

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 はじめに	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 数値実験ならびに考察	27
第6章 おわりに	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

はじめに

§ 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、受容性の高いログのための研究が行われている [1]。しかし GPS は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、手動でログデータを取得するアプリケーションも多く、未だ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、ログデータを取得するという物理的負担が少ないログは多くの人に広く受け入れられると考える。ログ自体が多くの人々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、ログの在り方は改善すべきであると考える。

§ 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す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、現状のログ・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に、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、ユーザー自身的健康管理や学習 [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。

しかし、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NTT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 2.2: SmartBand 2

図 2.1: Lifelog

(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ディスプレイを行う。シースルーディスプレイの実現にはビデオシースルーディスプレイと光学シースルーディスプレイの 2 通りがあり、MOVERIO は光学シースルーディスプレイを使用することが可能である [10]。ビデオシースルーディスプレイは、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。ヘッドマウントディスプレイを用いて、カメラを通じた外の様子を見るため、タイムラグが生じ移動中や作業には向いていない。一方、光学シースルーディスプレイ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/>

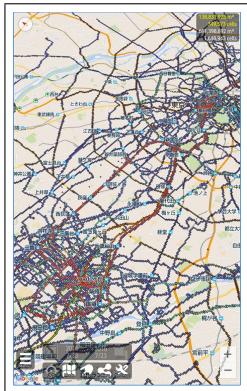


図 2.3: マッピング-GPS ログまとめて全部記録

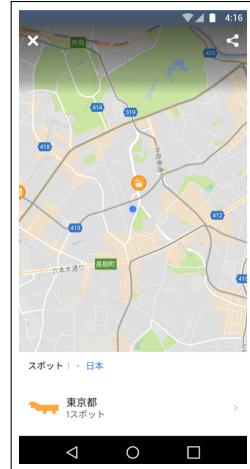


図 2.4: Swarm



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、画像が持つ様々な情報を取得することができる。その一つとして、テキスト情報の取得が特徴として挙げられる。

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。

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ます [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。



図 2.7: MOVERIO BT-300

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)

ログデータの取得を検討する。

§ 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ことができたり、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できる、KH Coder のホームページよりダウンロード¹²後...\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、Be 動詞のような一般的な語は複数回出てくるが分析には不要となるため、Stop words と

¹¹<http://khc.sourceforge.net/>

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

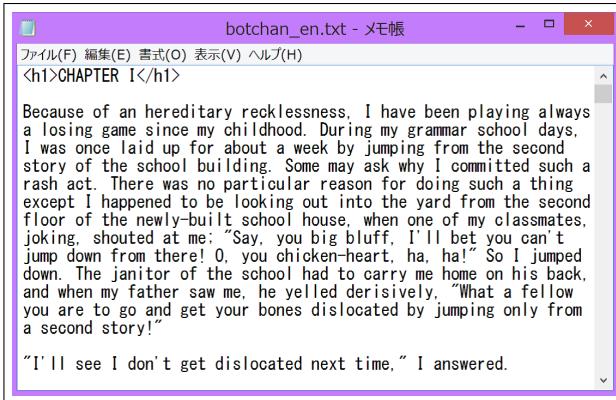


図 3.3: 「坊ちゃん」英語版テキストデータの一部



図 3.4: Stop words の一部

して指定しておくと、分析対象から外すことができる（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ない。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、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 ではウォ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。ウォード法は、計算量が多いが分類感度がいいため用いられることが多く、ウォ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

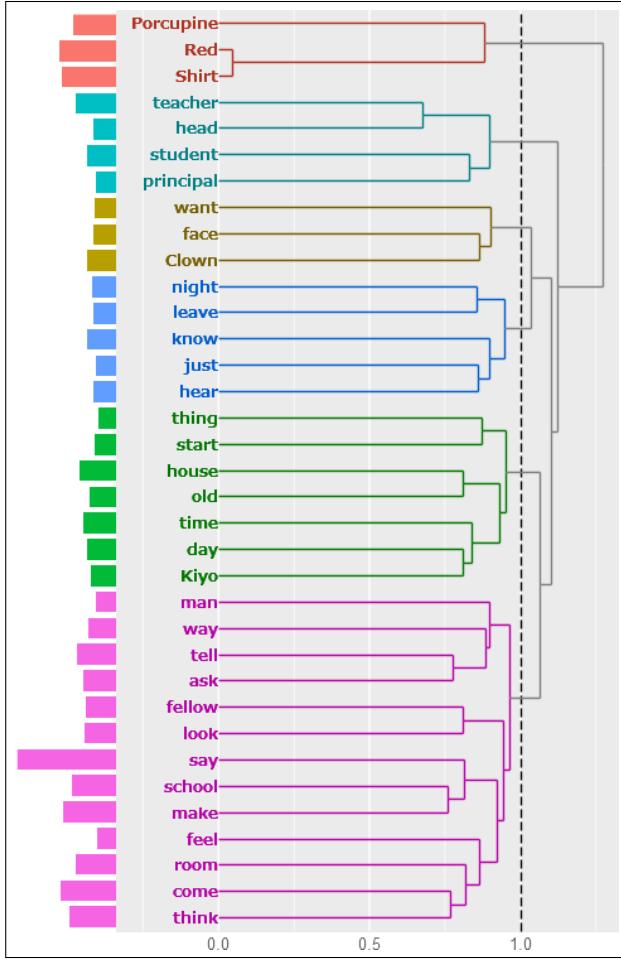


図 3.5: 「坊ちゃん」データから作成したクラスター分析

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同士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」文書の数を、「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

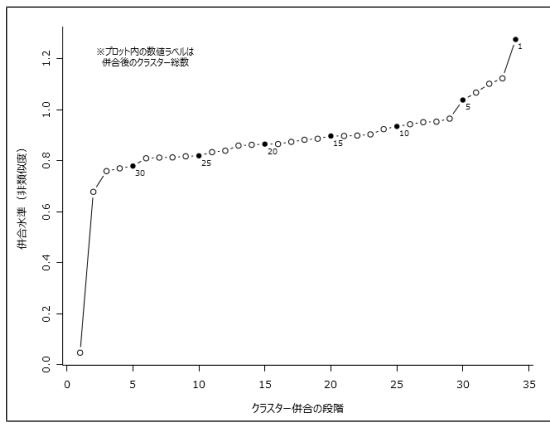


図 3.6: クラスター分析の併合標準

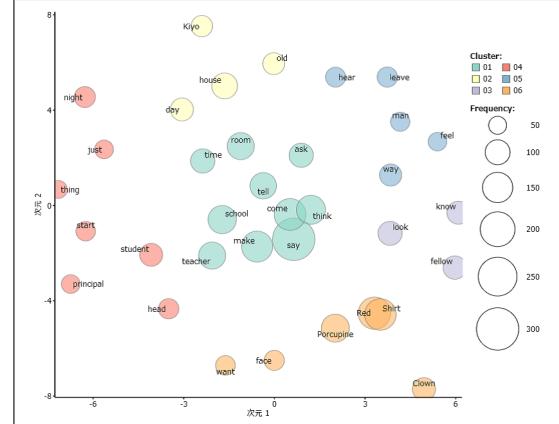


図 3.7: 「坊ちゃん」データから作成した MDS

るため軸の方向性に意味はない。

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

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。この時軸に表示されている数字は固有値と寄与率である。

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

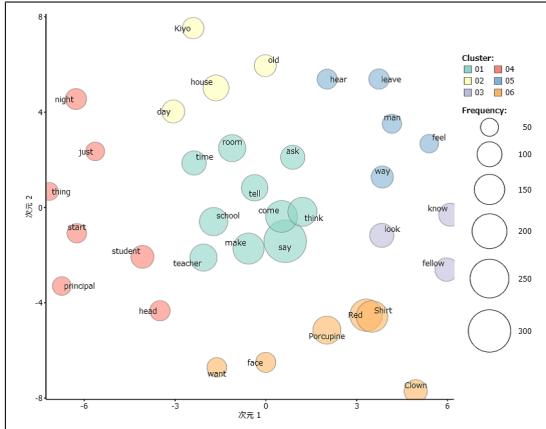


図 3.8: 「坊ちゃん」データから作成した MDS

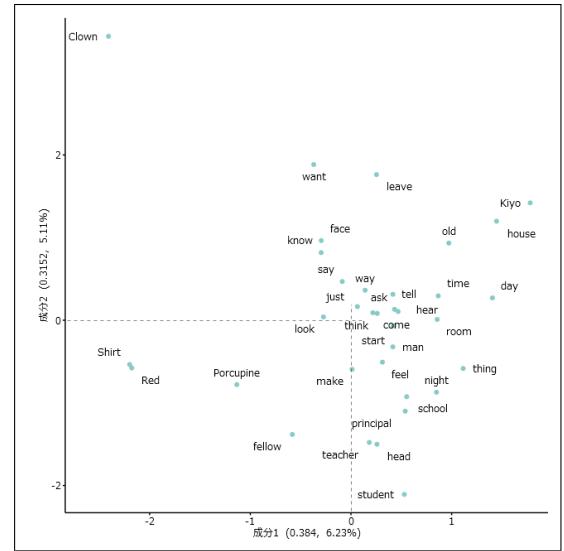


図 3.9: 「坊ちゃん」データから作成した対応分析

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 と共に関係があり、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実行していると考える。入力データベクトルと呼ばれる入力層から出力層へ

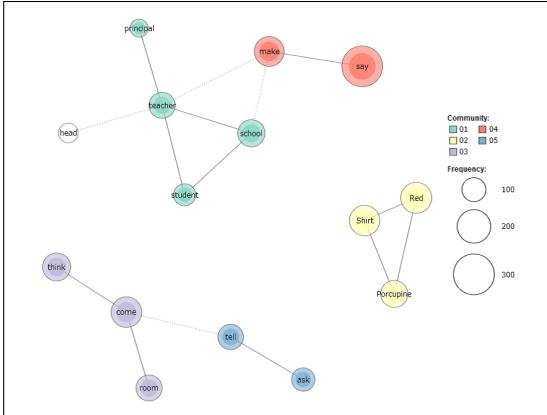


図 3.10: 「坊ちゃん」データから作成した共起ネットワーク

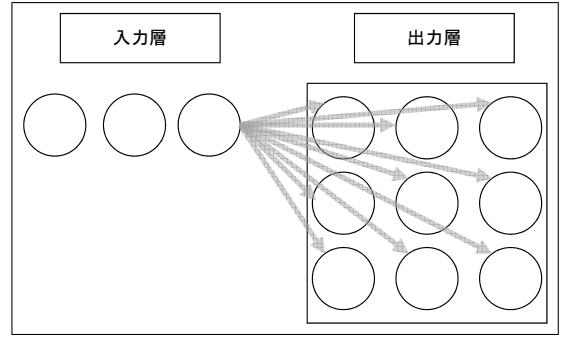


図 3.11: 入力層と出力層

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、下側に変数がとる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ウォ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。

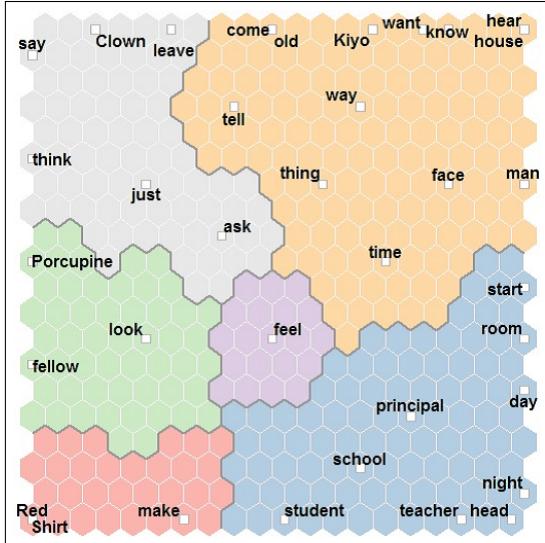


図 3.12: 「坊ちゃん」データから KH Coder で作成した SOM

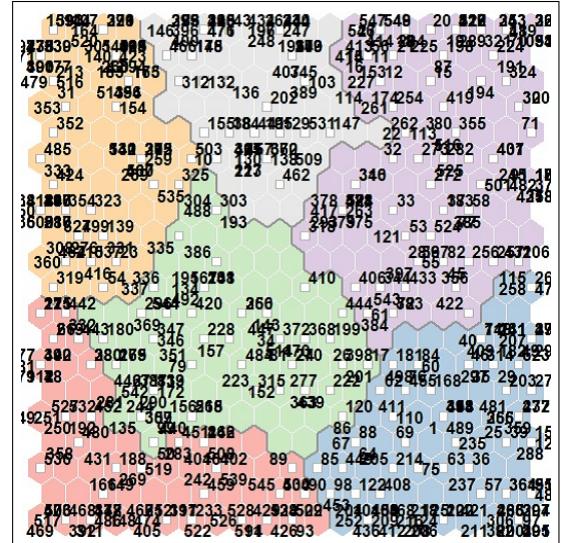


図 3.13: 「坊ちゃん」データから R で作成した SOM

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と考えられる。

KH Coder と R を用いてログデータであるテキストデータの多変量解析を行い、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、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、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る。つまり同じ行動を行っていることや、その行動がデータ内で占める割合が似ていることが類似性のあるログデータだと解釈する。また、ユーザー自身の複数のログデータの中で類似性のあるログデータが多ければ、そのログデータは平常日を表していると考えられる。一方でログデータの類似性ではなく、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

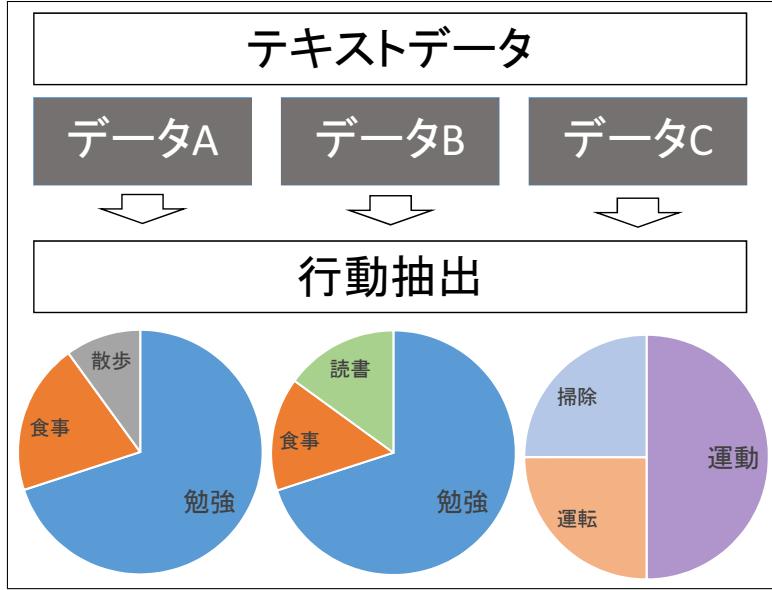


図 3.14: データ A, データ B, データ C の例

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、多くのライフソグの中でも類似したライフソグか、特徴的でイベント性のあるライフソグかわかる。

図 3.14 はテキストデータとして、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ある割合で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ソグデータであれば、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、ライフソ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。ライフソ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]。このようにライフソ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

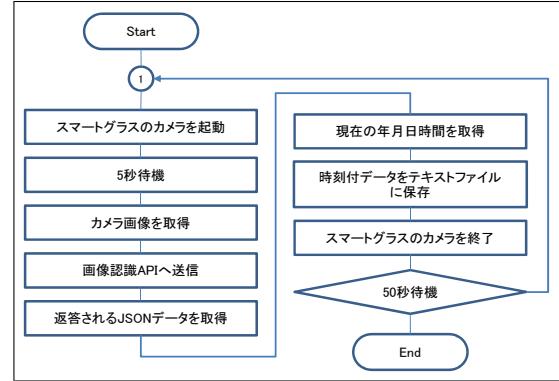


図 4.2: アプリケーションのフローチャート

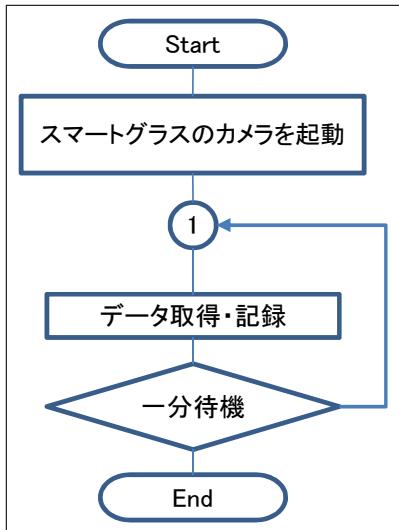


図 4.3: 省電力化前

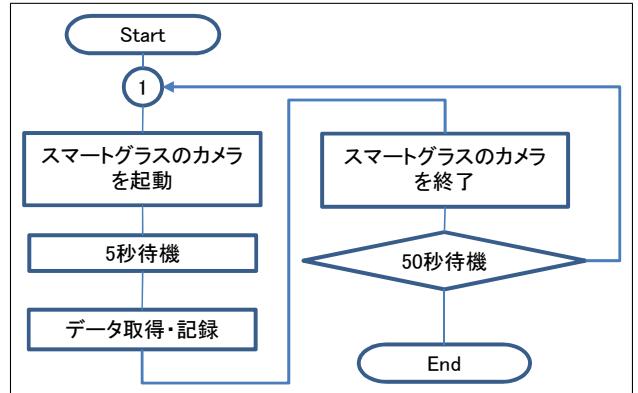


図 4.4: 省電力化後

§ 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耗について述べる。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耗よりも、連続起動の方がバッテリー消耗が大きいと考え、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、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くなると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、机上にあるノート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、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。

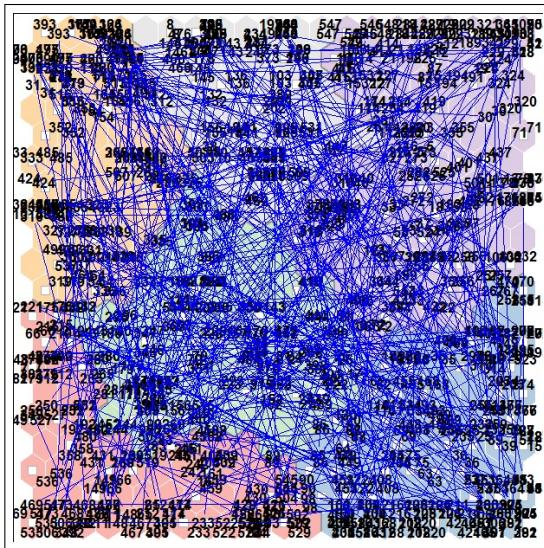


図 4.5: 「坊ちゃん」データから作成した SOM に線分を追加

```
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、ライフソ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。

数値実験ならびに考察

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作成する（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いる。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ふ。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

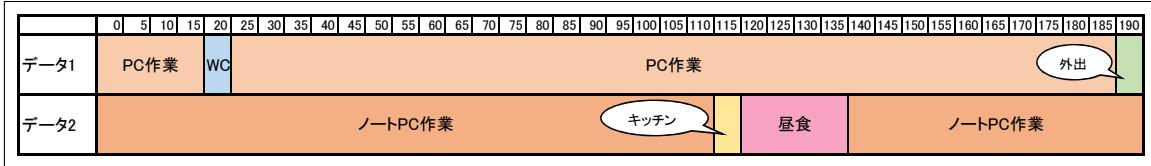


図 5.1: データ 1 とデータ 2 のタイムスケジュール

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うことがわかる。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行っ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 screen shot 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るが、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 なども含まれてしまっている。食事していることをより正確に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 3 のクラスター分析を行う。図 5.6 より、データ 1 はピンクのクラスターに近

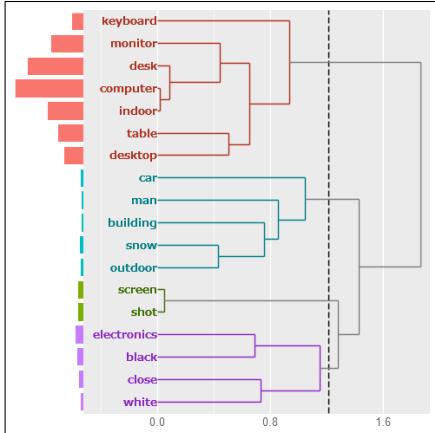


図 5.2: データ 1 のクラスター分析

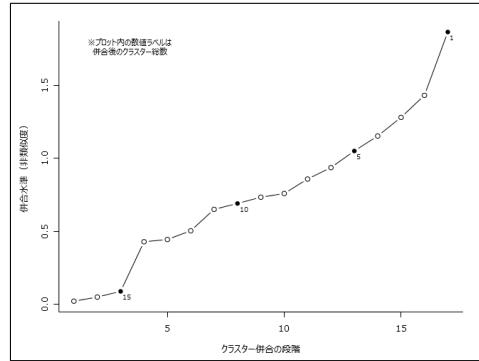


図 5.3: データ 1 の併合標準

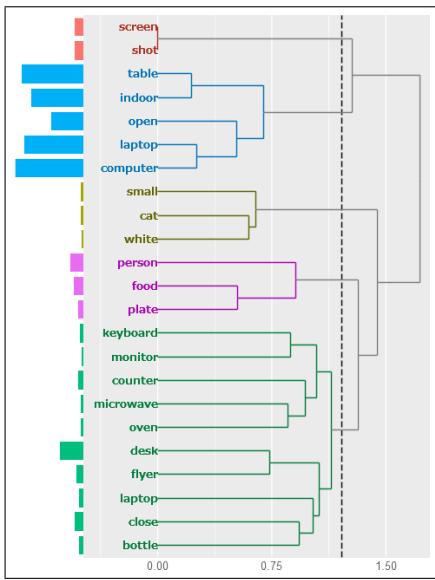


図 5.4: データ 2 のクラスター分析

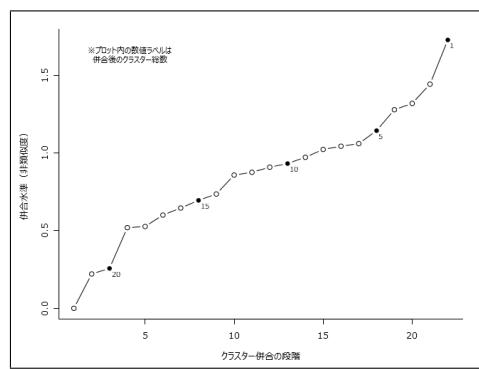


図 5.5: データ 2 の併合標準

く、同じクラスターには computer や desktop という単語がある。また、データ 2 は、laptop や screen shot という単語が多く含まれる。このことから、データ 1 のほうが computer との関係性が強いのではないかと考察できる。

次に、データ 1 とデータ 2 の MDS を行う。図 5.7 はデータ 1 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 4 とした。PC 作業に関する 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 に近いものと考える。

図 5.8 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

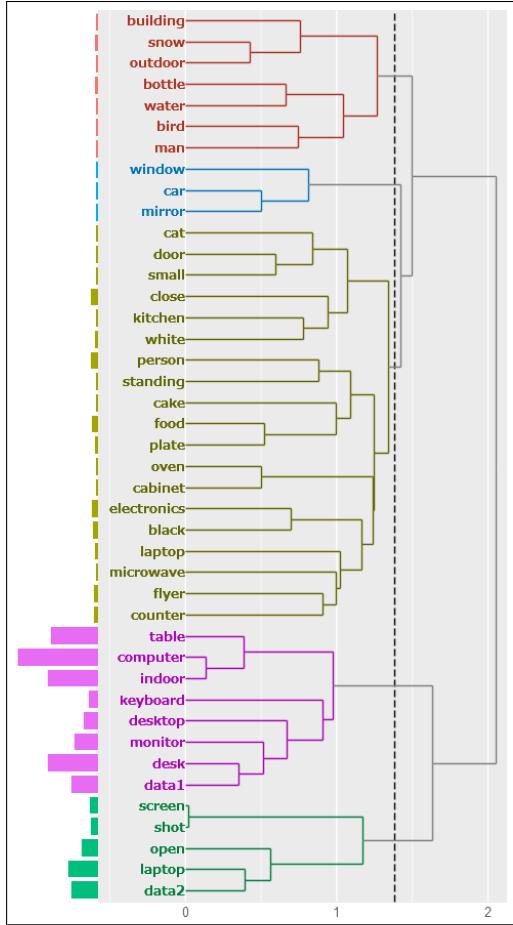


図 5.6: データ 3 のクラスター分析

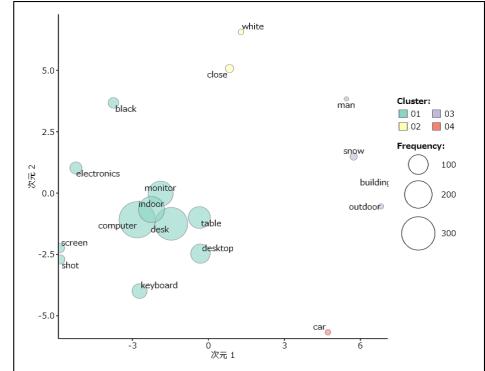


図 5.7: データ 1 の MDS

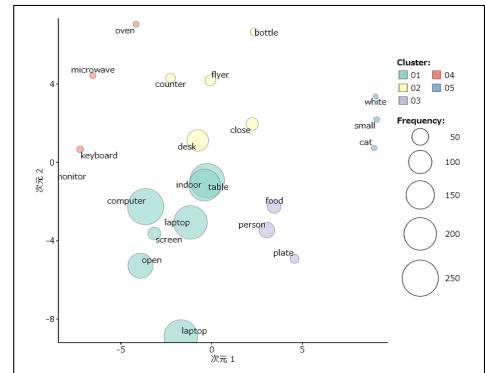


図 5.8: データ 2 の MDS

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めノイズではないかと考えられる。

図 5.7 と図 5.8 を比較すると、データ 2 のほうがクラスター間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。本実験のデータでは MDS で新しい発見や考察は難しい結果となった。

次に、データ 1 とデータ 2 の対応分析を行う。図 5.9 はデータ 1 から作成した対応分析である。図 5.9 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。

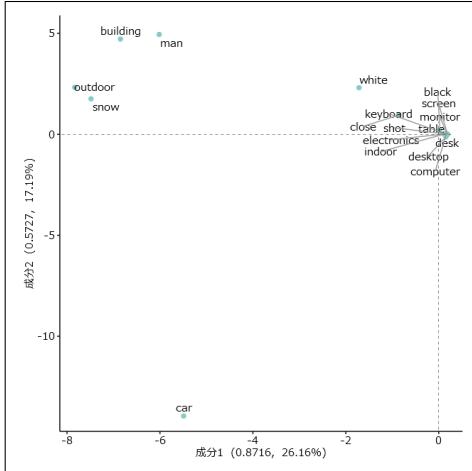


図 5.9: データ 1 の対応分析

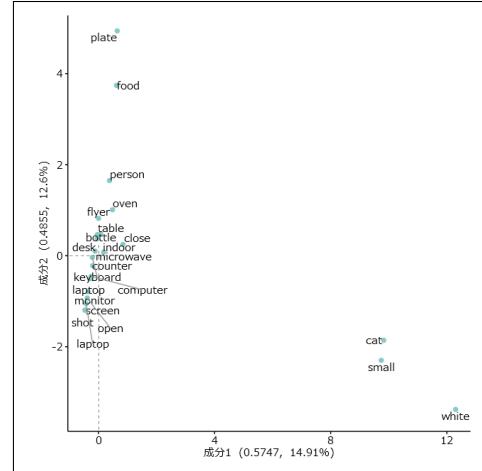


図 5.10: データ 2 の対応分析

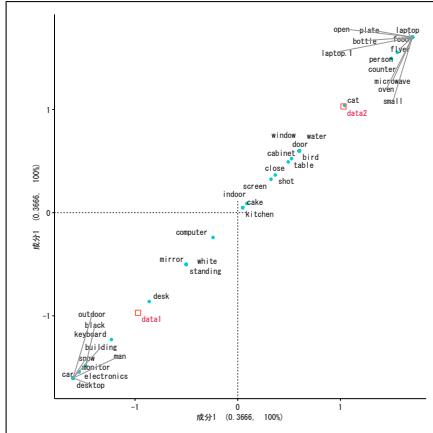


図 5.11: データ 3 の対応分析

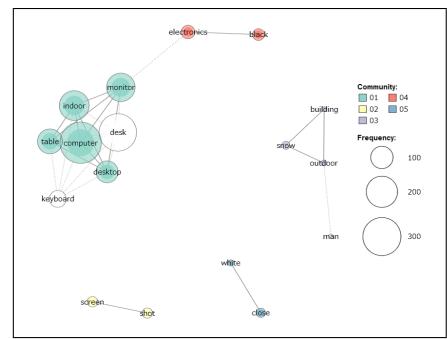


図 5.12: データ 1 の共起ネットワーク

図 5.10 はデータ 2 から作成した対応分析である。図 5.9 と図 5.10 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。

さらに、データ 1 とデータ 2 の関係性を同時に output することができるため、データ 3 の対応分析を行う。5.11 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 5.12 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 指数が 0.2 以上の共起関係を描画している。図 5.12 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

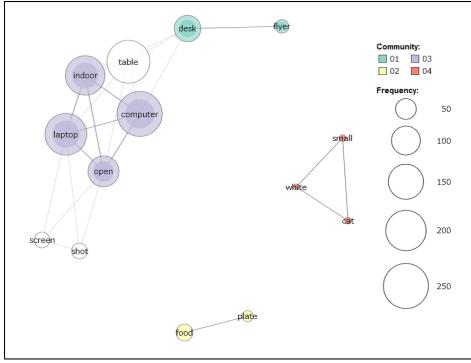


図 5.13: データ 2 の共起ネットワーク

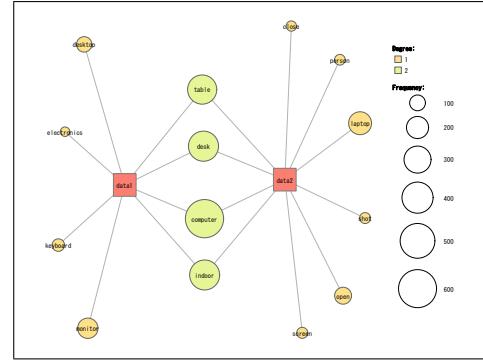


図 5.14: データ 3 の共起ネットワーク

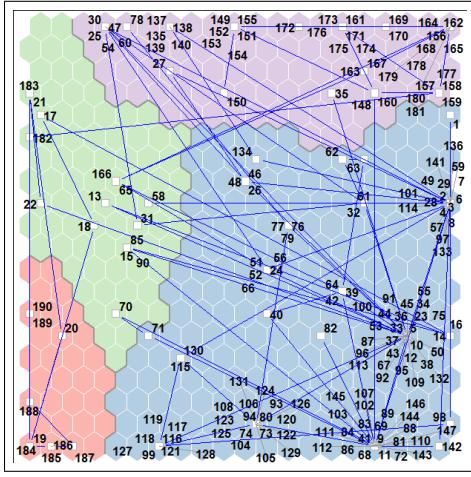


図 5.15: データ 1 の SOM

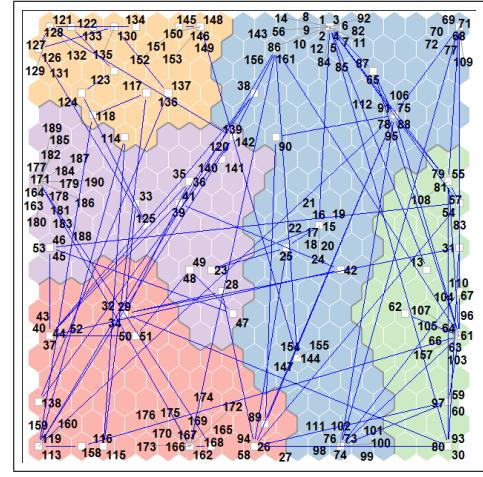


図 5.16: データ 2 の SOM

共起関係があることがわかる。図 5.13 はデータ 2 から作成した共起ネットワークであり、図 5.12 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表示するため、データ 3 の共起ネットワークを行う。5.14 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。

最後に今までの解析を踏まえて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 5.15 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。

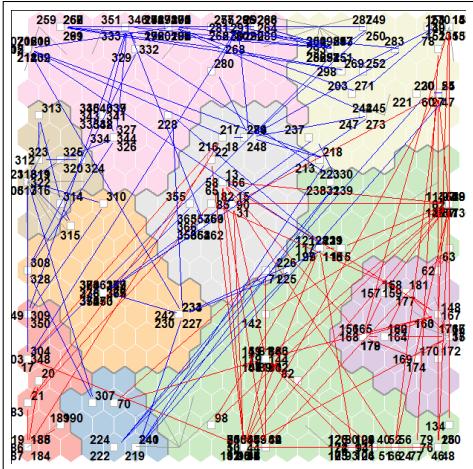


図 5.17: データ 3 の SOM

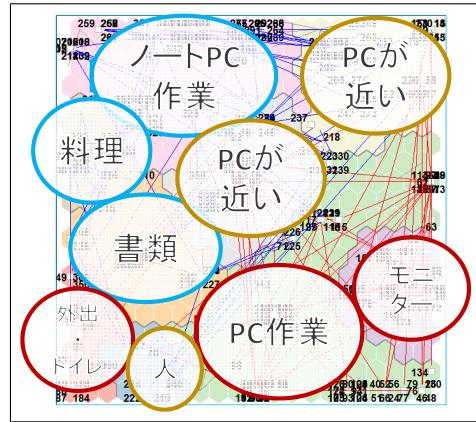


図 5.18: データ 3 の SOM に行動を追記した SOM

図5.16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5.17の、クラスターの分かれ方を5.18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る。

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つてゐると言える. 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に, 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言える.

おわりに

本研究の目的は、多くの人に広く受け入れられるライログとして、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に、ライ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

福嶋 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－画像処理による食事記録－”, 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, 横原裕大, 芝星帆, 真島一貴, 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ーフェース”, 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: 一デザインの視覚的要素を中心にー”, 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』のケインズ的要素-テキストマイニングを加味した量的・質的分析-”, 2017.
- [27] 斎藤堯幸, “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.gifunct.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ティングシステム (UBI), Vol. 2013, No. 17, pp. 1–7, 2013.

付録

A. 1 ライログデータ取得アプリケーションのソースコード

ライ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/
47   describe";
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
59     () + "\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(amap)
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   omi=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(
   "x0",
   "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$x1[i]
146            )
147          )
148        ]
149      }
150      if (is.null(seg_bl) == F){
151        colnames(seg_bl) <- c("x0", "y0",
152          "x1", "y1")
153        seg_bl <- as.data.frame(seg_bl)
154        if ( max(seg_bl$y1) > cutpoint ){
155          p <- p + geom_hline(
156            yintercept = cutpoint,
157            colour="black",
158            linetype=5,
159            size=0.5
160          )
161        }
162      }
163  colnames(seg_cl) <- c("x0", "y0",
164    "x1", "y1", "c")
165  seg_cl <- as.data.frame(seg_cl)
166  seg_cl$c <- col_vec[seg_cl$c]
167  p <- p + geom_text(
168    data=data.frame(
169      x=label(ddata)$x,
170      y=label(ddata)$y,
171      text=labels[ as.numeric( as.
172        vector( ddata$labels$text
173        ) ) ],
174      cols= col_vec[ memb[ as.
175        numeric( as.vector( ddata
176          $labels$text ) ) ] ]
177    ),
178    aes_string(
179      x="x",
180      y="y",
181      label="text",
182      colour="cols"
183    ),
184    hjust=1,
185    angle =0,
186    family = font.family,
187    fontface = "bold",
188    size = 5 * 0.85 * font_size
189  )
190  p <- p + geom_segment(
191    data=seg_cl,
192    aes_string(x="x0", y="y0", xend
193      ="x1", yend="y1", colour="c
194      "),
195    size=0.5
196  )
197  } else {
198    memb <- rep( c("a"), length(
199      labels) )
200    p <- p + scale.colour.manual(
201      values=c("black"))
202    seg_bl <- ddata$segment
203    col_vec <- c("001")
204    p <- p + geom_text(
205      data=data.frame(
206        x=label(ddata)$x,
207        y=label(ddata)$y,
208        text=labels[ as.numeric( as.
209          vector( ddata$labels$text
210          ) ) ],
211        cols= col_vec[ memb[ as.
212          numeric( as.vector( ddata
213            $labels$text ) ) ] ]
214      ),
215      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$labels$`text` ) ) ] ]
236   ),
237   aes_string(
238     x="x",
239     y="y",
240     label="text",
241     colour="cols"
242   ),
243   hjust=1,
244   angle =0,
245   family = font_family,
246   fontface = "bold",
247   size = 5 * 0.85 * font_size
248 )
249 y_max <- max( ddata$segment$y1 )
250 y_min <- 0.2
251 # "strwidth" crashes if the device is
252 #   cairo_pdf or cairo_ps
253 if (
254   is.na(dev.list()["cairo_pdf"])
255   && is.na(dev.list()["cairo_ps"]))
256 ){
257   y_min <- max(
258     strwidth(
259       labels[ as.numeric( as.vector(
260         ddata$labels$text ) ) ],
261       units = "figure",
262     )
263   )
264   y_min <- ( 6 * y_max * y_min ) / (
265     5 - 6 * y_min )
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$labels$`text` ) + 0.5 , 1 - 0.5 )
284   )
285   p <- p + scale_y_continuous(
286     limits=c(y_min,y_max),
287     breaks=c(0,b1/2,b1),
288     expand = c(0.02,0.02)
289   )
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
303   if (n_cls <= 1){
304     p <- p + theme(
305       axis.text.y = element_blank(),
306       axis.text.x = element_text(size
307         =12,colour="black"),
308       axis.ticks = element_line(colour =
309         "black"),
310       #panel.grid.major = theme_blank
311       (),
312       #panel.grid.minor = theme_blank
313       (),
314       #panel.background = theme_blank
315       (),
316       axis.line = element_line(colour =
317         "black"))

```

```

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), "lines")
321   )
322
323 bard <- data.frame(
324   nm <- labels[ as.numeric( as.
325     vector( ddata$labels$text ) ) ],
326   ht <- freq[ as.numeric( as.
327     vector( ddata$labels$text ) ) ],
328   cl <- col_vec[ memb[ as.
329     numeric( as.vector( ddata$labels$text ) ) ] ],
330   od <- nrow(d):1
331 )
332
333 if (n_cls <= 1){
334   bard$cl <- "001"
335 }
336
337 p2 <- NULL
338 p2 <- ggplot()
339 p2 <- p2 + geom_bar(
340   stat="identity",
341   position = "identity",
342   width=0.75,
343   data=bard,
344   aes(
345     x=reorder(od,od),
346     y=ht,
347     fill=cl
348   )
349 p2 <- p2 + coord_flip()
350 p2 <- p2 + scale_y_reverse(
351   expand = c(0,0))
352 p2 <- p2 + scale_x_discrete(
353   expand = c(0,0) )
354 p2 <- p2 + theme(
355   axis.title.y = element_blank(),
356   axis.title.x = element_blank(),
357   axis.ticks = element_blank(),
358   axis.text.y = element_blank(),
359   axis.text.x = element_text(size
360     =12,colour="white"),
361   legend.position = "none",
362   panel.background = element_rect
363     (fill="white", colour="white"
364      ),
365   panel.grid.major = element_
366     blank(),
367   panel.grid.minor = element_
368     blank()
369 )
370
371 margin <- 0.002 * nrow(d) +
372   0.00001 * nrow(d)^2 - 0.12
373 p2 <- p2 + theme(
374   plot.margin = unit(c
375     (0.25,0,0.25,0), "lines") # r:
376     -0.75
377 )
378
379 grid.newpage()
380 pushViewport(viewport(layout=
381   grid.layout(1,2, width=c(1,5)))
382 )
383 print(p, vp=viewport(layout.pos.
384   row=1, layout.pos.col=2) )
385 print(p2, vp=viewport(layout.pos.
386   row=1, layout.pos.col=1) )
387 } else {
388   print(p)
389 }
390
391 if (
392   is.na(dev.list()["pdf"])
393   && is.na(dev.list()["postscript"]
394   ])
395   && is.na(dev.list()["cairo_pdf"])
396   && is.na(dev.list()["cairo_ps"]))
397 }
398
399 if ( grepl("darwin", R.version$  

400   platform) ){
401   quartzFonts(HiraKaku=quartzFont  

402     (rep("Meiryo UI",4)))
403   grid.gedit("GRID.text", grep=
404     TRUE, global=TRUE, gp=
405     gpar(fontfamily="HiraKaku"))
406 }
407
408 detach("package:ggdendro", unload
409   =T)
410
411 # for clickable image map
412 exp <- (y_max - y_min) * 0.02
413 coord <- cbind(
414   (1 / 6 + 5 / 6 * -1 * (y_min -
415     exp) / ((y_max + exp) - (y_
416     min - exp))) * 1.03,
417   1:length(ddata$labels$text) /
418   length(ddata$labels$text)
419 )
420
421 rownames(coord) <-
422   labels[ as.numeric( as.vector(
423     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(amap)
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
         dimension 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
         dimension 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 }
```

```

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   # SMA COF
88   library(smacof)
89   c <- mds(dj, ndim=dim_n, type="  

89       ordinal", itmax=3000)
90   if (random_starts == 1){
91     print("Running random starts  

91         . . .")
92     set.seed(123)
93     for (i in 1:200){ # 200 -> 246sec
94       run <- mds(dj, ndim=dim_n,  

94           type="ordinal", init = "  

94             random", itmax=3000)
95       if (run$stress < c$stress){
96         c <- run
97         print(paste("random start  

97             #", i, ":", c$stress, sep=  

97                 ""))
98       }
99     }
100   }
101   cl <- c$conf
102 }
103 save(d,cl,dim_n, file="C:/  

103     khcoder3/config/R-bridge/  

103     khc6_word_mds")
104 use_alpha <- 1
105
106 if ( exists("saving_emf") ||  

106     exists("saving_eps")){
107   use_alpha <- 0
108 }
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()$  

153       major ) >= 3 )
154     && ( as.numeric( R.Version()$  

154       minor ) >= 1.0 )
155   ){ # >= R 3.1.0
156     hcl <- hclust(djj,method="ward.  

156       D2")
157   } else { # <= R 3.0
158     djj <- djj^2
159     hcl <- hclust(djj,method="ward"  

159       )
160     #hcl$height <- sqrt( hcl$height )
161   }
162 }
163
164 b_size <- NULL
165
166 for (i in rownames(cl)){
167   if ( is.na(i) || is.null(i) || is.nan(i) )
168   {
169     b_size <- c( b_size, 1 )
170   } else {
171     b_size <- c( b_size, sum( d[i] ) )
172   }
173 }
174
175 if (plot_mode == "color") {
176
177   png_width <- 822
178   png_height <- 640
179   if (png_width > png_height){
180     png_width <- png_width - 0.16
180       * 1 * bubble_size / 100 * png-
180       width
181   }

```

```

182 dpi <- 72 * min(png_width, png-
183     width) / 640 * 1
184 p_size <- 12 * dpi / 72;
185 png("temp.png", width=png_width,
186     height=png_height, unit="px",
187     pointsize=p_size)
188 #
189 #if ( exists("PERL_font_family") ){
190 # par(family=PERL_font_family)
191 #
192 plot(cl)
193 library(maptools)
194 labcd <- pointLabel(
195     x=cl[,1],
196     y=cl[,2],
197     labels=rownames(cl),
198     cex=font_size,
199     offset=0,
200     doPlot=F
201 )
202
203 xorg <- cl[,1]
204 yorg <- cl[,2]
205 cex <- font_size
206 segs <- NULL
207
208 if ( length(xorg) < 300 ) {
209     library(wordcloud)
210
211 # fix for "wordlayout" function
212 filename <- tempfile()
213 writeLines("wordlayout <-
214     function (x, y, words, cex =
215         1, rotate90 = FALSE, xlim = c
216         (-Inf,
217          Inf), ylim = c(-Inf, Inf),
218          tstep = 0.1, rstep = 0.1,
219          ...)
220
221 tails <- \g|j|p|q|y\
222 n <- length(words)
223 sdx <- sd(x, na.rm = TRUE)
224 sdy <- sd(y, na.rm = TRUE)
225 iterations <- 0
226 if (sdx == 0)
227     sdx <- 1
228 if (sdy == 0)
229     sdy <- 1
230 if (length(cex) == 1)
231     cex <- rep(cex, n)
232 if (length(rotate90) == 1)
233     rotate90 <- rep(rotate90, n)
234 boxes <- list()
235 for (i in 1:length(words)) {
236     rotWord <- rotate90[i]
237     r <- 0
238     theta <- runif(1, 0, 2 * pi)
239     x1 <- xo <- x[i]
240     y1 <- yo <- y[i]
241     wid <- strwidth(words[i],
242
243         cex = cex[i], ...)
244         ht <- strheight(words[i],
245             cex = cex[i], ...)
246         if (grepl(tails, words[i]))
247             ht <- ht + ht * 0.2
248         if (rotWord) {
249             tmp <- ht
250             ht <- wid
251             wid <- tmp
252         }
253         isOverlaped <- TRUE
254         while (isOverlaped) {
255             if (!overlap(x1 - 0.5 *
256                 wid, y1 - 0.5 * ht, wid,
257                 ht, boxes) && x1 - 0.5 *
258                 wid > xlim[1] && y1 -
259                 0.5 * ht > ylim[1] && x1
260                 + 0.5 * wid < xlim[2]
261                 &&
262                 y1 + 0.5 * ht < ylim[2])
263             {
264                 boxes[[length(boxes) +
265                     1]] <- c(x1 - 0.5 *
266                     wid,
267                     y1 - 0.5 * ht, wid, ht)
268                 isOverlaped <- FALSE
269             }
270         else {
271             theta <- theta + tstep
272             r <- r + rstep * tstep /
273                 (2 * pi)
274             x1 <- xo + sdx * r * cos
275                 (theta)
276             y1 <- yo + sdy * r * sin
277                 (theta)
278             iterations <- iterations
279                 + 1
280             if (iterations > 500000){
281                 boxes[[length(boxes) +
282                     1]] <- c(x1 - 0.5 *
283                     wid,
284                     y1 - 0.5 * ht, wid, ht)
285                 isOverlaped = FALSE
286             }
287         }
288     }
289     print( paste("iterations: ",
290                 iterations) )
291     result <- do.call(rbind,
292         boxes)
293     colnames(result) <- c(\"x\",
294         \"y\", \"width\", \"ht\")
295     rownames(result) <- words
296     result
297   }
298   ", filename)
299   insertSource(filename, package="wordcloud", force=FALSE)
300

```

```

276 nc <- wordlayout(
277   labcd$x,
278   labcd$y,
279   rownames(cl),
280   cex=cex * 1.25,
281   xlim=c( par("usr")[1], par("
282     usr")[2]),
283   ylim=c( par("usr")[3], par("
284     usr")[4] )
285 )
286 xlabel <- par("usr")[2] - par("
287   usr")[1]
288 ylabel <- par("usr")[4] - par("
289   usr")[3]
290 for (i in 1:length(rownames(cl)) {
291   x <- ( nc[i,1] + .5 * nc[i,3] -
292     labcd$x[i] ) / xlabel
293   y <- ( nc[i,2] + .5 * nc[i,4] -
294     labcd$y[i] ) / ylabel
295   dst <- sqrt( x^2 + y^2 )
296   if ( dst > 0.05) {
297     segs <- rbind(
298       segs,
299       c(
300         nc[i,1] + .5 * nc[i,3],
301         nc[i,2] + .5 * nc[i,4],
302         xorg[i],
303         yorg[i]
304       )
305     )
306   }
307   xorg <- labcd$x
308   yorg <- labcd$y
309   labcd$x <- nc[,1] + .5 * nc[,3]
310   labcd$y <- nc[,2] + .5 * nc[,4]
311 }
312 dev.off()
313 library(grid)
314 library(ggplot2)
315
316 font_family <- "Meiryo UI"
317
318 if ( exists("PERL_font_family") ){
319   font_family <- PERL_font_family
320 }
321
322 if (use_alpha == 1){
323   alpha_value = 0.6
324 } else {
325   alpha_value = 1
326 }
327
328 if ( n_cls > 0 ){
329   cls_labels <- cutree(hcl, k=n_cls)
330   cls_labels <- formatC(cls_labels,
331     width=2,flag="0")
332   cls_labels <- paste(cls_labels, " "
333   )
334 } else {
335   cls_labels <- "cluster 1"
336 }
337 cl2 <- data.frame(
338   d1 = cl[,1],
339   d2 = cl[,2],
340   s = b_size,
341   col_f = cls_labels,
342   lx = labcd$x,
343   ly = labcd$y,
344   labels = rownames(cl),
345   stringsAsFactors = F
346 )
347
348 g <- ggplot()
349
350 # Plot
351 if ( bubble == 1 ){
352   g <- g + geom_point(
353     data=cl2,
354     aes(x=d1, y=d2, size=s, colour=
355       col_f, fill=col_f),
356     shape=21,
357     colour="gray40",
358     alpha=alpha_value
359   )
360   g <- g + scale_size_area(
361     max_size= 30 * bubble_size / 100,
362     guide = guide_legend(
363       title = "Frequency:",
364       override.aes = list(colour =
365         "black", alpha=1),
366       label.hjust = 1,
367       order = 2
368     )
369   )
370 } else {
371   if ( n_cls > 0 ){
372     g <- g + geom_point(
373       data=cl2,
374       aes(x=d1, y=d2, colour=col_f,
375         fill=col_f),
376       size=5.5,
377       shape=21,
378       colour="gray40",
379       alpha=alpha_value
380     )
381   } else {
382     g <- g + geom_point(
383       data=cl2,
384       aes(x=d1, y=d2),
385       size=2,
386       shape=16,
387       colour="mediumaquamarine"
388     )

```

```

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

```

```

484 out_coord[,2] <- ( out_coord[,2] +
485   add ) / div
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     ) ){
493   saving_file <- 1
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], "mm",
520         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 } else { # ggplot2 >= 2.2.0
532   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
2   =T, nrow=190, ncol=19 )
3 d <- d[,-1]
4 colnames(d) <- c("desk",...省略...,"outdoor")
5 doc_length_mtr <- matrix( c(
5   70,18,...省略...45,17), ncol=2,
5   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 <-
16   v_pch[rowSums(d) > 0] }
17 doc_length_mtr <- subset(doc_
17   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

```

```

29 flt <- 0
30 flw <- 60
31 bubble_plot <- 0
32 biplot <- 0
33 cex=1
34 use_alpha <- 1
35 show_origin <- 1
36 scaling <- "none"
37
38 if ( exists("saving_emf") ||
      exists("saving_eps") ){
39   use_alpha <- 0
40 }
41 name_dim <- '\u6210\u5206'
42 name_eig <- '\u56fa\u6709\u5024'
43 name_exp <- '\u5bc4\u4e0e\u7387
,
44 library(MASS)
45
46 # Filter words by chi-square value
47 if ( (flw > 0) && (flw < ncol(d)) ){
48   sort <- NULL
49   for (i in 1:ncol(d) ){
50     # print( paste(colnames(d)[i],
51       chisq.test( cbind(d[,i], n_total
52       - d[,i]) )$statistic) )
53     sort <- c(
54       sort,
55       chisq.test( cbind(d[,i], n_total -
56         d[,i]) )$statistic
57     )
58   }
59   d <- d[order(sort,decreasing=T)]
60   d <- d[,1:flw]
61
62   d <- subset(d, rowSums(d) > 0)
63   if(exists("doc_length_mtr")){
64     doc_length_mtr <- subset(doc_
65       length_mtr, rowSums(d) > 0)
66     n_total <- doc_length_mtr[,2]
67   }
68 }
69
70 d_max <- min( nrow(d), ncol(d) )
71   - 1
72 if (d_x > d_max){
73   d_x <- d_max
74 }
75 if (d_y > d_max){
76   d_y <- d_max
77 }
78
79 c <- corresp(d, nf=d_max )
80
81 if (d_max == 1){
82   c$cscore <- as.matrix( c$cscore )
83   c$rscore <- as.matrix( c$rscore )
84   colnames(c$cscore) <- c("X1")
85   colnames(c$rscore) <- c("X1")
86 }
87
88 # Dilplay Labels only for distinctive
89 # words
90 if ( (flt > 0) && (flt < nrow(c$cscore
91   )) ){
92   sort <- NULL
93   limit <- NULL
94   names <- NULL
95   ptype <- NULL
96
97   # compute distance from (0,0)
98   for (i in 1:nrow(c$cscore) ){
99     sort <- c(sort, c$cscore[i,d_x] ^
100       2 + c$cscore[i,d_y] ^ 2 )
101   }
102
103   # Put labels to top words
104   limit <- sort[order(sort,
105     decreasing=T)][flt]
106   for (i in 1:nrow(c$cscore) ){
107     if ( sort[i] >= limit ){
108       names <- c(names,
109         rownames(c$cscore)[i])
110       ptype <- c(ptype, 1)
111     } else {
112       names <- c(names, NA)
113       ptype <- c(ptype, 2)
114     }
115   }
116   rownames(c$cscore) <- names;
117 } else {
118   ptype <- 1
119 }
120
121 pch_cex <- 1
122 if ( v_count > 1 ){
123   pch_cex <- 1.25
124 }
125
126 # Zooming area near the origin
127 log_conv <- function(x, y, a){
128   log_base <- 10
129
130   # Find Cosine theta
131   OA <- sqrt( x^2 + y^2 )
132   OA[OA == 0] <-
133     0.00000000000000000000000000000001
134   Cos <- x / OA
135
136   # Convert OA
137   OA <- log(OA + 1, log_base)
138   OA <- OA * a
139   OA <- log(OA + 1, log_base)
140   OA <- OA * a
141   OA <- log(OA + 1, log_base)
142
143   # Find OB
144   OB <- Cos * OA
145
146   # Find AB
147   AB = sqrt( OA^2 - OB^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$cscore[,d_x], c$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$rscore[,d_x], c$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$cor[i]
162 c$rscore[,i] <- c$rscore[,i] * c$cor[i]
163 }
164 asp <- 1
165 } else if (scaling == "symbi"){
166 for (i in 1:d_max){
167 c$cscore[,i] <- c$cscore[,i] * sqrt(c$cor[i])
168 c$rscore[,i] <- c$rscore[,i] * sqrt(c$cor[i])
169 }
170 asp <- 1
171 }
172
173 k <- c$cor^2
174 txt <- cbind( 1:length(k), round(k,4), round(100*k / sum(k),2) )
175 colnames(txt) <- c(name_dim, name_eig, name_exp)
176 print( txt )
177 inertias <- round(k,4)
178 k <- round(100*k / sum(k),2)
179 font_size <- 1
180 resize_vars <- 1
181 bubble_size <- 100
182 labcd <- NULL
183
184 plot_mode <- "color"
185
186
187
188 library(ggplot2)
189
190 font_family <- "Meiryo UI"
191
192 if ( exists("PERL_font_family") ){
193 font_family <- PERL_font_family
194 }
195
196 if ( exists("bs_fixed") == F ) {
197 bubble_size <- bubble_size / 1.3
198 bs_fixed <- 1
199 }
200
201 if (biplot == 1 && plot_mode != "vars"){
202 cb <- rbind(
203 cbind(c$cscore[,d_x], c$cscore[,d_y], ptype),
204 cbind(c$rscore[,d_x], c$rscore[,d_y], v_pch)
205 )
206 } else if (plot_mode == "vars") {
207 cb <- cbind(c$rscore[,d_x], c$rscore[,d_y], v_pch)
208 } else {
209 cb <- cbind(c$cscore[,d_x], c$cscore[,d_y], ptype)
210 }
211
212 if ( (is.null(labcd) && plot_mode != "dots" ) || plot_mode == "vars" ){
213 png_width <- 640
214 png_height <- 640
215 png_width <- png_width - 0.16 *
1.3 * bubble_size / 100 * png_width
216 dpi <- 72 * min(png_width, png_height) / 640 * 1.3
217 p_size <- 12 * dpi / 72;
218 png("temp.png", width=png_width,
height=png_height, unit="px",
pointsize=p_size)
219
220 #if ( exists("PERL_font_family") ){
221 # par(family=PERL_font_family)
222 #
223
224 plot(
225 x=c(c$cscore[,d_x],c$rscore[,d_x]),
226 y=c(c$cscore[,d_y],c$rscore[,d_y]),
227 asp=asp
228 )
229
230
231 library(maptools)
232 labcd <- pointLabel(
233 x=cb[,1],
234 y=cb[,2],
235 labels=rownames(cb),
236 cex=font_size,
237 offset=0,
238 doPlot=F
239 )

```

```

240
241 xorg <- cb[,1]
242 yorg <- cb[,2]
243 #cex <- 1
244
245 n_words_chk <- c( length(c$cscore
246   [,d_x]) )
246 if (ft > 0) {
247   n_words_chk <- c(n_words_chk,
248     ft)
249 }
250 if (flw > 0) {
251   n_words_chk <- c(n_words_chk,
252     flw)
253 }
254 if (
255   ( biplot == 0 ) && (min(n_
256     words_chk) < 300) )
257 || (
258   (biplot == 1)
259   && ( min(n_words_chk) < 300
260     )
261   && ( length(c$rscore[,d_x]) <
262     r_max )
263 )
264 }
265 library(wordcloud)
266
267 # fix for "wordlayout" function
268 filename <- tempfile()
269 writeLines("wordlayout <-
270   function (x, y, words, cex =
271     1, rotate90 = FALSE, xlim = c
272     (-Inf,
273     Inf), ylim = c(-Inf, Inf),
274     tstep = 0.1, rstep = 0.1,
275     ...)
276 {
277   tails <- \g|j|p|q|y\
278   n <- length(words)
279   sdx <- sd(x, na.rm = TRUE)
280   sdy <- sd(y, na.rm = TRUE)
281   iterations <- 0
282   if (sdx == 0)
283     sdx <- 1
284   if (sdy == 0)
285     sdy <- 1
286   if (length(cex) == 1)
287     cex <- rep(cex, n)
288   if (length(rotate90) == 1)
289     rotate90 <- rep(rotate90, n)
290   boxes <- list()
291   for (i in 1:length(words)) {
292     rotWord <- rotate90[i]
293     r <- 0
294     theta <- runif(1, 0, 2 * pi)
295     x1 <- xo <- x[i]
296     y1 <- yo <- y[i]
297     wid <- strwidth(words[i],
298       cex = cex[i], ...)
299     ht <- strheight(words[i],
300       cex = cex[i], ...)
301     if (grepl(tails, words[i]))
302       ht <- ht + ht * 0.2
303     if (rotWord) {
304       tmp <- ht
305       ht <- wid
306       wid <- tmp
307     }
308     isOverlaped <- TRUE
309     while (isOverlaped) {
310       if (!overlap(x1 - 0.5 *
311         wid, y1 - 0.5 * ht, wid,
312         ht, boxes) && x1 - 0.5 *
313         wid > xlim[1] && y1 -
314         0.5 * ht > ylim[1] && x1
315         + 0.5 * wid < xlim[2]
316         &&
317         y1 + 0.5 * ht < ylim[2])
318       {
319         boxes[[length(boxes) +
320           1]] <- c(x1 - 0.5 *
321           wid,
322           y1 - 0.5 * ht, wid, ht)
323         isOverlaped <- FALSE
324       }
325     else {
326       theta <- theta + tstep
327       r <- r + rstep * tstep /
328         (2 * pi)
329       x1 <- xo + sdx * r * cos
330         (theta)
331       y1 <- yo + sdy * r * sin
332         (theta)
333       iterations <- iterations
334         + 1
335       if (iterations > 500000){
336         boxes[[length(boxes) +
337           1]] <- c(x1 - 0.5 *
338             wid,
339             y1 - 0.5 * ht, wid, ht)
340         isOverlaped = FALSE
341       }
342     }
343   }
344   print( paste("iterations: ",
345     iterations) )
346   result <- do.call(rbind,
347     boxes)
348   colnames(result) <- c(\"x\",
349     \"y\", \"width\", \"ht\")
350   rownames(result) <- words
351   result
352 }
353 ", filename)
354 insertSource(filename, package =
355   "wordcloud", force=FALSE)
356

```

```

331 nc <- wordlayout(
332   labcd$x,
333   labcd$y,
334   rownames(cb),
335   cex=cex * 1.05,
336   xlim=c( par("usr")[1], par(
337     "usr")[2]),
338   ylim=c( par("usr")[3], par(
339     "usr")[4] )
340 )
341 xlen <- par("usr")[2] - par(
342   "usr")[1]
343 ylen <- par("usr")[4] - par(
344   "usr")[3]
345 segs <- NULL
346 for (i in 1:length( rownames(cb)
347   )) {
348   x <- ( nc[i,1] + .5 * nc[i,3] -
349     labcd$x[i] ) / xlen
350   y <- ( nc[i,2] + .5 * nc[i,4] -
351     labcd$y[i] ) / ylen
352   dst <- sqrt( x^2 + y^2 )
353   if ( dst > 0.05 ) {
354     segs <- rbind(
355       segs,
356       c(
357         nc[i,1] + .5 * nc[i,3], nc[i,
358           ,2] + .5 * nc[i,4],
359           xorg[i], yorg[i]
360         )
361       )
362     }
363   xorg <- labcd$x
364   yorg <- labcd$y
365   labcd$x <- nc[,1] + .5 * nc[,3]
366   labcd$y <- nc[,2] + .5 * nc[,4]
367 }
368 }
369 b_size <- NULL
370 for (i in rownames(c$cscore)){
371   if ( is.na(i) || is.null(i) || is.nan(i) )
372     {
373       b_size <- c( b_size, 1 )
374     } else {
375       b_size <- c( b_size, sum( d[,i] ) )
376     }
377 }
378 col_bg_words <- NA
379 col_bg_vars <- NA
380
381
382 if (plot_mode == "color"){
383   col_dot_words <- "#00CED1"
384   col_dot_vars <- "#FF6347"
385   if ( use_alpha == 1 ){
386     col_bg_words <- "#48D1CC"
387     col_bg_vars <- "#FFA07A"
388   }
389   rgb <- col2rgb(col_bg_words) /
390     255
391   col_bg_words <- rgb( rgb[1],
392     rgb[2], rgb[3])
393
394   rgb <- rgb * 0.5
395   col_dot_words <- "#87CAC6" #
396   <- rgb( rgb[1], rgb[2], rgb[3])
397 }
398 }
399
400 if (plot_mode == "gray"){
401   col_dot_words <- "gray55"
402   col_dot_vars <- "gray30"
403 }
404
405 if (plot_mode == "vars"){
406   col_dot_words <- "#ADD8E6"
407   col_dot_vars <- "red"
408 }
409
410 if (plot_mode == "dots"){
411   col_dot_words <- "black"
412   col_dot_vars <- "black"
413 }
414
415 g <- ggplot()
416
417 df.words <- data.frame(
418   x = c$cscore[,d_x],
419   y = c$cscore[,d_y],
420   size = b_size,
421   type = ptype
422 )
423
424 df.words.sub <- subset(df.words,
425   type==2)
426 df.words <- subset(df.words, type
427   ==1)
428
429 if (bubble_plot == 1){
430   g <- g + geom_point(
431     data=df.words,
432     aes(x=x, y=y, size=size),
433     shape=21,
434     #colour = NA,
435     fill = col_bg_words,
436     alpha=0.15
437   )

```

```

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,
496     fill = col_bg_vars,
497     alpha=0.2,
498     show.legend = F
499   )
500
501 g <- g + geom_point(
502   data=df.vars,
503   aes(x=x, y=y, size=size, shape=
504     factor(type)),
505   colour = col_dot_vars,
506   fill = NA,
507   alpha=1,
508   show.legend = F
509 )
510 } else {
511   g <- g + geom_point(
512     data=df.vars,
513     aes(x=x, y=y, shape=factor(
514       type)),
515     colour = NA,
516     fill = col_bg_vars,
517     alpha=0.2,
518     size=3.5,
519     show.legend = F
520 )
521
522 g <- g + geom_point(
523   data=df.vars,
524   aes(x=x, y=y, shape=factor(
525     type)),
526   colour = col_dot_vars,
527   fill = NA,
528   alpha=1,
529   size=3.5,
530   show.legend = F
531 )
532
533 # label colors
534 if (plot_mode == "color"){
535   #if (bubble_plot == 1){
536     col_txt_words <- "black"
537     col_txt_vars <- "#DC143C"
538   } else {
539     # col_txt_words <- "black"
540     # col_txt_vars <- "#FF6347"
541   }
542 }
543
544 if (plot_mode == "gray"){
545   col_txt_words <- "black"
546   col_txt_vars <- "black"
547 }
548
549 if (plot_mode == "vars"){
550   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   df.labels.save <- data.frame(
567     x = labcd$x,
568     y = labcd$y,
569     labs = rownames(cb),
570     cols = cb[,3]
571   )
572 }
573
574 if (plot_mode != "dots") {
575   df.labels <- data.frame(
576     x = labcd$x,
577     y = labcd$y,
578     labs = rownames(cb),
579     cols = cb[,3]
580   )
581   if (plot_mode == "gray"){
582     df.labels.var <- subset(df.labels,
583       labels, cols == 3)
584     df.labels <- subset(df.labels,
585       cols != 3)
586     g <- g + geom_label(
587       data=df.labels.var,
588       family=font_family,
589       fontface="bold",
590       #label.size=0.25,
591       label.padding=unit(1.8, "mm"),
592       colour="white",
593       fill="gray50",
594       #alpha=0.7,
595       aes(x=x, y=y, label=labs)
596     )
597     if ( (resize_vars == 0) || (bubble_
598       plot == 0) ) {
599       g <- g + geom_point(
600         data=df.vars,
601         aes(x=x, y=y, shape=factor(
602           type)),
603         colour = col_dot_vars,
604         fill = NA,
605         alpha=1,
606         size=3.5,
607         show.legend = F
608     }
609   }
610   g <- g + geom_text(
611     data=df.labels,
612     aes(x=x, y=y, label=labs, colour=
613       factor(cols)),
614     size=4,
615     family=font_family,
616     fontface=font_face
617     #colour="black"
618   )
619   #label_legend <- guide_legend(
620   #  title ="Labels:",
621   #  key.theme = element_rect(colour
622   #    = "gray30"),
623   #  override.aes = list(size=5),
624   #  order = 1
625   #)
626   label_legend <- "none"
627
628   g <- g + scale_color_manual(
629     values = c(col_txt_words, col_txt_
630       vars, col_txt_vars),
631     breaks = c(1,3),
632     labels = c("Words / Codes", "Variables"),
633     guide = label_legend
634   )
635
636   if (exists("segs")){
637     if (is.null(segs) == F){
638       colnames(segs) <- c("x1", "y1", "x2", "y2")
639       segs <- as.data.frame(segs)
640       g <- g + geom_segment(
641         aes(x=x1, y=y1, xend=x2,
642           yend=y2),
643         data=segs,
644         colour="gray60"
645       )
646     }
647   }
648   if (plot_mode == "vars"){
649     labcd <- NULL
650   }
651   #if (asp == 1){
652   #  g <- g + coord_fixed()
653   #}
654
655   g <- g + labs(
656     x=paste(name_dim, d_x, " (",
657     inertias[d_x], ", ", k[d_x], "%)", sep=""),
658     y=paste(name_dim, d_y, " (",
659     inertias[d_y], ", ", k[d_y], "%)", 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
672 if (show_origin == 1){
673   line_color <- "gray30"
674
675   lim_chk <- ggplot_build(g)
676   xlims <- lim_chk$panel$ranges[[1]]
     $x.range
677   ylims <- lim_chk$panel$ranges[[1]]
     $y.range
678   if (is.null(xlims)){
679     xlims <- lim_chk$layout$panel_
       ranges[[1]]$x.range
680     ylims <- lim_chk$layout$panel_
       ranges[[1]]$x.range
681   }
682
683   if (zoom_factor >= 1){
684     g <- g + scale_x_continuous(
       limits=xlims, expand=c(0,0),
       breaks=c(0) )
685     g <- g + scale_y_continuous(
       limits=ylims, expand=c(0,0),
       breaks=c(0) )
686   } else {
687     g <- g + scale_x_continuous(
       limits=xlims, expand=c(0,0)
     )
688     g <- g + scale_y_continuous(
       limits=ylims, expand=c(0,0)
     )
689   }
690
691   m_x <- (xlims[2] - xlims[1]) * 0.03
692   m_y <- (ylims[2] - ylims[1]) * 0.03
693
694   g <- g + geom_segment(
695     aes(x = xlims[1], y = 0, xend = m_
       _x, yend = 0),
696     size=0.25,
697     linetype="dashed",
698     colour=line_color
699   )
700   g <- g + geom_segment(
701     aes(x = 0, y = ylims[1], xend = 0,
       yend = m_y),
702     size=0.25,
703     linetype="dashed",
704     colour=line_color
705   )
706 } else {
707   if (zoom_factor >= 1){
708     g <- g + scale_x_continuous(
       breaks=c(0) )
709     g <- g + scale_y_continuous(
       breaks=c(0) )
710   }
711 }
712
713 # fix range
714 if (exists("xlimv") == F ){
715   # for setting xlim & ylim
716   out_coord <- cbind(
717     c( df.labels.save$x, df.words$x),
718     c( df.labels.save$y, df.words$y)
719   )
720
721   xlimv <- c(
722     min( out_coord[,1] ) - 0.04 * (
       max( out_coord[,1] ) - min(
         out_coord[,1] )),
723     max( out_coord[,1] ) + 0.04 * (
       max( out_coord[,1] ) - min(
         out_coord[,1] )))
724   )
725   ylimv <- c(
726     min( out_coord[,2] ) - 0.04 * (
       max( out_coord[,2] ) - min(
         out_coord[,2] )),
727     max( out_coord[,2] ) + 0.04 * (
       max( out_coord[,2] ) - min(
         out_coord[,2] )))
728   )
729
730   # for saving
731   out_coord <- cbind(
732     df.labels.save$x,
733     df.labels.save$y
734   )
735   rownames(out_coord) <- df.
     labels.save$labs
736 }
737
738 # aspect ratio
739 if (asp == 1){
740   g <- g + coord_fixed(
741     xlim=xlimv,
742     ylim=ylimv,
743     expand = F
744   )

```

```

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$labs
763
764   add <- -1 * xlimv[1]
765   div <- add + xlimv[2]
766   out_coord[,1] <- ( out_coord[,1] +
767     add ) / div
768
769   add <- -1 * ylimv[1]
770   div <- add + ylimv[2]
771   out_coord[,2] <- ( out_coord[,2] +
772     add ) / div
773
774 }
775
776 # fixing width of legends to 22%
777 library(grid)
778 library(gtable)
779 g <- ggplotGrob(g)
780
781 if ( bubble_plot == 0 ){
782   saving_file <- 1
783 }
784
785 if ( exists("saving_file") ){
786   if ( saving_file == 0){
787     target_legend_width <-
788       convertX(
789         unit( image_width * 0.22, "in"
790           ),
791         "mm"
792       )
793
794   }
795
796   if ( as.numeric( substr(
797     packageVersion("ggplot2"), 1,
798     3) ) <= 2.1 ){ # ggplot2 <=
799     2.1.0
800     diff_mm <- diff( c(
801       convertX( g$widths[5], "mm" ),
802       target_legend_width
803     ))
804     if ( diff_mm > 0 ){
805       g <- gtable_add_cols(g, unit(
806         diff_mm, "mm"))
807     }
808   } else { # ggplot2 >= 2.2.0
809     diff_mm <- diff( c(
810       convertX( g$widths[7], "mm",
811         valueOnly=T ) +
812       convertX( g$widths[8], "mm",
813         valueOnly=T ),
814       target_legend_width
815     ))
816     if ( diff_mm > 0 ){
817       print(diff_mm)
818       g <- gtable_add_cols(g, unit(
819         diff_mm, "mm"))
820     }
821   }
822
823   # fixing width of left spaces to 4.25
824   char
825   if ( as.numeric( substr(
826     packageVersion("ggplot2"), 1, 3)
827     ) <= 2.1 ){ # ggplot2 <= 2.1.0
828     diff_char <- diff( c(
829       convertX( g$widths[1] + g$widths
830         [2] + g$widths[3], "char" ),
831       unit(4.25, "char")
832     ))
833     if ( diff_char > 0 ){
834       g <- gtable_add_cols(g, unit(diff
835         _char, "char"), pos=0)
836     }
837   }
838
839   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
2   =T, nrow=190, ncol=19 )
3 d <- d[-1]

```

```

4 colnames(d) <- c("desk",⋯省略⋯,"
4   outdoor")
5 doc_length_mtr <- matrix( c(
5   70,18,⋯省略⋯,45,17), ncol=2,
5   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")) &!
38   (method_coef == "binary")){
39   leng <- as.numeric(doc_length_mtr[,2])
40   leng[leng == 0] <- 1
41   d <- t(d)
42   d <- d / leng
43   d <- d * 1000
44   d <- t(d)
45 } if (method_coef == "euclid"){ #
46   # standardize for each word
47   d <- t(scale(t(d)))
48
49 dr <- d
50 library(amap)
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
62 # standardize
63 if (exists("com_method")) {
64   if (com_method == "twomode_c" ||
65     com_method == "twomode_g"){
66     d[1:n_words,] <- 0
67     std <- d[(n_words+1):nrow(d), 1:n_words]
68     chkm <- std
69     std <- t(std)
70     std <- scale(std, center=T,
71                  scale=F)
72     std <- t(std)
73   if (min(std[!is.na(std)]) < 0.0005
74     ){
75     std <- std + (0.0005 - min(
76       std[!is.na(std)]));
77   }
78   std <- std / max(std[!is.na(std)])
79   std[chkm == 0] <- 0
80   d[(n_words+1):nrow(d), 1:n_words]
81   <- std
82 }
83
84 # Make a graph
85 # For igraph > 1.0.0
86 library(igraph)
87 new_igraph <- 0
88 igraph_ver <- (
89   as.numeric(substr(sessionInfo()
90     $otherPkgs$igraph$Version,
91     1,1)) * 10
92   +
93   as.numeric(substr(sessionInfo()
94     $otherPkgs$igraph$Version,
95     3,3))
96 )
97 if (igraph_ver > 5){
98   new_igraph <- 1
99 }
100 n <- graph.adjacency(d, mode =
101   "lower", weighted=T, diag=F)
102 n <- igraph::set.vertex.attribute(
103   n,
104   "name",
105   (0+new_igraph):(length(d[1,])-1+
106   new_igraph),
107   as.character(1:length(d[1,])))
108
109 # Prepare for deleting weak edges
110 el <- data.frame(
111   edge1 = get.edgelist(n, name=T)[,1],
112   edge2 = get.edgelist(n, name=T)[,2],
113   weight = igraph::get.edge.attribute(
114     n, "weight"),
115   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$weight,
118     names = F,
119     probs = 1 - edges / length(el[,1])
120   )
121 }
122
123 # Delete weak edges and make a graph
124 again
124 el2 <- subset(el, el[,3] >= th)
125 if ( nrow(el2) == 0 ){
126   stop(message = "No edges to
127     draw!", call. = F)
128 }
129 n2 <- graph.edgelist(
130   matrix( as.matrix(el2)[,1:2], ncol
131     =2 ),
132   directed =F
133 )
134 n2 <- igraph::set.edge.attribute(
135   n2,
136   "weight",
137   (0+new_igraph):(length(get.
138     edgelist(n2)[,1])-1+new_igraph)
139   el2[,3]
140 )
141
142 if ( min_sp_tree_only == 1 ){
143   n2 <- minimum.spanning.tree(
144     n2,
145     weights = 1 - igraph::get.edge.
146       attribute(n2, "weight"),
147       algorithm="prim"
148   )
149
150 if (length(igraph::get.vertex.attribute
151   (n2,"name")) < 2){
152   com_method <- "none"
153 }
154
155 # Centrality
156 if ( com_method == "cnt-b" || com_
157   method == "cnt-d" || com_
158   method == "cnt-e"){
159   ccol <- NULL
160   if (com_method == "cnt-b"){ #
161     betweenness
162     ccol <- igraph::betweenness(
163       n2,
164       v=(0+new_igraph):(length(
165         igraph::get.vertex.attribute(
166           n2,"name")))-1+new_igraph
167         ),
168       directed=F
169     )
170   if (com_method == "cnt-d"){ #
171     degree
172     ccol <- igraph::degree(
173       n2,
174       v=(0+new_igraph):(length(
175         igraph::get.vertex.attribute(
176           n2,"name")))-1+new_igraph
177         )
178   }
179   if (com_method == "cnt-e"){ #
180     evcent
181     try(
182       ccol <- igraph::evcent(n2)$
183         vector,
184       silent = T
185     )
186   }
187   ccol_raw <- ccol
188   edg_col <- "gray55"
189   edg_lty <- 1
190 }
191
192 # Community detection
193 if (com_method == "com-b" || com_
194   method == "com-g" || com_
195   method == "com-r"){
196   merge_step <- function(n2, m){
197     for ( i in 1:( trunc( length( m )
198       / 2 ) ) ){
199       temp_csize <- community.to.
200         membership(n2, m,i)$csize
201       num_max <- max( temp_csize
202         )
203       num_alone <- sum( temp_csize
204         [ temp_csize == 1 ] )
205       num_cls <- length( temp_csize
206         [temp_csize > 1] )
207       #print( paste(i, "a", num_alone,
208         "max", num_max, "cls",
209         num_cls) )
210     if (
211       num_max / length(igraph
212         ::get.vertex.attribute(
213           n2,"name")) >= 0.225
214     && num_max > num_alone
215     && num_cls < 12
216     ){
217       return(i)
218     }
219   }
220
221   if (num_max / length(igraph::

```

```

          get.vertex.attribute(n2,"  

          name")) >= 0.4 ){  

201      }  

202      }  

203      }  

204      return( trunc(length( m ) / 2) )  

205  } # For igraph > 1.0.0  

206  if (com_method == "com-b"){ #  

207      Betweenness  

208      com <- edge.betweenness.  

209      community(n2, directed=F)  

210      if (igraph_ver < 10){  

211          com_m <- community.to.  

212          membership(  

213              n2, com$merges, merge_step(  

214                  n2,com$merges)  

215          )  

216          com_m$membership <- com_m  

217          $membership + new_igraph  

218      }  

219      if (com_method == "com-g"){ #  

220          Modularity  

221          com <- fastgreedy.community (n2,  

222              merges=TRUE,  

223              modularity=TRUE)  

224          if (igraph_ver < 10){  

225              com_m <- community.to.  

226              membership(  

227                  n2, com$merges, merge_step(  

228                      n2,com$merges)  

229          )  

230          com_m$membership <- com_m  

231          $membership + new_igraph  

232      }  

233      if (com_method == "com-r"){ #  

234          Random walks  

235          com <- walktrap.community(n2,  

236              weights=igraph::get.edge.  

237              attribute(n2, "weight")  

238          )  

239          if (igraph_ver < 10){  

240              com_m <- NULL  

241              com_m$membership <- com$  

242              membership  

243              com_m$csize <- table(com$  

244              membership)  

245          }  

246          if (igraph_ver >= 10){  

247              com_m <- NULL  

248              com_m$membership <- as.  

249              vector( membership(com) )  

250              com_m$csize <- table(com_m$  

251              membership)  

252          }  

253      } # Configure Edges  

254      edg_lty <- NULL  

255      edg_col <- NULL  

256      for (i in 1:nrow(get.edgelist(n2,  

257          name=F))){  

258          if (  

259              com_m$membership[ get.  

260                  edgelist(n2,name=F)[i,1]  

261                  + 1 - new_igraph]  

262                  == com_m$membership[ get.  

263                  edgelist(n2,name=F)[i,2] +  

264                  1 - new_igraph]  

265          ){  

266              edg_col <- c( edg_col, "gray55"  

267                  )  

268              edg_lty <- c( edg_lty, 1 )  

269          } else {  

270              edg_col <- c( edg_col, "gray"  

271                  )  

272              edg_lty <- c( edg_lty, 3 )  

273          }  

274      }  

275      if (com_method == "twomode_c" ||  

276          com_method == "twomode_g"){  

277          if ( exists("var_select" ) ){  

278              var_select_bak <- var_select  

279          }  

280          var_select <- substring(  

281              colnames(d)[ as.numeric( igraph  

282                  ::get.vertex.attribute(n2,"name"  

283                  " ) ) ],  

284                  1,  

285                  2  

286          ) == "<>"  

287          if (length(var_select[var_select==  

288              TRUE]) == 0 && exists("var_  

289              select_bak")){  

290              var_select <- var_select_bak;  

291          }  

292          if (com_method == "twomode_c"){  

293              ccol <- igraph::degree(  

294                  n2,  

295                  v=(0+new_igraph):(length(  

296                      igraph::get.vertex.attribute(n2  

297                      , "name"))-1+new_igraph)  

298              )  

299              # ggplot2  

300              ccol_raw <- ccol  

301              ccol_raw[var_select] <- NA  

302              ccol_raw[ccol_raw >= 5] <- 5  

303              ccol_raw <- as.character(ccol_raw  

304                  )  

305              ccol_raw[ccol_raw=="5"] <- "5+"  

306              edg_col <- "gray70"

```

```

289   edg_lty <- 1
290 }
291 }
292 }
293
294 if (com_method == "none" || com_
295     method == "twomode_g"){
296   edg_lty <- 1
297   edg_col <- "gray40"
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.
309         attribute(n2, "weight"),
310         algorithm="prim"
311   )
312
313 #if (length(edg_col) == 1){
314   edg_col <- rep("gray55",
315     length( igraph::get.edge.
316         attribute(n2, "weight") ))
317 #}
318
319 n2_edges <- get.edgelist(n2,name=
320   T);
321 mst_edges <- get.edgelist(mst,
322   name=T);
323
324 if (exists("edg_lty") == F){
325   edg_lty <- 1
326 }
327 for ( i in 1:ecount(n2) ){
328   name_n2 <- paste(
329     n2_edges[i,1],
330     n2_edges[i,2]
331   )
332   for ( j in 1:ecount(mst) ){
333     name_mst <- paste(
334       mst_edges[j,1],
335       mst_edges[j,2]
336     )
337     if ( name_n2 == name_mst ){
338       edg_col[i] <- "gray30" #
339         edge color
340         if ( length(edg_lty) > 1 ){
341           edg_lty[i] <- 1 # edge
342             linetype
343           }
344           break
345         }
346       }
347     }
348   }
349   if ( length(igraph::get.vertex.
350     attribute(n2,"name")) >= 3 ){
351     d4l <- as.dist( shortest.paths(n2) )
352     if ( min(d4l) < 1 ){
353       d4l <- as.dist( shortest.paths(n2,
354         weights=NA ) )
355     }
356     if ( max(d4l) == Inf ){
357       d4l[d4l == Inf] <- vcount(n2)
358     }
359     try( lay <- cmdscale( d4l, k=2 ),
360         silent=TRUE )
361     if ( is.null(lay) == F ){
362       lay <- round(lay, digits=5)
363       check4fr <- function(d){
364         chk <- 0
365         for ( i in combn( length(d[,1]),
366           2, simplify=F ) ){
367           if (
368             d[i[1],1] == d[i[2],1]
369             && d[i[1],2] == d[i[2],2]
370           ){
371             return( i[1] )
372           }
373         }
374       }
375       return( NA )
376     }
377     while ( is.na(check4fr(lay)) == 0
378       ){
379       mv <- check4fr(lay)
380       lay[mv,1] <- lay[mv,1] + 0.001
381       #print( paste( "Moved:", mv ) )
382     }
383   }
384   set.seed(100)
385   if (
386     (com_method == "twomode_c" ||
387      com_method == "twomode_-
388        g")
389     && ( igraph::is.connected(n2) )
390   ){
391     # For igraph > 1.0.0
392     if (igraph_ver >= 10){
393       lay_f <- layout.kamada.kawai(
394         n2,
395         #coords = lay,
396         weights = igraph::get.edge.
397           attribute(n2, "weight")
398       )
399     } else {
400       lay_f <- layout.kamada.kawai(
401         n2,
402         start = lay,
403       )
404     }
405   }

```

```

395     weights = igraph::get.edge.
396         attribute(n2, "weight")
397     }
398 } else {
399 # For igraph > 1.0.0
400 if (igraph_ver >= 10){
401     lay_f <- layout.fruchterman.
402         reingold(
403             n2,
404             #coords = lay,
405             niter = vcount(n2) * 512,
406             weights = igraph::get.edge.
407                 attribute(n2, "weight")
408         )
409     } else {
410         lay_f <- layout.fruchterman.
411             reingold(
412                 n2,
413                 start = lay,
414                 niter = vcount(n2) * 512,
415                 weights = igraph::get.edge.
416                     attribute(n2, "weight")
417         )
418     }
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" ||
427     com_method == "twomode_g"){
428     colnames(d)[
429         substring(colnames(d), 1, 2) ==
430             "<>"]
431     ] <- substring(
432         colnames(d)[
433             substring(colnames(d), 1, 2)
434                 == "<>"]
435     ],
436     nchar(colnames(d)[
437         substring(colnames(d), 1, 2)
438             == "<>"]
439     ]),
440
441
442 if ( exists("saving_emf") || exists("-
443     saving_eps" ) ){
444     use_alpha <- 0

```

```

445 target_ids <- NULL
446 if ( exists("target_words" ) ){
447 # get word IDs
448 for (i in 1:length( igraph::get.
449     vertex.attribute(n2,"name") ) ){
450     for (w in target_words){
451         if (
452             colnames(d)[ as.numeric(
453                 igraph::get.vertex.
454                 attribute(n2,"name")[i]] ]
455                 == w
456         ){
457             target_ids <- c(target_ids, i)
458         }
459     }
460     edge_label <- NULL
461
462 font_fam <- "Meiryo UI"
463 if ( exists("PERL_font_family" ) ){
464     font_fam <- PERL_font_family
465 }
466
467 if ( length(igraph::get.vertex.
468     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         labcd <- NULL
475         labcd$x <- lay_f[,1]
476         labcd$y <- lay_f[,2]
477         word_labs <- colnames(d)[ as.
478             numeric( igraph::get.vertex
479                 .attribute(n2,"name") ) ]
480
481 library(wordcloud)
482
483 # fix for "wordlayout" function
484 filename <- tempfile()
485 writeLines("wordlayout <-
486     function (x, y, words, cex =
487         1, rotate90 = FALSE, xlim = c
488             (-Inf,
489             Inf), ylim = c(-Inf, Inf),
490             tstep = 0.1, rstep = 0.1,
491             ...)
492 {
493     tails <- \g|j|p|\q|y\"
494     n <- length(words)
495     sdx <- sd(x, na.rm = TRUE)
496     sdy <- sd(y, na.rm = TRUE)
497     iterations <- 0
498     if (sdx == 0){
499         sdx <- 1
500     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("iterations: \",
540     iterations) )
541   result <- do.call(rbind,
542     boxes)
543   colnames(result) <- c("x\",
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$x, labcd$y)
564   if_fixed <- 1
565   }
566   if (com_method == "cor"){
567     dr <- as.data.frame( t(dr) )
568     dv <- data.frame(
569       khvar_ = as.numeric(v0)
570     )
571     dr <- cbind(dr,dv)
572     edge_pos <- NULL
573     edges <- get.edgelist(n2, names=
574       TRUE)
575     for (i in 1:nrow(edges)){
576       i1 <- as.numeric( edges[i,1] )
577       i2 <- as.numeric( edges[i,2] )
578       edge_pos <- c(
579         edge_pos,
580         cor(
581           as.numeric( dr[,i1] ) > 0 & dr[,i2] > 0 ),
582             dr$khvar_,
583               method="pearson"
584             )
585             #mean( dr[dr[,i1] > 0 & dr[,i2]
586               > 0,]$khvar_ )
587             )
588             }
589             if ( length( edge_pos[is.na(edge_
590               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 #edge_pos <- edge_pos - mean(
602 #     edge_pos)
603 #edge_pos <- edge_pos / sd(edge_
604 #     pos)
605 ##edge_pos <- edge_pos * 10 + 50
606
607 #limv <- 0.15
608 #maxv <- max( abs( edge_pos ) )
609
610 #if ( limv < maxv ){
611 # edge_pos[edge_pos > limv] <-
612 #     limv
613 # edge_pos[edge_pos < -limv] <-
614 #     -limv
615 #
616 n2 <- igraph::set.edge.attribute(
617     n2,
618     "edge_pos",
619     1:length(igraph::get.edge.
620             attribute(n2, "weight")),
621     edge_pos
622 )
623
624 ver_pos <- NULL
625 vertices <- as.numeric( igraph::
626         get.vertex.attribute(n2, "name")
627     )
628 for (i in 1:length(vertices)){
629     ver_pos <- c(
630         ver_pos,
631         cor(
632             as.numeric( dr[,vertices[i]] >
633                 0 ),
634             dr$khvar_,
635             method="pearson"
636         )
637     )
638
639 if ( length( ver_pos[is.na(ver_pos)
640             == F] ) == 0 ){
641     ver_pos <- 0
642 }
643
644 ver_pos[ver_pos > max(edge_pos)] <-
645     max(edge_pos)
646 ver_pos[ver_pos < min(edge_pos)] <-
647     min(edge_pos)
648
649 #n2 <- igraph::set.vertex.attribute(
650 # n2,
651 # "ver_pos",
652 # 1:length(igraph::get.vertex.
653 #         attribute(n2, "name")),
654 # ver_pos
655 #
656 ccol_raw <- ver_pos
657
658 if ( is.null( igraph::get.vertex.
659             attribute(n2, "com") ) ==
660             FALSE ){
661     n2 <- remove.vertex.attribute(
662         n2, "com")
663     edg_lty <- 1
664 }
665
666 n2 <- igraph::set.vertex.attribute(
667     n2,
668     "lab",
669     1:length(igraph::get.vertex.
670             attribute(n2, "name")),
671     colnames(d)[ as.numeric( igraph::
672                 get.vertex.attribute(n2, "name")
673             ) ]
674 )
675
676 ver_freq <- freq[ as.numeric( igraph
677             ::get.vertex.attribute(n2, "name") )
678         ]
679
680 if ( com_method == "twomode_c" ||
681         com_method == "twomode_g" ){
682     ver_freq[var_select] <- NA
683 }
684
685 if ( is.null(target_ids) == FALSE ){
686     ver_freq[target_ids] <- NA
687 }
688
689 if ( use_freq_as_size == 0 ){
690     ver_freq[ver_freq > 0] <- 1
691 }
692
693 n2 <- igraph::set.vertex.attribute(
694     n2,
695     "size",
696     1:length(igraph::get.vertex.
697             attribute(n2, "name")),
698     ver_freq
699 )
700
701 # For community detection
702
703 if ( exists("ccol" ) ){ # clean up
704     previous data
705     try( n2 <- remove.vertex.
706             attribute(n2, "com"), silent=T )
707 }
708
709 if ( exists("com_m" ) ){
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2
```

```

688 com_label <- NULL
689
690 for (h in 1:length(com_m$membership)){
691   i <- com_m$membership[h]
692   if (com_m$csize[i] > 1) {
693     if (i < 10){
694       com_label <- c(
695         com_label,
696         paste("0", as.character(i),
697           " ", sep=""))
698     } else {
699       com_label <- c(com_label, as.character(i))
700     }
701   } else {
702     com_label <- c(com_label, NA)
703   }
704 }
705
706 n2 <- igraph::set.vertex.attribute(
707   n2,
708   "com",
709   1:length(igraph::get.vertex.
710             attribute(n2,"name")),
711   com_label
712 )
713
714 if (exists("ccol_raw")){
715   n2 <- igraph::set.vertex.attribute(
716   n2,
717   "com",
718   1:length(igraph::get.vertex.
719             attribute(n2,"name")),
720   ccol_raw
721 )
722 com_label <- ccol_raw
723 }
724 if (com_method == "none" || com_
725   method == "twomode_g"){
726   n2 <- igraph::set.vertex.attribute(
727   n2,
728   "com",
729   1:length(igraph::get.vertex.
730             attribute(n2,"name")),
731   rep("na", length(igraph::get.
732             vertex.attribute(n2,"name")))
733 )
734 com_label <- NA
735 }
736
737 if (exists("edg_mst")){
738   n2 <- igraph::set.edge.attribute(
739   n2,
740   "edg_col",
741   1:length(igraph::get.edge.
742             attribute(n2,"weight")),
743   edg_mst
744   )
745   #print(edg_mst)
746 }
747
748 if (exists("edg_lty") == F ){
749   edg_lty <- 1
750 }
751 edg_lty[edg_lty==1] <- "solid"
752 edg_lty[edg_lty==3] <- "dotted"
753
754 n2 <- igraph::set.edge.attribute(
755   n2,
756   "line",
757   1:length(igraph::get.edge.attribute(
758   n2,"weight")),
759   edg_lty
760 )
761
762 library(ggplot2)
763 library(ggnetwork)
764
765 p <- ggplot(
766   ggnetwork(n2, layout=lay_f),
767   aes(x = x, y = y, xend = xend, yend
768     = yend),
769 )
770
771 if (use_alpha == 1){
772   alpha_value = 0.62
773   gray_color_n <- "gray20"
774 } else {
775   alpha_value = 1
776   gray_color_n <- "gray40"
777 }
778
779 if (text_font == 2){
780   face <- "bold"
781 } else {
782   face <- "plain"
783 }
784 if (smaller_nodes == 1 ){
785   edge_colour <- "gray68"
786   nudge <- 0.015
787   hjust <- "left"
788 } else {
789   edge_colour <- "gray55"
790   nudge <- 0
791   hjust <- "center"
792 }
793
794 if (com_method == "twomode_c"){
795   edge_colour <- "gray70"
796 }
797
798 if (com_method == "none" || com_
799   method == "twomode_g"){
800   edge_colour <- "gray40"
801   gray_color_n <- "black"
802 }

```

```

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  

851     p <- p + scale_color_gradientn(  

852         colours = myPalette,  

853         #limits = c( min(edge_pos), limv )  

854         ,  

855         #limits = c( 0 - limv, 0 + limv ),  

856         guide = guide_colourbar(  

857             title = "Correlation:\n",  

858             title.theme = element_text(  

859                 family=sans,  

860                 face="bold",  

861                 size=11,  

862                 lineheight=0.4,  

863                 angle=0  

864             ),  

865             order = 1,  

866             #override.aes = list(size=6,  

867                 shape=22),  

868             label.hjust = 1,  

869             #reverse = TRUE,  

870             #ncol=2,  

871             #keyheight = unit(1.5,"line")  

872         )  

873         )  

874         p <- p + scale_fill_gradientn(  

875             colours = myPalette,  

876             guide = FALSE  

877         )  

878     } else if (min_sp_tree == 1){  

879         edg_col2 <- p$data$edg_col  

880         edg_col2[edg_col2=="gray30"] <-  

881             "MST"  

882         edg_col2[edg_col2=="gray55"] <-  

883             "non-MST"  

884         edg_col2[edg_col2=="gray70"] <-  

885             "non-MST"  

886         p <- p + geom_edges(  

887             aes(linetype = as.character(line),  

888                 alpha=edg_col2),  

889                 #size = 0.8,  

890                 color = "grey10"  

891         )  

892         p <- p + scale_alpha_discrete(  

893             range = c(1, 0.3),  

894             guide = guide_legend(  

895                 title = "Edge:",  

896                 keyheight = unit(1.2,"line"),  

897                 order = 2  

898             )  

899         )  

900     } else if ( use_weight_as_width == 1 )  

901     {  

902         p <- p + geom_edges(  

903             aes(linetype = as.character(line),  

904                 alpha=weight),  

905                 #size = 0.8,  

906                 color = "grey10"  

907         )  

908         p <- p + scale_alpha(  

909             range = c(0.2, 1),  

910             guide = guide_legend(  

911                 title = "Coefficient:",  

912                 label.hjust = 1,  

913                 keyheight = unit(1.2,"line"),  

914                 order = 2  

915             )  

916         )  

917     } 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$csize[com.m$  
  csizes >= 2]) >= 13 )
922   && ( length(com.m$csize[com.m$  
  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  
      / 3,
961     shape = 16
962 )
963 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  
  the legend
974   aes( fill = com ),
975   size=0,
976   colour = gray_color_n,
977   alpha = 0,
978   shape = 21
979 )
980
981 if ( use.freq.as.size == 1 ){
982   p <- p + scale_size_area(  
    "Frequency",
983    max.size = 30 * bubble.size /  
      100,
984    guide = guide_legend(  
      title = "Frequency:",
985      override.aes = list(colour = "  
        black", alpha=1, shape=1),
986      label.hjust = 1,
987      order = 3
988    )
989  )
990 }
991 }
992 } else {
993   p <- p + scale_size_area(  
    max.size = vv,
994    guide = F
995  )
996 }
997 }
998
999 if ( (com.method == "twomode_c" ||  
  com.method == "twomode_g") ) {  
  # (is.null(target_ids) == FALSE)
1000
1001   if ( com.method == "twomode_c" ){  
    var.outline.c <- "gray50"  
    var.fill.c <- "#FB8072"
1002  }
1003   if ( com.method == "twomode_g" ){  
    var.outline.c <- "black"  
    var.fill.c <- "white"
1004  }
1005
1006   p <- p + geom_point(  
    data = data.frame(  
      x = lay.f[var.select,1],
1007      y = lay.f[var.select,2]
1008    ),
1009    aes(
1010      x = x,
1011      y = y,
1012      xend = x,
1013      yend = y
1014    )
1015  )
1016
1017 
```

```

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,
1082       yend = y,
1083       fill = com_label[var_select]
1084     ),
1085     show.legend = F,
1086     colour = gray_color_n,
1087     alpha = alpha_value,
1088     size = vv,
1089     shape = 22
1090   )
1091
1092   p <- p + geom_point(
1093     data = data.frame(
1094       x = lay_f[var_select,1],
1095       y = lay_f[var_select,2]
1096     ),
1097     aes(
1098       x = x,
1099       y = y,
1100       xend = x,
1101       yend = y
1102     ),
1103     fill = NA,
1104     show.legend = F,
1105     colour = gray_color_n,
1106     alpha = alpha_value,
1107     size = vv * 1.4,
1108     shape = 22
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 } else {
1138   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$csizes[com_m$csizes > 1]) <= 12 ){
1186       p <- p + scale_color_brewer(
1187         palette = "Set3",
1188         na.value = "white",
1189         guide = FALSE
1190       )
1191       p <- p + scale_fill_brewer(
1192         palette = "Set3",
1193         na.value = "white",
1194         guide = guide_legend(
1195           title = "Community:",
1196           override.aes = list(size=5.5,
1197             alpha=1, shape=22),
1198             keyheight = unit(1.25,"line"),
1199             ncol=2,
1200             order = 1
1201           )
1202         )
1203       } else if (length(com_m$csizes[com_m$csizes > 1]) <= 20) {
1204         library(ggsci)
1205         p <- p + scale_color_d3(
1206           palette = "category20",
1207           na.value = "white",
1208           guide = FALSE
1209         )
1210         p <- p + scale_fill_d3(
1211           palette = "category20",
1212           na.value = "white",
1213           guide = guide_legend(
1214             title = "Community:",
1215             override.aes = list(size=5.5,
1216               alpha=1, shape=22),
1217               keyheight = unit(1.25,"line"),
1218               ncol=2,
1219               order = 1
1220             )
1221         )
1222       } else {
1223         p <- p + scale_color_hue(
1224           c = 50,
1225           l = 85,
1226           na.value = "white",
1227           guide = FALSE
1228         )
1229         p <- p + scale_fill_hue(
1230           c = 50,
1231           l = 85,
1232           na.value = "white",
1233           guide = guide_legend(
1234             title = "Community:",
1235             override.aes = list(size=5.5,
1236               alpha=1, shape=22,
1237               colour="gray45"),
1238               keyheight = unit(1.25,"line"),
1239               ncol=2,
1240               order = 1
1241             )
1242           )
1243         } else if ( com_method == "cnt-b" || com_
1244           method == "cnt-d" || com_
1245           method == "cnt-e"){
1246           if (gray_scale == 1){
1247             myPalette <- gray( seq(1, 0.4,
1248               length.out=100) )

```

```

1244 } else {
1245   if (color_universal_design == 0){
1246     myPalette <- cm.colors(99)
1247   } else {
1248     library(RColorBrewer)
1249     col_seed <- brewer.pal(8,"YlGnBu")[1:6]
1250
1251     myPalette <-
1252       colorRampPalette( col_seed
1253         )
1254     myPalette <- myPalette(99)
1255   }
1256
1257   p <- p + scale_color_gradientn(
1258     colours = myPalette,
1259     guide = FALSE
1260   )
1261   p <- p + scale_fill_gradientn(
1262     colours = myPalette,
1263     guide = guide.colourbar(
1264       title = "Centrality:\n",
1265       title.theme = element_text(
1266         family=sans,
1267         face="bold",
1268         size=11,
1269         lineheight=0.4,
1270         angle=0
1271       ),
1272       order = 1,
1273       #override.aes = list(size=6,
1274         shape=22),
1275       label.hjust = 1,
1276       #reverse = TRUE,
1277       #ncol=2,
1278       #keyheight = unit(1.5,"line")
1279     )
1280   }
1281
1282   if (com_method == "twomode_c"){
1283     p <- p + scale_color_manual(
1284       values = brewer.pal(8, "Spectral"
1285         )[4:8],
1286       guide = FALSE
1287     )
1288     p <- p + scale_fill_manual(
1289       values = brewer.pal(8, "Spectral"
1290         )[4:8],
1291       guide = guide.legend(
1292         title = "Degree:",
1293         order = 1,
1294         override.aes = list(size=5.5,
1295           shape=22, alpha=1),
1296         #label.hjust = "left",
1297         #reverse = TRUE,
1298         #ncol=2,
1299         keyheight = unit(1.2,"line")
1300       )
1301   }
1302
1303   if ( com_method == "none" || com_
1304     method == "twomode_g"){
1305     p <- p + scale_color_manual(
1306       values = c("white"),
1307       na.value = "white",
1308       guide = F
1309     )
1310     p <- p + scale_fill_manual(
1311       values = c("white"),
1312       na.value = "white",
1313       guide = F
1314     )
1315   }
1316
1317   p <- p + theme_blank(
1318     base_family = font_fam
1319   )
1320
1321   if (com_method == "cor" && gray_
1322     scale == 0){ # cor
1323     if ( cor_var_darker == 1 ){
1324       col_backg <- "gray50"
1325     } else {
1326       col_backg <- "gray60"
1327     }
1328     p <- p + theme(
1329       panel.background = element_rect(
1330         fill = col_backg, colour = NA
1331       )
1332     )
1333
1334   p <- p + theme(
1335     legend.title = element_text(family
1336       =sans, face="bold", size=11,
1337       angle=0),
1338     legend.text = element_text(face="
1339       plain", size=11, angle=0)
1340   )
1341
1342   # make a small space between the
1343   # graph and the legend
1344   margin <- 0.04
1345   #if (smaller_nodes == 1){
1346   # extra <- 0.05
1347   # p <- p + coord_fixed(ratio=1, xlim
1348   # =c(0-margin-extra,1+margin+
1349   # extra), ylim=c(0-margin,1+
1350   # margin), expand = F )
1351   #} else {
1352   extra <- 0.025
1353   p <- p + coord_fixed(ratio=1, xlim
1354   =c(0-margin-extra,1+
1355   margin+extra), ylim=c(0-
1356   margin,1+margin), expand =
1357   F )
1358   #
1359

```

```

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   if ( as.numeric( substr(
1382     packageVersion("ggplot2"), 1,
1383     3) ) <= 2.1 ){ # ggplot2 <=
1384     2.1.0
1385     diff_mm <- diff( c(
1386       convertX( g$widths[5], "mm" ),
1387       target_legend_width
1388     ))
1389     if ( diff_mm > 0 ){
1390       print(diff_mm)
1391       g <- gtable_add_cols(g, unit(
1392         diff_mm, "mm"))
1393     }
1394   } else { # ggplot2 >= 2.2.0
1395     diff_mm <- diff( c(
1396       convertX( g$widths[7], "mm",
1397         valueOnly=T ) +
1398       convertX( g$widths[8], "mm",
1399         valueOnly=T ),
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   }
1408 }
1409 grid.draw(g)
1410
1411 if (exists("com_m")){
1412   rm("com_m")
1413 }
1414 if (exists("ccol_raw")){
1415   rm("ccol_raw")
1416 }
1417 if (exists("edg_mst")){
1418   rm("edg_mst")
1419 }
1420 if (exists("edg_lty")){
1421   rm("edg_lty")
1422 }
1423 ccol <- igraph::get.vertex.attribute(
1424   n2, "com")
1425 }
```

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

```

1 d <- NULL
2 d <- matrix( c(1,⋯省略⋯,0), byrow

```

```

=TRUE, nrow=380, ncol=41 )
3 d <- d[, -1]
4 colnames(d) <- c("desk",⋯省略⋯,"
```

```

5 doc_length_mtr <- matrix( c(
6   70,18,...省略...,57,19), ncol=2,
7   byrow=T)
8 colnames(doc_length_mtr) <- c("length_c", "length_w")
9 d <- t(d)
10 # END: DATA
11
12 n_nodes <- 20
13 rlen1 <- 1000
14 rlen2 <- 200000
15 d <- t(d)
16
17 if (exists("doc_length_mtr")){
18   leng <- as.numeric(doc_length_
19     mtr[,2])
20   leng[leng ==0] <- 1
21   d <- d / leng
22   d <- d * 1000
23 }
24
25 d <- subset(d, rowSums(d) > 0)
26 d <- scale(d)
27 d <- t(d)
28 rownames(d) <- 1:nrow(d)
29 # SOM
30 library(som)
31 somm <- som(
32   d,
33   n_nodes,
34   n_nodes,
35   topol="hexa",
36   rlen=c(rlen1,rlen2)
37 )
38
39 word_labs <- rownames(d)
40 n_words <- length(word_labs)
41
42 color_universal_design <- 1
43 # END: DATA
44
45 cex <- 1
46 text_font <- 2
47 if_cls <- 1
48 n_cls <- 9
49 if_plothex <- 1
50
51 # n_nodes <- 20
52 # rlen1 <- 1000
53 # rlen2 <- 200000
54
55 row2coods <- NULL
56 eve <- 0
57 for (i in 0:(n_nodes - 1)){
58   for (h in 0:(n_nodes - 1)){
59     row2coods <- c(row2coods, h +
60       eve, i)
61   }
62   if (eve == 0){
63     eve <- 0.5
64   } else {
65     eve <- 0
66   }
67   row2coods <- matrix( row2coods,
68     byrow=T, ncol=2 )
69
70 if ( if_cls == 1 ){
71   library( RColorBrewer )
72   if (
73     ( as.numeric( R.Version()$major ) >= 3 )
74     && ( as.numeric( R.Version()$minor ) >= 1.0 )
75   ){ # >= R 3.1.0
76     hcl <- hclust( dist(somm$code,
77       method="euclidean"),
78       method="ward.D2" )
79   } else { # <= R 3.0
80     hcl <- hclust( dist(somm$code,
81       method="euclidean")^2,
82       method="ward" )
83   }
84
85 colors <- NULL
86 if (n_cls <= 9){
87   pastel <- brewer.pal(9, "Pastel1"
88     "[6]")
89   # pastel[6] = brewer.pal(9, "Pastel1")[9]
90   # pastel[9] = brewer.pal(9, "Pastel1")[6]
91   pastel[6] = "gray91"
92   pastel[9] = "#F5F5DC" # FAF3C8
93   F7F1C6 EEE8AA F0E68C
94   colors <- pastel[cutree(hcl,k=n_
95     cls)]}
96 } else {
97   colors <- rep("gray90", n_nodes
98     ^2)
99 }
100 labcd <- NULL
101 plot_mode <- "color"
102

```

```

104 par(mai=c(0,0,0,0), mar=c(0,0,0,0),
105      omi=c(0,0,0,0), oma =c(0,0,0,0) )
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_plothex == 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 } if ( plot_mode == "color" ) {
135   color_act <- colors
136   color_line <- "white"
137   if_points <- 1
138   w_lwd <- 1
139   if (n_cls > 9) {
140     color_cls <- "gray45"
141   } else {
142     color_cls <- "gray60"
143   }
144   color_ptf <- "white"
145 }
146 if ( plot_mode == "freq" ){
147   color_act <- somm$code.sum$nobs
148   ;
149   if (max(color_act) == 1){
150     color_act <- color_act * 3 + 1;
151   } else {
152     color_act <- color_act - min(
153       color_act)
154     color_act <- round( color_act /
155       max(color_act) * 6 ) + 1
156     #color_act[color_act==7] <- 6
157   }
158   color_seed <- brewer.pal(6,"GnBu")
159   #color_seed <- brewer.pal(6,"
160   YlOrRd")
161   color_seed <- c("white", color_seed
162     )
163
164   color_act <- color_seed[color_act]
165   color_line <- "gray70"
166   if_points <- 0
167   w_lwd <- 1
168   color_cls <- "gray45"
169   color_ptf <- "white"
170 }
171 if ( plot_mode == "umat" ){
172
173   dist_u <- NULL
174   dist_m <- as.matrix( dist(somm$code, method="euclid") )
175
176   for (i in 0:(n_nodes - 1)){
177     for (h in 0:(n_nodes - 1)){
178       cu <- NULL
179       n <- 0
180
181       if (h != n_nodes -1){
182         cu <- c(
183           cu,
184           dist_m[
185             h + i * n_nodes + 1,
186             h + 1 + i * n_nodes + 1
187           ]
188         )
189
190       if (h != 0){
191         cu <- c(
192           cu,
193           dist_m[
194             h + i * n_nodes + 1,
195             h - 1 + i * n_nodes + 1
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 }
```

```

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   } else {
230     if (h != n_nodes -1){
231       cu <- c(
232         cu,
233         dist_m[
234           h + i * n_nodes + 1,
235           h + 1 + ( i - 1 ) * n_
236             nodes + 1
237         ]
238     }
239   }
240 }
241
242 if (i != n_nodes - 1){
243   if (h %% 2 == 0){
244     if (h != 0){
245       cu <- c(
246         cu,
247         dist_m[
248           h + i * n_nodes + 1,
249             h - 1 + ( i + 1 ) * n_
250               nodes + 1
251         ]
252     }
253   } else {
254     cu <- c(
255       cu,
256       dist_m[
257         h + i * n_nodes + 1,
258           h + ( i + 1 ) * n_nodes
259             + 1
260         ]
261     }
262   }
263
264 if (i != 0){
265   if (h %% 2 == 0){
266     if (h != 0){
267       cu <- c(
268         cu,
269         dist_m[
270           h + i * n_nodes + 1,
271             h - 1 + ( i - 1 ) * n_
272               nodes + 1
273     ]
274   } else {
275     cu <- c(
276       cu,
277       dist_m[
278         h + i * n_nodes + 1,
279         h + ( i - 1 ) * n_nodes
280           + 1
281       ]
282   }
283 }
284 dist_u <- c(dist_u, median(cu)
285 )
286 }
287
288 print( summary(dist_u) )
289
290 dist_u <- dist_u - min(dist_u)
291 dist_u <- round( dist_u / max(
292   dist_u) * 100 ) + 1
293
294 if (color_universal_design == 0){
295   color_act <- cm.colors(101)[dist
296     _u]
297   color_line <- "gray70"
298   color_cls <- "gray45"
299 } else {
300   library(RColorBrewer)
301   if (T){
302     col_seed <- brewer.pal(9,
303       "GnBu")
304     myPalette <-
305       colorRampPalette( col_seed
306         )
307     color_act <- myPalette(101)[
308       dist_u]
309     color_act <- adjustcolor(color_
310       act, alpha=0.8)
311     color_line <- "white"
312     color_cls <- "gray30"
313   } else {
314     col_seed <- rev(brewer.pal(9,
315       "RdYlBu"))
316     myPalette <-
317       colorRampPalette( col_seed
318         )
319     color_act <- myPalette(101)[
320       dist_u]
321     color_act <- adjustcolor(color_
322       act, alpha=0.8)
323     color_line <- "gray50"
324     color_cls <- "gray25"
325   }
326 }
327
328 #color_line <- "gray70"
329 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 <- row2coods[i,1]
326     y <- row2coods[i,2]
327
328     polygon(
329       x=c( x + 0.5, x + 0.5, x, x - 0.5,
330             x - 0.5, x ),
330       y=c( y + a, y - a, y - b, y - a,
331             y + a, y + b ),
331       col=color_act[i],
332       border="white",
333       lty=0,
334     )
335   }
336
337   for (i in 0:(n_nodes - 1)){
338     for (h in 0:(n_nodes - 2)){
339       if ( colors[h + i * n_nodes + 1]
340           == colors[h + i * n_nodes +
341             2] ){
340         x <- h
341         y <- i
342         if ( y %% 2 == 1 ){
343           x <- x + 0.5
344         }
345
346         segments(
347           x + 0.5, y + a,
348           x + 0.5, y - a,
349           col=color_line,
350           lwd=w_lwd,
351         )
352       }
353     }
354   }
355
356   for (i in 0:(n_nodes - 1)){
357     for (h in c(-1, n_nodes-1) ){
358       x <- h
359       y <- i
360       if ( y %% 2 == 1 ){
361         x <- x + 0.5
362       }
363       segments(
364         x + 0.5, y + a,
365         x + 0.5, y - a,
366         col=color_line,
367         lwd=w_lwd,
368       )
369     }
370     if ( y %% 2 == 0 ){
371       segments(
372         -0.5, y + a,
373         0 , y + 1 - a,
374         col=color_line,
375         lwd=w_lwd,
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
413         || is.na(colors[h + chk + (i+1)
414           * n_nodes + 1]) == 1
414         || h + chk == n_nodes
415       ){
416         next
417       }
418
419       if (
420         colors[h + i * n_nodes + 1]
421         == colors[h + chk + (i+1) * n
422           _nodes + 1]
422       ){
423         x <- h
424         y <- i
425         if ( y %% 2 == 1 ){
426           x <- x + 0.5
427         }
428
429         segments(
430           x, y + b,
431           x + 0.5, y + a,
432           col=color_line,
433           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
503     if (
504       is.na(colors[h + i * n_nodes
505           + 1]) == 1
506       || is.na(colors[h + chk + (i+1)
507           * n_nodes + 1]) == 1
508       || h + chk == n_nodes
509     ){
510       next
511     }
512     if (
513       colors[h + i * n_nodes + 1]
514       != colors[h + chk + (i+1) * n_
515           nodes + 1]
516     ){
517       x <- h
518       y <- i
519       if (y %% 2 == 1){
520         x <- x + 0.5
521
522       segments(
523         x, y + b,
524         x + 0.5, y + a,
525         col=color_cls,
526         lwd=cls_lwd,
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
539     if (
540       is.na(colors[h + i * n_nodes
541           + 1]) == 1
542       || is.na(colors[h - chk + (i+1)
543           * n_nodes + 1]) == 1
544       || h - chk < 0
545     ){
546       next
547     }

```

```

546
547   if (
548     colors[h + i * n_nodes + 1]
549     != colors[h - chk + (i+1) * n_
550       nodes + 1]
551     ){
552       x <- h
553       y <- i
554       if ( y %% 2 == 1 ){
555         x <- x + 0.5
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=
581   T, ncol=2 )
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
590           - c, x ),
591         y=c( y + a, y - a, y - b, y -
592           a, y + a, y + b ),
593         col=color_ptf,
594         border=color_pte,
595         lty=1,
596       )
597     }
598   } else {
599     symbols(
600       points[,1],
601       points[,2],
602       squares=rep(0.35,length(
603         points[,1])),
604       #circles=rep(0.2,length(points
605       [,1])),,
606       fg="gray70",
607       bg=color_ptf,
608       inches=F,
609       add=T,
610     )
611   }
612 library(maptools)
613 if (is.null(labcd) == 1){
614   labcd <- pointLabel(
615     x=points[,1],
616     y=points[,2],
617     labels=word_labs,
618     doPlot=F,
619     cex=cex,
620     offset=0
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 <-
632       function (x, y, words, cex =
633         1, rotate90 = FALSE, xlim = c
634         (-Inf,
635           Inf), ylim = c(-Inf, Inf),
636           tstep = 0.1, rstep = 0.1,
637           ...)
638   }
639   tails <- \g|j|p|q|y\
640   n <- length(words)
641   sdx <- sd(x, na.rm = TRUE)
642   sdy <- sd(y, na.rm = TRUE)
643   iterations <- 0
644   if (sdx == 0)
645     sdx <- 1
646   if (sdy == 0)
647     sdy <- 1
648   if (length(cex) == 1)
649     cex <- rep(cex, n)
650   if (length(rotate90) == 1)
651     rotate90 <- rep(rotate90, n)
652   boxes <- list()
653   for (i in 1:length(words)) {
654     rotWord <- rotate90[i]
655     r <- 0
656     theta <- runif(1, 0, 2 * pi)
657     x1 <- xo <- x[i]
658     y1 <- yo <- y[i]
659     wid <- strwidth(words[i],
660       cex = cex[i], ...)
661     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     0.5 * ht > ylim[1] && x1
670     + 0.5 * wid < xlim[2]
671     &&
672     y1 + 0.5 * ht < ylim[2])
673   {
674     boxes[[length(boxes) +
675       1]] <- c(x1 - 0.5 *
676       wid,
677       y1 - 0.5 * ht, wid, ht)
678     isOverlaped <- FALSE
679   }
680   else {
681     theta <- theta + tstep
682     r <- r + rstep * tstep/
683     (2 * pi)
684     x1 <- xo + sdx * r * cos
685     (theta)
686     y1 <- yo + sdy * r * sin
687     (theta)
688     iterations <- iterations
689     + 1
690     if (iterations > 500000){
691       boxes[[length(boxes) +
692         1]] <- c(x1 - 0.5 *
693         wid,
694         y1 - 0.5 * ht, wid, ht)
695       isOverlaped = FALSE
696     }
697   }
698   print( paste("iterations: ",
699     iterations) )
700   result <- do.call(rbind,
701     boxes)
702   colnames(result) <- c(\"x\",
703     \"y\", \"width\", \"ht\")
704   rownames(result) <- words
705   result
706 }
707   ", filename)
708 insertSource(filename, package="
709   "wordcloud", force=FALSE)
710 nc <- wordlayout(
711   labcd$x,
712   labcd$y,
713   word_labs,
714   cex=cex * 1.25,
715   xlim=c( par( "usr" )[1], par( "
716     usr" )[2]),
717   ylim=c( par( "usr" )[3], par( "
718     usr" )[4] )
719 )
720
721
722
723
724
725
726
727
728
729
730
731
732 if ( exists("segs") ){
733   if ( is.null(segs) == F){
734     for (i in 1:nrow(segs)){
735       segments(
736         segs[i,1],segs[i,2],segs[i,3],segs[i
737         ,4],
738         col="gray60",
739         lwd=1
740       )
741     }
742   }
743
744 text(
745   labcd$x,
746   labcd$y,
747   labels=word_labs,
748   cex=cex,
749   offset=0,
750   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 points2<-tail(points,n=190)
765 par(new=T)
766 plot(points2[,1],points2[,2],type="c",
767       col="blue")
```
