卒業論文

環境認識ライフログからの行動パターン解析による類似性・イベント検出

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

富山県立大学　電子・情報工学科
1415048　福嶋 瑞希
指導教員　奥原 浩之 教授
平成30年2月6日

記号一覧

以下に本論文において用いられる用語と記号の対応表を示す.

用語	記号
クラスター	X，Y
クラスター内での重心とサンプルとの距離の 2 乗和	$\mathrm{L}(X)$，$\mathrm{L}(Y)$
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	$\mathrm{L}(X \cup Y)$
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

第1章

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ライフログデータは，個人の生活に生かしたり，社会に生かしたりできると考えられている．

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

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

§ 1.2　本研究の目的

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

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

§ 1.3　本論文のあらまし

本論文は次のように構成される．

第 1 章：本章 第 1 章では，本研究の概要と目的について説明した．

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

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

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

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

第 6 章 第 6 章では，まとめと今後の課題を述べる．

第2章

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「Lifelog[1]」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 2[2]」(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「Swarm[4]」(図 2.4 参照) というアプリケーションがある．このアプリケーションは行動の記録を取ることができるため，日々の生活や旅行の記録として使用できる．上記のアプリケーションは GPS のアクセス許可が必要であり，上記以外のライフログアプリケーションも GPS を必要とすることが多い．

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

しかし，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 Edition[5]」

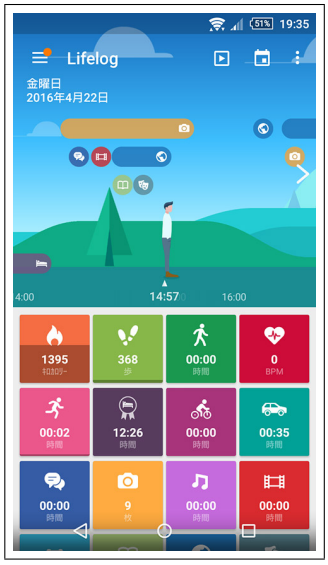

図 2.1: Lifelog

図 2.2: SmartBand 2

(図 2.5 参照)，SmartEyeglass[6]」(図 2.6 参照) などが挙げられる．

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

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

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

[5]https://www.x.company/glass/
[6]https://developer.sony.com/ja/develop/smarteyeglass-sed-e1/

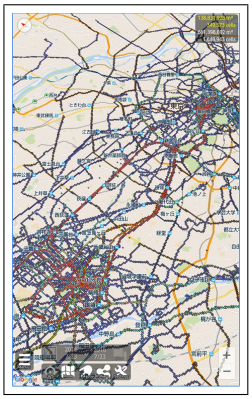

図 2.3: マッピング-GPS ログまとめて全部記録

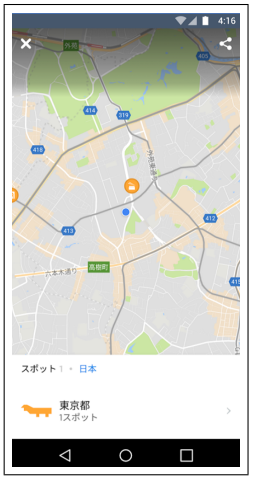

図 2.4: Swarm

図 2.5: Glass Enterprise Edition

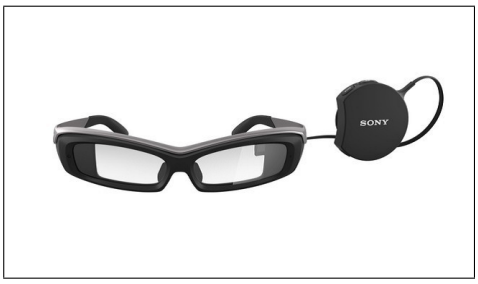

図 2.6: SmartEyeglass

§ 2.3　画像認識 API

画像認識技術とは，コンピュータに画像を理解をさせる技術である．画像内のピクセル信号のパターンから意味を抽出するパターン認識により，人間の視覚機能をコンピュータに処理させることができる．画像認識技術を用いることで，画像が持つ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 API[7]は，写真の被写体を機械判定し，ラベリングする機能をもっている．Google Cloud Vision API を用いて個々の写真の情報（ラベル）を取得できるが，撮影内容が不明瞭のときは，1 つも得られないこともあり [12]，同じ単語を複数個返して来る場合もある．初年度無料である．

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 API[8]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 API[9]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

第3章

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 Kinect™(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の行動は認識できない [16]．なお，Kinect™ 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 Kinect™ 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)

図 3.2: Kinect™

フ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 Coder[11]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可能であり，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必要となるため，Stop words と

[11]http://khc.sourceforge.net/
[12]http://khc.sourceforge.net/dl3.html

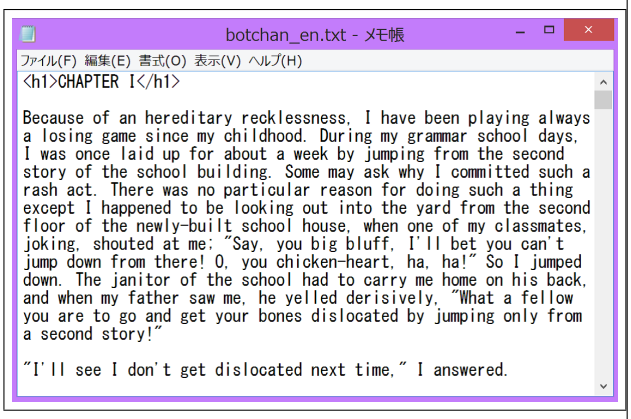

図 3.3: 「坊ちゃん」英語版テキストデータの一部

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 Tagger[13]を使用して自然言語処理を行う（表 3.1）．前処理を行うことで，多変量解析に使用する「各文書に，それぞれの語が何度出現していたのかという集計表」である「文書×抽出語」表 [21](表 3.2 参照) を出力することができる．h1 から h5 というのは見出し番号であり，h1 タグや h2 タグが存在すると 1 増加する．dan は段落番号で，bun は文番号である．id は文書の通し番号で，リセットされることは無い．length_c は文書の長さを文字数で表し，length_w は文書の長さを語数で表したものである．

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

階層的クラスター分析は，抽出語の最も似ている組み合わせから順番にクラスターにしていく方法であり，デンドログラムを表示する [22]．指定されたクラスター数に全体を分割し，その結果を色分けによって表示する．なお，KH Coder では，デフォルトの Auto では，抽出語数の平方根を四捨五入したものを用いている [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 ∪ 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，計算量は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

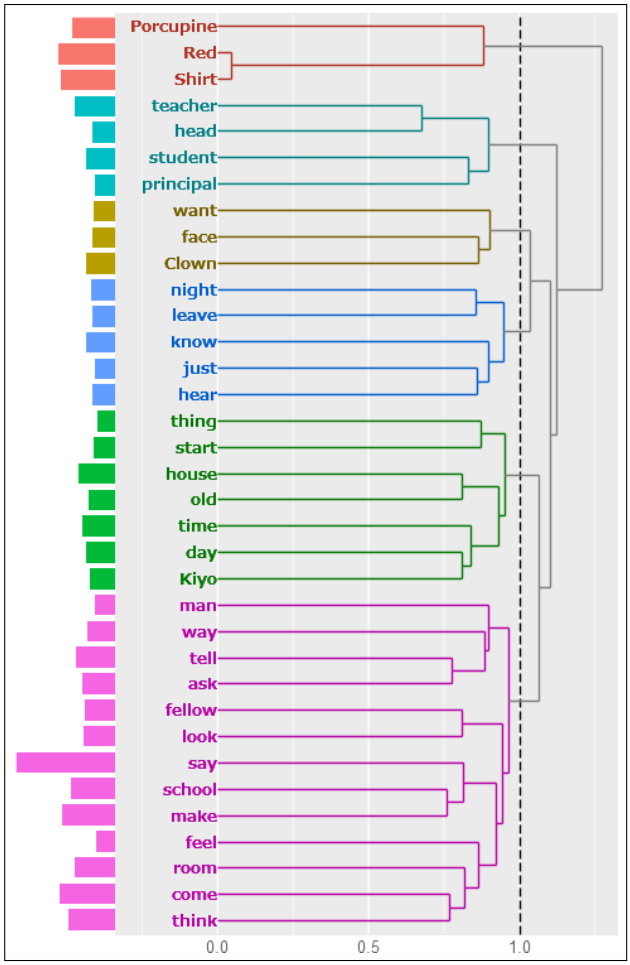

図 3.5: 「坊ちゃん」データから作成したクラスター分析

いという強みがある [25] [26].

$$\Delta = L(X \cup Y) - L(X) - L(Y) \tag{3.1}$$

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

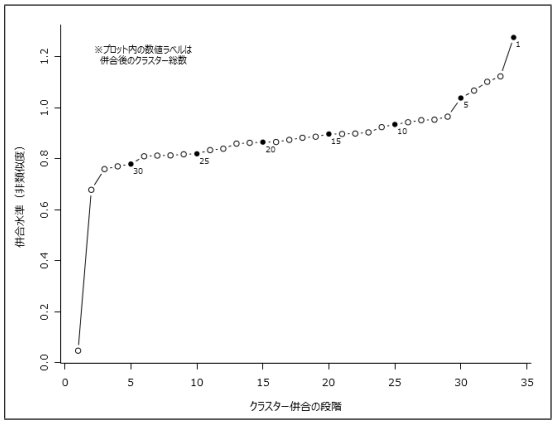

図 3.6: クラスター分析の併合標準

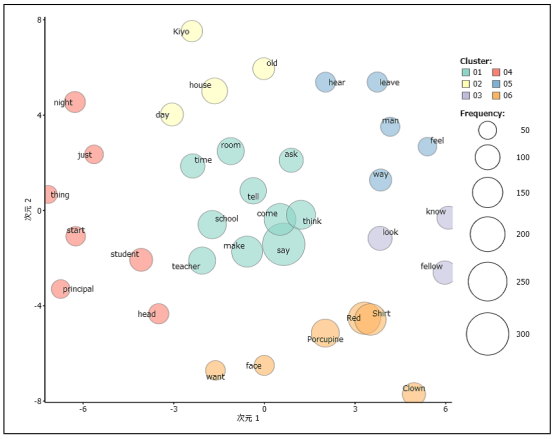

図 3.7: 「坊ちゃん」データから作成した MDS

るため軸の方向性に意味はない．

$$Jaccard\,係数 = \frac{語\,A\,と語\,B\,を含む文書}{語\,A\,もしくは語\,B\,を含む文書} \tag{3.2}$$

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

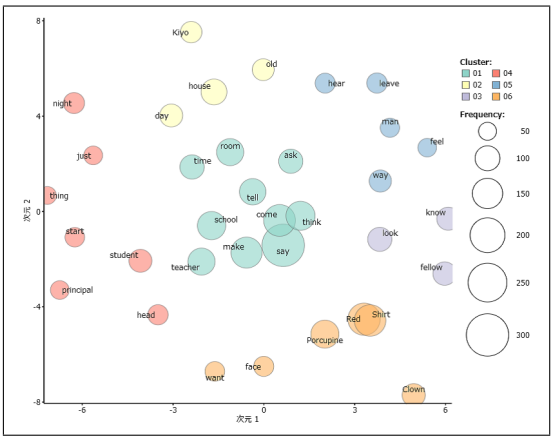

図 3.8: 「坊ちゃん」データから作成した MDS

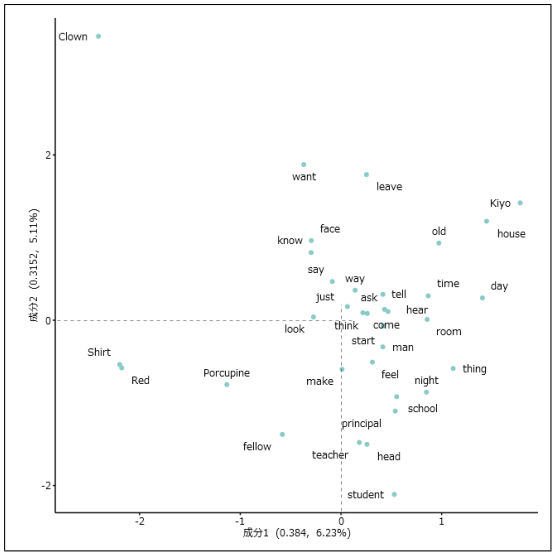

図 3.9: 「坊ちゃん」データから作成した対応分析

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 と共起関係があり，make や say という動詞とも関係性があることがわかる．room は come や think などの動詞と関係性があり，Red と Shirt は関係性があることが，視覚的に理解しやすくなっていると考えられる．

§ 3.3　類似性・イベント性

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

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

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

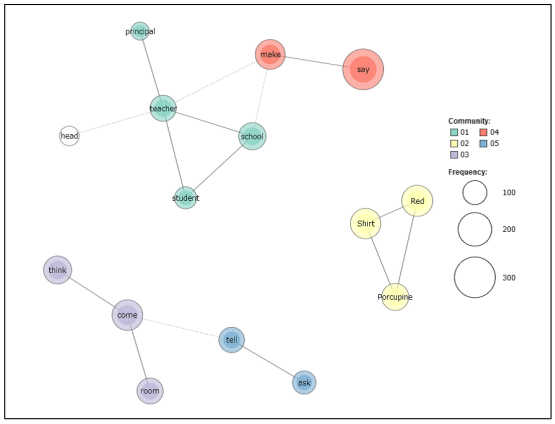

図 3.10: 「坊ちゃん」データから作成した共起ネットワーク

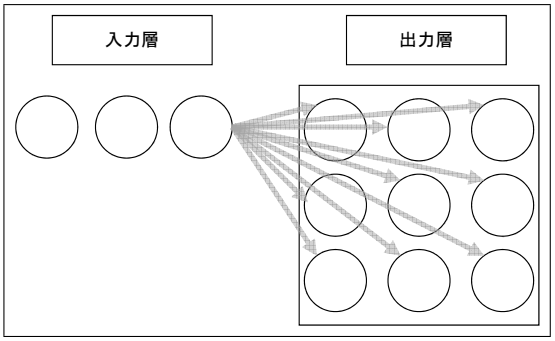

図 3.11: 入力層と出力層

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．

$$c = \arg\min_i \{||x - m_i||\} \tag{3.3}$$

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．

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

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

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．

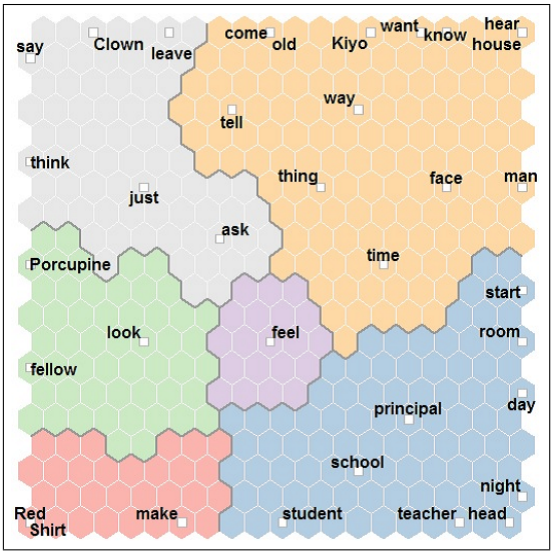

図 3.12: 「坊ちゃん」データから KH Coder で作成した SOM

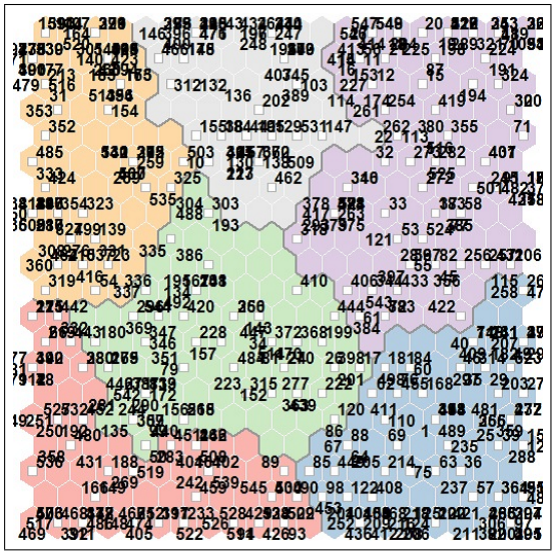

図 3.13: 「坊ちゃん」データから R で作成した SOM

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，今回のライフログデータの解析に用いることには向かないためである．

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

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

KH Coder と R を用いてライフログデータであるテキストデータの多変量解析を行い，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，ライフログデータの類似性やイベント性を検出できると考える．

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

階層的クラスター分析，共起ネットワークはクラスターの分かれ方やデータを構成する

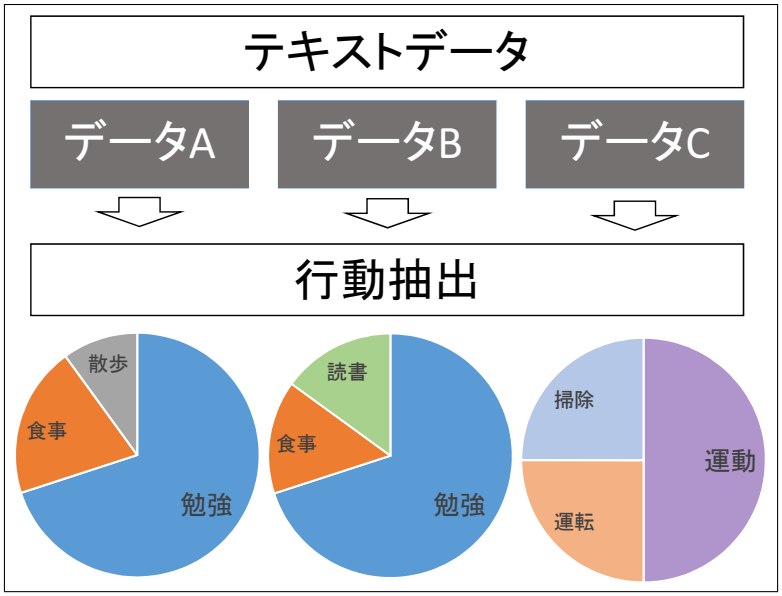

図 3.14: データ A, データ B, データ C の例

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，多くのライフログの中でも類似したライフログか，特徴的でイベント性のあるライフログかわかる．

図 3.14 は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ログデータであれば，平常日とイベント日の比較に利用できる．もし，三つのデータがバラバラのユーザーである場合，ライフログの類似性があるユーザーどうしでコミュニケーションを促進することができたり．ライフログの類似性がないユーザーどうしの比較を行うことで行動の中で改善すべき行動を検出できたりする [38]．このようにライフログデータの類似性やイベント性の検出は様々な応用が可能であると考えられる．

第4章

提案手法

§ 4.1　開発システムの概要

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

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

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

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

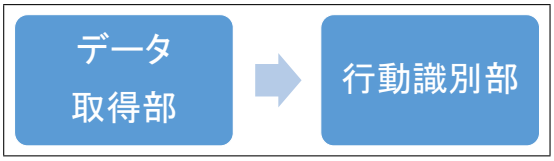

図 4.1: システム全体図

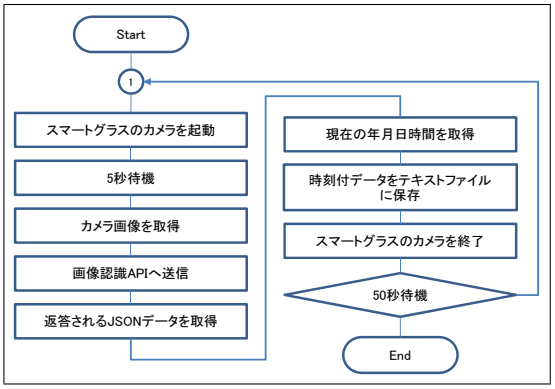

図 4.2: アプリケーションのフローチャート

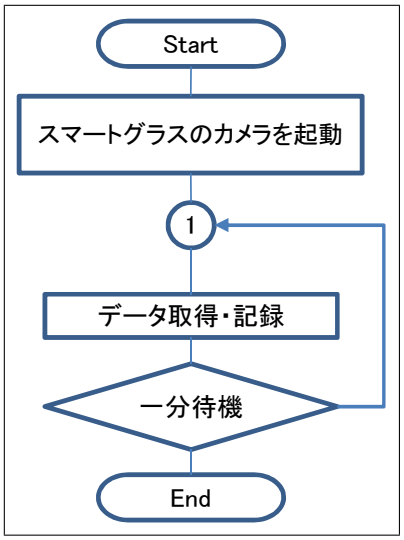

図 4.3: 省電力化前

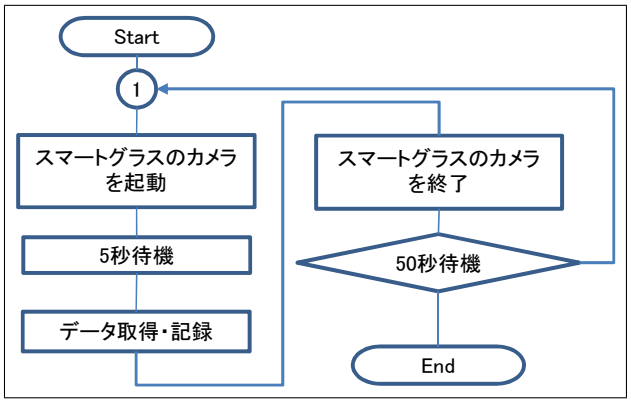

図 4.4: 省電力化後

§ 4.2　省電力化

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

この問題に対し，本研究ではカメラの起動・終了にかかるバッテリー消耗よりも，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，ライフログデータの周期性を検出する．

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

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

多変量解析を行う前に，前処理としてテキストデータのうち多変量解析に必要なデータのみを抽出する．Computer Vision APIはタグとキャプションをライフログデータとして取得できるが，一度に取得するデータが多すぎるとライフログデータとしてノイズとなってしまう．なお，この時の視界は，机上にノート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，ライフログデータの時系列関係がわかるように書き換える必要がある．

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

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

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

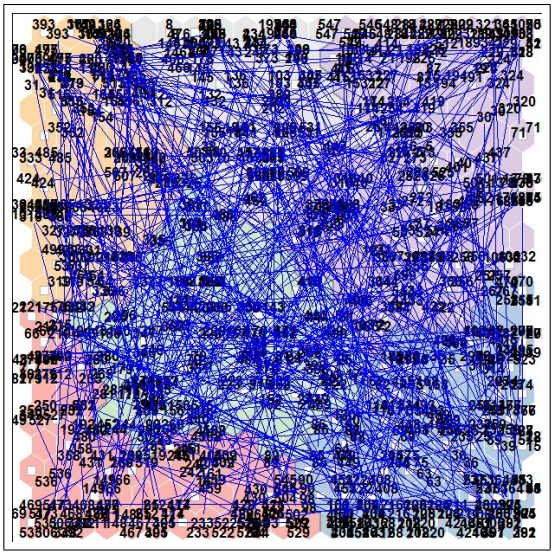

図 4.5: 「坊ちゃん」データから作成した SOM に線分を追加

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

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

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

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

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

第5章

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，ライフログデータを取得する．また，取得したデータを多変量解析を用いて，行動パターンの類似性やイベント性を検出する．

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

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

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

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

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

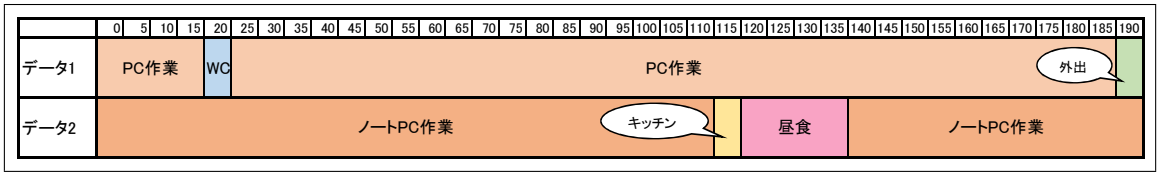

図 5.1: データ 1 とデータ 2 のタイムスケジュール

表 5.1: データ 3

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

スター分析である．この時クラスター数は Auto では 4 となり，図 5.3 の併合標準よりクラスター数 4 前後の傾きに大きな変化がないためクラスター数は 4 のままとした．データ 1 のクラスター分析より，赤のクラスターは computer や keyboard が含まれることから PC 作業であり出現回数もその他のクラスターに比べると最も多いことがわかる．青のクラスターは car や snow が含まれていること，outdoor という単語が含まれていることから外出時のことを表していると考えられる．紫のクラスターは女子トイレの壁の色である white が出現していることからトイレに行くことを示しているように考えた．緑のクラスターは 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ライフログデータとして取得するには，食べ物をなるべく視界に入れてデータを取得しなくてはいけないのではないかと考えた．また，図 5.2 と図 5.4 を比較すると，データ 2 の自宅の方が視界に入る物体が多いことから bottle や flyer などライフログデータに関係のないものまで取得されていることがわかる．

次にデータ 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

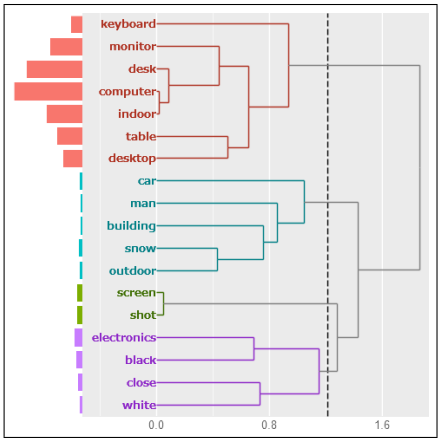

図 5.2: データ 1 のクラスター分析

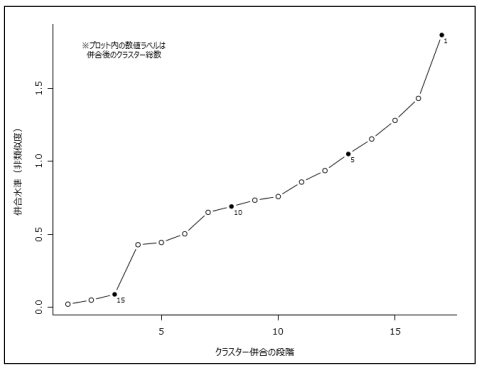

図 5.3: データ 1 の併合標準

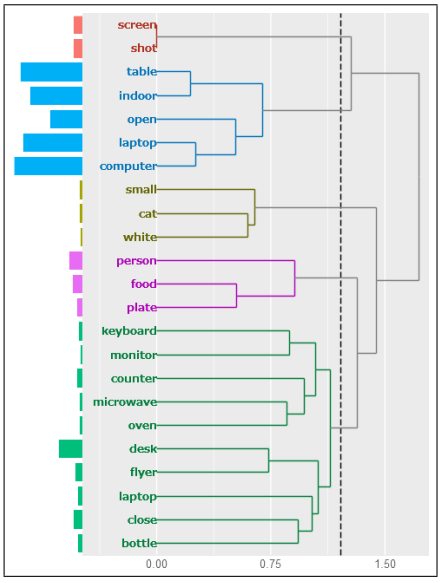

図 5.4: データ 2 のクラスター分析

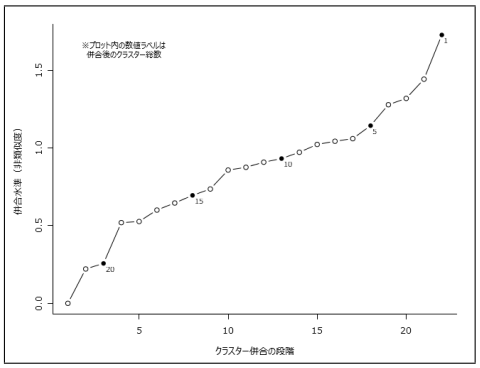

図 5.5: データ 2 の併合標準

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

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

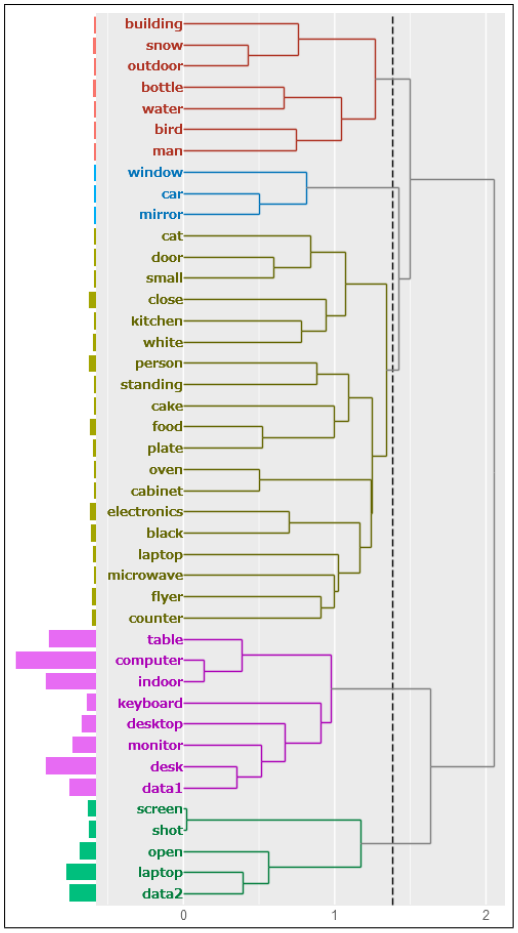

図 5.6: データ 3 のクラスター分析

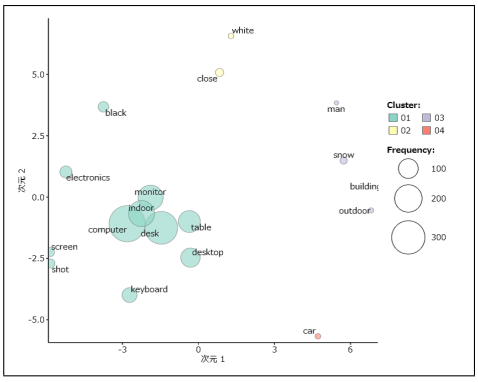

図 5.7: データ 1 の MDS

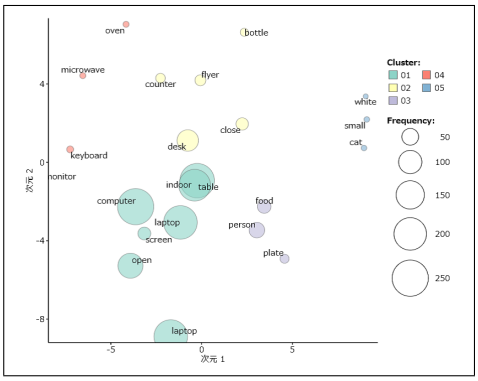

図 5.8: データ 2 の MDS

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

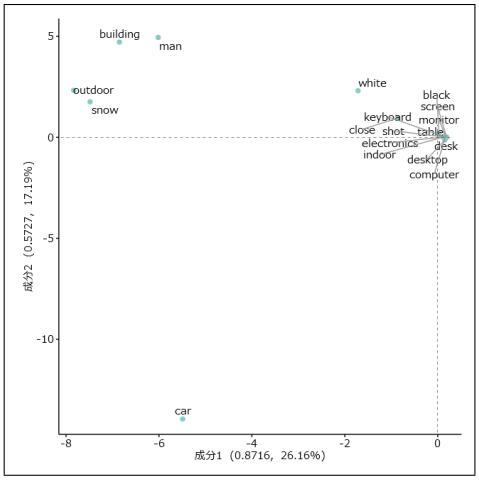

図 5.9: データ 1 の対応分析

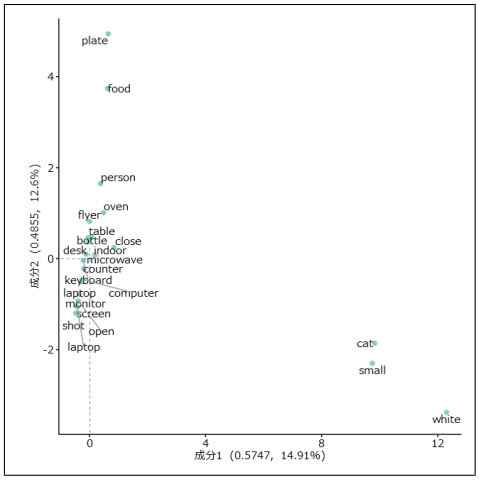

図 5.10: データ 2 の対応分析

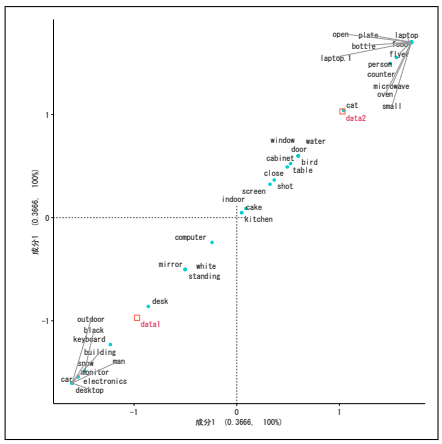

図 5.11: データ 3 の対応分析

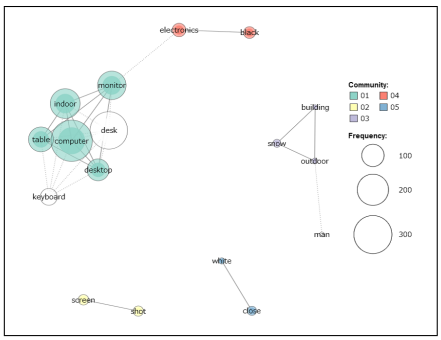

図 5.12: データ 1 の共起ネットワーク

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 の関係性を同時に出力することができるため，データ 3 の対応分析を行う．5.11 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 係数が 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，elevtronics と black とも，クラスターは違っているが

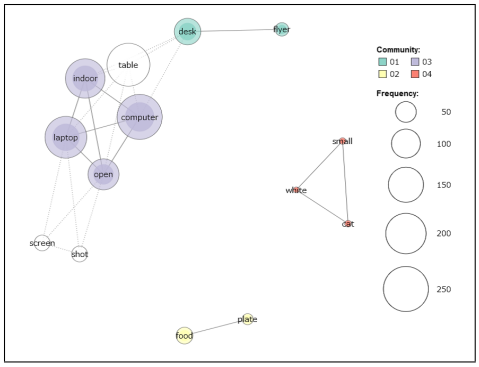

図 5.13: データ 2 の共起ネットワーク

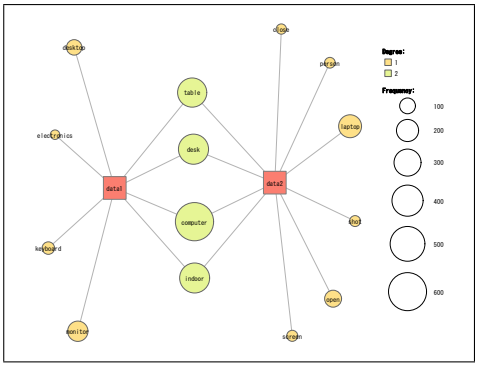

図 5.14: データ 3 の共起ネットワーク

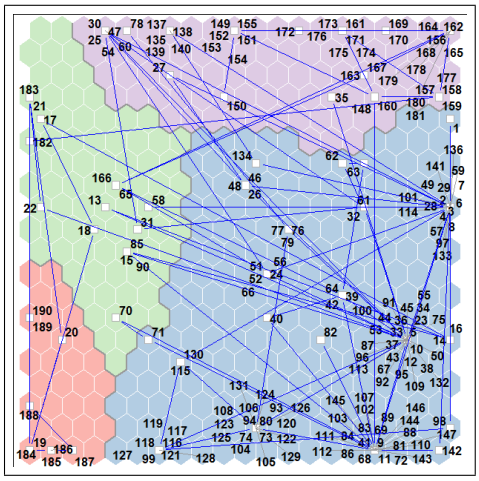

図 5.15: データ 1 の SOM

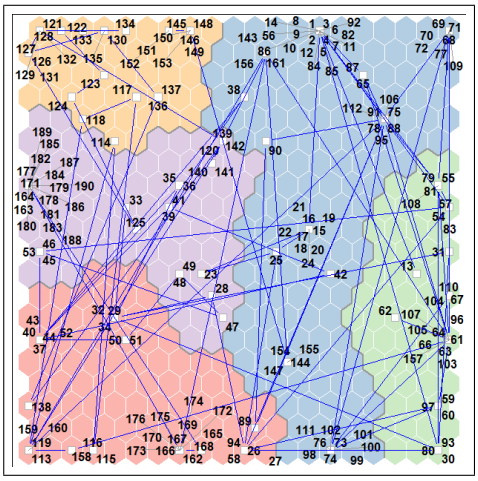

図 5.16: データ 2 の SOM

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出力するため，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 accard 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ライフログデータの時系列を可視化するため，SOM を作成する．クラスター数はデータ 1 は 4，データ 2 は 5 とした．

図 5.15 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表していると考えることができた．なお，外出時もトイレにいる際も視界は白色が多く，white という単語が共通するため，プロットが近いのだと考えた．また，紫のクラスターは a screen shot of a computer や a close up of a computer などの PC 作業の中で出現したノイズのようなテキストから構成されていたが，これはコンピュータースクリーンに近づいて作業を行っている際に出現するテキストであり，青と紫のクラスターは同じ PC 作業を表している．

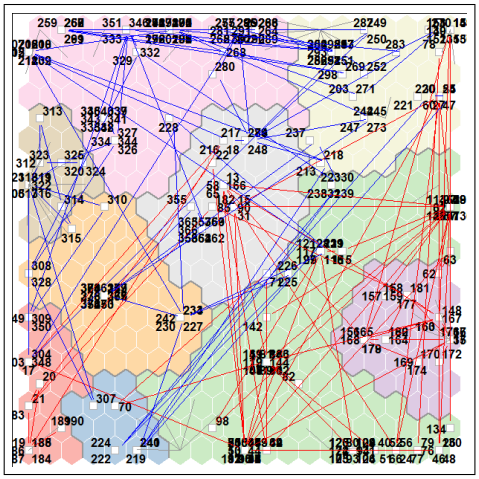

図 5.17: データ 3 の SOM

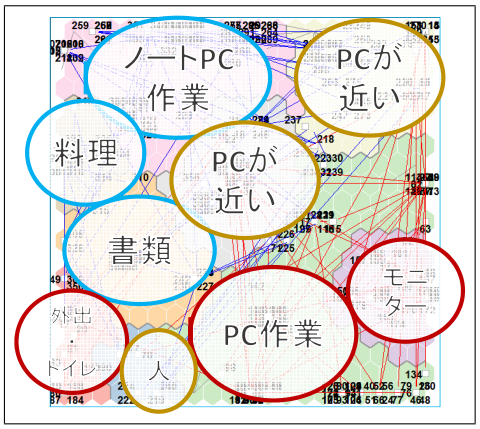

図 5.18: データ 3 の SOM に行動を追記した SOM

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ースクリーンに近づいて作業を行っている際に出現するテキストから構成されているため，青と緑のクラスターは同じ PC 作業を表している．また，クラスター間を大きくまたぐ線などもあり，常に同じ行動を行っていても，視界に入る物体の変化から線が乱雑になっていると感じた．

データ 3 の SOM を作成し，データ 1 とデータ 2 の時系列の類似性を比較する．データは赤色の線分，データ 2 は青色の線分で示す．クラスター数はデータ 1 とデータ 2 のクラスター数を合わせ 9 とした．また，データ 3 の SOM を作成するソースコードはソースコード A.??に示す．図 5.17 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っていると考える．また，SOM の解析から，同じ行動でも視界に写る物体の違いから行動の類似性やイベント性を検出できた．この結果から，同じ行動でも使用する場所や物体の変化によって別行動として認識させることができるため，GPS を使用せず，ライフログデータに位置情報を付加できると考えられる．よって，個人情報保護に着目し取得したライフログデータから類似性やイベント性を検出できたと考える．

第6章

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

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

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

謝辞

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

2018 年 2 月

福嶋 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, 伊藤愼, “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ＮＴＴ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, 千竃航平, 放地宏佳, 小木真人, 樫原裕大, 芝星帆, 眞島一貴, 努吉永, “AirTarget：光学シースルー方式 HMD とマーカレス画像認識による高可搬性実世界志向インタフェース”, 情報処理学会論文誌, Vol. 55, No. 4, pp. 1415–1427, 2014.

[11] 川上晃平, “スマートグラスを利用した授業支援システムの開発”, 2017.

[12] 倉田陽平, 真田風, 鈴木祥平, 石川博, “Flickr と Google Cloud Vision API によりテーマ別観光マップを作る試み”, http://db-event. jpn.org/deim2017/papers/321.pdf, 閲覧日 2018,1,4.

[13] 大雄治, 吉川眞, 田中一成, “ソーシャルメディアを活用した景観の分析と評価”, 日本都市計画学会関西支部研究発表会講演概要集, Vol. 15, pp. 13–16, 2017.

[14] 小林亜令, 岩本健嗣, 西山智, “釈迦：携帯電話を用いたユーザ移動状態推定・共有方式-モバイルコンピューティング, モバイルアプリケーション, ユビキタス通信, モバイルマルチメディア通信-”, 電子情報通信学会技術研究報告. MoMuC, モバイルマルチメディア通信, Vol. 108, No. 44, pp. 115–120, 2008.

[15] 寺田努, “ウェアラブルセンサを用いた行動認識技術の現状と課題”, コンピュータ ソフトウェア, Vol. 28, No. 2, pp. 43–54, 2011.

[16] 貴志一樹, 山崎俊彦, 相澤清晴, “机上行動のライフログのための行動認識”, 映像情報メディア学会年次大会講演予稿集, Vol. 2014, , 2014.

[17] 前川卓也, 柳沢豊, 岸野泰恵, 石黒勝彦, 亀井剛次, 櫻井保志, 岡留剛, “ウェアラブルセンサによるモノを用いた行動の認識について”, Technical Report 57, 研究報告ユビキタスコンピューティングシステム（UBI）, 2010.

[18] 樋口耕一, “テキスト型データの計量的分析：2 つのアプローチの峻別と統合”, 理論と方法, Vol. 19, No. 1, pp. 101–115, 2004.

[19] 佐野香織, 李在鎬, “KH Coder で何ができるか：日本語習得・日本語教育研究利用への示唆”, 言語文化と日本語教育, Vol. 33, pp. 94–95, 2007.

[20] “KH Coder を用いた研究事例のリスト”, http://khc.sourceforge.net/bib.html, 閲覧日 2018,1,7.

[21] 二宮隆次, 小野浩幸, 高橋幸司, 野田博行, “新聞記事を基にしたテキストマイニング手法による産学官連携活動分析”, 科学・技術研究, Vol. 5, No. 1, pp. 93–104, 2016.

[22] “クラスター分析の手法（階層クラスター分析） — データ分析基礎知識”, https://www.albert2005.co.jp/knowledge/data_mining/cluster/hierarchical_clustering, 閲覧日 2018,1,24.

[23] “階層的クラスター分析について”, http://koichi.nihon.to/cgi-bin/bbs_khn/khcf.cgi?list=&no=977&mode=allread&page=0, 閲覧日 2018,1,24.

[24] 吉原一紘, 徳高平蔵, “クラスター分析の概要”, *Journal of Surface Analysis*, Vol. 21, No. 1, pp. 10–17, 2014.

[25] 李美龍, 田中恒也, 成田吉弘, “画像を用いた製品の「飽き」に関する感性評価: －デザインの視覚的要素を中心に－”, 日本感性工学会論文誌, Vol. 11, No. 3, pp. 407–417, 2012.

[26] 小峯敦・下平裕之, “ベヴァリッジ『自由社会における完全雇用』のケインズ的要素-テキストマイニングを加味した量的・質的分析-”, 2017.

[27] 齋藤堯幸, “多次元尺度構成法”, 計測と制御, Vol. 22, No. 1, pp. 126–131, 1983.

[28] Joseph B Kruskal, “Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis”, *Psychometrika*, Vol. 29, No. 1, pp. 1–27, 1964.

[29] “Jaccard 係数の計算式と特徴（1）”, https://www.slideshare.net/khcoder/jaccard1, 閲覧日 2018,2,3.

[30] 中山慶一郎, “< 研究ノート > 対応分析によるデータ解析”, 関西学院大学社会学部紀要, No. 108, pp. 133–145, 2009.

[31] 田中京子, “KH Coder と R を用いたネットワーク分析”, 久留米大学コンピュータジャーナル, Vol. 28, pp. 37–52, 2014.

[32] “共起ネットワークにおける中心性の解釈について”, http://www.koichi.nihon.to/cgi-bin/bbs_khn/khcf.cgi?no=2493&mode=allread#2496, 閲覧日 2018,2,2.

[33] 横田尚己, 山田圭二郎, “熊本地震のつぶやきに見る感情極性値の時空間解析”, 都市計画論文集, Vol. 52, No. 3, pp. 1081–1087, 2017.

[34] 増田正, Masuda Tadashi, 高崎経済大学地域政策学部, “地方議会の会議録に関するテキストマイニング分析 : 高崎市議会を事例として”, 地域政策研究 = Studies of regional policy, Vol. 15, No. 1, pp. 17–31, 2012.

[35] T. KOHONEN, “Self-organized formation of topologically correct feature map”, *Biol. Cybern.*, Vol. 43, pp. 59–69, 1982.

[36] 岡晋之介, “自己組織化マップを用いた気象要素の分類と予測”, http://www.gifu-nct.ac.jp/elec/deguchi/sotsuron/oka/oka.html, 閲覧日 2018,1,7.

[37] “自己組織化特徴マップ（SOM）”, http://www.sist.ac.jp/ kanakubo/research/neuro/selforganizingmap.html, 閲覧日 2018,1,31.

[38] 勝治宏基, 米澤拓郎, 中澤仁, 高汐一紀, 徳田英幸ほか, “Synchrometer: ライフログを利用した日常行動における他者との類似度生成”, 研究報告ユビキタスコンピューティングシステム (UBI), Vol. 2013, No. 17, pp. 1–7, 2013.

付録

A. 1　ライフログデータ取得アプリケーションのソースコード

ライフログデータ取得アプリケーションのソースコード A.1 をしめす.

ソースコード A. 1: app.cs

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

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

A. 2　クラスター分析を作成するソースコード

クラスター分析を作成するソースコード A.2 をしめす.

ソースコード A. 2: kura.r

```
d <- NULL
d <- matrix( c(1,…省略…,0), byrow=T, nrow=190, ncol=19 )
d <- d[,-1]
colnames(d) <- c("desk",…省略…,"outdoor")
doc_length_mtr <- matrix( c(70,18,…省略…45,17), ncol=2, byrow=T)
colnames(doc_length_mtr) <- c("length_c", "length_w")
color_universal_design <- 1

d <- t(d)
# END: DATA
n_cls <- 4
font_size <- 1
labels <- rownames(d)
rownames(d) <- NULL
freq <- NULL
for (i in 1:nrow(d)) {
  freq[i] = sum( d[i,] )
}
method_dist <- "binary"
method_clst <- "ward"

library(amap)
dj <- Dist(d,method=method_dist)

if (
      ( as.numeric( R.Version()$major ) >= 3 )
  && ( as.numeric( R.Version()$minor ) >= 1.0)
){ # >= R 3.1.0
  if (method_clst == "ward"){
    method_clst <- "ward.D2"
  }
  hcl <- hclust(dj,method=method_clst)
} else { # <= R 3.0
  if (method_clst == "ward"){
    dj <- dj^2
    hcl <- hclust(dj,method=method_clst)
    hcl$height <- sqrt( hcl$height )
  } else {
    hcl <- hclust(dj,method=method_clst)
  }
}
par(
      mai=c(0,0,0,0),
      mar=c(1,2,1,0),
      omi=c(0,0,0,0),
      oma=c(0,0,0,0)
    )


library(grid)
library(ggplot2)
library(ggdendro)

ddata <- dendro_data(as.dendrogram(hcl), type="rectangle")

p <- NULL
p <- ggplot()

font_family <- "Meiryo UI"
if ( exists("PERL_font_family") ){
  font_family <- PERL_font_family
}

if (n_cls > 1){
  memb <- cutree(hcl,k=n_cls)

  p <- p + scale_colour_hue(l=40, c=100)

  cutpoint <- mean(
    c(
      rev(hcl$height)[n_cls-1],
      rev(hcl$height)[n_cls]
    )
  )

  n <- length( unique(memb[hcl$order]) )
  new_col <- NULL
  for (i in 1:ceiling(n / 2) ){
    new_col <- c(new_col, i)
    if (i + ceiling(n / 2) <= n){
      new_col <- c(new_col, i + ceiling(n / 2))
    }
  }

  col_tab <- cbind(
    unique(memb[hcl$order]),
    new_col
  )
  colnames(col_tab) <- c("org","new")
  col_vec <- NULL
  for (i in col_tab[order(col_tab[,1]),2]){
    c <- as.character(i)
    while (nchar(c) < 3){
      c <- paste("0",c,sep="")
    }
    col_vec <- c(col_vec, c)
  }

  seg_bl <- NULL
  seg_cl <- NULL
  colnames(ddata$segment) <- c(
    "x0",
    "y0",
```

```
    "x1",
    "y1"
  )
  colnames(ddata$labels) <- c(
    "x",
    "y",
    "text"
  )
  for ( i in 1:nrow( ddata$segment ) ) {
    if (
          ddata$segment$y0[i] > cutpoint
      || ddata$segment$y1[i] > cutpoint
      || (
            ddata$segment$y0[i] >= cutpoint
        && ddata$segment$y1[i] >= cutpoint
          )
    ) {
      seg_bl <- c(
        seg_bl,
        ddata$segment$x0[i],
        ddata$segment$y0[i],
        ddata$segment$x1[i],
        ddata$segment$y1[i]
      )
    } else {
      seg_cl <- c(
        seg_cl,
        ddata$segment$x0[i],
        ddata$segment$y0[i],
        ddata$segment$x1[i],
        ddata$segment$y1[i],
        #col_vec[
          memb[hcl$order][
            floor(
              mean(
                ddata$segment$x0[i],
                ddata$segment$x1[i])
              )
          ]
        #]
      )
    }
  }
  seg_bl = matrix(seg_bl, byrow=T, ncol=4 )
  seg_cl = matrix(seg_cl, byrow=T, ncol=5 )

  if (is.null(seg_bl) == F){
    colnames(seg_bl) <- c("x0", "y0", "x1", "y1")
    seg_bl <- as.data.frame(seg_bl)

    if ( max(seg_bl$y1) > cutpoint ){
      p <- p + geom_hline(
        yintercept = cutpoint,
        colour="black",
        linetype=5,
        size=0.5
      )
    }
  }
  colnames(seg_cl) <- c("x0", "y0", "x1", "y1", "c")
  seg_cl <- as.data.frame(seg_cl)
  seg_cl$c <- col_vec[seg_cl$c]

  p <- p + geom_text(
    data=data.frame(
      x=label(ddata)$x,
      y=label(ddata)$y,
      text=labels[ as.numeric( as.vector( ddata$labels$text ) ) ],
      cols= col_vec[ memb[ as.numeric( as.vector( ddata$labels$text ) ) ] ]
    ),
    aes_string(
      x="x",
      y="y",
      label="text",
      colour="cols"
    ),
    hjust=1,
    angle =0,
    family = font_family,
    fontface = "bold",
    size = 5 * 0.85 * font_size
  )

  p <- p + geom_segment(
    data=seg_cl,
    aes_string(x="x0", y="y0", xend="x1", yend="y1", colour="c"),
    size=0.5
  )
} else {
  memb <- rep( c("a"), length( labels) )
  p <- p + scale_colour_manual( values=c("black"))
  seg_bl <- ddata$segment
  col_vec <- c("001")
  p <- p + geom_text(
    data=data.frame(
      x=label(ddata)$x,
      y=label(ddata)$y,
      text=labels[ as.numeric( as.vector( ddata$labels$text ) ) ],
      cols= col_vec[ memb[ as.numeric( as.vector( ddata$labels$text ) ) ] ]
    ),
    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
          ="x1", yend="y1"),
222     color="gray50",
223     linetype=1,
224   )
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.
          vector( ddata$labels$text )
          ) ],
232     cols= col_vec[ memb[ as.numeric
          ( as.vector( ddata$labels$
          text ) ) ] ]
233   ),
234   aes_string(
235     x="x",
236     y="y",
237     label="text",
238     colour="cols"
239   ),
240   hjust=1,
241   angle =0,
242   family = font_family,
243   fontface = "bold",
244   size = 5 * 0.85 * font_size
245 )
246
247
248 y_max <- max( ddata$segment$y1 )
249 y_min <- 0.2
250 # "strwidth" crashes if the device is
      cairo_pdf or cairo_ps
251 if (
252   is.na(dev.list()["cairo_pdf"])
253   && is.na(dev.list()["cairo_ps"])
254 ){
255   y_min <- max(
256     strwidth(
257       labels[ as.numeric( as.vector(
            ddata$labels$text ) ) ],
258       units = "figure",
259       font = 2
260     )
261   )
262 }
263 y_min <- ( 6 * y_max * y_min ) / (
      5 - 6 * y_min )
264 y_min <- y_min * 1.1
265 if (y_min > y_max * 2){
266   y_min <- y_max * 2
267 }
268 y_min <- y_min * -1
269
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="
        gray60"),
295   axis.text.y = element_text(size=12,
        colour="gray40"),
296   axis.text.x = element_text(size=12,
        colour="gray40"),
297   legend.position="none"
298 )
299
300 if (n_cls <= 1){
301   p <- p + theme(
302     axis.text.y = element_blank(),
303     axis.text.x = element_text(size
          =12,colour="black"),
304     axis.ticks = element_line(colour="
          black"),
305     #panel.grid.major = theme_blank
          (),
306     #panel.grid.minor = theme_blank
          (),
307     #panel.background = theme_blank
          (),
308     axis.line = element_line(colour =
          "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.
              vector( ddata$labels$text )
              ) ],
325       ht <- freq[ as.numeric( as.
              vector( ddata$labels$text )
              ) ],
326       cl <- col_vec[ memb[ as.
              numeric( as.vector( ddata$
              labels$text ) ) ] ],
327       od <- nrow(d):1
328     )
329
330     if (n_cls <= 1){
331       bard$cl <- "001"
332     }
333
334     p2 <- NULL
335     p2 <- ggplot()
336     p2 <- p2 + geom_bar(
337       stat="identity",
338       position = "identity",
339       width=0.75,
340       data=bard,
341       aes(
342         x=reorder(od,od),
343         y=ht,
344         fill=cl
345       )
346     )
347     p2 <- p2 + coord_flip()
348     p2 <- p2 + scale_y_reverse(
              expand = c(0,0))
349     p2 <- p2 + scale_x_discrete(
              expand = c(0,0) )
350     p2 <- p2 + theme(
351       axis.title.y = element_blank(),
352       axis.title.x = element_blank(),
353       axis.ticks = element_blank(),
354       axis.text.y = element_blank(),
355       axis.text.x = element_text(size
              =12,colour="white"),
356       legend.position = "none",
357       panel.background = element_rect
              (fill="white", colour="white
              "),
358       panel.grid.major = element_
              blank(),
359       panel.grid.minor = element_
              blank()
360     )
361
362     margin <- 0.002 * nrow(d) +
              0.00001 * nrow(d)^2 - 0.12
363     p2 <- p2 + theme(
364       plot.margin = unit(c
              (0.25,0,0.25,0), "lines") # r:
              -0.75
365     )
366
367     grid.newpage()
368     pushViewport(viewport(layout=
              grid.layout(1,2, width=c(1,5))
              ) )
369     print(p, vp= viewport(layout.pos.
              row=1, layout.pos.col=2) )
370     print(p2, vp= viewport(layout.pos.
              row=1, layout.pos.col=1) )
371   } else {
372     print(p)
373   }
374
375   if (
376     is.na(dev.list()["pdf"])
377     && is.na(dev.list()["postscript"
              ])
378     && is.na(dev.list()["cairo_pdf"])
379     && is.na(dev.list()["cairo_ps"])
380   ){
381     if ( grepl("darwin", R.version$
              platform) ){
382       quartzFonts(HiraKaku=quartzFont
              (rep("Meiryo UI",4)))
383       grid.gedit("GRID.text", grep=
              TRUE, global=TRUE, gp=
              gpar(fontfamily="HiraKaku"))
384     }
385   }
386
387   detach("package:ggdendro", unload
          =T)
388
389   # for clickable image map
390   exp <- (y_max - y_min ) * 0.02
391   coord <- cbind(
392     (1 / 6 + 5 / 6 * -1 * (y_min -
              exp) / ( (y_max + exp) - (y_
              min - exp) )) * 1.03,
393     1:length(ddata$labels$text) /
              length(ddata$labels$text)
394   )
395   rownames(coord) <-
396     labels[ as.numeric( as.vector(
              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

```
d <- NULL
d <- matrix( c(1,…Abbreviated…,0), byrow=T, nrow=190, ncol=19 )
d <- d[,-1]
colnames(d) <- c("desk",…省略…,"outdoor")
doc_length_mtr <- matrix( c(70,18,…省略…,45,17), ncol=2, byrow=T)
colnames(doc_length_mtr) <- c("length_c", "length_w")
color_universal_design <- 1
d <- t(d)
# END: DATA

library(amap)
check4mds <- function(d){
  jm <- as.matrix(Dist(d, method="binary"))
  jm[upper.tri(jm,diag=TRUE)] <- NA
  if ( length( which(jm==0, arr.ind=TRUE) ) ){
    return( which(jm==0, arr.ind=TRUE)[,1][1] )
  } else {
    return( NA )
  }
}

while ( is.na(check4mds(d)) == 0 ){
  n <- check4mds(d)
  print( paste( "Dropped object:", row.names(d)[n]) )
  d <- d[-n,]
}
dj <- Dist(d,method="binary")
random_starts <- 1
dim_n <- 2
method_mds <- "K"

if (method_mds == "K"){
  # Kruskal
  library(MASS)
  c <- isoMDS(dj, k=dim_n, maxit=3000, tol=0.000001)
  if (random_starts == 1){
      print("Running random starts ...")
      set.seed(123)
      for (i in 1:1000){ # 200sec
        if (dim_n == 1){
          init <- cbind( rnorm( nrow(d)) )
        } else if (dim_n == 2){
          init <- cbind( rnorm( nrow(d)), rnorm(nrow(d)) )
        } else if (dim_n == 3){
          init <- cbind( rnorm( nrow(d)), rnorm(nrow(d)), rnorm(nrow(d)) )
        } else {
          warn("Error: invalid dimesion number!")
        }
        ct <- isoMDS(dj, y=init, k=dim_n, maxit=3000, tol=0.000001, trace=F)
        if (ct$stress < c$stress){
          c <- ct
          print( paste("random start #", i, ": ", c$stress, sep=""))
        }
      }
  }
  cl <- c$points
} else if (method_mds == "S"){
  #Sammon
  library(MASS)
  c <- sammon(dj, k=dim_n, niter=3000, tol=0.000001)
  if (random_starts == 1){
      print("Running random starts ...")
      set.seed(123)
      for (i in 1:1000){ # 200sec
        if (dim_n == 1){
          init <- cbind( rnorm( nrow(d)) )
        } else if (dim_n == 2){
          init <- cbind( rnorm( nrow(d)), rnorm(nrow(d)) )
        } else if (dim_n == 3){
          init <- cbind( rnorm( nrow(d)), rnorm(nrow(d)), rnorm(nrow(d)) )
        } else {
          warn("Error: invalid dimesion number!")
        }
        ct <- sammon(dj, y=init, k=dim_n, niter=3000, tol=0.000001, trace=F)
        if (ct$stress < c$stress){
          c <- ct
          print( paste("random
```

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

```
  dpi <- 72 * min(png_width, png_width) / 640 * 1
  p_size <- 12 * dpi / 72;
  png("temp.png", width=png_width, height=png_height, unit="px", pointsize=p_size)

  #if ( exists("PERL_font_family") ){
  # par(family=PERL_font_family)
  #}

  plot(cl)
  library(maptools)
  labcd <- pointLabel(
    x=cl[,1],
    y=cl[,2],
    labels=rownames(cl),
    cex=font_size,
    offset=0,
    doPlot=F
  )

  xorg <- cl[,1]
  yorg <- cl[,2]
  cex <- font_size
  segs <- NULL

  if ( length(xorg) < 300 ) {
    library(wordcloud)

# fix for "wordlayout" function
filename <- tempfile()
writeLines("wordlayout <- function (x, y, words, cex = 1, rotate90 = FALSE, xlim = c(-Inf,
  Inf), ylim = c(-Inf, Inf), tstep = 0.1, rstep = 0.1, ...)
{
  tails <- \"g|j|p|q|y\"
  n <- length(words)
  sdx <- sd(x, na.rm = TRUE)
  sdy <- sd(y, na.rm = TRUE)
  iterations <- 0
  if (sdx == 0)
    sdx <- 1
  if (sdy == 0)
    sdy <- 1
  if (length(cex) == 1)
    cex <- rep(cex, n)
  if (length(rotate90) == 1)
    rotate90 <- rep(rotate90, n)
  boxes <- list()
  for (i in 1:length(words)) {
    rotWord <- rotate90[i]
    r <- 0
    theta <- runif(1, 0, 2 * pi)
    x1 <- xo <- x[i]
    y1 <- yo <- y[i]
    wid <- strwidth(words[i], cex = cex[i], ...)
    ht <- strheight(words[i], cex = cex[i], ...)
    if (grepl(tails, words[i]))
      ht <- ht + ht * 0.2
    if (rotWord) {
      tmp <- ht
      ht <- wid
      wid <- tmp
    }
    isOverlaped <- TRUE
    while (isOverlaped) {
      if (!.overlap(x1 - 0.5 * wid, y1 - 0.5 * ht, wid, ht, boxes) && x1 - 0.5 * wid > xlim[1] && y1 -
        0.5 * ht > ylim[1] && x1 + 0.5 * wid < xlim[2] &&
        y1 + 0.5 * ht < ylim[2]) {
        boxes[[length(boxes) + 1]] <- c(x1 - 0.5 * wid,
          y1 - 0.5 * ht, wid, ht)
        isOverlaped <- FALSE
      }
      else {
        theta <- theta + tstep
        r <- r + rstep * tstep/(2 * pi)
        x1 <- xo + sdx * r * cos(theta)
        y1 <- yo + sdy * r * sin(theta)
        iterations <- iterations + 1
        if (iterations > 500000){
          boxes[[length(boxes) + 1]] <- c(x1 - 0.5 * wid,
          y1 - 0.5 * ht, wid, ht)
          isOverlaped = FALSE
        }
      }
    }
  }
  print( paste(\"iterations: \", iterations) )
  result <- do.call(rbind, boxes)
  colnames(result) <- c(\"x\", \"y\", \"width\", \"ht\")
  rownames(result) <- words
  result
}
", filename)
insertSource(filename, package="wordcloud", force=FALSE)
```

```
276
277     nc <- wordlayout(
278       labcd$x,
279       labcd$y,
280       rownames(cl),
281       cex=cex * 1.25,
282       xlim=c( par( "usr" )[1], par( "
              usr" )[2] ),
283       ylim=c( par( "usr" )[3], par( "
              usr" )[4] )
284     )
285
286     xlen <- par("usr")[2] - par("
          usr")[1]
287     ylen <- par("usr")[4] - par("
          usr")[3]
288
289     for (i in 1:length(rownames(cl))
          ){
290       x <- ( nc[i,1] + .5 * nc[i,3] -
              labcd$x[i] ) / xlen
291       y <- ( nc[i,2] + .5 * nc[i,4] -
              labcd$y[i] ) / ylen
292       dst <- sqrt( x^2 + y^2 )
293       if ( dst > 0.05 ){
294         segs <- rbind(
295           segs,
296           c(
297             nc[i,1] + .5 * nc[i,3],
298             nc[i,2] + .5 * nc[i,4],
299             xorg[i],
300             yorg[i]
301           )
302         )
303       }
304     }
305     xorg <- labcd$x
306     yorg <- labcd$y
307     labcd$x <- nc[,1] + .5 * nc[,3]
308     labcd$y <- nc[,2] + .5 * nc[,4]
309   }
310   dev.off()
311 }
312
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,
          width=2,flag="0")
331
332   cls_labels <- paste(cls_labels, " "
          )
333 } else {
334   cls_labels <- "cluster 1"
335 }
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=
            col_f, fill=col_f),
355     shape=21,
356     colour="gray40",
357     alpha=alpha_value
358   )
359   g <- g + scale_size_area(
360     max_size= 30 * bubble_size / 100,
361     guide = guide_legend(
362       title = "Frequency:",
363       override.aes = list(colour="
              black", alpha=1),
364       label.hjust = 1,
365       order = 2
366     )
367   )
368 } else {
369   if ( n_cls > 0 ){
370     g <- g + geom_point(
371       data=cl2,
372       aes(x=d1, y=d2, colour=col_f,
              fill=col_f),
373       size=5.5,
374       shape=21,
375       colour="gray40",
376       alpha=alpha_value
377     )
378   } else {
379     g <- g + geom_point(
380       data=cl2,
381       aes(x=d1, y=d2),
382       size=2,
383       shape=16,
384       colour="mediumaquamarine"
385     )
```

```
  }
}

if ( n_cls > 0 ){
  g <- g + scale_fill_brewer(
    palette = "Set3",
    guide = guide_legend(
      title = "Cluster:",
      override.aes = list(size=5.5,
          alpha=1, shape=22),
      keyheight = unit(1.25,"line"),
      ncol=2,
      order = 1
    )
  )
} else {
  g <- g + scale_fill_brewer(
    palette = "Set3",
    guide = "none"
  )
}

# Text
if (plot_mode == "color") {
  if (text_font == 1){
    face <- "plain"
  } else {
    face <- "bold"
  }
  g <- g + geom_text(
    data=cl2,
    aes(x=lx,y=ly,label=labels),
    size=4,
    colour="black",
    family=font_family,
    fontface=face
  )
  if (length(segs) > 0){
    colnames(segs) <- c("x1", "y1",
        "x2", "y2")
    segs <- as.data.frame(segs)
    g <- g + geom_segment(
      aes(x=x1, y=y1, xend=x2, yend
          =y2),
      data=segs,
      colour="gray60"
    )

  }
}

# Appearance / Theme
g <- g + labs(x=name_dim1, y=
    name_dim2)
g <- g + theme_classic(base_family=
    font_family)
g <- g + theme(
  legend.key = element_rect(colour =
      "transparent"),
  axis.line.x = element_line(colour =
      "black", size=0.5),
  axis.line.y = element_line(colour =
      "black", size=0.5),
  axis.title.x = element_text(face="
      plain", size=11, angle=0),
  axis.title.y = element_text(face="
      plain", size=11, angle=90),
  axis.text.x = element_text(face="
      plain", size=11, angle=0),
  axis.text.y = element_text(face="
      plain", size=11, angle=0),
  legend.title = element_text(face="
      bold", size=11, angle=0),
  legend.text = element_text(face="
      plain", size=11, angle=0),
  plot.margin = margin(6, 6, 6, 0, "
      pt")
)

# fix range
out_coord <- cbind( cl2$lx, cl2$ly )
rownames(out_coord) <- cl2$labels
xlimv <- c(
  min( out_coord[,1] ) - 0.04 * ( max
      ( out_coord[,1] ) - min( out_
      coord[,1] ) ),
  max( out_coord[,1] ) + 0.04 * (
      max( out_coord[,1] ) - min(
      out_coord[,1] ) )
)
ylimv <- c(
  min( out_coord[,2] ) - 0.04 * ( max
      ( out_coord[,2] ) - min( out_
      coord[,2] ) ),
  max( out_coord[,2] ) + 0.04 * (
      max( out_coord[,2] ) - min(
      out_coord[,2] ) )
)

# aspect ratio
if (fix_asp == 1){
  g <- g + coord_fixed(
    xlim=xlimv,
    ylim=ylimv,
    expand = F
  )
} else {
  g <- g + coord_cartesian(
    xlim=xlimv,
    ylim=ylimv,
    expand = F
  )
}

# coordinates for saving
add <- -1 * xlimv[1]
div <- add + xlimv[2]
out_coord[,1] <- ( out_coord[,1] +
    add ) / div

add <- -1 * ylimv[1]
div <- add + ylimv[2]
```

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

A. 4　対応分析を作成するソースコード

対応分析を作成するソースコード A.4 をしめす.

ソースコード A. 4: taiou.r

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

```
flt <- 0
flw <- 60
bubble_plot <- 0
biplot <- 0
cex=1
use_alpha <- 1
show_origin <- 1
scaling <- "none"

    if ( exists("saving_emf") ||
         exists("saving_eps") ){
      use_alpha <- 0
    }
  name_dim <- '\u6210\u5206'
name_eig <- '\u56fa\u6709\u5024'
name_exp <- '\u5bc4\u4e0e\u7387
    '
library(MASS)

# Filter words by chi-square value
if ( (flw > 0) && (flw < ncol(d)) ){
  sort <- NULL
  for (i in 1:ncol(d) ){
    # print( paste(colnames(d)[i],
         chisq.test( cbind(d[,i], n_total
         - d[,i]) )$statistic) )
    sort <- c(
      sort,
      chisq.test( cbind(d[,i], n_total -
           d[,i]) )$statistic
    )
  }
  d <- d[,order(sort,decreasing=T)]
  d <- d[,1:flw]

  d <- subset(d, rowSums(d) > 0)
  if (exists("doc_length_mtr")){
    doc_length_mtr <- subset(doc_
         length_mtr, rowSums(d) > 0)
    n_total <- doc_length_mtr[,2]
  }
}

d_max <- min( nrow(d), ncol(d) )
    - 1
if (d_x > d_max){
  d_x <- d_max
}
if (d_y > d_max){
  d_y <- d_max
}

c <- corresp(d, nf=d_max )

if (d_max == 1){
  c$cscore <- as.matrix( c$cscore )
  c$rscore <- as.matrix( c$rscore )
  colnames(c$cscore) <- c("X1")
  colnames(c$rscore) <- c("X1")
}

# Dilplay Labels only for distinctive
    words
if ( (flt > 0) && (flt < nrow(c$cscore
    )) ){
  sort <- NULL
  limit <- NULL
  names <- NULL
  ptype <- NULL

  # compute distance from (0,0)
  for (i in 1:nrow(c$cscore) ){
    sort <- c(sort, c$cscore[i,d_x] ^
         2 + c$cscore[i,d_y] ^ 2 )
  }

  # Put labels to top words
  limit <- sort[order(sort,
       decreasing=T)][flt]
  for (i in 1:nrow(c$cscore) ){
    if ( sort[i] >= limit ){
      names <- c(names,
           rownames(c$cscore)[i])
      ptype <- c(ptype, 1)
    } else {
      names <- c(names, NA)
      ptype <- c(ptype, 2)
    }
  }
  rownames(c$cscore) <- names;
} else {
  ptype <- 1
}

pch_cex <- 1
if ( v_count > 1 ){
  pch_cex <- 1.25
}

# Zooming area near the origin

log_conv <- function(x, y, a){
  log_base <- 10

  # Find Cosine theta
  OA <- sqrt( x^2 + y^2 )
  OA[OA == 0] <-
       0.00000000000000000001
  Cos <- x / OA

  # Convert OA
  OA <- log(OA + 1, log_base)
  OA <- OA * a
  OA <- log(OA + 1, log_base)
  OA <- OA * a
  OA <- log(OA + 1, log_base)

  # Find OB
  OB <- Cos * OA

  # Find AB
  AB = sqrt( OA^2 - OB^2 )
```

```
  AB[y < 0] <- AB[y < 0] * -1

  cbind(OB, AB)
}

axp <- NULL
if (zoom_factor >= 1 ){
  scaling <- "none"
  axp <- c(0,0,1)

  r <- log_conv( c$cscore[,d_x], c$
      cscore[,d_y], zoom_factor )
  c$cscore[,d_x] <- r[,1]
  c$cscore[,d_y] <- r[,2]

  r <- log_conv( c$rscore[,d_x], c$
      rscore[,d_y], zoom_factor )
  c$rscore[,d_x] <- r[,1]
  c$rscore[,d_y] <- r[,2]
}

# Scaling
asp <- 0
if (scaling == "sym"){
  for (i in 1:d_max){
    c$cscore[,i] <- c$cscore[,i] * c$
        cor[i]
    c$rscore[,i] <- c$rscore[,i] * c$
        cor[i]
  }
  asp <- 1
} else if (scaling == "symbi"){
  for (i in 1:d_max){
    c$cscore[,i] <- c$cscore[,i] * sqrt
        ( c$cor[i] )
    c$rscore[,i] <- c$rscore[,i] * sqrt
        ( c$cor[i] )
  }
  asp <- 1
}


k <- c$corˆ2
txt <- cbind( 1:length(k), round(k
    ,4), round(100*k / sum(k),2) )
colnames(txt) <- c(name_dim,name
    _eig,name_exp)
print( txt )
inertias <- round(k,4)
k <- round(100*k / sum(k),2)
font_size <- 1
resize_vars <- 1
bubble_size <- 100
labcd <- NULL

plot_mode <- "color"


library(ggplot2)

font_family <- "Meiryo UI"


if ( exists("PERL_font_family") ){
  font_family <- PERL_font_family
}

if ( exists("bs_fixed") == F ) {
  bubble_size <- bubble_size / 1.3
  bs_fixed <- 1
}

if (biplot == 1 && plot_mode != "
    vars"){
  cb <- rbind(
    cbind(c$cscore[,d_x], c$cscore[,d_
        y], ptype),
    cbind(c$rscore[,d_x], c$rscore[,d_y
        ], v_pch)
  )
} else if (plot_mode == "vars") {
  cb <- cbind(c$rscore[,d_x], c$
      rscore[,d_y], v_pch)
} else {
  cb <- cbind(c$cscore[,d_x], c$
      cscore[,d_y], ptype)
}

if ( (is.null(labcd) && plot_mode !
    = "dots" ) || plot_mode == "
    vars"){

  png_width <- 640
  png_height <- 640
  png_width <- png_width - 0.16 *
      1.3 * bubble_size / 100 * png_
      width
  dpi <- 72 * min(png_width, png_
      height) / 640 * 1.3
  p_size <- 12 * dpi / 72;
  png("temp.png", width=png_width,
      height=png_height, unit="px",
      pointsize=p_size)

  #if ( exists("PERL_font_family") ){
  # par(family=PERL_font_family)
  #}

  plot(
    x=c(c$cscore[,d_x],c$rscore[,d_x]),
    y=c(c$cscore[,d_y],c$rscore[,d_y]),
    asp=asp
  )

  library(maptools)
  labcd <- pointLabel(
    x=cb[,1],
    y=cb[,2],
    labels=rownames(cb),
    cex=font_size,
    offset=0,
    doPlot=F
  )
```

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

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

```
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="
              black", fill=NA, alpha=1),
452         label.hjust = 1,
453         order = 2
454       )
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) /
              max(n_total) * 0.6,
485       type = v_pch
486     )
487
488     if ( (resize_vars == 1) && (bubble_
          plot == 1) ) {
489       g <- g + geom_point(
490         data=df.vars,
491         aes(x=x, y=y, size=size, shape=
              factor(type) ),
492         #colour = NA,
493         fill = col_bg_vars,
494         alpha=0.2,
495         show.legend = F
496       )
497
498       g <- g + geom_point(
499         data=df.vars,
500         aes(x=x, y=y, size=size, shape=
              factor(type) ),
501         colour = col_dot_vars,
502         fill = NA,
503         alpha=1,
504         show.legend = F
505       )
506     } else {
507       g <- g + geom_point(
508         data=df.vars,
509         aes(x=x, y=y, shape=factor(
              type) ),
510         colour = NA,
511         fill = col_bg_vars,
512         alpha=0.2,
513         size=3.5,
514         show.legend = F
515       )
516
517       g <- g + geom_point(
518         data=df.vars,
519         aes(x=x, y=y, shape=factor(
              type) ),
520         colour = col_dot_vars,
521         fill = NA,
522         alpha=1,
523         size=3.5,
524         show.legend = F
525       )
526     }
527
528     g <- g + scale_shape_manual(
529       values = c(22:25,0-6)
530     )
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"
```

```
  col_txt_vars <- "black"
}

if (plot_mode == "dots"){
  col_txt_words <- NA
  col_txt_vars <- NA
}

if ( text_font == 1 ){
  font_face <- "plain"
} else {
  font_face <- "bold"
}

if ( exists("df.labels.save ") == F
    ){
  df.labels.save <- data.frame(
    x = labcd$x,
    y = labcd$y,
    labs = rownames(cb),
    cols = cb[,3]
  )
}

if (plot_mode != "dots") {
  df.labels <- data.frame(
    x = labcd$x,
    y = labcd$y,
    labs = rownames(cb),
    cols = cb[,3]
  )
  if ( plot_mode == "gray" ){
    df.labels.var <- subset(df.
        labels, cols == 3)
    df.labels <- subset(df.labels,
        cols != 3)
    g <- g + geom_label(
      data=df.labels.var,
      family=font_family,
      fontface="bold",
      #label.size=0.25,
      label.padding=unit(1.8, "mm"),
      colour="white",
      fill="gray50",
      #alpha=0.7,
      aes(x=x, y=y,label=labs)
    )
    if ( (resize_vars == 0) || (bubble_
        plot == 0) ) {
      g <- g + geom_point(
        data=df.vars,
        aes(x=x, y=y, shape=factor(
            type) ),
        colour = col_dot_vars,
        fill = NA,
        alpha=1,
        size=3.5,
        show.legend = F
      )
    }
  }

  g <- g + geom_text(
    data=df.labels,
    aes(x=x, y=y,label=labs,colour=
        factor(cols)),
    size=4,
    family=font_family,
    fontface=font_face
    #colour="black"
  )

  #label_legend <- guide_legend(
  # title = "Labels:",
  # key.theme = element_rect(colour
      = "gray30"),
  # override.aes = list(size=5),
  # order = 1
  #)
  label_legend <- "none"

  g <- g + scale_color_manual(
    values = c(col_txt_words, col_txt_
        vars, col_txt_vars),
    breaks = c(1,3),
    labels = c("Words / Codes", "
        Variables"),
    guide = label_legend
  )



  if ( exists("segs") ){
    if ( is.null(segs) == F){
      colnames(segs) <- c("x1", "
          y1", "x2", "y2")
      segs <- as.data.frame(segs)
      g <- g + geom_segment(
        aes(x=x1, y=y1, xend=x2,
            yend=y2),
        data=segs,
        colour="gray60"
      )
    }
  }
}

if (plot_mode == "vars"){
  labcd <- NULL
}

#if ( asp == 1 ){
# g <- g + coord_fixed()
#}

g <- g + labs(
  x=paste(name_dim,d_x," (",
      inertias[d_x],", ", k[d_x],"%)",
      sep=""),
  y=paste(name_dim,d_y," (",
      inertias[d_y],", ", k[d_y],"%)",
      sep="")
```

```
)
g <- g + theme_classic(base_family=
    font_family)
g <- g + theme(
  legend.key = element_rect(colour =
      NA, fill= NA),
  axis.line.x = element_line(colour =
      "black", size=0.5),
  axis.line.y = element_line(colour =
      "black", size=0.5),
  axis.title.x = element_text(face="
      plain", size=11, angle=0),
  axis.title.y = element_text(face="
      plain", size=11, angle=90),
  axis.text.x = element_text(face="
      plain", size=11, angle=0),
  axis.text.y = element_text(face="
      plain", size=11, angle=0),
  legend.title = element_text(face="
      bold", size=11, angle=0),
  legend.text = element_text(face="
      plain", size=11, angle=0)
)

if (show_origin == 1){
  line_color <- "gray30"

  lim_chk <-ggplot_build(g)
  xlims <- lim_chk$panel$ranges[[1]]
      $x.range
  ylims <- lim_chk$panel$ranges[[1]]
      $y.range
  if ( is.null(xlims) ){
    xlims <- lim_chk$layout$panel_
        ranges[[1]]$x.range
    ylims <- lim_chk$layout$panel_
        ranges[[1]]$x.range
  }

  if (zoom_factor >= 1){
    g <- g + scale_x_continuous(
        limits=xlims, expand=c(0,0),
         breaks=c(0) )
    g <- g + scale_y_continuous(
        limits=ylims, expand=c(0,0),
         breaks=c(0) )
  } else {
    g <- g + scale_x_continuous(
        limits=xlims, expand=c(0,0)
        )
    g <- g + scale_y_continuous(
        limits=ylims, expand=c(0,0)
        )
  }

  m_x <- (xlims[2] - xlims[1]) * 0.03
  m_y <- (ylims[2] - ylims[1]) * 0.03

  g <- g + geom_segment(
    aes(x = xlims[1], y = 0, xend = m
        _x, yend = 0),
    size=0.25,
    linetype="dashed",
    colour=line_color
  )
  g <- g + geom_segment(
    aes(x = 0, y = ylims[1], xend = 0,
        yend = m_y),
    size=0.25,
    linetype="dashed",
    colour=line_color
  )
} else {
  if (zoom_factor >= 1){
    g <- g + scale_x_continuous(
        breaks=c(0) )
    g <- g + scale_y_continuous(
        breaks=c(0) )
  }
}

# fix range
if ( exists("xlimv") == F ){
  # for setting xlim & ylim
  out_coord <- cbind(
    c( df.labels.save$x, df.words$x),
    c( df.labels.save$y, df.words$y)
  )

  xlimv <- c(
    min( out_coord[,1] ) - 0.04 * (
        max( out_coord[,1] ) - min(
        out_coord[,1] ) ),
    max( out_coord[,1] ) + 0.04 * (
        max( out_coord[,1] ) - min(
        out_coord[,1] ) )
  )
  ylimv <- c(
    min( out_coord[,2] ) - 0.04 * (
        max( out_coord[,2] ) - min(
        out_coord[,2] ) ),
    max( out_coord[,2] ) + 0.04 * (
        max( out_coord[,2] ) - min(
        out_coord[,2] ) )
  )

  # for saving
  out_coord <- cbind(
    df.labels.save$x,
    df.labels.save$y
  )
  rownames(out_coord) <- df.
      labels.save$labs
}

# aspect ratio
if (asp == 1){
  g <- g + coord_fixed(
    xlim=xlimv,
    ylim=ylimv,
    expand = F
  )
```

```
} else {
  g <- g + coord_cartesian(
    xlim=xlimv,
    ylim=ylimv,
    expand = F
  )
}

# coordinates for saving
if (plot_mode == "color"){
  df.labels.save <- subset(df.
      labels.save, cols != 3)
  out_coord <- cbind(
    df.labels.save$x,
    df.labels.save$y
  )
  rownames(out_coord) <- df.
      labels.save$labs

  add <- -1 * xlimv[1]
  div <- add + xlimv[2]
  out_coord[,1] <- ( out_coord[,1] +
      add ) / div

  add <- -1 * ylimv[1]
  div <- add + ylimv[2]
  out_coord[,2] <- ( out_coord[,2] +
      add ) / div
}

# fixing width of legends to 22%
library(grid)
library(gtable)
g <- ggplotGrob(g)

if ( bubble_plot == 0 ){
  saving_file <- 1
}

if ( exists("saving_file") ){
  if ( saving_file == 0){
    target_legend_width <-
        convertX(
      unit( image_width * 0.22, "in"
          ),
      "mm"
    )
    if ( as.numeric( substr(
        packageVersion("ggplot2"), 1,
         3) ) <= 2.1 ){ # ggplot2 <=
        2.1.0
      diff_mm <- diff( c(
        convertX( g$widths[5], "mm" ),
        target_legend_width
      ))
      if ( diff_mm > 0 ){
        g <- gtable_add_cols(g, unit(
            diff_mm, "mm"))
      }
    } else { # ggplot2 >= 2.2.0

      diff_mm <- diff( c(
        convertX( g$widths[7], "mm",
            valueOnly=T ) +
            convertX( g$widths[8], "
            mm", valueOnly=T ),
        target_legend_width
      ))
      if ( diff_mm > 0 ){
        print(diff_mm)
        g <- gtable_add_cols(g, unit(
            diff_mm, "mm"))
      }
    }
  }
}

# fixing width of left spaces to 4.25
    char
if ( as.numeric( substr(
    packageVersion("ggplot2"), 1, 3)
    ) <= 2.1 ){ # ggplot2 <= 2.1.0
  diff_char <- diff( c(
    convertX( g$widths[1] + g$widths
        [2] + g$widths[3], "char" ),
    unit(4.25, "char")
  ))
  if ( diff_char > 0 ){
    g <- gtable_add_cols(g, unit(diff
        _char, "char"), pos=0)
  }
}

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

```
d <- NULL
d <- matrix( c(1,…省略…,0), byrow
    =T, nrow=190, ncol=19 )
d <- d[,-1]
colnames(d) <- c("desk",…省略…,"
    outdoor")
doc_length_mtr <- matrix( c(
    70,18,…省略…,45,17), ncol=2,
    byrow=T)
```

```
colnames(doc_length_mtr) <- c("length_c", "length_w")
color_universal_design <- 1

d <- t(d)
# END: DATA
edges <- 0
th <- 0.2
cex <- 1
view_coef <- 0
fix_lab <- 1
use_freq_as_size <- 1
bubble_size <- 100
use_weight_as_width <- 0
smaller_nodes <- 0
text_font <- 1
min_sp_tree <- 0
min_sp_tree_only <- 0
cor_var <- 0
cor_var_darker <- 0
use_alpha <- 1
gray_scale <- 0
method_coef <- "binary"
com_method <- "com-b"

# Count frequency of each word
freq <- NULL
for (i in 1:length( rownames(d) )) {
  freq[i] = sum( d[i,] )
}

# Compute co-occurrence coefficient
if ( (exists("doc_length_mtr")) &! ( method_coef == "binary")){
  leng <- as.numeric(doc_length_mtr[,2])
  leng[leng ==0] <- 1
  d <- t(d)
  d <- d / leng
  d <- d * 1000
  d <- t(d)
}
if (method_coef == "euclid"){ # standardize for each word
  d <- t( scale( t(d) ) )
}

dr <- d
library(amap)
d <- Dist(d,method=method_coef)

d <- as.matrix(d)
if ( method_coef == "euclid" ){
  d <- max(d) - d
  d <- d / max(d)
} else {
  d <- 1 - d
}

# Delete unnecessary edges and standardize
if ( exists("com_method") ){
  if (com_method == "twomode_c" || com_method == "twomode_g"){
    d[1:n_words,] <- 0

    std <- d[(n_words+1):nrow(d),1:n_words]
    chkm <- std

    std <- t(std)
    std <- scale(std, center=T, scale=F)
    std <- t(std)

    if ( min(std[!is.na(std)]) < 0.0005 ){
      std <- std + ( 0.0005 - min( std[!is.na(std)]) );
    }
    std <- std / max(std[!is.na(std)])

    std[chkm == 0] <- 0
    d[(n_words+1):nrow(d),1:n_words] <- std
  }
}

# Make a graph
# For igraph > 1.0.0
library(igraph)
new_igraph <- 0
igraph_ver <- (
    as.numeric( substr(sessionInfo()$otherPkgs$igraph$Version, 1,1) ) * 10
  + as.numeric( substr(sessionInfo()$otherPkgs$igraph$Version, 3,3) )
)
if ( igraph_ver > 5){
  new_igraph <- 1
}

n <- graph.adjacency(d, mode="lower", weighted=T, diag=F)
n <- igraph::set.vertex.attribute(
  n,
  "name",
  (0+new_igraph):(length(d[1,])-1+new_igraph),
  as.character( 1:length(d[1,]) )
)

# Prepare for deleting weak edges
el <- data.frame(
  edge1 = get.edgelist(n,name=T)[,1],
  edge2 = get.edgelist(n,name=T)[,2],
  weight = igraph::get.edge.attribute(n, "weight"),
  stringsAsFactors = FALSE
```

```
)

# Find a threshold value
if (th < 0){
  if(edges > length(el[,1])){
    edges <- length(el[,1])
  }
  th = quantile(
    el$weight,
    names = F,
    probs = 1 - edges / length(el[,1])
  )
}

# Delete weak edges and make a graph again
el2 <- subset(el, el[,3] >= th)
if ( nrow(el2) == 0 ){
  stop(message = "No edges to draw!", call. = F)
}
n2 <- graph.edgelist(
  matrix( as.matrix(el2)[,1:2], ncol =2 ),
  directed =F
)
n2 <- igraph::set.edge.attribute(
  n2,
  "weight",
  (0+new_igraph):(length(get.edgelist(n2)[,1])-1+new_igraph),
  el2[,3]
)

if ( min_sp_tree_only == 1 ){
  n2 <- minimum.spanning.tree(
    n2,
    weights = 1 - igraph::get.edge.attribute(n2, "weight"),
    algorithm="prim"
  )
}


if (length(igraph::get.vertex.attribute(n2,"name")) < 2){
  com_method <- "none"
}

# Centrality
if ( com_method == "cnt-b" || com_method == "cnt-d" || com_method == "cnt-e"){
  ccol <- NULL
  if (com_method == "cnt-b"){ # betweenness
    ccol <- igraph::betweenness(
      n2,
      v=(0+new_igraph):(length(igraph::get.vertex.attribute(n2,"name"))-1+new_igraph),
      directed=F
    )
  }
  if (com_method == "cnt-d"){ # degree
    ccol <- igraph::degree(
      n2,
      v=(0+new_igraph):(length(igraph::get.vertex.attribute(n2,"name"))-1+new_igraph)
    )
  }
  if (com_method == "cnt-e"){ # evcent
    try(
      ccol <- igraph::evcent(n2)$vector,
      silent = T
    )
  }
  ccol_raw <- ccol

  edg_col <- "gray55"
  edg_lty <- 1
}

# Community detection
if (com_method == "com-b" || com_method == "com-g" || com_method == "com-r"){
  merge_step <- function(n2, m){
    for ( i in 1:( trunc( length( m ) / 2 ) ) ){
      temp_csize <- community.to.membership(n2, m,i)$csize
      num_max <- max( temp_csize )
      num_alone <- sum( temp_csize [ temp_csize == 1 ] )
      num_cls <- length( temp_csize [temp_csize > 1] )
      #print( paste(i, "a", num_alone, "max", num_max, "cls", num_cls) )
      if (

        num_max / length(igraph::get.vertex.attribute(n2,"name")) >= 0.225

        && num_max > num_alone

        && num_cls < 12
      ){
        return(i)
      }

      if (num_max / length(igraph::
```

```
            get.vertex.attribute(n2,"
            name")) >= 0.4 ){
201           return(i-1)
202         }
203       }
204       return( trunc(length( m ) / 2) )
205     }
206     # For igraph > 1.0.0
207     if (com_method == "com-b"){ #
          Betweenness
208       com <- edge.betweenness.
            community(n2, directed=F)
209       if (igraph_ver < 10){
210         com_m <- community.to.
              membership(
211           n2, com$merges, merge_step(
                n2,com$merges)
212         )
213         com_m$membership <- com_m
              $membership + new_igraph
214       }
215     }
216     if (com_method == "com-g"){ #
          Modularity
217       com <- fastgreedy.community (
            n2, merges=TRUE,
            modularity=TRUE)
218       if (igraph_ver < 10){
219         com_m <- community.to.
              membership(
220           n2, com$merges, merge_step(
                n2,com$merges)
221         )
222         com_m$membership <- com_m
              $membership + new_igraph
223       }
224     }
225     if (com_method == "com-r"){ #
          Random walks
226       com <- walktrap.community(
227         n2,
228         weights=igraph::get.edge.
              attribute(n2, "weight")
229       )
230       if (igraph_ver < 10){
231         com_m <- NULL
232         com_m$membership <- com$
              membership
233         com_m$csize <- table(com$
              membership)
234       }
235     }
236     if (igraph_ver >= 10){
237       com_m <- NULL
238       com_m$membership <- as.
            vector( membership(com) )
239       com_m$csize <- table(com_m$
            membership)
240     }
241
242     # Configure Edges
243     edg_lty <- NULL
244     edg_col <- NULL
245     for (i in 1:nrow(get.edgelist(n2,
          name=F))){
246       if (
247             com_m$membership[ get.
                  edgelist(n2,name=F)[i,1]
                  + 1 - new_igraph]
248         == com_m$membership[ get.
              edgelist(n2,name=F)[i,2] +
              1 - new_igraph]
249       ){
250         edg_col <- c( edg_col, "gray55
              " )
251         edg_lty <- c( edg_lty, 1 )
252       } else {
253         edg_col <- c( edg_col, "gray"
              )
254         edg_lty <- c( edg_lty, 3 )
255       }
256     }
257   }
258
259
260   if (com_method == "twomode_c" ||
        com_method == "twomode_g"){
261     if ( exists("var_select") ){
262       var_select_bak <- var_select
263     }
264
265     var_select <- substring(
266       colnames(d)[ as.numeric( igraph
            ::get.vertex.attribute(n2,"name
            ") ) ],
267       1,
268       2
269     ) == "<>"
270
271     if (length(var_select[var_select==
          TRUE]) == 0 && exists("var_
          select_bak")){
272       var_select <- var_select_bak;
273     }
274   }
275
276   if (com_method == "twomode_c"){
277     ccol <- igraph::degree(
278       n2,
279       v=(0+new_igraph):(length(
            igraph::get.vertex.attribute(n2
            ,"name"))-1+new_igraph)
280     )
281     # ggplot2
282     ccol_raw <- ccol
283     ccol_raw[var_select] <- NA
284     ccol_raw[ccol_raw >= 5] <- 5
285     ccol_raw <- as.character(ccol_raw
            )
286     ccol_raw[ccol_raw=="5"] <- "5+"
287
288     edg_col <- "gray70"
```

```
  edg_lty <- 1

}


if (com_method == "none" || com_
    method == "twomode_g"){
  edg_lty <- 1
  edg_col <- "gray40"
}

if (com_method == "twomode_g"){
  edg_lty <- 3
}

# Minimum Spanning Tree
if ( min_sp_tree == 1 ){

  mst <- minimum.spanning.tree(
    n2,
    weights = 1 - igraph::get.edge.
        attribute(n2, "weight"),
    algorithm="prim"
  )


  #if (length(edg_col) == 1){
    edg_col <- rep("gray55",
        length( igraph::get.edge.
        attribute(n2, "weight") ))
  #}

  n2_edges <- get.edgelist(n2,name=
      T);
  mst_edges <- get.edgelist(mst,
      name=T);

  if (exists("edg_lty") == F){
    edg_lty <- 1
  }
  for ( i in 1:ecount(n2) ){
    name_n2 <- paste(
      n2_edges[i,1],
      n2_edges[i,2]
    )
    for ( j in 1:ecount(mst) ){
      name_mst <- paste(
        mst_edges[j,1],
        mst_edges[j,2]
      )
      if ( name_n2 == name_mst ){
        edg_col[i] <- "gray30" #
            edge color
        if ( length(edg_lty) > 1 ){
          edg_lty[i] <- 1 # edge
              linetype
        }
        break
      }
    }
  }
  edg_mst <- edg_col
}


lay <- NULL
if ( length(igraph::get.vertex.
    attribute(n2,"name")) >= 3 ){
  d4l <- as.dist( shortest.paths(n2) )
  if ( min(d4l) < 1 ){
    d4l <- as.dist( shortest.paths(n2,
        weights=NA ) )
  }
  if ( max(d4l) == Inf){
    d4l[d4l == Inf] <- vcount(n2)
  }
  try( lay <- cmdscale( d4l, k=2 ),
      silent=TRUE )
  if ( is.null(lay) == F ){
    lay <- round(lay, digits=5)
    check4fr <- function(d){
      chk <- 0
      for (i in combn( length(d[,1]),
          2, simplify=F ) ){
        if (
            d[i[1],1] == d[i[2],1]
          && d[i[1],2] == d[i[2],2]
        ){
          return( i[1] )
        }
      }
      return( NA )
    }
    while ( is.na(check4fr(lay)) == 0
        ){
      mv <- check4fr(lay)
      lay[mv,1] <- lay[mv,1] + 0.001
      #print( paste( "Moved:", mv ) )
    }
  }
}


set.seed(100)
if (
     (com_method == "twomode_c" ||
         com_method == "twomode_
         g")
  && ( igraph::is.connected(n2) )
){
  # For igraph > 1.0.0
  if (igraph_ver >= 10){
    lay_f <- layout.kamada.kawai(
      n2,
      #coords = lay,
      weights = igraph::get.edge.
          attribute(n2, "weight")
    )
  } else {
    lay_f <- layout.kamada.kawai(
      n2,
      start = lay,
```

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

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

```
    sdy <- 1
  if (length(cex) == 1)
    cex <- rep(cex, n)
  if (length(rotate90) == 1)
    rotate90 <- rep(rotate90, n)
  boxes <- list()
  for (i in 1:length(words)) {
    rotWord <- rotate90[i]
    r <- 0
    theta <- runif(1, 0, 2 * pi)
    x1 <- xo <- x[i]
    y1 <- yo <- y[i]
    wid <- strwidth(words[i],
        cex = cex[i], ...)
    ht <- strheight(words[i],
        cex = cex[i], ...)
    if (grepl(tails, words[i]))
      ht <- ht + ht * 0.2
    if (rotWord) {
      tmp <- ht
      ht <- wid
      wid <- tmp
    }
    isOverlaped <- TRUE
    while (isOverlaped) {
      if (!.overlap(x1 - 0.5 *
          wid, y1 - 0.5 * ht, wid,
        ht, boxes) && x1 - 0.5 *
            wid > xlim[1] && y1
          -
        0.5 * ht > ylim[1] && x1
            + 0.5 * wid < xlim[2]
            &&
        y1 + 0.5 * ht < ylim[2])
          {
        boxes[[length(boxes) +
            1]] <- c(x1 - 0.5 *
            wid,
          y1 - 0.5 * ht, wid, ht)
        isOverlaped <- FALSE
      }
      else {
        theta <- theta + tstep
        r <- r + rstep * tstep/
            (2 * pi)
        x1 <- xo + sdx * r * cos
            (theta)
        y1 <- yo + sdy * r * sin
            (theta)
        iterations <- iterations
            + 1
        if (iterations > 500000){
          boxes[[length(boxes) +
              1]] <- c(x1 - 0.5 *
              wid,
          y1 - 0.5 * ht, wid, ht)
          isOverlaped = FALSE
        }
      }
    }
  }
  print( paste(\"iterations: \",
      iterations) )
  result <- do.call(rbind,
      boxes)
  colnames(result) <- c(\"x\",
      \"y\", \"width\", \"ht\")
  rownames(result) <- words
  result
}
", filename)
insertSource(filename, package="
    wordcloud", force=FALSE)
nc <- wordlayout(
        labcd$x,
        labcd$y,
        word_labs,
        cex=cex * 1.28,
        xlim=c( -1, 1 ),
        ylim=c( -1, 1 )
      )

      xorg <- labcd$x
      yorg <- labcd$y
      labcd$x <- nc[,1] + .5 * nc[,3]
      labcd$y <- nc[,2] + .5 * nc[,4]
      lay_f <- cbind(labcd$x, labcd$
          y)

      if_fixed <- 1
    }
  }

if ( com_method == "cor" ){ # cor
  dr <- as.data.frame( t(dr) )
  dv <- data.frame(
    khvar_ = as.numeric(v0)
  )
  dr <- cbind(dr,dv)

  edge_pos <- NULL
  edges <- get.edgelist(n2, names=
      TRUE)
  for (i in 1:nrow(edges)){
    i1 <- as.numeric( edges[i,1] )
    i2 <- as.numeric( edges[i,2] )

    edge_pos <- c(
      edge_pos,
      cor(
        as.numeric( dr[,i1] > 0 & dr[,
            i2] >0 ),
        dr$khvar_,
        method="pearson"
      )
      #mean( dr[dr[,i1] > 0 & dr[,i2]
          >0,]$khvar_ )
    )
  }

  if ( length( edge_pos[is.na(edge_
      pos) == F] ) == 0 ){
```

```
    edge_pos <- 0
  }

  #n2 <- igraph::set.edge.attribute(
  # n2,
  # "edge_pos_o",
  # 1:length(igraph::get.edge.attribute(
      n2,"weight")),
  # edge_pos
  #)

  #edge_pos <- edge_pos - mean(
      edge_pos)
  #edge_pos <- edge_pos / sd(edge_
      pos)
  ##edge_pos <- edge_pos * 10 + 50

  #limv <- 0.15
  #maxv <- max( abs( edge_pos ) )

  #if ( limv < maxv ){
  # edge_pos[edge_pos > limv] <-
      limv
  # edge_pos[edge_pos < -limv] <- -
      limv
  #}

  n2 <- igraph::set.edge.attribute(
    n2,
    "edge_pos",
    1:length(igraph::get.edge.
        attribute(n2,"weight")),
    edge_pos
  )

  ver_pos <- NULL
  vertices <- as.numeric( igraph::
      get.vertex.attribute(n2,"name")
      )
  for (i in 1:length(vertices) ){
    ver_pos <- c(
      ver_pos,
      cor(
        as.numeric( dr[,vertices[i]] >
            0 ),
        dr$khvar_,
        method="pearson"
      )
    )
  }

  if ( length( ver_pos[is.na(ver_pos)
      == F] ) == 0 ){
    ver_pos <- 0
  }

  ver_pos[ver_pos > max(edge_pos)]
      <- max(edge_pos)
  ver_pos[ver_pos < min(edge_pos)]
      <- min(edge_pos)

  #n2 <- igraph::set.vertex.attribute(
  # n2,
  # "ver_pos",
  # 1:length(igraph::get.vertex.
      attribute(n2,"name")),
  # ver_pos
  #)
  ccol_raw <- ver_pos
  if ( is.null( igraph::get.vertex.
      attribute(n2,"com") ) ==
      FALSE ){
    n2 <- remove.vertex.attribute(
        n2, "com")
    edg_lty <- 1
  }
}

n2 <- igraph::set.vertex.attribute(
  n2,
  "lab",
  1:length(igraph::get.vertex.
      attribute(n2,"name")),
  colnames(d)[ as.numeric( igraph::
      get.vertex.attribute(n2,"name")
      ) ]
)

ver_freq <- freq[ as.numeric( igraph
    ::get.vertex.attribute(n2,"name") )
    ]

if (com_method == "twomode_c" ||
    com_method == "twomode_g"){
  ver_freq[var_select] <- NA
}

if ( is.null(target_ids) == FALSE ){
  ver_freq[target_ids] <- NA
}

if (use_freq_as_size == 0){
  ver_freq[ver_freq > 0] <- 1
}

n2 <- igraph::set.vertex.attribute(
  n2,
  "size",
  1:length(igraph::get.vertex.
      attribute(n2,"name")),
  ver_freq
)

# For community detection

if ( exists("ccol") ){ # clean up
    previous data
  try( n2 <- remove.vertex.
      attribute(n2, "com"), silent=T )
}

if ( exists("com_m") ){
```

```
  com_label <- NULL

  for (h in 1:length(com_m$membership)){
    i <- com_m$membership[h]
    if ( com_m$csize[i] > 1 ) {
      if (i < 10){
        com_label <- c(
          com_label,
          paste("0", as.character(i), " ", sep="")
        )
      } else {
        com_label <- c(com_label, as.character(i))
      }
    } else {
      com_label <- c(com_label, NA)
    }
  }

  n2 <- igraph::set.vertex.attribute(
    n2,
    "com",
    1:length(igraph::get.vertex.attribute(n2,"name")),
    com_label
  )
}

if ( exists("ccol_raw") ){
  n2 <- igraph::set.vertex.attribute(
    n2,
    "com",
    1:length(igraph::get.vertex.attribute(n2,"name")),
    ccol_raw
  )
  com_label <- ccol_raw
}

if ( com_method == "none" || com_method == "twomode_g"){
  n2 <- igraph::set.vertex.attribute(
    n2,
    "com",
    1:length(igraph::get.vertex.attribute(n2,"name")),
    rep( "na", length(igraph::get.vertex.attribute(n2,"name")) )
  )
  com_label <- NA
}

if ( exists("edg_mst") ){
  n2 <- igraph::set.edge.attribute(
    n2,
    "edg_col",
    1:length(igraph::get.edge.attribute(n2,"weight")),
    edg_mst
  )
  #print(edg_mst)
}

if ( exists("edg_lty") == F ){
  edg_lty <- 1
}
edg_lty[edg_lty==1] <- "solid"
edg_lty[edg_lty==3] <- "dotted"

n2 <- igraph::set.edge.attribute(
  n2,
  "line",
  1:length(igraph::get.edge.attribute(n2,"weight")),
  edg_lty
)

library(ggplot2)
library(ggnetwork)

p <- ggplot(
  ggnetwork(n2, layout=lay_f),
  aes(x = x, y = y, xend = xend, yend = yend),
)

if (use_alpha == 1){
  alpha_value = 0.62
  gray_color_n <- "gray20"
} else {
  alpha_value = 1
  gray_color_n <- "gray40"

}

if (text_font == 2){
  face <- "bold"
} else {
  face <- "plain"
}
if (smaller_nodes == 1 ){
  edge_colour <- "gray68"
  nudge <- 0.015
  hjust <- "left"
} else {
  edge_colour <- "gray55"
  nudge <- 0
  hjust <- "center"
}

if (com_method == "twomode_c"){
  edge_colour <- "gray70"
}

if (com_method == "none" || com_method == "twomode_g"){
  edge_colour <- "gray40"
  gray_color_n <- "black"
}
```

```
rownames(lay_f) <- colnames(d)[
    as.numeric( igraph::get.vertex.
    attribute(n2,"name") ) ]
lay_f[,1] <- lay_f[,1] - min(lay_f[,1])
lay_f[,1] <- lay_f[,1] / max(lay_f[,1])
lay_f[,2] <- lay_f[,2] - min(lay_f[,2])
lay_f[,2] <- lay_f[,2] / max(lay_f[,2])
lay_f_df <- data.frame(
  x = lay_f[,1],
  y = lay_f[,2],
  lab = rownames(lay_f)
)

if ( smaller_nodes == 1 ){
  vv <- 6.2
} else {
  vv <- 20
}

sans <- "sans"
if ( exists("saving_eps") ){
  sans <- NULL
}

if (com_method == "cor"){ # cor

  if ( gray_scale == 1) {
    myPalette <- gray( seq(1, 0,
        length.out=100) )
    gray_color_n <- "black"
    if (alpha_value < 0.8) {
      alpha_value <- 0.8
    }
    p <- p + geom_edges(
      color = "black",
      size = 1.5
    )
    p <- p + geom_edges(
      aes(
        color = edge_pos
      ),
      size = 1,
    )
  } else {
    myPalette <- colorRampPalette(
      rev( brewer.pal(9, "RdYlBu") )
    )(100) #Spectral
    p <- p + geom_edges(
      aes(
        color = edge_pos
      ),
      size = 0.6,
    )
  }

  p <- p + scale_color_gradientn(
    colours = myPalette,
    #limits = c( min(edge_pos), limv )
        ,
    #limits = c( 0 - limv, 0 + limv ),
    guide = guide_colourbar(
      title = "Correlation:\n",
      title.theme = element_text(
        family=sans,
        face="bold",
        size=11,
        lineheight=0.4,
        angle=0
      ),
      order = 1,
      #override.aes = list(size=6,
          shape=22),
      label.hjust = 1,
      #reverse = TRUE,
      #ncol=2,
      #keyheight = unit(1.5,"line")
    )
  )
  p <- p + scale_fill_gradientn(
    colours = myPalette,
    guide = FALSE
  )
} else if (min_sp_tree == 1){
  edg_col2 <- p$data$edg_col
  edg_col2[edg_col2=="gray30"] <-
      "MST"
  edg_col2[edg_col2=="gray55"] <-
      "non-MST"
  edg_col2[edg_col2=="gray70"] <-
      "non-MST"
  p <- p + geom_edges(
    aes(linetype = as.character(line),
        alpha=edg_col2),
    #size = 0.8,
    color = "grey10"
  )
  p <- p + scale_alpha_discrete(
    range = c(1, 0.3),
    guide = guide_legend(
      title = "Edge:",
      keyheight = unit(1.2,"line"),
      order = 2
    )
  )
} else if ( use_weight_as_width == 1 )
    {
  p <- p + geom_edges(
    aes(linetype = as.character(line),
        alpha=weight),
    #size = 0.8,
    color = "grey10"
  )
  p <- p + scale_alpha(
    range = c(0.2, 1),
    guide = guide_legend(
      title = "Coefficient:",
      label.hjust = 1,
      keyheight = unit(1.2,"line"),
      order = 2
    )
  )
} else {
```

```
    p <- p + geom_edges(
      aes(linetype = as.character(line)),
      size = 0.4,
      color = edge_colour
    )
  }

  p <- p + scale_linetype_identity()

  alpha_config <- 0
  if (
        ( com_method == "com-b" ||
          com_method == "com-g" ||
          com_method == "com-r" )
    && ( length(com_m$csize[com_m$csize >= 2]) >= 13 )
    && ( length(com_m$csize[com_m$csize >= 2]) <= 20 )
  ){
    alpha_config <- -0.5
    p <- p + geom_nodes(
      aes(
        size = size * 0.41
      ),
      alpha = 0.3, # 0.65
      color = "white",
      show.legend = F,
      shape = 16
    )
    p <- p + geom_nodes(
      aes(
        size = size
      ),
      alpha = 0.65,
      color = "white",
      show.legend = F,
      shape = 16
    )
  }

  p <- p + geom_nodes(
    aes(
      size = size * 0.41,
      color = com
    ),
    alpha = 0.85 + alpha_config,
    show.legend = F,
    shape = 16
  )
  p <- p + geom_nodes(
    aes(
      size = size,
      color = com,
      shape = shape
    ),
    alpha = alpha_value + alpha_config / 3,
    shape = 16
  )
  p <- p + geom_nodes(
    aes(
      size = size,
      shape = shape
    ),
    colour = gray_color_n,
    show.legend = F,
    alpha = alpha_value,
    shape = 1
  )
  p <- p + geom_nodes( # dummy for the legend
    aes( fill = com ),
    size=0,
    colour = gray_color_n,
    alpha = 0,
    shape = 21
  )

  if ( use_freq_as_size == 1 ){
    p <- p + scale_size_area(
      "Frequency",
      max_size = 30 * bubble_size / 100,
      guide = guide_legend(
        title = "Frequency:",
        override.aes = list(colour="black", alpha=1, shape=1),
        label.hjust = 1,
        order = 3
      )
    )
  } else {
    p <- p + scale_size_area(
      max_size = vv,
      guide = F
    )
  }

  if ( (com_method == "twomode_c" || com_method == "twomode_g") ) {
    # (is.null(target_ids) == FALSE)

    if ( com_method == "twomode_c" ){
      var_outline_c <- "gray50"
      var_fill_c <- "#FB8072"
    }
    if ( com_method == "twomode_g" ){
      var_outline_c <- "black"
      var_fill_c <- "white"
    }

    p <- p + geom_point(
      data = data.frame(
        x = lay_f[var_select,1],
        y = lay_f[var_select,2]
      ),
      aes(
        x = x,
        y = y,
        xend = x,
        yend = y
```

```
    ),
    fill = var_fill_c,
    show.legend = F,
    colour = NA,
    alpha = 0.8,
    size = vv * 2 / 3,
    shape = 22
  )

  p <- p + geom_point(
    data = data.frame(
      x = lay_f[var_select,1],
      y = lay_f[var_select,2]
    ),
    aes(
      x = x,
      y = y,
      xend = x,
      yend = y
    ),
    fill = var_fill_c,
    show.legend = F,
    colour = var_outline_c,
    alpha = alpha_value,
    size = vv,
    shape = 22
  )
}

if ( (is.null(target_ids) == FALSE) )
    {
  var_select <- target_ids

  p <- p + geom_point(
    data = data.frame(
      x = lay_f[var_select,1],
      y = lay_f[var_select,2]
    ),
    aes(
      x = x,
      y = y,
      xend = x,
      yend = y,
      fill = com_label[var_select]
    ),
    show.legend = F,
    colour = NA,
    alpha = 0.8,
    size = vv * 2 / 3,
    shape = 22
  )

  p <- p + geom_point(
    data = data.frame(
      x = lay_f[var_select,1],
      y = lay_f[var_select,2]
    ),
    aes(
      x = x,
      y = y,
      xend = x,
      yend = y,
      fill = com_label[var_select]
    ),
    show.legend = F,
    colour = gray_color_n,
    alpha = alpha_value,
    size = vv,
    shape = 22
  )

  p <- p + geom_point(
    data = data.frame(
      x = lay_f[var_select,1],
      y = lay_f[var_select,2]
    ),
    aes(
      x = x,
      y = y,
      xend = x,
      yend = y
    ),
    fill = NA,
    show.legend = F,
    colour = gray_color_n,
    alpha = alpha_value,
    size = vv * 1.4,
    shape = 22
  )
}

if (
  ( com_method == "com-b" || com_
      method == "com-g" || com_
      method == "com-r" || com_
      method == "cor")
  && gray_scale == 1
  && smaller_nodes == 0
){
  p <- p + geom_label(
    data = lay_f_df,
    aes(
      x = x,
      y = y,
      xend = x,
      yend = y,
      label = lab
    ),
    size=4,
    hjust = hjust,
    nudge_x = nudge,
    nudge_y = nudge * 1.25,
    family=font_fam,
    na.rm = T,
    label.size = NA,
    label.padding = unit(0.2, "lines")
        ,
    label.r = unit(0.1, "lines"),
    fontface=face
  )
} else {
  p <- p + geom_text(
```

```
    data = lay_f_df,
    aes(
      x = x,
      y = y,
      xend = x,
      yend = y,
      label = lab
    ),
    size=4,
    hjust = hjust,
    nudge_x = nudge,
    nudge_y = nudge * 1.25,
    family=font_fam,
    na.rm = T,
    fontface=face
  )
}

if (view_coef == 1){
  p <- p + geom_edgetext(
    aes(label = substring( round(
        weight, digits = 2), 2, 4) ),
    color = "#000080",
    fill = NA,
    size=3.5,
  )
}

if ( com_method == "com-b" || com_
    method == "com-g" || com_
    method == "com-r"){
  if ( gray_scale == 1) {
    p <- p + scale_color_grey(
      na.value = "white",
      guide = FALSE
    )
    p <- p + scale_fill_grey(
      na.value = "white",
      guide = guide_legend(
        title = "Community:",
        override.aes = list(size=5.5,
            alpha=1, shape=22),
        keyheight = unit(1,"line"),
        ncol=2,
        order = 1
      )
    )
  } else {
    if ( length(com_m$csize[com_m$
        csize > 1]) <= 12 ){
      p <- p + scale_color_brewer(
        palette = "Set3",
        na.value = "white",
        guide = FALSE
      )
      p <- p + scale_fill_brewer(
        palette = "Set3",
        na.value = "white",
        guide = guide_legend(
          title = "Community:",
          override.aes = list(size=5.5,
              alpha=1, shape=22),
          keyheight = unit(1.25,"line
              "),
          ncol=2,
          order = 1
        )
      )
    } else if (length(com_m$csize[
        com_m$csize > 1]) <= 20) {
      library(ggsci)
      p <- p + scale_color_d3(
        palette = "category20",
        na.value = "white",
        guide = FALSE
      )
      p <- p + scale_fill_d3(
        palette = "category20",
        na.value = "white",
        guide = guide_legend(
          title = "Community:",
          override.aes = list(size=5.5,
              alpha=1, shape=22),
          keyheight = unit(1.25,"line
              "),
          ncol=2,
          order = 1
        )
      )
    } else {
      p <- p + scale_color_hue(
        c = 50,
        l = 85,
        na.value = "white",
        guide = FALSE
      )
      p <- p + scale_fill_hue(
        c = 50,
        l = 85,
        na.value = "white",
        guide = guide_legend(
          title = "Community:",
          override.aes = list(size=5.5,
              alpha=1, shape=22,
              colour="gray45"),
          keyheight = unit(1.25,"line
              "),
          ncol=2,
          order = 1
        )
      )
    }
  }
}

if ( com_method == "cnt-b" || com_
    method == "cnt-d" || com_
    method == "cnt-e"){

  if (gray_scale == 1){
    myPalette <- gray( seq(1, 0.4,
        length.out=100) )
```

```
    } else {
      if (color_universal_design == 0){
        myPalette <- cm.colors(99)
      } else {
        library(RColorBrewer)
        col_seed <- brewer.pal(8,"YlGnBu")[1:6]

        myPalette <- colorRampPalette( col_seed )
        myPalette <- myPalette(99)
      }
    }

    p <- p + scale_color_gradientn(
      colours = myPalette,
      guide = FALSE
    )
    p <- p + scale_fill_gradientn(
      colours = myPalette,
      guide = guide_colourbar(
        title = "Centrality:\n",
        title.theme = element_text(
          family=sans,
          face="bold",
          size=11,
          lineheight=0.4,
          angle=0
        ),
        order = 1,
        #override.aes = list(size=6, shape=22),
        label.hjust = 1,
        #reverse = TRUE,
        #ncol=2,
        #keyheight = unit(1.5,"line")
      )
    )
  }

  if (com_method == "twomode_c"){
    p <- p + scale_color_manual(
      values = brewer.pal(8, "Spectral")[4:8],
      guide = FALSE
    )
    p <- p + scale_fill_manual(
      values = brewer.pal(8, "Spectral")[4:8],
      guide = guide_legend(
        title = "Degree:",
        order = 1,
        override.aes = list(size=5.5, shape=22, alpha=1),
        #label.hjust = "left",
        #reverse = TRUE,
        #ncol=2,
        keyheight = unit(1.2,"line")
      )
    )
  }

  if ( com_method == "none" || com_method == "twomode_g"){
    p <- p + scale_color_manual(
      values = c("white"),
      na.value = "white",
      guide = F
    )
    p <- p + scale_fill_manual(
      values = c("white"),
      na.value = "white",
      guide = F
    )
  }

  p <- p + theme_blank(
    base_family = font_fam
  )

  if (com_method == "cor" && gray_scale == 0){ # cor
    if ( cor_var_darker == 1 ){
      col_backg <- "gray50"
    } else {
      col_backg <- "gray60"
    }
    p <- p + theme(
      panel.background = element_rect(fill = col_backg, colour = NA)
    )
  }

  p <- p + theme(
    legend.title = element_text(family=sans, face="bold", size=11, angle=0),
    legend.text = element_text(face="plain", size=11, angle=0)
  )


  # make a small space between the graph and the legend
  margin <- 0.04
  #if (smaller_nodes == 1){
  # extra <- 0.05
  # p <- p + coord_fixed(ratio=1, xlim=c(0-margin-extra,1+margin+extra), ylim=c(0-margin,1+margin), expand = F )
  #} else {
    extra <- 0.025
    p <- p + coord_fixed(ratio=1, xlim=c(0-margin-extra,1+margin+extra), ylim=c(0-margin,1+margin), expand = F )
  #}

```

```
#p <- p + theme(plot.margin= unit(c(5, 0, 5, 0), "pt"))

g <- ggplotGrob(p)

if ( length( g$grobs[[8]][[1]][[1]] ) > 1) {
  if (
    (com_method == "cnt-b" || com_method == "cnt-d" || com_method == "cnt-e")
    && ( gray_scale == 0 )
  ){
    g$grobs[[8]][[1]][[1]]$grobs[[5]]$gp$col <- "gray45"
    g$grobs[[8]][[1]][[1]]$grobs[[5]]$gp$lwd <- 1.25
  }
  if (
    (com_method == "cnt-b" || com_method == "cnt-d" || com_method == "cnt-e")
    && ( gray_scale == 1 )
  ){
    g$grobs[[8]][[1]][[1]]$grobs[[5]]$gp$col <- "gray30"
    g$grobs[[8]][[1]][[1]]$grobs[[5]]$gp$lwd <- 1.25
  }
  if ( com_method == "cor" && gray_scale == 0){
    g$grobs[[8]][[1]][[1]]$grobs[[5]]$gp$col <- "gray40"
    g$grobs[[8]][[1]][[1]]$grobs[[5]]$gp$lwd <- 1.1
  }
}

library(grid)
library(gtable)

# fixing width of legends to 22%
if ( exists("saving_file") ){
  if ( saving_file == 0){
    target_legend_width <- convertX( unit( image_width * 0.22, "in" ), "mm"
    )
    if ( as.numeric( substr( packageVersion("ggplot2"), 1, 3) ) <= 2.1 ){ # ggplot2 <= 2.1.0
      diff_mm <- diff( c( convertX( g$widths[5], "mm" ), target_legend_width ))
      if ( diff_mm > 0 ){
        print(diff_mm)
        g <- gtable_add_cols(g, unit( diff_mm, "mm"))
      }
    } else { # ggplot2 >= 2.2.0

      diff_mm <- diff( c( convertX( g$widths[7], "mm", valueOnly=T ) + convertX( g$widths[8], "mm", valueOnly=T ), target_legend_width ))
      if ( diff_mm > 0 ){
        print(diff_mm)
        g <- gtable_add_cols(g, unit( diff_mm, "mm"))
      }
    }
  }
}

grid.draw(g)

if (exists("com_m")){
  rm("com_m")
}
if (exists("ccol_raw")){
  rm("ccol_raw")
}
if (exists("edg_mst")){
  rm("edg_mst")
}
if (exists("edg_lty")){
  rm("edg_lty")
}
ccol <- igraph::get.vertex.attribute( n2,"com")
}
```

A. 6　時系列SOMを作成するソースコード

時系列 SOM を作成するソースコード A.6 をしめす.

ソースコード A. 6: som.r

```
d <- NULL
d <- matrix( c(1,…省略…,0), byrow=T, nrow=380, ncol=41 )
d <- d[,-1]
colnames(d) <- c("desk",…省略…,"
```

```
      small")
doc_length_mtr <- matrix( c(
      70,18,…省略…57,19), ncol=2,
      byrow=T)
colnames(doc_length_mtr) <- c("
      length_c", "length_w")
d <- t(d)
# END: DATA

n_nodes <- 20
rlen1 <- 1000
rlen2 <- 200000
d <- t(d)

if (exists("doc_length_mtr")){
  leng <- as.numeric(doc_length_
      mtr[,2])
  leng[leng ==0] <- 1
  d <- d / leng
  d <- d * 1000
}

d <- subset(d, rowSums(d) > 0)
d <- scale(d)
d <- t(d)
d <- t(d)
rownames(d) <- 1:nrow(d)
# SOM
library(som)
somm <- som(
  d,
  n_nodes,
  n_nodes,
  topol="hexa",
  rlen=c(rlen1,rlen2)
)

word_labs <- rownames(d)
n_words <- length(word_labs)

color_universal_design <- 1
# END: DATA

cex <- 1
text_font <- 2
if_cls <- 1
n_cls <- 9
if_plothex <- 1

# n_nodes <- 20
# rlen1 <- 1000
# rlen2 <- 200000

row2coods <- NULL
eve <- 0
for (i in 0:(n_nodes - 1)){
  for (h in 0:(n_nodes - 1)){
    row2coods <- c(row2coods, h +
        eve, i)
  }
  if (eve == 0){
    eve <- 0.5
  } else {
    eve <- 0
  }
}
row2coods <- matrix( row2coods,
    byrow=T, ncol=2 )

if ( if_cls == 1 ){
  library( RColorBrewer )

  if (
      ( as.numeric( R.Version()$
          major ) >= 3 )
    && ( as.numeric( R.Version()$
        minor ) >= 1.0)
  ){ # >= R 3.1.0
    hcl <- hclust( dist(somm$code,
        method="euclidean"),
        method="ward.D2" )
  } else { # <= R 3.0
    hcl <- hclust( dist(somm$code,
        method="euclidean")^2,
        method="ward" )
  }

  colors <- NULL
  if (n_cls <= 9){
    pastel <- brewer.pal(9, "Pastel1
        ")
    # pastel[6] = brewer.pal(9, "
        Pastel1")[9]
    # pastel[9] = brewer.pal(9, "
        Pastel1")[6]
    pastel[6] = "gray91"
    pastel[9] = "#F5F5DC" # FAF3C8
        F7F1C6 EEE8AA F0E68C
    colors <- pastel[cutree(hcl,k=n_
        cls)]
  }
  if (n_cls > 9) {

    library(colorspace)
    new_col <- order( runif(n_cls)
        )
    colors <-
      rainbow_hcl(n_cls, start=20,
          end=340, l=92, c=20)[
      #terrain_hcl(n_cls, c = c(35, 5),
          l = c(85, 95), power = c
          (0.5,1))[
        new_col[cutree(hcl,k=n_cls)]
      ]
  }
} else {
  colors <- rep("gray90", n_nodes
      ^2)
}
labcd <- NULL
plot_mode <- "color"

```

```
104 par(mai=c(0,0,0,0), mar=c(0,0,0,0),
        omi=c(0,0,0,0), oma =c(0,0,0,0) )
105
106 plot(
107   NULL,NULL,
108   xlim=c(0,n_nodes-0.5),
109   ylim=c(0,n_nodes-1),
110   axes=F,
111   frame.plot=F
112 )
113
114 if (if_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
        ^2)
126   if_points <- 1
127   w_lwd <- 1
128   cls_lwd <- 2.25
129   color_cls <- "gray35"
130   color_line <- "gray50"
131   color_pte <- "gray40"
132   color_ptf <- "gray85"
133 }
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   if (max(color_act) == 1){
149     color_act <- color_act * 3 + 1;
150   } else {
151     color_act <- color_act - min(
          color_act)
152     color_act <- round( color_act /
          max(color_act) * 6 ) + 1
153     #color_act[color_act==7] <- 6
154   }
155   color_seed <- brewer.pal(6,"GnBu")
156   #color_seed <- brewer.pal(6,"
        YlOrRd")
157   color_seed <- c("white", color_seed
        )
158   color_act <- color_seed[color_act]
159
160   color_line <- "gray70"
161   if_points <- 0
162   w_lwd <- 1
163   color_cls <- "gray45"
164   color_ptf <- "white"
165 }
166 if ( plot_mode == "umat" ){
167
168
169   dist_u <- NULL
170
171   dist_m <- as.matrix( dist(somm$
        code, method="euclid") )
172
173   for (i in 0:(n_nodes - 1)){
174     for (h in 0:(n_nodes - 1)){
175       cu <- NULL
176       n <- 0
177
178       if (h != n_nodes -1){
179         cu <- c(
180           cu,
181           dist_m[
182             h + i * n_nodes + 1,
183             h + 1 + i * n_nodes + 1
184           ]
185         )
186       }
187
188       if (h != 0){
189         cu <- c(
190           cu,
191           dist_m[
192             h + i * n_nodes + 1,
193             h - 1 + i * n_nodes + 1
194           ]
195         )
196       }
197
198       if (i != n_nodes - 1){
199         if (h %% 2 == 0){
200           cu <- c(
201             cu,
202             dist_m[
203               h + i * n_nodes + 1,
204               h + ( i + 1 ) * n_nodes
                    + 1
205             ]
206           )
207         } else {
208           if (h != n_nodes -1){
209             cu <- c(
210               cu,
211               dist_m[
212                 h + i * n_nodes + 1,
213                 h + 1 + ( i + 1 ) * n_
                      nodes + 1
214               ]
215             )
```

```
      }
    }
  }

  if (i != 0){
    if (h %% 2 == 0){
      cu <- c(
        cu,
        dist_m[
          h + i * n_nodes + 1,
          h + ( i - 1 ) * n_nodes
              + 1
        ]
      )
    } else {
      if (h != n_nodes -1){
        cu <- c(
          cu,
          dist_m[
            h + i * n_nodes + 1,
            h + 1 + ( i - 1 ) * n_
                nodes + 1
          ]
        )
      }
    }
  }

  if (i != n_nodes - 1){
    if (h %% 2 == 0){
      if (h != 0){
        cu <- c(
          cu,
          dist_m[
            h + i * n_nodes + 1,
            h - 1 + ( i + 1 ) * n_
                nodes + 1
          ]
        )
      }
    } else {
      cu <- c(
        cu,
        dist_m[
          h + i * n_nodes + 1,
          h + ( i + 1 ) * n_nodes
              + 1
        ]
      )
    }
  }

  if (i != 0){
    if (h %% 2 == 0){
      if (h != 0){
        cu <- c(
          cu,
          dist_m[
            h + i * n_nodes + 1,
            h - 1 + ( i - 1 ) * n_
                nodes + 1
          ]
        )
      }
    } else {
      cu <- c(
        cu,
        dist_m[
          h + i * n_nodes + 1,
          h + ( i - 1 ) * n_nodes
              + 1
        ]
      )
    }
    dist_u <- c(dist_u, median(cu)
        )
  }
}

print( summary(dist_u) )

dist_u <- dist_u - min(dist_u)
dist_u <- round( dist_u / max(
    dist_u) * 100 ) + 1

if (color_universal_design == 0){
  color_act <- cm.colors(101)[dist
      _u]
  color_line <- "gray70"
  color_cls <- "gray45"
} else {
  library(RColorBrewer)
  if (T){
    col_seed <- brewer.pal(9, "
        GnBu")
    myPalette <-
        colorRampPalette( col_seed
        )
    color_act <- myPalette(101)[
        dist_u]
    color_act <- adjustcolor(color_
        act, alpha=0.8)
    color_line <- "white"
    color_cls <- "gray30"
  } else {
    col_seed <- rev(brewer.pal(9,
        "RdYlBu"))
    myPalette <-
        colorRampPalette( col_seed
        )
    color_act <- myPalette(101)[
        dist_u]
    color_act <- adjustcolor(color_
        act, alpha=0.8)
    color_line <- "gray50"
    color_cls <- "gray25"
  }
}

#color_line <- "gray70"
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,
              x - 0.5, x ),
330       y=c( y + a, y - a, y - b, y - a,
              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]
              == colors[h + i * n_nodes +
              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
                  + 1]) == 1
413         || is.na(colors[h + chk + (i+1)
                * 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
                _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,
```

```
        )
      }
    }
  }

for (i in 0:(n_nodes - 2)){
  for (h in 0:(n_nodes - 1)){
    if (i %% 2 == 0){
      chk <- 1
    } else {
      chk <- 0
    }
    if (
        is.na(colors[h + i * n_nodes
            + 1]) == 1
      || is.na(colors[h - chk + (i+1)
          * n_nodes + 1]) == 1
      || h - chk < 0
    ){
      next
    }

    if (
        colors[h + i * n_nodes + 1]
    == colors[h - chk + (i+1) * n
        _nodes + 1]
    ){
      x <- h
      y <- i
      if ( y %% 2 == 1 ){
        x <- x + 0.5
      }

      segments(
        x, y + b,
        x - 0.5, y + a,
        col=color_line,
        lwd=w_lwd,
      )
    }
  }
}



for (i in 0:(n_nodes - 1)){
  for (h in 0:(n_nodes - 2)){
    if ( colors[h + i * n_nodes + 1] !
        = colors[h + i * n_nodes + 2]
        ){
      x <- h
      y <- i
      if ( y %% 2 == 1 ){
        x <- x + 0.5
      }

      segments(
        x + 0.5, y + a,
        x + 0.5, y - a,
        col=color_cls,
        lwd=cls_lwd,
        )
      }
    }
  }
}

for (i in 0:(n_nodes - 2)){
  for (h in 0:(n_nodes - 1)){
    if (i %% 2 == 1){
      chk <- 1
    } else {
      chk <- 0
    }

    if (
        is.na(colors[h + i * n_nodes
            + 1]) == 1
      || is.na(colors[h + chk + (i+1)
          * n_nodes + 1]) == 1
      || h + chk == n_nodes
    ){
      next
    }

    if (
        colors[h + i * n_nodes + 1]
    != colors[h + chk + (i+1) * n_
            nodes + 1]
    ){
      x <- h
      y <- i
      if ( y %% 2 == 1 ){
        x <- x + 0.5
      }

      segments(
        x, y + b,
        x + 0.5, y + a,
        col=color_cls,
        lwd=cls_lwd,
      )
    }
  }
}

for (i in 0:(n_nodes - 2)){
  for (h in 0:(n_nodes - 1)){
    if (i %% 2 == 0){
      chk <- 1
    } else {
      chk <- 0
    }

    if (
        is.na(colors[h + i * n_nodes
            + 1]) == 1
      || is.na(colors[h - chk + (i+1)
          * n_nodes + 1]) == 1
      || h - chk < 0
    ){
      next
    }
```

```
546
547       if (
548             colors[h + i * n_nodes + 1]
549         != colors[h - chk + (i+1) * n_
                nodes + 1]
550       ){
551         x <- h
552         y <- i
553         if ( y %% 2 == 1 ){
554           x <- x + 0.5
555         }
556
557         segments(
558           x, y + b,
559           x - 0.5, y + a,
560           col=color_cls,
561           lwd=cls_lwd,
562         )
563       }
564     }
565   }
566
567   points <- NULL
568   sf <- 0.35
569   a <- a * sf;
570   b <- b * sf;
571   c <- 0.5 * sf;
572   for (i in 1:nrow(somm$visual)){
573     x <- somm$visual[i,1]
574     y <- somm$visual[i,2]
575     if ( y %% 2 == 1 ){
576       x <- x + 0.5
577     }
578     points <- c(points, x, y)
579   }
580   points <- matrix( points, byrow=
        T, ncol=2 )
581
582   if( if_points == 1 ){
583     if (F){
584       for (i in 1:nrow(points)){
585         x <- points[i,1]
586         y <- points[i,2]
587
588         polygon(
589           x=c( x + c, x + c, x, x - c, x
                  - c, x ),
590           y=c( y + a, y - a, y - b, y -
                  a, y + a, y + b ),
591           col=color_ptf,
592           border=color_pte,
593           lty=1,
594         )
595       }
596     } else {
597       symbols(
598         points[,1],
599         points[,2],
600         squares=rep(0.35,length(
              points[,1])),
601         #circles=rep(0.2,length(points
                [,1])),
602         fg="gray70",
603         bg=color_ptf,
604         inches=F,
605         add=T,
606       )
607     }
608   }
609
610   library(maptools)
611   if (is.null(labcd) == 1){
612     labcd <- pointLabel(
613       x=points[,1],
614       y=points[,2],
615       labels=word_labs,
616       doPlot=F,
617       cex=cex,
618       offset=0
619     )
620
621
622     xorg <- points[,1]
623     yorg <- points[,2]
624     #cex <- 1
625
626     if ( length(xorg) < 300 ) {
627       library(wordcloud)
628
629   # fix for "wordlayout" function
630   filename <- tempfile()
631   writeLines("wordlayout <-
        function (x, y, words, cex =
        1, rotate90 = FALSE, xlim = c
        (-Inf,
632     Inf), ylim = c(-Inf, Inf),
          tstep = 0.1, rstep = 0.1,
          ...)
633   {
634     tails <- \"g|j|p|q|y\"
635     n <- length(words)
636     sdx <- sd(x, na.rm = TRUE)
637     sdy <- sd(y, na.rm = TRUE)
638     iterations <- 0
639     if (sdx == 0)
640       sdx <- 1
641     if (sdy == 0)
642       sdy <- 1
643     if (length(cex) == 1)
644       cex <- rep(cex, n)
645     if (length(rotate90) == 1)
646       rotate90 <- rep(rotate90, n)
647     boxes <- list()
648     for (i in 1:length(words)) {
649       rotWord <- rotate90[i]
650       r <- 0
651       theta <- runif(1, 0, 2 * pi)
652       x1 <- xo <- x[i]
653       y1 <- yo <- y[i]
654       wid <- strwidth(words[i],
            cex = cex[i], ...)
655       ht <- strheight(words[i],
```

```
          cex = cex[i], ...)
    if (grepl(tails, words[i]))
      ht <- ht + ht * 0.2
    if (rotWord) {
      tmp <- ht
      ht <- wid
      wid <- tmp
    }
    isOverlaped <- TRUE
    while (isOverlaped) {
      if (!.overlap(x1 - 0.5 *
          wid, y1 - 0.5 * ht, wid,
        ht, boxes) && x1 - 0.5 *
            wid > xlim[1] && y1
          -
        0.5 * ht > ylim[1] && x1
            + 0.5 * wid < xlim[2]
            &&
        y1 + 0.5 * ht < ylim[2])
          {
        boxes[[length(boxes) +
          1]] <- c(x1 - 0.5 *
          wid,
          y1 - 0.5 * ht, wid, ht)
        isOverlaped <- FALSE
      }
      else {
        theta <- theta + tstep
        r <- r + rstep * tstep/
          (2 * pi)
        x1 <- xo + sdx * r * cos
          (theta)
        y1 <- yo + sdy * r * sin
          (theta)
        iterations <- iterations
            + 1
        if (iterations > 500000){
          boxes[[length(boxes) +
              1]] <- c(x1 - 0.5 *
              wid,
          y1 - 0.5 * ht, wid, ht)
          isOverlaped = FALSE
        }
      }
    }
  }
  print( paste(\"iterations: \",
      iterations) )
  result <- do.call(rbind,
      boxes)
  colnames(result) <- c(\"x\",
      \"y\", \"width\", \"ht\")
  rownames(result) <- words
  result
}
", filename)
insertSource(filename, package="
    wordcloud", force=FALSE)
nc <- wordlayout(
      labcd$x,
      labcd$y,
      word_labs,
      cex=cex * 1.25,
      xlim=c( par( "usr" )[1], par( "
          usr" )[2] ),
      ylim=c( par( "usr" )[3], par( "
          usr" )[4] )
    )

    xlen <- par("usr")[2] - par("
        usr")[1]
    ylen <- par("usr")[4] - par("
        usr")[3]

    segs <- NULL
    for (i in 1:length(word_labs) ){
      x <- ( nc[i,1] + .5 * nc[i,3] -
          labcd$x[i] ) / xlen
      y <- ( nc[i,2] + .5 * nc[i,4] -
          labcd$y[i] ) / ylen
      dst <- sqrt( x^2 + y^2 )
      if ( dst > 0.05 ){
        segs <- rbind(
          segs,
          c(
            nc[i,1] + .5 * nc[i,3], nc[i
                ,2] + .5 * nc[i,4],
            xorg[i], yorg[i]
          )
        )
      }
    }

    xorg <- labcd$x
    yorg <- labcd$y
    labcd$x <- nc[,1] + .5 * nc[,3]
    labcd$y <- nc[,2] + .5 * nc[,4]
  }


}

if ( exists("segs") ){
  if ( is.null(segs) == F){
    for (i in 1:nrow(segs) ){
      segments(
        segs[i,1],segs[i,2],segs[i,3],segs[i
            ,4],
        col="gray60",
        lwd=1
      )
    }
  }
}

text(
  labcd$x,
  labcd$y,
  labels=word_labs,
  cex=cex,
  offset=0,
  font=text_font
```

```
)

if ( exists("out_coord") == F ) {
  out_coord <- cbind(
    labcd$x / (n_nodes-0.5),
    labcd$y / (n_nodes-1)
  )
}

points1<-head(points,n=190)
par(new=T)
plot(points1[,1],points1[,2],type="c",
    col="red")

points2<-tail(points,n=190)
par(new=T)
plot(points2[,1],points2[,2],type="c",
    col="blue")
```