財務データとESGレーティングデータの前処理と結合

地道正行
阪智香

要　旨

本稿では，Bureau van Dijk社のデータベースOsirisから抽出された財務データとFTSE Russell社のESGレーティングデータの前処理を動的に行う方法について検討し，これらのデータを結合することを考える．さらに，これらの全ての処理を，再現可能研究の視点から実行する方策について議論する．

キーワード：財務データ（Financial Data），ESGレーティングデータ（ESG Rating Data），前処理（Preprocessing），結合（Join），再現可能性（Reproducibility）

I　はじめに

投資意思決定に環境・社会・ガバナンス（Environment, Social, Governance: ESG）要素を反映させようとする国連責任投資原則（Principles for Responsible Investment: PRI）の公表（2006年）を期に，ESG投資が拡大している．これは，企業の長期的成長に必要となるESGへの取り組みに着目し，市場，企業，ステークホルダー，ひいては社会全体の持続可能性を確保しようとするものである．ESG投資にあたっては，企業の財務データに加えて，ESGデータを考慮した投資意思決定が必要となる．そこで，本稿では，Bureau van Dijk（BvD）社[1)]のデータベースOsirisから連結決算主体で抽出さ

れた世界の全上場企業の財務データセット（以下，DS-Osiris-C-2020 と略）[2] と，FTSE Russell 社[3] の ESG レーティングデータセット（以下，DS-FTSE-Russell-2020 と略）の前処理と，これらの処理されたデータを結合することに関して詳細を述べる．なお，本稿では，データの処理を UNIX 系オペレーティングシステム[4] のもとでシェルスクリプト，コマンド，データ解析環境 R を利用して実行している[5]．この仕様は，前処理と結合の再現性を重視しているためであり，この意味で再現可能研究の視点に立って行われる[6]．

なお，付録 A には，データの前処理・結合を行うためのディレクトリとファイルの構成を与えており，付録 B には R に関する環境についての情報を与えている．また，付録 C には，CSV ファイルと Parquet ファイルの R への読み込みに関して述べている．

II　データセット DS-Osiris-C-2020 の前処理

データセット DS-Osiris-C-2020 の前処理に関しては，地道（2018-a）と本質的に同じものであるが，以下の改良点がある：

（P1）データファイルにおける文字列置換等の処理を行う際に GNU Parallel[7] を利用してジョブを並列化することによって，処理時間を短縮化
（P2）データをシリアライズするために，Apache Parquet[8] 形式を採用

改良点（P1）における，GNU Parallel は（1 台以上の）コンピュータで CPU コアを同時に利用することによってジョブを並列化するためのシェルツール

1） http://www.bvdinfo.com/ja-jp/home
2） 本稿では，データベース Osiris の2020年 3 月版を利用している．
3） https://www.ftserussell.com
4） 本稿では，主な処理を macOS Catalina（バージョン10.15.7）のもとで行っている．
5） コマンドラインでデータ処理や解析を実行することに関しては，Janssens（2014）が詳しい．
6） 本稿は，Sweave を利用し，（Osiris データを除く）前処理とデータの結合を含め，動的文書生成を行い，再現可能性を持たせることによって作成している．
7） https://www.gnu.org/software/parallel/
8） https://parquet.apache.org/

（shell tool）である．最近は，マルチコアをもつCPUが搭載されたコンピュータ環境が一般的に普及していることから，このツールを使うことによって手軽に並列処理を実行できる．詳細については，Tange（2018）を参照のこと．なお，GNU Parallelを利用した規模の大きな財務データの処理については，地道（2020-a）で詳しく議論されているので参照されたい．

また，改良点（P2）におけるApache Parquetは，カラムナーストレージ形式（columnar storage format）の一つであり，データを列（カラム）単位で管理することによって，高速に列を参照することを可能としている．また，列形式の大量のデータを効率的に圧縮することができるため，データの転送時間を短縮することができる．詳細は，Apache ParquetのWebページ（`https://parquet.apache.org/`）を参照されたい．なお，付録Cには，Rへのデータの読み込みをCSVファイルとParquetファイルで比較しているので参照されたい．

データの前処理を実行するためスクリプトは，`Makefile`（コード1）に記述しておき，`make`コマンドを（カレント）ディレクトリ`OsirisC2020`（図14を参照）で実行することによって自動的に処理される．なお，データの処理のフローと`Makefile`ファイルから呼び出されるスクリプトのフローを図1に与える．

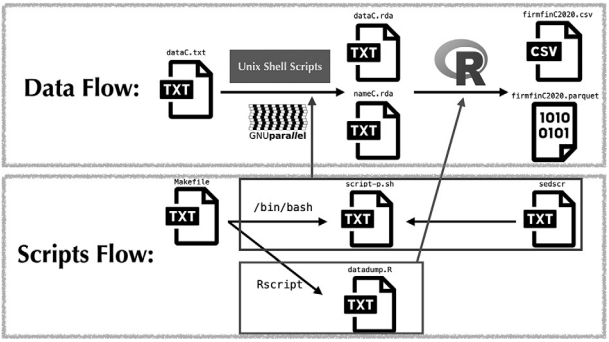

図1：データセットDS-Osiris-C-2020の前処理のデータフローとスクリプトのフロー

コード 1 ：ファイル Makefile（ターゲット：preprocessOsiris）

```
preprocessOsiris:
	date > start-preprocessOsiris.txt
	/bin/bash ./script-p.sh
	Rscript datadump.R "dataC.rda" "nameC.rda" "firmfinC2020.csv" "
	    firmfinC2020.parquet"
	date > end-preprocessOsiris.txt
```

コード 1 の 1 行目にはターゲット名 preprocessOsiris が与えられており，2 行目で処理の開始時間がテキストファイル start-preprocessOsiris.txt に書き出されている．これに対をなす 5 行目で処理の終了時間がテキストファイル end-preprocessOsiris.txt に記録され，両方のファイルを記録することによって，処理時間を計測することができる．また，3 行目で指定されているシェル・スクリプト・ファイル script-p.sh（コード 2）を使って行われる処理は，図 1 のデータフローの左半部分に対応している．

コード 2 ：シェル・スクリプト・ファイル script-p.sh

```
#!/bin/bash
# copy original data to dataC.txt
cp ./rawdata/SJ_Project_2020_OS_C_96377.asc dataC.txt
#
#echo "Remove BOM codes"
gsed -i -s -e '1s/^\xef\xbb\xbf//' dataC.txt
#
echo "dos2unix"
parallel --pipepart -k --block 100M -a dataC.txt "dos2unix" > tmp
echo "separate␣data␣file"
parallel --pipepart -k --block 100M -a tmp 'grep␣-E␣"th\sUSD\)"' > dataC.
    part
parallel --pipepart -k --block 100M -a tmp 'grep␣-v␣-E␣"th\sUSD\)"' > nameC
    .part
echo "replacement␣special␣character"
parallel --pipepart -k --block 100M -a dataC.part "sed␣-f␣sedscr" > tmp
parallel --pipepart -k --block 100M -a tmp "sed␣s/^$'\t'//g" > dataC.rda
parallel --pipepart -k --block 100M -a nameC.part "sed␣s/#//g" > nameC.rda
rm tmp
```

データフローで行われている処理は，データセット DS-Osiris-C-2020 が納められたテキストファイル dataC.txt（コード 3）に対して，文字コードの変換やデータ部分とヘッダー部分への分離，文字列置換などを GNU Parallel

を使って実行している（これらの個々の処理の詳細は，地道，2018-a を参照されたい）．なお，テキストファイル dataC.txt は，1 行目に変数名が与えられており，2 行目以降にデータが与えられている（コード 3 の 1 行目と 2 行目をそれぞれ参照のこと）．また，14行目で指定されている文字列置換のための正規表現が納められたファイル sedscr はコード 4 に与えられている．

コード 3：データセット DS-Osiris-C-2020 が納められたテキストファイル dataC.txt（先頭 2 行を掲載）

```
1 AGGREGATE INDUSTRIES HOLDINGS LIMITED            BvD ID number       Address
    of incorp. - Country ISO code      US SIC, Primary code(s) (M)      US SIC,
    primary code(s) description Main exchange       Consolidation     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 end Monthly - Closing price -
    January   Monthly - Closing price - February   Monthly - Closing price -
    March     Monthly - Closing price - April      Monthly - Closing price -
    May Monthly - Closing price - June      Monthly - Closing price - July
          Monthly - Closing price - August     Monthly - Closing price -
    September Monthly - Closing price - October    Monthly - Closing price -
    November  Monthly - Closing price - December   Market capitalisation (mil
    )    Ordinary dividends      IPO date     Delisted date      Date of
    incorporation    ISIN number Ticker symbol      GICS code    GICS
    description
2      1990(th USD)      GB00182412   GB     3272  Concrete products, except
         block and brick Delisted     Cons  31/12/1990  12     Qualif n.a.
         Acc. Std na AR     th     GBP    1.92801     802,958      n.a.
         793,472     9,486 246,361      49,762     81,092      115,507
         70,874      1,049,318   500,067      132,223     367,845
         327,453     267,800      59,653       33,875     221,798     74,460
               58,862       88,476     1,049,318    71,992      24,563
         n.a.  3,529 490,505     487,998      -262,228    228,276
         -109,916    96,111      10,199       -16,041     4,627 84,697
```

```
-17,005     67,692     -15,019     52,673     n.a.  -108,200
-22,249     -26,240    0     74,922     226,367    118,360
-17,005     19,511     23,483     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.  -28,265     25/03/1973  22/04/2005  1922  GB0003238267
AGG  20102010    BUILDING PRODUCTS
```

コード 4 ：文字列置換のための正規表現が納められたファイル sedscr

```
1  # 置換 n.a.  -> NA
2  s/n\.a\./NA/g
3  # 置換 , -> なし
4  s/,//g
5  # 置換 # -> なし
6  s/#//g
7  # 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
8  s/                    /    NA    NA    NA    NA    /g
9  # 置換 タブタブタブタブ -> タブNAタブNAタブNAタブ
10 s/                   /    NA    NA    NA    /g
11 # 置換 タブタブタブ -> タブNAタブNAタブ
12 s/                 /    NA    NA    /g
13 # 置換 タブタブ -> タブNAタブ
14 s/           /    NA    /g
```

コード 1 の 4 行目は，R を利用してデータ変換（data transformation）を行っている．ここで，`Rscript` コマンドの第 1 引数で呼び出されている R スクリプト `datadump.R`（コード 5 ）を使って行われる処理は，図 1 におけるデータフローの右半部分に対応している．ここで行われる処理は，細かい箇所で相違があるものの，地道（2018-a）に与えられているものと同様であるため，詳細は割愛する．`Rscript` コマンドの第 2 ，3 番目の引数には，コード 1 の 3 行目を実行することによって与えられた中間ファイル `dataC.rda`（処理されたデータ部分の RDA ファイル），`nameC.rda`（処理されたヘッダー部分の RDA ファイル）を指定している．ここで，RDA ファイルは，R データ（R Data）ファイルであり，適当な分離記号でデータ間が区切られたテキストファイルである（拡張子 `rda`）．さらに，第 4 ，5 番目の引数には，前処理の結果として出力されるファイル名 `firmfinC2020.csv`（前処理された OsirisC 2020 データの CSV ファイル），`firmfinC 2020.parquet`（前処理された OsirisC2020 データの Apache Parquet 形式ファイル）が与えられている．

コード 5 ：R スクリプトファイル datadump.R

```
require(dplyr)
require(readr)
require(arrow)
args <- commandArgs(trailingOnly = T)
tmp1 <- read_tsv(args[1],na="NA", quote="", col_names=FALSE,
col_types = cols(
.default = col_double(),
X1 = col_character(),
X2 = col_character(),
X3 = col_character(),
X4 = col_integer(),
X5 = col_character(),
X6 = col_character(),
X7 = col_character(),
X8 = col_character(),
X9 = col_integer(),
X11 = col_character(),
X12 = col_character(),
X13 = col_character(),
X14 = col_character(),
X15 = col_character(),
X68 = col_character(),
X85 = col_character(),
X86 = col_character(),
X87 = col_character(),
X88 = col_character(),
X89 = col_character(),
X90 = col_integer(),
X91 = col_character())
)
tmp2 <- read_tsv(args[2],na="NA", quote="", col_names=FALSE) %>% data.frame
    ()
firmname <- tmp2[,1]
firmfin.raw.frame <- data.frame(rep(firmname, each=30), tmp1)
varnames <- scan("variablenameOsirisforSpark2020.txt", what = "")
colnames(firmfin.raw.frame) <- varnames
firmfin.frame <- mutate(firmfin.raw.frame ,
              SIC_code = rep(SIC_code[seq(1,dim(firmfin.raw.frame)[1],30)],
                  each = 30),
              SIC_name = rep(SIC_name[seq(1,dim(firmfin.raw.frame)[1],30)],
                  each =30),
              costs_goods = -costs_goods ,
              expenses_other = -expenses_other,
              tax = -tax,
              tax_income = -tax_income,
              costs_material = -costs_material,
              costs_employees = -costs_employees,
              depr_amor = -depr_amor,
              interest_paid = -interest_paid,
              R_D = -R_D,
              firmID = paste(firm,ID),
              year = rep(seq(1990,2019), length(firmname)))
write_csv(x = firmfin.frame, path = args[3])
write_parquet(firmfin.frame,args[4])
```

先にも述べたが，データセット DS-Osiris-C-2020 の前処理を行うためには，make コマンドを（カレント）ディレクトリ OsirisC2020（図14を参照）で以下のように実行することによって自動的に処理される．

make コマンド実行：ターゲット preprocessOsiris

```
% make preprocessOsiris
```

この実行時間は make コマンドの実行時に出力される start-preprocessOsiris.txt と end-preprocessOsiris.txt を比較することによって以下のようにわかる：

ターゲット preprocessOsiris に対する make コマンド実行時間

```
% cat start-preprocessOsiris.txt
2020年 4月16日 木曜日 11時32分32秒 JST
% cat end-preprocessOsiris.txt
2020年 4月16日 木曜日 11時35分37秒 JST
```

この結果から，処理時間は 3 分 5 秒であることがわかる[9]．

Ⅲ　データセット DS-FTSE-Russell-2020 の前処理

FTSE Russell ESG レーティングデータは，日々更新されているが，毎年 6 月末と12月末に大きな更新がおこなわれる[10]．データセットは，新興国（emerging country）の指標（FTSE Emerging ESG Index）と先進国（developed country）の指標（FTSE Developed ESG Index）が納められたファイルに分かれており，それぞれのファイル名におけるコーディングは，E（Emerging），G（Global）である．さらに，指標の構成は，以下のようなも

9）MacBook Pro 2018（OS: macOS Catalina（バージョン10.15.6），CPU: Intel(R) Core (TM) i9，2.9GHz）で計測した．ただし，arrow パッケージのバージョンを2.0.0にした場合は，処理時間が 3 分26秒になっている．これは，write_parquet 関数による Parquet ファイルの書き出し時間が遅くなったことに起因するものと考えられる（付録 C の脚注も参照されたい）．

10）本稿で扱っているデータのうち，最初のデータである2015年は 9 月に発表されている．

のである：

（S）1つのESGレーティング（summary: 要約）の指標
（P）3つのピーラー（pillar; 支柱）である「環境」（Environment），「社会」（Social），「ガバナンス」（Governance）の指標
（T）「気候変動」（Climate Change），「健康・安全」（Health & Safety），「税の透明性」（Tax Transparency）等の14個の項目からなるテーマ（themes）の指標

なお，これらのファイル名におけるコーディングは，それぞれ，S，P，Tである．詳細はFTSE Russell（2019）を参照されたい．

例えば，2020年度のデータが納められたCSVファイル名は以下のようなものである．

新興国・先進国\構成	ESGレーティング指標（S）	ピーラー（P）	テーマ（T）
新興国（E）	ESGRESJuly2020.csv	ESGREPJuly2020.csv	ESGRETJuly2020.csv
先進国（G）	ESGRGSJuly2020.csv	ESGRGPJuly2020.csv	ESGRGTJuly2020.csv

ここでは，新興国に関する3種類のCSVファイルの説明を与える．まず，ESGレーティング指標のCSVファイルESGRESJuly2020.csvは以下のようなものである．

コード6：2020年7月の新興国のESGレーティング指標のCSVファイル ESGRESJuly2020.csv（一部）

```
1 30/06/2020 (C) FTSE International Limited 2020. All Rights Reserved
    ,,,,,,,,,,,,,,,
2 FTSE ESG Ratings (Emerging) - Summary,,,,,,,,,,,,,,,
3  ,,,,,,,,,,,,,,,
4 Cons Code,SEDOL,ISIN,Local Market,CUSIP,Constituent Name,Country Code,ISO
    Code,Exchange Code,Industry Code,Supersector Code,Sector Code,Subsector
    Code,ESG Rating (Absolute),Previous ESG Rating (Absolute),ESG Rating (
    Supersector Relative),Previous ESG Rating (Supersector Relative)
5 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC SH),CHN ,CN,XSSC
    ,2000,2700,2750,2757,0.7,0.7,1,1
```

ここで，コード6の1行目から3行目はファイルの説明が書かれたヘッダー

部分であり，4行目はデータの列名が与えられている．5行目は，企業毎のESGレーティング指標に関するデータが収録されている．具体的には，360 SECURITY TECHNOLOGY社（360 Security (A) (SC SH)）のデータが納められており，ESGレーティング指標（ESG Rating (Absolute)）の値は，0.7であることがわかる．

また，新興国のピーラーのCSVファイルESGREPJuly2020.csvは以下のようなものである．

コード7：2020年7月の新興国のピーラーのCSVファイル ESGREPJuly2020.csv（一部）

```
1 30/06/2020 (C) FTSE International Limited2020. All Rights Reserved
    ,,,,,,,,,,,,,,,,,,
2 FTSE ESG Ratings (Emerging) - Pillar ,,,,,,,,,,,,,,,,,,
3  ,,,,,,,,,,,,,,,,,,
4 Cons Code,SEDOL,ISIN,Local Market,CUSIP,Constituent Name,Country Code,ISO
    Code,Exchange Code,Industry Code,Supersector Code,Sector Code,Subsector
    Code,Pillar,Exposure,Previous Exposure,Score,Previous Score,Score (
    Supersector Relative),Previous Score (Supersector Relative)
5 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
    ,2000,2700,2750,2757,Environmental Pillar,2,2,0,0,1,1
6 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
    ,2000,2700,2750,2757,Social Pillar,2.3,2.3,0.6,0.6,1,1
7 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
    ,2000,2700,2750,2757,Governance Pillar,2,2,1.5,1.5,1,1
```

ここで，コード7の1行目から3行目はファイルの説明が書かれたヘッダー部分であり，4行目はデータの列名が与えられている．5，6，7行目には，企業毎の3つのピラー（Pillar）のスコア（Score）に関するデータがそれぞれ収録されている．ここでは，360 SECURITY TECHNOLOGY社（360 Security (A) (SC SH)）のデータが納められており，「環境」（Environmental Pillar），「社会」（Social Pillar），「ガバナンス」（Governance Pillar）のスコア値は，それぞれ，0，0.6，1.5であることがわかる．

さらに，新興国のテーマのCSVファイルESGRETJuly2020.csvは以下のようなものである．

コード8：2020年7月の新興国のテーマのCSVファイル ESGRETJuly2020.csv（一部）

```
1  30/06/2020 (C) FTSE International Limited 2020. All Rights Reserved
       ,,,,,,,,,,,,,,,,
2  FTSE ESG Ratings (Emerging) - Theme,,,,,,,,,,,,,,,,
3   ,,,,,,,,,,,,,,,,
4  Cons Code,SEDOL,ISIN,Local Market,CUSIP,Constituent Name,Country Code,ISO
       Code,Exchange Code,Industry Code,Supersector Code,Sector Code,Subsector
       Code,Theme,Exposure,Previous Exposure,Score,Previous Score
5  C178957,BDF57C1,CNE100002RZ2,601360,,360Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Pollution & Resources,2,2,0,0
6  C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Climate Change,2,2,0,0
7  C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Water Use,2,2,0,0
8  C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Biodiversity ,na,na,na,na
9  C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Environmental Supply Chain,na,na,na,na
10 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Customer Responsibility,na,na,na,na
11 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Labour Standards,3,3,0,0
12 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Health & Safety,2,2,0,0
13 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Human Rights & Community,2,2,2,2
14 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Social Supply Chain,na,na,na,na
15 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Corporate Governance,2,2,3,3
16 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Risk Management,na,na,na,na
17 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Tax Transparency,na,na,na,na
18 C178957,BDF57C1,CNE100002RZ2,601360,,360 Security (A) (SC,CHN ,CN,XSSC
       ,2000,2700,2750,2757,Anti -Corruption ,2,2,0,0
```

ここで，コード8の1行目から3行目はファイルの説明が書かれたヘッダー部分であり，4行目はデータの列名が与えられている．5行目から18行目には，企業毎の14項目からなるテーマ（Theme）のスコア（Score）に関するデータがそれぞれ収録されている．ここでは，360 SECURITY TECHNOLOGY社（360Security (A) (SC SH)）のデータが納められており，「汚染と資源」（Pollution & Resources）から「腐敗防止」（Anti-Corruption）のスコア値は，それぞれ，0,...,0であることがわかる．

これらのCSVファイルに関する情報をRを用いて企業毎に結合する処理

を行う．そのためには，各種のパッケージを読み込んでおく必要がある：

パッケージの読み込み

```
> library(readr)
> library(dplyr)
> library(tidyr)
```

この設定のもとで，カレントディレクトリのサブディレクトリ ./data/FTSE2020/csv/ に CSV ファイルがあると仮定し，readr パッケージに付属する read_csv 関数を利用して，以下のようにデータを読み込む：

R への読み込み

```
> dataES202007 <- read_csv("./data/FTSE2020/csv/ESGRESJuly2020.csv", skip=3)
> dataEP202007 <- read_csv("./data/FTSE2020/csv/ESGREPJuly2020.csv", skip=3)
> dataET202007 <- read_csv("./data/FTSE2020/csv/ESGRETJuly2020.csv", skip=3)
```

これらの読み込んだ情報は R のデータフレーム（より正確には tibble 形式）として作業空間に保存される．これらのデータフレームから，企業の ISIN コード（ISIN）や企業名（‘Constituent Name’），国名（‘ISO Code’），産業コード（‘Industry Code’），ESG レーティング指標（‘ESG Rating (Absolute)’），ピラー（Pillar）とそのスコア（Score），テーマ（Theme）とそのスコア（Score）等の列を dplyr パッケージに付属の select 関数で選択し，以下のように新しいデータフレームに付値する：

データフレーム結合

```
> dfES2020 <- dataES202007 %>%
+         select(ISIN, `Constituent Name`, `ISO Code`, `Industry Code`,
+                `ESG Rating (Absolute)`)
> dfEP2020 <- dataEP202007 %>% select(ISIN, Pillar, Score) %>% na.omit() %>%
+         pivot_wider(names_from = "Pillar", values_from = "Score")
> dfET2020 <- dataET202007 %>% select(ISIN, Theme, Score) %>% na.omit() %>%
+         pivot_wider(names_from = "Theme", values_from = "Score")
```

ここで，ピラー（Pillar）とそのスコア（Score）は，列形式（縦型）でデータが納められているので，tidyr パッケージに付属する pivot_wider 関数を利用して，企業毎にこれらのデータを展開する形式（横型）に変換し

ている．また，テーマ（Theme）とそのスコア（Score）も同様の変換を行っている．なお，全体の処理は（パイプ）演算子（%>%）を利用してパイプライン化している．

さらに，これらのオブジェクトを企業毎にISINコードを主キーとし，dplyrパッケージに付属するinner_join関数で結合し，データフレームdfESPT2020に付値する：

データフレームの結合

```
> dfESPT2020 <- dfES2020 %>%
+          inner_join(dfEP2020, by = "ISIN") %>%
+          inner_join(dfET2020, by = "ISIN")
```

なお，全体の処理は（パイプ）演算子（%>%）を利用してパイプライン化している．

2020年の新興国における企業のESGレーティングデータの前処理のフローは，図2のように与えられる．

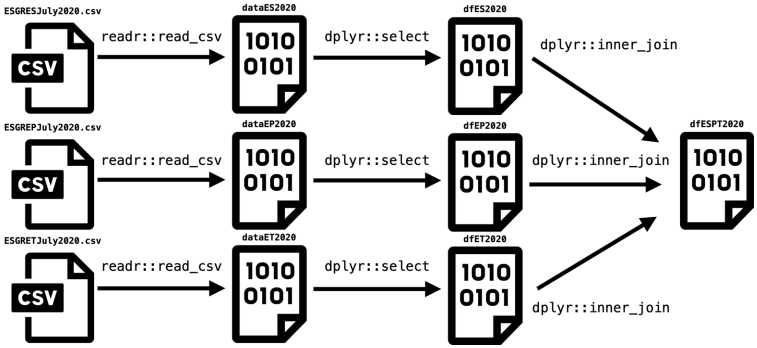

図2：2020年の新興国における企業のESGレーティングデータの前処理のフロー

なお，2020年の新興国における企業のESGレーティングデータの前処理の結果として得られたデータフレームdfESPT2020は以下のようなものである：

データフレーム dfESPT2020

```
> dfESPT2020
# A tibble: 1,778 x 22
   ISIN `Constituent Na… `ISO Code``Industry Code``ESG Rating (Ab… `Environmental … `Social Pillar`
   <chr> <chr>           <chr>              <dbl> <chr>          <chr>            <chr>
 1 CNE1… 360 Security (A… CN                  2000 0.7            0                0.6
 2 INE4… 3M India         IN                  2000 2              1                2
 3 KYG8… 3SBio (P Chip)   CN                  4000 2.1            1.5              1.7
 4 US31… 51job ADS (N Sh… CN                  2000 1.2            1                0.3
 5 US31… 58.com ADS (N S… CN                  5000 1.9            1                1
 6 INE1… ABB India        IN                  2000 3.7            2.6              4.1
 7 SA12… Abdullah Al Oth… SA                  5000 0.5            0                0.3
 8 PHY0… Aboitiz Power    PH                  7000 2.5            1.6              2
 9 ZAE0… Absa Group Limi… ZA                  8000 4.3            5                3.8
10 AEA0… Abu Dhabi Comme… AE                  8000 3              3                1.8
# … with 1,768 more rows, and 15 more variables: `Governance Pillar` <chr>, `Pollution & Resources` <chr>,
#   `Climate Change` <chr>, `Water Use` <chr>, Biodiversity <chr>, `Environmental Supply Chain` <chr>,
#   `Customer Responsibility` <chr>, `Labour Standards` <chr>, `Health & Safety` <chr>, `Human Rights &
#   Community` <chr>, `Social Supply Chain` <chr>, `Corporate Governance` <chr>, `Risk Management` <chr>,
#   `Tax Transparency` <chr>, `Anti-Corruption` <chr>
```

この処理と同様の処理を各「年」(2015〜2020) と「新興国」(E) または「先進国」(G) に対して実行することによって，新興国に対するデータフレーム dfESPT2015〜dfESPT2020 と先進国に対するデータフレーム dfGSPT2015〜dfGSPT2020 が作成できる．

次に，これらのデータフレームを結合することを考える．まず，新興国に対するデータフレーム dfESPT2015〜dfESPT2020 に年の情報を与える列を dplyr パッケージに付属する mutate 関数で追加し，dplyr パッケージに付属する bind_rows 関数で行結合したものを，さらに dplyr パッケージに付属する arrange 関数で ISIN コードと年で並べ替えを行う：

データフレーム dfESPT2015〜dfESPT2020 の結合

```
> dfESPT2015 <- dfESPT2015 %>% dplyr::mutate(year = 2015)
> dfESPT2016 <- dfESPT2016 %>% dplyr::mutate(year = 2016)
> dfESPT2017 <- dfESPT2017 %>% dplyr::mutate(year = 2017)
> dfESPT2018 <- dfESPT2018 %>% dplyr::mutate(year = 2018)
> dfESPT2019 <- dfESPT2019 %>% dplyr::mutate(year = 2019)
> dfESPT2020 <- dfESPT2020 %>% dplyr::mutate(year = 2020)
> Eall <- dplyr::bind_rows(dfESPT2015, dfESPT2016, dfESPT2017,
+                          dfESPT2018, dfESPT2019, dfESPT2020) %>%
+                                  dplyr::arrange(ISIN, year)
```

さらに，dplyr パッケージに付属する mutate，mutate_at 関数を利用して，各列の型変換を行い，データフレーム SPT.E へ付置する：

列の型変換とデータフレームSPT.Eへの付値

```
> SPT.E <- Eall %>%
+         mutate_at(
+         vars(-ISIN, -`Constituent Name`, -`ISO Code`, -`Industry Code`),
+         as.numeric) %>%
+                 mutate(`Industry Code` = as.factor(`Industry Code`))
```

これらの結合工程のフローは，図3のように与えられる．

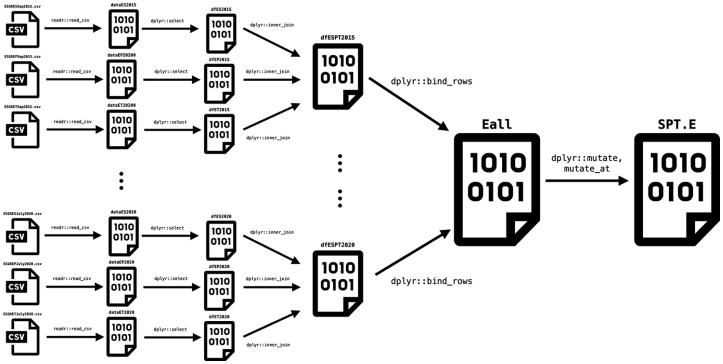

図3：2015年から2020年の新興国における企業のESGレーティングデータの結合

同様に先進国のデータについても処理することができて，処理のフローは図4のように与えられる．

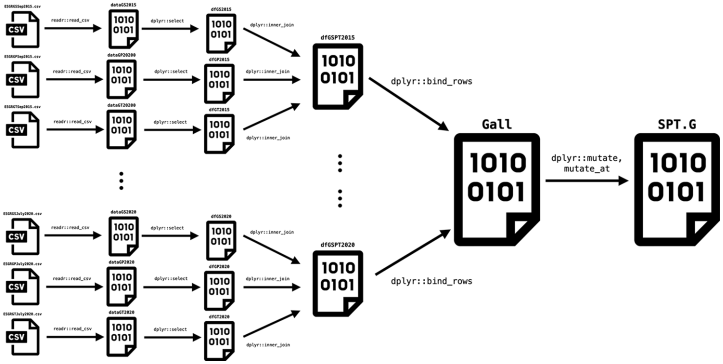

図4：2015年から2020年の先進国における企業のESGレーティングデータの結合

この処理を実行するRスクリプトは以下のようなものである：

データフレーム dfGSPT2015〜dfGSPT2020 の結合，型変換とデータフレーム SPT.G への付値

```
> dfGSPT2015 <- dfGSPT2015 %>% dplyr::mutate(year = 2015)
> dfGSPT2016 <- dfGSPT2016 %>% dplyr::mutate(year = 2016)
> dfGSPT2017 <- dfGSPT2017 %>% dplyr::mutate(year = 2017)
> dfGSPT2018 <- dfGSPT2018 %>% dplyr::mutate(year = 2018)
> dfGSPT2019 <- dfGSPT2019 %>% dplyr::mutate(year = 2019)
> dfGSPT2020 <- dfGSPT2020 %>% dplyr::mutate(year = 2020)
> Gall <- dplyr::bind_rows(dfGSPT2015, dfGSPT2016, dfGSPT2017,
+                  dfGSPT2018, dfGSPT2019, dfGSPT2020) %>%
+                        dplyr::arrange(ISIN, year)
> SPT.G <- Gall %>%
+         mutate_at(
+         vars(-ISIN, -`Constituent Name`, -`ISO Code`, -`Industry Code`),
+         as.numeric) %>%
+                  mutate(`Industry Code` = as.factor(`Industry Code`))
```

以上の処理によって得られた新興国と先進国のデータフレーム SPT.E，SPT.G を最終的に結合する．ここでは，この結合のための関数 merge.SPT を用意した：

データフレーム SPT.E，SPT.G を結合のための関数 merge.SPT

```
> merge.SPT <- function(x = SPT.E, y = SPT.G)
+{
+   require(dplyr)
+   tmp1 <- x %>% mutate(type = "E")
+   tmp2 <- y %>% mutate(type = "G")
+   # merge
+   SPT <- bind_rows(tmp1, tmp2)
+   # Rename of variables
+   colnames(SPT) <- c("ISIN", "firm", "country", "industry",
+                      "ESG", "E", "S", "G",
+                      "EPR", "ECC", "EWU", "EBD", "ESC",
+                      "SCR", "SLS", "SHS", "SHR", "SSC",
+                      "GCG", "GRM", "GTX", "GAC",
+                      "year", "type")
+   SPT
+ }
```

関数 merge.SPT は，データフレーム SPT.E，SPT.G を（行）結合するだけでなく，その企業が新興国（E）と先進国（G）のどちらのタイプに属すものであるかを判別する列 type を追加してから，結合しており，さらに結合されたオブジェクトの列名を関数 colnames を使ってリネームしている．列名については，表1を参照されたい．

表1：SPTオブジェクトの列名

列名（オリジナル）	列名（改名）	説明	備考
ISIN	ISIN	ISINコード	
ConstituentName	firm	社名	
ISOCode	country	国名	ISOコード
IndustryCode	industry	産業コード	icb（Industry Classification Benchmark）コード
ESGRating(Absolute)	ESG	ESGサマリースコア	
EnvironmentalPillar	E	ピラースコア（環境）	環境
SocialPillar	S	ピラースコア（社会）	社会
GovernancePillar	G	ピラースコア（ガバナンス）	ガバナンス
Pollution&Resources	EPR	テーマスコア（汚染と資源）	環境
ClimateChange	ECC	テーマスコア（気候変動）	環境
WaterUse	EWU	テーマスコア（水の使用）	環境
Biodiversity	EBD	テーマスコア（生物多様性）	環境
EnvironmentalSupplyChain	ESC	テーマスコア（環境サプライチェーン）	環境
CustomerResponsibility	SCR	テーマスコア（顧客に対する責任）	社会
LabourStandards	SLS	テーマスコア（労働基準）	社会
Health&Safety	SHS	テーマスコア（健康と安全）	社会
HumanRights&Community	SHR	テーマスコア（人権と地域社会）	社会
SocialSupplyChain	SSC	テーマスコア（社会サプライチェーン）	社会
CorporateGovernance	GCG	テーマスコア（コーポレートガバナンス）	ガバナンス
RiskManagement	GRM	テーマスコア（リスク管理）	ガバナンス
TaxTransparency	GTX	テーマスコア（税の透明性）	ガバナンス
Anti-Corruption	GAC	テーマスコア（腐敗防止）	ガバナンス
year	year	年	

この関数を実行することによって得られるデータフレームSPTは以下のようなものである：

関数merge.SPTの実行とデータフレームSPT

```
> SPT <- merge.SPT()
> SPT
# A tibble: 20,232 x 24
   ISIN  firm  country industry   ESG    E     S    G
   <chr> <chr> <chr>   <fct>    <dbl> <dbl> <dbl> <dbl>
 1 AEA0… Abu … AE      8000       1.8    2     1.6   1.9
 2 AEA0… Abu … AE      8000       2.9    3     1.6   3.7
 3 AEA0… Abu … AE      8000       3      3     2.2   3.7
 4 AEA0… Abu … AE      8000       3      2     2.3   4
 5 AEA0… Abu … AE      8000       3.4    3     2.7   4.3
 6 AEA0… Abu … AE      8000       3      3     1.8   4
 7 AEA0… Waha… AE      2000       0.5    0     0.6   0.9
 8 AEA0… Waha… AE      2000       0.8    0.4   0.5   1.7
 9 AEA0… Waha… AE      2000       1.2    0.4   0.8   2.7
10 AEA0… Waha… AE      8000       1      0     0.4   2.7
# … with 20,222 more rows, and 16 more variables:
#  EPR <dbl>, ECC <dbl>, EWU <dbl>, EBD <dbl>, ESC <dbl>,
#  SCR <dbl>, SLS <dbl>, SHS <dbl>, SHR <dbl>, SSC <dbl>,
#  GCG <dbl>, GRM <dbl>, GTX <dbl>, GAC <dbl>, year <dbl>,
#  type <chr>
```

以上の処理によって最終的に得られた SPT は，各行に企業の ESG レーティングデータを持つ20232行，24列のデータフレームである．最後に，このデータフレームを，readr パッケージの関数 write_csv を使って CSV ファイル SPT2020.csv へ出力する．

データフレーム SPT を CSV ファイル SPT2020.csv へ出力

```
> write_csv(SPT, "SPT2020.csv")
```

以上のデータの前処理全体を実行するためスクリプトは，Makefile（コード 9）に記述しておき，make コマンドを実行することによって自動的に処理される．なお，データセット DS-FTSE-Russell-2020 の前処理のフローとファイル Makefile から呼び出されるスクリプトのフローを図 5 に与える．

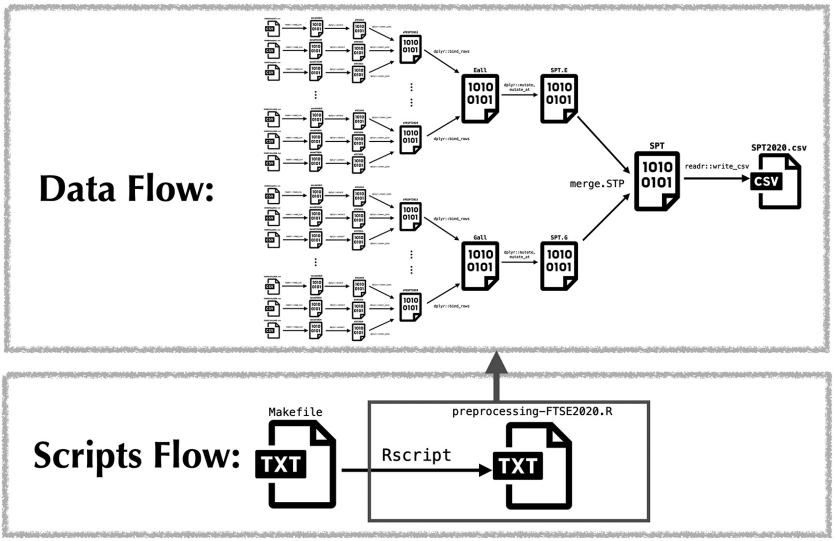

図 5 ：データセット DS-FTSE-Russell-2020 の前処理のデータフローとスクリプトのフロー

コード9：ファイル Makefile（ターゲット：preprocessFTSE）

```
preprocessFTSE:
	date > start-preprocessFTSE.txt
	Rscript preprocessing-FTSE2020.R
	date > end-preprocessFTSE.txt
```

コード9の1行目にはターゲット名 preprocessFTSE が与えられており，2行目で処理の開始時間がテキストファイル start-preprocessFTESE.txt に書き出されている．これに対をなす4行目で処理の終了時間がテキストファイル end-preprocessFTESE.txt に記録され，両方のファイルを比較することによって，処理時間を計測することができる．3行目で指定されている Rscript コマンドの引数で呼び出されている R スクリプト preprocessing-FTSE2020.R（コード10）を使って行われる処理は，図5におけるデータフロー全体に対応している．

コード10：R スクリプトファイル preprocessing-FTSE2020.R（一部）

```
library(readr)
library(dplyr)
library(tidyr)

#-----------------
# 2015 Emerging
#-----------------
dataES201507 <- read_csv("./csv/ESGRESSep2015.csv", skip=3)
dataEP201507 <- read_csv("./csv/ESGREPSep2015.csv", skip=3)
dataET201507 <- read_csv("./csv/ESGRETSep2015.csv", skip=3)

dfES2015 <- dataES201507 %>% select(ISIN, `Constituent Name`, `ISO Code`, `
    Industry Code`, `ESG Rating (Absolute)`)
dfEP2015 <- dataEP201507 %>% select(ISIN, Pillar, Score) %>% na.omit() %>%
    pivot_wider(names_from = "Pillar", values_from = "Score", names_sort =
    TRUE)
dfET2015 <- dataET201507 %>% select(ISIN, Theme, Score) %>% na.omit() %>%
    pivot_wider(names_from = "Theme", values_from = "Score", names_sort =
    TRUE)
dfESPT2015 <- dfES2015 %>% inner_join(dfEP2015, by = "ISIN") %>% inner_join
    (dfET2015, by = "ISIN")

#-----------------
# 2015 Grobal
#-----------------
dataGS201507 <- read_csv("./csv/ESGRGSJuly2015.csv", skip=3)
dataGP201507 <- read_csv("./csv/ESGRGPJuly2015.csv", skip=3)
dataGT201507 <- read_csv("./csv/ESGRGTJuly2015.csv", skip=3)

```

```
dfGS2015 <- dataGS201507 %>% select(ISIN, `Constituent Name`, `ISO Code`, `Industry Code`, `ESG Rating (Absolute)`)
dfGP2015 <- dataGP201507 %>% select(ISIN, Pillar, Score) %>% na.omit() %>% pivot_wider(names_from = "Pillar", values_from = "Score", names_sort = TRUE)
dfGT2015 <- dataGT201507 %>% select(ISIN, Theme, Score) %>% na.omit() %>% pivot_wider(names_from = "Theme", values_from = "Score", names_sort = TRUE)
dfGSPT2015 <- dfGS2015 %>% inner_join(dfGP2015, by = "ISIN") %>% inner_join(dfGT2015, by = "ISIN")

:
:(中略)
:

#-----------------
# 2020 Emerging
#-----------------
dataES202007 <- read_csv("./csv/ESGRESJuly2020.csv", skip=3)
dataEP202007 <- read_csv("./csv/ESGREPJuly2020.csv", skip=3)
dataET202007 <- read_csv("./csv/ESGRETJuly2020.csv", skip=3)

dfES2020 <- dataES202007 %>% select(ISIN, `Constituent Name`, `ISO Code`, `Industry Code`, `ESG Rating (Absolute)`)
dfEP2020 <- dataEP202007 %>% select(ISIN, Pillar, Score) %>% na.omit() %>% pivot_wider(names_from = "Pillar", values_from = "Score", names_sort = TRUE)
dfET2020 <- dataET202007 %>% select(ISIN, Theme, Score) %>% na.omit() %>% pivot_wider(names_from = "Theme", values_from = "Score", names_sort = TRUE)
dfESPT2020 <- dfES2020 %>% inner_join(dfEP2020, by = "ISIN") %>% inner_join(dfET2020, by = "ISIN")

#-----------------
# 2020 Grobal
#-----------------
dataGS202007 <- read_csv("./csv/ESGRGSJuly2020.csv", skip=3)
dataGP202007 <- read_csv("./csv/ESGRGPJuly2020.csv", skip=3)
dataGT202007 <- read_csv("./csv/ESGRGTJuly2020.csv", skip=3)

dfGS2020 <- dataGS202007 %>% select(ISIN, `Constituent Name`, `ISO Code`, `Industry Code`, `ESG Rating (Absolute)`)
dfGP2020 <- dataGP202007 %>% select(ISIN, Pillar, Score) %>% na.omit() %>% pivot_wider(names_from = "Pillar", values_from = "Score", names_sort = TRUE)
dfGT2020 <- dataGT202007 %>% select(ISIN, Theme, Score) %>% na.omit() %>% pivot_wider(names_from = "Theme", values_from = "Score", names_sort = TRUE)
fGSPT2020 <- dfGS2020 %>% inner_join(dfGP2020, by = "ISIN") %>% inner_join(dfGT2020, by = "ISIN")

# Emerging
dfESPT2015 <- dfESPT2015 %>% dplyr::mutate(year =2015)
dfESPT2016 <- dfESPT2016%>% dplyr::mutate(year =2016)
dfESPT2017 <- dfESPT2017%>% dplyr::mutate(year =2017)
dfESPT2018 <- dfESPT2018%>% dplyr::mutate(year =2018)
```

```
dfESPT2019 <- dfESPT2019%>% dplyr::mutate(year =2019)
dfESPT2020 <- dfESPT2020%>% dplyr::mutate(year =2020)
Eall <- dplyr::bind_rows(dfESPT2015, dfESPT2016, dfESPT2017, dfESPT2018,
    dfESPT2019, dfESPT2020) %>% dplyr::arrange(ISIN, year)

SPT.E <- Eall %>% mutate_at(vars(-ISIN, -`Constituent Name`, -`ISO Code`,
    -`Industry Code`), as.numeric) %>% mutate( `Industry Code` = as.factor(`
    Industry Code`))

# Global
dfGSPT2015 <- dfGSPT2015 %>% dplyr::mutate(year =2015)
dfGSPT2016 <- dfGSPT2016 %>% dplyr::mutate(year =2016)
dfGSPT2017 <- dfGSPT2017 %>% dplyr::mutate(year =2017)
dfGSPT2018 <- dfGSPT2018 %>% dplyr::mutate(year =2018)
dfGSPT2019 <- dfGSPT2019 %>% dplyr::mutate(year =2019)
dfGSPT2020 <- dfGSPT2020 %>% dplyr::mutate(year =2020)

Gall <- dplyr::bind_rows(dfGSPT2015, dfGSPT2016, dfGSPT2017, dfGSPT2018,
    dfGSPT2019, dfGSPT2020) %>%
  dplyr::arrange(ISIN, year)

SPT.G <- Gall %>% mutate_at(vars(-ISIN, -`Constituent Name`, -`ISO Code`,
     -`Industry Code`), as.numeric) %>%
  mutate(`Industry Code` = as.factor(`Industry Code`))

merge.SPT <- function(x = SPT.E, y = SPT.G)
{
  require(dplyr)
  tmp1 <- x %>% mutate(type = "E")
  tmp2 <- y %>% mutate(type = "G")
  # merge
  SPT <- bind_rows(tmp1, tmp2)
  # Rename of variables
  colnames(SPT) <- c("ISIN", "firm", "country", "industry",
                     "ESG", "E", "S", "G",
                     "EPR", "ECC", "EWU", "EBD", "ESC",
                     "SCR", "SLS", "SHS", "SHR", "SSC",
                     "GCG", "GRM", "GTX", "GAC",
                     "year", "type")
  SPT
}

# Summary Pillar Theme (SPT)
SPT <- merge.SPT()

# Data Dump
write_csv(SPT, "SPT2020.csv")
```

以上の設定のもとで，make コマンドを（カレント）ディレクトリ FTSE 2020（図14を参照）において以下のように実行することによってデータセット DS-FTSE-Russell-2020 の前処理が自動的に行なわれる．

make コマンド実行：ターゲット preprocessFTSE

```
% make preprocessFTSE
```

この実行時間は make コマンドの実行時に出力される start-preprocessFTSE.txt と end-preprocessFTSE.txt を比較することによって以下のようにわかる：

ターゲット preprocessFTSE に対する make コマンド実行時間

```
% cat start-preprocessFTSE.txt
2020年 11月 9日 月曜日 18時 24分 47秒 JST
% cat end-preprocessFTSE.txt
2020年 11月 9日 月曜日 18時 24分 50秒 JST
```

この結果から，処理時間は 3 秒であることがわかる[11]．

Ⅳ　OsirisC2020 データと FTSE2020 データの結合

本節では，II 節と III 節で前処理された，OsirisC2020 データ（firmfinC2020.parquet）と FTSE2020 データ（SPT2020.csv）の結合を行う．これらの CSV ファイルに関する情報を R を用いて企業毎に結合する処理を行う．そのためには，各種のパッケージを読み込んでおく必要がある：

パッケージの読み込み

```
> library(arrow)
> library(readr)
> library(dplyr)
> library(tidyr)
```

この設定のもとで，カレントディレクトリのサブディレクトリ ./data/OsirisC2020/ と ./data/FTSE2020/ にそれぞれのデータファイルが保存されていると仮定する．まず，Parquet ファイル firmfinC2020.par-

11）MacBook Pro 2018（OS: macOS Catalina（バージョン10.15.6），CPU: Intel(R) Core (TM) i9，2.9 GHz）で計測した．

quet を arrow パッケージに付属する read_parquet 関数を利用して，以下のようにデータを読み込み，さらに年別オブジェクトを作成する（図 6 も参照）：

Parquet ファイル firmfinC2020.parquet の R への読み込みと年別オブジェクトの作成

```
> firmfinC <- read_parquet("./data/OsirisC2020/firmfinC2020.parquet")
> firmfinC2013 <- firmfinC %>% filter(year == 2013)
> firmfinC2014 <- firmfinC %>% filter(year == 2014)
> firmfinC2015 <- firmfinC %>% filter(year == 2015)
> firmfinC2016 <- firmfinC %>% filter(year == 2016)
> firmfinC2017 <- firmfinC %>% filter(year == 2017)
> firmfinC2018 <- firmfinC %>% filter(year == 2018)
```

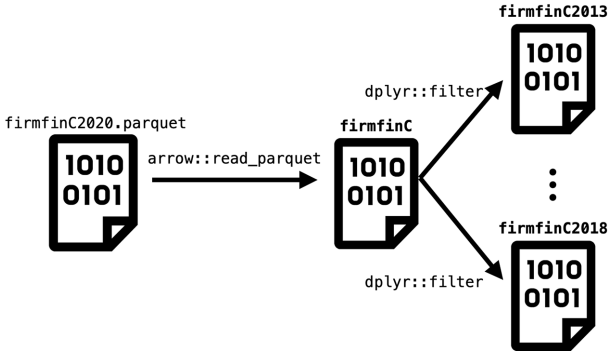

図 6 ：Parquet ファイル firmfinC2020.parquet の R への読み込みと年別オブジェクトの作成

ここで，Parquet ファイル firmfinC2020.parquet を読み込むために，read_parquet 関数を利用した理由については，付録 C を参照されたい．

続いて，CSV ファイル SPT2020.csv を readr パッケージに付属する read_csv 関数を利用して，以下のようにデータを読み込む：

CSV ファイル SPT2020.csv の R への読み込み

```
> SPT <- read_csv("./data/FTSE2020/SPT2020.csv")
```

次に，新興国（E）と先進国（G）に分けて，年別オブジェクトを作成する：

新興国（E）の年別オブジェクト作成

```
> # E
> dfESPT2015 <- SPT %>% filter(year == 2015, type == "E") %>%
+          mutate(year_SPT = year) %>% select(-year)
> dfESPT2016 <- SPT %>% filter(year == 2016, type == "E") %>%
+          mutate(year_SPT = year) %>% select(-year)
> dfESPT2017 <- SPT %>% filter(year == 2017, type == "E") %>%
+          mutate(year_SPT = year) %>% select(-year)
> dfESPT2018 <- SPT %>% filter(year == 2018, type == "E") %>%
+          mutate(year_SPT = year) %>% select(-year)
> dfESPT2019 <- SPT %>% filter(year == 2019, type == "E") %>%
+          mutate(year_SPT = year) %>% select(-year)
> dfESPT2020 <- SPT %>% filter(year == 2020, type == "E") %>%
+          mutate(year_SPT = year) %>% select(-year)
```

先進国（G）の年別オブジェクト作成

```
> # G
> dfGSPT2015 <- SPT %>% filter(year == 2015, type == "G") %>%
+          mutate(year_SPT = year) %>% select(-year)
> dfGSPT2016 <- SPT %>% filter(year == 2016, type == "G") %>%
+          mutate(year_SPT = year) %>% select(-year)
> dfGSPT2017 <- SPT %>% filter(year == 2017, type == "G") %>%
+          mutate(year_SPT = year) %>% select(-year)
> dfGSPT2018 <- SPT %>% filter(year == 2018, type == "G") %>%
+          mutate(year_SPT = year) %>% select(-year)
> dfGSPT2019 <- SPT %>% filter(year == 2019, type == "G") %>%
+          mutate(year_SPT = year) %>% select(-year)
> dfGSPT2020 <- SPT %>% filter(year == 2020, type == "G") %>%
+          mutate(year_SPT = year) %>% select(-year)
```

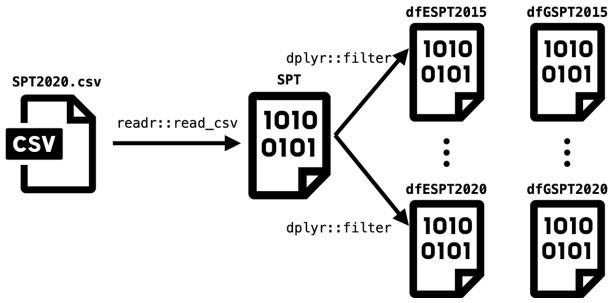

図7：CSV ファイル **SPT2020.csv** の R への読み込みと年別オブジェクトの作成

以上の準備のもとで，OsirisC2020データとFTSE2020データの結合を行う．本研究では，結合の際に，FTSE2020データについては，企業の事業年度に係る情報がアニュアルレポート等で開示され，FTSEの初期調査，企業からのフィードバック，最終調査結果を経て，ESGレーティングが算出されるまでに約2年を要するため，OsirisC2020データとのラグを2年として結合することにした．実際に結合するためには，両方のデータに共通の変数（主キー）が必要であり，今回はISINコード[12]を利用し，dplyrパッケージに付属する関数`inner_join`を使って以下のように結合した．例えば，OsirisC2020データ（2013年）とFTSE2020データ（2015年）を結合するためには以下のように入力する：

OsirisC2020データ（2013年）とFTSE2020データ（2015年）の結合

```
> #2015 Emerging
> E2015 <- firmfinC2013 %>% inner_join(dfESPT2015, by = c("ISIN"="ISIN"))
> # 2015 Global
> G2015 <- firmfinC2013 %>% inner_join(dfGSPT2015, by = c("ISIN"="ISIN"))
```

同様に結合することによって，その他の年に関しても同様のオブジェクトを作成した（図8も参照）．

さらに，新興国（E）と先進国（G）のそれぞれについて年に関するオブジェクトの結合をおこなう（図9参照）：

12）ISINコードとは，国際証券識別番号（ISIN: About International Securities Identification Number）のことであり，国際証券コード仕様ISO 6166で定められている全世界共通の12桁のコードを指す．（https://www.isin.org/ja/isin/）

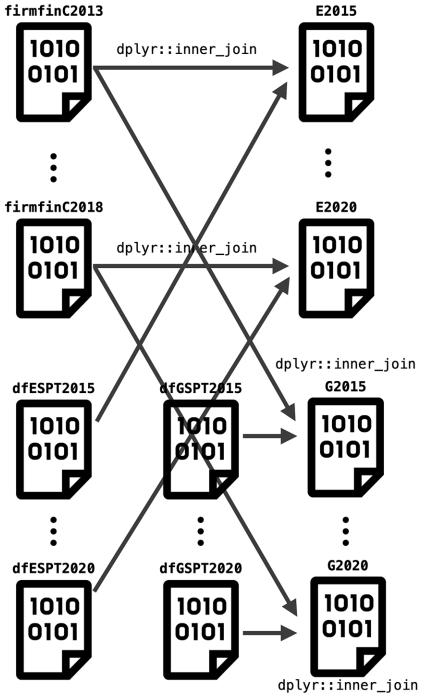

図 8 ：OsirisC2020 データと FTSE2020 データの結合

新興国（E）と先進国（G）の年に関するオブジェクトの結合

```
> # E
> Eall <- dplyr::bind_rows(E2015, E2016, E2017, E2018, E2019, E2020) %>%
+         dplyr::arrange(firmID, year)
> finSPT.E <- Eall[,c(91, 92, 4, 10, 26, 27, 44, 48, 52, 53, 54, 56, 84, 89, seq(93,117))]
> # G
> Gall <- dplyr::bind_rows(G2015, G2016, G2017, G2018, G2019, G2020) %>%
+         dplyr::arrange(firmID, year)
> finSPT.G <- Gall[,c(91, 92, 4, 10, 26, 27, 44, 48, 52, 53, 54, 56, 84, 89, seq(93,117))]
```

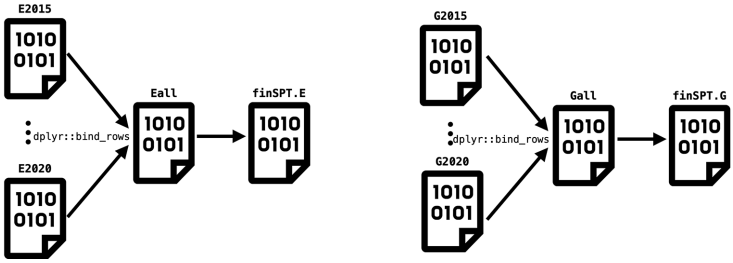

図9：新興国（E）と先進国（G）の年に関するオブジェクトの結合

以上の処理によって，新興国（E）と先進国（G）のそれぞれについて結合されたオブジェクトができたので，最後にこれらのオブジェクトを結合する：

新興国（E）と先進国（G）のオブジェクトの結合

```
> finSPT <- bind_rows(finSPT.E, finSPT.G)
```

結合されたオブジェクトfinSPTは，各行に企業の財務データとESGレーティングデータを持つ16236行，39列のデータフレームである（図10も参照）：

新興国（E）と先進国（G）の結合オブジェクトfinSPT

```
> head(finSPT)
# A tibble: 6 x 39
  GICS_code GICS_name country.x month assets_total shareholders  sales   EBIT
      <int> <chr>     <chr>     <int>         <dbl>        <dbl>  <dbl>  <dbl>
1  20106020 INDUSTRI… CN           12       3232632      2505586 1.84e6 544393
2  20106020 INDUSTRI… CN           12       4282545      3542901 1.88e6 501933
3  20105010 INDUSTRI… IN           12        355235       163999 4.24e5  73758
4  20105010 INDUSTRI… IN           12        306243       207143 4.37e5  76007
5  35201010 BIOTECHN… KY           12        376931       154207 1.85e5  60113
6  35201010 BIOTECHN… KY           12       1021402       868130 2.58e5  71464
# … with 31 more variables: PL_before_tax <dbl>, tax <dbl>, PL_after_tax <dbl>,
#   net_income <dbl>, market_cap <dbl>, ISIN <chr>, firmID <chr>, year <int>,
#   firm.y <chr>, country.y <chr>, industry <dbl>, ESG <dbl>, E <dbl>, S <dbl>, G <dbl>,
#   EPR <dbl>, ECC <dbl>, EWU <dbl>, EBD <dbl>, ESC <dbl>, SCR <dbl>, SLS <dbl>,
#   SHS <dbl>, SHR <dbl>, SSC <dbl>, GCG <dbl>, GRM <dbl>, GTX <dbl>, GAC <dbl>,
#   type <chr>, year_SPT <dbl>
```

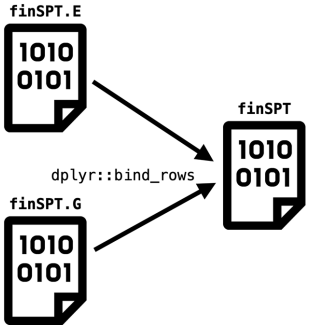

図10：新興国（E）と先進国（G）の結合オブジェクト

OsirisC2020 データと FTSE2020 データの結合の全工程を図11にまとめる．

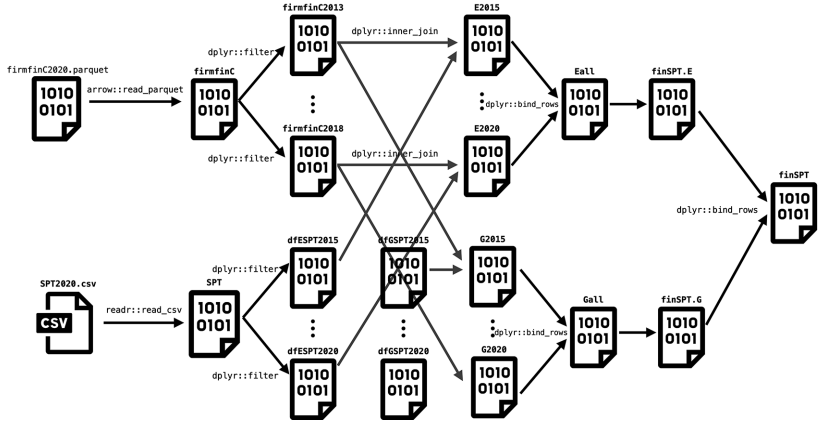

図11：OsirisC2020 データと FTSE2020 データの結合の全工程

最後に，このデータフレームを，readr パッケージの関数 `write_csv` を使って CSV ファイル `fin-SPT2020.csv` へ出力する．

データフレーム `finSPT` を CSV ファイル `finSPT2020.csv` へ出力

```
> write_csv(finSPT, "finSPT2020.csv")
```

以上のデータの結合と CSV ファイルへの出力を実行するためスクリプト

は，Makefile（コード11）に記述しておき，make コマンドを実行することによって自動的に処理される．なお，OsirisC2020 データと FTSE2020 データの結合のフローと Makefile ファイルから呼び出されるスクリプトのフローを図12に与える．

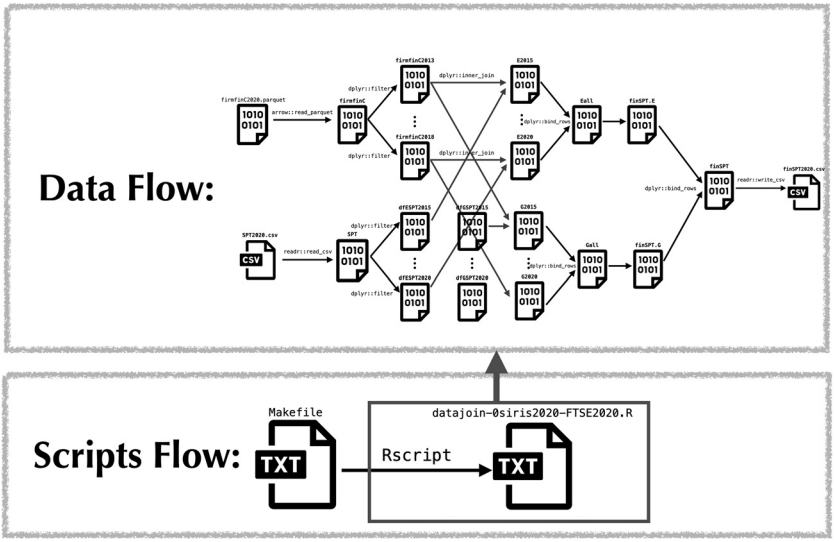

図12：OsirisC2020 データと FTSE2020 データの結合のフローとスクリプトのフロー

コード11：ファイル **Makefile**（ターゲット：**join**）

```
1 join:
2 	date > start-join.txt
3 	Rscript datajoin-0sirisC2020-FTSE2020.R
4 	date > end-join.txt
```

コード11の 1 行目にはターゲット名 join が与えられており，2 行目で処理の開始時間がテキストファイル start-join.txt に書き出されている．これに対をなす 4 行目で処理の終了時間がテキストファイル end-joinE.txt に記録され，両方のファイルを記録することによって，処理時間を計測することができる．3 行目で指定されている Rscript コマンドの引数で

呼び出されているRスクリプトdatajoin-0sirisC2020-FTSE2020.R（コード12）を使って行われる処理は，図12におけるデータフロー全体に対応している．

コード12：Rスクリプトファイル datajoin-0sirisC2020-FTSE2020.R（一部）

```
library(arrow)
library(readr)
library(dplyr)
library(tidyr)

# Osiris2020 データロード
firmfinC <- read_parquet("./OsirisC2020/firmfinC2020.parquet")
firmfinC2013 <- firmfinC %>% filter(year == 2013)
: (中略)
firmfinC2018 <- firmfinC %>% filter(year == 2018)

# FTSE データロード
SPT <- read_csv("./FTSE2020/SPT2020.csv")

dfESPT2015 <- SPT %>% filter(year == 2015, type == "E") %>% mutate(year_SPT
    = year) %>% select(-year)
: (中略)
dfESPT2020 <- SPT %>% filter(year == 2020, type == "E") %>% mutate(year_SPT
    = year) %>% select(-year)

dfGSPT2015 <- SPT %>% filter(year == 2015, type == "G") %>% mutate(year_SPT
    = year) %>% select(-year)
: (中略)
dfGSPT2020 <- SPT %>% filter(year == 2020, type == "G") %>% mutate(year_SPT
    = year) %>% select(-year)

#-------------------------------------------------------------------------
# 2015 Emerging
E2015 <- firmfinC2013 %>% inner_join(dfESPT2015, by = c("ISIN"="ISIN"))
# 2015 Global
G2015 <- firmfinC2013 %>% inner_join(dfGSPT2015, by = c("ISIN"="ISIN"))
: (中略)
#-------------------------------------------------------------------------
# 2020 Emerging
E2020 <- firmfinC2018 %>% inner_join(dfESPT2020, by = c("ISIN"="ISIN"))
# 2020 Global
G2020 <- firmfinC2018 %>% inner_join(dfGSPT2020, by = c("ISIN"="ISIN"))
#-------------------------------------------------------------------------


# Data Merge, Bind, and Dump
# E
Eall <- dplyr::bind_rows(E2015, E2016, E2017, E2018, E2019, E2020) %>%
   dplyr::arrange(firmID, year)
finSPT.E <- Eall[,c(91, 92, 4, 10, 26, 27, 44, 48, 52, 53, 54, 56, 84, 89,
   seq(93,117))]
```

```
41
42 # G
43 Gall <- dplyr::bind_rows(G2015, G2016, G2017, G2018, G2019, G2020) %>%
       dplyr::arrange(firmID, year)
44 finSPT.G <- Gall[,c(91, 92, 4, 10, 26, 27, 44, 48, 52, 53, 54, 56, 84, 89,
       seq(93,117))]
45
46 # Bind E and G
47 finSPT <- bind_rows(finSPT.E, finSPT.G)
48
49 # save
50 write_csv(finSPT, "finSPT2020.csv")
```

以上の設定のもとで，データの結合とCSVファイルへの出力を行うためには，makeコマンドを（カレント）ディレクトリdata（図14を参照）で以下のように実行することによって自動的に処理される．

makeコマンド実行：ターゲットjoin

```
% make join
```

この実行時間はmakeコマンドの実行時に出力されるstart-join.txtとend-join.txtを比較することによって以下のようにわかる：

ターゲットjoinに対するmakeコマンド実行時間

```
% cat start-join.txt
2020年 11月 8日 日曜日 09時 44分 03秒 JST
% cat end-join.txt
2020年 11月 8日 日曜日 09時 44分 11秒 JST
```

この結果から，処理時間は8秒であることがわかる[13]．

V　全工程の自動実行

本稿を通じて行ってきた前処理と結合全体を自動実行することを考える．全工程を実行するためスクリプトは，Makefile（コード13）に記述してお

13）MacBook Pro 2018（OS: macOS Catalina（バージョン10.15.6），CPU: Intel(R) Core (TM) i9，2.9GHz）で計測した．ただし，arrowパッケージのバージョンを2.0.0にした場合は，処理時間が45秒になっている．これは，read_parquet関数の読み込み時間が遅くなったことに起因するものと考えられる（付録Cの脚注も参照されたい）．

き，make コマンドを実行することによって自動的に処理される．全工程のデータのフローとMakefile ファイルから呼び出されるスクリプトのフローを図13に与える．

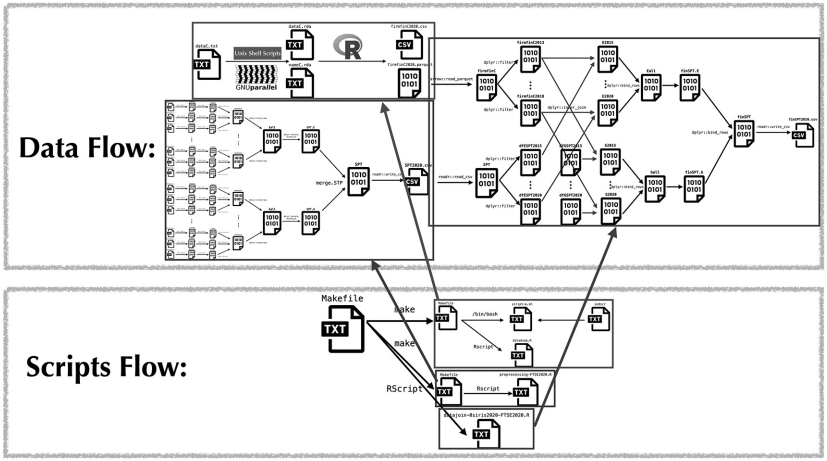

図13：全工程のフロー

コード13：ファイル **Makefile**（ターゲット：**all**）

```
1 all:
2 	date > start-all.txt
3 	(cd OsirisC2020; make)
4 	(cd FTSE2020; make)
5 	Rscript datajoin-OsirisC2020-FTSE2020.R
6 	date > end-all.txt
```

コード13の 1 行目にはターゲット名 all が与えられており，2 行目で処理の開始時間がテキストファイル start-all.txt に書き出されている．これに対をなす 6 行目で処理の終了時間がテキストファイル end-all.txt に記録され，両方のファイルを記録することによって，処理時間を計測することができる．3 行目で指定されているシェルスクリプト（cd OsirisC2020; make）は，II 節で述べたデータセット DS-Osiris-C-2020 の前処理を実行するためのものである（図13におけるデータフローの左上の部分参照）．また，4 行目で指定されているシェルスクリプト（cd FTSE2020; make）

は，III節で述べたデータセットDS-FTSE-Russell-2020の前処理を実行するためのものである（図13におけるデータフローの左下の部分参照）．さらに，5行目で指定されているシェルスクリプトは，IV節で扱ったOsirisC2020データとFTSE2020データの結合を実行するためのものである（図13におけるデータフローの右側の部分参照）．

以上の設定のもとで，makeコマンドを（カレント）ディレクトリdata（図14を参照）において以下のように実行することによって自動的に処理される．

makeコマンド実行：ターゲットall

```
% make all
```

この実行時間はmakeコマンドの実行時に出力されるstart-all.txtとend-all.txtを比較することによって以下のようにわかる：

ターゲットallに対するmakeコマンド実行時間

```
% cat start-all.txt
2020年 11月 7日 土曜日 16時 02分 45秒 JST
% cat end-all.txt
2020年 11月 7日 土曜日 16時 05分 52秒 JST
```

この結果から，処理時間は3分7秒であることがわかる[14)]．

VI　おわりに

本稿では，財務データとESGレーティングデータの前処理と結合について述べた．これらの全工程をmakeコマンドのみで自動実行する仕様としたが，処理の再現性を確保するためには有効であると考えられる．また，これ

14）MacBook Pro 2018（OS: macOS Catalina（バージョン10.15.6），CPU: Intel(R) Core (TM) i9，2.9GHz）で計測した．ただし，arrowパッケージのバージョンを2.0.0にした場合は，処理時間が4分15秒になっている．これは，write_parquet関数の書き出しとread_parquet関数の読み込み時間が遅くなったことに起因するものと考えられる（付録Cの脚注も参照されたい）．

らの処理には，dplyr，tidyr 等の RStudio 社が開発しているパッケージを利用したが，同社が開発している purrr というパッケージがあり，本稿で扱っている処理のためのコーディングを簡略化できるものと思われる．この点については今後の課題としたい．また，本稿で得られたデータを使った探索的データ解析については，別の機会で詳細を述べる予定である．

（筆者（地道）は関西学院大学商学部教授）
（筆者（阪）は関西学院大学商学部教授）

参考文献

［1］ FTSE Russell（2019）*ESG Data Model 6th Research Cycle, Methodology*（*2019/20*），https://qsd.ftserussell.com/Docs/ESG/FTSE%20Russell%20ESG%20Data%20Model%20Methodology%20-%20April-2019-March-2020-Rev2.pdf
［2］ 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.）
［3］ 地道正行（2018-a）『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－』，商学論究，第66巻，第 1 号，pp. 1-32，関西学院大学商学研究会.
［4］ 地道正行（2018-b）『探索的財務ビッグデータ解析－データ可視化，統計モデリング，モデル選択，モデル評価，動的文書生成，再現可能研究－』，商学論究，第66巻，第 2 号，pp. 1-41，関西学院大学商学研究会.
［5］ 地道正行（2018-c）『データサイエンスの基礎：R による統計学独習』，裳華房.
［6］ 地道正行（2020-a）『探索的財務ビッグデータ解析－前処理の並列化－』，商学論究，第67巻，第 3 号，pp. 1-19，関西学院大学商学研究会.
［7］ 地道正行（2020-b）『探索的財務ビッグデータ解析－PG-Strom によるデータラングリングの並列化－』，商学論究，第68巻，第 1 号，pp. 1-34，関西学院大学商学研究会.
［8］ Tange, Ole,（2018）*GNU Parallel 2018,* ISBN: 9781387509881, DOI: 10.5281/zenodo.1146014, URL: https://doi.org/10.5281/zenodo.1146014, Mar, 2018.
［9］ Wickham, H. and G. Grolemund（2016）*R for Data Science,* O'Reilly.

謝辞

本研究の一部は以下の助成を得ている．
科研費 KAKENHI 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～2019年），課題番号：16K04022

科研費 KAKENHI 科学研究費基盤研究 C：「共有価値創造（CSV）のための社会環境会計の構築」（2019年～2021年），課題番号：19K02006

2019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：jh191002-NWJ

2020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：jh201003-NWJ

関西学院大学 図書館 図書費 B，研究設備費（III），個人研究費

また，BvD 社 増田 歩氏には様々なご足労を賜った．ここに感謝の意を表する．

付録 A　データに関するディレクトリ・ファイル構成

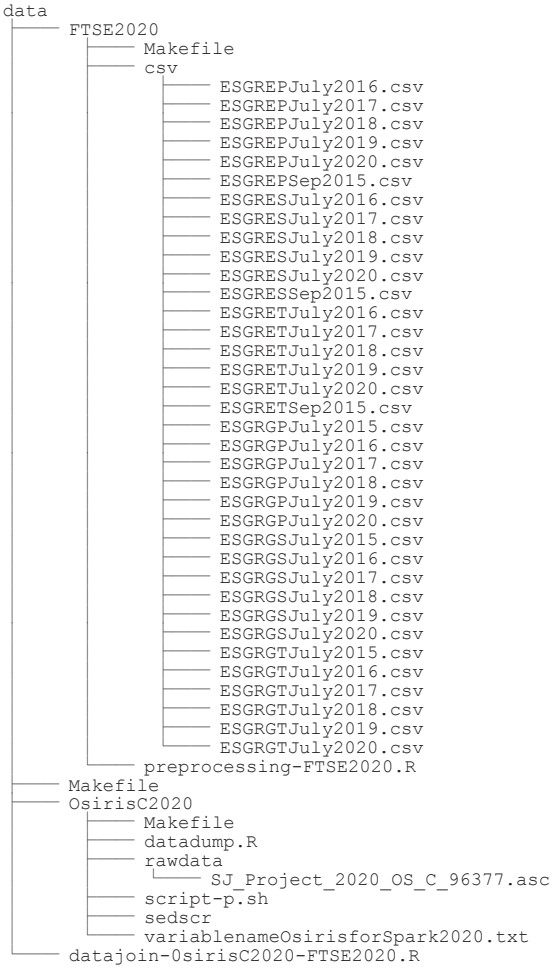

図14：データを前処理・結合を行うためのディレクトリとファイルの構成

付録B　Rに関する環境

本稿を執筆するために利用したRに関する情報を与える．

sessionInfoの実行結果

```
> sessionInfo()
R version 4.0.3 (2020-10-10)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Catalina 10.15.7

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.0/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] arrow_2.0.0 tidyr_1.1.2 dplyr_1.0.2 readr_1.4.0

loaded via a namespace (and not attached):
 [1] rstudioapi_0.11  magrittr_1.5     hms_0.5.3        tidyselect_1.1.0 bit_4.0.4
 [6] R6_2.5.0         rlang_0.4.8      fansi_0.4.1      tools_4.0.3      utf8_1.1.4
[11] cli_2.1.0        ellipsis_0.3.1   bit64_4.0.5      assertthat_0.2.1 tibble_3.0.4
[16] lifecycle_0.2.0  crayon_1.3.4     purrr_0.3.4      vctrs_0.3.4      ps_1.4.0
[21] glue_1.4.2       compiler_4.0.3   pillar_1.4.6     generics_0.1.0   pkgconfig_2.0.3
```

ただし，arrowパッケージの`read_parquet`関数を利用した一部の結果は，バージョンが1.0.0の環境で実行したものも含まれることに注意が必要である．

付録C　CSVファイルとParquetファイルのRへの読み込み

II節で前処理の結果として出力されたOsirisC2020データのCSVファイル`firmfinC2020.csv`とParquetファイル`firmfinC2020.parquet`は以下のようなサイズである：

CSVファイル`firmfinC2020.csv`とParquetファイル`firmfinC2020.parquet`のサイズ

```
% ls -la firmfinC2020.*
-rw-r--r--@ 1 masa  staff  1516587644  8  6 12:25 firmfinC2020.csv
-rw-r--r--  1 masa  staff   229688233  4 16  2020 firmfinC2020.parquet
```

この結果から，CSV ファイル firmfinC2020.csv が約 1.517 GB であるのに対して，Parquet ファイル firmfinC2020.parquet のサイズは約 230 MB とほぼ 1/6 以下に圧縮されている．このことから，R へのロードにかかる時間を大幅に短縮できる可能性がある．実際に，前処理に利用したコンピュータ環境で，readr パッケージの関数 read_csv と arrow パッケージの関数 read_parquet を利用し，system.time 関数を使って計測した結果を以下に与える：

ファイルのロード時間の計測

```
> library(readr)
> library(arrow)
> system.time(read_csv("firmfinC2020.csv"))
#|=============================================================| 100% 1446 MB
#ユーザ    システム       経過
#25.484      2.353     31.264
> system.time(read_parquet("firmfinC2020.parquet"))
#ユーザ    システム       経過
#9.549       2.793      3.538
```

1 回だけの実行であり，他のプロセスなどの関係もあるので，あくまでも参考程度であるが，ロードに関する時間がほぼ 1/10に短縮されていることがわかる．以上の結果から，特別な理由が無い限り，このデータを R へロードする際は，Parquet ファイルを利用することが推奨される[15]．

15）本稿執筆期間中に arrow パッケージのバージョンを2.0.0にアップデートしたところ，ここで与えたロード時間よりもパフォーマンスが下がったことに注意が必要である．今後の改良が望まれる．