426       g <- g + geom_segment(
427         aes(x=x1, y=y1, xend=x2, yend
428           =y2),
429         data=segs,
430         colour="gray60"
431       )
432     }
433
434     # Appearance / Theme
435     g <- g + labs(x=name_dim1, y=
436       name_dim2)
437     g <- g + theme_classic(base_family=
438       font_family)
439     g <- g + theme(
440       legend.key = element_rect(colour =
441         "transparent"),
442       axis.line.x = element_line(colour =
443         "black", size=0.5),

```

```

440     axis.line.y = element_line(colour =
441       "black", size=0.5),
442     axis.title.x = element_text(face="
443       plain", size=11, angle=0),
444     axis.title.y = element_text(face="
445       plain", size=11, angle=90),
446     axis.text.x = element_text(face="
447       plain", size=11, angle=0),
448     axis.text.y = element_text(face="
449       plain", size=11, angle=0),
450     legend.title = element_text(face="
451       bold", size=11, angle=0),
452     legend.text = element_text(face="
453       plain", size=11, angle=0),
454     plot.margin = margin(6, 6, 6, 0, "
455       pt")
456   )
457
458   # fix range
459   out_coord <- cbind( cl2$lx, cl2$ly )
460   rownames(out_coord) <- cl2$labels
461   xlimv <- c(
462     min( out_coord[,1] ) - 0.04 * ( max
463       ( out_coord[,1] ) - min( out_
464         coord[,1] ) ),
465     max( out_coord[,1] ) + 0.04 * (
466       max( out_coord[,1] ) - min(
467         out_coord[,1] ) )
468   )
469   ylimv <- c(
470     min( out_coord[,2] ) - 0.04 * ( max
471       ( out_coord[,2] ) - min( out_
472         coord[,2] ) ),
473     max( out_coord[,2] ) + 0.04 * (
474       max( out_coord[,2] ) - min(
475         out_coord[,2] ) )
476   )
477
478   # aspect ratio
479   if (fix_asp == 1){
480     g <- g + coord_fixed(
481       xlim=xlimv,
482       ylim=ylimv,
483       expand = F
484     )
485   } else {
486     g <- g + coord_cartesian(
487       xlim=xlimv,
488       ylim=ylimv,
489       expand = F
490     )
491   }
492
493   # coordinates for saving
494   add <- -1 * xlimv[1]
495   div <- add + xlimv[2]
496   out_coord[,1] <- ( out_coord[,1] +
497     add ) / div
498
499   add <- -1 * ylimv[1]
500   div <- add + ylimv[2]

```

```

484 out_coord[,2] <- ( out_coord[,2] +
      add ) / div
485
486 # fixing width of legends to 22%
487 library(grid)
488 library(gtable)
489 g <- ggplotGrob(g)
490
491 if ( ( n_cls == 0 ) && ( bubble == 0
      ) ){
492   saving_file <- 1
493 }
494
495 if ( exists("saving_file") ){
496   if ( saving_file == 0 ){
497     target_legend_width <-
498       convertX(
499         unit( image_width * 0.22, "in"
500             ),
501         "mm"
502       )
503     if ( as.numeric( substr(
504       packageVersion("ggplot2"), 1,
505       3 ) ) <= 2.1 ){ # ggplot2 <=
506       2.1.0
507       diff_mm <- diff( c(
508         convertX( g$widths[5], "mm" ),
509         target_legend_width
510       ) )
511       if ( diff_mm > 0 ){
512         g <- gtable_add_cols(g, unit(
513           diff_mm, "mm" ))
514       }
515     } else { # ggplot2 >= 2.2.0
516       diff_mm <- diff( c(
517         convertX( g$widths[7], "mm",
518           valueOnly=T ) +
519         convertX( g$widths[8], "
520           mm", valueOnly=T ),
521         target_legend_width
522       ) )
523       if ( diff_mm > 0 ){
524         print(diff_mm)
525         g <- gtable_add_cols(g, unit(
526           diff_mm, "mm" ))
527       }
528     }
529   }
530 }
531 # fixing width of left spaces to 4.25
532 char
533 if ( as.numeric( substr(
534   packageVersion("ggplot2"), 1, 3 )
535 ) <= 2.1 ){ # ggplot2 <= 2.1.0
536   diff_char <- diff( c(
537     convertX( g$widths[1] + g$widths
538       [2] + g$widths[3], "char" ),
539     unit(4.25, "char")
540   ) )
541   if ( diff_char > 0 ){
542     g <- gtable_add_cols(g, unit(diff
543       _char, "char"), pos=0)
544   }
545 }
546 grid.draw(g)

```

A. 4 対応分析を作成するソースコード

対応分析を作成するソースコード A.4 をしめす。

```

      ソースコード A. 4: taiou.r
1 d <- NULL
2 d <- matrix( c(1,...省略...,0), byrow
      =T, nrow=190, ncol=19 )
3 d <- d[,-1]
4 colnames(d) <- c("desk",...省略...,
      "outdoor")
5 doc_length_mtr <- matrix( c(
      70,18,...省略...45,17), ncol=2,
      byrow=T)
6 colnames(doc_length_mtr) <- c("
      length_c", "length_w")
7 color_universal_design <- 1
8
9 v_count <- 0
10 v_pch <- NULL
11
12 if ( length(v_pch) == 0 ) {
13   v_pch <- 3
14   v_count <- 1
15 }
16 if ( length(v_pch) > 1 ){ v_pch <-
      v_pch[rowSums(d) > 0] }
17 doc_length_mtr <- subset(doc_
      length_mtr, rowSums(d) > 0)
18 d <- subset(d, rowSums(d) > 0)
19 n_total <- doc_length_mtr[,2]
20 d <- t(d)
21 d <- subset(d, rowSums(d) > 0)
22 d <- t(d)
23 # END: DATA
24 text_font <- 1
25 r_max <- 150
26 zoom_factor <- 0
27 d_x <- 1
28 d_y <- 2

```



```

138 AB[y < 0] <- AB[y < 0] * -1
139
140 cbind(OB, AB)
141 }
142
143 axp <- NULL
144 if (zoom_factor >= 1 ){
145   scaling <- "none"
146   axp <- c(0,0,1)
147
148   r <- log_conv( c$cscore[d_x], c$
149                 cscore[d_y], zoom_factor )
149   c$cscore[d_x] <- r[1]
150   c$cscore[d_y] <- r[2]
151
152   r <- log_conv( c$rscore[d_x], c$
153                 rscore[d_y], zoom_factor )
153   c$rscore[d_x] <- r[1]
154   c$rscore[d_y] <- r[2]
155 }
156
157 # Scaling
158 asp <- 0
159 if (scaling == "sym"){
160   for (i in 1:d_max){
161     c$cscore[i] <- c$cscore[i] * c$
162       cor[i]
163     c$rscore[i] <- c$rscore[i] * c$
164       cor[i]
165   }
166   asp <- 1
167 } else if (scaling == "symbi"){
168   for (i in 1:d_max){
169     c$cscore[i] <- c$cscore[i] * sqrt
170       ( c$cor[i] )
171     c$rscore[i] <- c$rscore[i] * sqrt
172       ( c$cor[i] )
173   }
174   asp <- 1
175 }
176
177 k <- c$cor^2
178 txt <- cbind( 1:length(k), round(k
179             ,4), round(100*k / sum(k),2) )
180 colnames(txt) <- c(name_dim,name
181                   _eig,name_exp)
182 print( txt )
183 inertias <- round(k,4)
184 k <- round(100*k / sum(k),2)
185 font_size <- 1
186 resize_vars <- 1
187 bubble_size <- 100
188 labcd <- NULL
189
190 plot_mode <- "color"
191
192 library(ggplot2)
193 font_family <- "Meiryo UI"

```

```

191
192 if ( exists("PERL_font_family") ){
193   font_family <- PERL_font_family
194 }
195
196 if ( exists("bs_fixed") == F ) {
197   bubble_size <- bubble_size / 1.3
198   bs_fixed <- 1
199 }
200
201 if (biplot == 1 && plot_mode != "
202     vars"){
203   cb <- rbind(
204     cbind(c$cscore[d_x], c$cscore[d_
205           y], ptype),
206     cbind(c$rscore[d_x], c$rscore[d_y
207           ], v_pch)
208   )
209 } else if (plot_mode == "vars") {
210   cb <- cbind(c$rscore[d_x], c$
211               rscore[d_y], v_pch)
212 } else {
213   cb <- cbind(c$cscore[d_x], c$
214               cscore[d_y], ptype)
215 }
216
217 if ( (is.null(labcd) && plot_mode !
218      = "dots" ) || plot_mode == "
219      vars"){
220
221   png_width <- 640
222   png_height <- 640
223   png_width <- png_width - 0.16 *
224     1.3 * bubble_size / 100 * png_
225     width
226   dpi <- 72 * min(png_width, png_
227                   height) / 640 * 1.3
228   p_size <- 12 * dpi / 72;
229   png("temp.png", width=png_width,
230       height=png_height, unit="px",
231       pointsize=p_size)
232
233   #if ( exists("PERL_font_family") ){
234   # par(family=PERL_font_family)
235   #}
236
237 plot(
238   x=c(c$cscore[d_x],c$rscore[d_x]),
239   y=c(c$cscore[d_y],c$rscore[d_y]),
240   asp=asp
241 )
242
243 library(maptools)
244 labcd <- pointLabel(
245   x=cb[1],
246   y=cb[2],
247   labels=rownames(cb),
248   cex=font_size,
249   offset=0,
250   doPlot=F
251 )

```

```

240
241 xorg <- cb[,1]
242 yorg <- cb[,2]
243 #cex <- 1
244
245 n_words_chk <- c( length(c$cscore
246   [,d_x]) )
247 if (flt > 0) {
248   n_words_chk <- c(n_words_chk,
249     flt)
250 }
251 if (flw > 0) {
252   n_words_chk <- c(n_words_chk,
253     flw)
254 }
255 if (
256   ( (biplot == 0) && (min(n_
257     words_chk) < 300) )
258   || (
259     (biplot == 1)
260     && ( min(n_words_chk) < 300
261       )
262     && ( length(c$rscore[,d_x]) <
263       r_max )
264   )
265 ){
266   library(wordcloud)
267
268   # fix for "wordlayout" function
269   filename <- tempfile()
270   writeLines("wordlayout <-
271     function (x, y, words, cex =
272       1, rotate90 = FALSE, xlim = c
273       (-Inf,
274       Inf), ylim = c(-Inf, Inf),
275       tstep = 0.1, rstep = 0.1,
276       ...)
277 {
278   tails <- "\"g|j|p|q|y\"
279   n <- length(words)
280   sdx <- sd(x, na.rm = TRUE)
281   sdy <- sd(y, na.rm = TRUE)
282   iterations <- 0
283   if (sdx == 0)
284     sdx <- 1
285   if (sdy == 0)
286     sdy <- 1
287   if (length(cex) == 1)
288     cex <- rep(cex, n)
289   if (length(rotate90) == 1)
290     rotate90 <- rep(rotate90, n)
291   boxes <- list()
292   for (i in 1:length(words)) {
293     rotWord <- rotate90[i]
294     r <- 0
295     theta <- runif(1, 0, 2 * pi)
296     x1 <- xo <- x[i]
297     y1 <- yo <- y[i]
298     wid <- strwidth(words[i],

```

```

299     cex = cex[i], ...)
300     ht <- strheight(words[i],
301       cex = cex[i], ...)
302     if (grepl(tails, words[i]))
303       ht <- ht + ht * 0.2
304     if (rotWord) {
305       tmp <- ht
306       ht <- wid
307       wid <- tmp
308     }
309     isOverlaped <- TRUE
310     while (isOverlaped) {
311       if (!.overlap(x1 - 0.5 *
312         wid, y1 - 0.5 * ht, wid,
313         ht, boxes) && x1 - 0.5 *
314         wid > xlim[1] && y1
315         -
316         0.5 * ht > ylim[1] && x1
317         + 0.5 * wid < xlim[2]
318         &&
319         y1 + 0.5 * ht < ylim[2])
320       {
321         boxes[[length(boxes) +
322           1]] <- c(x1 - 0.5 *
323             wid,
324             y1 - 0.5 * ht, wid, ht)
325         isOverlaped <- FALSE
326       }
327     }
328   }
329   else {
330     theta <- theta + tstep
331     r <- r + rstep * tstep /
332       (2 * pi)
333     x1 <- xo + sdx * r * cos
334       (theta)
335     y1 <- yo + sdy * r * sin
336       (theta)
337     iterations <- iterations
338       + 1
339     if (iterations > 500000){
340       boxes[[length(boxes) +
341         1]] <- c(x1 - 0.5 *
342           wid,
343           y1 - 0.5 * ht, wid, ht)
344       isOverlaped = FALSE
345     }
346   }
347 }
348 }
349 print( paste("\ iterations: ",
350   iterations) )
351 result <- do.call(rbind,
352   boxes)
353 colnames(result) <- c("\x\",
354   "\y\", \"width\", \"ht\")
355 rownames(result) <- words
356 result
357 }
358 ", filename)
359 insertSource(filename, package="
360   wordcloud", force=FALSE)
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400

```

```

331
332   nc <- wordlayout(
333     labcd$x,
334     labcd$y,
335     rownames(cb),
336     cex=cex * 1.05,
337     xlim=c( par( "usr" )[1], par( "
338       usr" )[2] ),
339     ylim=c( par( "usr" )[3], par( "
340       usr" )[4] )
341   )
342   xlen <- par("usr")[2] - par("
343     usr")[1]
344   ylen <- par("usr")[4] - par("
345     usr")[3]
346   segs <- NULL
347   for (i in 1:length( rownames(cb)
348     )){
349     x <- ( nc[i,1] + .5 * nc[i,3] -
350       labcd$x[i] ) / xlen
351     y <- ( nc[i,2] + .5 * nc[i,4] -
352       labcd$y[i] ) / ylen
353     dst <- sqrt( x^2 + y^2 )
354     if ( dst > 0.05 ){
355       segs <- rbind(
356         segs,
357         c(
358           nc[i,1] + .5 * nc[i,3], nc[i
359             ,2] + .5 * nc[i,4],
360           xorg[i], yorg[i]
361         )
362       )
363     }
364   }
365   xorg <- labcd$x
366   yorg <- labcd$y
367   labcd$x <- nc[,1] + .5 * nc[,3]
368   labcd$y <- nc[,2] + .5 * nc[,4]
369 }
370 text(labcd$x, labcd$y, rownames(
371   cb))
372 dev.off()
373 }
374
375 b_size <- NULL
376 for (i in rownames(c$cscore)){
377   if ( is.na(i) || is.null(i) || is.nan(i) )
378   {
379     b_size <- c( b_size, 1 )
380   } else {
381     b_size <- c( b_size, sum( d[,i] ) )
382   }
383 }
384 col_bg_words <- NA
385 col_bg_vars <- NA

```

```

386 if (plot_mode == "color"){
387   col_dot_words <- "#00CED1"
388   col_dot_vars <- "#FF6347"
389   if ( use_alpha == 1 ){
390     col_bg_words <- "#48D1CC"
391     col_bg_vars <- "#FFA07A"
392
393     rgb <- col2rgb(col_bg_words) /
394       255
395     col_bg_words <- rgb( rgb[1],
396       rgb[2], rgb[3])
397
398     rgb <- rgb * 0.5
399     col_dot_words <- "#87CAC6" #
400       <- rgb( rgb[1], rgb[2], rgb[3])
401
402     rgb <- col2rgb(col_bg_vars) /
403       255
404     col_bg_vars <- rgb( rgb[1], rgb
405       [2], rgb[3])
406   }
407 }
408 if (plot_mode == "gray"){
409   col_dot_words <- "gray55"
410   col_dot_vars <- "gray30"
411 }
412 if (plot_mode == "vars"){
413   col_dot_words <- "#ADD8E6"
414   col_dot_vars <- "red"
415 }
416 if (plot_mode == "dots"){
417   col_dot_words <- "black"
418   col_dot_vars <- "black"
419 }
420 g <- ggplot()
421 df.words <- data.frame(
422   x = c$cscore[,d_x],
423   y = c$cscore[,d_y],
424   size = b_size,
425   type = ptype
426 )
427 df.words.sub <- subset(df.words,
428   type==2)
429 df.words <- subset(df.words, type
430   ==1)
431 if (bubble_plot == 1){
432   g <- g + geom_point(
433     data=df.words,
434     aes(x=x, y=y, size=size),
435     shape=21,
436     #colour = NA,
437     fill = col_bg_words,
438     alpha=0.15
439   )

```

```

436
437 g <- g + geom_point(
438   data=df.words,
439   aes(x=x, y=y, size=size),
440   shape=21,
441   colour = col_dot_words,
442   fill = NA,
443   alpha=1,
444   show.legend = F
445 )
446
447 g <- g + scale_size_area(
448   max_size= 30 * bubble_size / 100,
449   guide = guide_legend(
450     title = "Frequency:",
451     override.aes = list(colour="
452       black", fill=NA, alpha=1),
453     label.hjust = 1,
454     order = 2
455   )
456 ) else {
457   g <- g + geom_point(
458     data=df.words,
459     aes(x=x, y=y),
460     size = 2,
461     shape=16,
462     colour = col_dot_words,
463     alpha=1,
464     show.legend = F
465   )
466 }
467
468 if ( nrow(df.words.sub) > 0 ){
469   g <- g + geom_point(
470     data=df.words.sub,
471     aes(x=x, y=y),
472     shape=19,
473     size=2,
474     colour = "#ADD8E6",
475     alpha=1,
476     show.legend = F
477   )
478 }
479
480 if ( biplot == 1 ){
481   df.vars <- data.frame(
482     x = c$rscore[,d_x],
483     y = c$rscore[,d_y],
484     size = n_total * max(b_size) /
485       max(n_total) * 0.6,
486     type = v_pch
487   )
488
489   if ( (resize_vars == 1) && (bubble_
490     plot == 1) ) {
491     g <- g + geom_point(
492       data=df.vars,
493       aes(x=x, y=y, size=size, shape=
494         factor(type) ),
495       #colour = NA,

```

```

493       fill = col_bg_vars,
494       alpha=0.2,
495       show.legend = F
496     )
497
498   g <- g + geom_point(
499     data=df.vars,
500     aes(x=x, y=y, size=size, shape=
501       factor(type) ),
502     colour = col_dot_vars,
503     fill = NA,
504     alpha=1,
505     show.legend = F
506   )
507 } else {
508   g <- g + geom_point(
509     data=df.vars,
510     aes(x=x, y=y, shape=factor(
511       type) ),
512     colour = NA,
513     fill = col_bg_vars,
514     alpha=0.2,
515     size=3.5,
516     show.legend = F
517   )
518
519   g <- g + geom_point(
520     data=df.vars,
521     aes(x=x, y=y, shape=factor(
522       type) ),
523     colour = col_dot_vars,
524     fill = NA,
525     alpha=1,
526     size=3.5,
527     show.legend = F
528   )
529 }
530
531 g <- g + scale_shape_manual(
532   values = c(22:25,0-6)
533 )
534
535 # label colors
536 if (plot_mode == "color"){
537   #if (bubble_plot == 1){
538     col_txt_words <- "black"
539     col_txt_vars <- "#DC143C"
540   #} else {
541     # col_txt_words <- "black"
542     # col_txt_vars <- "#FF6347"
543   #}
544 }
545
546 if (plot_mode == "gray"){
547   col_txt_words <- "black"
548   col_txt_vars <- "black"
549 }
550
551 if (plot_mode == "vars"){
552   col_txt_words <- "black"

```

```

551   col_txt_vars <- "black"
552 }
553
554 if (plot_mode == "dots"){
555   col_txt_words <- NA
556   col_txt_vars <- NA
557 }
558
559 if ( text_font == 1 ){
560   font_face <- "plain"
561 } else {
562   font_face <- "bold"
563 }
564
565 if ( exists("df.labels.save ") == F
566   ){
567   df.labels.save <- data.frame(
568     x = labcd$x,
569     y = labcd$y,
570     labs = rownames(cb),
571     cols = cb[,3]
572   )
573 }
574
575 if (plot_mode != "dots") {
576   df.labels <- data.frame(
577     x = labcd$x,
578     y = labcd$y,
579     labs = rownames(cb),
580     cols = cb[,3]
581   )
582   if ( plot_mode == "gray" ){
583     df.labels.var <- subset(df.
584       labels, cols == 3)
585     df.labels <- subset(df.labels,
586       cols != 3)
587     g <- g + geom_label(
588       data=df.labels.var,
589       family=font_family,
590       fontface="bold",
591       #label.size=0.25,
592       label.padding=unit(1.8, "mm"),
593       colour="white",
594       fill="gray50",
595       #alpha=0.7,
596       aes(x=x, y=y,label=labs)
597     )
598     if ( (resize_vars == 0) || (bubble_
599       plot == 0) ) {
600       g <- g + geom_point(
601         data=df.vars,
602         aes(x=x, y=y, shape=factor(
603           type) ),
604         colour = col_dot_vars,
605         fill = NA,
606         alpha=1,
607         size=3.5,
608         show.legend = F
609       )
610     }
611   }
612 }
613
614 g <- g + geom_text(
615   data=df.labels,
616   aes(x=x, y=y,label=labs,colour=
617     factor(cols)),
618   size=4,
619   family=font_family,
620   fontface=font_face
621   #colour="black"
622 )
623
624 #label_legend <- guide_legend(
625 # title = "Labels:",
626 # key.theme = element_rect(colour
627 #   = "gray30"),
628 # override.aes = list(size=5),
629 # order = 1
630 #)
631 label_legend <- "none"
632
633 g <- g + scale_color_manual(
634   values = c(col_txt_words, col_txt_
635     vars, col_txt_vars),
636   breaks = c(1,3),
637   labels = c("Words / Codes", "
638     Variables"),
639   guide = label_legend
640 )
641
642 if ( exists("segs" ) ){
643   if ( is.null(segs) == F ){
644     colnames(segs) <- c("x1", "
645       y1", "x2", "y2")
646     segs <- as.data.frame(segs)
647     g <- g + geom_segment(
648       aes(x=x1, y=y1, xend=x2,
649         yend=y2),
650       data=segs,
651       colour="gray60"
652     )
653   }
654 }
655
656 if (plot_mode == "vars"){
657   labcd <- NULL
658 }
659
660 #if ( asp == 1 ){
661 # g <- g + coord_fixed()
662 #}
663
664 g <- g + labs(
665   x=paste(name_dim,d_x," (",
666     inertias[d_x]," ", k[d_x],"%)" ,
667     sep=""),
668   y=paste(name_dim,d_y," (",
669     inertias[d_y]," ", k[d_y],"%)" ,
670     sep="")

```

```

658 )
659 g <- g + theme_classic(base_family =
      font_family)
660 g <- g + theme(
661   legend.key = element_rect(colour =
      NA, fill= NA),
662   axis.line.x = element_line(colour =
      "black", size=0.5),
663   axis.line.y = element_line(colour =
      "black", size=0.5),
664   axis.title.x = element_text(face="
      plain", size=11, angle=0),
665   axis.title.y = element_text(face="
      plain", size=11, angle=90),
666   axis.text.x = element_text(face="
      plain", size=11, angle=0),
667   axis.text.y = element_text(face="
      plain", size=11, angle=0),
668   legend.title = element_text(face="
      bold", size=11, angle=0),
669   legend.text = element_text(face="
      plain", size=11, angle=0)
670 )
671 if (show_origin == 1){
672   line_color <- "gray30"
673
674   lim_chk <- ggplot_build(g)
675   xlims <- lim_chk$panel$ranges[[1]]
676     $x.range
677   ylims <- lim_chk$panel$ranges[[1]]
678     $y.range
679   if ( is.null(xlims) ){
680     xlims <- lim_chk$layout$panel_
681       ranges[[1]]$x.range
682     ylims <- lim_chk$layout$panel_
683       ranges[[1]]$y.range
684   }
685   if (zoom_factor >= 1){
686     g <- g + scale_x_continuous(
687       limits=xlims, expand=c(0,0),
688       breaks=c(0) )
689     g <- g + scale_y_continuous(
690       limits=ylims, expand=c(0,0),
691       breaks=c(0) )
692   } else {
693     g <- g + scale_x_continuous(
694       limits=xlims, expand=c(0,0)
695     )
696     g <- g + scale_y_continuous(
697       limits=ylims, expand=c(0,0)
698     )
699   }
700   m_x <- (xlims[2] - xlims[1]) * 0.03
701   m_y <- (ylims[2] - ylims[1]) * 0.03
702
703   g <- g + geom_segment(
704     aes(x = xlims[1], y = 0, xend = m
705        _x, yend = 0),

```

```

706     size=0.25,
707     linetype="dashed",
708     colour=line_color
709   )
710   g <- g + geom_segment(
711     aes(x = 0, y = ylims[1], xend = 0,
712        yend = m_y),
713     size=0.25,
714     linetype="dashed",
715     colour=line_color
716   )
717 } else {
718   if (zoom_factor >= 1){
719     g <- g + scale_x_continuous(
720       breaks=c(0) )
721     g <- g + scale_y_continuous(
722       breaks=c(0) )
723   }
724   # fix range
725   if ( exists("xlimv") == F ){
726     # for setting xlim & ylim
727     out_coord <- cbind(
728       c( df.labels.save$x, df.words$x),
729       c( df.labels.save$y, df.words$y)
730     )
731     xlimv <- c(
732       min( out_coord[,1] ) - 0.04 * (
733         max( out_coord[,1] ) - min(
734           out_coord[,1] ) ),
735       max( out_coord[,1] ) + 0.04 * (
736         max( out_coord[,1] ) - min(
737           out_coord[,1] ) )
738     )
739     ylimv <- c(
740       min( out_coord[,2] ) - 0.04 * (
741         max( out_coord[,2] ) - min(
742           out_coord[,2] ) ),
743       max( out_coord[,2] ) + 0.04 * (
744         max( out_coord[,2] ) - min(
745           out_coord[,2] ) )
746     )
747   }
748   # for saving
749   out_coord <- cbind(
750     df.labels.save$x,
751     df.labels.save$y
752   )
753   rownames(out_coord) <- df.
754     labels.save$labs
755 }
756 # aspect ratio
757 if (asp == 1){
758   g <- g + coord_fixed(
759     xlim=xlimv,
760     ylim=ylimv,
761     expand = F
762   )

```

```

745 } else {
746   g <- g + coord_cartesian(
747     xlim=xlimv,
748     ylim=ylimv,
749     expand = F
750   )
751 }
752
753 # coordinates for saving
754 if (plot_mode == "color"){
755   df.labels.save <- subset(df,
756     labels.save, cols != 3)
757   out_coord <- cbind(
758     df.labels.save$x,
759     df.labels.save$y
760   )
761   rownames(out_coord) <- df.
762     labels.save$labs
763
764   add <- -1 * xlimv[1]
765   div <- add + xlimv[2]
766   out_coord[,1] <- ( out_coord[,1] +
767     add ) / div
768
769   add <- -1 * ylimv[1]
770   div <- add + ylimv[2]
771   out_coord[,2] <- ( out_coord[,2] +
772     add ) / div
773 }
774
775 # fixing width of legends to 22%
776 library(grid)
777 library(gtable)
778 g <- ggplotGrob(g)
779
780 if ( bubble_plot == 0 ){
781   saving_file <- 1
782 }
783
784 if ( exists("saving_file") ){
785   if ( saving_file == 0 ){
786     target_legend_width <-
787       convertX(
788         image_width * 0.22, "in"
789       ),
790     "mm"
791   )
792 }

```

```

786 if ( as.numeric( substr(
787   packageVersion("ggplot2"), 1,
788   3) ) <= 2.1 ){ # ggplot2 <=
789   2.1.0
790   diff_mm <- diff( c(
791     convertX( g$widths[5], "mm" ),
792     target_legend_width
793   ))
794   if ( diff_mm > 0 ){
795     g <- gtable_add_cols(g, unit(
796       diff_mm, "mm"))
797   }
798 } else { # ggplot2 >= 2.2.0
799
800   diff_mm <- diff( c(
801     convertX( g$widths[7], "mm",
802       valueOnly=T ) +
803     convertX( g$widths[8], "
804       mm", valueOnly=T ),
805     target_legend_width
806   ))
807   if ( diff_mm > 0 ){
808     print(diff_mm)
809     g <- gtable_add_cols(g, unit(
810       diff_mm, "mm"))
811   }
812 }
813 }
814 }
815
816 # fixing width of left spaces to 4.25
817 char
818
819 if ( as.numeric( substr(
820   packageVersion("ggplot2"), 1, 3)
821   ) <= 2.1 ){ # ggplot2 <= 2.1.0
822   diff_char <- diff( c(
823     convertX( g$widths[1] + g$widths
824       [2] + g$widths[3], "char" ),
825     unit(4.25, "char")
826   ))
827   if ( diff_char > 0 ){
828     g <- gtable_add_cols(g, unit(diff
829       _char, "char"), pos=0)
830   }
831 }
832
833 grid.draw(g)

```

A. 5 共起ネットワークを作成するソースコード

共起ネットワークを作成するソースコード A.5 をしめす。

ソースコード A. 5: kyoki.r

```

1 d <- NULL
2 d <- matrix( c(1,...省略...,0), byrow
3   =T, nrow=190, ncol=19 )
4 d <- d[,-1]

```

```

4 colnames(d) <- c("desk",...省略..., "
  outdoor")
5 doc_length_mtr <- matrix( c(
  70,18,...省略...,45,17), ncol=2,
  byrow=T)

```

```

6  colnames(doc_length_mtr) <- c("
    length_c", "length_w")
7  color_universal_design <- 1
8
9  d <- t(d)
10 # END: DATA
11 edges <- 0
12 th <- 0.2
13 cex <- 1
14 view_coef <- 0
15 fix_lab <- 1
16 use_freq_as_size <- 1
17 bubble_size <- 100
18 use_weight_as_width <- 0
19 smaller_nodes <- 0
20 text_font <- 1
21 min_sp_tree <- 0
22 min_sp_tree_only <- 0
23 cor_var <- 0
24 cor_var_darker <- 0
25 use_alpha <- 1
26 gray_scale <- 0
27 method_coef <- "binary"
28 com_method <- "com-b"
29
30 # Count frequency of each word
31 freq <- NULL
32 for (i in 1:length(rownames(d))) {
33   freq[i] = sum( d[i,] )
34 }
35
36 # Compute co-occurrence coefficient
37 if ( (exists("doc_length_mtr")) & ! (
    method_coef == "binary")){
38   leng <- as.numeric(doc_length_
    mtr[,2])
39   leng[leng==0] <- 1
40   d <- t(d)
41   d <- d / leng
42   d <- d * 1000
43   d <- t(d)
44 }
45 if (method_coef == "euclid"){ #
    standardize for each word
46   d <- t( scale( t(d) ) )
47 }
48
49 dr <- d
50 library(igraph)
51 d <- Dist(d,method=method_coef)
52
53 d <- as.matrix(d)
54 if ( method_coef == "euclid" ){
55   d <- max(d) - d
56   d <- d / max(d)
57 } else {
58   d <- 1 - d
59 }
60
61 # Delete unnecessary edges and
    standardize

```

```

62 if ( exists("com_method" ) ){
63   if (com_method == "twomode_c" ||
    com_method == "twomode_g" ){
64     d[1:n_words,] <- 0
65
66     std <- d[(n_words+1):nrow(d),1:
    n_words]
67     chkm <- std
68
69     std <- t(std)
70     std <- scale(std, center=T,
    scale=F)
71     std <- t(std)
72
73     if ( min(std[!is.na(std)]) < 0.0005
    ){
74       std <- std + ( 0.0005 - min(
    std[!is.na(std)]) );
75     }
76     std <- std / max(std[!is.na(std)
    ])
77
78     std[chkm == 0] <- 0
79     d[(n_words+1):nrow(d),1:n_words]
    <- std
80   }
81 }
82
83 # Make a graph
84 # For igraph > 1.0.0
85 library(igraph)
86 new_igraph <- 0
87 igraph_ver <- (
88   as.numeric( substr(sessionInfo()
    $otherPkgs$igraph$Version,
    1,1) ) * 10
89   + as.numeric( substr(sessionInfo()
    $otherPkgs$igraph$Version,
    3,3) )
90 )
91 if ( igraph_ver > 5 ){
92   new_igraph <- 1
93 }
94
95 n <- graph.adjacency(d, mode="
    lower", weighted=T, diag=F)
96 n <- igraph::set.vertex.attribute(
97   n,
98   "name",
99   (0+new_igraph):(length(d[1,])-1+
    new_igraph),
100   as.character( 1:length(d[1,]) )
101 )
102
103 # Prepare for deleting weak edges
104 el <- data.frame(
105   edge1 = get.edgelist(n,name=T)[,1],
106   edge2 = get.edgelist(n,name=T)[,2],
107   weight = igraph::get.edge.attribute(
    n, "weight"),
108   stringsAsFactors = FALSE

```

```

109 )
110
111 # Find a threshold value
112 if (th < 0){
113   if(edges > length(el[,1])){
114     edges <- length(el[,1])
115   }
116   th = quantile(
117     el$weight,
118     names = F,
119     probs = 1 - edges / length(el[,1])
120   )
121 }
122
123 # Delete weak edges and make a graph
124   again
125 el2 <- subset(el, el[,3] >= th)
126 if ( nrow(el2) == 0 ){
127   stop(message = "No edges to
128     draw!", call. = F)
129 }
130 n2 <- graph.edgelist(
131   matrix( as.matrix(el2)[,1:2], ncol
132     =2 ),
133   directed =F
134 )
135 n2 <- igraph::set.edge.attribute(
136   n2,
137   "weight",
138   (0+new_igraph):(length(get.
139     edgelist(n2)[,1])-1+new_igraph)
140
141   el2[,3]
142 )
143
144 if ( min_sp_tree_only == 1 ){
145   n2 <- minimum.spanning.tree(
146     n2,
147     weights = 1 - igraph::get.edge.
148       attribute(n2, "weight"),
149     algorithm="prim"
150   )
151 }
152
153 if (length(igraph::get.vertex.attribute
154   (n2,"name")) < 2){
155   com_method <- "none"
156 }
157
158 # Centrality
159 if ( com_method == "cnt-b" || com_
160   method == "cnt-d" || com_
161   method == "cnt-e"){
162   ccol <- NULL
163   if (com_method == "cnt-b"){ #
164     betweenness
165     ccol <- igraph::betweenness(
166       n2,
167       v=(0+new_igraph):(length(
168         igraph::get.vertex.attribute(
169           n2,"name"))-1+new_igraph
170         ),
171       directed=F
172     )
173   }
174   if (com_method == "cnt-d"){ #
175     degree
176     ccol <- igraph::degree(
177       n2,
178       v=(0+new_igraph):(length(
179         igraph::get.vertex.attribute(
180           n2,"name"))-1+new_igraph
181         )
182     )
183   }
184   if (com_method == "cnt-e"){ #
185     evcent
186     try(
187       ccol <- igraph::evcent(n2)$
188         vector,
189       silent = T
190     )
191   }
192   ccol_raw <- ccol
193
194   edg_col <- "gray55"
195   edg_lty <- 1
196 }
197
198 # Community detection
199 if (com_method == "com-b" || com_
200   method == "com-g" || com_
201   method == "com-r"){
202   merge_step <- function(n2, m){
203     for ( i in 1:( trunc( length( m )
204       / 2 ) ) ){
205       temp_csize <- community.to.
206         membership(n2, m,i)$csize
207       num_max <- max( temp_csize
208         )
209       num_alone <- sum( temp_csize
210         [ temp_csize == 1 ] )
211       num_cls <- length( temp_csize
212         [temp_csize > 1] )
213       #print( paste(i, "a", num_alone,
214         "max", num_max, "cls",
215         num_cls ) )
216       if (
217         num_max / length(igraph
218           ::get.vertex.attribute(
219             n2,"name")) >= 0.225
220         && num_max > num_alone
221         && num_cls < 12
222       ){
223         return(i)
224       }
225       if (num_max / length(igraph::

```

```

    get.vertex.attribute(n2,"
    name")) >= 0.4 ){
201     return(i-1)
202   }
203 }
204 return( trunc(length( m ) / 2) )
205 }
206 # For igraph > 1.0.0
207 if (com_method == "com-b"){ #
    Betweenness
208   com <- edge.betweenness.
    community(n2, directed=F)
209   if (igraph_ver < 10){
210     com_m <- community.to.
    membership(
211     n2, com$merges, merge_step(
    n2,com$merges)
212   )
213   com_m$membership <- com_m
    $membership + new_igraph
214 }
215 }
216 if (com_method == "com-g"){ #
    Modularity
217   com <- fastgreedy.community (
    n2, merges=TRUE,
    modularity=TRUE)
218   if (igraph_ver < 10){
219     com_m <- community.to.
    membership(
220     n2, com$merges, merge_step(
    n2,com$merges)
221   )
222   com_m$membership <- com_m
    $membership + new_igraph
223 }
224 }
225 if (com_method == "com-r"){ #
    Random walks
226   com <- walktrap.community(
    n2,
227   weights=igraph::get.edge.
    attribute(n2, "weight")
228 )
229   if (igraph_ver < 10){
230     com_m <- NULL
231     com_m$membership <- com$
    membership
232     com_m$size <- table(com$
    membership)
233   }
234 }
235 }
236 if (igraph_ver >= 10){
237   com_m <- NULL
238   com_m$membership <- as.
    vector( membership(com) )
239   com_m$size <- table(com_m$
    membership)
240 }
241 }
242 # Configure Edges

```

```

243   edg_lty <- NULL
244   edg_col <- NULL
245   for (i in 1:nrow(get.edgelist(n2,
    name=F))){
246     if (
247       com_m$membership[ get.
    edgelist(n2,name=F)[i,1]
    + 1 - new_igraph]
248     == com_m$membership[ get.
    edgelist(n2,name=F)[i,2] +
    1 - new_igraph]
249   ){
250     edg_col <- c( edg_col, "gray55
    " )
251     edg_lty <- c( edg_lty, 1 )
252   } else {
253     edg_col <- c( edg_col, "gray"
    )
254     edg_lty <- c( edg_lty, 3 )
255   }
256 }
257 }
258 }
259 }
260 if (com_method == "twomode_c" ||
    com_method == "twomode_g"){
261   if ( exists("var_select" ) ){
262     var_select_bak <- var_select
263   }
264 }
265 var_select <- substring(
266   colnames(d)[ as.numeric( igraph
    ::get.vertex.attribute(n2,"name
    " ) ) ],
267   1,
268   2
269 ) == "<>"
270 }
271 if (length(var_select[var_select==
    TRUE]) == 0 && exists("var_
    select_bak")){
272   var_select <- var_select_bak;
273 }
274 }
275 }
276 if (com_method == "twomode_c"){
277   ccol <- igraph::degree(
278     n2,
279     v=(0+new_igraph):(length(
    igraph::get.vertex.attribute(n2
    ,"name"))-1+new_igraph)
280 )
281   # ggplot2
282   ccol_raw <- ccol
283   ccol_raw[var_select] <- NA
284   ccol_raw[ccol_raw >= 5] <- 5
285   ccol_raw <- as.character(ccol_raw
    )
286   ccol_raw[ccol_raw=="5"] <- "5+"
287 }
288   edg_col <- "gray70"

```

```

289   edg_lty <- 1
290 }
291 }
292
293
294 if (com_method == "none" || com_
      method == "twomode_g"){
295   edg_lty <- 1
296   edg_col <- "gray40"
297 }
298
299 if (com_method == "twomode_g"){
300   edg_lty <- 3
301 }
302
303 # Minimum Spanning Tree
304 if ( min_sp_tree == 1 ){
305
306   mst <- minimum.spanning.tree(
307     n2,
308     weights = 1 - igraph::get.edge.
      attribute(n2, "weight"),
309     algorithm="prim"
310   )
311
312   #if (length(edg_col) == 1){
313     edg_col <- rep("gray55",
314       length( igraph::get.edge.
        attribute(n2, "weight") ))
315   #}
316
317   n2_edges <- get.edgelist(n2,name=
    T);
318   mst_edges <- get.edgelist(mst,
    name=T);
319
320   if (exists("edg_lty") == F){
321     edg_lty <- 1
322   }
323   for ( i in 1:ecount(n2) ){
324     name_n2 <- paste(
325       n2_edges[i,1],
326       n2_edges[i,2]
327     )
328     for ( j in 1:ecount(mst) ){
329       name_mst <- paste(
330         mst_edges[j,1],
331         mst_edges[j,2]
332       )
333       if ( name_n2 == name_mst ){
334         edg_col[i] <- "gray30" #
          edge color
335         if ( length(edg_lty) > 1 ){
336           edg_lty[i] <- 1 # edge
            linetype
337         }
338         break
339       }
340     }
341   }

```

```

342   edg_mst <- edg_col
343 }
344
345
346 lay <- NULL
347 if ( length(igraph::get.vertex.
      attribute(n2,"name")) >= 3 ){
348   d4l <- as.dist( shortest.paths(n2) )
349   if ( min(d4l) < 1 ){
350     d4l <- as.dist( shortest.paths(n2,
      weights=NA ) )
351   }
352   if ( max(d4l) == Inf){
353     d4l[d4l == Inf] <- vcount(n2)
354   }
355   try( lay <- cmdscale( d4l, k=2 ),
      silent=TRUE )
356   if ( is.null(lay) == F ){
357     lay <- round(lay, digits=5)
358     check4fr <- function(d){
359       chk <- 0
360       for (i in combn( length(d[,1]),
        2, simplify=F ) ){
361         if (
362           d[i[1],1] == d[i[2],1]
363           && d[i[1],2] == d[i[2],2]
364         ){
365           return( i[1] )
366         }
367       }
368       return( NA )
369     }
370     while ( is.na(check4fr(lay)) == 0
      ){
371       mv <- check4fr(lay)
372       lay[mv,1] <- lay[mv,1] + 0.001
373       #print( paste( "Moved:", mv ) )
374     }
375   }
376 }
377
378 set.seed(100)
379 if (
380   (com_method == "twomode_c" ||
      com_method == "twomode_
        g")
381   && ( igraph::is.connected(n2) )
382 ){
383   # For igraph > 1.0.0
384   if (igraph_ver >= 10){
385     lay_f <- layout.kamada.kawai(
386       n2,
387       #coords = lay,
388       weights = igraph::get.edge.
        attribute(n2, "weight")
389     )
390   } else {
391     lay_f <- layout.kamada.kawai(
392       n2,
393       start = lay,
394

```

```

395     weights = igraph::get.edge.
        attribute(n2, "weight")
396   )
397 }
398 } else {
399   # For igraph > 1.0.0
400   if (igraph_ver >= 10){
401     lay_f <- layout.fruchterman.
        reingold(
402       n2,
403       #coords = lay,
404       niter = vcount(n2) * 512,
405       weights = igraph::get.edge.
        attribute(n2, "weight")
406     )
407   } else {
408     lay_f <- layout.fruchterman.
        reingold(
409       n2,
410       start = lay,
411       niter = vcount(n2) * 512,
412       weights = igraph::get.edge.
        attribute(n2, "weight")
413     )
414   }
415 }
416
417 lay_f <- scale(lay_f, center=T, scale
    =F)
418 for (i in 1:2){
419   lay_f[,i] <- lay_f[,i] - min(lay_f[,i]);
420   lay_f[,i] <- lay_f[,i] / max(lay_f[,i]);
421   lay_f[,i] <- ( lay_f[,i] - 0.5 ) * 1.96;
422 }
423
424
425
426 if (com_method == "twomode_c" ||
    com_method == "twomode_g"){
427
428   colnames(d)[
429     substring(colnames(d), 1, 2) ==
        "<>"
430   ] <- substring(
431     colnames(d)[
432       substring(colnames(d), 1, 2)
        == "<>"
433     ],
434     3,
435     nchar(colnames(d)[
436       substring(colnames(d), 1, 2)
        == "<>"
437     ], type="c")
438   )
439 }
440
441
442 if ( exists("saving_emf") || exists("
    saving_eps") ){
443   use_alpha <- 0
444 }

```

```

445 target_ids <- NULL
446 if ( exists("target_words") ){
447   # get word IDs
448   for (i in 1:length( igraph::get.
        vertex.attribute(n2, "name") ) ){
449     for (w in target_words){
450       if (
451         colnames(d)[ as.numeric(
            igraph::get.vertex.
                attribute(n2, "name")[i] ]
452           == w
453         ){
454       }
455       target_ids <- c(target_ids, i)
456     }
457   }
458 }
459 }
460
461 edge_label <- NULL
462
463 font_fam <- "Meiryo UI"
464 if ( exists("PERL_font_family") ){
465   font_fam <- PERL_font_family
466 }
467
468 if ( length(igraph::get.vertex.
    attribute(n2, "name")) > 1 ){
469   if (fix_lab == 1){
470     if (exists("if_fixed") == 0){
471       plot.new()
472       plot.window(xlim=c(-1, 1),
473         ylim=c(-1, 1))
474
475       labcd <- NULL
476       labcd$x <- lay_f[,1]
477       labcd$y <- lay_f[,2]
478       word_labs <- colnames(d)[ as.
        numeric( igraph::get.vertex.
            attribute(n2, "name") ) ]
479
480       library(wordcloud)
481
482       # fix for "wordlayout" function
483       filename <- tempfile()
484       writeLines("wordlayout <-
        function (x, y, words, cex =
            1, rotate90 = FALSE, xlim = c
                (-Inf,
485                  Inf), ylim = c(-Inf, Inf),
                    tstep = 0.1, rstep = 0.1,
                    ...)
486       {
487         tails <- \"g|j|p|q|y\"
488         n <- length(words)
489         sdx <- sd(x, na.rm = TRUE)
490         sdy <- sd(y, na.rm = TRUE)
491         iterations <- 0
492         if (sdx == 0)
493           sdx <- 1
494         if (sdy == 0)

```

```

494     sdy <- 1
495     if (length(cex) == 1)
496       cex <- rep(cex, n)
497     if (length(rotate90) == 1)
498       rotate90 <- rep(rotate90, n)
499     boxes <- list()
500     for (i in 1:length(words)) {
501       rotWord <- rotate90[i]
502       r <- 0
503       theta <- runif(1, 0, 2 * pi)
504       x1 <- xo <- x[i]
505       y1 <- yo <- y[i]
506       wid <- strwidth(words[i],
507         cex = cex[i], ...)
508       ht <- strheight(words[i],
509         cex = cex[i], ...)
510       if (grepl(tails, words[i]))
511         ht <- ht + ht * 0.2
512       if (rotWord) {
513         tmp <- ht
514         ht <- wid
515         wid <- tmp
516       }
517       isOverlaped <- TRUE
518       while (isOverlaped) {
519         if (!overlap(x1 - 0.5 *
520           wid, y1 - 0.5 * ht, wid,
521           ht, boxes) && x1 - 0.5 *
522             wid > xlim[1] && y1
523               -
524               0.5 * ht > ylim[1] && x1
525                 + 0.5 * wid < xlim[2]
526                   &&
527                   y1 + 0.5 * ht < ylim[2])
528         {
529           boxes[[length(boxes) +
530             1]] <- c(x1 - 0.5 *
531               wid,
532               y1 - 0.5 * ht, wid, ht)
533           isOverlaped <- FALSE
534         }
535       }
536     }
537   }
538 }

539     print( paste("\niterations: \",
540       iterations) )
541     result <- do.call(rbind,
542       boxes)
543     colnames(result) <- c("\nx\",
544       \"y\", \"width\", \"ht\")
545     rownames(result) <- words
546     result
547   }
548   ", filename)
549   insertSource(filename, package="
550     wordcloud", force=FALSE)
551   nc <- wordlayout(
552     labcd$x,
553     labcd$y,
554     word_labs,
555     cex=cex * 1.28,
556     xlim=c( -1, 1 ),
557     ylim=c( -1, 1 )
558   )
559   xorg <- labcd$x
560   yorg <- labcd$y
561   labcd$x <- nc[,1] + .5 * nc[,3]
562   labcd$y <- nc[,2] + .5 * nc[,4]
563   lay_f <- cbind(labcd$x, labcd$
564     y)
565   if_fixed <- 1
566 }
567 }
568 if ( com_method == "cor" ){ # cor
569   dr <- as.data.frame( t(dr) )
570   dv <- data.frame(
571     khvar_ = as.numeric(v0)
572   )
573   dr <- cbind(dr,dv)
574   edge_pos <- NULL
575   edges <- get.edgelist(n2, names=
576     TRUE)
577   for (i in 1:nrow(edges)){
578     i1 <- as.numeric( edges[i,1] )
579     i2 <- as.numeric( edges[i,2] )
580     edge_pos <- c(
581       edge_pos,
582       cor(
583         as.numeric( dr[,i1] > 0 & dr[,
584           i2] > 0 ),
585         dr$khvar_,
586         method="pearson"
587       )
588     )
589     #mean( dr/dr[,i1] > 0 & dr[,i2]
590       > 0,$khvar_ )
591   }
592   if ( length( edge_pos[is.na(edge_
593     pos) == F] ) == 0 ){

```

```

591     edge_pos <- 0
592   }
593
594   #n2 <- igraph::set.edge.attribute(
595   # n2,
596   # "edge_pos_o",
597   # 1:length(igraph::get.edge.attribute(
598     n2,"weight")),
599   # edge_pos
600   #)
601
602   #edge_pos <- edge_pos - mean(
603     edge_pos)
604   #edge_pos <- edge_pos / sd(edge_
605     pos)
606   ##edge_pos <- edge_pos * 10 + 50
607
608   #limv <- 0.15
609   #maxv <- max( abs( edge_pos ) )
610
611   #if ( limv < maxv ){
612   # edge_pos[edge_pos > limv] <-
613     limv
614   # edge_pos[edge_pos < -limv] <- -
615     limv
616   #}
617
618   n2 <- igraph::set.edge.attribute(
619     n2,
620     "edge_pos",
621     1:length(igraph::get.edge.
622       attribute(n2,"weight")),
623     edge_pos
624   )
625
626   ver_pos <- NULL
627   vertices <- as.numeric( igraph::
628     get.vertex.attribute(n2,"name")
629   )
630   for (i in 1:length(vertices) ){
631     ver_pos <- c(
632       ver_pos,
633       cor(
634         as.numeric( dr[,vertices[i]] >
635           0 ),
636         dr$khvar_,
637         method="pearson"
638       )
639     )
640   }
641
642   if ( length( ver_pos[is.na(ver_pos)
643     == F] ) == 0 ){
644     ver_pos <- 0
645   }
646
647   ver_pos[ver_pos > max(edge_pos)]
648     <- max(edge_pos)
649   ver_pos[ver_pos < min(edge_pos)]
650     <- min(edge_pos)
651

```

```

640   #n2 <- igraph::set.vertex.attribute(
641   # n2,
642   # "ver_pos",
643   # 1:length(igraph::get.vertex.
644     attribute(n2,"name")),
645   # ver_pos
646   #)
647   ccol_raw <- ver_pos
648   if ( is.null( igraph::get.vertex.
649     attribute(n2,"com") ) ==
650     FALSE ){
651     n2 <- remove.vertex.attribute(
652       n2, "com")
653     edg_lty <- 1
654   }
655 }
656
657 n2 <- igraph::set.vertex.attribute(
658   n2,
659   "lab",
660   1:length(igraph::get.vertex.
661     attribute(n2,"name")),
662   colnames(d)[ as.numeric( igraph::
663     get.vertex.attribute(n2,"name")
664   ) ]
665 )
666
667 ver_freq <- freq[ as.numeric( igraph
668   ::get.vertex.attribute(n2,"name") )
669 ]
670
671 if ( com_method == "twomode_c" ||
672     com_method == "twomode_g" ){
673   ver_freq[var_select] <- NA
674 }
675
676 if ( is.null(target_ids) == FALSE ){
677   ver_freq[target_ids] <- NA
678 }
679
680 if ( use_freq_as_size == 0 ){
681   ver_freq[ver_freq > 0] <- 1
682 }
683
684 n2 <- igraph::set.vertex.attribute(
685   n2,
686   "size",
687   1:length(igraph::get.vertex.
688     attribute(n2,"name")),
689   ver_freq
690 )
691
692 # For community detection
693
694 if ( exists("ccol1") ){ # clean up
695   previous data
696   try( n2 <- remove.vertex.
697     attribute(n2, "com"), silent=T )
698 }
699
700 if ( exists("com_m") ){

```

```

688 com_label <- NULL
689
690 for (h in 1:length(com_m$
      membership)){
691   i <- com_m$membership[h]
692   if ( com_m$size[i] > 1 ) {
693     if (i < 10){
694       com_label <- c(
695         com_label,
696         paste("0", as.character(i),
697             " ", sep="")
698     )
699   } else {
700     com_label <- c(com_label, as
701       .character(i))
702   }
703   } else {
704     com_label <- c(com_label, NA)
705   }
706 }
707
708 n2 <- igraph::set.vertex.attribute(
709   n2,
710   "com",
711   1:length(igraph::get.vertex.
712     attribute(n2,"name")),
713   com_label
714 )
715 }
716
717 if ( exists("ccol_raw") ){
718   n2 <- igraph::set.vertex.attribute(
719     n2,
720     "com",
721     1:length(igraph::get.vertex.
722       attribute(n2,"name")),
723     ccol_raw
724 )
725 com_label <- ccol_raw
726 }
727
728 if ( com_method == "none" || com_
729   method == "twomode_g" ){
730   n2 <- igraph::set.vertex.attribute(
731     n2,
732     "com",
733     1:length(igraph::get.vertex.
734       attribute(n2,"name")),
735     rep( "na", length(igraph::get.
736       vertex.attribute(n2,"name")) )
737 )
738 com_label <- NA
739 }
740
741 if ( exists("edg_mst") ){
742   n2 <- igraph::set.edge.attribute(
743     n2,
744     "edg_col",
745     1:length(igraph::get.edge.
746       attribute(n2,"weight")),
747     edg_mst

```

```

748   )
749   #print(edg_mst)
750 }
751
752 if ( exists("edg_lty") == F ){
753   edg_lty <- 1
754 }
755 edg_lty[edg_lty==1] <- "solid"
756 edg_lty[edg_lty==3] <- "dotted"
757
758 n2 <- igraph::set.edge.attribute(
759   n2,
760   "line",
761   1:length(igraph::get.edge.attribute(
762     n2,"weight")),
763   edg_lty
764 )
765
766 library(ggplot2)
767 library(ggnetwork)
768
769 p <- ggplot(
770   ggnetwork(n2, layout=lay_f),
771   aes(x = x, y = y, xend = xend, yend
772     = yend),
773 )
774
775 if (use_alpha == 1){
776   alpha_value = 0.62
777   gray_color_n <- "gray20"
778 } else {
779   alpha_value = 1
780   gray_color_n <- "gray40"
781 }
782
783 if (text_font == 2){
784   face <- "bold"
785 } else {
786   face <- "plain"
787 }
788
789 if (smaller_nodes == 1 ){
790   edge_colour <- "gray68"
791   nudge <- 0.015
792   hjust <- "left"
793 } else {
794   edge_colour <- "gray55"
795   nudge <- 0
796   hjust <- "center"
797 }
798
799 if (com_method == "twomode_c"){
800   edge_colour <- "gray70"
801 }
802
803 if (com_method == "none" || com_
804   method == "twomode_g"){
805   edge_colour <- "gray40"
806   gray_color_n <- "black"
807 }

```

```

798 rownames(lay_f) <- colnames(d)[
      as.numeric( igraph::get.vertex.
        attribute(n2,"name") ) ]
799 lay_f[,1] <- lay_f[,1] - min(lay_f[,1])
800 lay_f[,1] <- lay_f[,1] / max(lay_f[,1])
801 lay_f[,2] <- lay_f[,2] - min(lay_f[,2])
802 lay_f[,2] <- lay_f[,2] / max(lay_f[,2])
803 lay_f_df <- data.frame(
804   x = lay_f[,1],
805   y = lay_f[,2],
806   lab = rownames(lay_f)
807 )
808
809 if ( smaller_nodes == 1 ){
810   vv <- 6.2
811 } else {
812   vv <- 20
813 }
814
815 sans <- "sans"
816 if ( exists("saving_eps") ){
817   sans <- NULL
818 }
819
820 if (com_method == "cor"){ # cor
821
822   if ( gray_scale == 1 ) {
823     myPalette <- gray( seq(1, 0,
824       length.out=100) )
825     gray_color_n <- "black"
826     if (alpha_value < 0.8) {
827       alpha_value <- 0.8
828     }
829     p <- p + geom_edges(
830       color = "black",
831       size = 1.5
832     )
833     p <- p + geom_edges(
834       aes(
835         color = edge_pos
836       ),
837       size = 1,
838     )
839   } else {
840     myPalette <- colorRampPalette(
841       rev( brewer.pal(9, "RdYlBu") )
842     )(100) #Spectral
843     p <- p + geom_edges(
844       aes(
845         color = edge_pos
846       ),
847       size = 0.6,
848     )
849   }
850   p <- p + scale_color_gradientn(
851     colours = myPalette,
852     #limits = c( min(edge_pos), limv )
853
854     #limits = c( 0 - limv, 0 + limv ),
855     guide = guide_colourbar(

```

```

855     title = "Correlation:\n",
856     title.theme = element_text(
857       family=sans,
858       face="bold",
859       size=11,
860       lineheight=0.4,
861       angle=0
862     ),
863     order = 1,
864     #override.aes = list(size=6,
865       shape=22),
866     label.hjust = 1,
867     #reverse = TRUE,
868     #ncol=2,
869     #keyheight = unit(1.5,"line")
870   )
871   p <- p + scale_fill_gradientn(
872     colours = myPalette,
873     guide = FALSE
874   )
875 } else if (min_sp_tree == 1){
876   edg_col2 <- p$data$edg_col
877   edg_col2[edg_col2=="gray30"] <-
878     "MST"
879   edg_col2[edg_col2=="gray55"] <-
880     "non-MST"
881   edg_col2[edg_col2=="gray70"] <-
882     "non-MST"
883   p <- p + geom_edges(
884     aes(linetype = as.character(line),
885       alpha=edg_col2),
886     #size = 0.8,
887     color = "grey10"
888   )
889   p <- p + scale_alpha_discrete(
890     range = c(1, 0.3),
891     guide = guide_legend(
892       title = "Edge:",
893       keyheight = unit(1.2,"line"),
894       order = 2
895     )
896   )
897 } else if ( use_weight_as_width == 1 )
898 {
899   p <- p + geom_edges(
900     aes(linetype = as.character(line),
901       alpha=weight),
902     #size = 0.8,
903     color = "grey10"
904   )
905   p <- p + scale_alpha(
906     range = c(0.2, 1),
907     guide = guide_legend(
908       title = "Coefficient:",
909       label.hjust = 1,
910       keyheight = unit(1.2,"line"),
911       order = 2
912     )
913   )
914 } else {

```

```

909 p <- p + geom_edges(
910   aes(linetype = as.character(line))
911   ,
912   size = 0.4,
913   color = edge_colour
914 )
915 }
916 p <- p + scale_linetype_identity()
917
918 alpha_config <- 0
919 if (
920   ( com_method == "com-b" ||
921     com_method == "com-g" ||
922     com_method == "com-r" )
921   && ( length(com_m$csizes[com_m$
922     csizes >= 2]) >= 13 )
922   && ( length(com_m$csizes[com_m$
923     csizes >= 2]) <= 20 )
923 ){
924   alpha_config <- -0.5
925   p <- p + geom_nodes(
926     aes(
927       size = size * 0.41
928     ),
929     alpha = 0.3, # 0.65
930     color = "white",
931     show.legend = F,
932     shape = 16
933   )
934   p <- p + geom_nodes(
935     aes(
936       size = size
937     ),
938     alpha = 0.65,
939     color = "white",
940     show.legend = F,
941     shape = 16
942   )
943 }
944
945 p <- p + geom_nodes(
946   aes(
947     size = size * 0.41,
948     color = com
949   ),
950   alpha = 0.85 + alpha_config,
951   show.legend = F,
952   shape = 16
953 )
954 p <- p + geom_nodes(
955   aes(
956     size = size,
957     color = com,
958     shape = shape
959   ),
960   alpha = alpha_value + alpha_config
961     / 3,
962   shape = 16
963 )
964 p <- p + geom_nodes(

```

```

964   aes(
965     size = size,
966     shape = shape
967   ),
968   colour = gray_color_n,
969   show.legend = F,
970   alpha = alpha_value,
971   shape = 1
972 )
973 p <- p + geom_nodes( # dummy for
974   the legend
975   aes( fill = com ),
976   size=0,
977   colour = gray_color_n,
978   alpha = 0,
979   shape = 21
980 )
981 if ( use_freq_as_size == 1 ){
982   p <- p + scale_size_area(
983     "Frequency",
984     max_size = 30 * bubble_size /
985       100,
986     guide = guide_legend(
987       title = "Frequency:",
988       override.aes = list(colour="
989         black", alpha=1, shape=1),
990       label.hjust = 1,
991       order = 3
992     )
993   )
994 } else {
995   p <- p + scale_size_area(
996     max_size = vv,
997     guide = F
998   )
999 }
1000 if ( (com_method == "twomode_c" ||
1001   com_method == "twomode_g") ) {
1002   # (is.null(target_ids) == FALSE)
1003   if ( com_method == "twomode_c" ){
1004     var_outline_c <- "gray50"
1005     var_fill_c <- "#FB8072"
1006   }
1007   if ( com_method == "twomode_g" ){
1008     var_outline_c <- "black"
1009     var_fill_c <- "white"
1010   }
1011 }
1012 p <- p + geom_point(
1013   data = data.frame(
1014     x = lay_f[var_select,1],
1015     y = lay_f[var_select,2]
1016   ),
1017   aes(
1018     x = x,
1019     y = y,
1020     xend = x,
1021     yend = y

```

```

1021   ),
1022   fill = var_fill_c,
1023   show.legend = F,
1024   colour = NA,
1025   alpha = 0.8,
1026   size = vv * 2 / 3,
1027   shape = 22
1028 )
1029
1030 p <- p + geom_point(
1031   data = data.frame(
1032     x = lay_f[var_select,1],
1033     y = lay_f[var_select,2]
1034   ),
1035   aes(
1036     x = x,
1037     y = y,
1038     xend = x,
1039     yend = y
1040   ),
1041   fill = var_fill_c,
1042   show.legend = F,
1043   colour = var_outline_c,
1044   alpha = alpha_value,
1045   size = vv,
1046   shape = 22
1047 )
1048 }
1049
1050 if ( (is.null(target_ids) == FALSE) )
1051 {
1052   var_select <- target_ids
1053
1054   p <- p + geom_point(
1055     data = data.frame(
1056       x = lay_f[var_select,1],
1057       y = lay_f[var_select,2]
1058     ),
1059     aes(
1060       x = x,
1061       y = y,
1062       xend = x,
1063       yend = y,
1064       fill = com_label[var_select]
1065     ),
1066     show.legend = F,
1067     colour = NA,
1068     alpha = 0.8,
1069     size = vv * 2 / 3,
1070     shape = 22
1071   )
1072
1073   p <- p + geom_point(
1074     data = data.frame(
1075       x = lay_f[var_select,1],
1076       y = lay_f[var_select,2]
1077     ),
1078     aes(
1079       x = x,
1080       y = y,
1081       xend = x,

```

```

1081       yend = y,
1082       fill = com_label[var_select]
1083     ),
1084     show.legend = F,
1085     colour = gray_color_n,
1086     alpha = alpha_value,
1087     size = vv,
1088     shape = 22
1089   )
1090
1091   p <- p + geom_point(
1092     data = data.frame(
1093       x = lay_f[var_select,1],
1094       y = lay_f[var_select,2]
1095     ),
1096     aes(
1097       x = x,
1098       y = y,
1099       xend = x,
1100       yend = y
1101     ),
1102     fill = NA,
1103     show.legend = F,
1104     colour = gray_color_n,
1105     alpha = alpha_value,
1106     size = vv * 1.4,
1107     shape = 22
1108   )
1109 }
1110
1111 if (
1112   ( com_method == "com-b" || com_
1113     method == "com-g" || com_
1114     method == "com-r" || com_
1115     method == "cor")
1116   && gray_scale == 1
1117   && smaller_nodes == 0
1118 ) {
1119   p <- p + geom_label(
1120     data = lay_f_df,
1121     aes(
1122       x = x,
1123       y = y,
1124       xend = x,
1125       yend = y,
1126       label = lab
1127     ),
1128     size=4,
1129     hjust = hjust,
1130     nudge_x = nudge,
1131     nudge_y = nudge * 1.25,
1132     family=font_fam,
1133     na.rm = T,
1134     label.size = NA,
1135     label.padding = unit(0.2, "lines")
1136   ,
1137     label.r = unit(0.1, "lines"),
1138     fontface=face
1139   )
1140 } else {
1141   p <- p + geom_text(

```

```

1138   data = lay_f_df,
1139   aes(
1140     x = x,
1141     y = y,
1142     xend = x,
1143     yend = y,
1144     label = lab
1145   ),
1146   size=4,
1147   hjust = hjust,
1148   nudge_x = nudge,
1149   nudge_y = nudge * 1.25,
1150   family=font_fam,
1151   na.rm = T,
1152   fontface=face
1153 )
1154 }
1155
1156 if (view_coef == 1){
1157   p <- p + geom_edgetext(
1158     aes(label = substring( round(
1159       weight, digits = 2), 2, 4 ) ,
1160       color = "#000080",
1161       fill = NA,
1162       size=3.5,
1163     )
1164   )
1165   if ( com_method == "com-b" || com_
1166     method == "com-g" || com_
1167     method == "com-r"){
1168     if ( gray_scale == 1) {
1169       p <- p + scale_color_grey(
1170         na.value = "white",
1171         guide = FALSE
1172       )
1173       p <- p + scale_fill_grey(
1174         na.value = "white",
1175         guide = guide_legend(
1176           title = "Community:",
1177           override.aes = list(size=5.5,
1178             alpha=1, shape=22),
1179           keyheight = unit(1,"line"),
1180           ncol=2,
1181           order = 1
1182         )
1183       )
1184     } else {
1185       if ( length(com_m$csizes[com_m$
1186         csizes > 1]) <= 12 ){
1187         p <- p + scale_color_brewer(
1188           palette = "Set3",
1189           na.value = "white",
1190           guide = FALSE
1191         )
1192         p <- p + scale_fill_brewer(
1193           palette = "Set3",
1194           na.value = "white",
1195           guide = guide_legend(
1196             title = "Community:",
1197             override.aes = list(size=5.5,
1198               alpha=1, shape=22),
1199             keyheight = unit(1.25,"line
1200             "),
1201             ncol=2,
1202             order = 1
1203           )
1204         )
1205       } else {
1206         p <- p + scale_color_d3(
1207           palette = "category20",
1208           na.value = "white",
1209           guide = FALSE
1210         )
1211         p <- p + scale_fill_d3(
1212           palette = "category20",
1213           na.value = "white",
1214           guide = guide_legend(
1215             title = "Community:",
1216             override.aes = list(size=5.5,
1217               alpha=1, shape=22),
1218             keyheight = unit(1.25,"line
1219             "),
1220             ncol=2,
1221             order = 1
1222           )
1223         )
1224       } else {
1225         p <- p + scale_color_hue(
1226           c = 50,
1227           l = 85,
1228           na.value = "white",
1229           guide = FALSE
1230         )
1231         p <- p + scale_fill_hue(
1232           c = 50,
1233           l = 85,
1234           na.value = "white",
1235           guide = guide_legend(
1236             title = "Community:",
1237             override.aes = list(size=5.5,
1238               alpha=1, shape=22,
1239               colour="gray45"),
1240             keyheight = unit(1.25,"line
1241             "),
1242             ncol=2,
1243             order = 1
1244           )
1245         )
1246       }
1247     }
1248   }
1249   if ( com_method == "cnt-b" || com_
1250     method == "cnt-d" || com_
1251     method == "cnt-e"){
1252     if (gray_scale == 1){
1253       myPalette <- gray( seq(1, 0.4,
1254         length.out=100) )

```

```

1244 } else {
1245   if (com_universal_design == 0){
1246     myPalette <- cm.colors(99)
1247   } else {
1248     library(RColorBrewer)
1249     col_seed <- brewer.pal(8,"
1250       YlGnBu")[1:6]
1251     myPalette <-
1252       colorRampPalette( col_seed
1253     )
1254     myPalette <- myPalette(99)
1255   }
1256   p <- p + scale_color_gradientn(
1257     colours = myPalette,
1258     guide = FALSE
1259   )
1260   p <- p + scale_fill_gradientn(
1261     colours = myPalette,
1262     guide = guide_colourbar(
1263       title = "Centrality:\n",
1264       title.theme = element_text(
1265         family=sans,
1266         face="bold",
1267         size=11,
1268         lineheight=0.4,
1269         angle=0
1270       ),
1271       order = 1,
1272       #override.aes = list(size=6,
1273         shape=22),
1274       label.hjust = 1,
1275       #reverse = TRUE,
1276       #ncol=2,
1277       #keyheight = unit(1.5,"line")
1278     )
1279   }
1280   if (com_method == "twomode_c"){
1281     p <- p + scale_color_manual(
1282       values = brewer.pal(8, "Spectral"
1283     )[4:8],
1284     guide = FALSE
1285   )
1286   p <- p + scale_fill_manual(
1287     values = brewer.pal(8, "Spectral"
1288     )[4:8],
1289     guide = guide_legend(
1290       title = "Degree:",
1291       order = 1,
1292       override.aes = list(size=5.5,
1293         shape=22, alpha=1),
1294       #label.hjust = "left",
1295       #reverse = TRUE,
1296       #ncol=2,
1297       keyheight = unit(1.2,"line")
1298     )
1299   }

```

```

1298   }
1299   if ( com_method == "none" || com_
1300     method == "twomode_g"){
1301     p <- p + scale_color_manual(
1302       values = c("white"),
1303       na.value = "white",
1304       guide = F
1305     )
1306     p <- p + scale_fill_manual(
1307       values = c("white"),
1308       na.value = "white",
1309       guide = F
1310     )
1311   }
1312   p <- p + theme_blank(
1313     base_family = font_fam
1314   )
1315   if (com_method == "cor" && gray_
1316     scale == 0){ # cor
1317     if ( cor_var_darker == 1 ){
1318       col_backg <- "gray50"
1319     } else {
1320       col_backg <- "gray60"
1321     }
1322     p <- p + theme(
1323       panel.background = element_rect
1324         (fill = col_backg, colour = NA
1325       )
1326     )
1327     p <- p + theme(
1328       legend.title = element_text(family
1329         =sans, face="bold", size=11,
1330         angle=0),
1331       legend.text = element_text(face="
1332         plain", size=11, angle=0)
1333     )
1334     # make a small space between the
1335     # graph and the legend
1336     margin <- 0.04
1337     #if (smaller_nodes == 1){
1338     # extra <- 0.05
1339     # p <- p + coord_fixed(ratio=1, xlim
1340       =c(0-margin-extra,1+margin+
1341       extra), ylim=c(0-margin,1+
1342       margin), expand = F )
1343     #} else {
1344     extra <- 0.025
1345     p <- p + coord_fixed(ratio=1, xlim
1346       =c(0-margin-extra,1+
1347       margin+extra), ylim=c(0-
1348       margin,1+margin), expand =
1349       F )
1350     #}

```

```

1344 #p <- p + theme(plot.margin= unit(
1345     c(5, 0, 5, 0), "pt"))
1346 g <- ggplotGrob(p)
1347
1348 if ( length( g$grobs[[8]][[1]][[1]] ) > 1)
1349 {
1350     if (
1351         (com_method == "cnt-b" || com_
1352         method == "cnt-d" || com_
1353         method == "cnt-e")
1354         && ( gray_scale == 0 )
1355     ){
1356         g$grobs[[8]][[1]][[1]]$grobs[[5]]$gp$
1357         col <- "gray45"
1358         g$grobs[[8]][[1]][[1]]$grobs[[5]]$gp$
1359         lwd <- 1.25
1360     }
1361     if (
1362         (com_method == "cnt-b" || com_
1363         method == "cnt-d" || com_
1364         method == "cnt-e")
1365         && ( gray_scale == 1 )
1366     ){
1367         g$grobs[[8]][[1]][[1]]$grobs[[5]]$gp$
1368         col <- "gray30"
1369         g$grobs[[8]][[1]][[1]]$grobs[[5]]$gp$
1370         lwd <- 1.25
1371     }
1372     if ( com_method == "cor" &&
1373         gray_scale == 0){
1374         g$grobs[[8]][[1]][[1]]$grobs[[5]]$gp$
1375         col <- "gray40"
1376         g$grobs[[8]][[1]][[1]]$grobs[[5]]$gp$
1377         lwd <- 1.1
1378     }
1379 }
1380
1381 library(grid)
1382 library(gtable)
1383
1384 # fixing width of legends to 22%
1385 if ( exists("saving_file") ){
1386     if ( saving_file == 0){
1387         target_legend_width <-
1388             convertX(
1389                 unit( image_width * 0.22, "in"
1390                     ),
1391                 "mm"
1392             )
1393     }
1394     if ( as.numeric( substr(
1395         packageVersion("ggplot2"), 1,
1396         3) ) <= 2.1 ){ # ggplot2 <=
1397         2.1.0
1398         diff_mm <- diff( c(
1399             convertX( g$widths[5], "mm" ),
1400             target_legend_width
1401         ))
1402         if ( diff_mm > 0 ){
1403             print(diff_mm)
1404             g <- gtable::add_cols(g, unit(
1405                 diff_mm, "mm"))
1406         }
1407     } else { # ggplot2 >= 2.2.0
1408         diff_mm <- diff( c(
1409             convertX( g$widths[7], "mm",
1410                 valueOnly=T ) +
1411             convertX( g$widths[8], "
1412                 mm", valueOnly=T ),
1413             target_legend_width
1414         ))
1415         if ( diff_mm > 0 ){
1416             print(diff_mm)
1417             g <- gtable::add_cols(g, unit(
1418                 diff_mm, "mm"))
1419         }
1420     }
1421 }
1422 grid.draw(g)
1423
1424 if (exists("com_m")){
1425     rm("com_m")
1426 }
1427 if (exists("ccol_raw")){
1428     rm("ccol_raw")
1429 }
1430 if (exists("edg_mst")){
1431     rm("edg_mst")
1432 }
1433 if (exists("edg_lty")){
1434     rm("edg_lty")
1435 }
1436 ccol <- igraph::get.vertex.attribute(
1437     n2,"com")

```

A. 6 時系列 SOM を作成するソースコード

時系列 SOM を作成するソースコード A.6 をしめす.

ソースコード A. 6: som.r <pre> 1 d <- NULL 2 d <- matrix(c(1,...省略...,0), byrow </pre>	<pre> =T, nrow=380, ncol=41) 3 d <- d[,-1] 4 colnames(d) <- c("desk",...省略...", </pre>
--	---

```

      small")
5 doc_length_mtr <- matrix( c(
      70,18,...省略...57,19), ncol=2,
      byrow=T)
6 colnames(doc_length_mtr) <- c("
      length_c", "length_w")
7 d <- t(d)
8 # END: DATA
9
10 n_nodes <- 20
11 rlen1 <- 1000
12 rlen2 <- 200000
13 d <- t(d)
14
15 if (exists("doc_length_mtr")){
16   leng <- as.numeric(doc_length_
      mtr[,2])
17   leng[leng==0] <- 1
18   d <- d / leng
19   d <- d * 1000
20 }
21
22 d <- subset(d, rowSums(d) > 0)
23 d <- scale(d)
24 d <- t(d)
25 d <- t(d)
26 rownames(d) <- 1:nrow(d)
27 # SOM
28 library(som)
29 somm <- som(
30   d,
31   n_nodes,
32   n_nodes,
33   topol="hexa",
34   rlen=c(rlen1,rlen2)
35 )
36
37 word_labs <- rownames(d)
38 n_words <- length(word_labs)
39
40 color_universal_design <- 1
41 # END: DATA
42
43 cex <- 1
44 text_font <- 2
45 if_cls <- 1
46 n_cls <- 9
47 if_plothex <- 1
48
49 # n_nodes <- 20
50 # rlen1 <- 1000
51 # rlen2 <- 200000
52
53 row2coods <- NULL
54 eve <- 0
55 for (i in 0:(n_nodes - 1)){
56   for (h in 0:(n_nodes - 1)){
57     row2coods <- c(row2coods, h +
      eve, i)
58   }
59   if (eve == 0){

```

```

60     eve <- 0.5
61   } else {
62     eve <- 0
63   }
64 }
65 row2coods <- matrix( row2coods,
      byrow=T, ncol=2 )
66
67 if ( if_cls == 1 ){
68   library( RColorBrewer )
69
70   if (
71     ( as.numeric( R.Version()$
      major ) >= 3 )
72     && ( as.numeric( R.Version()$
      minor ) >= 1.0)
73   ){ # >= R 3.1.0
74     hcl <- hclust( dist(somm$code,
      method="euclidean"),
      method="ward.D2" )
75   } else { # <= R 3.0
76     hcl <- hclust( dist(somm$code,
      method="euclidean")^2,
      method="ward" )
77   }
78
79   colors <- NULL
80   if (n_cls <= 9){
81     pastel <- brewer.pal(9, "Pastel1
      ")
82     # pastel[6] = brewer.pal(9, "
      Pastel1")[9]
83     # pastel[9] = brewer.pal(9, "
      Pastel1")[6]
84     pastel[6] = "gray91"
85     pastel[9] = "#F5F5DC" # FAF3C8
      F7F1C6 EEE8AA F0E68C
86     colors <- pastel[cutree(hcl,k=n_
      cls)]
87   }
88   if (n_cls > 9) {
89
90     library(colorspace)
91     new_col <- order( runif(n_cls)
      )
92     colors <-
93       rainbow_hcl(n_cls, start=20,
      end=340, l=92, c=20)[
94       #terrain_hcl(n_cls, c = c(35, 5),
      l = c(85, 95), power = c
      (0.5,1))][
95       new_col[cutree(hcl,k=n_cls)]
96     ]
97   }
98 } else {
99   colors <- rep("gray90", n_nodes
      ^2)
100 }
101 labcd <- NULL
102 plot_mode <- "color"
103

```

```

104 par(mai=c(0,0,0,0), mar=c(0,0,0,0),
      omi=c(0,0,0,0), oma =c(0,0,0,0) )
105
106 plot(
107   NULL,NULL,
108   xlim=c(0,n_nodes-0.5),
109   ylim=c(0,n_nodes-1),
110   axes=F,
111   frame.plot=F
112 )
113
114 if (if_plohex == 1){
115   a <- 0.333333333333
116 } else {
117   a <- 0.5
118 }
119 b <- 1-a
120
121 color_pte <- "gray70"
122 cls_lwd <- 2
123
124 if ( plot_mode == "gray"){
125   color_act <- rep("white",n_nodes
126     ^2)
127   if_points <- 1
128   w_lwd <- 1
129   cls_lwd <- 2.25
130   color_cls <- "gray35"
131   color_line <- "gray50"
132   color_pte <- "gray40"
133   color_ptf <- "gray85"
134 }
135 if ( plot_mode == "color" ) {
136   color_act <- colors
137   color_line <- "white"
138   if_points <- 1
139   w_lwd <- 1
140   if (n_cls > 9) {
141     color_cls <- "gray45"
142   } else {
143     color_cls <- "gray60"
144   }
145   color_ptf <- "white"
146 }
147 if ( plot_mode == "freq" ){
148   color_act <- somm$code.sum$nobs
149   ;
150   if (max(color_act) == 1){
151     color_act <- color_act * 3 + 1;
152   } else {
153     color_act <- color_act - min(
154       color_act)
155     color_act <- round( color_act /
156       max(color_act) * 6 ) + 1
157     #color_act[color_act==7] <- 6
158   }
159   color_seed <- brewer.pal(6,"GnBu")
160   #color_seed <- brewer.pal(6,"
161     YlOrRd")
162   color_seed <- c("white", color_seed
163     )

```

```

158 color_act <- color_seed[color_act]
159
160 color_line <- "gray70"
161 if_points <- 0
162 w_lwd <- 1
163 color_cls <- "gray45"
164 color_ptf <- "white"
165 }
166 if ( plot_mode == "umat" ){
167
168
169 dist_u <- NULL
170
171 dist_m <- as.matrix( dist(somm$
172   code, method="euclid") )
173
174 for (i in 0:(n_nodes - 1)){
175   for (h in 0:(n_nodes - 1)){
176     cu <- NULL
177     n <- 0
178
179     if (h != n_nodes - 1){
180       cu <- c(
181         cu,
182         dist_m[
183           h + i * n_nodes + 1,
184           h + 1 + i * n_nodes + 1
185         ]
186       )
187     }
188
189     if (h != 0){
190       cu <- c(
191         cu,
192         dist_m[
193           h + i * n_nodes + 1,
194           h - 1 + i * n_nodes + 1
195         ]
196       )
197     }
198
199     if (i != n_nodes - 1){
200       if (h %% 2 == 0){
201         cu <- c(
202           cu,
203           dist_m[
204             h + i * n_nodes + 1,
205             h + ( i + 1 ) * n_nodes
206               + 1
207           ]
208         )
209       } else {
210         if (h != n_nodes - 1){
211           cu <- c(
212             cu,
213             dist_m[
214               h + i * n_nodes + 1,
215               h + 1 + ( i + 1 ) * n_
216                 nodes + 1
217             ]
218           )
219         }
220       }
221     }
222   }
223 }

```

```

216     }
217   }
218 }
219
220 if (i != 0){
221   if (h %% 2 == 0){
222     cu <- c(
223       cu,
224       dist_m[
225         h + i * n_nodes + 1,
226         h + (i - 1) * n_nodes
227           + 1
228       ]
229     )
230   } else {
231     if (h != n_nodes - 1){
232       cu <- c(
233         cu,
234         dist_m[
235           h + i * n_nodes + 1,
236           h + 1 + (i - 1) * n_
237             nodes + 1
238         ]
239       )
240     }
241   }
242
243   if (i != n_nodes - 1){
244     if (h %% 2 == 0){
245       if (h != 0){
246         cu <- c(
247           cu,
248           dist_m[
249             h + i * n_nodes + 1,
250             h - 1 + (i + 1) * n_
251               nodes + 1
252           ]
253         )
254       } else {
255         cu <- c(
256           cu,
257           dist_m[
258             h + i * n_nodes + 1,
259             h + (i + 1) * n_nodes
260               + 1
261           ]
262         )
263       }
264     }
265
266     if (i != 0){
267       if (h %% 2 == 0){
268         if (h != 0){
269           cu <- c(
270             cu,
271             dist_m[

```

```

272               ]
273             )
274           }
275         } else {
276           cu <- c(
277             cu,
278             dist_m[
279               h + i * n_nodes + 1,
280               h + (i - 1) * n_nodes
281                 + 1
282             ]
283           )
284         }
285       }
286     }
287   }
288
289   print( summary(dist_u) )
290
291   dist_u <- dist_u - min(dist_u)
292   dist_u <- round( dist_u / max(
293     dist_u) * 100 ) + 1
294
295   if (color_universal_design == 0){
296     color_act <- cm.colors(101)[dist
297       _u]
298     color_line <- "gray70"
299     color_cls <- "gray45"
300   } else {
301     library(RColorBrewer)
302     if (T){
303       col_seed <- brewer.pal(9, "
304         GnBu")
305       myPalette <-
306         colorRampPalette( col_seed
307           )
308       color_act <- myPalette(101)[
309         dist_u]
310       color_act <- adjustcolor(color_
311         act, alpha=0.8)
312       color_line <- "white"
313       color_cls <- "gray30"
314     } else {
315       col_seed <- rev(brewer.pal(9,
316         "RdYlBu"))
317       myPalette <-
318         colorRampPalette( col_seed
319           )
320       color_act <- myPalette(101)[
321         dist_u]
322       color_act <- adjustcolor(color_
323         act, alpha=0.8)
324       color_line <- "gray50"
325       color_cls <- "gray25"
326     }
327   }
328
329   #color_line <- "gray70"
330   if_points <- 1

```

```

319 w_lwd <- 1
320 #color_cls <- "gray45"
321 color_ptf <- "white"
322 }
323
324 for (i in 1:n_nodes^2){
325   x <- row2coords[i,1]
326   y <- row2coords[i,2]
327
328   polygon(
329     x=c( x + 0.5, x + 0.5, x, x - 0.5,
330          x - 0.5, x ),
331     y=c( y + a, y - a, y - b, y - a,
332          y + a, y + b ),
333     col=color_act[i],
334     border="white",
335     lty=0,
336   )
337 }
338
339 for (i in 0:(n_nodes - 1)){
340   for (h in 0:(n_nodes - 2)){
341     if ( colors[h + i * n_nodes + 1]
342         == colors[h + i * n_nodes +
343                 2] ){
344       x <- h
345       y <- i
346       if ( y %% 2 == 1 ){
347         x <- x + 0.5
348       }
349
350       segments(
351         x + 0.5, y + a,
352         x + 0.5, y - a,
353         col=color_line,
354         lwd=w_lwd,
355       )
356     }
357   }
358 }
359
360 for (i in 0:(n_nodes - 1)){
361   for (h in c(-1, n_nodes-1) ){
362     x <- h
363     y <- i
364     if ( y %% 2 == 1 ){
365       x <- x + 0.5
366     }
367
368     segments(
369       x + 0.5, y + a,
370       x + 0.5, y - a,
371       col=color_line,
372       lwd=w_lwd,
373     )
374   }
375 }

```

```

376 )
377 if ( y != 0){
378   segments(
379     -0.5, y - a,
380     0, y - 1 + a,
381     col=color_line,
382     lwd=w_lwd,
383   )
384 }
385 } else {
386   if ( y != n_nodes - 1){
387     segments(
388       n_nodes - 0.5, y + 1 - a,
389       n_nodes, y + a,
390       col=color_line,
391       lwd=w_lwd,
392     )
393   }
394   segments(
395     n_nodes - 0.5, y - 1 + a,
396     n_nodes, y - a,
397     col=color_line,
398     lwd=w_lwd,
399   )
400 }
401 }
402
403 for (i in 0:(n_nodes - 2)){
404   for (h in 0:(n_nodes - 1)){
405     if (i %% 2 == 1){
406       chk <- 1
407     } else {
408       chk <- 0
409     }
410
411     if (
412       is.na(colors[h + i * n_nodes
413                + 1]) == 1
414       || is.na(colors[h + chk + (i+1)
415                      * n_nodes + 1]) == 1
416       || h + chk == n_nodes
417     ){
418       next
419     }
420
421     if (
422       colors[h + i * n_nodes + 1]
423       == colors[h + chk + (i+1) * n
424                _nodes + 1]
425     ){
426       x <- h
427       y <- i
428       if ( y %% 2 == 1 ){
429         x <- x + 0.5
430       }
431
432       segments(
433         x, y + b,
434         x + 0.5, y + a,
435         col=color_line,
436         lwd=w_lwd,

```

```

434     )
435   }
436 }
437 }
438
439 for (i in 0:(n_nodes - 2)){
440   for (h in 0:(n_nodes - 1)){
441     if (i %% 2 == 0){
442       chk <- 1
443     } else {
444       chk <- 0
445     }
446     if (
447       is.na(colors[h + i * n_nodes
448         + 1]) == 1
449       || is.na(colors[h - chk + (i+1)
450         * n_nodes + 1]) == 1
451       || h - chk < 0
452     ){
453       next
454     }
455     if (
456       colors[h + i * n_nodes + 1]
457       == colors[h - chk + (i+1) * n
458         _nodes + 1]
459     ){
460       x <- h
461       y <- i
462       if ( y %% 2 == 1 ){
463         x <- x + 0.5
464       }
465       segments(
466         x, y + b,
467         x - 0.5, y + a,
468         col=color_line,
469         lwd=w_lwd,
470       )
471     }
472   }
473 }
474
475 for (i in 0:(n_nodes - 1)){
476   for (h in 0:(n_nodes - 2)){
477     if ( colors[h + i * n_nodes + 1] !
478       = colors[h + i * n_nodes + 2]
479     ){
480       x <- h
481       y <- i
482       if ( y %% 2 == 1 ){
483         x <- x + 0.5
484       }
485       segments(
486         x + 0.5, y + a,
487         x + 0.5, y - a,
488         col=color_cls,
489         lwd=cls_lwd,

```

```

490     )
491   }
492 }
493 }
494
495 for (i in 0:(n_nodes - 2)){
496   for (h in 0:(n_nodes - 1)){
497     if (i %% 2 == 1){
498       chk <- 1
499     } else {
500       chk <- 0
501     }
502     if (
503       is.na(colors[h + i * n_nodes
504         + 1]) == 1
505       || is.na(colors[h + chk + (i+1)
506         * n_nodes + 1]) == 1
507       || h + chk == n_nodes
508     ){
509       next
510     }
511     if (
512       colors[h + i * n_nodes + 1]
513       != colors[h + chk + (i+1) * n_
514         nodes + 1]
515     ){
516       x <- h
517       y <- i
518       if ( y %% 2 == 1 ){
519         x <- x + 0.5
520       }
521       segments(
522         x, y + b,
523         x + 0.5, y + a,
524         col=color_cls,
525         lwd=cls_lwd,
526       )
527     }
528   }
529 }
530
531 for (i in 0:(n_nodes - 2)){
532   for (h in 0:(n_nodes - 1)){
533     if (i %% 2 == 0){
534       chk <- 1
535     } else {
536       chk <- 0
537     }
538     if (
539       is.na(colors[h + i * n_nodes
540         + 1]) == 1
541       || is.na(colors[h - chk + (i+1)
542         * n_nodes + 1]) == 1
543       || h - chk < 0
544     ){
545       next

```

```

546
547   if (
548       colors[h + i * n_nodes + 1]
549       != colors[h - chk + (i+1) * n_
          nodes + 1]
550   ){
551       x <- h
552       y <- i
553       if ( y %% 2 == 1 ){
554           x <- x + 0.5
555       }
556
557       segments(
558           x, y + b,
559           x - 0.5, y + a,
560           col=color_cls,
561           lwd=cls_lwd,
562       )
563   }
564 }
565 }
566
567 points <- NULL
568 sf <- 0.35
569 a <- a * sf;
570 b <- b * sf;
571 c <- 0.5 * sf;
572 for (i in 1:nrow(somm$visual)){
573     x <- somm$visual[i,1]
574     y <- somm$visual[i,2]
575     if ( y %% 2 == 1 ){
576         x <- x + 0.5
577     }
578     points <- c(points, x, y)
579 }
580 points <- matrix( points, byrow=
    T, ncol=2 )
581
582 if( if_points == 1 ){
583     if (F){
584         for (i in 1:nrow(points)){
585             x <- points[i,1]
586             y <- points[i,2]
587
588             polygon(
589                 x=c( x + c, x + c, x, x - c, x
                    - c, x ),
590                 y=c( y + a, y - a, y - b, y -
                    a, y + a, y + b ),
591                 col=color_ptf,
592                 border=color_pte,
593                 lty=1,
594             )
595         }
596     } else {
597         symbols(
598             points[,1],
599             points[,2],
600             squares=rep(0.35,length(
                points[,1])),
601             #circles=rep(0.2,length(points

```

```

        [,1])),
602         fg="gray70",
603         bg=color_ptf,
604         inches=F,
605         add=T,
606     )
607 }
608 }
609
610 library(maptools)
611 if (is.null(labcd) == 1){
612     labcd <- pointLabel(
613         x=points[,1],
614         y=points[,2],
615         labels=word_labs,
616         doPlot=F,
617         cex=cex,
618         offset=0
619     )
620
621
622     xorg <- points[,1]
623     yorg <- points[,2]
624     #cex <- 1
625
626     if ( length(xorg) < 300 ) {
627         library(wordcloud)
628
629         # fix for "wordlayout" function
630         filename <- tempfile()
631         writeLines("wordlayout <-
            function (x, y, words, cex =
                1, rotate90 = FALSE, xlim = c
                (-Inf,
632                 Inf), ylim = c(-Inf, Inf),
                    tstep = 0.1, rstep = 0.1,
                    ...)
633     {
634         tails <- \"g|j|p|q|y\"
635         n <- length(words)
636         sdx <- sd(x, na.rm = TRUE)
637         sdy <- sd(y, na.rm = TRUE)
638         iterations <- 0
639         if (sdx == 0)
640             sdx <- 1
641         if (sdy == 0)
642             sdy <- 1
643         if (length(cex) == 1)
644             cex <- rep(cex, n)
645         if (length(rotate90) == 1)
646             rotate90 <- rep(rotate90, n)
647         boxes <- list()
648         for (i in 1:length(words)) {
649             rotWord <- rotate90[i]
650             r <- 0
651             theta <- runif(1, 0, 2 * pi)
652             x1 <- xo <- x[i]
653             y1 <- yo <- y[i]
654             wid <- strwidth(words[i],
                cex = cex[i], ...)
655             ht <- strheight(words[i],

```

```

        cex = cex[i], ...)
656 if (grepl(tails, words[i]))
657   ht <- ht + ht * 0.2
658 if (rotWord) {
659   tmp <- ht
660   ht <- wid
661   wid <- tmp
662 }
663 isOverlaped <- TRUE
664 while (isOverlaped) {
665   if (!.overlap(x1 - 0.5 *
666     wid, y1 - 0.5 * ht, wid,
667     ht, boxes) && x1 - 0.5 *
668     wid > xlim[1] && y1
669     -
670     0.5 * ht > ylim[1] && x1
671     + 0.5 * wid < xlim[2]
672     &&
673     y1 + 0.5 * ht < ylim[2])
674   {
675     boxes[[length(boxes) +
676       1]] <- c(x1 - 0.5 *
677         wid,
678         y1 - 0.5 * ht, wid, ht)
679     isOverlaped <- FALSE
680   }
681   else {
682     theta <- theta + tstep
683     r <- r + rstep * tstep /
684       (2 * pi)
685     x1 <- xo + sdx * r * cos
686       (theta)
687     y1 <- yo + sdy * r * sin
688       (theta)
689     iterations <- iterations
690       + 1
691     if (iterations > 500000){
692       boxes[[length(boxes) +
693         1]] <- c(x1 - 0.5 *
694           wid,
695           y1 - 0.5 * ht, wid, ht)
696       isOverlaped = FALSE
697     }
698   }
699 }
700 print( paste("\ iterations: \",
701   iterations) )
702 result <- do.call(rbind,
703   boxes)
704 colnames(result) <- c(\"x\",
705   \"y\", \"width\", \"ht\")
706 rownames(result) <- words
707 result
708 }
709 ", filename)
710 insertSource(filename, package="
711   wordcloud", force=FALSE)
712 nc <- wordlayout(
713   labcd$x,
714   labcd$y,

```

```

715   word_labs,
716   cex=cex * 1.25,
717   xlim=c( par( "usr" )[1], par( "
718     usr" )[2] ),
719   ylim=c( par( "usr" )[3], par( "
720     usr" )[4] )
721 )
722
723 xlen <- par("usr")[2] - par("
724   usr")[1]
725 ylen <- par("usr")[4] - par("
726   usr")[3]
727
728 segs <- NULL
729 for (i in 1:length(word_labs)) {
730   x <- ( nc[i,1] + .5 * nc[i,3] -
731     labcd$x[i] ) / xlen
732   y <- ( nc[i,2] + .5 * nc[i,4] -
733     labcd$y[i] ) / ylen
734   dst <- sqrt( x^2 + y^2 )
735   if ( dst > 0.05 ) {
736     segs <- rbind(
737       segs,
738       c(
739         nc[i,1] + .5 * nc[i,3], nc[i
740           ,2] + .5 * nc[i,4],
741         xorg[i], yorg[i]
742       )
743     )
744   }
745 }
746
747 xorg <- labcd$x
748 yorg <- labcd$y
749 labcd$x <- nc[,1] + .5 * nc[,3]
750 labcd$y <- nc[,2] + .5 * nc[,4]
751 }
752
753 if ( exists("segs" ) ) {
754   if ( is.null(segs) == F ) {
755     for (i in 1:nrow(segs) ) {
756       segments(
757         segs[i,1],segs[i,2],segs[i,3],segs[i
758           ,4],
759         col="gray60",
760         lwd=1
761       )
762     }
763   }
764 }
765
766 text(
767   labcd$x,
768   labcd$y,
769   labels=word_labs,
770   cex=cex,
771   offset=0,
772   font=text_font

```

```
751 )
752
753 if ( exists("out_coord") == F ) {
754   out_coord <- cbind(
755     labcd$x / (n_nodes-0.5),
756     labcd$y / (n_nodes-1)
757   )
758 }
759
```

```
760 points1<-head(points,n=190)
761 par(new=T)
762 plot(points1[,1],points1[,2],type="c",
763       col="red")
764
765 points2<-tail(points,n=190)
766 par(new=T)
767 plot(points2[,1],points2[,2],type="c",
768       col="blue")
```
