探索的財務ビッグデータ解析
―前処理，データラングリング，再現可能性―

地　道　正　行

要　旨

本稿では，Bureau van Dijk 社から提供されるデータベース Osiris から抽出された世界157カ国の全上場企業（一般事業会社，上場廃止企業含む）の主要財務情報（売上高，営業利益，総資産など84項目，33年分）の財務データファイル（財務ビッグデータ）をコンピュータで利用できるファイル形式に変換する工程（前処理）と，その整形されたファイルをデータ解析環境 R に読み込み，実際にデータ解析が行えるオブジェクト形式に変換する工程（データラングリング）を自動実行化することによって再現可能性を確保する方法を検討する．本稿を含む一連の研究では，Tukey（1977）によって提唱された探索的データ解析をデータサイエンスを実行するカーネルとして位置付けし，財務ビッグデータから何らかの意味のある情報・知見を得ることを試みる．

キーワード：探索的データ解析（Exploratory Data Analysis），ビッグデータ（Big Data），データサイエンス（Data Science），前処理（Preprocessing），データラングリング（Data Wrangling），再現可能性（Reproducibility）

I　はじめに

近年，「ビッグデータ」（big data）という用語で呼ばれる規模が大きく，かつ構造をもたず非定型のものを含むようなデータがビジネスにおける実際の業務として扱われており，その処理・分析の重要性が指摘されている（例えば，西田（2017）を参照）．また，このようなデータを利用し，科学的な

視点からデータを分析・解析することによって新たな知見を得るための（学問）分野を「データサイエンス」（data science）と呼ぶようになっている．なお，データサイエンスを定義づけることや，それに関連する文献については付録 A に与えている．

一般に，粗データ（raw data）は，さまざまな理由（フォーマットが不統一，欠損値，特殊記号の存在など）から，直接データ解析環境に読み込む（load）ことができず，分析・解析することは難しい．よって，まずは「読み込めるファイル形式」や「分析・解析できる（オブジェクト）形式」に変換する必要があり，この作業はデータを処理・分析・解析する全工程の50％から90％を占めるとも言われる[1]．さらにビッグデータという用語に代表される規模が大きいデータとなると，その工程には時間・労力・資源が必要となることは容易に想像できよう．

本研究では，Bureau van Dijk（BvD）社[2] から提供されるデータベース Osiris から抽出された世界157カ国の全上場企業（一般事業会社，上場廃止企業含む）の主要財務情報（売上高，営業利益，総資産など84項目，33年分）の財務データを「財務ビッグデータ」（financial big data）と呼んでいる．ファイルの規模としては，約 1.4 GB（ギガバイト）・290万行強（正確には2,900,730行）である[3]．なお，このデータセットは，Jimichi *et al.*（2018）で実際に利用されているものである[4]．

本稿では，データベースから抽出された粗データのファイルをコンピュータ（ソフトウェア）で利用できるファイル形式に変換する工程を「前処理」（preprocessing）と呼び，そのファイルを R に読み込み，実際にデータ解析

1） 例えば，Patil（2012）（p. 18）では，“80% of the work in any data project is in cleaning the data.” と述べられている．

2） https://www.bvdinfo.com/en-gb/

3） 会計学の分野では世界の上場企業の財務データとしては，いわゆる規模が大きいものと考えられる．

4） 地道（2017-a, b）と Saka and Jimichi（2017）では2015年にデータベース Osiris から抽出されたデータファイルを利用しており，本稿で述べる方法と同様の処理が行われている．

が行えるオブジェクト形式に変換する工程を，R の統合開発環境である RStudio 上で行うため，Grolemund and Wickham（2016）にならって「データラングリング」（data wrangling）と呼ぶことにする[5]．その工程を詳しくみることによって，この段階でどのような問題があり，どのような処理が必要となり，それらをどのように自動実行化することによって，「再現可能性」（reproducibility）を確保することができるかの試みについて述べる．なお，データを処理するプラットフォームとしては，macOS High Sierra（10.13.5）を利用していることに注意しよう．また，本稿で利用した R の環境については付録 C を参照されたい．

本稿を含む一連の研究では，Tukey（1977）によって提唱された「探索的データ解析」（Exploratory Data Analysis: EDA）をデータサイエンスを実行するカーネルとして位置付けし（Grolemund and Wickham（2016），Bruce and Bruce（2017）も参照），上記の財務ビッグデータから何らかの意味のある情報・知見を得ることを試みる．さらに，この工程は，ビッグデータという通常の環境では扱うことが難しい対象のもとでも「再現可能研究」（reproducible research）という立場を保ったまま実行可能かどうかということも考察することにする．

II　前処理

粗データセットファイル（raw dataset file）は，データ間の分割符がタブ（\t）区切りのテキストファイル（data.txt）であり，サイズは 1.4 GB（ギガバイト）・2,900,730行である．以下にこのファイルの一部（先頭 3 行）

5）本稿で扱っている前処理とデータラングリングの工程は，データを扱う分野・業種などの違いから．「抽出・変換・読み込み」（Extract, Transform, Load: ETL），「データパイプライン」（data pipeline），「データクレンジング」（data cleansing），「データクリーニング」（data cleaning），「データ操作」（data manipulation）などと呼ばれ，それぞれ厳密には若干ニュアンスが異なっているものもあるけれども，データをコンピュータ（ソフトウェア）で「分析・解析できる形式」に変換する工程といえよう．なお，本橋（2018）で議論されている前処理とは若干異なった意味で使用していることに注意しよう．

を与える：

ソースコード1　データファイル **data.txt** の一部

```
1 AGGREGATE INDUSTRIES HOLDINGS LIMITED BvD ID number Address of incorp. - Country US SIC,
    Primary code(s) (M) US SIC, primary code(s) description Main exchange Consolidation
    code Closing date Number of months Audit Status Accounting standard Source Statement
    unit Currency of the statement Exchange Rate from Local Currency Fixed Assets
    Intangible Fixed Assets Tangible Fixed Assets Other Fixed Assets Current Assets Stock
    Debtors Others Cash & Cash Equivalent Total Assets Shareholders Funds Capital Other
    Non Current Liabilities Long Term Debt Other Non Current Liabilities Provisions
    Current Liabilities Loans Creditors Other Total Shareh. Funds & Liab. Working Capital
    Net Current Assets Enterprise Value Number of Employees Operating Revenue / Turnover
    Sales Costs of Goods Sold Gross Profit Other Operating Items Operating P/L Financial
    Revenue Financial P/L Other non Oper. /Financial Items P/L before Tax Taxation P/L
    after Tax Extraord. & Oth. Items P/L for Period Material Costs Costs of Employees
    Depreciation/Amortization Financial Expenses Research & Development expenses (memo)
    Cash Flow Added Value EBITDA Income taxes Income Tax Payable Deferred Taxes Def. Inc.
    Taxes & Invest. Tax Credit Date of current Market capitalisation Current market
    capitalisation Market price - Year Market price - January Market price - February
    Market price - March Market price - April Market price - May Market price - June
    Market price - July Market price - August Market price - September Market price -
    October Market price - November Market price - December Market Cap.
2       BVDID COUNTRY PSICUSCDE PSICUSSCR MAINEXCHANGE CONSCODE COMMON_CLOSING_DATE
          NMONTHS AUDSTAT ACCSTDFHDR SOURCE STAT_UNIT ISO_CURR EXCHRATE 30030 30040
          30035 30045 30005 30010 30015 30020 30025 30050 30090 30095 30100 30075
          30080 30085 30087 30055 30060 30065 30070 30105 30110 30112 30108 30120
          30205 30210 30215 30220 30225 30235 30240 30250 30255 30260 30265 30270
          30275 30280 30285 30290 30230 30245 30291 30295 30300 30320 13035 21040
          21100 15504 DATEMARKCAP MARKETCP MPRICDATE MPRICJAN MPRICFEB MPRICMAR
          MPRICAPR MPRICMAY MPRICJUN MPRICJUL MPRICAUG MPRICSEP MPRICOCT MPRICNOV
          MPRICDEC KF_MARCAP
3     1985 (th USD) GB00182412 UNITED KINGDOM 3272 Concrete products, except block and
        brick Delisted C1 31/12/1985 12 Qualif n.a. Acc. Std na AR th GBP 1.4445 85,702 n
        .a. 39,796 45,906 62,836 23,777 34,408 4,651 4,030 148,538 78,393 15,384 63,009
        14,142 12,639 1,502 n.a. 56,003 n.a. 26,333 29,670 148,538 31,851 6,832 n.a.
        2,416 145,216 143,901 -111,053 34,162 -18,186 13,159 n.a. -3,221 1,430 11,368
        -2,239 9,129 -29 9,100 n.a. -38,034 -2,817 -3,221 0 11,917 55,411 15,976 -2,239
        1,690 n.a. n.a. n.a. n.a. n.a. n.a. n.a. n.a. n.a. n.a. n.a. n.a. n.a. n.a. n.a
        . n.a.
```

データの構造は，2行のヘッダー部分（ヘッダー行）と33行のデータ部分（データ行）を1ブロックとする82,878ブロックから構成される．（図1）

一般に，財務関連のデータベースから抽出されたデータを，そのまま何らかのソフトウェア環境で解析することは難しい場合が多いけれども，今回扱ったデータファイルは以下のような問題があった：

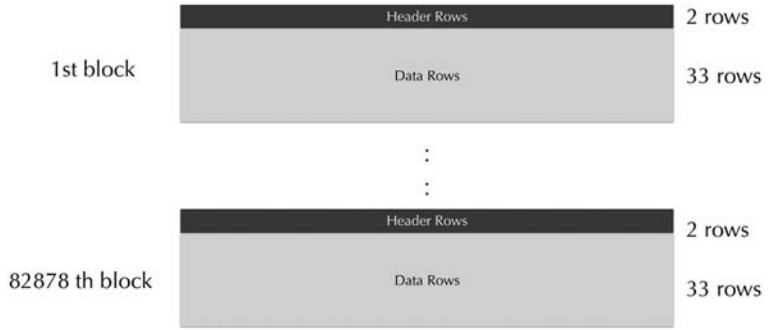

図1　データファイル **data.txt** の構造

（PU1）BOM[6] の存在
（PU2）オペレーティングシステム（Operating System: OS）間での行末コードの相違
（PU3）レイアウトの不統一（レイアウトが異なったヘッダー行とデータ行の混合）
（PU4）金額に関するフォーマット（カンマ区切り）
（PU5）欠損値コード（`n.a.`, `NA` など）の統一と欠損値コードが存在しない欠損値の存在
（PU6）データ行の先頭に余分なタブコード（`\t`）が存在
（PU7）特殊記号の存在（`#`[7] など）

以上の問題に対処することによって，R に読み込むことができるけれども，データ解析を実行するためには，さらに以下のような問題に対処する必要があった：

6）BOM とは，Byte Order Mark の略称であり，Unicode の UTF-16 など16ビット幅のエンコーディング方式において，エンディアンを指定するためにファイルの先頭に記入される16ビットの値である．（IT 用語辞典 e-Words http://e-words.jp/ も参照）．
7）実際に，“DATA#3 LIMITED” という社名をもつ企業が存在する．

（PR1）列（変量）毎の型（数値，文字，年月日など）の指定（構造化）
（PR2）データ部分が収められたファイルには企業名が存在しない（ヘッダー部分に含まれる企業名との結合が必要）
（PR3）データセットの列名[8] の付与
（PR4）不完全なデータが存在（企業分類コード・分類名が各ブロックの先頭のみに収録されている）
（PR5）税金等に関連するデータの符号（マイナスからプラスへ）の変換
（PR6）企業名が一意ではない（同名の企業の存在）
（PR7）通貨換算レートの年のみの情報が存在しない

これらの問題に加えて本稿で扱っているデータファイルは，ある程度の規模があるので通常のエディタや表計算ソフトウェア[9] では整形が困難であるため，以下のような手順で処理した（図２も参照）：

（S1）UNIX コマンドやインタプリタ（`grep`，`dos2unix`，`>`，`(g) sed` など）を利用して整形し，R に読み込める形式 `name.rda`（企業名を含むヘッダー部分が収められたテキストファイル），`data.rda`（データ部分が収められたテキストファイル）へ変換
（S2）データ解析環境 R を用いて処理後，CSV，RDS ファイルに変換し保存

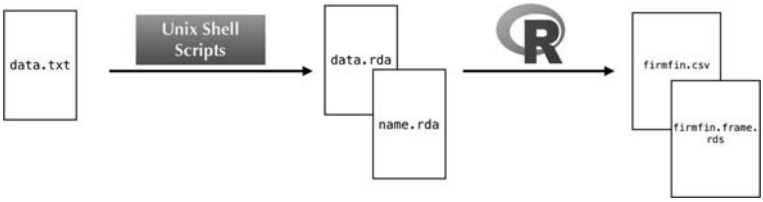

図２　UNIX コマンドと R による前処理

8）ここでの列名は，データベースから抽出するデータセットの列の特性（企業名，決算年月日，財務関連の情報など）を表す．
9）Microsoft Excel 2016 では，$2^{20}=1{,}048{,}576$行を超えるファイルを扱うことはできないことに注意しよう．

以下にこれらの処理の手順の詳細を解説する．

手順（S1）の説明

手順（S1）において，BOM の問題（PU1）はインタープリタ gsed[10] を利用して除去し，問題（PU2）の行末コードは，dos2unix コマンドを利用して変換した[11]．

BOM の除去と行末コードの変換

```
$ gsed -i -s -e '1s/^\xef\xbb\xbf//' data.txt
$ dos2unix data.txt
```

ここで，$ は UNIX ターミナルのプロンプトを表すことに注意しよう．

次に，問題（PU3）については，ヘッダー行（社名が含まれる）とデータ行を grep コマンドとリダイレクション機能（>）を以下のように入力することによって，それぞれ別ファイル name.part（社名を含むヘッダー部分のファイル），data.part（データ部分のファイル）に分離した（図 3 も参照）：

grep によるファイルの分離

```
$ grep -E "th\sUSD\)" data.txt > data.part
$ grep -v -E "th\sUSD\)" data.txt > name.part
```

さらに，問題（PU4）は，データ解析を容易に行うことができるデータファイルの形式として CSV[12] ファイルに変換することを考えると，金額にカン

10）gsed は，GNU プロジェクト（https://www.gnu.org/）によって開発されたソフトウェアの一つであり，UNIX に標準的に用意されているストリームエディター（stream editor）sed を改良したものである．

11）Windows 上でデータベースからデータが抽出されているため，行末コードが復帰（Carridge Return: CR）かつ改行（Line Feed: LF）（CR+LF）であるが，データ解析は macOS（Unix）または Ubuntu（Linux）上で行うため，行末コードを復帰（LF）のみに変換する必要がある．これらは制御コード（control code）の一種であることに注意しよう．

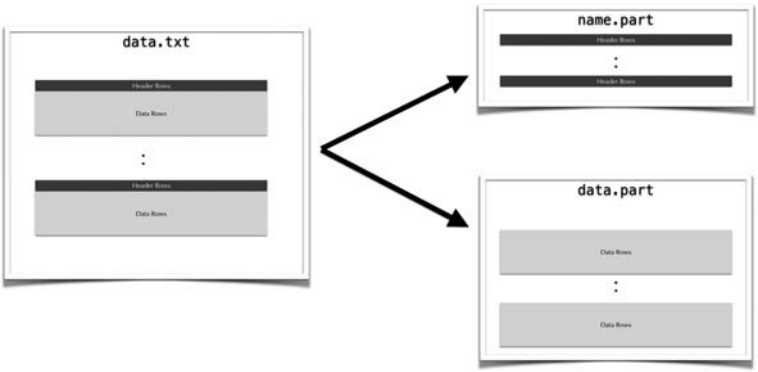

図3　データファイルの分離

マが含まれることによって，データを解析ソフトウェアに読み込む際にトラブルを引き起こす可能性があることを示している．このため，データファイル data.part からカンマを取り除くことが必要となる．また，問題（PU5）に対しては R で扱うことを見越して，欠損値コードを n.a．から NA に全て置換する必要がある．さらに，連続するタブのパターンを試行錯誤によって探し，欠損値コードを適切に挿入する必要がある．これらの必要性に対して，文字列置換のパターンを正規表現で記述したスクリプトファイル（sedscr）として用意しておいて sed を利用して以下のように処理し，リダイレクション（>）を使ってデータファイル data.rda に書き出した：

sed による文字列置換

```
$ sed -f sedscr data. part > data.rda
```

ソースコード2　文字列置換のための正規表現のスクリプトファイル：**sedscr**

```
1 # 置換 n.a. -> NA
2   s/n\.a\./NA/g
3 # 置換 , -> なし
4 s/,//g
5 # 置換 タブ タブ タブ タブ タブ -> タブNA タブNA タブNA タブNA タブ
```

12）CSV は，Comma-Separated Values の略称であり，データ間の分割符としてカンマ（,）を利用したテキストファイルを指す．

```
 6 s/     /NA NA NA NA /g
 7 # 置換 タブ タブ タブ タブ -> タブNA タブNA タブNA タブ
 8 s/     /NA NA NA /g
 9 # 置換 タブタブタブ -> タブNA タブNA タブ
10 s/  /NA NA /g
11 # 置換 タブタブ -> タブNA タブ
12 s/  / NA /g
```

さらに，問題（PU6）は，先頭のタブコードがデータを読み込む際にトラブルとなる可能性があるため削除する必要がある．また，問題（PU7）に関しては，シャープ（#）が，R ではコメント行と判断されるため，削除する必要がある．ここでは，以下のように sed を利用することによって処理した（図 5 も参照のこと）．

sed によるデータファイル data.part における文字列置換

```
$ sed -i -e s/^$'\t'//g data.rda
$ sed -i -e s/#//g data.rda
```

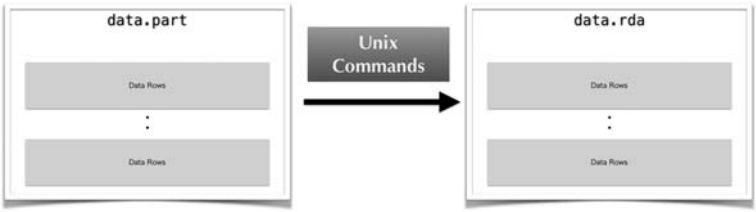

図 4　データファイルに対する文字列置換

さらに，社名を含むヘッダー部分のファイル name.part からも特殊記号であるシャープ（#）を sed を使って以下のように削除した：

sed によるヘッダー部分のファイル name.part における文字列置換

```
$ sed -e s/#//g name.part > name.rda
```

以上の処理より，データファイル data.txt は R に読み込んで処理できる形式 data.rda，name.rda に変換することができた．

なお，手順（S1）における全処理を，以下のようなシェルスクリプトファ

図 5　ヘッダーファイルに対する文字列置換

イル（script.sh）に記述しておき，シェルを起動することによって自動実行し，結果が再現することを確かめた（図 6 も参照のこと）：

シェルスクリプトによるデータファイル操作の自動実行

```
$ /bin/sh ./script.sh
```

ソースコード 3　データファイル操作のシェルスクリプトファイル：**script.sh**

```
 1 #!/bin/sh
 2 echo "Remove␣BOM␣codes"
 3 gsed -i -s -e '1s/^\xef\xbb\xbf//' data.txt
 4 echo "dos2unix"
 5 dos2unix data.txt
 6 echo "seperate␣data␣file"
 7 grep -E "th\sUSD\)" data.txt > data.part
 8 grep -v -E "th\sUSD\)" data.txt > name.part
 9 echo "replacement␣special␣character"
10 sed -f sedscr data.part > data.rda
11 sed -i -e s/^$'\t'//g data.rda
12 sed -i -e s/#//g data.rda
13 sed -e s/#//g name.part > name.rda
```

手順（S2）の説明

問題（PR1）～（PR6）に対処するために，Rscript を利用することによって，ターミナルのコマンドラインで R スクリプトを自動実行することによって，CSV ファイルと RDS[13] ファイルへ変換した：

Rscript によるデータファイルのダンプ

```
$ Rscript datadump.R "data.rda" "name.rda" "firmfin.csv" "firmfin.frame.rds"
```

13）RDS ファイルは，R おいて単一のオブジェクトを高速に入出力するための一つのファイル形式（バイナリファイル）である．ここで，フルデータセットのうち，一部の列を選択したものをこの形式で出力している．

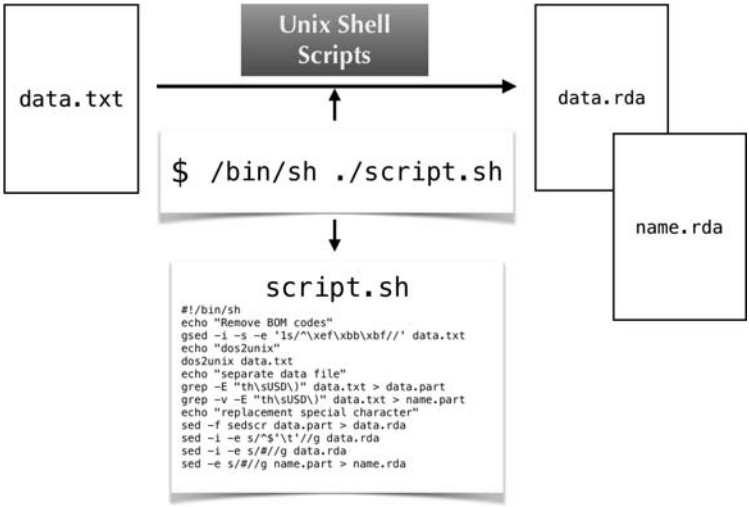

図 6　前処理のシェルスクリプトによる自動実行

ここで，R で行う一連の処理を R スクリプトファイル datadump.R（ソースコード 4）として与え，読み込むファイルを引数 "data.rda" と "name.rda" に，さらに出力するデータファイルを引数 "firmfin.csv" と "firmfin.frame.rds" に与えることによって R スクリプトへ引き渡していることに注意しよう．

ソースコード 4　データ変換のための R スクリプトファイル：**datadump.R**

```
require(dplyr)
require(readr)
args <- commandArgs(trailingOnly = T)
tmp1<-read_tsv(args[1],na="NA",quote="",col_names=FALSE)
tmp1<-data.frame(tmp1)
for(j in 1:7) tmp1[,j]<-as.character(tmp1[,j])
tmp1[,8]<-as.Date(tmp1[,8],format="%d/%m/%Y")
tmp1[,9]<-as.numeric(tmp1[,9])
for(j in 10:14) tmp1[,j]<-as.character(tmp1[,j])
for(j in 15:83) tmp1[,j]<-as.numeric(tmp1[,j])
tmp2<-read_tsv(args[2],na="NA",quote="",col_names=FALSE)
tmp2<-data.frame(tmp2)
firmname<-tmp2[seq(1,dim(tmp2)[1],2),1]
firmfin.raw.frame<-data.frame(rep(firmname,each=33),tmp1)
varnames<-scan("variablenameOsirisforSpark.txt",what="")
colnames(firmfin.raw.frame)<-varnames
```

```
17 firmfin.frame<-mutate(firmfin.raw.frame,
18             SIC_code=rep(SIC_code[seq(1,dim(firmfin.raw.frame)[1],33)],each=33),
19             SIC_name=rep(SIC_name[seq(1, dim(firmfin.raw.frame)[1],33)],each=33),
20             costs_goods=-costs_goods,
21             expenses_other=-expenses_other,
22             tax=-tax,
23             tax_income=-tax_income,
24             costs_material=-costs_material,
25             costs_employees = -costs_employees,
26             depr_amor = -depr_amor,
27             interest_paid = -interest_paid,
28             R_D =-R_D,
29             firmID=paste(firm,ID),
30             year=rep(seq(1985,2017),length(firmname)))
31 write.csv(firmfin.frame,args[3],row.names=FALSE)
32 saveRDS(select(firmfin.frame,
33             firm,firmID,year,month,country,SIC_code,SIC_name,sales,employees,assets_total
                 ),
34      args[4])
```

ソースコード4に与えられている処理を問題（PR1）～（PR7）に対応させながら以下に説明する．まず，ソースコードの１，２行目では，この処理を行うために追加で必要なRパッケージ dplyr[14] と readr[15] を読み込んでいる．次に，３行目でコマンドラインで与えた引数をオブジェクト args に付値している．また，４，５行目でデータ部分が納められたファイル data.rda を関数 read_tsv[16] で読み込んだ後，data.frame オブジェクト（tmp1）に変換している．

次に，問題（PR1）に対応するために，６行目から10行目で列（変量）毎の型（数値，文字，年月日など）を再定義している．

また，11，12行目で社名を含むヘッダーが納められたファイル name.rda

14）dplyr は，R で表形式のデータを扱うための標準的なデータ構造である data.frame に対して，列抽出（select），行抽出（filter），列追加（mutate），要約（summarise），並べ替え（arrange）などの処理を高速に行うためのパッケージである．詳細は https://dplyr.tidyverse.org/ を参照されたい．本稿で扱っているデータ解析を行うために必要不可欠なパッケージである．

15）readr は，タブ区切りのテキストファイルや CSV ファイルを高速に読み込んで data.frame オブジェクトに変換するための関数群が納められたパッケージである．詳細は，https://readr.tidyverse.org/ を参照されたい．このパッケージも，本稿で扱っているデータの入力を行うために必要不可欠なパッケージである．

16）read_tsv は，readr パッケージに収録されているタブ区切りのテキストファイルからデータを高速に読み込むための関数である．R に標準的に用意されている関数 read.table と比較して（場合によるが）10倍程度早いという報告もある．

を関数 read_tsv で読み込んだ後，data.frame オブジェクト（tmp2）に変換し，13行目で data.frame オブジェクト tmp2 の１列目に１行飛ばしに与えられている企業名を抽出し，オブジェクト firmname に付値している．このオブジェクトの各成分を33回繰り返すことによって各企業の33年分の企業名を作り，14行目でこのオブジェクトとデータ部分のオブジェクト tmp1 を列結合することによって，data.frame オブジェクト firmfin.raw.frame を定義している．この処理によってデータ部分が収められたファイルに存在しなかった企業名の列を追加することができた．よって問題（PR2）は解決した．

問題（PR3）は，15行目でテキストファイル（variablenameOsiris forSpark.txt）として別途用意したもの[17] を読み込み，16行目で列名として付値することによって処理した．

これ以降の処理は，dplyr パッケージに付属の関数 mutate を利用して行っていることに注意しよう．

まず，企業分類コード（SIC コード）と企業分類名称（SIC 分類名）が各データブロックの先頭のみしか収録されていないという問題（PR4）に対しては，収録されているコードと名称を33回ずつ繰り返したものを再定義することによって対処した（18，19行目を参照）．

また，税金等に関連するデータが負の値として収録されているものがあるため，データ解析の観点から正の値へ変換しておいた方がよいため，同じく再定義することによって対処した（20～28行目を参照）．

さらに，問題（PR5）の同名の企業の存在については，社名と BvD 社が定義している企業コード（BvD ID number）を結合した新たな列 firmID を

17）データセットの列名は，ヘッダー情報が収められたファイル name.rda にも存在するが，これは BvD 社の付与したものであり，空白などが存在したりフォーマットの観点から R で処理する際に問題となる可能性があるため，このような方法で対応した．なお，データベースからデータセットを抽出する際に，抽出対象となる財務関連の情報はデータセットの特徴を決定する大切な事項であるため，BvD 社と綿密な打ち合わせのもとで行っている．なお，抽出対象となった変数の説明を付録 E の表２に与える．

定義することによって企業の一意性を確保した（29行目を参照）.

最後に，問題（PR6）の通貨換算レートの年については，今回扱うデータが1985年から2017年のものであるという条件を使って，1985から2017の数列を企業数の分だけ繰り返すという処理によって新たな列 `year` を定義した（30行目を参照）.

最終的に出力された CSV ファイルの規模は，2,734,975行，1.3 GB になった（31行目を参照）．なお，一部の列を選択したデータセットを RDS 形式で出力していることにも注意しよう（32，33，34行目を参照）.

Ⅲ データラングリング

前節で処理されたデータファイル `firmfin.csv` の規模（2 GB 未満）のファイルであれば，R に標準的に用意されている関数 `read.table` や，readr パッケージに付属する `read_csv` などを用いて R に読み込み，分析することも可能であるけれども，今後，さらに規模の大きなものを扱うことを見越して，ここでは高速かつ汎用的なクラスター・コンピューティング・システム Apache Spark™.（以下，Spark と略す）を利用する．Spark については付録 B を参照されたい.

データサイズがメモリー容量を超える場合，従来の方法では，一旦データをローカルまたはリモートのデータベースに保存しておいて，分析するときに適当なソフトウェアと API[18] を利用して，SQL を協調して利用することが一般的な方法であった[19]．しかしながら，これらの言語は異なったものであり，協調性に欠けるという欠点があった.

この問題に対して，Spark を利用すれば，データサイズがメモリー容量を

18）API とは Application Program Interface の略称であり，「あるコンピュータプログラム（ソフトウェア）の機能や管理するデータなどを，外部の他のプログラムから呼び出して利用するための手順やデータ形式などを定めた規約のこと」である．（IT 用語辞典 e-Words `http://e-words.jp/` より引用）

19）例えば，R からデータベース MySQL へ RMySQL パッケージを利用して接続する場合がそれにあたる.

超える場合にも対応しており，Spark と Scala[20] を利用することによって，リモート・ローカルに関わらず一連の処理・分析をシームレスに行うことができることが利点の一つといえる．さらに，R から，SparkR[21] と sparklyr[22] というパッケージを使って，データを一旦 Spark に読み込み，さらに，R でデータ解析できる形式（data.frame オブジェクト）に変換する方法について以下で説明する．

1．SparkR によるデータの読み込みと変換

SparkR は R から Spark を利用するためのフロントエンドとして Spark 純正の R パッケージとして提供されている[23]．Spark 1.4.0 から実装された Spark DataFrame は規模の大きなデータセットを扱うことができるという利点があることに注意しよう．また，Spark 2.1.0 から列選択（select）や行選択（filter）等に対応する機能が SparkR に実装された[24]．なお，付録 D に処理を行うための関数群を dplyr パッケージに用意されている関数群と対比した表を与えているので参照されたい．

ここでは，R の統合開発環境である RStudio[25] 上で以下のように入力し，SparkR パッケージを利用した：

SparkR パッケージを利用するための設定：macOS の場合

```
> Sys.setenv(SPARK_HOME = "/usr/local/Cellar/apache-spark/2.2.0/libexec")
> library(SparkR, lib.loc = c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib")))
> sparkR.session(master = "local[*]", sparkConfig = list(spark. driver. memory = "2g"))
Java ref type org.apache.spark.sql.SparkSession id 1
```

20）Scala はオブジェクト指向プログラミング言語の一つ．
21）http://spark.apache.org/docs/latest/sparkr.html#overview
22）http://spark.rstudio.com/
23）SparkR は UC，Berkeley の AMPLab のチームによって Spark とは独立したプロジェクトとして開発されていたが，Spark 1.4.0 以降から正式に Spark プロジェクトに統合された．
24）R パッケージ dplyr パッケージに実装されているものと同様の機能があるが，規模の大きなデータセットに適用可能であることに注意しよう．
25）https://www.rstudio.com/

SparkR パッケージに付属する関数 read.df を利用することによって，CSV ファイルを R のオブジェクト（Spark DataFrame オブジェクト）として読み込むことができる．

read.df によるデータの読み込み

```
> firmfin.sdf<-read.df("firmfin.csv",sourc="csv",
+                      header=TRUE, inferSchema = "true", na.strings = "NA")
```

読み込まれた Spark DataFrame オブジェクトは，そのままでは可視化や統計モデリングを行うことに適していないため，R で標準的に扱われるオブジェクト形式（data.frame オブジェクト）に変換する．その際，本研究では，2015年のデータにもとづく売上高を従業員数と総資産で説明するためのモデルを構築するため，それらのデータを再抽出する必要がある．なお，抽出にあたっては各指標の値が正のものと，さらに決算月数が12カ月のものという条件を与えることとした．この抽出にあたって，Spark 2.1.0 の関数 filter，select を利用し，さらに Spark DataFrame オブジェクトから R の data.frame オブジェクトへ変換するために関数 collect を利用した．なお，magrittr パッケージに付属するパイプ（%>%）を利用すると，変換過程をパイプライン標記することができるため，コードの視認性が向上するという意味で便利である．

これらの関数を利用することによって，以下のように最終的に R のデータ・フレーム・オブジェクト（firmfin2015）に変換した（図 7）：

R データ・フレーム・オブジェクト firmfin2015 への抽出・変換

```
> library(magrittr)
> firmfin2015<-firmfin.sdf %>%
+   filter(firmfin.sdf$year=="2015" &
+   firmfin.sdf$sales>0 & firmfin.sdf$employees>0 & firmfin.sdf$assets_total>0 &
+   firmfin.sdf$month==12) %>%
+   select(firmfin.sdf$firmID, firmfin.sdf$country,
+   firmfin.sdf$sales, firmfin.sdf$employees, firmfin.sdf$assets_total) %>%
+   collect()
> colnames(firmfin2015)<-c("firmID","country", "sales", "employees","assets.total")
```

最後の行で列名を若干修正していることに注意されたい[26]．

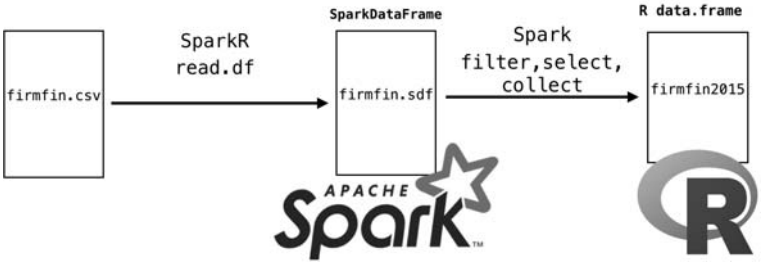

図7　SparkR によるデータの読み込みと変換

以上の操作によって得られたデータは以下のようなものである（最初の5件のみ抽出）：

R データ・フレーム・オブジェクト firmfin2015

```
> head(firmfin2015,5)
                          firmID        country     sales employees assets.total
1 ELECTROCOMPONENTS PLC GB00647788 UNITED KINGDOM   1859571      6024      1299582
2             COBHAM PLC GB00030470 UNITED KINGDOM   3070497     12527      4916351
3      REDHALL GROUP PLC GB00263995 UNITED KINGDOM    104624       796        61349
4           BT GROUP PLC GB04190816 UNITED KINGDOM  27426186    102500     61345242
5                BP PLC GB00102498 UNITED KINGDOM 222894000     79800    261832000
```

ここで，データ件数は26,682社であり，各列（変量）は以下のようなものである：

`firmID`: 企業名と BvD 企業コードを結合したもの

`country`: 国名

`sales`: 売上高（単位：1,000米ドル）

`employees`: 従業員数（単位：人）

`assets.total`: 総資産（単位：1,000米ドル）

26）Spark は Java を利用したシステムであることから，ドット（.）は特別な機能（メソッドチェーン）を持つためオブジェクト名などの分割符などに利用することはできない．よって，オブジェクト名の分割符として，慣習としてアンダーバー（_）が利用される．一方，アンダーバーはかつて（S 言語などでは）付値を行う記号として利用されていたため，習慣的理由で，本稿では R のオブジェクト名の分割符としてドットを利用する．なお，現在 R ではアンダーバーにこのような機能は割り当てられていないためオブジェクト名の分割符として利用できることに注意しよう．

2．sparklyr によるデータの読み込みと変換

sparklyr とは，SparkR と同様に R から Spark に接続する機能を提供する R パッケージであるが，R 純正の dplyr のバックエンドを提供する点が異なっている[27]．Spark から dplyr の機能を使って R へデータを転送することができ，この意味で，R の純正の機能を使って解析や可視化が可能となる．また，R から Spark が提供する機械学習のライブラリの利用が可能であることにも注意しよう．

sparklyr の利用に関しては，R の統合開発環境である RStudio 上で以下のような入力によって sparklyr パッケージをロードし利用した：

sparklyr パッケージを利用するための設定

```
> library(sparklyr)
> library(dplyr)
> sc <- spark_connect(master = "local")
```

sparklyr パッケージに付属する関数 spark_read_csv を利用することによって CSV ファイルを Spark に読み込むことができる．

spark_read_csv によるデータの読み込み

```
> firmfin_tbl<-spark_read_csv(sc,
+                             name="firmfin", memory=FALSE,
+                             path="firmfin.csv",
+                             header＝TRUE, delimiter = ",")
```

読み込まれたオブジェクト firmfin_tbl は，tbl_spark クラスに属しており，dplyr パッケージの機能を Spark DataFrame へ適用できるような親和性を持つインターフェースを提供している．よって，sparklyr パッケージを利用するときは，dplyr パッケージに付属する関数 filter，select を利用し，さらに関数 collect を利用できる．なお，SparkR，sparklyr パッ

27）sparklyr は RStudio 社が開発・配布を行っており，この会社が R ネイティブの機能を改良・強化するパッケージを開発しているという点からも，R により親和的な仕様となっていることに注意しよう．

ケージには，同じ関数名で同様の機能があるけれども，それらの利用法は若干異なっていることに注意しよう[28]．

なお，Spark DataFrame オブジェクト同様，firmfin_tbl オブジェクトは，そのままでは可視化や統計モデリングを行うことに適していないため，dplyr パッケージを導入することによって tibble[29] オブジェクトに変換する．

その際，SparkR パッケージを利用した場合と同様に，2015年のデータにもとづく売上高を従業員数と総資産で説明するためのモデルを構築するため，各指標の値が正のものと，さらに決算月数が12カ月のものという条件を与え，データを再抽出する．これらの関数を利用することによって，以下のように最終的に R の tiblle オブジェクト（firmfin2015.tbl）に変換した（図８）：

R tiblle オブジェクト firmfin2015.tbl への抽出・変換

```
> firmfin2015.tbl<-firmfin_tbl %>%
+   mutate(year=as.integer(year),
+          sales=as.numeric(sales),
+          employees=as.numeric(employees),
+          assets_total=as.numeric(assets_total),
+          month=as.integer(month)) %>%
+   filter(year==2015, sales>0, employees>0, assets_total>0, month==12) %>%
+   select(firmID, country, sales, employees, assets_total) %>%
+   collect()
> colnames(firmfin2015.tbl)<-c("firmID","country","sales","employees","assets.total")
```

SparkR の場合と同様に，最後の行で列名を修正していることに注意されたい．

以上の操作によって得られたデータ（tibble オブジェクト）は以下のようなものである：

28）SparkR＋magrittr パッケージが「Spark 寄り」の仕様であるのに対して，sparklyr＋dplyr パッケージは「R 寄り」の仕様である．なお，これらのパッケージを同時に使用することはトラブルのもとになるので避けるべきである．

29）tibble は，R の表形式のデータを扱うときに標準的なオブジェクトクラス data.frame を拡張したものである．詳しくは https://github.com/tidyverse/tibble を参照されたい．

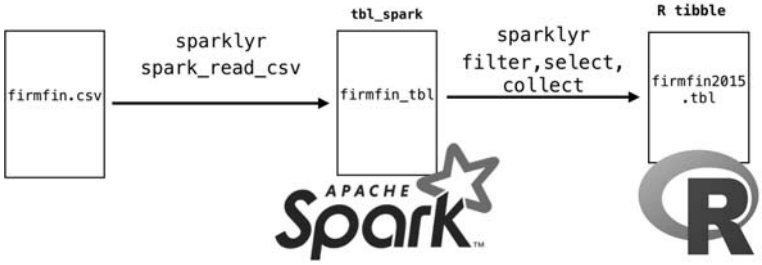

図 8　sparklyr によるデータの読み込みと変換

R tiblle オブジェクト firmfin2015.tbl

```
> firmfin2015.tbl
# A tibble: 26,682 x 5
   firmID                                                  country            sales employees assets.total
   <chr>                                                   <chr>              <dbl>     <dbl>        <dbl>
 1 ELECTROCOMPONENTS PLC GB00647788                        UNITED KINGDOM   1859571      6024      1299582
 2 COBHAM PLC GB00030470                                   UNITED KINGDOM   3070497     12527      4916351
 3 REDHALL GROUP PLC GB00263995                            UNITED KINGDOM    104624       796        61349
 4 BT GROUP PLC GB04190816                                 UNITED KINGDOM  27426186    102500     61345242
 5 BP PLC GB00102498                                       UNITED KINGDOM 222894000     79800    261832000
 6 BRITISH LAND COMPANY PUBLIC LIMITED COMPANY(THE) GB00621920 UNITED KINGDOM    849777       692      19984157
 7 BAE SYSTEMS PLC GB01470151                              UNITED KINGDOM  24876655     75000      29760997
 8 BRAMMER LIMITED GB00162925                              UNITED KINGDOM   1062967      3862        670856
 9 SPIRENT COMMUNICATIONS PLC GB00470893                   UNITED KINGDOM    477100      1754        592000
10 DE LA RUE PLC GB03834125                                UNITED KINGDOM    703154      3566        651880
# ... with 26,672 more rows
```

ここで，先頭10行の情報が表示されており[30]，さらに，オブジェクトのサイズなどの情報も与えられていることに注意しよう．

Ⅳ　自動実行による再現可能性

本節では，Ⅱ節で扱った前処理と，Ⅲ節で述べたデータラングリングの工程を自動実行することを実現することによって再現可能性を確保する方法について考察する．

まず，前処理に関しては，ソースコード 3 によって与えられるシェルスク

30）R に標準的に用意されているデータ構造である data.frame は，その内容をオブジェクト名をコンソールに入力することによって表示すると，全体が「流れる」という仕様となっており，行数が大きい場合に視認性の観点から問題となる．この理由のため関数 head などと併用して先頭の行を表示する方法がとられる．なお，tibble オブジェクトはこのような問題に対して改良されていることに注意しよう．

リプトファイル script.sh と，ソースコード 4 によって与えられる R スクリプトファイル datadump.R を，それぞれ，標準シェル /bin/sh[31] とシェルコマンド Rscirpt[32] で実行することによって実行されていることに注意しよう．

よって，これらのスクリプトをファイル Makefile に記述しておいて，UNIX コマンド make を以下のように実行することによって自動実行することが可能である（図 9 も参照）：

make による前処理（CSV ファイルの作成）

```
$ make csv
/bin/sh ./script.sh
Remove BOM codes
dos2unix
dos2unix: converting file data.txt to Unix format...
seperate data file
replacement special character
Rscript datadump.R "data.rda" "name.rda" "firmfin.csv" "firmfin.frame.rds"
:
:
Parsed with column specification:
cols(
  .default = col_integer(),
  X1 = col_character(),
  X2 = col_character(),
  X3 = col_character(),
:
:
Parsed with column specification:
cols(
  .default = col_character()
)
See spec(...) for full column specifications.
Read 84 items
```

ここで，csv は作成するファイルが CSV ファイルであることを表しており，ソースコード 5 に実際のファイル Makefile を与える．

31）macOS は標準シェル /bin/sh の実態は（/bin/bash とは挙動が若干異なるけれども）GNU bash，version 3.2.5 であることに注意しよう．

32）Rscirpt は，R のスクリプトを実行するためのフロントエンドであり，UNIX コマンドとして実装されている．

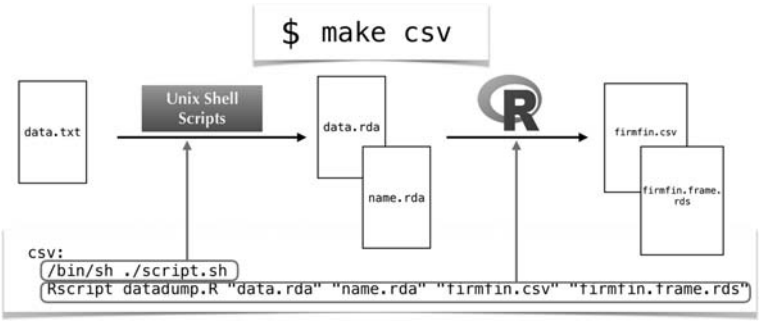

図 9　**make** コマンドによる前処理の自動化

ソースコード 5　**Makefile**

```
1  all:
2        /bin/sh ./script.sh
3        Rscript datadump.R "data.rda" "name.rda" "firmfin.csv" "firmfin.frame.rds"
4        ~/Library/TeXShop/Engines/Sweave-utf8.engine paper.Rnw
5  csv:
6        /bin/sh ./script.sh
7        Rscript datadump.R "data.rda" "name.rda" "firmfin.csv" "firmfin.frame.rds"
8  paper:
9        ~/Library/TeXShop/Engines/Sweave-utf8.engine paper.Rnw
10 presentation:
11       ~/Library/TeXShop/Engines/Sweave-utf8.engine presentation.Rnw
12 preview-paper:
13       open paper.pdf
14 preview-presen:
15       open presentation.pdf
16 clean-tex:
17       rm paper-* *.out *.log *.tex *.aux
18 clean-data:
19       rm *.rda *.rda-e *.part
20 clean-pdf:
21       rm *.pdf
```

ソースコード 5 における 5，6，7 行目が今回の前処理に利用された部分である．ここで，これらの 3 行は「ルール」と呼ばれ，csv は「ターゲット」と呼ばれることに注意しよう[33]．

次に，データラングリングを自動実行することは RStudio 上では R スク

33）make については，https://www.oreilly.co.jp/library/4873112699/ などを参照されたい．

リプトを実行することによって可能であるが，本稿で扱ったのはデータを読み込んだ段階であるので，この後に続く探索的データ解析による可視化や統計モデリングの工程が定まらないうちは，シェルスクリプトを利用し自動的に実行することは本質的に重要でないと思われるため，本稿では割愛した．ただし，探索的データ解析を実行することによって，興味深い結果（新たな価値）を `Rnw` ファイルとして記述しておけば，Sweave（Leisch（2002）を参照）で処理することによって「動的文書」（dynamic documents）を生成することができる．さらに，それらの工程を `Makefile` に記述しておいて，`make` コマンドで自動実行することによって再現可能性を確保することが可能であることに注意しよう．（ソースコード 5 における 5，6 行目がその役割を果たすルールである．）例えば，Jimichi *et al.*（2018）や本稿は，このような仕様で文書を生成している．なお，動的文書生成と再現可能研究については，Gandrud（2015），Xie（2015），高橋（2014，2018），地道，豊原（2018）などを参照されたい．

V　おわりに

本稿では，macOS 上でデータファイルの前処理に UNIX コマンド・インタープリタと R を利用した．また，データラングリングに Spark と R パッケージ SparkR，sparklyr，dplyr，magrittr を RStudio 上で使った．これらの工程は，最近使用者が急増している Python[34] を利用したり，直接 Scala と Spark を利用して処理を行う方法もあろう．また，RStudio 社が提供する tidyverse パッケージ群等を用いて全ての工程を RStudio 上で行うことも可能と思われる．

本稿で扱ったデータセットは 1.4 GB 程度であり，現時点でのビッグデータ関連の文献では「スモールデータ」と呼ばれるかもしれないが，ビッグデータという用語自体は，いわゆる「バズワード」であり，利用される分野や業

34）`https://www.python.org/`

種，立場などの文脈から判断される必要があろう．さらに，ビッグデータは「相対的な」用語であることにも注意が必要である．現在，筆者が扱っているさらに規模の大きなデータセットは，BvD 社から提供されているデータベース Orbis から抽出された世界の上場・非上場企業（22,312,669社）の主要財務情報（売上高，営業利益，総資産など80項目）を最長10年分収集したものである．サイズとしは，テキスト形式のファイルで約 122 GB（2.2億行）であり，通常のコンピュータ環境ではデータ容量がメモリー容量を超えるため扱うことが難しいことに注意しよう．このデータは本稿で扱っているものに比べて大きいことから，「ビッグデータ」と呼べよう．なお，このデータセットを扱うことは，通常の情報環境では難しいため，東京大学情報基盤センターに設置された専有利用型リアルタイムデータ解析ノード（FENNEL）を利用し，複数台の高性能クラスタに配置されたデータセットを前述の Spark と R のみならず，Apache Hadoop，Hive を協調して前処理することを試みている．

今後の研究課題として，本稿で扱った財務ビッグデータを再現可能性を確保し，実際に探索的データ解析を行う方法について，引き続き考察する予定である．また，さらに規模が大きなデータセットについても同様の観点から再現可能性を維持したまま前処理，データラングリング，探索的データ解析が実行できるかについても検討する予定である．

（筆者は関西学院大学商学部教授）

謝辞

本研究の一部は以下の研究費より助成を得ていることに感謝の意を述べたい：

- 科学研究費基盤研究 C：「グラフィカル・データ・アナリシスによる格差研究と社会環境会計による解決方法の提案」（2016年～2018年），課題番号：16K04022，研究代表者：阪智香
- 平成29年度学際大規模情報基盤共同利用・共同研究拠点（JHPCN）課題：「財務ビッグデータの可視化と統計モデリング」，課題番号：jh171002-NWJ，研究代表者：地道正行
- 平成30年度学際大規模情報基盤共同利用・共同研究拠点（JHPCN）課題：「財務ビッ

グデータの可視化と統計モデリング」，課題番号：jh181001-NWJ，研究代表者：地道正行
- 関西学院大学図書館図書費 B
- 関西学院大学個人研究費

また，BvD の増田歩氏にはデータの抽出に関して多大なるご協力いただいた．ここに感謝の意を述べる．

参考文献

［1］ Bruce, P. and A. Bruce（2017）*Practical Statistics for Data Scientists: 50 Essential Concepts,* O'Reilly Media.
（大橋真也監修，黒川利明訳（2018）『データサイエンスのための統計学入門：予測，分類，統計モデリング，統計的機械学習と R プログラミング』，オライリージャパン.）
［2］ Chambers, J. M. and T. J. Hastie ed.（1991）*Statistical Models in S.* Chapman and Hall/CRC.
［3］ Efron, B. and T. Hastie（2016）*Computer Age Statistical Inference: Algorithms, Evidence, and Data Science,* Cambridge University Press.
［4］ Gandrud, C.（2015）*Reproducible Research with R and RStudio,* Second Edition, CRC Press.
［5］ Janssens, J.（2014）*Data Science at the Command Line,* O'Reilly Media.
（太田満久，下田倫大，増田泰彦監訳，長尾高弘訳（2015）『コマンドラインではじめるデータサイエンス：分析プロセスを自在に進めるテクニック』，オライリー・ジャパン.）
［6］ 地道正行（2010）『財務データベースサーバの構築』，ISBN: 978-4-9905530-0-5, https://kwansei.repo.nii.ac.jp/
［7］ 地道正行（2014）『R を利用した財務データの可視化と統計モデリング：探索的データ解析の視点から』，商学論究，61巻，3号，pp. 241-295.
［8］ 地道正行（2017）『R による対数非対称正規線形モデルによる財務データの統計モデリング』，商学論究，第64巻，第5号，pp. 159-185，2017年3月，関西学院大学商学研究会.
［9］ 地道正行（2017）『R を利用した対数非対称分布族にもとづく財務データの統計モデリング』，経済学論究，第71巻，第2号，pp. 141-174，2017年9月，関西学院大学経済学部研究会.
［10］ Jimichi, M., Miyamoto, D., Saka, C. and Nagata, S.（2018）*Visualization and Statistical Modeling of Financial Big Data: Log-Linear Modeling with Skew Error,* SSRN: https://ssrn.com/abstract＝3166440, submitted.
［11］ 地道正行，豊原法彦（2018）『景気先行指数の動的文書生成にもとづく再現可能研究』，豊原法彦編著『関西経済の構造分析』，第5章，pp. 77-111，中央経済社.
［12］ Karau, H., A. Konwinski, P. Wendell, and M. Zaharia（2015）*Learning Spark,* O'Reilly.

（玉川竜司訳（2015）『初めての Spark』，オーライリー・ジャパン.）
［13］ Knuth, D. E.（1984）Literate Programming, *The Computer Journal, British Computer Society,* Vol. 27, No. 2, pp. 97-111.
［14］ Leisch, F.（2002）*Sweave: Dynamic generation of statistical reports using literate data analysis,* In Wolfgang Härdle and Bernd Rönz, editors, Compstat 2002 -Proceedings in Computational Statistics, pp. 575-580. Physica Verlag, Heidelberg. ISBN 3-7908-1517-9.
［15］ 本橋智光（2018）『前処理大全：データ分析のための SQL/R/Python 実践テクニック』，技術評論社.
［16］ 西田圭介（2017）『ビッグデータを支える技術：刻々とデータが脈打つ自動化の世界』，技術評論社.
［17］ Patil, DJ（2012）*Data Jujitsu: The Art of Turning Data into Product,* An O'Reilly Radar Report, O'Reilly.
［18］ Ryza, S., U. Laserson, S. Owen, and J. Wills（2016）*Advanced Analytics with Spark*, O'Reilly.（玉川竜司訳（2016）『Spark による実践データ解析』，オライリー・ジャパン.）
［19］ Saka, C. and M. Jimichi（2017）Evidence of inequality from accounting data visualisation, *Taiwan Accounting Review,* Vol. 13, No. 2, pp. 193-234.
［20］ 猿田浩輔，土橋昌，吉田耕陽，佐々木徹，都築正宜，下垣徹監修（2015）『Apache Spark 入門：動かして学ぶ最新並列分散処理フレームワーク』，翔泳社.
［21］ 下田倫大，師岡一成，今井雄太，石川有，田中裕一，小宮篤史，加嵜長門（2016）『詳解 Apache Spark』，技術評論社.
［22］ 高橋康介（2014）『シリーズ Useful R 9：ドキュメント・プレゼンテーション生成』，共立出版.
［23］ 高橋康介（2018）『Wonderful R 3：再現可能性のすゝめ：RStudio によるデータ解析とレポート作成』，共立出版.
［24］ Tukey, J. W.（1977）*Exploratory Data Analysis,* Addison-Wesley Publishing Co.
［25］ Xie, Y.（2015）*Dynamic Documents with R and knitr, Second Edition,* CRC Press.
［26］ Wickham, H. and G. Grolemund（2016）*R for Data Science,* O'Reilly.

付録A　データサイエンス

データサイエンスという用語は，ビッグデータと同様に分野・業種・立場の観点から，それぞれ異なったニュアンスで使用されており，その定義もさまざまである．たとえば，柴田（2015）は，「データから新たな価値を創出する科学」（p. 75 を参照）と述べており，データサイエンスの本質的な定義を与えることを試みている．この定義は，米国のデータサイエンス協会

（Data Science Association: DSA）のものとほぼ同義であることに注意しよう[35]（Efron and Hastie（2016）も参照）．また，柴田（2015）の第6章には，「データ分析」と「データ解析」の違いや，データサイエンスを実践する上で重要な視点・考え方についても例を用いて詳しく説明されている．

一方，Mason and Wiggins（2010）では，データを入手し，分析した結果を考察する工程を，獲得（Obtain），クレンジング（Scrub），精査（Explore），モデリング（Models），解釈（iNterpret）という5段階に分けることによって簡潔に定義している[36]（Janssens（2015）も参照）．この定義は，データサイエンスを実行する工程を明確にすることによって，実践的な側面からその定義付けを行ったものと見なすことができる．

また，Grolemund and Wickham（2016）では，典型的なデータサイエンスにもとづくプロジェクトにおいて利用される手法に対するモデルを与えており，それは「読み込み」（import）→「整理」（tidy）→「変換」（transform）→「可視化」（visualization）→「モデル」（model）→「伝達」（communicate）という段階をもつ工程である．ここで，「読み込み」，「整理」，「変換」の工程はデータラングリングと呼ばれ，また「変換」，「可視化」，「モデル」の工程はサイクルの構造を持つことに注意しよう．なお，実際にこれらの工程を実行するための環境として，データ解析環境 R と tidyverse パッケージ[37] が RStudio 社から提供されている．

35）http://www.datascienceassn.org/about-data-science から引用すると，“Data Science” means the scientific study of the creation, validation and transformation of data to create meaning.
である．

36）OSEMN と略される．OSEMN は，awesome（すごい，荘厳な）と同一の発音が当てられる．

37）tidyverse（https://www.tidyverse.org/）は，R を用いてデータサイエンスを実行するためのパッケージ群であり，dplyr（データ整形），ggplot2（データ可視化），readr（データ読み込み），tibble（データ構造），tidyr（データ整理），purrr（繰り返し）から構成される．

付録B　Apache Spark

Apache Spark とは，高速かつ汎用的なクラスタ・コンピューティング・システムの一つであり，Java，Scala，Python，R 向けの API が提供されている．Spark には，SQL 対応データベースへアクセスするための SparkSQL や機械学習のための MLlib，グラフ処理のための GraphX，リアルタムデータ処理のための Spark Streaming などの高性能なツールも潤沢にサポートされている．本稿では，Spark 2.2.0 を利用しているけれども，原稿執筆時点での最新バージョンは Spark 2.3.1 である．なお，Spark に関する情報は，Karau *et al.*（2015），猿田他（2015），Ryza *et al.*（2016），下田他（2016）などを参照されたい．また，最新の情報については，URL `http://spark.apache.org/docs/latest/index.html` を参照されたい．

付録C　R に関する環境

本稿で利用した R に関する環境は以下のようなものである：

R に関する環境

```
> sessionInfo()
R version 3.4.4 (2018-03-15)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS High Sierra 10.13.5

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib

locale:
[1] ja_JP.UTF-8/ja_JP.UTF-8/ja_JP.UTF-8/C/ja_JP.UTF-8/ja_JP.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] dplyr_0.7.6    sparklyr_0.8.2

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.16     dbplyr_1.2.1     compiler_3.4.4  pillar_1.2.2
 [5] later_0.7.2      plyr_1.8.4       bindr_0.1.1     base64enc_0.1-3
 [9] tools_3.4.4      digest_0.6.15    jsonlite_1.5    tibble_1.4.2
[13] nlme_3.1-137     lattice_0.20-35  pkgconfig_2.0.1 rlang_0.2.0
```

```
[17] psych_1.8.4       cli_1.0.0         shiny_1.0.5      DBI_1.0.0
[21] rstudioapi_0.7    yaml_2.1.19       parallel_3.4.4   bindrcpp_0.2.2
[25] withr_2.1.2       stringr_1.3.0     httr_1.3.1       rprojroot_1.3-2
[29] grid_3.4.4        tidyselect_0.2.4  glue_1.2.0       R6_2.2.2
[33] foreign_0.8-70    reshape2_1.4.3    purrr_0.2.4      tidyr_0.8.0
[37] magrittr_1.5      backports_1.1.2   promises_1.0.1   htmltools_0.3.6
[41] assertthat_0.2.0  mnormt_1.5-5      mime_0.5         xtable_1.8-2
[45] httpuv_1.4.2      config_0.3        utf8_1.1.3       stringi_1.2.2
[49] openssl_1.0.1     lazyeval_0.2.1    broom_0.4.4      crayon_1.3.4
```

付録D　データ処理関数

R における data.frame オブジェクトや Spark における Spark DataFrame に対する列選択や行選択などの処理を行う関数を，SparkR，dplyr，R（標準）にそれぞれ用意されているものを以下の表にまとめる．

表 1　処理関数

処理	SparkR	dplyr	R 標準
列選択	select	select	df[,"colName"]
行選択	filter	filter	df[condition,]
列追加	withColumn	mutate	df$colName<-col
並べ替え	orderBy	arrange	NA
グループ化	groupBy	group_by	NA
集計	agg, summarize	summarize	NA
結合	join	join	merge

付録E　抽出変数名対応表

データベース Osiris から抽出対象となった変数の説明を表 2 に与える．

表 2 ：抽出変数名対応表

No.	BvD Variable Name	R Variable Name	R Variable Name for Spark	変数説明
1	firm	firm	firm	企業名
2	year(USD) year.	USD	year_USD	年（通貨単位）
3	BvD ID number	ID	ID	企業コード
4	Address of incorp. - Country	country	country	国名

5	US SIC, Primary code(s) (M)	SIC.code	SIC_code	業種コード
6	US SIC, primary code(s) description	SIC.name	SIC_name	業種名
7	Main exchange	exchange	exchange	主取引所
8	Consolidation code	cons	cons	連結・単独
9	Closing date	date	date	決算日
10	Number of months	month	month	月数
11	Audit Status	audit	audit	監査
12	Accounting standard	practice	practice	会計基準
13	Source	source	source	データの出所
14	Statement unit	units	units	単位（価格）
15	Currency of the statement	currncy	currncy	現地通貨
16	Exchange Rate from Local Currency exchange.	rate	exchange_rate	換算レート
17	Fixed Assets	assets.fix	assets_fix	固定資産
18	Intangible Fixed Assets	assets.int	assets_int	無形固定資産
19	Tangible Fixed Assets	assets.tang	assets_tang	有形固定資産
20	Other Fixed Assets	assets.other.fix	assets_other_fix	その他の固定資産
21	Current Assets	assets.cur	assets_cur	流動資産
22	Stock	stock	stock	株式
23	Debtors	debtors	debtors	売掛金
24	Others	assets.other.cur	assets_other_cur	その他の流動資産
25	Cash & Cash Equivalent	cash	cash	現金及び現金同等物
26	Total Assets	assets.total	assets_total	資産合計
27	Shareholders Funds	shareholders	shareholders	株主資本（≒純資産）
28	Capital	capital	capital	資本
29	Other	shareholders.other	shareholders_other	その他の株主資本
30	Non Current Liabilities	liabilities.non-cur	liabilities_non-cur	非流動負債
31	Long Term Debt	debt.long	debt_long	固定負債
32	Other Non Current Liabilities	liabilities.other.non-cur	liabilities_other_non-cur	その他の非流動負債
33	Provisions	provisions	provisions	引当金
34	Current Liabilities	liabilities.cur	liabilities_cur	流動負債
35	Loans	loans	loans	借入金
36	Creditors	creditors	creditors	買掛金
37	Other	liabilities.other.cur	liabilities_other_cur	その他の流動負債
38	Total Shareh. Funds & Liab.	total.s.l	total_s_l	負債純資産合計
39	Working Capital	capital.working	capital_working	運転資本
40	Net Current Assets	assets.net.cur	assets_net_cur	正味流動資産
41	Enterprise Value	enterprise.value	enterprise_value	企業価値
42	Number of Employees	employees	employees	従業員数
43	Operating Revenue / Turnover	operating.revenue	operating_revenue	営業収益
44	Sales	sales	sales	売上高
45	Costs of Goods Sold	costs.goods	costs_goods	売上原価
46	Gross Profit	profit.gross	profit_gross	売上総利益
47	Other Operating Items	expenses.other	expenses_other	その他の営業費用
48	Operating P/L	EBIT	EBIT	営業利益
49	Financial Revenue	revenue.fin	revenue_fin	金融収益
50	Financial P/L	PL.fin	PL_fin	金融収支

51	Other non Oper./Financial Items	PL.other	PL_other	その他の営業外損益
52	P/L before Tax	PL.before.tax	PL_before_tax	税引前利益
53	Taxation	tax	tax	税金
54	P/L after Tax	PL.after.tax	PL_after_tax	税引後利益
55	Extraord. & Oth. Items	PL.extr	PL_extr	特別収支
56	P/L for Period	net.income	net_income	純利益
57	Material Costs	costs.material	costs_material	原材料費
58	Costs of Employees	costs.employees	costs_employees	人件費
59	Depreciation/Amortization	depr.amor	depr_amor	有形及び無形資産償却
60	Financial Expenses	interest.paid	interest_paid	支払利息
61	Research & Development expenses (memo)	R.D	R_D	研究開発費
62	Cash Flow	cash.flow	cash_flow	キャッシュフロー
63	Added Value	add.value	add_value	付加価値
64	EBITDA	EBITDA	EBITDA	EBITDA
65	Income taxes	tax.income	tax_income	法人税
66	Income Tax Payable	tax.pay.income	tax_pay_income	未払法人税
67	Deferred Taxes	tax.deferred	tax_deferred	繰延税金
68	Def. Inc. Taxes & Invest. Tax Credit	tax.credit	tax_credit	法人税等調整額&投資税額控除
69	Date of current Market capitalisation	date.cur.market.cap	date_cur_market_cap	株式時価総額の日付
70	Current market capitalisation	market.cap.cur	market_cap_cur	時価総額（現在）
71	Market price - Year	year.mp	year_mp	年（市場価格）
72	Market price - January	market.price1	market_price1	市場価格１月末分
73	Market price - February	market.price2	market_price2	市場価格２月末分
74	Market price - March	market.price3	market_price3	市場価格３月末分
75	Market price - April	market.price4	market_price4	市場価格４月末分
76	Market price - May	market.price5	market_price5	市場価格５月末分
77	Market price - June	market.price6	market_price6	市場価格６月末分
78	Market price - July	market.price7	market_price7	市場価格７月末分
79	Market price - August	market.price8	market_price8	市場価格８月末分
80	Market price - September	market.price9	market_price9	市場価格９月末分
81	Market price - October	market.price10	market_price10	市場価格10月末分
82	Market price - November	market.price11	market_price11	市場価格11月末分
83	Market price - December	market.price12	market_price12	市場価格12月末分
84	Market Cap.	market.cap	market_cap	期末時価総額