加古川市公用車データの分析報告書

同志社大学文化情報学部統計科学研究室[1]

[1]八島裕史，柚木慎太郎，浦上新太，大谷諒，岡部格明，宿久洋

目 次

第1章　はじめに

本稿は，同志社大学文化情報学部統計科学研究室が行なった，加古川市における公用車の走行データに関する分析をまとめた報告書である．今回提供されたデータは，株式会社ゼンリンデータコムによって取得された専用装置やスマートフォンといったセンサー端末から収集された車両の位置情報，車両の加速度を含むものである．

石井 (2018) や NEC(2018) によると，近年，社会インフラの老朽化が問題となっており，その解決策の一つとしてディープラーニングを活用した道路のひび割れ検出を行なっている．しかし，ディープラーニングを活用したモデルを用いるためには大量の画像データとそのラベルが必要であるため，モデル作成が容易ではなく，一般に活用を行うことは難しいと考えられる．今回の分析においては，公用車にスマートフォンなどの安価な端末を設置した上で集計を行うことで，ディープラーニングを用いたひび割れ検出より安価な道路状況の異常検出を行うことが可能になることが期待される．

一方で，車の運転と事故に関して，事故につながる運転の検出には，車の加速度に対して閾値を設けることにより危険度合いを定義する方法が用いられる．

西堀ら (2010) によると，自動車の走行時における加速度と，交通事故の発生状況に関係があることを指摘している．西堀ら (2010) では，加速度について閾値を設定し，その閾値を超えると，急加速，急減速が起こっていると定義しており，走行時における，その発生頻度が増えると，人身事故の件数も増えることを報告している．具体的には，前後加速度，左右加速度に対して，その絶対値が 0.3G を上回る挙動を「ヒヤリハット」と定義した場合,「ヒヤリハット」が多く発生している箇所では事故も多く発生していることを報告している．Klauer et al.(2009) では，左右加速度が 0.3G を超える運転は，そうでない運転よりも事故との関連性が大きくなることが報告されている．

また，横関 (2012) によると，事故例データを用いた分析では，速度が高いと雨量が少なくても事故になり，雨量が多くなると速度が低くとも事故が発生する傾向があることを報告している．横関 (2012) にて示されている図 1.1 は，高速自動車国道における運転操作ミスの事故割合を示している．横軸の危険認知速度とは相手車両などの危険を認知して，ブレーキやハンドル操作などの危険回避措置をとる直前までの走行速度を示す．図より，雨天時の運転操作ミスの割合は晴天時・曇天時に比べると高いことが分かる．これらのことから，天候が運転に影響を与えることが分かった．

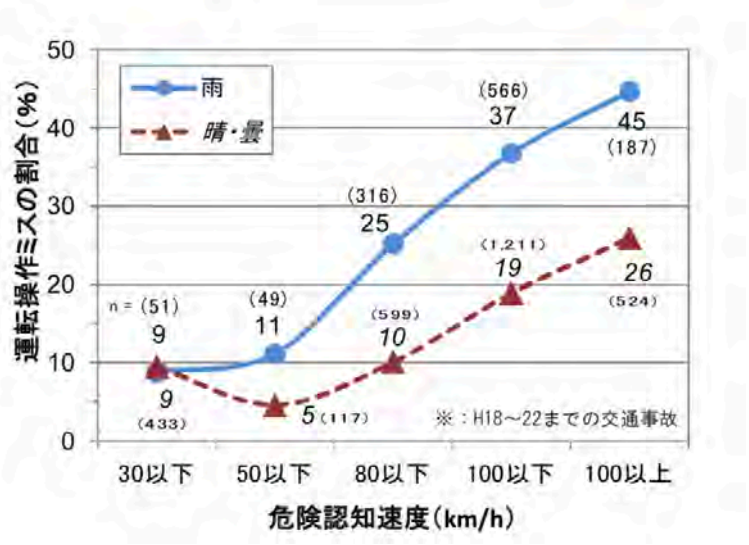

図 1.1: 高速自動車国道における運転操作ミスの事故割合 (横関ほか (2012)) より引用

以上の背景より，事故の原因となる危険な運転は急減速と急ハンドルであることがわかる．今回の分析において，加古川市の車両運転の情報から与えられたデータの加速度の値を利用して，加古川市の自動車走行時の運転状況を調べていく．

本稿では道路状況についての分析を主眼におき，この分析に関する視点として，道路が劣化している箇所，天候を考慮した危険な運転が起こりうる箇所に着目する．道路が劣化している箇所の分析に関しては，各時点での加速度における分散を利用し，「道路状態が悪いところでは車体が揺れる，つまり加速度の分散が大きくなる」という考えを基に行なっている．天候を考慮した危険な運転が起こりうる箇所の分析に関しては，横関 (2012) における加速度閾値を用いており，晴天時と雨天時では加速度に関する危険度合いが違うという考えを元に分析を行っている．

今回このデータを分析する意義として，市町村によるデータ分析の促進と加速度センサおよび計測用スマホアプリの利活用促進があげられる．また，本稿による分析結果を用いることで，老朽化した道路や天候により危険度が変化する恐れがある道路を抽出し，より安全なまちづくりに活用することが期待される．

この報告書は，第 1 章から第 6 章の計 6 章と 1 章の付録により構成されている．

第 2 章では，本稿の分析で使用した，加古川市提供のデータについて述べる．加古川市より提供された，加古川市車両台帳，obd2 データ，Phone データの詳細について説明する．

第 3 章では，分析を行う場合に必要な前準備について述べる．主な内容は，欠損値除外などの前処理，および本分析の指針となる基礎分析である．

第 4 章では，先行研究を基に実施した，データの本分析について述べる．道路状況の良くない場所の検出法，地図データを用いた可視化の方法についてをこの章で述べる．

第 5 章では，4 章までで行った分析結果より深く分析を行うにあたりその基礎となる分析の結果に関して述べる．

第 6 章では，本稿のまとめと今後の課題について述べる．

付録第 1 章では，3 章と 4 章，および 5 章で用いたデータの分析についてのソースコードを提示する．

第2章　データの説明

本章では，加古川市より提供されたデータについて説明する．提供されたデータは次の 3 種類のデータである．これらのデータについてその概要と変数について説明する．

- 加古川市車両台帳
- obd2 データ
- Phone データ

2.1　加古川市車両台帳

加古川市車両台帳の例を図 2.1 に示し，加古川市車両台帳に含まれる変数を表 2.1 に示す．

2.2　obd2 データ

obd2 データの例を図 2.2 に示し，また，obd2 データに含まれる主な変数を表 2.2 に示す．obd2 データの加速度変数は符号付の整数型の 2 の補数によって記録されており，(2.1) 式による変換によって加速度 aG[1]に変換される．

緯度経度の異常値には “4295.6121580000”が記録されている．これは GPS 測位状態に対応しており，測位状態の値が “255”であるとき，緯度経度は “4295.6121580000”となっている．

$$a = \begin{cases} \frac{x}{1024} & (x < 2^{-13}) \\ -\frac{2^{14}-x}{1024} & (x \geq 2^{-13}) \end{cases} \tag{2.1}$$

図 2.1: 加古川市車両台帳の例

[1]1G は 9.80665(m/s^2)

表 2.1: 加古川市車両台帳の説明

列名	内容
No.	各車両ごとの ID
取付状況	全て「取付済」
取付日	センサー取り付け日
端末種別	OBD II もしくは スマホ
SIM 番号	それぞれ UNIID に対応
UNID（アプリ ID）※要加工	unit_id(obd2 データ), user_id(Phone データ) との紐付け
所属課名	車両の所属課
スマホ機種	Phone データの取得端末
用途	公用車の用途（連絡用/貨物など）
車種	普通車，貨物車など
車名	製造メーカー
車体の形状	バン，乗用など
燃料	ガソリン，軽油など
定員	車両の定員
最大積載量	貨物車の最大積載量
車両重量	車両自体の重量
車両総重量	車両の重量と積載可能な最大の重量を足した重量
長さ	車両の長さ
幅	車両の幅
高さ	車両の高さ
排気量	車両の排気量
保管場所	車両の保管場所

2.3 Phone データ

Phone データの例を図 2.3 に示し，また，Phone データに含まれる変数を表 2.3 に示す．

データ中の加速度は，加速度の単位 G で記録されている．緯度経度の異常値は “0.0000000000” が記録されている．

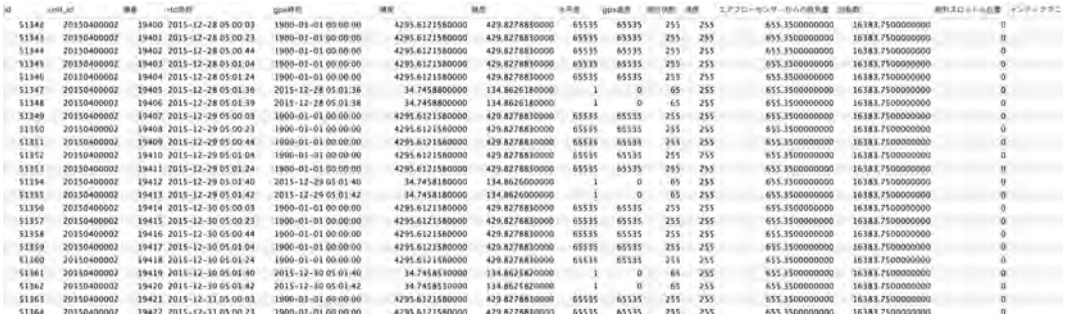

図 2.2: obd2 データの例

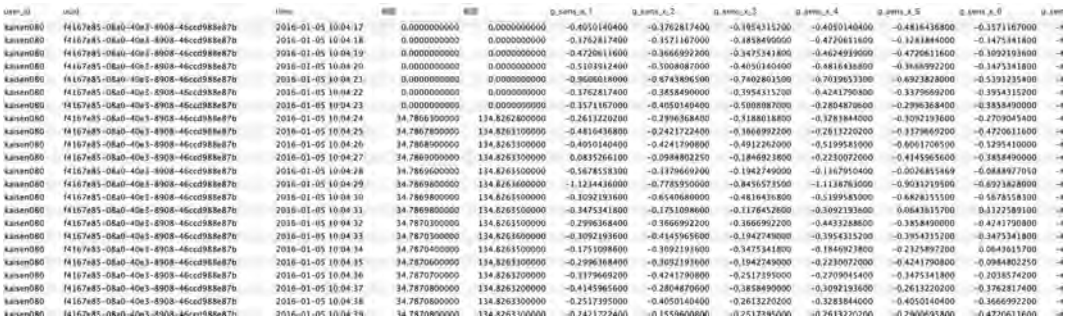

図 2.3: phone データの例

表 2.2: obd2 データの主な変数の説明

列名	内容
id	
unit_id	obd2 端末の ID(各車につき 1 つの ID が付与)
rtc 時刻	データが取得された RTC 時刻
gps 時刻	データが取得された GPS 時刻 (通常は RTC 時刻と対応)
緯度	データ取得時点の車両の緯度
経度	データ取得時点の車両の経度
水平度	（0 または 1）
gps 速度	gps によって測定された速度
測位状態	GPS の測位状態
速度	車両速度
回転数	エンジンの回転数
故障診断	(0 もしくは 255)
e_comp_x	e コンパスセンサー x 軸の値
e_comp_y	e コンパスセンサー y 軸の値
e_comp_z	e コンパスセンサー z 軸の値
プロセス状態	（1 から 5）
g_sens_x_1〜100	1/100 秒ごとの加速度センサー x 軸の値
g_sens_y_1〜100	1/100 秒ごとの加速度センサー y 軸の値
g_sens_z_1〜100	1/100 秒ごとの加速度センサー z 軸の値

表 2.3: Phone データの説明

列名	内容
user_id	回線番号
uuid	Universally Unique Identifier (回線番号に対応した id が付与)
time	データが取得された RTC 時刻
緯度	データ取得時点の車両の緯度
経度	データ取得時点の車両の経度
g_sens_x_1～100	1/100 秒ごとの加速度センサー x 軸の値
g_sens_y_1～100	1/100 秒ごとの加速度センサー y 軸の値
g_sens_z_1～100	1/100 秒ごとの加速度センサー z 軸の値

第3章　分析前の準備

本章では，分析前の準備として，今回の分析に用いた車両の詳細について説明し，それらに取り付けた，センサー端末から得られたデータに対する基礎集計の結果を提示する．また，データに含まれる異常値の処理方法についても説明する．

3.1　今回の分析対象車の詳細

計算量，および天候による比較の都合上，今回は特定の車両に絞り分析を行った．本節ではその車両について説明する．

実際に分析に用いた車両一覧について，表 3.1 に示す．

表 3.1: 本稿で分析に用いた車両

車両の UNID	車両台帳の No
kaisen080	108
kaisen082	114
kaisen084	116

3.1.1　kaisen080

user_id が kaisen080 である車両について，その概要を以下の表 3.3 に示す．また，車両台帳よりわかる kaisen080 の車両の外観・内装についてを図 3.1，図 3.2，図 3.3，図 3.4 に示す[1]．

[1] http://catalog.carsensorlab.net/nissan/vanette/f002/m004/g005/ より引用 (2019 年 3 月 21 日アクセス)

表 3.3: kaisen080 についての加古川市車両台帳の説明

項目	内容
取付状況	取付済
取付日	2015/11/2
端末種別	スマホ
SIM 番号	回線 80
UNID（アプリ ID）※要加工	kaisen080
所管 C	400
符号 C	394
ナンバー	姫路 400 せ 7662
所属コード	320400
所属課名	社会教育・スポーツ振興課
住所	加古川市米田町平津 384-2
スマホ機種	LG G2mini
用途	加古川西公民館
初年度	16
登録 HMD	160517
車種	小型貨物
車名	日産
取得 H	160520
会計 C	1
会計	一般会計
車台番号	SK82VN-319671
車体の形状	バン
型式	TC-SK82VN
原動機の型式	F8
燃料	ガソリン
定員	39513
最大積載量	700
車両重量	1340
車両総重量	2255
長さ	428
幅	163
高さ	185
排気量	1780
購入年度	H16 年度購入
コード 1	1
保管場所	施設

図 3.1: 日産バネットの外観 1

図 3.2: 日産バネットの外観 2

図 3.3: 日産バネットの内装 1

図 3.4: 日産バネットの内装 2

3.1.2 kaisen082

user_id が kaisen082 である車両について，その概要を以下の表 3.5 に示す．また，車両台帳よりわかる kaisen082 の車両の外観・内装についても，図 3.5，図 3.6，図 3.7，図 3.8 に示す[2]．

[2] `http://asahi-jidosya.com/car/905.html` より引用 (2019 年 3 月 19 日アクセス)

表 3.5: kaisen082 についての加古川市車両台帳の説明

項目	内容
取付状況	取付済
取付日	2015/11/2
端末種別	スマホ
SIM 番号	回線 82
UNID（アプリ ID）※要加工	kaisen082
所管 C	300
符号 C	42
ナンバー	姫路 100 さ 4122
所属コード	70600
所属課名	リサイクルセンター
住所	加古川市平荘町磐 1146
スマホ機種	LG G2mini
用途	貸物
初年度	14
登録 HMD	140702
車種	普通貨物
車名	三菱
取得 H	140702
会計 C	1
会計	一般会計
車台番号	FK71HE760025
車体の形状	脱着装置付コンテナ専用車
型式	KK-FK71HE
原動機の型式	6M61
燃料	軽油
定員	3
最大積載量	4000
車両重量	3180
車両総重量	7975
長さ	585
幅	217
高さ	243
排気量	8200
購入年度	H14 年度購入
コード 1	1
保管場所	施設

図 3.5: 三菱 ファイター 4t 脱着装置付コンテナ専用車の外観 1

図 3.6: 三菱 ファイター 4t 脱着装置付コンテナ専用車の外観 2

図 3.7: 三菱 ファイター 4t 脱着装置付コンテナ専用車の内装 1

図 3.8: 三菱 ファイター 4t 脱着装置付コンテナ専用車の内装 2

3.1.3 kaisen084

同じく user_id が kaisen084 である車両について，その概要を以下の表 3.7 に示す．また，車両台帳よりわかる kaisen084 の車両の外観・内装についても，図 3.9，図 3.10，図 3.11，図 3.12 に示す[3]．

[3] http://www.tau-trade.com/ja/gen/DetailCarUsed.do?stk_no=CECEC8CCCCC9CCC8&cnt=e8c440f6a7c91d3e&stkNoTitleFlg=12d31a10dd18b239d7d2518df123c8d4 より引用 (2019 年 3 月 19 日アクセス)

表 3.7: kaisen084 についての加古川市車両台帳の説明

項目	内容
取付状況	取付済
取付日	2015/11/2
端末種別	スマホ
SIM 番号	回線 84
UNID（アプリ ID）※要加工	kaisen084
所管 C	300
符号 C	46
ナンバー	姫路 800 は 363
所属コード	70600
所属課名	リサイクルセンター
住所	加古川市平荘町磐 1146
スマホ機種	LG G2mini
用途	塵芥車
初年度	15
登録 HMD	150828
車種	塵芥車
車名	日産
取得 H	150828
会計 C	1
会計	一般会計
車台番号	CK53A30014
車体の形状	塵芥車
型式	KL-CK53A
原動機の型式	RG8
燃料	軽油
定員	2
最大積載量	5000
車両重量	10800
車両総重量	15910
長さ	812
幅	249
高さ	308
排気量	17990
購入年度	H15 年度購入
コード 1	1
保管場所	施設

図 3.9: 日産ディーゼルの外観 1

図 3.10: 日産ディーゼルの外観 2

図 3.11: 日産ディーゼルの内装 1

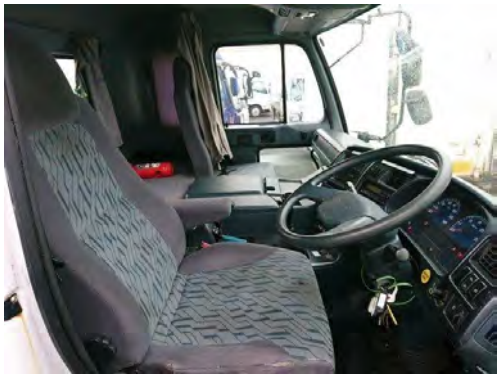

図 3.12: 日産ディーゼルの内装 2

3.2 異常値，欠損値を取り除くための前処理

3.2.1 Phone データ

Phone データにおいて，緯度と経度が「0」になっている異常なデータが複数存在した．これは，端末の異常，または，トンネル中や山奥など GPS が測位不可能になったことが一つの原因と考えられる．これらのデータについて，地図上でプロットする際に不都合が生じるため，緯度と経度が「0」となっている行を削除する処置を施した．欠損として削除した行については，次に示す各車両の走行時間とともに表している．

3.3 分析対象車の対象日付ごとの走行時間

今回分析の対象とした各車両について，各日付ごとの走行時間をそれぞれ示す．

3.3.1 kaisen080

kaisen080 の各日付ごとの走行時間について，以下の表 3.9 に示す．

表 3.9: kaisen080 の各日付ごとの走行時間

日付	走行時間	用いた分析
2016/1/5	15 分 40 秒（42 秒欠損）	4.2 節
2016/1/6	8 分 30 秒	4.2 節
2016/1/7	13 分 30 秒	4.2 節
2016/1/26	16 分 30 秒	4.2 節
2016/1/28	27 分	4.2 節
2016/1/29	31 分 30 秒	4.2 節
2016/1/30	21 分 17 秒（15 分 13 秒欠損）	4.2 節
2016/2/16	15 分 30 秒	4.2 節
2016/2/23	1 分 10 秒	4.2 節

3.3.2 kaisen082

kaisen082 の各日付ごとの走行時間について，以下の表 3.11 に示す．

表 3.11: kaisen082 の各日付ごとの走行時間

日付	走行時間	用いた分析
2016/1/15	1 時間 27 分 30 秒	5.1 節
2016/1/21	49 分 13 秒	5.1 節
2016/1/22	30 分	5.1 節

3.3.3 kaisen084

kaisen084 の各日付ごとの走行時間について，以下の表 3.13 に示す．

表 3.13: kaisen084 の各日付ごとの走行時間

日付	走行時間	用いた分析
2016/3/23	2 時間 42 分 36 秒	4.3 節 (天気：晴れ)
2016/6/23	2 時間 48 分 54 秒	4.3 節 (天気：雨)

3.4 センサーデータの基礎集計

次に，センサーデータがどのような特徴を持っているのかを調べるために，各車両について時系列に基づく挙動を図示した．表 2.3 に示すとおり，Phone データにおける加速度センサーデータの値自体は 1/100 単位であるが，これらは 1 つの変数として横持ちのデータとして含まれている．そ

のため，これを時系列でプロットできるように縦持ちのデータに変換する処理を行った後，折れ線グラフを描画した．

3.4.1 kaisen080

計算量の都合上，一部のデータを用いてデータ概要の把握を行うため，ここでは kaisen080 の 2016 年 2 月 16 日における，計 15 分 30 秒の走行データを対象とし，集計を行った．その結果が図 3.13，図 3.14，図 3.15，図 3.16 である．

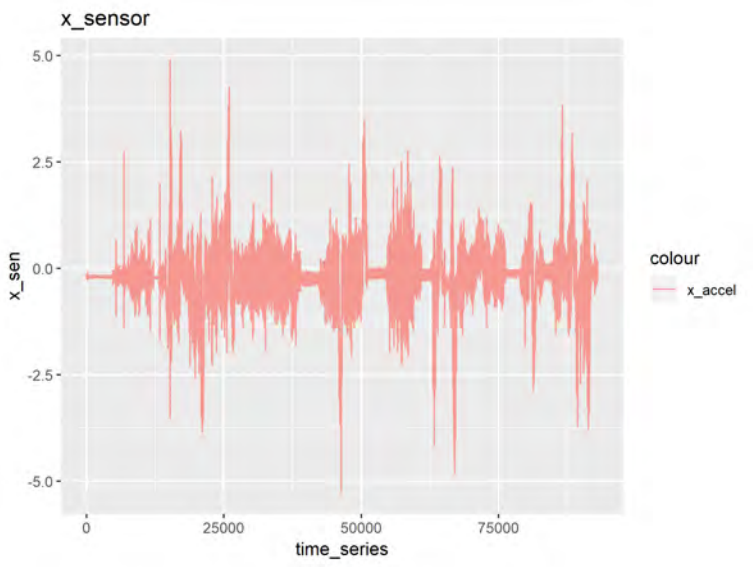

図 3.13: x 軸加速度センサーの折れ線グラフ (kaisen080)

図 3.13 から，正負の値が散在しており，0 を中心に値が揺れていることが分かる．

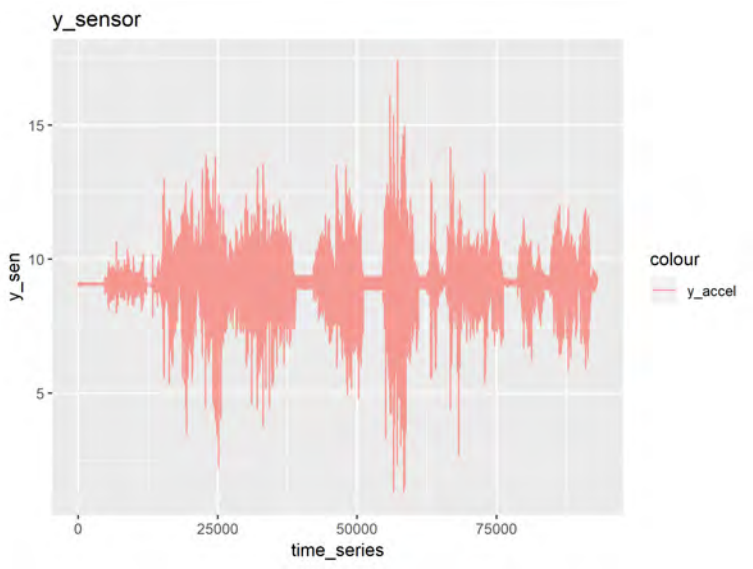

図 3.14: y 軸加速度センサーの折れ線グラフ (kaisen080)

図 3.14 から，全ての値において，正の値をとっていることが分かる．

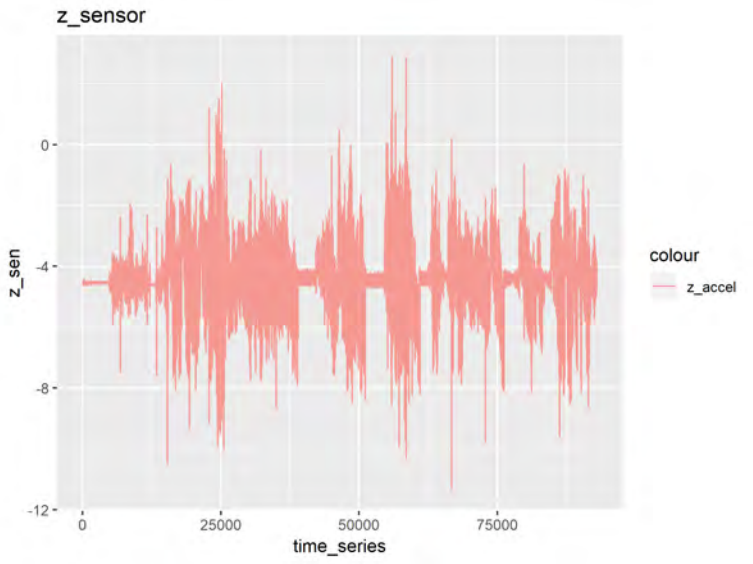

図 3.15: z 軸加速度センサーの折れ線グラフ (kaisen080)

図 3.15 から，基本的に負の値をとっていることが分かる．

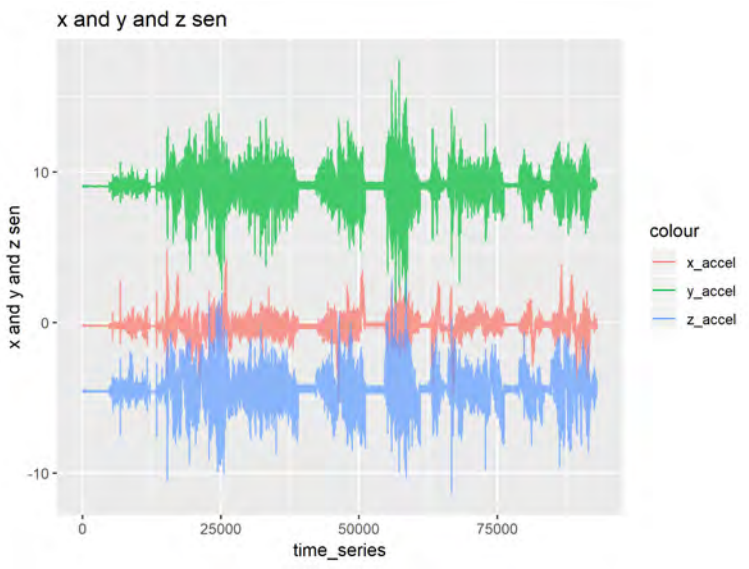

図 3.16: x 軸，y 軸，z 軸の加速度センサーの折れ線グラフ (kaisen080)

図 3.16 から，それぞれのセンサーで同じような傾向を示していることが分かる．

図 3.16 を見ると，z 軸センサーについて，折れ線グラフの振れ幅が一番大きい様子が見て取れる．実際に散らばり度合いを示す分散値を計算してみると，x 軸センサーが約 0.3，y 軸センサーが約 0.49，z 軸センサーが約 0.7 となっており，数値からも z 軸センサーの散らばり度合いが大きいことが分かる．

3.4.2 kaisen082

kaisen080 と同じく，kaisen082 のセンサーデータがどのような特徴を持っているのかを調べるため，ここでは 2016 年 1 月 15 日のデータを対象とし，集計を行った．その結果が図 3.17，図 3.18，図 3.19，図 3.20 である．

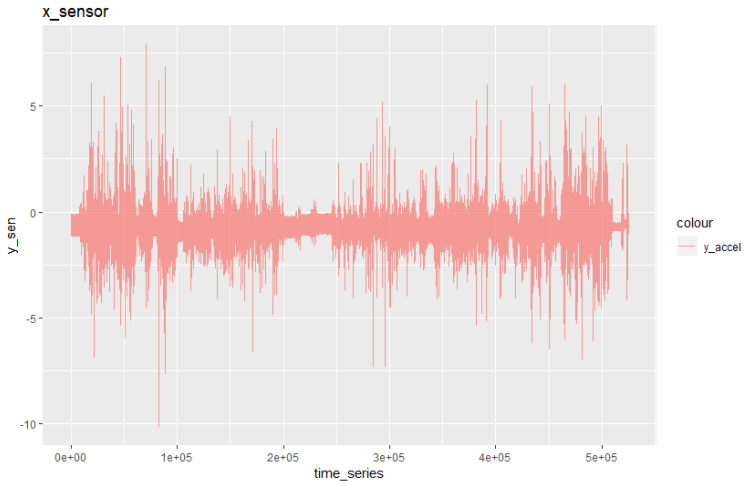

図 3.17: x 軸加速度センサーの折れ線グラフ (kaisen082)

図 3.17 から，全ての値において，正の値をとっていることが分かる．

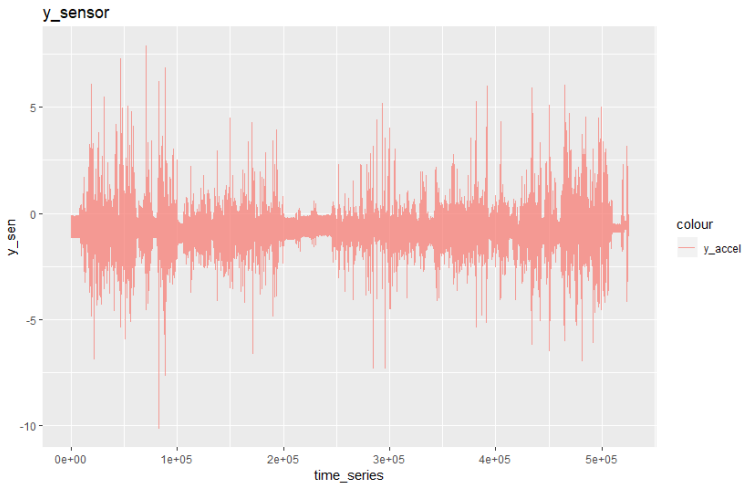

図 3.18: y 軸加速度センサーの折れ線グラフ (kaisen082)

図 3.18 から，正負の値が散在しており，0 を中心に値が揺れていることが分かる．

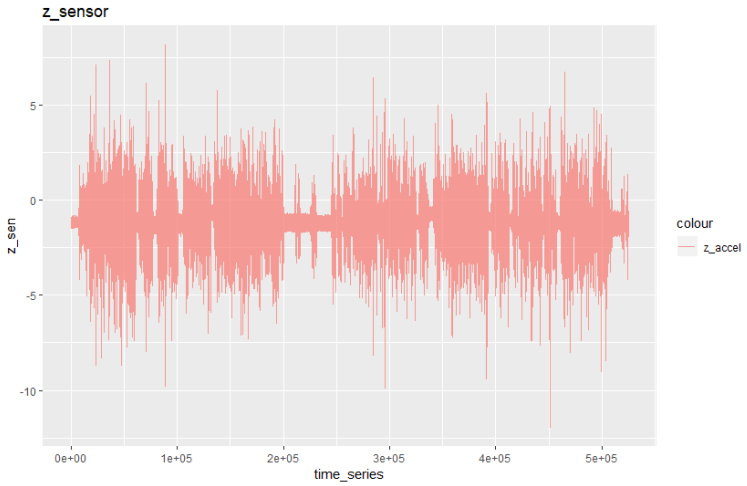

図 3.19: z 軸加速度センサーの折れ線グラフ (kaisen082)

図 3.19 から，ほとんど全ての値が正の値をとっている．

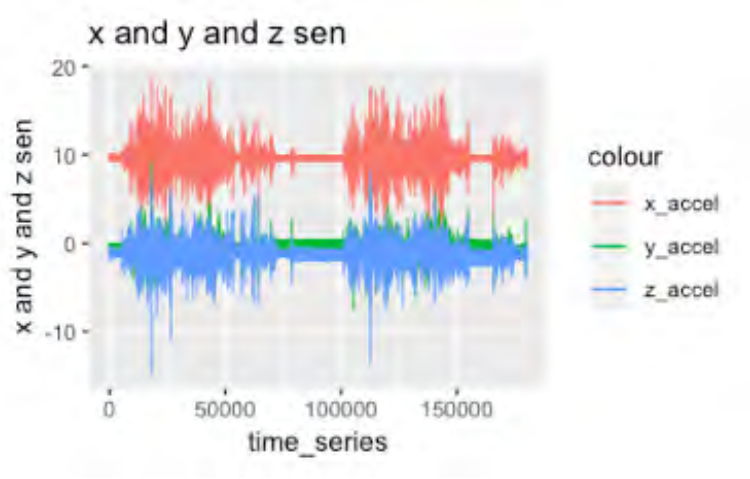

図 3.20: x 軸，y 軸，z 軸の加速度センサーの折れ線グラフ (kaisen082)

図 3.20 から，各軸の挙動は類似している一方，挙動の中心と考えられる箇所は異なることがわかる．

3.4.3 kaisen084

同様に，kaisen084 のセンサーデータがどのような特徴を持っているのかを調べる．本小節で示すデータは，4.3 節における分析の対象になっている．

その結果が図 3.21，図 3.22，図 3.23，図 3.24 である．

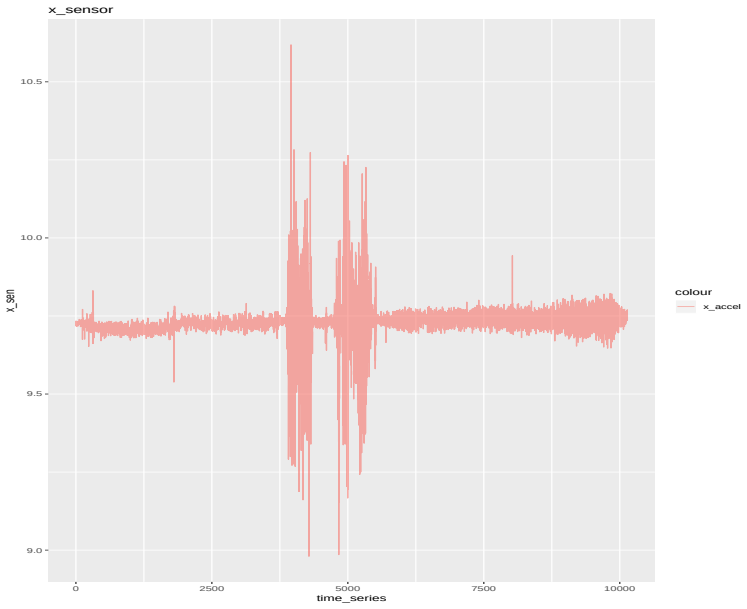

図 3.21: x 軸加速度センサーの折れ線グラフ (kaisen084)

図 3.21 から，全ての値において，正の値をとっていることが分かる．

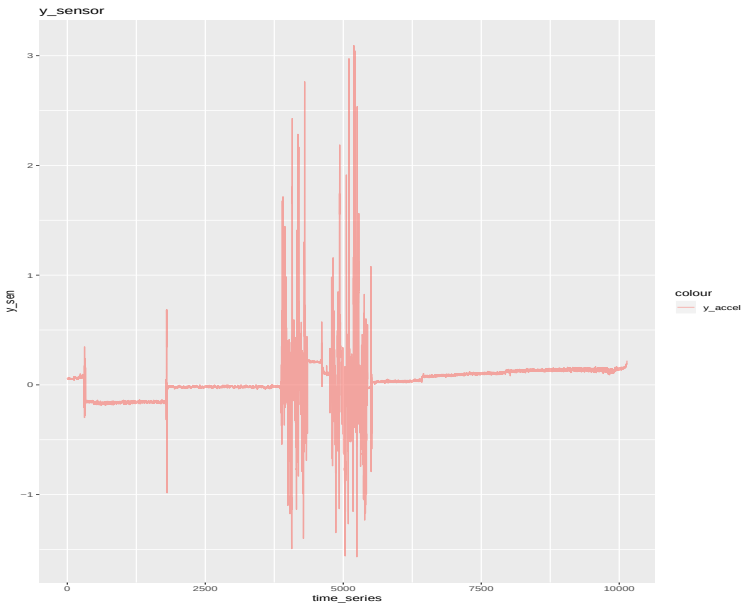

図 3.22: y 軸加速度センサーの折れ線グラフ (kaisen084)

図 3.22 から，正負の値が散在しており，0 を中心に値が揺れていることが分かる．

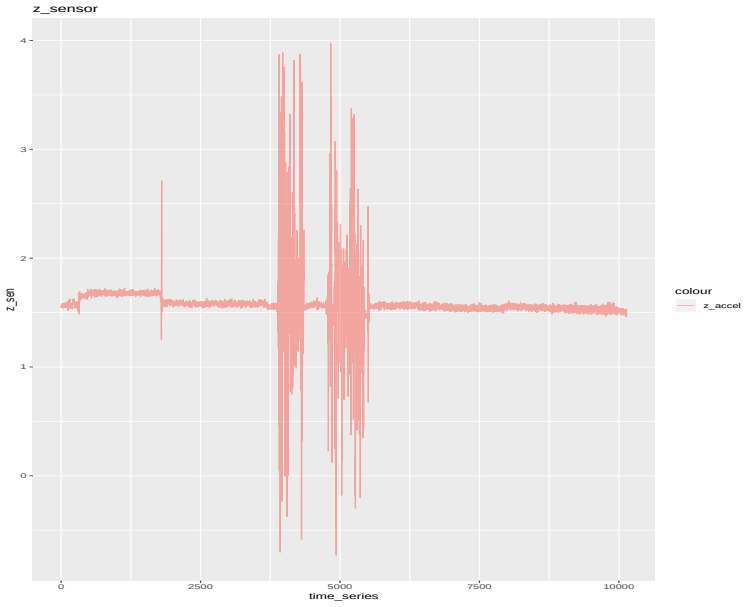

図 3.23: z 軸加速度センサーの折れ線グラフ (kaisen084)

図 3.23 から，基本的に正の値をとっていることが分かる．

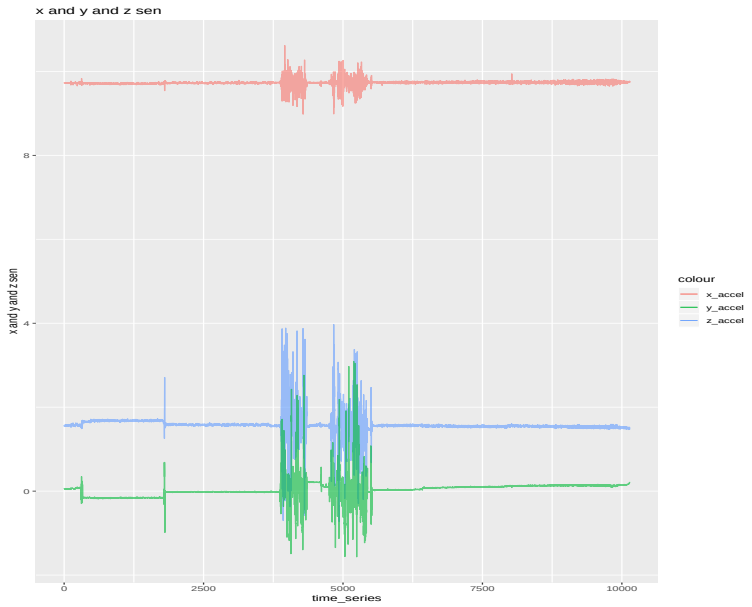

図 3.24: x 軸，y 軸，z 軸の加速度センサーの折れ線グラフ (kaisen084)

図 3.24 と図 3.16 を比較すると，kaisen080 と同様に，各軸の挙動が類似していることが分かる．

第4章　データの分析

本章は，まず，今回分析に用いるスコアを定義し，与えられたデータの分析結果を示す．本稿では次の 2 種類の事柄についての分析を行った．

- 動揺強度スコアによる道路状況把握の分析
- 雨天時と晴天時の違いについての分析

4.1　スコアの定義

今回のデータは表 2.3，2.2 に示す通り，加速度は 1/100 秒ごとのデータとして得られ，また，緯度，経度は 1 秒ごとのデータとして得られているため，緯度，経度と対応づけを行った加速度をより算出されるスコアを定義する．

4.1.1　動揺強度スコア

(4.1) 式に示すような，1 秒の中で得られている 100 個の加速度の分散を動揺強度スコアとする．

$$\text{動揺強度スコア} = \frac{1}{100}\sum_{i=1}^{100}(x_i - \bar{x})^2 \tag{4.1}$$

ここで，x_i は $i/100$ 秒における加速度センサーの値，$\bar{x}$ はその算術平均である．

各地点 1 秒で得られた車両の動きの「激しさ」をこのスコアによって表す．このスコアを用いて以降の分析において，道路状況の善し悪しを評価する．

例：動揺強度スコアの計算

kaisen080 における動揺強度スコア算出方法について，1 行に対して 1/100 秒ごとに記録した加速度センサーの値が，g_sens_z_1,g_sens_z_2,... , g_sens_z_99,g_sens_z_100 という変数名で記録されている．この 100 個の値を使い，1 行ごとに分散値を求めた．具体例は表 4.1 に示す通りである．

表 4.1: z 軸方向の 1/100 の加速度をもとに分散値を算出

g_sens_z_1	g_sens_z_2	g_sens_z_3	⋯	g_sens_z_98	g_sens_z_99	g_sens_z_100	分散
-0.4	5	-3	⋯	0.3	0.2	0.1	0.3

4.1.2 危険運転スコア

(4.2) 式に示すような，1 秒の中で得られている 100 個の加速度のうち，ある閾値を超えたものの数を危険運転スコアとする．

$$\text{危険運転スコア} = \begin{cases} \sum_{i=1}^{100} I(|x_i| > t) & \text{上下方向} \\ \sum_{i=1}^{100} I(x_i < -t) & \text{進行方向 (減速)} \\ \sum_{i=1}^{100} I(x_i > t) & \text{進行方向 (加速)} \end{cases} \tag{4.2}$$

ここで，x_i は $i/100$ 秒における加速度センサーの値，t は閾値，$I(\cdot)$ は指示関数である．

各地点 1 秒で得られた車両走行時の「急な動き」をこのスコアによって表す．このスコアを用いて以降の分析において，運転に着目した道路状況の善し悪しを評価する．

例：危険運転スコアの計算

kaisen080 における危険運転スコア算出方法について，1 行に対して 1/100 秒ごとに記録した加速度センサーの値が，g_sens_z_1,g_sens_z_2,... , g_sens_z_99,g_sens_z_100 という変数名で記録されている．この 100 個の値を使い，1 行ごとに閾値 t を超えている数を求めた．

加速度には正負があり，それぞれ加速，減速を指す．そのため，閾値が 0.3G であることは,「加速で 0.3G，減速で-0.3G を閾値とする」ことに対応する．ここでは加速，減速を別の現象として捉え，それぞれ閾値が 0.3G として考えられる根拠を探る．

減速については，畠中ら (2007) では，-0.3G がヒヤリハットを検出する閾値として適切であるとしている．一方，加速については，Hoberock(1977) において，0.3G が不快と感じる基準であることが示されている．これらに基づくと、0.3G は「減速においてはヒヤリハット、加速においては不快」の基準と考えられる．

さらに，Klauer et al.(2009) は，曲がる際の加速度が 0.3G を超えると危険である，と結論づけている．ここから，交差点等で曲がっている最中に 0.3G を超えているものは危険と考えられ，加速時に 0.3G を閾値として考えることが妥当であるこが示唆される．

西堀ら (2010) では，加速度の値に関する閾値を± 0.25 から± 0.3 と定めており，これを参考に閾値 $t = 0.3$ に設定しスコアの計算を行う．

g_sens_x_1	g_sens_x_2	g_sens_x_3	・・・	g_sens_x_97	g_sens_x_98	g_sens_x_99	g_sens_x_100
-0.4	5	10	・・・	0.38	0.2	0.4	0.1

図 4.1: x 軸方向の 1/100 の加速度

表のように，1 行に 100 個の加速度が 3 方向それぞれ，記録されている．これに対して，閾値の判別を行いフラグを付与する．今回は加速度の閾値を± 0.3 と定め，各加速度が± 0.3 を超えるか超えないか確認する．その際，加速度の絶対値が閾値を超えた場合は 1，超えなかった場合は 0 とする．そしてその数を合計した変数を追加する．

g_sens_x_1$	g_sens_x_2	g_sens_x_3	・・・	g_sens_x_97	g_sens_x_98	g_sens_x_99	g_sens_x_100	合計
1	1	0	・・・	1	0	1	0	40

図 4.2: 閾値によりフラグを立てた x 軸方向の 1/100 の加速度

4.2　動揺強度スコアによる道路状況把握の分析

前章の基礎集計より，z 軸が他の x 軸や y 軸に比べると，一番揺れが大きいことが判明した．そこで，今回は z 軸を車両の動きの「激しさ」とし，z 軸方向について，1 秒ごとに分散値（＝散らばり度合い）を計算した．この分散値を動揺強度スコアとして定義し，道路状況の善し悪しを測る．

4.2.1　分析の目的

加速度センサーを用いて道路状況を把握することにより，道路補修の計画をデータに基づき客観的な証拠から立てられるようにする．

4.2.2　使用するデータ

kaisen080 について，計算量を考慮した上で，10,000 行を取り出すことにし，この 10,000 行に対して，探索的な分析を行った．この 10,000 行の内訳は，基礎集計の表 3.9 の通りである．

4.2.3　結果の概観

今回分析対象とした車の走行場所，及び分散値をヒートマップを用いて地図で視覚化した全体の図が以下の図 4.3 である．色が濃い部分が動揺強度スコアが大きい部分であり，道路状況が悪いと示唆される部分である．

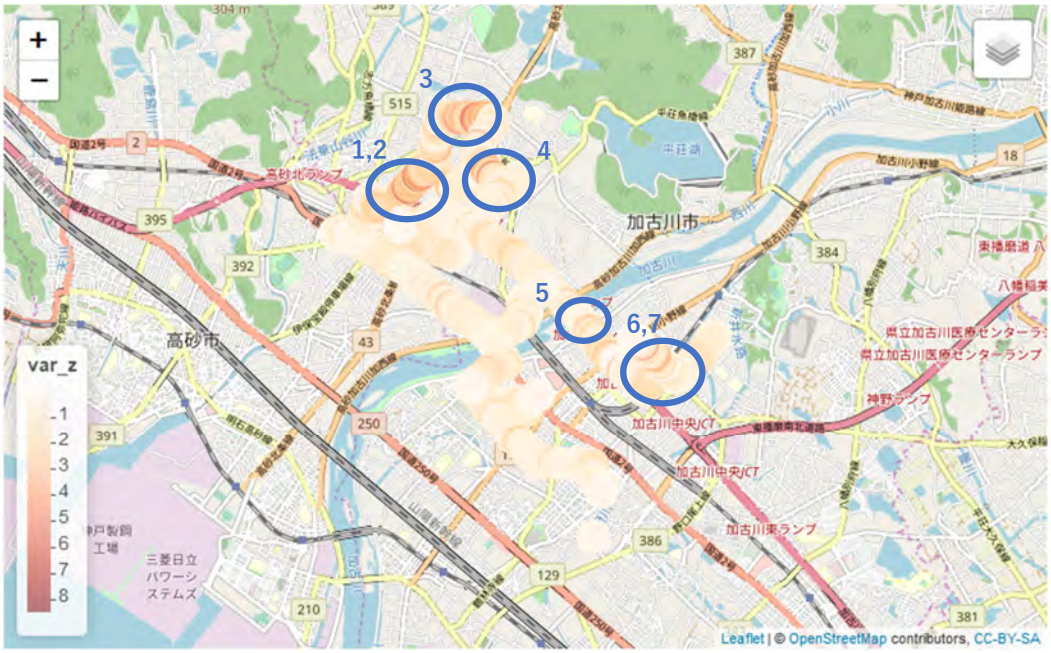

図 4.3: 走行場所とセンサーデータの z 軸分散値のヒートマップ

4.2.4　道路状況が悪い箇所

青色で丸をつけた部分について，1 から順にピックアップしてみていく．

動揺強度スコアが大きい，色が濃い部分を図 4.3 の 1 から順にピックアップしてみていく．その際に，Google ストリートビューを用いて実際の道路状況の写真も併せて確認する．

1.　まんてん加古川，かみおかこどもクリニック周辺

まんてん加古川，かみおかこどもクリニック周辺は，次のような結果になった．

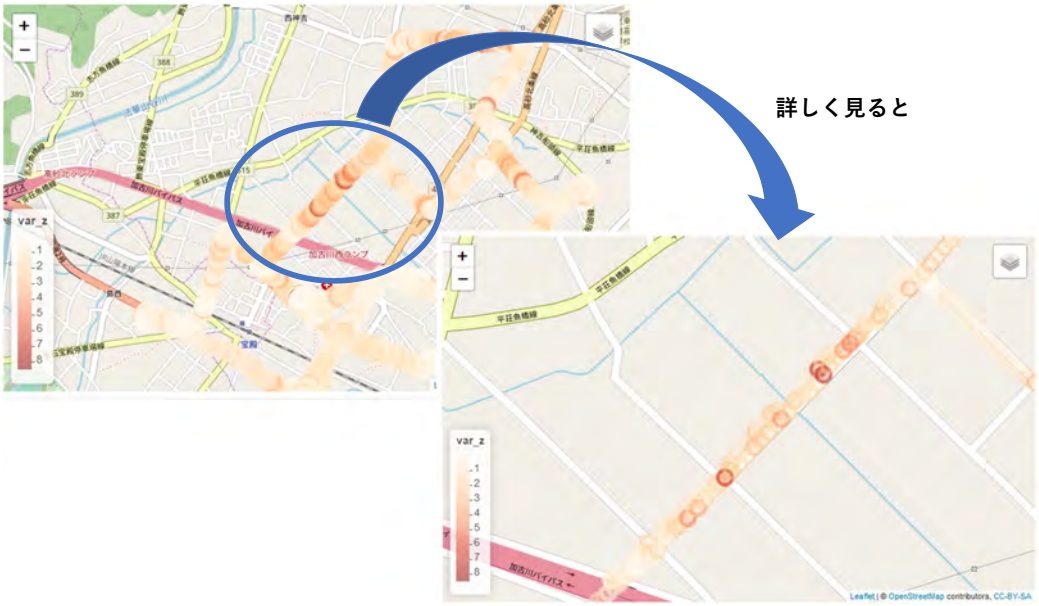

図 4.4: まんてん加古川，かみおかこどもクリニック周辺の地図

図 4.5: 2013 年 3 月時点の Google ストリートビューによる道路状況確認 1

地面のひび割れが確認できた．また，全体的に舗装状態が悪い状態であることも同時に分かった．

2. 西神吉町中西の交差点周辺

西神吉町中西の交差点周辺は，次のような結果になった．

図 4.6: 西神吉町中西の交差点周辺

図 4.7: 2013 年 3 月時点の Google ストリートビューによる道路状況確認 2

大きなひび割れは見られなかったが小さめの地面のひび割れが確認が出来た.

3. 西神吉幼稚園周辺

西神吉幼稚園周辺は，次のような結果になった.

図 4.8: 西神吉幼稚園周辺の地図

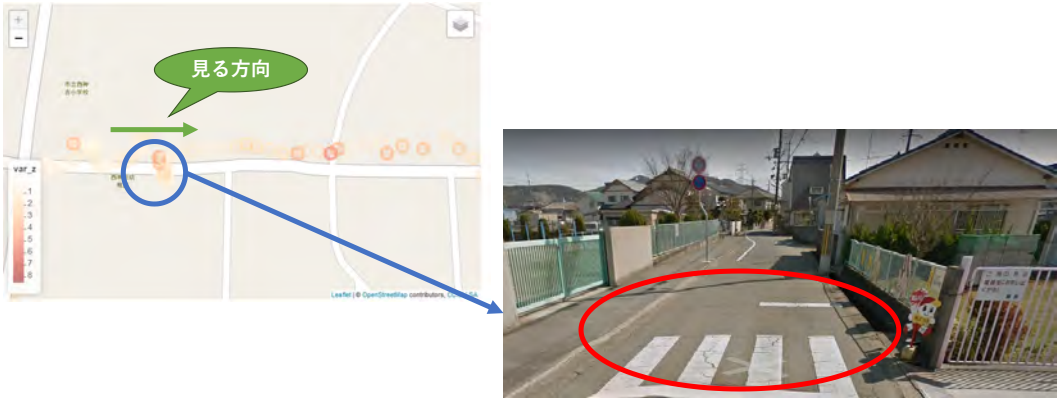

図 4.9: 2013 年 3 月時点の Google ストリートビューによる道路状況確認 3

横断歩道付近で地面のひび割れが確認できた．

図 4.10: 2013 年 3 月時点の Google ストリートビューによる道路状況確認 4

大きな地面のひび割れは見当たらなかったが，マンホールにセンサーが反応したと思われる．

4. 東神吉町神吉周辺

東神吉町神吉周辺は，次のような結果になった．

図 4.11: 東神吉町神吉周辺の地図

図 4.12: 2013 年 3 月時点の Google ストリートビューによる道路状況確認 5

比較的多きな断続的に続く地面のひび割れが確認できた.

図 4.13: 2013 年 3 月時点の Google ストリートビューによる道路状況確認 6

コンクリートのひび割れと，アスファルトのつなぎ目が目立つ．この部分にセンサーが反応したと考えられる．

5. 加古川河川敷緑地周辺

加古川河川敷緑地周辺は，次のような結果になった．

図 4.14: 加古川河川敷緑地周辺の地図

図 4.15: 2018 年 12 月時点の Google ストリートビューによる道路状況確認 7

少し見にくいが，一部のみセンサーが大きく反応していた．ここは，コンクリートのつなぎ目に反応したと思われる．

6. 加古川市立氷丘小学校周辺

加古川市立氷丘小学校周辺は，次のような結果になった．

図 4.16: 加古川市立氷丘小学校周辺の地図

図 4.17: 2018 年 12 月時点の Google ストリートビューによる道路状況確認 8

ここも，一部だけ大きくセンサーが反応していた．ここは，地面のひび割れか，踏切がセンサーに反応したと思われる．

7. 加古川町大野周辺

加古川町大野周辺は，次のような結果になった．

図 4.18: 加古川町大野周辺の地図

図 4.19: 2018 年 12 月時点の Google ストリートビューによる道路状況確認 9

この部分は断続的に比較的大きめの動揺強度スコアを示していた．ここは，地面のひび割れに加えてマンホールがセンサーに反応したと思われる．

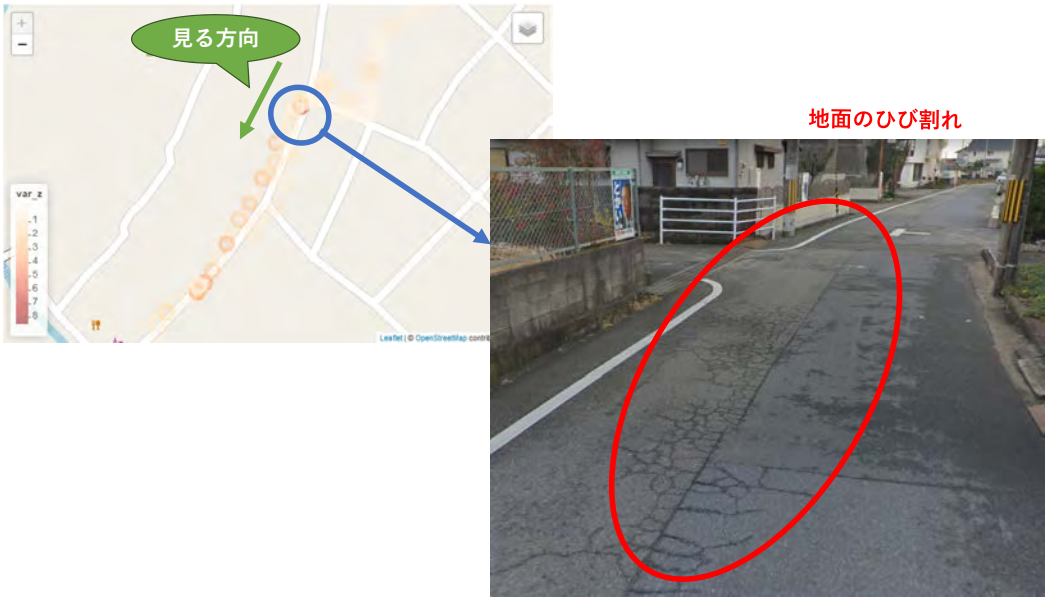

図 4.20: 2018 年 12 月時点の Google ストリートビューによる道路状況確認 10

この部分は，地面のひび割れが断続的に見られた．

4.2.5　閾値の設定

4.2.3 節及び節での分析は，走行した箇所全てについて，動揺強度スコアが高い部分を手探りで探していた．しかし，悪い道を探すという点においては，走行箇所全てを探索する必要があるため効率が悪い．そこで，ある一定程度を越えると「道路状況が悪い」と判断できる基準，すなわち閾値を設定することで，効率よく道が悪い場所を探すすことができるようにすることが求められる．

今回は，これまでの分析の経験則に基づき，アドホックな形で，閾値を設定した．その結果が以下の図 4.21，図 4.22 である．

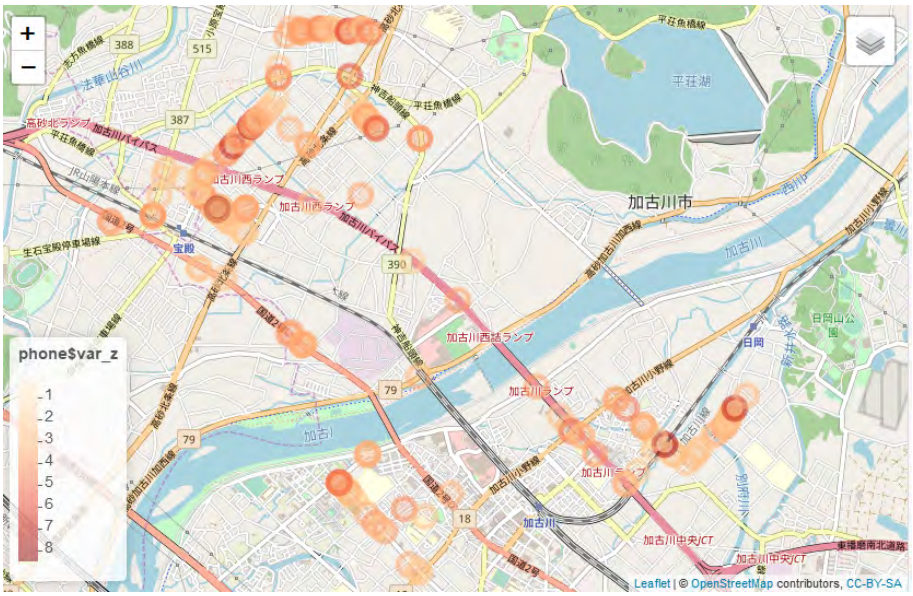

図 4.21: 分散値が 2.5 以上の z 軸センサーデータのヒートマップ

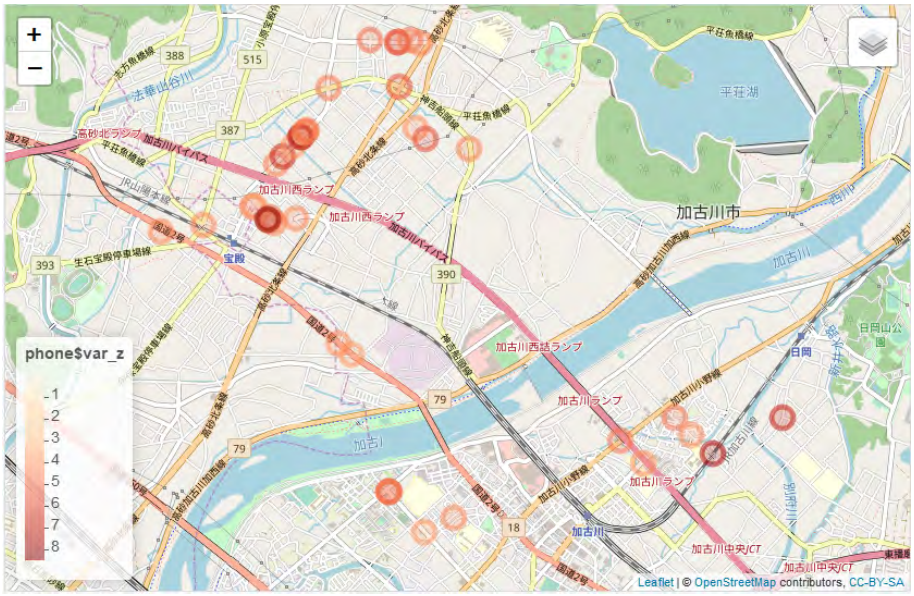

図 4.22: 分散値が 4.0 以上の z 軸センサーデータのヒートマップ

これより，動揺強度スコアが高い箇所のみをピックアップすることが可能になる．

4.2.6 考察

動揺強度スコアが高い地点について，1 点 1 点 google ストリートビューを利用して確認したところ，センサーデータが主に反応する場所として，次の 4 つが主なものとしてあった．

- 地面の比較的大きなひび割れ
- コンクリートのつなぎ目
- マンホール
- 踏切

踏切やマンホールといった，道路に固定されているような道を除くと，動揺強度スコアを用いて道の悪い地点を洗い出すことが可能であることが分かった．

この分析結果より,「センサーデータの分散算出 → 地図でヒートマップを描く → 実際の道路状況把握 → 閾値設定による動揺強度スコアが高い地点のピックアップ」というスキーマを通して，道路補修が必要な道の予測をすることが可能になると考えられる．

4.2.7 今後の課題

今回はアドホックな形で閾値を設定した．しかし，道路の補修が必要な動揺強度スコアはどの程度かを実際の道路状況から定めることが出来ていない．従って，そのような道路補修が必要な程度の閾値はどのくらいかを定め，閾値の最適化を行うことで，補修が必要な道の自動検出を行えるようにすることがこれからの課題である．

4.3　雨天時と晴天時の違いについての分析

4.3.1　分析の目的

雨天時と晴天時での同経路の走行に注目し，危険運転スコアに基づいて加速度を可視化することで，雨天時に危険運転スコアが高い道の発見を行うことを目的とする．

4.3.2　使用するデータ

- tenki.jp より取得した，兵庫県三木市のある日の積算降雨量であるアメダスデータ (tenki.jp には加古川市のアメダスデータが存在しなかった為，最も加古川市に近い三木市を選択した)
- Phone データより，経路が概ね同じである 2016 年 6 月 23 日の 8 時から 13 時 30 分までの kaisen084(合計降雨量 40.0mm) と，2016 年 3 月 23 日の 8 時から 11 時までの kaisen084(合計降雨量 0.0mm) のセンサーデータ

ここで用いたデータは表 3.13 に示した kaisen084 のデータである．それぞれ，2016/6/23 の雨の日 9780 行 (2 時間 48 分 54 秒)，2016/3/23 の晴れの日 10134 行 (2 時間 42 分 36 秒) である．

4.3.3　取得データの概要

例として，今回使用した 2016 年 6 月 23 日の tenki.jp の該当ページ[1]を図 4.23 に示す．このページより，R のパッケージ rvest を用いて，図 4.24 の用なデータを作成した．

2016 年 6 月 23 日を選択した理由としては，図 4.24 より，前日 (13.5mm) と前々日 (19.5mm) も合計降雨量が多く，蓄積された雨水により道路状況が悪いと判断した為である．

図 4.23: 2016 年 6 月 23 日の三木市のアメダス

[1] https://tenki.jp/past/2016/06/23/amedas/6/31/63461.html より引用 (2019 年 3 月 21 日アクセス)

日付	積算降雨量
2016-06-20	0.0
2016-06-21	19.5
2016-06-22	13.5
2016-06-23	40.0
2016-06-24	41.0

図 4.24: 2016 年 6 月 23 日付近の合計降雨量

次に，選択したデータでの経路を図示する．
図 4.25，4.26 を比較しても概ね一致していることが分かる．

図 4.25: kaisen084 雨天時

図 4.26: kaisen084 晴天時

4.3.4 基礎集計に基づくセンサー方向の解釈

kaisen084 のセンサーデータの x 軸，y 軸の解釈を与える．
x 軸，y 軸どちらも 0.30G という閾値で危険運転スコアを可視化し，それに基づく kaisen084 における 2016 年 3 月 23 日 (日積算降雨量 0.0) のデータのプロットを図 4.27，図 4.28 に示す．
ここでは，危険運転スコアが高いほど急加速もしくは急カーブを表す．これは本分析で扱っている危険運転スコアとは異なり，軸の解釈を行うためだけに使うものとする．
ただし今回，図 3.24 を見ると x 軸の加速度は，かなり上に分布していることが分かるので平均による中心化を行い，0 を中心に分布させる．
平均による中心化とは，データの値から平均値を引くことで,「0」に，解釈可能な得点を移動させることを意味する[2]．
これより，x 軸の閾値を「(x 軸の加速度の平均)+0.30G」とした．
aaa 図 4.27 は x 軸の危険運転スコアを示していて，色が濃くなるほど危険運転スコアが高いことを表す．図 4.27 を見ると，直線の道路で色が濃くなっていることが分かる．直線の道路で急カーブをする可能性は低い．これより，x 軸は車両の進行方向を示すと推測することができる．

[2] http://www.juen.ac.jp/lab/okumura/class/g2/node3.html より引用 (2019 年 3 月 22 日アクセス)

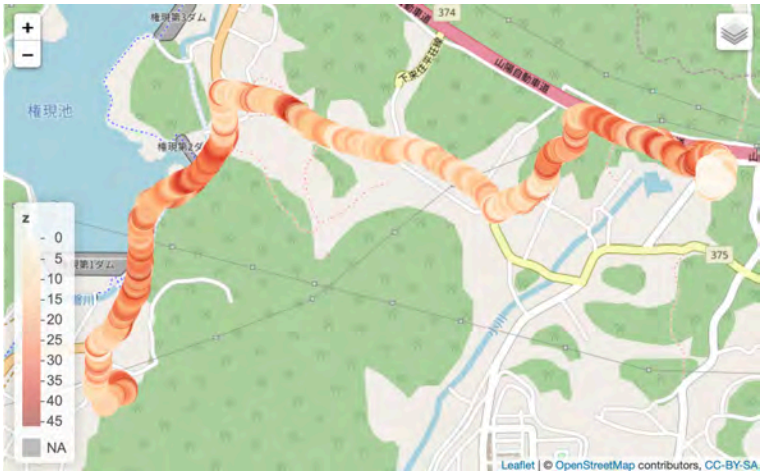

図 4.27: x 軸の解釈

図 4.28 は y 軸の危険運転スコアを示していて，色が濃くなるほど危険運転スコアが高いことを表す．図 4.28 を見ると，交差点において色が最も濃くなっている．交差点において進行方向に急加速する可能性は低い．これより，y 軸は車両の横断方向を示すと推測することができる．

図 4.28: y 軸の解釈

4.3.5 分析

前節での kaisen084 のセンサー方向の解釈に基づくと，x 軸が進行方向，y 軸が左右方向となるので，それぞれ閾値を設定して進行方向，左右方向の危険運転スコアをプロットした結果を図示する．

本分析では，畠中ら (2007)，Klauer et al.(2009) において用いられている 0.3 を閾値とし，危険運転スコアを算出することとする．

ここでの危険運転スコアの解釈として，進行方向に関しては危険運転スコアが高いほど急減速を表し，左右方向に関しては急カーブを表す．しかし，先ほど述べたように x 軸 (進行方向) は中心化を行う必要がある為，進行方向の閾値は「(進行方向の加速度の平均)-0.3G」とした．

初めに，天候別の進行方向の危険運転スコアを図示する．
図 4.29，4.30 で色が濃くなっている地点は，危険運転スコアが高く，急減速している地点である．

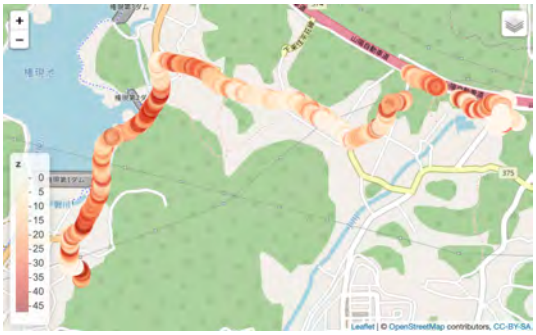

図 4.29: 進行方向の危険運転スコア (雨天時)

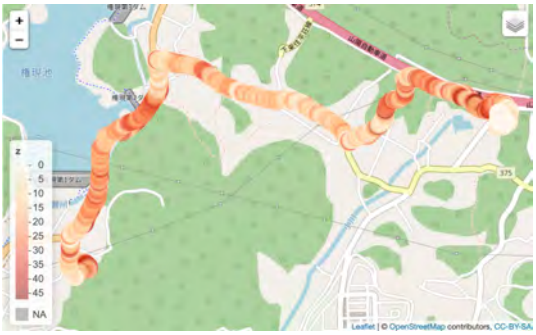

図 4.30: 進行方向の危険運転スコア (晴天時)

ここで，二つの図を比較した時，比較的に雨天時の方が色が濃くなっている地点を例として二箇所，Google ストリートビューにて確認する．

- 例 1

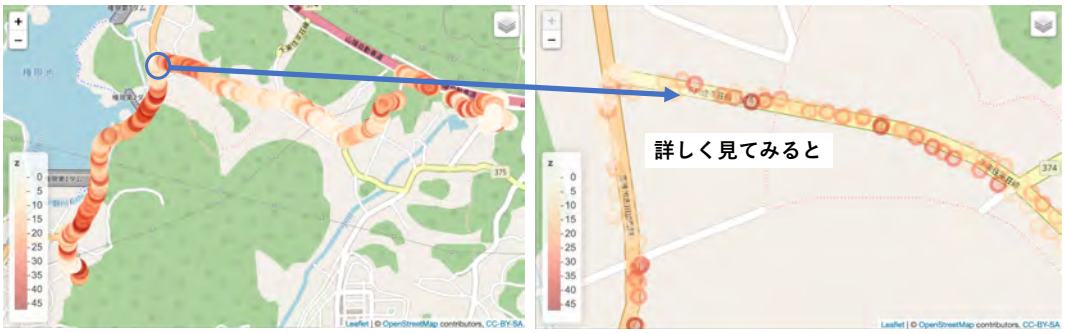

図 4.31: 例 1(雨天時)

図 4.32: 例 1(晴天時)

図 4.33: 例 1 の Google ストリートビュー

図 4.31,4.32 を見てみると，雨天時は交差点に差し掛かる前から減速しているが，晴天時は交差点の直前で減速していることが分かる．

これらの原因としては，見晴らしの悪い雨天時には他車両が見えにくい，かつ滑りやすいので速度を落としてから交差点に差しかかることなどが考えられる．また，Google ストリートビューの画像 (図 4.33) をみて分かる通り，交差点より手前に合流地点が存在することも原因の一つであろう．

- 例 2

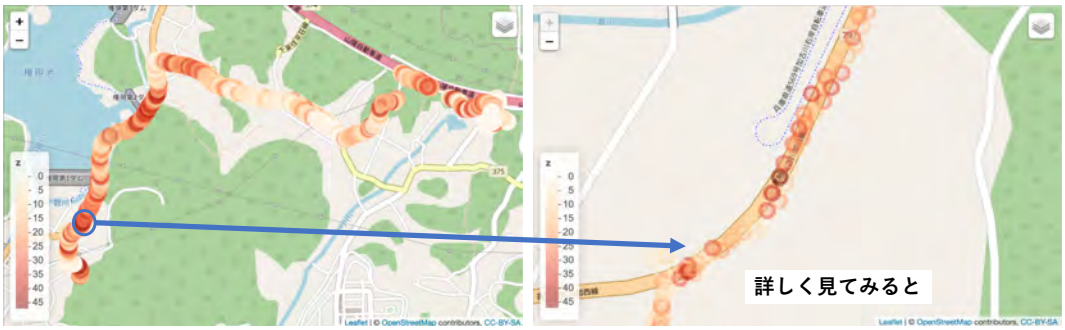

図 4.34: 例 2(雨天時)

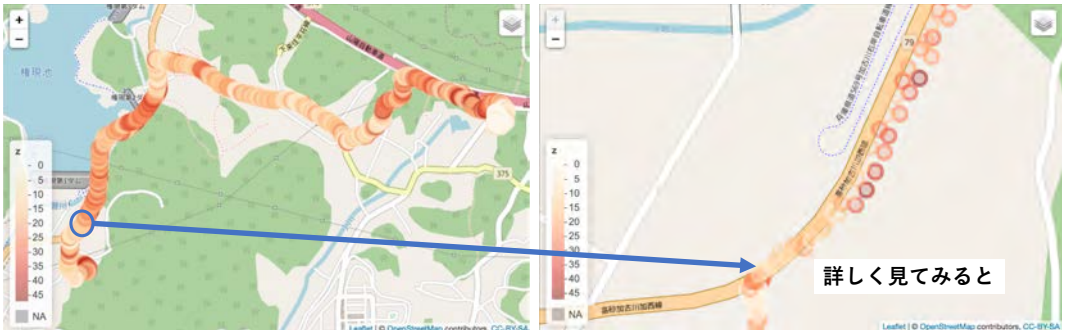

図 4.35: 例 2(晴天時)

図 4.36: 例 2 の Google ストリートビュー

図 4.34,4.35 を見ると，雨天時の方がカーブ地点での減速の仕方が激しい．また Google ストリートビューの画像 (図 4.36) で確認すると，若干の傾斜があることが分かる．

つまり，この道は雨天時は傾斜によって加速してしまうので激しい減速が必要であることが考えられる．

次に，天候別の左右方向に関する危険運転スコアを図示する

図 4.37，4.38 で色が濃くなっている地点は，危険運転スコアが高く，急カーブもしくは激しいハンドル操作となっている地点である．

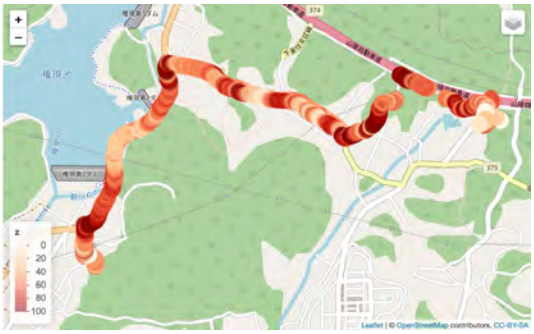

図 4.37: 左右方向の危険運転スコア (雨天時)

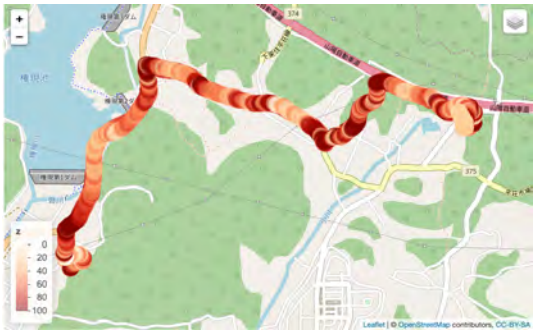

図 4.38: 左右方向の危険運転スコア (晴天時)

先ほどと同じく，二つの図を比較した時，比較的に雨天時の方が色が濃くなっている地点を例として二箇所，Google ストリートビューにて確認する．

- 例 3

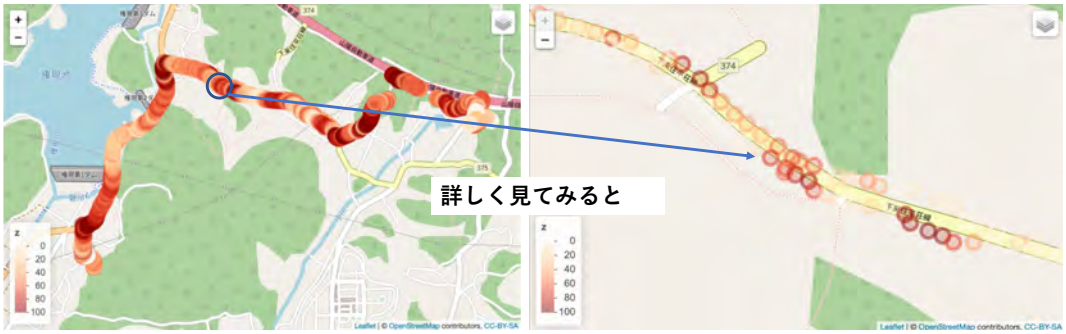

図 4.39: 例 3(雨天時)

図 4.40: 例 3(晴天時)

図 4.41: 例 3 の Google ストリートビュー

Google ストリートビューの画像 (図 4.41) より，図 4.39,4.40 に示した箇所は，一車線ほどの幅であることが分かる．つまり，見晴らしの良い晴天時は安定して走行できるが，見晴らしの悪い雨天時には，車線が無いためハンドルが安定しないことなどが考えられる．

- 例 4

図 4.42: 例 4(雨天時)

図 4.43: 例 4(晴天時)

図 4.44: 例 4 の Google ストリートビュー

Google ストリートビューの画像 (図 4.44) より，図 4.42,4.43 に示した箇所は，見通しの良くない山道であることが分かる．晴天時もハンドル操作を要する山道であるが，雨天時は更に激しいハンドル操作が必要であることが例 4 より分かった．

4.3.6 参考

例 1 に関して，左右方向に関する危険運転スコアで同じ地点を確認したものが図 4.45，4.46 である．

雨天時の進行方向の図 4.31 において交差点付近で減速をあまり行わなかった為，図 4.45 では，ハンドルが安定していないことが考えられる．

図 4.45: 例 1 の左右方向の危険運転スコア (雨天時)

図 4.46: 例 1 の左右方向の危険運転スコア (晴天時)

これらの例は目視で違いを確認したが，危険運転スコアがある値より高いものだけを取り出すことによって，更に検出が容易くなることが分かった．今回は例として進行方向の危険運転スコアに対して，下限を 30 と 40 に設定し抽出しているが，コードの値を変更することで自由に設定が可能である．

下限を 30 に設定したものが図 4.47，4.48 である．

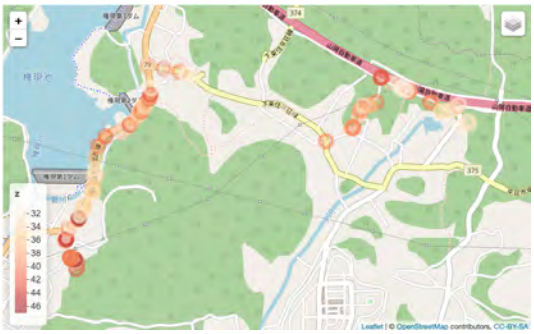

図 4.47: 下限を 30 とした進行方向の危険運転スコア (雨天時)

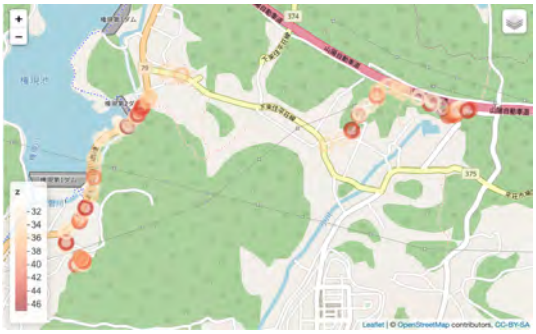

図 4.48: 下限を 30 とした進行方向の危険運転スコア (晴天時)

下限を 40 に設定したものが図 4.49，4.50 である．

図 4.49: 下限を 40 とした進行方向の危険運転スコア (雨天時)

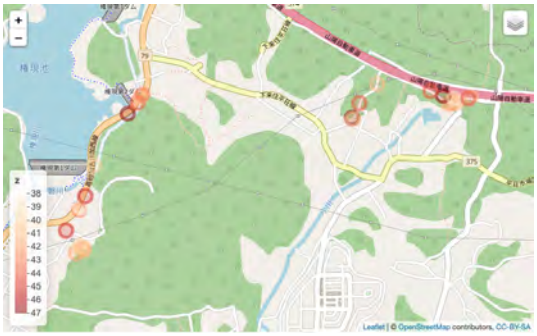

図 4.50: 下限を 40 とした進行方向の危険運転スコア (晴天時)

また，晴天時の走行経路に対して，雨天時における走行経路をマッチングし，マッチングした点

に対して,「(雨天時の進行方向の危険運転スコア)-(晴天時の進行方向の危険運転スコア)」を地図上にプロットした．今回，雨天時の雲が GPS へ影響を及ぼし正確な緯度経度を測定できていない地点が存在した為，晴天時と全く同じ緯度経度となる地点はほとんど存在しなかった．そのため，晴天時の走行経路各点に対して，もっとも近い雨天時の点を用いた．これを図 4.51 に示す．

図 4.51 は，青になればなるほど雨天時の方が急減速している地点となり．赤になればなるほど晴天時の方が急減速している地点となる．

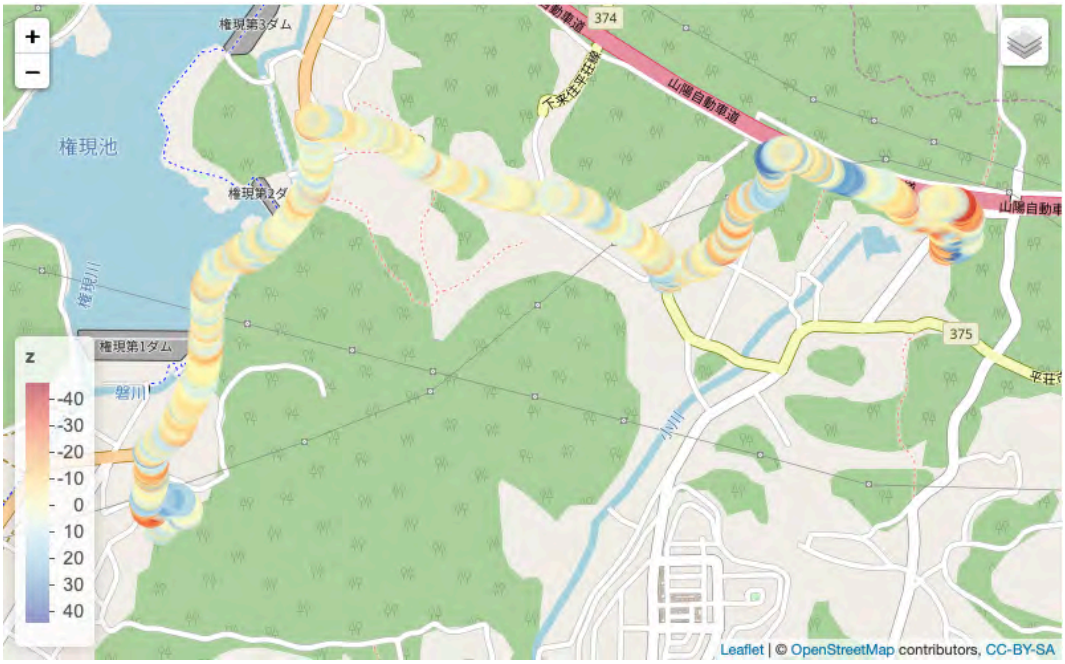

図 4.51: 天候別の進行方向の危険運転スコアの差異

4.3.7　今後の課題

今後の課題として，以下の 3 点が挙げられる．

- 雨天時に危険となりうる道を自動で検出できるようにする
- 加古川市の交通事故データとの比較を行う
- 進行方向と左右方向を混合した総合指標を作成する

第5章　より深い分析に向けて

本章では，より深い分析を行うための基礎となる分析結果について述べる．

5.1　速度を考慮した加速度の閾値の設定

5.1.1　背景

第 4.3 節における “雨天時と晴天時の違いについての分析”では，1/100 の加速度の値の閾値を，0.30G と定めて危険運転スコアを可視化していた. 加速度の閾値を設定する際に，速度に関わらず閾値を一定値と設定してしまうと，速度の違いから生み出される加速度の危険度の変化を見逃してしまう可能性がある．

速度によって，加速度の値がもつ危険度は変わっていく．例えば，20km/h で走行時の急ブレーキと 80km/h で走行時の急ブレーキでは事故を引き起こす可能性は異なる．また，加速度は速度のように，実際に車両が動いたスピードを表すものではなく，速度の変化量で求めるものである．ある 2 つの加速度の値は同じでも，そのときの速度によって，危険度の指標は大きく異なる．

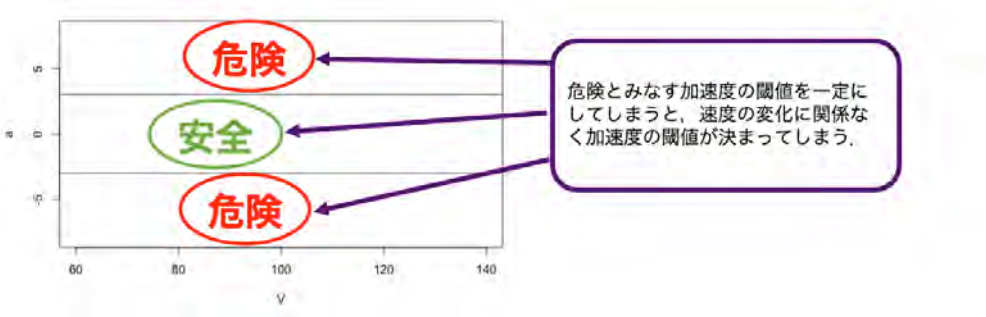

図 5.1: 閾値を± 0.3 にした際の加速度と速度の関係図

加速度の閾値を一定にすると，速度が変化しても同じ幅の加速度の危険度の判定になってしまう．そのため，走行の危険度合いを甘く判断してしまう可能性がある．そこで，今回は速度の大きさによって危険とみなす加速度の範囲が変化することを想定して分析する．

5.1.2　分析内容

GPS 座標による速さと得られた加速度の関係性に関する分析を行う．

5.1.3　使用するデータ

kaisen082 について，計算量を考慮した上で，10,000 行を取り出すことにし，この 10,000 行に対して分析を行った．この 10,000 行の内訳は，基礎集計の表 3.11 の通りである．

5.1.4　変数の作成

図 5.2: km_h の作成方法

各時点 t について (5.1) 式に示すような，2 地点間の距離 (km) を時間 (hour) で割ったものを速さ (km/h) とする．

$$v^{(t)} = d(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t+1)}) \times \frac{3600}{1000} \tag{5.1}$$

ここで，$v^{(t)}$ は t 時点の速さ，$\boldsymbol{x}^{(t)}$ は t 時点における GPS 座標，$d(\cdot, \cdot)$ は 2 地点間の距離 (m) である．

この速さを用いて，実際に得られた加速度との関係性に関する分析を行った．

5.1.5　分析データの作成

上記の速さの求め方から，変数 km_h を作成した．この km_h を既存のデータに追加する．加速度に関しては，各時刻，1/100 秒で取られたものの算術平均とした．

g_sens_x_1	g_sens_x_2	g_sens_x_3	・・・	g_sens_x_97	g_sens_x_98	g_sens_x_99	g_sens_x_100	km_h
0.4	1.4	2.1	・・・	0.9	0.5	5.6	2.3	45

図 5.3:

5.1.6 分析結果

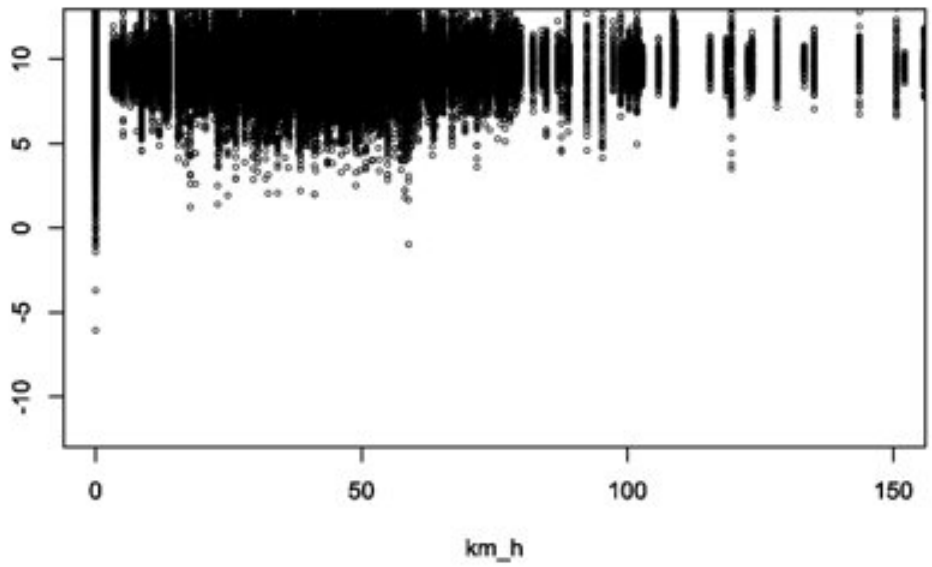

図 5.4: kaisen082 の速度と x 軸方向の加速度のプロット

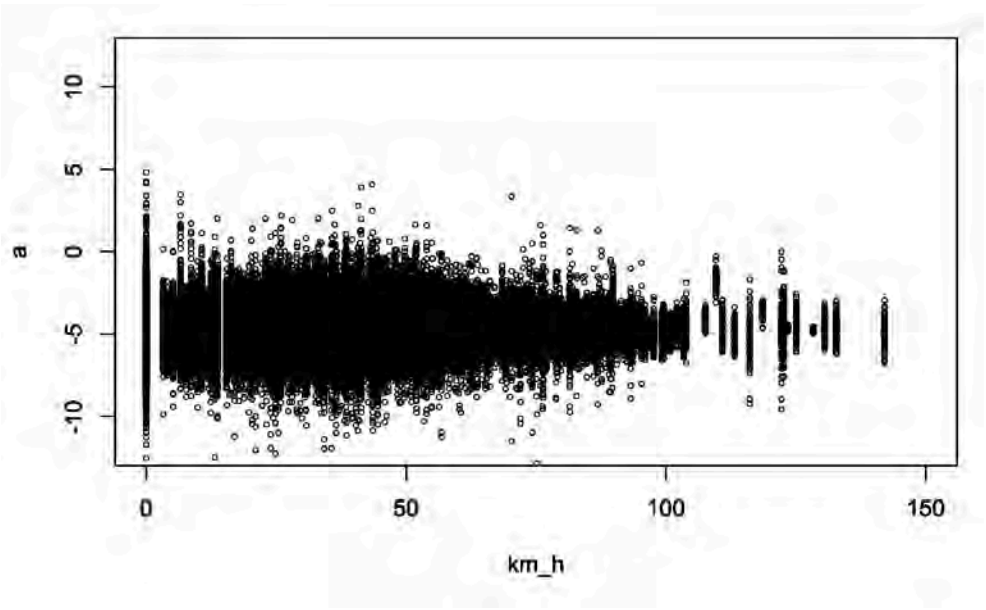

図 5.5: kaisen082 の速度と y 軸方向の加速度のプロット

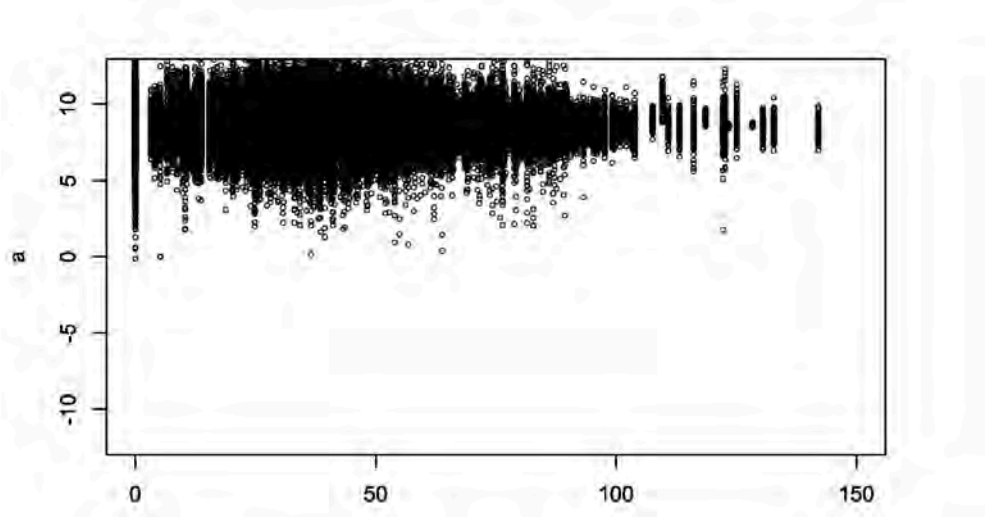

図 5.6: kaisen082 の速度と z 軸方向の加速度のプロット

方向によって加速度の集合する中心は異なるが，どれも，速度が大きくなればなるほど，プロットされている数が少なくなり，加速度の大きさの範囲も狭小になっている．このことから，速度の変化が加速度の大きさの幅に影響を与えていると考えられる．

5.1.7 参考

実際に，速度を用いた加速度の閾値の設定方法の 1 例を紹介する．Laura et al.(2016) によると，速度によって危険と判断する加速度の閾値は変化すると述べている．Laura et al.(2016) によると，速度と加速度の関係における速度を固定したときの閾値は (5.2) 式で与えられる．ここで，a はある方向の加速度，V は速度，g は重力加速度である．

$$|a| = g\left[0.198\left(\frac{V}{100}\right)^2 - 0.592\left(\frac{V}{100}\right) + 0.569\right] \tag{5.2}$$

実際にグラフ化すると，図 5.7 のようになり Laura et al.(2016) から，理論的にも速度の大きさによって加速度のとる値に影響があることがわかった．

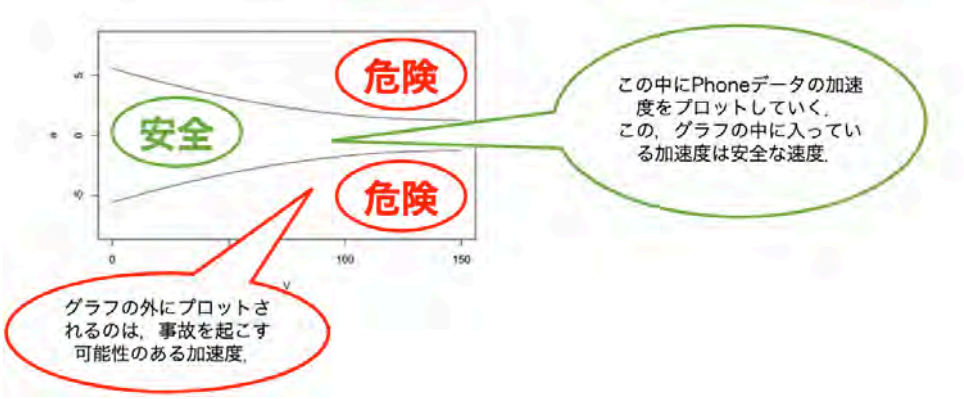

図 5.7: (5.2) 式をグラフ化したもの

5.1.8 今後の課題

実際に，今回の加古川市の公用車データに当てはまるような，速度を考慮した上で危険と捉える加速度の閾値の設定方法を確立していく必要がある．Laura et al.(2016) の研究で言えば，係数の調整や，データの加工方法などをすることで，速度に対応した加速度の閾値の決定方法を作成することが可能になる．

第6章　おわりに

本章では，本稿のまとめと今回の分析の中で残っている課題について述べる．

6.1　本稿のまとめ

本稿では，加古川市の公用車に搭載されたスマートフォンによって取得された加速度データを用いて，動揺強度スコアと危険運転スコアを定義し，これを用いて，道路状況についての分析を行った．

道路が劣化している箇所の分析に関しては，車体が揺れる場所を抽出し，google ストリートビューの画像による検証から，車両の揺れが発生しうる場所について考察を行い，実際にひび割れが生じている可能性がある地点を複数特定した．

また，天候の違いに関する道路状況の分析に関しては，危険と考えられる閾値を元にした危険運転スコアを地図上に表すことで天候により危険が生じると考えられる箇所の抽出を行った．また，抽出された箇所に関して google ストリートビューの画像による検証を行った．これにより，雨天時に運転が不安定になる可能性がある道を特定することができた．

6.2　今後の課題

本稿全体における課題として，加速度に対する補正が挙げられる．加速度は車体の挙動を示す重要な指標であるため，想定している方向の加速度として得られていない場合は，角度を用いて補正する必要がある．本稿ではある程度正確に得られているものと仮定し，分析を行っているが，測位状態によっては補正を行う必要がある．

角度データによる補正法の一例として，ここでは木山ら (2014) によるスマートフォンを用いた先行研究を挙げる．この研究では，まず加速度の分散から車両の停止状態を検出し，停止状態におけるスマートフォンの角度情報を加速度の数値から推定する．その後，方位角センサから得た方位角と，推定した角をもとに，角度が既知である場合の補正法である，回転行列による補正を行い，車の軸に一致した，正確な加速度を得ることを試みている．

この方法における問題点としては，停止状態を適切に把握できない可能性がある点が挙げられる．特に地震による揺れが発生した場合は，車体が停止していても，誤って「停止していない」と認識してしまう恐れがある．例えば GPS 情報を用いれば，停止している場合は座標が変化しないので，この問題を解決できる考えられる．また，角度の推定を行っているが，今回提供された OBD2 データには，角度情報と考えられる変数 (e-conpass 値) が含まれているため，分析において改めて角度を推定する必要はないと言える．ただし，Phone データについては，角度情報が付与されていないので，あらかじめ設置状態を確認し，車における軸とどの程度ずれているか，測定しておく必要がある．

また，今回の報告書では，OBD2 データを利用した分析は行うことが出来なかった．これは，2 の補数で表現されたセンサーデータの値を符号を勘案した値に変換する際の計算量が多く，処理に膨大な時間がかかることで適切な大きさでの分析が行えなかったためである．しかし，OBD2 端末には Phone データにはない変数が多く含まれており，これらの変数を用いれば，より精緻な運転者の特性の把握が可能になると考えられる．

参考文献

[1] Eboli, L., Mazzulla, G., & Pungillo, G. (2016). Combining speed and acceleration to define car users ’ safe or unsafe driving behaviour. *Transportation research part C: emerging technologies*, **68**, 113-125.

[2] Hoberock, L. L. (1977). A survey of longitudinal acceleration comfort studies in ground transportation vehicles. *Journal of Dynamic Systems, Measurement, and Control*, **99**, 76-84.

[3] Klauer, S. G., Dingus, T. A., Neale, V. L., Sudweeks, J. D., & Ramsey, D. J. (2009). Comparing real-world behaviors of drivers with high versus low rates of crashes and near crashes (No. DOT-HS-811-091).

[4] NEC(2018). 「新潟発　舗装道路の損傷を AI で判定　点検コストを大幅削減」.https://wisdom.nec.com/ja/collaboration/2018092701/index.html.2019 年 3 月 22 日アクセス

[5] 石井誉仁 (2018). 「ディープラーニングで“道路のひび割れ”を検知する」.https://www.nttpc.co.jp/gpu/article/technical05.html. 2019 年 3 月 22 日アクセス

[6] 木山昇, 高橋利光, 祖父江恒夫, 相川哲盛 (2014). 傾斜したスマートフォンによる自動車の 3 軸加速度算出手法. マルチメディア, 分散協調とモバイルシンポジウム 2014 論文集, 16-23.

[7] 西堀泰英, 稲垣具志, 加知範康, 安藤良輔, 三村泰広 (2010). 自動車走行時の加速度発生状況と交通事故発生箇所の関連分析. 土木計画学研究・講演集, 42.

[8] 畠中秀人, 平沢隆之, 真部泰幸, 渡邊寧, 井上洋, 竹中憲郎 (2007). プローブデータを活用した安全走行支援サービスに関する検討. 第 6 回 ITS シンポジウム, 315-319.

[9] 松下穂高, 林高樹 (2018). ETC2.0 プローブデータを活用したドライバーの危険運転度スコアリング・システムの提案.PACIS2018 主催記念特別全国研究発表大会要旨集 , 79-82.

[10] 横関俊也, 森健二, 矢野伸裕, 萩田賢司, 牧下寛 (2012). 雨量と事故データの分析からみた高速道路における安全な速度. 土木学会論文集 D3 (土木計画学), **68**, 1309-1317.

[11] 加藤秀樹 (2012). 「プローブ (車両探査) データからみたヒヤリハットの実態-有効な危険箇所抽出法の検討-」.https://www.ttri.or.jp/cms/wp-content/uploads/2018/06/9ed37d790dbe4bff664e190f3ad95629.pdf.2019 年 3 月 21 日アクセス

付 録A　本稿における分析と対応するソースコード

本章では，分析に用いたソースコードを示す．ソースコードのRファイルは，各節と対応した形でB1-1.RからB6.Rというファイル名で別ファイルにて与える．

A.1　3.4節における基礎分析

A.1.1　3.4.1節における kaisen080 の基礎分析

探索的な分析を行い，データの様子を確認したいため，まずはDBの最初の10,000行を分析対象として考える．このようなデータがphone_limit10000.csvとファイルで与えられていることを想定する．

ソースコード A.1: 3.4.1 節における基礎分析

```
## データの取り込み ##
# csvファイルをパッケージdata.tableのfread関数を用いて取り込む.
# パッケージを読み込む
library(data.table)
# phoneデータを取り込む
phone <- fread("phone_limit10000.csv",encoding="UTF-8")
# phoneの変数timeをcharacter型から時刻型に変更
phone$time <- as.data.frame(as.POSIXlt(phone$time))


## 欠損値処理
phone <- phone[緯度 != 0 & 経度 != 0,]


## 日付ごとの走行時間の基礎集計##
library(dplyr) #データ処理を簡単にするためのパッケージ
# 日付をカウントするための準備
date <- substr(phone$time,1,10) #日付だけを取り出す（時間を削除）
names(date) <- date
head(date)
phone <- data.frame(phone,date)
```

```
# 日付をカウントすることで，その日に何秒車の走行ログデータがあるのか調べる
phone %>%
  group_by(date) %>% # 集約単位は”date”
  summarise(count=n()) # dateが何個あるのかカウントする
# head(phone) # 最初の6行を確認する


## sensorデータの基礎集計 ##
# 例として，2016−02−16の走行データを見てみる
EX <- subset(phone, grepl("2016-02-16", phone$time)) # 2016−02−16の走行データがある行を取り上げる
# for文を回して，横持ちのデータを縦持ちに変換する(時間がかかります)
kekka <- data.frame(NULL) # 初期化
for(i in 1:nrow(EX)) {
  x <- t(EX[i,c(6:105)]) # i行目のx_senを縦持ちに変換
  y <- t(EX[i,c(106:205)]) # i行目のy_senを縦持ちに変換
  z <- t(EX[i,c(206:305)]) # i行目のz_senを縦持ちに変換
  datetime <- ((i-1)*100+1):(i*100) #(i−1)*100+1番目からi*100番目まで，その番号を振る
  colnames(x) <- "x_sen" # 変数名をx_senとする
  colnames(y) <- "y_sen" # 変数名をy_senとする
  colnames(z) <- "z_sen" # 変数名をz_senとする
  j = cbind(data.frame(datetime),x,y,z) #オブジェクトのdatetime ,x,y, zを横に並べて結合
  kekka <- rbind(kekka,j) #オブジェクトのkekkaとjを縦に並べて結合
  # print(i) #for文が今何番目の処理をしているのかを可視化させる
}
rownames(kekka) <- 1:nrow(kekka)
kekka_name <- c("time_series","x_sen","y_sen","z_sen")
names(kekka) <- kekka_name


## 分散の計算 ##
var(kekka$x_sen) # xセンサー
var(kekka$y_sen) # yセンサー
var(kekka$z_sen) # zセンサー


## センサーの挙動をマッピングする ##
library(ggplot2) #グラフをきれいに描画するためのパッケージ
# グラフ描画のための準備
```

```
g <- ggplot(kekka, aes(time_series))
g_x <- geom_line(aes(y = x_sen, colour = "x_accel"),alpha = 0.7)
g_y <- geom_line(aes(y = y_sen, colour = "y_accel"),alpha = 0.7)
g_z <- geom_line(aes(y = z_sen, colour = "z_accel"),alpha = 0.7)
# グラフ描画
plot(g + g_y + ggtitle("x_sensor")) # xセンサーのグラフ
plot(g + g_y + ggtitle("y_sensor")) # yセンサーのグラフ
plot(g + g_z + ggtitle("z_sensor")) # zセンサーのグラフ
plot(g + g_x + g_y + g_z + ylab("x and y and z sen") + ggtitle("x
    and y and z sen")) # x,y,z3つ合わせたセンサーのグラフ
```

以上のスクリプトにより，表 3.9，図 3.13，図 3.14，図 3.15，図 3.16 が出力される．

A.1.2 3.4.2 節における kaisen082 の基礎分析

今回，本分析で利用している 2016 年 1 月 15 日のセンサーデータを分析対象として考える．このようなデータが kaisen_082.csv とファイルで与えられていることを想定する．

ソースコード A.2: 3.4.1 節における基礎分析

```
## データの取り込み ##
# csvファイルをパッケージdata.tableのfread関数を用いて取り込む.
# パッケージを読み込む
library(data.table)
# phoneデータを取り込む
phone <- fread("kaisen_082.csv",encoding="UTF-8")
# phoneの変数timeをcharacter型から時刻型に変更
phone$time <- as.data.frame(as.POSIXlt(phone$time))


## 欠損値処理
phone <- phone[緯度 != 0 & 経度 != 0,]


## 日付ごとの走行時間の基礎集計##
library(dplyr) #データ処理を簡単にするためのパッケージ
# 日付をカウントするための準備
date <- substr(phone$time,1,10) #日付だけを取り出す（時間を削除）
names(date) <- date
head(date)
phone <- data.frame(phone,date)
# 日付をカウントすることで，その日に何秒車の走行ログデータがあるの
    か調べる
phone %>%
```

```
24    group_by(date) %>% # 集約単位は”date”
25    summarise(count=n()) # dateが何個あるのかカウントする
26  # head(phone) # 最初の6行を確認する
27
28
29  ## sensorデータの基礎集計 ##
30  # 例として，2016−01−15の走行データを見てみる
31  EX <- subset(phone, grepl("2016-01-15", phone$time)) # 2016−01−15
        の走行データがある行を取り上げる
32  # for文を回して，横持ちのデータを縦持ちに変換する(時間がかかりま
        す)
33  kekka <- data.frame(NULL) # 初期化
34  for(i in 1:nrow(EX)) {
35    x <- t(EX[i,c(6:105)]) # i行目のx_senを縦持ちに変換
36    y <- t(EX[i,c(106:205)]) # i行目のy_senを縦持ちに変換
37    z <- t(EX[i,c(206:305)]) # i行目のz_senを縦持ちに変換
38    datetime <- ((i-1)*100+1):(i*100) #(i−1)*100+1番目からi*100番目
          まで，その番号を振る
39    colnames(x) <- "x_sen" # 変数名をx_senとする
40    colnames(y) <- "y_sen" # 変数名をy_senとする
41    colnames(z) <- "z_sen" # 変数名をz_senとする
42    j = cbind(data.frame(datetime),x,y,z) #オブジェクト
          のdatetime ,x,y, zを横に並べて結合
43    kekka <- rbind(kekka,j) #オブジェクトのkekkaとjを縦に並べて結合
44    # print(i) #for文が今何番目の処理をしているのかを可視化させる
45  }
46  rownames(kekka) <- 1:nrow(kekka)
47  kekka_name <- c("time_series","x_sen","y_sen","z_sen")
48  names(kekka) <- kekka_name
49
50
51  ## 分散の計算 ##
52  var(kekka$x_sen) # xセンサー
53  var(kekka$y_sen) # yセンサー
54  var(kekka$z_sen) # zセンサー
55
56
57  ## センサーの挙動をマッピングする ##
58  library(ggplot2) #グラフをきれいに描画するためのパッケージ
59  # グラフ描画のための準備
60  g <- ggplot(kekka, aes(time_series))
61  g_x <- geom_line(aes(y = x_sen, colour = "x_accel"),alpha = 0.7)
62  g_y <- geom_line(aes(y = y_sen, colour = "y_accel"),alpha = 0.7)
```

```
63 g_z <- geom_line(aes(y = z_sen, colour = "z_accel"),alpha = 0.7)
64 # グラフ描画
65 plot(g + g_y + ggtitle("x_sensor")) # xセンサーのグラフ
66 plot(g + g_y + ggtitle("y_sensor")) # yセンサーのグラフ
67 plot(g + g_z + ggtitle("z_sensor")) # zセンサーのグラフ
68 plot(g + g_x + g_y + g_z + ylab("x and y and z sen") + ggtitle("x
       and y and z sen")) # x,y,z3つ合わせたセンサーのグラフ
```

以上のスクリプトにより，表 3.4.2，図 3.17，図 3.18，図 3.19，図 3.20 が出力される．

A.1.3 3.4.3 節における kaisen084 の基礎分析

今回，本分析で利用している 2016 年 6 月 23 日のセンサーデータを分析対象として考える．このようなデータが kaisen084_sunny.csv とファイルで与えられていることを想定する．

ソースコード A.3: 3.4.1 節における基礎分析

```
1  ## データの取り込み ##
2  # kaisen084_sunnyデータを取り込む
3  kaisen084_sunny <- read.csv("kaisen084_sunny.csv")
4  ## sensorデータの基礎集計 ##
5  # センサーデータの平均的な挙動を確認するため，三軸それぞれの1秒ごと
       との加速度の平均を計算する
6  # rowMeansで三軸それぞれの1秒ごとの加速度の平均を取得できる
7  # それらをkaisen084_sunnyに新たな変数x_sen，y_sen，z_
       senとして格納する
8  kaisen084_sunny$x_sen<-rowMeans(kaisen084_sunny[,6:105])
9  kaisen084_sunny$y_sen<-rowMeans(kaisen084_sunny[,106:205])
10 kaisen084_sunny$z_sen<-rowMeans(kaisen084_sunny[,206:305])
11 ## センサーの挙動をマッピングする ##
12 library(ggplot2) # グラフをきれいに描画するためのパッケージ
13 # グラフ描画のための準備
14 time_series <- 1:nrow(kaisen084_sunny)
15 g <- ggplot(kaisen084_sunny, aes(time_series))
16 g_x <- geom_line(aes(y =x_sen, colour="x_accel"),alpha=0.7)
17 g_y <- geom_line(aes(y =y_sen, colour = "y_accel"),alpha=0.7)
18 g_z <- geom_line(aes(y =z_sen, colour = "z_accel"),alpha=0.7)
19 # グラフ描画
20 plot(g + g_x + ggtitle("x_sensor")) # xセンサーのグラフ
21 plot(g + g_y + ggtitle("y_sensor")) # yセンサーのグラフ
22 plot(g + g_z + ggtitle("z_sensor")) # zセンサーのグラフ
23 plot(g + g_x + g_y + g_z + ylab("x and y and z sen") + ggtitle("x
       and y and z sen")) # x,y,z3つ合わせたセンサーのグラフ
```

以上のスクリプトにより，表 3.13，図 3.21，図 3.22，図 3.23，図 3.24 が出力される．

A.2　4.2節における分析

ソースコード A.4: 地図に描画する

```
1  ################### 前処理始め ###################
2  ## データの取り込み ##
3  # パッケージを読み込む
4  library(data.table)
5  # phoneデータを取り込む
6  phone <- fread("phone_limit10000.csv",encoding="UTF-8")
7  # phoneの変数timeをcharacter型から時刻型に変更
8  phone$time <- as.data.frame(as.POSIXlt(phone$time))
9
10
11 ## 欠損値処理
12 phone <- phone[緯度 != 0 & 経度 != 0,]
13 ################### 前処理終わり ###################
14
15
16 ## 地図にヒートマップを作成するための，新たな変数を用意する．
17 # z軸の加速度センサーにおける，1秒ごとの分散を求める(変数var_
       zを作成，少し時間がかかります)．
18 var_z <- data.frame(NULL)
19 for(i in 1:nrow(phone)){
20   c = var(c(as.numeric(phone[i,106:205])))
21   var_z <- rbind(var_z,c)
22 }
23 colnames(var_z) <- "var_z"
24
25 # var_zをphoneデータの入っているオブジェクトphoneに加える
26 phone <- data.frame(phone,var_z)
27
28
29 ## 地図に描画
30 library(leaflet) #地図に描画するためのパッケージ
31 # OSM(OpenStreetMap)または，GSI(国土地理院)の地図を用いて
       z軸の分散をヒートマップで描く(注:インターネットの接続が必要です
       .)
32 pal <- colorNumeric("OrRd", phone$var_z)
33 leaflet(phone) %>%
34   addTiles(group="OSM") %>% #OSMの地図を用いる
35   addTiles("http://cyberjapandata.gsi.go.jp/xyz/std/{z}/{x}/{y}.
       png", group="GSI") %>% # GSIの地図を用いる
```

```
36    addCircleMarkers(~経度, ~緯度
          , color = ~pal(var_z), group = "circles") %>%
37    addLegend(pal = pal, values = ~var_z, group = "circles",
          position = "bottomleft") %>%
38    addLayersControl(overlayGroups = c("circles"),baseGroups=c("OSM"
          ,"GSI"))
39
40
41
42
43  ## 分散値が大きいところだけピックアップして地図に描画する
44
45  # 分散値がどう分布しているか, 箱ひげ図で確認
46  boxplot(var_z)
47
48  # 分散の閾値をアドホックに設定
49  library(dplyr)
50  phone2 <- dplyr::filter(phone,phone$var_z > 2.5) # 分散の閾値を設
        定する. 左記のスクリプト「dplyr::filter(phone,phone$var_z >
        hogehoge)」の
        hogehogeの部分の数値を変えると分散値の閾値を再設定可能.
51
52
53  ## 地図に描画
54  library(leaflet) # 地図に描画するためのパッケージ
55  # OSM(OpenStreetMap)または, GSI(国土地理院)の地図を用いて
        z軸の分散をヒートマップで描く(注:インターネットの接続が必要です
        .)
56  pal <- colorNumeric("OrRd", phone$var_z)
57  leaflet(phone2) %>%
58    addTiles(group="OSM") %>% #OSMの地図を用いる
59    addTiles("http://cyberjapandata.gsi.go.jp/xyz/std/{z}/{x}/{y}.
          png", group="GSI") %>% # GSIの地図を用いる
60    addCircleMarkers(~経度, ~緯度
          , color = ~pal(var_z), group = "circles") %>%
61    addLegend(pal = pal, values = ~phone$var_z, group = "circles",
          position = "bottomleft") %>%
62    addLayersControl(overlayGroups = c("circles"),baseGroups=c("OSM"
          ,"GSI"))
```

このスクリプトにより，図 4.3 が出力される.

A.3　4.3 節における分析

経路の出力を行う.

ソースコード A.5: 地図に描画する

```
##雨の日と晴れの日の経路を示す##

# kaisen084の雨の日と晴れの日を使用する
kaisen084_rain <- read.csv("kaisen084_rain.csv",encoding="UTF-8")
kaisen084_sunny <- read.csv("kaisen084_sunny.csv",encoding="UTF-8"
    )
# leadlet使用
library(leaflet)
# 雨の日の経路
atr <- "<a href='http://maps.gsi.go.jp/development/ichiran.html'
    target='_blank'>地理院タイル</a>"
leaflet(kaisen084_rain) %>%
    addTiles(group="OSM") %>%
    addCircleMarkers(lng=~kaisen084_rain$経度
        ,lat=~kaisen084_rain$緯度) %>%
    addTiles("http://cyberjapandata.gsi.go.jp/xyz/std/{z}/{x}/{y}.
        png",attribution = atr, group="GSI") %>%
    addLayersControl(baseGroups=c("OSM","GSI"), options=
        layersControlOptions(collapsed = FALSE))
# 晴れの日の経路
leaflet(kaisen084_sunny) %>%
    addTiles(group="OSM") %>%
    addCircleMarkers(lng=~kaisen084_sunny$経度
        ,lat=~kaisen084_sunny$緯度) %>%
    addTiles("http://cyberjapandata.gsi.go.jp/xyz/std/{z}/{x}/{y}.
        png",attribution = atr, group="GSI") %>%
    addLayersControl(baseGroups=c("OSM","GSI"), options=
        layersControlOptions(collapsed = FALSE))
```

このスクリプトにより，図 4.25，4.26 が出力される.

ソースコード A.6: 地図に描画する

```
##x軸の解釈で使用した図##
library(readr)
# データの読み込み
# kaisen084の2016/6/23を使用する
kaisen084_sunny<-read.csv("kaisen084_sunny.csv")
# 中心化を行う為，x軸の平均を求める
kaisen084_sunny$x_ave<-rowMeans(kaisen084_sunny[,6:105])
```

```
# x軸の平均による中心化を行った
kaisen084_sunny[,6:105]<-kaisen084_sunny[,6:105]-kaisen084_sunny$x
    _ave
# 緯度・経度のデータの作成
lon_lat<-kaisen084_sunny[,4:5]
# エラー対策の為，アルファベット表記に変更
colnames(lon_lat)<-c("lattitude","longitude")
# 閾値である0.3より大きいものの個数を求める
x_sunny<- kaisen084_sunny[,6:105]>0.3
# この時点では，0と1だけのデータ
x_sunny1<-1*x_sunny
# 合算することで個数を算出
x_sunny1<- rowSums(x_sunny1)
x_sunny2<- cbind(kaisen084_sunny$time,x_sunny1)
x_sunny2<- cbind(x_sunny2,lon_lat)
x_sunny2<-as.data.frame(x_sunny2)
colnames(x_sunny2)<-c("time","n_over","lattitude","longitude")
library(leaflet)
# leafletを使って図に示す．色など諸々の設定
leaflet() %>% addTiles() %>% addLegend(
  position = "bottomright",
  colors = rgb(t(col2rgb(palette())) / 255),
  labels = palette(), opacity = 1,
  title = "An Obvious Legend"
)
# 閾値超えた回数でプロット
df <- local({
  n <- nrow(x_sunny2); x <- x_sunny2$longitude; y <- x_sunny2$
      lattitude
  z <- x_sunny2$n_over
  data.frame(x, y, z)
})
pal <- colorNumeric("OrRd", df$z)
leaflet(df) %>%
  addTiles() %>%
  addCircleMarkers(~x, ~y, color = ~pal(z), group = "circles") %>%
  addLegend(pal = pal, values = ~z, group = "circles", position =
      "bottomleft") %>%
  addLayersControl(overlayGroups = c("circles"))

##y軸の解釈で使用した図##
# データは先ほどのkaisen084_sunnyを使用する
```

```
# 緯度・経度のデータの作成
lon_lat<-kaisen084_sunny[,4:5]
# エラー対策の為，アルファベット表記に変更
colnames(lon_lat)<-c("lattitude","longitude")
# 閾値である0.3より大きいものの個数を求める
y_sunny<- kaisen084_sunny[,106:205]>0.3
# この時点では，0と1だけのデータ
y_sunny1<-1*y_sunny
# 合算することで個数を算出
y_sunny1<- rowSums(y_sunny1)
y_sunny2<- cbind(kaisen084_sunny$time,y_sunny1)
y_sunny2<- cbind(y_sunny2,lon_lat)
y_sunny2<-as.data.frame(y_sunny2)
colnames(y_sunny2)<-c("time","n_over","lattitude","longitude")
library(leaflet)
# leafletを使って図に示す．色など諸々の設定
leaflet() %>% addTiles() %>% addLegend(
  position = "bottomright",
  colors = rgb(t(col2rgb(palette())) / 255),
  labels = palette(), opacity = 1,
  title = "An Obvious Legend"
)
# 閾値超えた回数でプロット
df <- local({
  n <- nrow(y_sunny2); x <- y_sunny2$longitude; y <- y_sunny2$
     lattitude
  z <- y_sunny2$n_over
  data.frame(x, y, z)
})
pal <- colorNumeric("OrRd", df$z)
leaflet(df) %>%
  addTiles() %>%
  addCircleMarkers(~x, ~y, color = ~pal(z), group = "circles") %>%
  addLegend(pal = pal, values = ~z, group = "circles", position =
     "bottomleft") %>%
  addLayersControl(overlayGroups = c("circles"))
```

このスクリプトにより，図 4.27，4.28 が出力される．

ソースコード A.7: 地図に描画する

```
##x軸の危険運転スコア(雨天時)##
# パッケージの読み込み
library(readr)
```

```
4  # データの読み込み
5  # kaisen084の雨の日，2016/3/23を使用する
6  kaisen084_rain<-read.csv("kaisen084_rain.csv")
7  # 中心化を行う為，x軸の平均を求める
8  kaisen084_rain$x_ave<-rowMeans(kaisen084_rain[,6:105])
9  # x軸の平均による中心化を行う
10 kaisen084_rain[,6:105]<-kaisen084_rain[,6:105]-kaisen084_rain$x_
       ave
11 # 緯度・経度のデータの作成
12 lon_lat<-kaisen084_rain[,4:5]
13 # エラー対策の為，アルファベット表記に変更
14 colnames(lon_lat)<-c("lattitude","longitude")
15 # 閾値である−0.3より小さいものの個数を求める
16 dan_x_rain<- kaisen084_rain[,6:105]<(-0.3)
17 # この時点では，0と1だけのデータ
18 dan_x_rain1<-1*dan_x_rain
19 # 合算することで個数を算出
20 dan_x_rain1<- rowSums(dan_x_rain1)
21 dan_x_rain2<- cbind(kaisen084_rain$time,dan_x_rain1)
22 dan_x_rain2<- cbind(dan_x_rain2,lon_lat)
23 dan_x_rain2<-as.data.frame(dan_x_rain2)
24 colnames(dan_x_rain2)<-c("time","n_over","lattitude","longitude")
25 library(leaflet)
26 # leafletを使って図に示す．色など諸々の設定
27 leaflet() %>% addTiles() %>% addLegend(
28   position = "bottomright",
29   colors = rgb(t(col2rgb(palette())) / 255),
30   labels = palette(), opacity = 1,
31   title = "An Obvious Legend"
32 )
33 # 閾値超えた回数でプロット
34 df <- local({
35   n <- nrow(dan_x_rain2); x <- dan_x_rain2$longitude; y <- dan_x_
       rain2$lattitude
36   z <- dan_x_rain2$n_over
37   data.frame(x, y, z)
38 })
39 pal <- colorNumeric("OrRd", df$z)
40 leaflet(df) %>%
41   addTiles() %>%
42   addCircleMarkers(~x, ~y, color = ~pal(z), group = "circles") %>%
43   addLegend(pal = pal, values = ~z, group = "circles", position =
```

```
      "bottomleft") %>%
  addLayersControl(overlayGroups = c("circles"))

##x軸の危険運転スコア(晴天時)##
# パッケージの読み込み
library(readr)
#データの読み込み
# kaisen084の晴れの日，2016/6/23を使用する
kaisen084_sunny<-read.csv("kaisen084_sunny.csv")
# 中心化を行う為，x軸の平均を求める
kaisen084_sunny$x_ave<-rowMeans(kaisen084_sunny[,6:105])
# x軸の平均による中心化を行う
kaisen084_sunny[,6:105]<-kaisen084_sunny[,6:105]-kaisen084_sunny$x
    _ave
# 緯度・経度のデータの作成
lon_lat<-kaisen084_sunny[,4:5]
# エラー対策の為，アルファベット表記に変更
colnames(lon_lat)<-c("lattitude","longitude")
# 閾値である−0.3より小さいものの個数を求める
dan_x_sunny<- kaisen084_sunny[,6:105]<(-0.3)
# この時点では，0と1だけのデータ
dan_x_sunny1<-1*dan_x_sunny
# 合算することで個数を算出
dan_x_sunny1<- rowSums(dan_x_sunny1)
dan_x_sunny2<- cbind(kaisen084_sunny$time,dan_x_sunny1)
dan_x_sunny2<- cbind(dan_x_sunny2,lon_lat)
dan_x_sunny2<-as.data.frame(dan_x_sunny2)
colnames(dan_x_sunny2)<-c("time","n_over","lattitude","longitude")
library(leaflet)
# leafletを使って図に示す．色など諸々の設定
leaflet() %>% addTiles() %>% addLegend(
  position = "bottomright",
  colors = rgb(t(col2rgb(palette())) / 255),
  labels = palette(), opacity = 1,
  title = "An Obvious Legend"
)
# 閾値超えた回数でプロット
df <- local({
  n <- nrow(dan_x_sunny2); x <- dan_x_sunny2$longitude; y <- dan_x
     _sunny2$lattitude
  z <- dan_x_sunnyd2$n_over
  data.frame(x, y, z)
```

```
83 })
84 pal <- colorNumeric("OrRd", df$z)
85 leaflet(df) %>%
86   addTiles() %>%
87   addCircleMarkers(~x, ~y, color = ~pal(z), group = "circles") %>%
88   addLegend(pal = pal, values = ~z, group = "circles", position =
       "bottomleft") %>%
89   addLayersControl(overlayGroups = c("circles"))
```

このスクリプトにより，図 4.29，4.30 が出力される．

ソースコード A.8: 地図に描画する

```
1  ##一定値より高い運転スコアの取り出しとプロット(今回は
       x軸に関してのみ行います)##
2  # 先ほど作成したdan_x_rain2，dan_x_sunny2のn_
       overにはそれぞれ雨天時と晴天時の危険運転スコアなどが格納されている

3  # 初めに雨天時のプロットから行う
4  dan_x_rain2_limit<-as.data.frame(NULL)
5  # kaisen084の雨の日，2016/3/23を使用する
6  kaisen084_rain<-read.csv("kaisen084_rain.csv")
7  # 中心化を行う為，x軸の平均を求める
8  kaisen084_rain$x_ave<-rowMeans(kaisen084_rain[,6:105])
9  # x軸の平均による中心化を行う
10 kaisen084_rain[,6:105]<-kaisen084_rain[,6:105]-kaisen084_rain$x_
       ave
11 # 緯度・経度のデータの作成
12 lon_lat<-kaisen084_rain[,4:5]
13 # エラー対策の為，アルファベット表記に変更
14 colnames(lon_lat)<-c("lattitude","longitude")
15 # 閾値である−0.3より小さいものの個数を求める
16 dan_x_rain<- kaisen084_rain[,6:105]<(-0.3)
17 # この時点では，0と1だけのデータ
18 dan_x_rain1<-1*dan_x_rain
19 # 合算することで個数を算出
20 dan_x_rain1<- rowSums(dan_x_rain1)
21 dan_x_rain2<- cbind(kaisen084_rain$time,dan_x_rain1)
22 dan_x_rain2<- cbind(dan_x_rain2,lon_lat)
23 dan_x_rain2<-as.data.frame(dan_x_rain2)
24 colnames(dan_x_rain2)<-c("time","n_over","lattitude","longitude")
25 # 危険運転スコアが30を超えるもののみをdan_x_rain2_limitに格納する
       (30の部分を自由な値に設定して下限を調節可能である)
26 dan_x_rain2_limit<-subset(dan_x_rain2,dan_x_rain2$n_over>30)
```

```
27 # 閾値超えた回数でプロット
28 df <- local({
29 n <- nrow(dan_x_rain2_limit); x <- dan_x_rain2_limit$longitude; y
       <- dan_x_rain2_limit$lattitude
30 z <- dan_x_rain2_limit$n_over
31 data.frame(x, y, z)
32  })
33 pal <- colorNumeric("OrRd", df$z)
34 leaflet(df) %>%
35 addTiles() %>%
36 addCircleMarkers(~x, ~y, color = ~pal(z), group = "circles") %>%
37 addLegend(pal = pal, values = ~z, group = "circles", position = "
       bottomleft") %>%
38 addLayersControl(overlayGroups = c("circles"))
39
40 # 次に晴天時のプロットから行う
41 dan_x_sunny2_limit<-as.data.frame(NULL)
42 # データの読み込み
43 # kaisen084の晴れの日, 2016/6/23を使用する
44 kaisen084_sunny<-read.csv("kaisen084_sunny.csv")
45 # 中心化を行う為, x軸の平均を求める
46 kaisen084_sunny$x_ave<-rowMeans(kaisen084_sunny[,6:105])
47 # x軸の平均による中心化を行う
48 kaisen084_sunny[,6:105]<-kaisen084_sunny[,6:105]-kaisen084_sunny$x
       _ave
49 # 緯度・経度のデータの作成
50 lon_lat<-kaisen084_sunny[,4:5]
51 # エラー対策の為, アルファベット表記に変更
52 colnames(lon_lat)<-c("lattitude","longitude")
53 # 閾値である−0.3より小さいものの個数を求める
54 dan_x_sunny<- kaisen084_sunny[,6:105]<(-0.3)
55 # この時点では, 0と1だけのデータ
56 dan_x_sunny1<-1*dan_x_sunny
57 # 合算することで個数を算出
58 dan_x_sunny1<- rowSums(dan_x_sunny1)
59 dan_x_sunny2<- cbind(kaisen084_sunny$time,dan_x_sunny1)
60 dan_x_sunny2<- cbind(dan_x_sunny2,lon_lat)
61 dan_x_sunny2<-as.data.frame(dan_x_sunny2)
62 colnames(dan_x_sunny2)<-c("time","n_over","lattitude","longitude")
63 # 危険運転スコアが30を超えるもののみをdan_x_sunny2_limitに格納する
       (30の部分を自由な値に設定して下限を調節可能である)
64 dan_x_sunny2_limit<-subset(dan_x_sunny2,dan_x_sunny2$n_over>30)
```

```
65 # 閾値超えた回数でプロット
66 df <- local({
67 n <- nrow(dan_x_sunny2_limit); x <- dan_x_sunny2_limit$longitude;
       y <- dan_x_sunny2_limit$lattitude
68 z <- dan_x_sunny2_limit$n_over
69 data.frame(x, y, z)
70 })
71 pal <- colorNumeric("OrRd", df$z)
72 leaflet(df) %>%
73 addTiles() %>%
74 addCircleMarkers(~x, ~y, color = ~pal(z), group = "circles") %>%
75 addLegend(pal = pal, values = ~z, group = "circles", position = "
       bottomleft") %>%
76 addLayersControl(overlayGroups = c("circles"))
```

このスクリプトにより，図 4.47，4.48 が出力される．上のコードの 30 を 40 に変更することにより図 4.49，4.50 が出力される．

ソースコード A.9: 地図に描画する

```
1  ##y軸の危険運転スコア(雨天時)##
2  #データの読み込み
3  kaisen084_rain<-read.csv("kaisen084_rain.csv")
4  # 緯度・経度のデータの作成
5  lon_lat<-kaisen084_rain[,4:5]
6  # エラー対策の為，アルファベット表記に変更
7  colnames(lon_lat)<-c("lattitude","longitude")
8  # 閾値である−0.3より小さいものの個数を求める
9  dan_y_rain<- kaisen084_rain[,106:205]<(-0.3)
10 dan_y_rain1<-1*dan_y_rain        # この時点では，0と1だけのデータ
11 dan_y_rain1<- rowSums(dan_y_rain1)  # 合算することで個数を算出
12 # 同じく閾値である0.3より大きいものの個数を求める
13 dan_y_rain<- kako[,106:205]>0.3
14 dan_y_rain2<-1*dan_y_rain
15 dan_y_rain2<- rowSums(dan_y_rain2)
16 # 閾値は±0.3なのでこれら二つを合計する
17 dan_y_rain3<-cbind(dan_y_rain1,dan_y_rain2)
18 dan_y_rain3<-rowSums(dan_y_rain3)
19 dan_y_rain4<- cbind(kaisen084_rain$time,dan_y_rain3)
20 dan_y_rain4<- cbind(dan_y_rain4,lon_lat)
21 dan_y_rain4<-as.data.frame(dan_y_rain4)
22 colnames(dan_y_rain4)<-c("time","n_over","lattitude","longitude")
23 library(leaflet)
24 # leafletを使って図に示す．色など諸々の設定
```

```
leaflet() %>% addTiles() %>% addLegend(
  position = "bottomright",
  colors = rgb(t(col2rgb(palette())) / 255),
  labels = palette(), opacity = 1,
  title = "An Obvious Legend"
)
# 閾値超えた回数でプロット
df <- local({
  n <- nrow(dan_y_rain4); x <- dan_y_rain4$longitude; y <- dan_y_
      rain4$lattitude
  z <- dan_y_rain4$n_over
  data.frame(x, y, z)
})
pal <- colorNumeric("OrRd", df$z)
leaflet(df) %>%
  addTiles() %>%
  addCircleMarkers(~x, ~y, color = ~pal(z), group = "circles") %>%
  addLegend(pal = pal, values = ~z, group = "circles", position =
      "bottomleft") %>%
  addLayersControl(overlayGroups = c("circles"))

##y軸の危険運転スコア(晴天時)##
#データの読み込み
kaisen084_sunny<-read.csv("kaisen084_sunny.csv")
# 緯度・経度のデータの作成
lon_lat<-kaisen084_sunny[,4:5]
# エラー対策の為，アルファベット表記に変更
colnames(lon_lat)<-c("lattitude","longitude")
# 閾値である−0.3より小さいものの個数を求める
dan_y_sunny<- kaisen084_sunny[,106:205]<(-0.3)
dan_y_sunny1<-1*dan_y_sunny      # この時点では，0と1だけのデータ
dan_y_sunny1<- rowSums(dan_y_sunny1)  # 合算することで個数を算出
# 同じく閾値である0.3より大きいものの個数を求める
dan_y_sunny<- kaisen084_sunny[,106:205]>0.3
dan_y_sunny2<-1*dan_y_sunny
dan_y_sunny2<- rowSums(dan_y_sunny2)
# 閾値は±0.3なのでこれら二つを合計する
dan_y_sunny3<-cbind(dan_y_sunny1,dan_y_sunny2)
dan_y_sunny3<-rowSums(dan_y_sunny3)
dan_y_sunny4<- cbind(kaisen084_sunny$time,dan_y_sunny3)
dan_y_sunny4<- cbind(dan_y_sunny4,lon_lat)
dan_y_sunny4<-as.data.frame(dan_y_sunny4)
```

```
colnames(dan_y_sunny4)<-c("time","n_over","lattitude","longitude")
library(leaflet)
# leafletを使って図に示す．色など諸々の設定
leaflet() %>% addTiles() %>% addLegend(
  position = "bottomright",
  colors = rgb(t(col2rgb(palette())) / 255),
  labels = palette(), opacity = 1,
  title = "An Obvious Legend"
)
# 閾値超えた回数でプロット
df <- local({
  n <- nrow(dan_y_sunny4); x <- dan_y_sunny4$longitude; y <- dan_y
     _sunny4$lattitude
  z <- dan_y_sunny4$n_over
  data.frame(x, y, z)
})
pal <- colorNumeric("OrRd", df$z)
leaflet(df) %>%
  addTiles() %>%
  addCircleMarkers(~x, ~y, color = ~pal(z), group = "circles") %>%
  addLegend(pal = pal, values = ~z, group = "circles", position =
     "bottomleft") %>%
  addLayersControl(overlayGroups = c("circles"))
```

このスクリプトにより，図 4.37，4.38 が出力される．

ソースコード A.10: 地図に描画する

```
##負の閾値に基づいたx軸の危険運転スコア(合算したときの差)##
# パッケージの読み込み
library(readr)
## ---データの読み込み ---
# kaisen084の晴れの日，2016/6/23を使用する
kaisen084_sunny<-read.csv("kaisen084_sunny.csv")
# 中心化を行う為，x軸の平均を求める
kaisen084_sunny$x_ave<-rowMeans(kaisen084_sunny[,6:105])
# x軸の平均による中心化を行う
kaisen084_sunny[,6:105]<-kaisen084_sunny[,6:105]-kaisen084_sunny$x
    _ave
# 緯度・経度のデータの作成
lon_lat<-kaisen084_sunny[,4:5]
# エラー対策の為，アルファベット表記に変更
colnames(lon_lat)<-c("lattitude","longitude")
# 閾値である-0.3より小さいものの個数を求める
```

```
dan_x_sunny<- kaisen084_sunny[,6:105]<(-0.3)
# この時点では，0と1だけのデータ
dan_x_sunny1<-1*dan_x_sunny
# 合算することで個数を算出
dan_x_sunny1<- rowSums(dan_x_sunny1)
dan_x_sunny2<- cbind(kaisen084_sunny$time,dan_x_sunny1)
dan_x_sunny2<- cbind(dan_x_sunny2,lon_lat)
dan_x_sunny2<-as.data.frame(dan_x_sunny2)
colnames(dan_x_sunny2)<-c("time","n_over","lattitude","longitude")
# kaisen084の雨の日，2016/3/23を使用する
kaisen084_rain<-read.csv("kaisen084_rain.csv")
# 中心化を行う為，x軸の平均を求める
kaisen084_rain$x_ave<-rowMeans(kaisen084_rain[,6:105])
# x軸の平均による中心化を行う
kaisen084_rain[,6:105]<-kaisen084_rain[,6:105]-kaisen084_rain$x_
    ave
# 緯度・経度のデータの作成
lon_lat<-kaisen084_rain[,4:5]
# エラー対策の為，アルファベット表記に変更
colnames(lon_lat)<-c("lattitude","longitude")
# 閾値である−0.3より小さいものの個数を求める
dan_x_rain<- kaisen084_rain[,6:105]<(-0.3)
# この時点では，0と1だけのデータ
dan_x_rain1<-1*dan_x_rain
# 合算することで個数を算出
dan_x_rain1<- rowSums(dan_x_rain1)
dan_x_rain2<- cbind(kaisen084_rain$time,dan_x_rain1)
dan_x_rain2<- cbind(dan_x_rain2,lon_lat)
dan_x_rain2<-as.data.frame(dan_x_rain2)
colnames(dan_x_rain2)<-c("time","n_over","lattitude","longitude")

## データのマッチング
# 各々のデータに足して，各GPS座標における最大のスコアを出す
library(dplyr)
dan_x_rain2 %>%
  group_by(lattitude,longitude) %>%
  summarise(max_n_over = max(n_over), time=max(time)) %>%
  data.frame() -> dan_x_rain3
dan_x_sunny2 %>%
  group_by(lattitude,longitude) %>%
  summarise(max_n_over = max(n_over), time=max(time)) %>%
  data.frame() -> dan_x_sunny3
```

```
57 # ----------------------------------------
58 # 合わせたデータフレームを作る
59 n_sunny <- nrow(dan_x_sunny3)
60 n_rain <- nrow(dan_x_rain3)
61 dan_x <- rbind(dan_x_sunny3, dan_x_rain3)
62 # 天気を表す指示変数
63 dan_x$IsSunny <- c(rep(1,n_sunny), rep(0,n_rain))
64 # 距離を算出
65 dist_dan <- as.matrix(dist(dan_x[,c(1,2)]))
66 # ================================================
67 # 晴天時を基準に
68 # ================================================
69 # 各行ごとに距離が最小のindexを取得
70 apply(dist_dan[,seq(n_sunny+1, n_sunny+n_rain)], 1, which.min) ->
       mindis
71 rain_nov <- c()
72 for(i in mindis[1:n_sunny]){
73   rain_nov <- c(rain_nov, dan_x_rain3[i,"max_n_over"])
74   #print(i)
75 }
76
77 dan_x_match <- cbind(dan_x_sunny3[,-4], rain_nov)
78 colnames(dan_x_match) <- c("lattitude","longitude","sunny_nov","
       rain_nov")
79
80 dan_x_match$nov_diff <- dan_x_match$rain_nov - dan_x_match$sunny_
       nov
81
82
83 library(leaflet)
84 # leafletを使って図に示す. 色など諸々の設定
85 leaflet() %>% addTiles() %>% addLegend(
86   position = "bottomright",
87   colors = rgb(t(col2rgb(palette())) / 255),
88   labels = palette(), opacity = 1,
89   title = "An Obvious Legend"
90 )
91 # 閾値超えた回数でプロット
92 df <- local({
93   n <- nrow(dan_x_match); x <- dan_x_match$longitude; y <- dan_x_
         match$lattitude
94   z <- dan_x_match$nov_diff
```

```
95    data.frame(x, y, z)
96  })
97  pal <- colorNumeric("RdYlBu", df$z)
98  leaflet(df) %>%
99    addTiles() %>%
100   addCircleMarkers(~x, ~y, color = ~pal(z), group = "circles") %>%
101   addLegend(pal = pal, values = ~z, group = "circles", position =
        "bottomleft") %>%
102   addLayersControl(overlayGroups = c("circles"))
```

このスクリプトにより，図 4.51 が出力される．

ソースコード A.11: 地図に描画する

```
1   # スクレイピング 用のパッケージrvest
2   library(rvest)
3
4   # データの読み込み
5   kaisen084_rain <- read.csv("kaisen084_rain.csv",encoding="UTF-8")
6
7   # dateとして日付だけ取っておく
8   date <- substr(kaisen084_rain$time,1,10)
9
10  # kaisen084に日付を追加する
11  kaisen084_rain <- data.frame(kaisen084_rain,date)
12
13  # levelsで，どの日付に走ったかを見ることが出来る．
14  level <- levels(kaisen084_rain$date)
15  level_date <- as.Date(level)
16
17  # 年月日を切り分けて取得する
18  year <- substr(level_date,1,4)
19  month <- substr(level_date,6,7)
20  day <- substr(level_date,9,10)
21
22  # スクレイピングの為に箱を作っておく
23  y <- as.character(NULL)
24  data <- as.character(NULL)
25  media <- as.character(NULL)
26  success <- as.character(NULL)
27
28  # tenki.jp の年月日を挿れる所に，先ほど作った年月日を挿れて html
      を取得する
29  for (i in 1:length(level)){
```

```
30    d <- c(paste("https://tenki.jp/past/",year[i],"/",month[i],"/",
          day[i],"/amedas/6/31/63461.html",sep=""))
31    y <- append(y,d)
32  }
33
34  # その html から合計降雨量を取得する. media の4番目には合計降雨
        量が記載されている.
35  for(i in 1:length(level)){
36    data <- read_html(y[i])
37    media <- html_nodes(data,"td") %>%
38           html_text()
39    success <- append(success,media[4])
40  }
41
42  # 合計降雨量は文字で取得されているので数値型に変換する.
43  rainfall <- as.numeric(success)
44
45  # データフレームとして格納する.
46  data.frame(level_date,rainfall)
```

A.4 5.1節における分析

ソースコード A.12: 加速度の閾値が 0.3 と-0.3 のグラフ

```
1  # 範囲外にプロットが来るような値を設定
2  a = 100
3  V = 100
4  # 点をプロットする. ylimは描画されるy軸の範囲
5  plot(V,a,ylim=range(-8,8))
6  # y= 3, y = −3の直線を描く
7  abline(h = 3)
8  abline(h = -3)
```

ソースコード A.13: Laura et al.(2016) の式のグラフ

```
1  # Laura et al.(2016)の式をplotするためのコード
2  # g:重力加速度
3  g = 9.80665
4  # aは、正の方向から. bは負の方向からの線対称のグラフ
5  a <-function(V){g*(0.198*(V/100)^2-0.592*(V/100)+0.569)}
6  b <-function(V){(g*(0.198*(V/100)^2-0.592*(V/100)+0.569))*-1}
7  # 関数aを描画
```

```
plot(a,xlim=range(0,150),ylim=range(-8,8), ann = F)
# グラフを重ね合わせる
par(new=T)
# 関数bを描画
plot(b,xlim=range(0,150),ylim=range(-8,8), ann = F)
```

ソースコード A.14: 速さの作成からと加速度と速さのプロット

```
# データの読み込み
library(dplyr)
library(data.table)
library(geosphere)
#kaisen082のデータを読み込み
phone <- fread("phone_kaisen082.csv",encoding="UTF-8")
# 初期化
result_dist <- data.frame(NULL)

for(i in 1:nrow(phone)){
  keido1 <- as.numeric(phone[i,"経度"]) # ある地点の経度
  ido1 <- as.numeric(phone[i,"緯度"]) # ある地点の緯度
  bine1 <- c(keido1,ido1) # c(経度, 緯度)の作成

  keido2 <- as.numeric(phone[i + 1,"経度
      "]) # ある地点から1秒後の経度
  ido2 <- as.numeric(phone[i + 1,"緯度"]) # ある地点から1秒後の緯度
  bine2 <- c(keido2,ido2) # c(経度, 緯度)の作成

  dist <- (distGeo(bine1,bine2)/1000)/((1/60)*(1/60)) # 1秒の瞬間
      の速度(km/h)を算出 distGeo()/1000で1秒間の移動距離（
      km）算出，1行が1秒なので，(1/60)*(1/60)で単位を
      minをhourに変換する．さらにkm/hの計算を行う
  result_dist <- rbind(result_dist,dist) # 縦に結合していく
}

colnames(result_dist) <- "km_h" # 変数名をkm/hとする

phone <- data.frame(phone,result_dist) # dataframeに追加

# x軸方向のセンサーデータの速さと加速度のプロット
for (i in 6:105){
uu<-select(.data = phone, km_h,names(phone[i]))
plot(uu,ylab="a",xlim=range(0,150),ylim=range(-12,12), cex =0.5)
par(new=T)}

```

```
# y軸方向のセンサーデータの速さと加速度のプロット
for (i in 106:205){
uu<-select(.data = phone, km_h,names(phone[i]))
plot(uu,ylab="a",xlim=range(0,150),ylim=range(-12,12), cex =0.5)
par(new=T)}

# z軸方向のセンサーデータの速さと加速度のプロット
for (i in 206:305){
uu<-select(.data = phone, km_h,names(phone[i]))
plot(uu,ylab="a",xlim=range(0,150),ylim=range(-12,12), cex =0.5)
par(new=T)}
```

A.5 その他

A.5.1 obd2 データの変換

OBD2 端末にに記録される G センサーデータは，符号型の整数である．そのため，(2.1) 式による変換を行わなければならない．データが obd2_limit1000.csv とファイルで与えられていることを想定する．

ソースコード A.15: OBD2 端末からの G センサーデータの符号を勘案した値に変換

```
1  ################### 前処理始め ###################
2  # データの読み込み
3  library(data.table)
4  obd2 <- fread("obd2_limit1000.csv",encoding="UTF-8")
5
6  ## データ分析ができるようにするための前処理
7  # obd2のunit_idがinteger64になっているので，それを直す(64
       bitでは扱えないpackageがあったので．)
8  obd2$unit_id <- as.data.frame(as.numeric(obd2$unit_id))
9
10 ## 1900-01-01 00:00:00を含む列全て削除
11 obd2 <- subset(obd2, !grepl("1900-01-01 00:00:00", obd2$gps時刻))
12
13 ## 時刻型に変換
14 obd2$rtc時刻 <- as.data.frame(as.POSIXlt(obd2$rtc時刻))
15 obd2$gps時刻 <- as.data.frame(as.POSIXlt(obd2$gps時刻))
16 ################### 前処理終わり ###################
17
18 ## Gセンサーデータの変換（WARNING:処理に非常に時間がかかります．）
19 n <- nrow(obd2) #何行分析するか
```

```
conv <- data.frame(matrix(rep(NA, n), ncol=1)) # for文を回すために空の行列を作成
conv2 <- data.frame(NULL) # for文を回すために空のデータフレームを作成

for(j in 34:333) { # 列(sensor_x,y,zそれぞれ全て100個 ,34~133列にx軸 ,134~233列にy軸 ,234~333列にz軸が入っている)
  for(i in 1:n) { # 行,object名適宜変更
    if(obd2[i,..j] < 2^13){
      pri <- obd2[i,..j]/1024 # 2^13未満なら左記の処理を施す
    }else{
      pri <- (-(2^14 - obd2[i,..j]))/1024 ## 2^13以上なら左記の処理を施す
    }
    conv2 <- rbind.data.frame(conv2,pri)
    if(34 <= j && j <= 133){
      colnames(conv2) <- paste("g_sens_x_",j - 33,sep="") # xセンサーの列名を1/100ごとにつける
    }else if(134 <= j && j <= 233){
      colnames(conv2) <- paste("g_sens_y_",j - 133,sep="") # yセンサーの列名を1/100ごとにつける
    }else{
      colnames(conv2) <- paste("g_sens_z_",j - 233,sep="") # zセンサーの列名を1/100ごとにつける
    }
  }
  print(((j - 33)/300)*100) # 処理が何%終わったのかconsoleで明示する
  conv <- cbind.data.frame(conv,conv2) # conv2で作成した列をconvに加える
  conv2 <- data.frame(NULL) # 行のforを初期化する
}

conv <- conv[,-1] # for文を回すために作った空の行列を削除
obd2 <- data.frame(obd2[1:n,-34:-333],conv) # 2の補数で再現されたデータを削除し, 符号を勘案した値に変換したものに置き換える

## csvで変換後のデータを"obd2_new.csv"という名前で書き出す
write.csv(obd2,"obd2_new.csv")
```

A.5.2 天気データの取得

本文での分析では計算量の都合上，データが存在する日付の天気のみを取得したが，次に示すようなコードによって，2016 年から 2017 年の各日の積算降雨量を取得することができる．

ソースコード A.16: 2016 年から 2017 年の各日の積算降雨量を取得する

```
1
2  # 2016年から2017年の各日の積算降雨量
3  library(rvest)
4  datetime <- read.csv("datetime.csv",encoding="UTF-8") # 2016~2017
       年までのデータの読み込み
5  head(datetime)
6
7  # 年月日を切り分けて取得する
8  year <- substr(datetime$date,1,4)
9  month <- substr(datetime$date,6,7)
10 day <- substr(datetime$date,9,10)
11
12 # スクレイピングの為に箱を作っておく
13 y <- as.character(NULL)
14 data <- as.character(NULL)
15 media <- as.character(NULL)
16 success <- as.character(NULL)
17
18 # tenki.jp の年月日を挿れる所に，先ほど作った年月日を挿れて html
       を取得する
19 for (i in 1:nrow(datetime)){
20   d <- c(paste("https://tenki.jp/past/",year[i],"/",month[i],"/",
         day[i],"/amedas/6/31/63461.html",sep=""))
21   y <- append(y,d)
22 }
23
24 # その html から合計降雨量を取得する． media の4番目には合計降雨
       量が記載されている．
25 for(i in 1:nrow(datetime)){
26   data <- read_html(y[i])
27   media <- html_nodes(data,"td") %>%
28           html_text()
29   success <- append(success,media[4])
30 }
31
32 # 合計降雨量は文字で取得されているので数値型に変換する．
33 rainfall <- as.numeric(success)
```

```
34
35 # データフレームとして格納する.
36 data.frame(datetime,rainfall)
```

A.6 本レポートで用いたRパッケージ

本レポートでは，次に示すRのパッケージを用いた.

- geosphere
- data.table
- dplyr
- leaflet
- readr

これらをインストールするためには，ソースコードA.17を順次実行すれば良い.

ソースコードA.17: パッケージのインストール

```
1 install.packages("geosphere", dependencies=T)
2 install.packages("data.table", dependencies=T)
3 install.packages("dplyr", dependencies=T)
4 install.packages("leaflet", dependencies=T)
5 install.packages("readr", dependencies=T)
```