探索的財務ビッグデータ解析と再現可能研究
—mdx 環境とローカル環境の協調による非上場企業データのラングリングと可視化の自動化—

地　道　正　行
阪　　智　香

要　旨

本稿では，Bureau van Dijk 社のデータベース Orbis から抽出されたデータを，mdx 環境とローカル環境を協調することによって，世界の非上場企業に関する規模の大きな財務（諸表）データのラングリングと可視化を行う．なお，本研究において行われる処理は自動的に実行され，また本稿も動的文書生成によって作成されており，この意味で再現可能研究の立場から行われている．

キーワード：財務ビッグデータ（Financial Big Data），データラングリング（Data Wrangling），探索的データ解析（Exploratory Data Analysis），動的文書生成（Dynamic Documents），再現可能研究（Reproducible Research）

I　はじめに

筆者の研究グループは，Bureau van Dijk[1]（ビューロー・ヴァン・ダイク）社のデータベース Orbis から抽出された世界の「非上場企業」（unlisted firms and delisted firms）に対する2018会計年度の連結主体で抽出した財務（諸表）データと単体主体で抽出した財務（諸表）データをラングリングしたものを結合することについて検討した（cf. 地道（2020-a, b），地道，阪（2022），

1）　Bureau van Dijk Web Page: https://www.bvdinfo.com/en-gb/

地道ら（2022-a）．さらに，地道ら（2022-c）では，会計年度を2009年から2018年度の10年間に拡大し，データラングリングを実行したものを可視化することを試みている．本稿では，mdx 環境[2] とローカル環境を協調することによって上記のデータを前処理，ラングリング，可視化することについて議論する．この工程は，規模の大きな財務データに対して探索的データ解析（Tukey（1977））を実行することを意味していることに留意しよう．

本稿の構成は以下のようなものである．まず，利用するデータベースとデータセットに関する情報を与えたもとで，mdx 環境におけるデータ前処理の概要を述べ（II 節），データの属性を与えるとともに mdx 環境におけるラングリングを実行し，データを抽出することについて述べる（III 節）．さらに，抽出されたデータをローカル環境に転送後，データの変換を含むラングリングに関する工程を詳述し（IV 節），最終的に得られたデータの可視化を自動化するための方法を与える（V 節）．最後に，今後の課題などを与える（VI 節）．

付録 A には，本研究で利用しているコンピュータ環境に関する情報を与えており，付録 B に本稿で利用したデータ可視化のための R スクリプトを与えている．なお，本研究で行われる前処理，ラングリング，可視化の全工程は，データ解析環境 R[3] と `make` コマンド（cf. Mecklenburg（2015））による自動実行によって行われており，本稿も Sweave（cf. Leisch（2002））による動的文書生成（dynamic documents）によって作成し，再現可能性[4] を確保している（付録 C も参照のこと）．

2） mdx（データ活用社会創成プラットフォーム）は，研究環境を用途に合わせてオンデマンドで短時間に構築・拡張・融合できる，データ収集・集積・解析のためのプラットフォームである（`https://mdx.jp` より引用）．mdx 環境についての詳細については，Suzumura *et al.*（2022）を参照されたい．

3） `https://www.r-project.org`

4） R を利用した動的文書生成については，例えば，Xie（2015）を，また，再現可能研究に関しては，Peng（2011），Gandrud（2020）を参照されたい．

Ⅱ　データベースとデータセット

本稿で扱うデータやその処理については，地道，阪（2022），地道ら（2022-a, c）で説明されているが，今回の研究では，新たに mdx 環境において構築したシステムを利用しているため，簡単な解説を与える（地道（2020-a），地道ら（2020-a, b）も参照のこと）.

まず，データベース Orbis（2019年12月版）[5] には，連結（Consolidated）と単体（Unconsolidated）の両方の財務諸表からのデータが収録されている．今回は，以下のような企業について，主要財務情報を最長10年分抽出したデータを利用している：

（1）連結財務諸表を優先的に抽出した26,353,934社（以後，「連結ベース」と略す）

（2）単体財務諸表を優先的に抽出した26,352,382社（以後，「単体ベース」と略す）

地道ら（2020-a, b）では，2020年に抽出された Orbis データセット DS-Orbis-C-2019（連結ベース），DS-Orbis-U-2019（非連結ベース）を，地道（2020-a）で議論された GNU parallel[6]（cf. Tange（2022））を用いて並列化する方法で処理し，表１で与えられる CSV ファイルを得ている．今回利用しているデータは，mdx 環境を利用して地道（2020-a）と同様の方法で前処理を行うことによって得られた CSV ファイルを利用している．

表１：前処理後のデータセット：サイズ

データセット名	社数	行数	列数	CSV ファイル	サイズ
DS-Orbis-C-2019	26,353,934	263,539,341	88	`firmfinBC2019.csv`	約 140.5 GB
DS-Orbis-U-2019	26,352,382	263,523,821	88	`firmfinBU2019.csv`	約 140.5 GB

データの前処理と `Makefile` の処理命令との対応については図１[7] を参

5）本稿で利用している2019年12月版 Orbis には，約３億社以上が収録されている．

6）`https://www.gnu.org/software/parallel/`

7）ここでは，連結決算データの場合を与えているが，単体の場合も同様の処理によって行われる．

照されたい（前処理の詳細については，地道（2020-a）を参照されたい）．

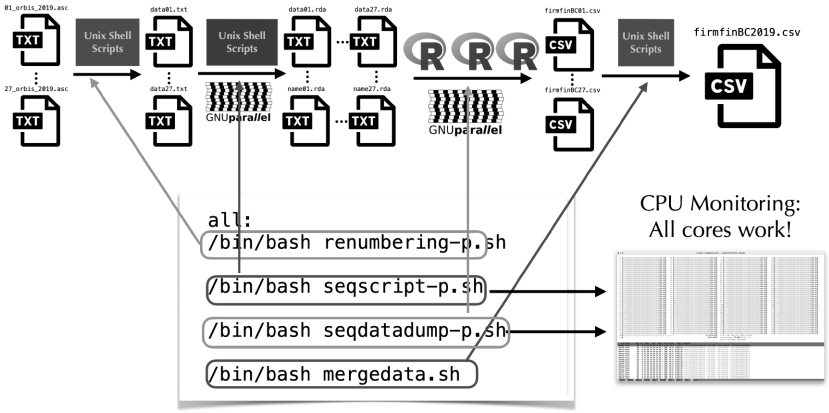

図 1 ：mdx 環境における Orbis データの前処理：連結決算データの場合

なお，今回利用した mdx 環境における処理時間の計測結果は，以下のようなものである：

mdx 環境における処理時間の計測結果

```
# mdx (fast)
masa@mdx-kg-05: $ cat starttime-parallel.txt
Wed 23 Nov 2022 01:25:37 PM JST
masa@mdx-kg-05:$ cat endtime-parallel.txt
Wed 23 Nov 2022 02:26:28 PM JST
```

この結果から，所要時間は約 1 時間であることがわかる．地道ら（2020-a, b）の結果では，この工程は，ローカル環境[8] で 1 時間20分程度かかっていたので，約20分短縮されたことがわかる．この理由の一つとしては，今回利用している mdx 環境の CPU ノードのコア数が152であるため，GNU parallel による並列処理が高速化したことによる．なお，mdx 環境の仕様については，付録 A を参照されたい．

8） Machine Name: Dell Precision T7910, CPU: Intel Xeon プロセッサー E5-2687W v4（コア数；48），Memory: 128 GB, Strage: SSD 4 TB, OS: Ubuntu 18.04

Ⅲ　データの属性と抽出

本節では，データの属性について述べた後，その抽出について詳細に議論する．

1. 属性

本研究で利用するデータ属性（企業情報・会計情報）は，地道，阪（2022）でも説明されているが，今回の研究では，新たな指標（支払利息）も追加しているため再掲する：

firmid: 企業名 + BvD ID（例: "ITOHAM YONEKYU HOLDINGS INC. JP6011001110293"; 伊藤ハム米久ホールディングス +BvD ID）

id: BvD ID（例: "JP6011001110293"; 伊藤ハム米久ホールディングスの BvD ID）

year: 会計年度（2009, ... , 2018）

country: 国別情報（例: "Japan"; 日本）

listed: 上場情報（"Delisted": 上場廃止, "Listed": 上場, "Unlisted": 非上場）

cons: 連結コード（"C1", "C2", "U1", "U2"）

exchange: 主取引所（例: "New York Stock Exchange (NYSE)"; ニューヨーク証券取引所）

date: 決算年月日

month: 決算月数

practice: 会計基準（"IFRS": 国際会計基準, "Local GAAP": 国内基準）

infoprov: インフォメーションプロバイダ

assets_fix: 固定資産（単位: 1,000 US ドル）

assets_cur: 流動資産（単位: 1,000 US ドル）

assets_total: 資産合計（単位: 1,000 US ドル）

shareholders：株主資本（単位：1,000 US ドル）
debt_long：固定負債（単位：1,000 US ドル）
liabilities_cur：流動負債（単位：1,000 US ドル）
employees：従業員数（単位：人）
operating_revenue：営業収益（単位：1,000 US ドル）
ebit：営業利益（単位：1,000 US ドル）
pl_before_tax：税引前利益（単位：1,000 US ドル）
tax：税金（単位：1,000 US ドル）
net_income：純利益（単位：1,000 US ドル）
costs_employees：人件費（単位：1,000 US ドル）
interest_paid：支払利息（単位：1,000 US ドル）

ここで，国別情報は，企業の本社が存在する国を表しており，連結コード cons の種類としては以下のようなものがある：

C1：連結財務諸表のみを保有している企業
C2：連結財務諸表を保有しており，何らかの理由で単体財務諸表も保有している企業
U1：単体財務諸表のみを保有している企業
U2：単体財務諸表を保有しており，何らかの理由で連結財務諸表も保有している企業

2. 抽出

mdx 環境において抽出に利用したスクリプトファイルを格納したディレクトリ構成を図 2 に与える．

```
DW-mdx-Orbis2019
├── DW-Orbis2019c-fennel-kg-01.R
├── DW-Orbis2019u-fennel-kg-01.R
└── Makefile
```

図 2 ：mdx 環境用スクリプトファイルのディレクトリ構成

mdx 環境における抽出元になる PostgreSQL のデータベース jhpcn には，

テーブル orbis2019c（連結ベース）と orbis2019u（単体ベース）が用意されている[9]．以下に，連結ベースと単体ベースのデータの抽出について，それぞれの場合に分けて述べる．

連結ベースのデータ抽出

連結ベースのデータを抽出するためには，ディレクトリ DW-mdx-Orbis 2019 の Makefile におけるターゲット DW-c を利用する（図 2，及び，スクリプト 1 参照）．

スクリプト 1 ：Makfile：ターゲット DW-c

```
DW -c:
	date > start-DW-c-mdx-kg-02.txt
	Rscript DW-Orbis2019c-mdx-kg-02.R
	date > end-DW-c-mdx-kg-02.txt
```

ここで，スクリプト 1 における，(2, 4) 行目は処理時間を計測するための指定である．また，3 行目で Rscript コマンドによって，R スクリプトファイル DW-Orbis2019c-mdx-kg-02.R（スクリプト 2 参照）が実行されている．

スクリプト 2 ：DW-Orbis2019c-mdx-kg-02.R

```
library(RPostgreSQL)
library(arrow)
drv <- dbDriver("PostgreSQL")
con <- dbConnect(drv, host="mdx-kg-02", user= "*****", password="*****",
   dbname="jhpcn")
# Data Wrangling from orbis2019c
for(i in 2009:2018)
{
sql <- paste0("select␣firmid,␣id,␣year,␣country,␣listed,␣cons,␣exchange,␣
   date,␣month,␣practice,␣infoprov,␣assets_fix,␣assets_cur,␣assets_total,␣
   shareholders,␣debt_long,␣liabilities_cur,␣employees,␣operating_revenue,␣
   ebit,␣pl_before_tax,␣tax,␣net_income,␣costs_employees,␣interest_paid␣
   FROM␣orbis2019c␣WHERE␣(cons␣=␣'C1'␣OR␣cons␣=␣'C2')␣AND␣year␣=␣",i)
x <- fetch(dbSendQuery(con, sql), n = -1)
# dump parquet
write_parquet(x, sink = paste0("DWOrbis2019c-",i,".parquet"))
rm(sql, x)
}
```

9) データファイル firmfinBC2019.csv，firmfinBU2019.csv のデータベース化は，東京大学の宮本大輔氏の協力を仰いだ．

スクリプト 2 によって行われる処理は以下のようなものである：

（DWc1）データベース `jhpcn` とのコネクション（1〜4 行目）

（DWc2）連結ベースのデータテーブル `orbis2019c` に対して SQL 問合せ（8 行目）を行い，2009年から2018年のデータを順次抽出したものを Parquet[10] ファイル `DW-mdx-Orbis2019c-2009.parquet` 〜 `DW-mdx-Orbis2019c-2018.parquet` として書き出す（6〜13 行目）

データ抽出にともなうスクリプトファイルと Orbis データファイルの流れとそれらの対応を可視化したものを図 3 に与える．

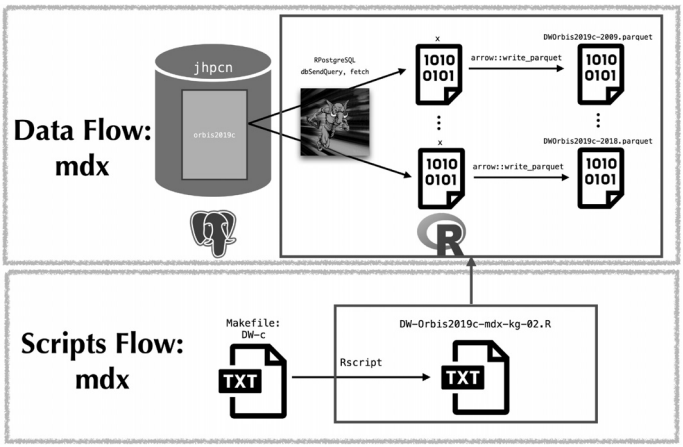

図 3 ：mdx 環境のもとでの Orbis データ（連結ベース）抽出にともなうスクリプトファイルとデータファイルの流れ

以上の処理は，mdx 環境におけるディレクトリ `DW-mdx-Orbis2019`（図 2 参照）をカレント（ディレクトリ）とし，ターミナル（コマンドライン）

10）ここで，データをファイルに書き出すために Parquet 形式を利用しているが，その理由としては，R とファイルを高速に読み書きできるためである（cf. 地道ら（2021-b））．なお，Apache Parquet については，`https://parquet.apache.org` を，また，Parquet 形式のファイルを R で扱うためのパッケージ **{arrow}** については，`https://arrow.apache.org/docs/r/` を参照されたい．

上で以下のように入力することによって実行できる[11]．

ターゲット DW-c の実行

```
$ make DW-c
```

この処理時間は，スクリプト 1 における，(2, 4) 行目で実行される結果を比較することによってわかる．

ターゲット DW-c の処理時間の計測

```
$ cat start-DW-c-mdx-kg-02.txt
Mon 24 Oct 2022 08:29:12 AM JST
$ cat end-DW-c-mdx-kg-02.txt
Mon 24 Oct 2022 12:47:15 PM JST
```

この結果から，約 4 時間18分であることがわかる[12]．なお，実際に出力されたファイルの情報は以下のように与えられる：

出力ファイル DW-mdx-Orbis2019c-20*.parquet に関する情報

```
% ls -la DW-mdx-Orbis2019c-20*.parquet
-rw-rw-r-- 1 masa masa  82533627 Oct 24 08:54 DW-mdx-Orbis2019c-2009.parquet
-rw-rw-r-- 1 masa masa  83472324 Oct 24 09:20 DW-mdx-Orbis2019c-2010.parquet
-rw-rw-r-- 1 masa masa  84177415 Oct 24 09:46 DW-mdx-Orbis2019c-2011.parquet
-rw-rw-r-- 1 masa masa  84941744 Oct 24 10:11 DW-mdx-Orbis2019c-2012.parquet
-rw-rw-r-- 1 masa masa  92566770 Oct 24 10:37 DW-mdx-Orbis2019c-2013.parquet
-rw-rw-r-- 1 masa masa  99713304 Oct 24 11:02 DW-mdx-Orbis2019c-2014.parquet
-rw-rw-r-- 1 masa masa 102263969 Oct 24 11:28 DW-mdx-Orbis2019c-2015.parquet
-rw-rw-r-- 1 masa masa 102676830 Oct 24 11:55 DW-mdx-Orbis2019c-2016.parquet
-rw-rw-r-- 1 masa masa 103534885 Oct 24 12:20 DW-mdx-Orbis2019c-2017.parquet
-rw-rw-r-- 1 masa masa  96581360 Oct 24 12:47 DW-mdx-Orbis2019c-2018.parquet
```

単体ベースのデータ抽出

単体ベースのデータを抽出するためには，ディレクトリ DW-mdx-Orbis2019 の Makefile におけるターゲット DW-u を利用する（図 2，及び，スクリプト 3 参照）．

11）連結ベースのデータ抽出は，GPU ノードで実行している．
12）この結果は，mdx における PG-Strom 環境のチューニングの余地があることを示している．

スクリプト 3：Makefile：ターゲット DW-u

```
1 DW-u:
2       date > start-DW-u-mdx-kg-03.txt
3       Rscript DW-Orbis2019u-mdx-kg-02.R
4       date > end-DW-u-mdx-kg-03.txt
```

ここで，スクリプト 3 における，(2, 4) 行目は処理時間を計測するための指定である．また，3 行目で Rscript コマンドによって，R スクリプトファイル DW-orbis2019u-mdx-kg-02.R（スクリプト 4 参照）が実行されている．

スクリプト 4：DW-orbis2019u-mdx-kg-02.R

```
1  library(RPostgreSQL)
2  library(arrow)
3  drv <- dbDriver("PostgreSQL")
4  con <- dbConnect(drv, host="mdx-kg-02", user= "*****", password="*****",
       dbname="jhpcn")
5  # Data Wrangling from orbis2019u
6  for(i in 2009:2018)
7  {
8  sql <- paste0("select␣firmid,␣id,␣year,␣country,␣listed,␣cons,␣exchange,␣
       date,␣month,␣practice,␣infoprov,␣assets_fix,␣assets_cur,␣assets_total,␣
       shareholders,␣debt_long,␣liabilities_cur,␣employees,␣operating_revenue,␣
       ebit,␣pl_before_tax,␣tax,␣net_income,␣costs_employees,␣interest_paid␣
       FROM␣orbis2019u␣WHERE␣(cons␣=␣'U1'␣OR␣cons␣=␣'U2')␣AND␣year␣=␣",i)
9  x <- fetch(dbSendQuery(con, sql), n = -1)
10 # dump parquet
11 write_parquet(x, sink = paste0("DWOrbis2019u-",i,".parquet"))
12 rm(sql, x)
13 }
```

スクリプト 4 によって行われる処理は以下のようなものである：

(DWu1) データベース jhpcn とのコネクション（1〜4 行目）

(DWu2) 連結ベースのデータテーブル orbis2019u に対して SQL 問合せ（8 行目）を行い，2009年から2018年のデータを順次抽出したものを Parquet ファイル DWorbis2019u-2009.parquet 〜 DWorbis2019u-2018.parquet として書き出す（6〜13行目）

データ抽出にともなうスクリプトファイルと Orbis データファイルの流れとそれらの対応を可視化したものを図 4 に与える．

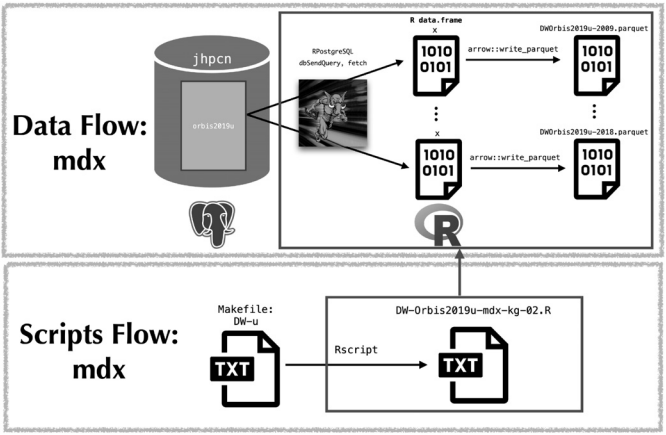

図 4 ：mdx 環境のもとでの Orbis データ（単体ベース）抽出にともなうスクリプトファイルとデータファイルの流れ

以上の処理は，mdx 環境におけるディレクトリ DW-mdx-Orbis2019（図 2 参照）をカレント（ディレクトリ）とし，ターミナル（コマンドライン）上で以下のように入力することによって実行できる[13]．

ターゲット DW-u の実行

```
$ make DW-u
```

この処理時間は，スクリプト 3 における，(2, 4) 行目で実行される結果を比較することによってわかる．

ターゲット DW-u の処理時間の計測

```
$ cat start-DW-u-mdx-kg-03.txt
Wed 19 Oct 2022 04:11:23 PM JST
$ cat end-DW-u-mdx-kg-03.txt
Wed 19 Oct 2022 08:34:28 PM JST
```

この結果から， 4 時間23分であることがわかる[14]．

13）単体ベースのデータ抽出は，（主記憶の容量の制限から）CPU ノードで実行している．
14）この結果も，mdx における PG-Strom 環境のチューニングの余地があることを示し

なお，実際に出力されたファイルの情報は以下のように与えられる：

出力ファイル DW-mdx-Orbis2019u-20*.parquet に関する情報

```
% ls -la DW-mdx-Orbis2019u-20*.parquet
-rw-rw-r-- 1 masa masa 1076259815 Oct 19 16:38 DW-mdx-Orbis2019u-2009.parquet
-rw-rw-r-- 1 masa masa 1087612408 Oct 19 17:04 DW-mdx-Orbis2019u-2010.parquet
-rw-rw-r-- 1 masa masa 1113734278 Oct 19 17:30 DW-mdx-Orbis2019u-2011.parquet
-rw-rw-r-- 1 masa masa 1137747503 Oct 19 17:56 DW-mdx-Orbis2019u-2012.parquet
-rw-rw-r-- 1 masa masa 1164920861 Oct 19 18:22 DW-mdx-Orbis2019u-2013.parquet
-rw-rw-r-- 1 masa masa 1182233995 Oct 19 18:48 DW-mdx-Orbis2019u-2014.parquet
-rw-rw-r-- 1 masa masa 1190431648 Oct 19 19:14 DW-mdx-Orbis2019u-2015.parquet
-rw-rw-r-- 1 masa masa 1204757809 Oct 19 19:41 DW-mdx-Orbis2019u-2016.parquet
-rw-rw-r-- 1 masa masa 1222600485 Oct 19 20:07 DW-mdx-Orbis2019u-2017.parquet
-rw-rw-r-- 1 masa masa 1154017024 Oct 19 20:34 DW-mdx-Orbis2019u-2018.parquet
```

3. データ転送

この後のデータラングリングと可視化は，ローカル環境で行うため[15]，mdx 環境で抽出されたデータファイル DW-mdx-Orbis2019c-2009.parquet 〜 DW-mdx-Orbis2019c-2018.parquet（連結ベース），DW-mdx-Orbis2019u-2009.parquet 〜 DW-mdx-Orbis2019u-2009.parquet（単体ベース）を sftp コマンドでローカルに転送した（図 5）.

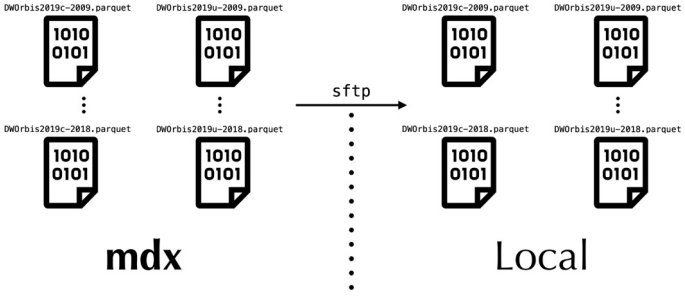

図 5：sftp コマンドによる mdx 環境からローカル環境への Orbis データファイルの転送

ている.

15）当然，この後で述べるデータラングリングを mdx 環境で行うことも可能であり，その方が処理時間も短縮されるであろうが，処理のための R スクリプトのコーディングや，データ可視化などの実行は，利便性の観点からローカル環境で行うほうがよいと判断した．なお，処理を実行するためのスクリプトがフィックスすれば，mdx 環境で実行することを検討する予定である.

次節では，転送されたファイルを利用してデータをラングリングすることについて詳細に述べる．

Ⅳ　データラングリング

本研究では，データをRに読み込み，解析できるオブジェクト形式に変換する工程をラングリング（wrangling）と呼ぶ（cf. Wickham and Grolemund (2016)）．この工程には，データの読み込み（load），行抽出（filter）や列選択（select），結合（join）等の操作（manipulate）も含まれることに留意する必要がある．なお，本稿ではラングリングの結果をファイル（Parquet形式）に書き出したものを再度利用してラングリングを行っている．

本稿のデータの利用における方針や，データがもつ問題とその解決法は，地道，阪（2022），地道ら（2022-a）で議論されたものに沿っているが，地道ら（2022-c）で議論されているように，2009年から2018年度の10年分に期間を拡大してデータをラングリングしたものを結合している．なお，データ利用における方針（Pol1）～（Pol3）とデータがもつ問題（Pro1）～（Pro5）とその解決法（Sol1）～（Sol5）を地道，阪（2022）から以下に引用する：

（Pol1）連結ベースのデータから，上場していないもの（`listed != "Listed"`）を抽出する[16]．

（Pol2）単体ベースのデータから，上場していないもの（`listed != "Listed"`）かつ単体財務諸表のみを保有している企業（`cons == "U1"`）を抽出する．

（Pol3）方針（Pol1），（Pol2）で得られたデータを結合する．
方針（Pol2）において，単体財務諸表のみを保有している企業（`cons == "U1"`）を選んだ理由としては，単体財務諸表を保有しており，何らかの理由で連結財務諸表も保有している企業

16）地道，阪（2022）でも指摘されているが，ヴェトナム（Vietnam）等の企業は単体であっても連結として収録されていることが判明したため，連結主体で抽出したデータから非上場企業のものを抽出している．

（cons == "U2"）は方針（Pol1）で利用する連結財務諸表を保有している企業の中に含まれるからである．

（Pro1）国別情報（country）に欠測がある．

（Pro2）国別情報として，例えば，自治領である "Isle Of Man (United Kingdom)"（マン島）というものがあり，このような場合にその企業の国別情報をどのように処理するかが問題となる．

（Pro3）BvD 社が企業ごとに付与しているユニークな ID コード（以後 BvD ID と呼ぶ）の先頭 2 文字には iso2c コード[17] が付与されており，これと国別情報（country）が異なる場合がある．

（Pro4）インフォメーションプロバイダ（データの提供元：infoprov）は国毎に「基本的にはユニーク」であるはずであるが，データの収集段階で複数から提供される場合がある．

（Pro5）方針（Pol1），（Pol2）にしたがって抽出されたそれぞれのデータセットの両方に属する企業が存在する．

（Sol1）国別情報（country）が欠測しているデータを除去する．

（Sol2）国別情報が自治領の場合は，括弧内の情報，例えば，マン島の場合は，"United Kingdom" を国別情報として利用する．

（Sol3）国別情報（country）を iso2c コードへ変換し，これと BvD ID の先頭 2 文字の iso2c コードが一致する企業のみを選択する．

（Sol4）国毎にインフォメーションプロバイダ（infoprov）が提供する企業数を算出し，最も多い企業数を提供するインフォメーションプロバイダに限定する．

（Sol5）データセット間に重複する企業は，連結コードが単体財務諸表のみを保有している企業（cons == "U1"）を選択することによっ

17）国際標準化機構（International Organization for Standardization：ISO）がラテン文字 2 文字で定める国名コードのこと．正式名称は，ISO 3166-1 alpha-2 である（https://www.iso.org/iso-3166-country-codes.html）．

て，重複を排除する．

1. ローカル環境におけるディレクトリ・ファイル構成

ローカル環境において，以降に行うデータラングリングと可視化を実行するためのデータファイルとスクリプトファイルを格納したディレクトリ構成を図 6 に与える．

```
working
├── 0DW-JHPCN
│   └── mdx
│       ├── DWOrbis2019c-2009.parquet
│       ├── DWOrbis2019c-2010.parquet
│       ├── DWOrbis2019c-2011.parquet
│       ├── DWOrbis2019c-2012.parquet
│       ├── DWOrbis2019c-2013.parquet
│       ├── DWOrbis2019c-2014.parquet
│       ├── DWOrbis2019c-2015.parquet
│       ├── DWOrbis2019c-2016.parquet
│       ├── DWOrbis2019c-2017.parquet
│       ├── DWOrbis2019c-2018.parquet
│       ├── DWOrbis2019u-2009.parquet
│       ├── DWOrbis2019u-2010.parquet
│       ├── DWOrbis2019u-2011.parquet
│       ├── DWOrbis2019u-2012.parquet
│       ├── DWOrbis2019u-2013.parquet
│       ├── DWOrbis2019u-2014.parquet
│       ├── DWOrbis2019u-2015.parquet
│       ├── DWOrbis2019u-2016.parquet
│       ├── DWOrbis2019u-2017.parquet
│       └── DWOrbis2019u-2018.parquet
├── 1DW-Local
│   ├── DW-Local-c-u1-join.R
│   ├── Makefile
│   ├── merge.R
│   ├── merge.sh
│   └── seqDW-c-u1-join-p-2.sh
├── 2Visualization
│   ├── Makefile
│   ├── figures
│   └── makepng.R
├── 3paper
│   ├── DataWrangling-Visualization-OrbisData.Rnw
│   ├── Makefile
│   └── makepaper.sh
└── Makefile
```

図 6 ：ローカル環境用スクリプトファイルのディレクトリ構成

2. ラングリング手順

ラングリングの具体的な手順を以下に与える：[18]:

（DW-L-1）単年度分の連結データの Parquet ファイル DW-mdx-Orbis2019c-20xx.parquet と，単年度分の単体データの Parquet ファイル DW-mdx-Orbis2019u-20xx.parquet をラングリングし結合したものを，年度毎に Parquet ファイル Orbis2019c-u1-20xxpreped.parquet と，今回の可視化に利用する列のみを含む Parquet ファイル Orbis2019c-u1-sub-20xxpreped.parquet に出力することを10回繰り返す（ここで，xx には 09，...，18 が入る）．

（DW-L-2）手順（DW-L-1）で得られた Parquet ファイル Orbis2019c-u1-sub-20xxpreped.parquet を順次読み込み，可視化に必要な列（変数）を選択したものを10年分結合し，CSV ファイル Orbis2019c-u1-sub.csv と Parquet ファイル Orbis2019c-u1-sub.parquet として書き出す．

実際には，これらの手順を実行するターゲット all を working/1DW-Local/Makefile に記述し，make コマンドを利用することによって実行する仕様とした．

スクリプト 5 ：working/1DW-Local/Makefile

```
1 all:
2       date > start-all.txt
3       /bin/sh seqDW-c-u1-join-p-2.sh
4       /bin/sh merge.sh
5       date > end-all.txt
```

スクリプト 5 の 3 行目が手順（DW-L-1）の処理を行い，4 行目が手順（DW-L-2）を実行する．以下にこれらの手順で行われる具体的な処理について説明する．

18）本来は，この手順において，一時ファイルを書き出すことなしに行いたいところであるが，メモリの制限の関係で段階的に行わざるを得なかった．

手順（DW-L-1）の処理

スクリプト5の3行目では，シェル・スクリプト・ファイル seqDW-c-u1-join-p-2.sh（スクリプト6）が実行されるように定義されている．

スクリプト6：seqDW-c-u1-join-p-2.sh

```
#!/bin/sh
echo "Start␣process␣with␣parallel"
seq 2009 2010 | parallel --jobs 100% Rscript DW-Local-c-u1-join.R "../0DW-JHPCN/mdx/DWOrbis2019c-"{}".parquet" "../0DW-JHPCN/mdx/DWOrbis2019u-"{}".parquet" "Orbis2019c-u1-"{}"preped.parquet" "Orbis2019c-u1-sub-"{}"preped.parquet"
seq 2011 2012 | parallel --jobs 100% Rscript DW-Local-c-u1-join.R "../0DW-JHPCN/mdx/DWOrbis2019c-"{}".parquet" "../0DW-JHPCN/mdx/DWOrbis2019u-"{}".parquet" "Orbis2019c-u1-"{}"preped.parquet" "Orbis2019c-u1-sub-"{}"preped.parquet"
seq 2013 2014 | parallel --jobs 100% Rscript DW-Local-c-u1-join.R "../0DW-JHPCN/mdx/DWOrbis2019c-"{}".parquet" "../0DW-JHPCN/mdx/DWOrbis2019u-"{}".parquet" "Orbis2019c-u1-"{}"preped.parquet" "Orbis2019c-u1-sub-"{}"preped.parquet"
seq 2015 2016 | parallel --jobs 100% Rscript DW-Local-c-u1-join.R "../0DW-JHPCN/mdx/DWOrbis2019c-"{}".parquet" "../0DW-JHPCN/mdx/DWOrbis2019u-"{}".parquet" "Orbis2019c-u1-"{}"preped.parquet" "Orbis2019c-u1-sub-"{}"preped.parquet"
seq 2017 2018 | parallel --jobs 100% Rscript DW-Local-c-u1-join.R "../0DW-JHPCN/mdx/DWOrbis2019c-"{}".parquet" "../0DW-JHPCN/mdx/DWOrbis2019u -"{}".parquet" "Orbis2019c-u1-"{}"preped.parquet" "Orbis2019c-u1-sub-"{}"preped.parquet"
```

シェル・スクリプト・ファイル seqDW-c-u1-join-p-2.sh（スクリプト6）の3行目から7行目では，GNU parallel を利用して Rscript コマンドから R スクリプトファイル DW-Local-c-u1-join.R（スクリプト7）を実行することによって，2年分のファイルを並列処理する仕様となっている[19]．

スクリプト7：DW-Local-c-u1-join.R

```
require(tidyverse)
require(arrow)
require(countrycode)
#
args <- commandArgs(trailingOnly = T)
#
```

19）この工程において，並列処理するファイルの個数のパターンを幾つか試した結果，macOS 環境で経験的に 2 個が時間的にも安定的であった．

```
xc <- read_parquet(args[1]) %>% tibble() %>% filter(listed != "Listed")
#
xu <- read_parquet(args[2]) %>% tibble() %>% filter(listed != "Listed")
#
pattern.p.xc <- xc %>% pull(country) %>% str_detect("\\(")
xc.p <- xc[pattern.p.xc,]
xc.np <- xc[!pattern.p.xc,]
#
pattern.p.xu <- xu %>% pull(country) %>% str_detect("\\(")
xu.p <- xu[pattern.p.xu,]
xu.np <- xu[!pattern.p.xu,]
#
yc <- bind_rows(
  xc.np %>%
    filter(!is.na(country)) %>%
    mutate(country.x = country),
  xc.p %>%
    filter(!is.na(country)) %>%
    mutate(country.x = as.character(str_match(country, "(?<=\\().*?(?=\\))"
        )))
) %>%
  mutate(
    iso2c.x = countrycode(country.x, origin = 'country.name', destination =
        'iso2c'),
    iso2c.y = substr(id, 1, 2)
    )
#
yu <- bind_rows(
  xu.np %>%
    filter(!is.na(country)) %>%
    mutate(country.x = country),
  xu.p %>%
    filter(!is.na(country)) %>%
    mutate(country.x = as.character(str_match(country, "(?<=\\().*?(?=\\))"
        )))
) %>%
  mutate(
    iso2c.x = countrycode(country.x, origin = 'country.name', destination =
         'iso2c'),
    iso2c.y = substr(id, 1, 2)
    )
#
yc.sel <- yc %>% filter(iso2c.x == iso2c.y)
#
yu.sel <- yu %>% filter(iso2c.x == iso2c.y)
#
list.country.infoprov <- function(df)
{
  require(dplyr)
  require(countrycode)
  ISO2C <- df %>% pull(iso2c.x) %>% unique() %>% sort()
  nc <- length(ISO2C)
  info.list <- list()
  for(i in 1:nc)
  {
    x <- df %>%
```

```
    filter(iso2c.x == ISO2C[i]) %>%
    pull(infoprov) %>% table() %>%
    sort(decreasing = TRUE) %>% list()
  info.list <- c(info.list, x)
 }
 names(info.list) <- countrycode(ISO2C, origin = 'iso2c', destination = '
    country.name')
 info.list
}
#
all.list.cip.yc <- list.country.infoprov(df = yc.sel)
#
all.list.cip.yu <- list.country.infoprov(df = yu.sel)
#
str(all.list.cip.yc)
#
cip <- function(x)
{
 require(countrycode)
 require(dplyr)
 require(tibble)
 n <- length(x)
 country <- names(x)
 top.ip <- rep(0, n)
 for(i in 1:n) top.ip[i] <- names(x[i][[1]])[1]
 iso2c <- countrycode(country, origin = 'country.name', destination = '
    iso2c')
 x <- tibble(country, iso2c, top.ip) %>% na.omit() # for Kosovo
 x
}
#
cip.yc.frame <- cip(x = all.list.cip.yc)
#
cip.yu.frame <- cip(x = all.list.cip.yu)
#
zc <- yc.sel %>%
 filter(country.x != "Kosovo", country.x != "Namibia") %>%
 left_join(cip.yc.frame[, c(2, 3)], by = c("iso2c.x" = "iso2c"))
#
zu <- yu.sel %>%
 filter(country.x != "Kosovo", country.x != "Namibia") %>%
 left_join(cip.yu.frame[, c(2, 3)], by = c("iso2c.x" = "iso2c"))
zu1 <- zu %>% filter(cons == "U1")
#
zc.top.ip <- zc %>% filter(infoprov == top.ip)
#
zu1.top.ip <- zu1 %>% filter(infoprov == top.ip)
#
zcu <- bind_rows(zc.top.ip, zu1.top.ip)
#
count.firms <- zcu %>% pull(firmid) %>% table()
dc.firms <- names(count.firms[count.firms >=2])
#
`%nin%` <- Negate(`%in%`)
#
zcu.ndc <- zcu %>% filter(firmid %nin% dc.firms)
```

```
113 #
114 zcu.dc <- zcu %>% filter(firmid %in% dc.firms) %>% arrange(firmid) %>%
        filter(cons == "U1")
115 #
116 cu <- bind_rows(zcu.ndc, zcu.dc)
117 #
118 write_parquet(cu, args[3])
119 #
120 subcol <- c(
121   "firmid",
122   "year",
123   "iso2c.x",
124   "month",
125   "assets_total",
126   "shareholders",
127   "employees",
128   "operating_revenue",
129   "pl_before_tax",
130   "tax",
131   "net_income",
132   "costs_employees",
133   "interest_paid")
134 #
135 cu.sub <- cu %>% select(one_of(subcol))
136 write_parquet(cu.sub, args[4])
```

スクリプト 7 は以下のような処理を行う.

1 行目～ 3 行目：R パッケージの読み込み

5 行目：スクリプト 7 の実行時に読み込まれた引数をオブジェクト `args` へ付値

7 行目：**{arrow}** パッケージに付属する関数 `read_parquet` を利用して 20xx 年度分の連結ベースデータファイル（`DW-mdx-Orbis2019c-20xx.parquet`）の読み込んだものを `tibble` 関数で変換後，非上場企業を選択（`filter(listed != "Listed")`）したものをオブジェクト `xc` に付値

9 行目：**{arrow}** パッケージに付属する関数 `read_parquet` を利用して 20xx 年度分の単体ベースデータファイル（`DW-mdx-Orbis2019u-20xx.parquet`）の読み込んだものを `tibble` 関数で変換後，非上場企業を選択（`filter(listed != "Listed")`）したものをオブジェクト `xu` に付値

11行目：オブジェクト `xc` から国別情報に自治領等（括弧 `()` を含むもの）

のパターンを抽出したものをオブジェクト pattern.p.xc に付値

12行目：オブジェクト xc の国別情報に自治領等（括弧 () を含むもの）のパターン pattern.p.xc にしたがうものを抽出し，オブジェクト xc.p に付値

13行目：オブジェクト xc の国別情報に自治領等（括弧 () を含むもの）のパターン pattern.p ではないものを抽出し，オブジェクト xc.np に付値

15行目：オブジェクト xu から国別情報に自治領等（括弧 () を含むもの）のパターンを抽出したものをオブジェクト pattern.p.xu に付値

16行目：オブジェクト xu の国別情報に自治領等（括弧 () を含むもの）のパターン pattern.p.xu にしたがうものを抽出し，オブジェクト xu.p に付値

17行目：オブジェクト xu の国別情報に自治領等（括弧 () を含むもの）のパターン pattern.p ではないものを抽出し，オブジェクト xu.np に付値

19行目～30行目：ここでの処理は以下のようなものである：

- 国別情報に括弧 () を含まないパターンのオブジェクト xc.np から，関数 filter を使って国別情報が欠測しているものを取り除き（対処法（Sol1）の実現），国別情報（country）と同じ情報の列 county.x を関数 mutate を使って追加
- 国別情報に括弧 () を含むパターンのオブジェクト xc.np から，関数 filter を使って国別情報が欠測しているものを取り除き（(対処法（Sol1）の実現），国別情報（country）の括弧の中の文字列（国情報）をもつ列 county.x を関数 mutate を使って追加（対処法（Sol2）の実現）
- 以上の処理によって生成されたオブジェクトを関数 bind_rows で行結合
- 新しく追加された国別情報の列 country.x を **{countrycode}**

パッケージに付属する関数 countrycode を利用して，iso2c コードに変換し，列 iso2c.x として追加

- BvD ID（id）の先頭 2 文字に存在する iso2c コードを抽出し，列 iso2c.y として追加
- 以上の処理によって生成されるオブジェクトを yc に付値

32行目～43行目：ここでの処理は以下のようなものである：

- 国別情報に括弧 () を含まないパターンのオブジェクト xu.np から，関数 filter を使って国別情報が欠測しているものを取り除き（対処法（Sol1）の実現），国別情報（country）と同じ情報の列 county.x を関数 mutate を使って追加
- 国別情報に括弧 () を含むパターンのオブジェクト xu.np から，関数 filter を使って国別情報が欠測しているものを取り除き（(対処法（Sol1）の実現），国別情報（country）の括弧の中の文字列（国情報）をもつ列 county.x を関数 mutate を使って追加（対処法（Sol2）の実現）
- 以上の処理によって生成されたオブジェクトを関数 bind_rows で行結合
- 新しく追加された国別情報の列 country.x を **{countrycode}** パッケージに付属する関数 countryuode を利用して，iso2c コードに変換し，列 iso2c.x として追加
- BvD ID（id）の先頭 2 文字に存在する iso2c コードを抽出し，列 iso2c.y として追加
- 以上の処理によって生成されるオブジェクトを yu に付値

45行目：オブジェクト yc から国別情報（country.x）にもとづく iso2c コード（iso2c.x）と，BvD ID の先頭 2 文字の iso2c コード（iso2c.y）が一致するものを選択し，オブジェクト yc.sel に付値（対処法（Sol3）の実現）

47行目：オブジェクト yu から国別情報（country.x）にもとづく iso2c

コード（iso2c.x）と，BvD ID の先頭 2 文字の iso2c コード（iso2c.y）が一致するものを選択し，オブジェクト yu.sel に付値（対処法（Sol3）の実現）

49行目～66行目：オブジェクト yc.sel，yu.sel から，国毎にインフォメーションプロバイダによって情報提供される企業数をカウントし，出力する関数 list.country.infoprov を定義

68行目：オブジェクト yu に対して関数 list.country.infoprov の実行結果をオブジェクト all.list.cip.yc に付値

70行目：オブジェクト yc に対して関数 list.country.infoprov の実行結果をオブジェクト all.list.cip.yu に付値

74行目～86行目：オブジェクト all.list.cip.yc，all.list.cip.yu から，国毎に最も多くの企業情報を提供しているインフォメーションプロバイダ（トップ・インフォメーション・プロバイダ；top.ip）を出力する関数 cip を定義

88行目：オブジェクト yc.sel に対して関数 cip の実行結果をデータ・フレーム・オブジェクト cip.yc.frame に付値

90行目：オブジェクト yu.sel に対して関数 cip の実行結果をデータ・フレーム・オブジェクト cip.yu.frame に付値

92行目～94行目：オブジェクト yc.sel から，コソボ（Kosobo）とナミビア（Namibia）に関する行を削除[20] し，トップ・インフォメーション・プロバイダが収録されたデータ・フレーム・オブジェクト cip.yc.frame と結合（left_join）し，オブジェクト zc に付値

96行目～98行目：オブジェクト yu.sel から，コソボ（Kosobo）とナミビア（Namibia）に関する行を削除し，トップ・インフォメーション・プロバイダが収録されたデータ・フレーム・オブジェクト cip.yu.frame と結合（left_join）し，オブジェクト zu に付値

20）コソボ（Kosobo）とナミビア（Namibia）は countrycode 関数で扱うことが難しいため，今回は削除することとした．

99行目：オブジェクト zu から連結コードを単体財務諸表のみを保有している企業に限定（filter(cons == "U1")）したものをオブジェクト zu1 に付値

101行目：オブジェクト zc から，インフォメーションプロバイダに関する情報が（トップ・インフォメーション・プロバイダに一致する（infoprov == top.ip）企業を抽出し，オブジェクト zc.top.ip に付値（対処法（Sol4）の実現）

103行目：オブジェクト zu1 から，インフォメーションプロバイダに関する情報が（トップ・インフォメーション・プロバイダに一致する（infoprov == top.ip）企業を抽出し，オブジェクト zu1.top.ip に付値（対処法（Sol4）の実現）

105行目：オブジェクト zc.top.ip とオブジェクト zu1.top.ip を関数 bind_rows で行結合し，オブジェクト zcu に付値

107行目：オブジェクト zcu の列 firmid（企業名 ＋BvD ID）を関数 pull で選択し，関数 table でクロス集計した結果をオブジェクト count.firms へ付値

108行目：複数存在する企業名を抽出し，オブジェクト dc.firms へ付値

110行目：ある集合に「属さない」要素を「走査」する演算子 %nin% を定義

112行目：オブジェクト zcu の中でユニークな企業（オブジェクト dc.firms に属さない企業）を抽出し，オブジェクト zcu.ndc に付値

114行目：オブジェクト zcu の中で重複している企業（オブジェクト dc.firms に属す企業）を抽出し，企業名で並べ変えた後，連結コードを単体財務諸表のみを保有している企業に限定（filter(cons == "U1")）し，オブジェクト zcu.dc に付値（対処法（Sol5）の実現）

116行目：オブジェクト zcu.ndc とオブジェクト zcu.dc を行結合し，オブジェクト cu に付値

118行目：オブジェクト cu を 3 番目に指定された引数名の Parquet ファイ

ルとして出力

120行目～133行目：可視化に必要な列（変数）名をオブジェクト subcol へ付値

135行目：オブジェクト cu からオブジェクト subcol に含まれる列を選択しオブジェクト cu.sub へ付値

136行目：オブジェクト cu.sub を 4 番目に指定された引数名の Parquet ファイルとして出力

データラングリングにともなうスクリプトファイルと Orbis データファイルの流れとそれらの対応を可視化したものを図 7 に与える.

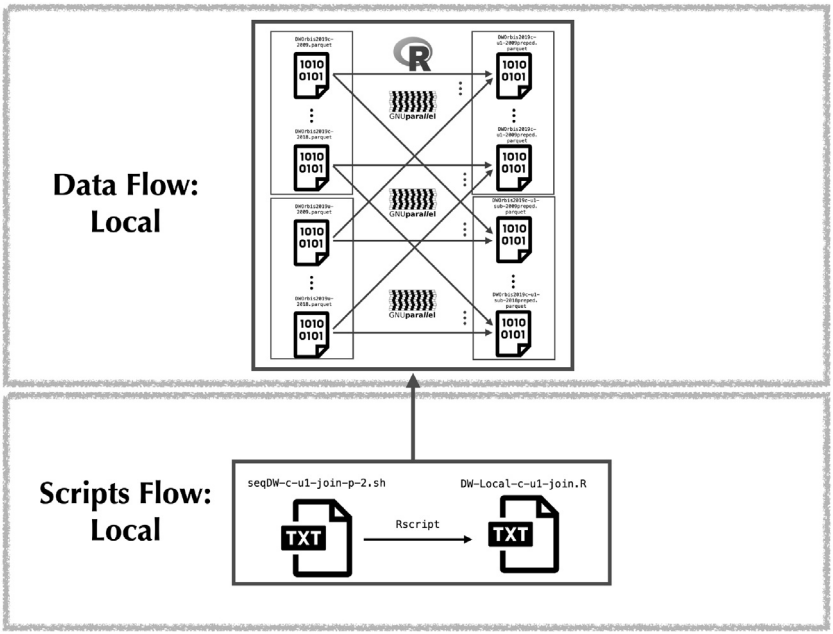

図 7：データのラングリング

実際に出力されたファイルの情報は以下のように与えられる：

出力ファイル Orbis2019c-u1-20**preped.parquet に関する情報

```
% ls -l Orbis2019c-u1-20*
-rw-r--r--@ 1 masa  staff  1152263801 12  2 17:56 Orbis2019c-u1-2009preped.parquet
-rw-r--r--@ 1 masa  staff  1164201640 12  2 17:56 Orbis2019c-u1-2010preped.parquet
-rw-r--r--@ 1 masa  staff  1188640945 12  2 18:06 Orbis2019c-u1-2011preped.parquet
-rw-r--r--@ 1 masa  staff  1213596726 12  2 18:06 Orbis2019c-u1-2012preped.parquet
-rw-r--r--@ 1 masa  staff  1248052849 12  2 18:16 Orbis2019c-u1-2013preped.parquet
-rw-r--r--@ 1 masa  staff  1270598282 12  2 18:16 Orbis2019c-u1-2014preped.parquet
-rw-r--r--@ 1 masa  staff  1281907491 12  2 18:26 Orbis2019c-u1-2015preped.parquet
-rw-r--r--@ 1 masa  staff  1298428977 12  2 18:26 Orbis2019c-u1-2016preped.parquet
-rw-r--r--@ 1 masa  staff  1312365620 12  2 18:36 Orbis2019c-u1-2017preped.parquet
-rw-r--r--@ 1 masa  staff  1242696339 12  2 18:36 Orbis2019c-u1-2018preped.parquet
```

出力ファイル Orbis2019c-u1-sub-20**preped.parquet に関する情報

```
% ls -l Orbis2019c-u1-sub-20*
-rw-r--r--@ 1 masa  staff  815292943 12  2 17:56 Orbis2019c-u1-sub-2009preped.parquet
-rw-r--r--@ 1 masa  staff  821805559 12  2 17:56 Orbis2019c-u1-sub-2010preped.parquet
-rw-r--r--@ 1 masa  staff  842594851 12  2 18:06 Orbis2019c-u1-sub-2011preped.parquet
-rw-r--r--@ 1 masa  staff  859858080 12  2 18:06 Orbis2019c-u1-sub-2012preped.parquet
-rw-r--r--@ 1 masa  staff  882596450 12  2 18:17 Orbis2019c-u1-sub-2013preped.parquet
-rw-r--r--@ 1 masa  staff  897569827 12  2 18:17 Orbis2019c-u1-sub-2014preped.parquet
-rw-r--r--@ 1 masa  staff  902124818 12  2 18:27 Orbis2019c-u1-sub-2015preped.parquet
-rw-r--r--@ 1 masa  staff  911790058 12  2 18:27 Orbis2019c-u1-sub-2016preped.parquet
-rw-r--r--@ 1 masa  staff  923182701 12  2 18:37 Orbis2019c-u1-sub-2017preped.parquet
-rw-r--r--@ 1 masa  staff  867909365 12  2 18:37 Orbis2019c-u1-sub-2018preped.parquet
```

手順（DW-L-2）の処理

ディレクトリ working/1DW-Local（図 6 参照）においてデータを結合し，その結果を CSV, Parquet ファイルとして出力する．このディレクトリに配置された Makefile（スクリプト 5）の 4 行目で指定されたシェル・スクリプト・ファイル merge.sh（スクリプト 8）では，Rscript コマンドで R スクリプトファイル merge.R（スクリプト 9）を実行することが定義されている．

スクリプト 8 ：merge.sh

```
1 #!/bin/sh
2 Rscript merge.R
```

スクリプト 9 ：merge.R

```
1 library(tidyverse)
2 #
3 merge.data <- function()
```

```
 4 {
 5   require(arrow)
 6   require(dplyr)
 7   y <- NULL
 8   for(i in 2009:2018)
 9     {
10     x <- read_parquet(paste0("Orbis2019c-u1-sub-",i,"preped.parquet"))
11     y <- x %>% bind_rows(y, .)
12     rm(x)
13     gc();gc()
14     cat("year␣=",i,",")
15     }
16   y %>% arrange(firmid, year)
17 }
18 x <- merge.data()
19 x %>% write_csv(file = "Orbis2019c-u1-sub.csv")
20 x %>% arrow::write_parquet(sink = "Orbis2019c-u1-sub.parquet")
```

スクリプト 9 は以下のような処理を行う：

1 行目：R パッケージ（群）**{tidyverse}** の読み込み

3 行目〜17行目：ファイル `Orbis2019c-u1-sub-20xxpreped.parquet` を読み込み，行結合する関数 `merge.data` を定義

18行目：関数 `merge.data` を実行し，データの結合を行った結果をオブジェクト `x` へ付値

19行目：オブジェクト `x` を CSV ファイル `Orbis2019c-u1-sub.csv` として出力

20行目：オブジェクト `x` を Parquet ファイル `Orbis2019c-u1-sub.parquet` として出力

実際に出力されたファイルの情報は以下のように与えられる：

出力ファイル `Orbis2019c-u1*.parquet` に関する情報

```
% ls -l Orbis2019c-u1-sub.*
-rw-r--r--@ 1 masa  staff  20154219231 12  2 19:52 Orbis2019c-u1-sub.csv
-rw-r--r--@ 1 masa  staff   2736252471 12  2 19:54 Orbis2019c-u1-sub.parquet
```

なお，データラングリングにともなうスクリプトファイルと Orbis データファイルの流れとそれらの対応を可視化したものを図 8 に与える．

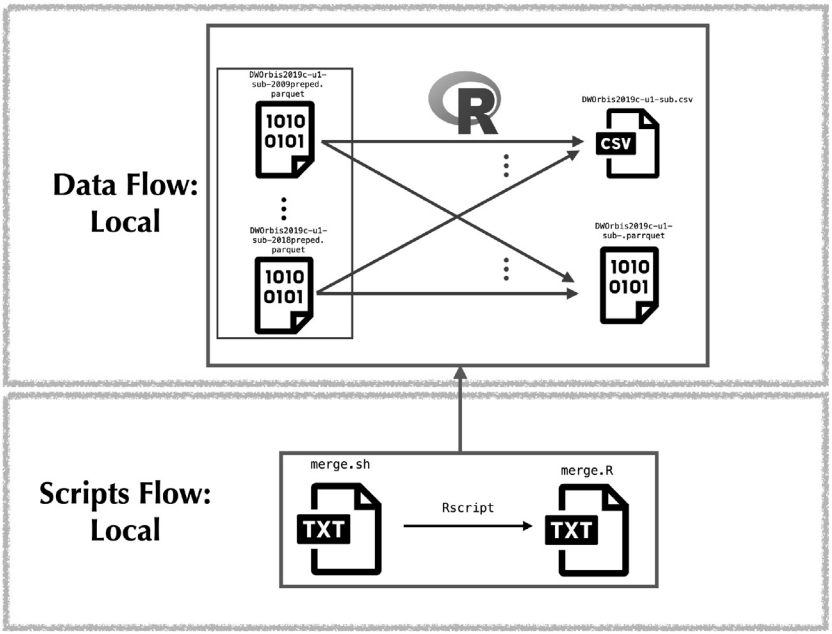

図 8 ：データの結合とファイル出力

ラングリング全体の自動実行

ラングリングの全工程を一括で処理するためには，working/1DW-Local をカレントディレクトリ（図 6 参照）として，ターミナル（コマンドライン）上でターゲット all を make コマンドで実行すればよい：

ターゲット all の実行

```
$ make all
```

この処理時間は，スクリプト 5 における，(2, 5) 行目で実行される結果を比較することによってわかる．

ターゲット all の処理時間の計測

```
% cat start-all.txt
2022年 12月 25日 日曜日 14時 10分 57秒 JST
% cat end-all.txt
2022年 12月 25日 日曜日 17時 32分 48秒 JST
```

この結果から，3時間22分であることがわかる[21]．working/1DW-Local/Makefile におけるターゲット all の処理とファイル・ディレクトリとの対応を図9に与える．

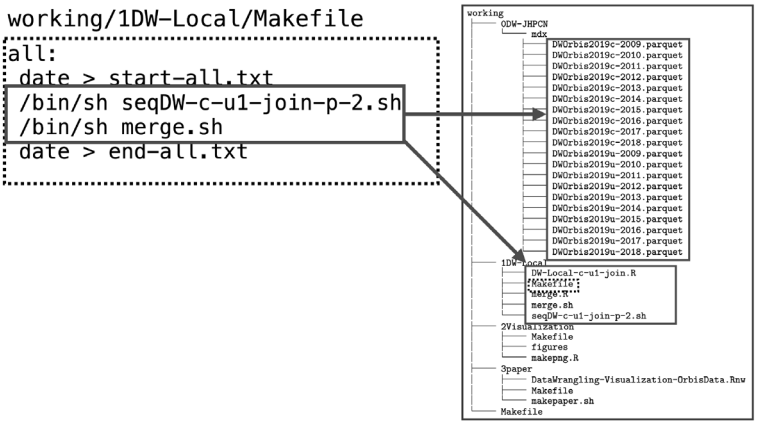

図9：working/1DW-Local/Makefile におけるターゲット all の処理とファイル・ディレクトリとの対応

また，データラングリングの工程に利用されるスクリプトファイルと Orbis データファイルの流れとそれらの対応を可視化したものを図10に与える．

21）この結果は，Mac Studio 2022 で計測した．

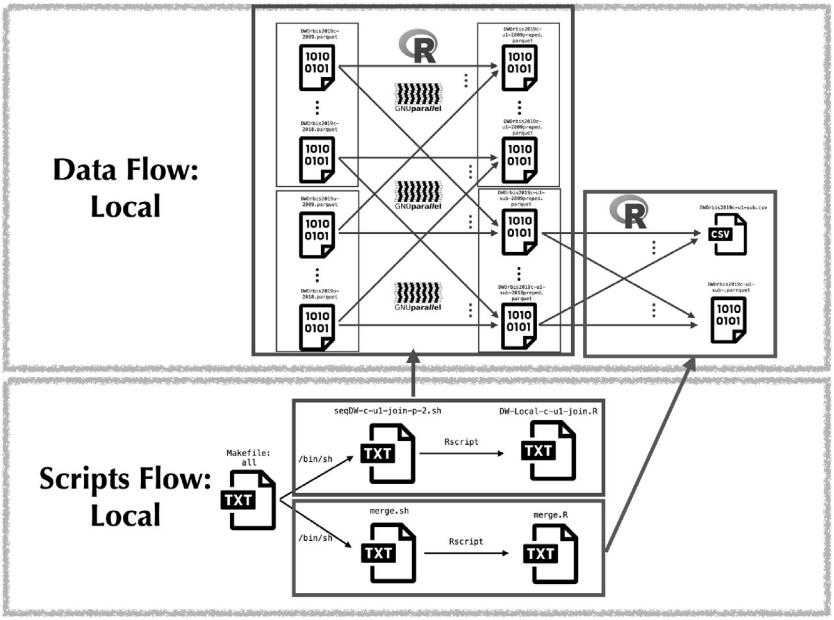

図10：データラングリングの工程の自動実行

V　データ可視化とその自動化

1. データ可視化の自動化

ここでは，データを可視化するための工程を自動化する方法を議論することによって再現性を確保する方策について述べる[22]．具体的には，ディレクトリ `working/2Visualization`（図 6 参照）においてデータを読み込み・可視化し，その結果を PNG ファイルとして出力する．この処理は，ディレクトリ `working/2Visualization` に配置された `Makefile`（スクリプト 10）を利用し，3 行目で `Rscript` コマンドで R スクリプトファイル `makepng.R`（付録 B のスクリプト11）を実行する仕様となっている．

22）データを可視化した結果から得られる知見や考察については別の機会に譲る．

スクリプト10：working/2Visualization/Makefile

```
1 png:
2 	date > start-png.txt
3 	Rscript makepng.R
4 	date > end-png.txt
```

データ可視化の工程を自動処理するためには，working/2Visualization をカレントディレクトリ（図6参照）として，ターミナル（コマンドライン）上でターゲット png を make コマンドで実行すればよい：

ターゲット png の実行

```
$ make png
```

この処理時間は，スクリプト10における，(2, 4) 行目で実行される結果を比較することによってわかる．

ターゲット png の処理時間の計測

```
% cat start-png.txt
2022年 12月 25日 日曜日 17時 32分 48秒 JST
% cat end-png.txt
2022年 12月 25日 日曜日 18時 59分 22秒 JST
```

この結果から，1時間27分であることがわかる[23]．working/2Visualization/Makefile におけるターゲット png の処理とファイル・ディレクトリとの対応を図11に与える．

23）この結果は，Mac Studio 2022 で計測した．

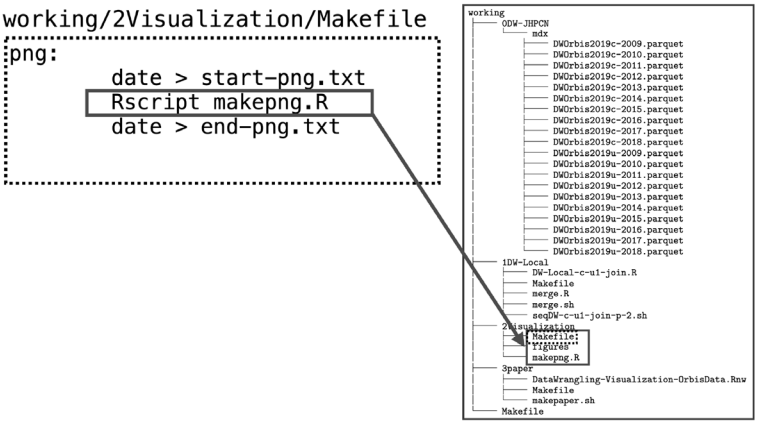

図11：working/2Visualization/Makefile におけるターゲット png の処理とファイル・ディレクトリとの対応

また，データ可視化の工程に利用されるスクリプトファイルと Orbis データファイルとその可視化した結果の画像ファイルの流れとそれらの対応を図12に与える．

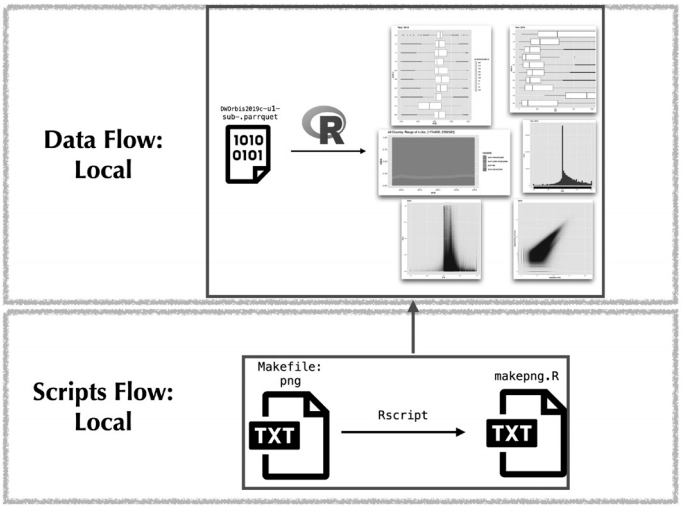

図12：データ可視化の画像ファイルの自動生成

2. データ可視化の具体例

ここでは，前小節で与えられた幾つかの可視化を行った結果を与える．まず，2018会計年度の非上場企業に関する対数資産合計（log(assets_total)）と対数売上高（log(operating_revenue)）の散布図を与える（図13参照）[24]．図13をよく見ると，峰は少なくとも2つ存在することがわかり，何らかの混合分布に従うことが見て取れる．

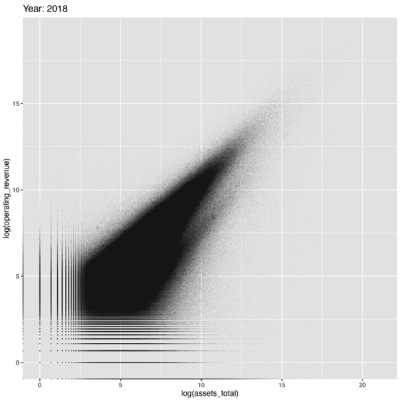

図13：2018会計年度の非上場企業に関する対数資産合計と対数売上高の散布図

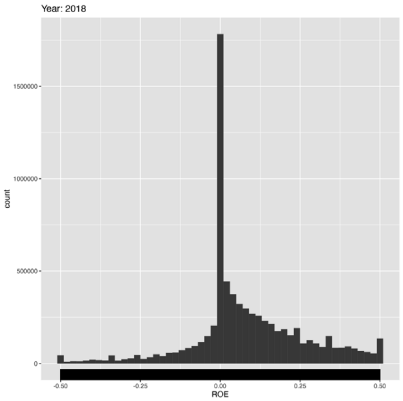

図14：自己資本利益率のヒストグラム：範囲を $-0.5<\mathrm{ROE}<0.5$ に限定

また，収益性を確認するにあたり，世界の非上場企業の2018会計年度の(1)式で定義される自己資本利益率（Return On Equity: ROE）についての全体の分布状況を可視化した（図14）．

$$\mathrm{ROE} := \frac{\text{当期純利益}}{\text{純資産}} = \left(\frac{\texttt{net\_income}}{\texttt{shareholders}}\right) \tag{1}$$

これは，表示範囲を“$-0.5<\mathrm{ROE}<0.5$”（±50%）に限定したヒストグラムである．このヒストグラムからは，分布が $\mathrm{ROE}=0$ で非対称となっている

24）Jimichi *et al.*（2018）では，世界の上場企業に関する売上高を従業員数と資産合計によって統計モデリングを行うために，散布図を描くことによって知見を得ている．

ことが確認でき，作為的な会計処理の可能性がみえる．

VI おわりに

本稿では，mdx 環境とローカル環境を協調することによって，データベース Orbis から抽出された非上場企業の財務データを前処理，ラングリング，可視化する工程を自動化し，探索的データ解析を再現可能性をもって行うことを考察した．本稿で扱った処理工程を全て mdx 環境で行うことや，非上場企業に対する財務データの可視化（の結果）についての考察，及び得られた知見にもとづく統計モデリングについては今後の課題としたい．

（筆者らは関西学院大学商学部教授）

付記

本稿は2022年11月27日（日）に富山国際会議場で開催された日本計算機統計学会第34回シンポジウムでの報告（タイトル：『探索的財務ビッグデータ解析と再現可能研究：非上場企業のデータ可視化』）における予稿集用原稿に大幅な加筆・修正を加えたものである．

参考文献

［1］ Gandrud, C.（2020）*Reproducible Research with R and RStudio, Third Edition*, CRC Press.

［2］ 地道正行（2018）『データサイエンスの基礎：R による統計学独習』，裳華房．

［3］ 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．

［4］ 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．

［5］ Jimichi, M., D. Miyamoto, C. Saka, and S. Nagata（2018）Visualization and statistical modeling of financial big data: Double-log modeling with skew-symmetric error distributions, *Japanese Journal of Statistics and Data Science*, Vol. 1, No. 2, pp. 347–371, https://doi.org/10.1007/s42081-018-0019-1

［6］ 地道正行，宮本大輔，阪智香，永田修一（2020-a）『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』，日本計算機統計学会，第34回大会，オンライン開催，2020年 5 月31日（日），講演論文集，pp. 41–44.

［7］ 地道正行，宮本大輔，阪智香，永田修一（2020-b）『財務ビッグデータの可視化と統計モデリング』，学際大規模情報基盤共同利用・共同研究拠点（JHPCN），第12回シンポジウム，オンライン開催，2020年 7 月 9 日（木），発表用ポスター．https://jhpcn-kyoten.itc.u-tokyo.ac.jp/abstract/jh191002-NWJ

［8］ 地道正行，宮本大輔，阪智香，永田修一（2020-c）『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』，国際数理科学協会，2020年度年会「統計的推測と統計ファイナンス」分科会研究集会，大阪大学，オンライン開催，2020年8月22日（土），配付資料.

［9］ 地道正行，宮本大輔，阪智香，永田修一（2021-a）『財務ビッグデータの可視化と統計モデリング』，学際大規模情報基盤共同利用・共同研究拠点（JHPCN），第13回シンポジウム，オンライン開催，2021年7月8日（金），発表用ポスター．https://jhpcn-kyoten.itc.u-tokyo.ac.jp/abstract/jh211001-NWJ

［10］ 地道正行，宮本大輔，阪智香，永田修一（2021-b）『R による財務ビッグデータのラングリング再考』，2021年度「データ解析環境 R の整備と利用」研究集会，オンライン開催，2021年12月18日（土），発表用資料 https://drive.google.com/file/d/1s6gA3m0T9F_xy6nce3nEGuD7GKuEtZKt/view

［11］ 地道正行，阪智香（2022）『探索的財務ビッグデータ解析と再現可能研究：非上場企業のデータラングリング』，商学論究，第69巻，第3・4号，pp. 83–120，関西学院大学商学研究会.

［12］ 地道正行，阪智香，宮本大輔，永田修一（2022-a）『探索的財務ビッグデータ解析と再現可能研究：非上場企業のデータラングリング』日本計算機統計学会，第36回大会，愛媛県県民文化会館本館（ハイブリッド開催），2022年5月22日（日），講演論文集， pp. 87–90.

［13］ 地道正行，宮本大輔，阪智香，永田修一（2022-b）『財務ビッグデータの可視化と統計モデリング』，学際大規模情報基盤共同利用・共同研究拠点（JHPCN），第13回シンポジウム，オンライン開催，2022年7月7日（木），発表用ポスター．https://jhpcn-kyoten.itc.u-tokyo.ac.jp/abstract/jh221001

［14］ 地道正行，阪智香，宮本大輔，永田修一（2022-c）『探索的財務ビッグデータ解析と再現可能研究：非上場企業のデータ可視化』，日本計算機統計学会，第34回シンポジウム，ハイブリッド開催，2022年11月27日（日），講演論文集，pp. 160–163.

［15］ 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.

［16］ Mecklenburg, R.（2005）*Managing Projects with GNU Make, Third Edition*, O'Reilly Media, Inc.

［17］ Peng, R. D.（2011）Reproducible research in computational science, *Science*, Vol. 334, pp. 1226–1227.

［18］ Suzumura, T., A. Sugiki, H. Takizawa, A. Imakura, H. Nakamura, K. Taura, T. Kudoh, T. Hanawa, Y. Sekiya, H. Kobayashi, Y. Kuga, R. Nakamura, R. Jiang, J. Kawase, M. Hanai, H. Miyazaki, T. Ishizaki, D. Shimotoku, D. Miyamoto, K. Aida, A. Takefusa, T. Kurimoto, K. Sasayama, N. Kitagawa, I. Fujiwara, Y. Tanimura, T. Aoki, T. Endo, S. Ohshima, K. Fukazawa, S. Date, and T Uchibayashi（2022）mdx: A Cloud Platform for Supporting Data

Science and Cross-Disciplinary Research Collaborations, *The 8th IEEE International Conference on Cloud and Big Data Computing*（*CBDCom 2022*）, doi 10.48550/ARXIV.2203.14188, https://arxiv.org/abs/2203.14188

［19］ Tange, O.（2022）*GNU Parallel20221022*（*'Nord Stream'*）, Zenodo, DOI:10.5281/zenodo.7239559, URL: https://doi.org/10.5281/zenodo.7239559

［20］ Tukey, J. W.（1977）*Exploratory Data Analysis*, Addison-Wesley Publishing Co.

［21］ Wickham, H.（2016）*ggplot2 : Elegant Graphics for Data Analysis, Second Edition*, Springer.（石田 基広，石田 和枝 共訳（2011）『グラフィックスのための R プログラミング：ggplot2 入門』，シュプリンガー・ジャパン株式会社.）

［22］ Wickham, H. and G. Grolemund（2016）*R for Data Science*, O'Reilly.（黒川利明 訳，（2017）『R ではじめるデータサイエンス』，オライリージャパン.）

［23］ Xie, Y.（2015）*Dynamic Documents with R and knitr, Second Edition*, CRC Press.

謝辞

東京大学の宮本大輔准教授にはラングリングのための mdx 環境の整備を，そして Moody's 社の増田 歩氏にはデータの抽出や詳細な情報を提供して頂いた．ここに感謝の意を表する．

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：「共有価値創造（CSV）のための社会環境会計の構築」（2019年～2022年），課題番号：19K02006

学際大規模情報基盤共同利用・共同研究点（JHPCN）課題：「財務ビッグデータの可視化と統計モデリング」（2017年度～2022年度），課題番号：jh171002-NWJ, jh181001-NWJ, jh191002-NWJ, jh201003-NWJ, jh211001-NWJ, jh221001

統計数理研究所 2022年度統計数理研究所公募型共同利用（一般研究 2）課題：「財務ビッグデータの時空間分析と可視化に関する研究」，課題番号：2022-ISMCRP-2030

統計数理研究所 2022年度統計数理研究所公募型共同利用（共同研究集会）課題：「データ解析環境 R の整備と利用」，課題番号：2022-ISMCRP-5003

関西学院大学：図書館図書費 B，研究設備費（III），個人研究費

付録 A　コンピュータ環境

mdx 環境

本研究を行うために，東京大学柏キャンパスに敷設された mdx 環境を利用した．各ノードの簡単な仕様を以下に与える[25]：

CPU ノード：

OS: Ubuntu 20.04

CPU: Intel® Xeon® Platinum 8368 Processor (38core, 2.4 GHz)× 2 個

CPU Cores: 152（＝38×2×2）

Main Memory: 229.78 GB

GPU ノード：

OS: Ubuntu 20.04

CPU Cores: 18

Main Memory: 57.6 GB

GPU Unit and Memory: NVIDIA® Tesla® A100, 40 GB

ストレージ：lustre（高速内部ストレージ（DDN ES400NVX)：8 TB，大容量内部ストレージ（DDN ES7990X)：8 TB）

ローカル環境

Machine: MacBook Pro 2021

OS: macOS Monterey (12.6)

CPU: Apple M1 Max (arm64), 10 Cores

Main Memory: 64 GB

Machine: Mac Studio 2022

OS: macOS Monterey (12.6)

CPU: Apple M1 Ultra (arm64), 20 Cores

25）今回利用しているノードは，ログインノード (1)，CPU ノード (3)，GPU ノード (1) の合計 5 である（ただし，() 内はノード数である)．

Main Memory: 128 GB

ソフトウェア環境

- R（R. Ihaka, R. Gentleman, R Core Team, https://www.r-project.org/）
- R Packages
  - **{arrow}**（N. Richardson, https://arrow.apache.org/docs/r/）
  - **{countrycode}**（V. Arel-Bundock, https://vincentarelbundock.github.io/countrycode/）
  - **{RPostgreSQL}**（J. Conway *et al.*, https://github.com/tomoakin/RPostgreSQL）
  - **{tidyverse}**（H. Wickham *et al.*, https://www.tidyverse.org）
- RStudio（Posit, https://www.posit.co）
- Sweave（F. Leisch, https://leisch.userweb.mwn.de/Sweave/）
- PostgreSQL（The PostgreSQL Global Development Group, https://www.postgresql.org）
- PG-Strom（K. Kaigai, https://heterodb.github.io/pg-strom/ja/）

R 関数 sessionInfo を実行することによって，本稿を執筆することに利用した R に関する環境情報を以下に与える：

sessionInfo による情報

- R version 4.2.2 (2022-10-31), aarch64-apple-darwin20
- Locale: ja_JP.UTF-8/ja_JP.UTF-8/ja_JP.UTF-8/C/ja_JP.UTF-8/ja_JP.UTF-8
- Running under: macOS Monterey 12.6
- Matrix products: default
- BLAS: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
- LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: arrow 10.0.1, countrycode 1.4.0, DBI 1.1.3, dplyr 1.0.10, forcats 0.5.2, ggplot2 3.4.0, purrr 1.0.0, readr 2.1.3, RPostgreSQL 0.7-4, stringr 1.5.0, tibble 3.1.8, tidyr 1.2.1, tidyverse 1.3.2
- Loaded via a namespace (and not attached): assertthat 0.2.1, backports 1.4.1, bit 4.0.5, bit64 4.0.5, broom 1.0.2, cellranger 1.1.0, cli 3.5.0, colorspace 2.0-3, compiler 4.2.2, crayon 1.5.2, dbplyr 2.2.1, ellipsis 0.3.2, fansi 1.0.3, fs 1.5.2, gargle 1.2.1, generics 0.1.3, glue 1.6.2, googledrive 2.0.0, googlesheets4 1.0.1, grid 4.2.2, gtable 0.3.1, haven 2.5.1, hms 1.1.2, httr 1.4.4, jsonlite 1.8.4, lifecycle 1.0.3, lubridate 1.9.0, magrittr 2.0.3, modelr 0.1.10, munsell 0.5.0, pillar 1.8.1, pkgconfig 2.0.3, R6 2.5.1, readxl 1.4.1, reprex 2.0.2, rlang 1.0.6, rvest 1.0.3, scales 1.2.1, stringi 1.7.8, tidyselect 1.2.0, timechange 0.1.1, tools 4.2.2, tzdb 0.3.0, utf8 1.2.2, vctrs 0.5.1, withr 2.5.0, xml2 1.3.3

付録 B　データ可視化のための R スクリプト

本稿で利用したデータ可視化のための R スクリプトをスクリプト11に与える[26]．

スクリプト11：makepng.R

```
library(arrow)
library(tidyverse)
library(countrycode)
x <- read_parquet("../1DW-Local/Orbis2019c-u1-sub.parquet")
x2018 <- x %>% filter(year == 2018)
#--------------------------------
# ROE
#--------------------------------
ROE2018 <- x2018 %>%
  mutate(country = countrycode(iso2c.x, origin = 'iso2c', destination = '
      country.name')) %>%
  filter(shareholders != 0) %>%
  mutate(ROE = net_income/shareholders) %>%
  select(firmid, country, iso2c.x, net_income, shareholders, ROE)
```

26）本稿で与えている可視化の結果は，ここで与えている R スクリプトを実行した結果の一部である．なお，その他の可視化から得られる知見や考察については別の機会に譲る．

```
#
p.histROE2018 <- ROE2018 %>%
  filter(ROE >= -0.5, ROE <= 0.5) %>%
  ggplot(aes(ROE)) +
  geom_histogram(binwidth = 0.02) +
  geom_rug() +
  ggtitle("Year:␣2018")
ggsave(file = "./figures/hist-ROE50-2018cu.png",
       plot = p.histROE2018,
       width = 20, height = 20, units = "cm")
#
p.histROE2018.pm10 <- ROE2018 %>%
  filter(ROE >= -10, ROE <= 10) %>%
  ggplot(aes(ROE)) +
  geom_histogram(binwidth = 0.02) +
  geom_rug() +
  ggtitle("Year:␣2018")
#
ggsave(file = "./figures/hist-ROE1000-2018cu.png",
       plot = p.histROE2018.pm10,
       width = 20, height = 20, units = "cm")
#--------------------------------
# ROE: GDP Top 10 Countries
#--------------------------------
plot.boxplot.GDP10 <- function(df = ROE2018, p = 0.5)
{
  require(ggplot2)
  require(dplyr)
  GDP10 <- c("US","CN","JP","DE","IN","GB","FR","BR","IT","CA")
  p <- df %>% filter(ROE >= -p, ROE <= p, iso2c.x %in% GDP10) %>%
    ggplot(aes(ROE, iso2c.x, color = as.factor(iso2c.x))) +
    geom_boxplot() +
    ggtitle("Year:␣2018")
  print(p)
}
#
ggsave(file = "./figures/bp-GDP10-ROE50-2018cu.png",
       plot = plot.boxplot.GDP10(),
       width = 20, height = 20, units = "cm")
#--------------------------------
# CR
#--------------------------------
CR2018 <- x2018 %>%
  filter(month == 12, assets_total > 0) %>%
  mutate(CR = shareholders/assets_total) %>%
  select(firmid, iso2c.x, shareholders, assets_total, CR)
#
p.CR2018.hist <- CR2018 %>%
  filter(CR > -10, CR < 2) %>%
  ggplot(aes(CR)) +
  geom_histogram(binwidth = 0.05) +
  ggtitle("Year:␣2018")
#
p.CR2018.density <- CR2018 %>%
```

```
  filter(CR > -10, CR < 2) %>%
  ggplot(aes(CR)) +
  stat_density() +
  ggtitle("Year: 2018")
#
ggsave(file = "./figures/hist-CR2018uc.png",
       plot = p.CR2018.hist,
       width = 20, height = 20, units = "cm")
#
ggsave(file = "./figures/density-CR2018uc.png",
       plot = p.CR2018.density,
       width = 20, height = 20, units = "cm")
#
plot.boxplot.CR.GDP10 <- function(df = CR2018, rg = c(0, 1))
{
  require(ggplot2)
  require(dplyr)
  GDP10 <- c("US","CN","JP","DE","IN","GB","FR","BR","IT","CA")
  p <- df %>% filter(iso2c.x %in% GDP10) %>%
    ggplot(aes(CR, iso2c.x)) + geom_boxplot() +
    coord_cartesian(xlim = rg) +
    ggtitle("Year: 2018")
 print(p)
}
#--------------------------------
#  Box Plot: GDP Top 10 Countries
#--------------------------------
OE2018 <- x2018 %>%
  filter(month == 12, operating_revenue > 0, employees > 0) %>%
  mutate(
    country = countrycode(iso2c.x, origin = 'iso2c', destination = 'country
        .name'),
    OE = operating_revenue/employees
  ) %>%
  select(firmid, country, iso2c.x, OE)
#
plot.boxplot.OE.GDP10 <- function(df = OE2018, rg = c(0, 10^5))
{
  require(ggplot2)
  require(dplyr)
  GDP10 <- c("US","CN","JP","DE","IN","GB","FR","BR","IT","CA")
  p <- df %>% filter(iso2c.x %in% GDP10) %>%
    ggplot(aes(OE, iso2c.x)) + geom_boxplot() +
    coord_cartesian(xlim = rg) +
    ggtitle("Year: 2018")
  print(p)
}
ggsave(file = "./figures/bp-OE-2018uc.png",
       plot = plot.boxplot.OE.GDP10(rg = c(0, 10^3)),
       width = 20, height = 20, units = "cm")
#--------------------------------
# SAP
#--------------------------------
`%nin%`<- Negate(`%in%`)
```

```
#--------------------------------
y.all <- x %>% #filter(iso2c.x != "ZW", iso2c.x != "TR") %>%
  dplyr::filter(firmid %nin% c( "EURL␣IMPOLUX␣DZ09B0543183", "JOHNSON␣&␣
      JOHNSON␣POLAND␣SP.␣Z␣O.O.␣PL006934330", "SOUF␣BOIS␣SAKER␣ET␣CIE␣
      DZ05B0542751")) %>%
  dplyr::filter(month == 12) %>%
  dplyr::filter(!is.na(interest_paid), !is.na(costs_employees), !is.na(tax)
      , !is.na(net_income)) %>%
  dplyr::group_by(year) %>%
  dplyr::summarize(
    obs = n(),
    sum.interest.paid = sum(interest_paid, na.rm = TRUE),
    sum.costs.employees = sum(costs_employees, na.rm = TRUE),
    sum.tax = sum(tax, na.rm = TRUE),
    sum.net.income = sum(net_income, na.rm = TRUE),
    sum.interest.paid.pos = ifelse(sum.interest.paid >= 0, sum.interest.
        paid, 0),
    sum.costs.employees.pos = ifelse(sum.costs.employees >= 0, sum.costs.
        employees, 0),
    sum.tax.pos = ifelse(sum.tax >= 0, sum.tax, 0),
    sum.net.income.pos = ifelse(sum.net.income >= 0, sum.net.income, 0)
  )
#
plot.SAP.all <- function(DF = y.all)
{
  library(dplyr)
  library(ggplot2)
  library(reshape)
  n.obs <- DF %>% pull(obs)
  p <- DF %>%
    data.frame() %>%
    select(year, sum.interest.paid, sum.costs.employees, sum.tax, sum.net.
        income) %>%
    melt(id.vars = c("year")) %>%
    ggplot(aes(x = year, y = value, fill = variable)) + geom_area(position
        = "fill") +
    ggtitle(paste0("All␣Country:␣Range␣of␣n.obs;␣[",min(n.obs),",␣",max(n.
        obs),"]"))
  print(p)
}
#
plot.SAP.pos.all <- function(DF = y.all)
{
  library(dplyr)
  library(ggplot2)
  library(reshape)
  n.obs <- DF %>% pull(obs)
  p <- DF %>%
    data.frame() %>%
    select(year, sum.interest.paid.pos, sum.costs.employees.pos, sum.tax.
        pos, sum.net.income.pos) %>%
    melt(id.vars = c("year")) %>%
    ggplot(aes(x = year, y = value, fill = variable)) + geom_area(position
        = "fill") +
```

```
    ggtitle(paste0("All.Country:.Range.of.n.obs;.[",min(n.obs),",.",max(n.
        obs),"]"))
  print(p)
}
#
plot.SAP.standard.all <- function(DF = y.all)
{
  library(dplyr)
  library(ggplot2)
  library(reshape)
  n.obs <- DF %>% pull(obs)
  p <- DF %>%
    data.frame() %>%
    select(year, sum.interest.paid, sum.costs.employees, sum.tax, sum.net.
        income) %>%
    melt(id.vars = c("year")) %>%
    ggplot(aes(x = year, y = value, fill = variable)) +
    geom_area() +
    ggtitle(paste0("All␣Country:␣Range␣of␣n.obs;␣[",min(n.obs),",␣",max(n.
        obs),"]"))
  print(p)
}
#
plot.SAP.pos.standard.all <- function(DF = y.all)
{
  library(dplyr)
  library(ggplot2)
  library(reshape)
  n.obs <- DF %>% pull(obs)
  p <- DF %>%
    data.frame() %>%
    select(year, sum.interest.paid.pos, sum.costs.employees.pos, sum.tax.
        pos, sum.net.income.pos) %>%
    melt(id.vars = c("year")) %>%
    ggplot(aes(x = year, y = value, fill = variable)) +
    geom_area() +
    ggtitle(paste0("All␣Country:␣Range␣of␣n.obs;␣[",min(n.obs),",␣",max(n.
        obs),"]"))
  print(p)
}
#
ggsave(file = "./figures/SAP-all.png",
       plot = plot.SAP.all(),
       width = 20, height = 10, units = "cm")
ggsave(file = "./figures/SAP-standard-all.png",
       plot = plot.SAP.standard.all(),
       width = 20, height = 10, units = "cm")
#--------------------------------
# SAP
#--------------------------------
y <- x %>%
  filter(firmid %nin% c( "EURL␣IMPOLUX␣DZ09B0543183", "JOHNSON␣&␣JOHNSON␣
      POLAND␣SP.␣Z␣O.O.␣PL006934330", "SOUF␣BOIS␣SAKER␣ET␣CIE␣DZ05B0542751")
      )  %>%
```

```
  dplyr::filter(month == 12) %>%
  dplyr::group_by(year, iso2c.x) %>%
  dplyr::filter(!is.na(interest_paid), !is.na(costs_employees), !is.na(tax)
      , !is.na(net_income)) %>%
  dplyr::group_by(iso2c.x, year) %>%
  dplyr::summarize(
    obs = n(),
    sum.interest.paid = sum(interest_paid, na.rm = TRUE),
    sum.costs.employees = sum(costs_employees, na.rm = TRUE),
    sum.tax = sum(tax, na.rm = TRUE),
    sum.net.income = sum(net_income, na.rm = TRUE),
    sum.interest.paid.pos = ifelse(sum.interest.paid >= 0, sum.interest.
        paid, 0),
    sum.costs.employees.pos = ifelse(sum.costs.employees >= 0, sum.costs.
        employees, 0),
    sum.tax.pos = ifelse(sum.tax >= 0, sum.tax, 0),
    sum.net.income.pos = ifelse(sum.net.income >= 0, sum.net.income, 0)
  ) %>%
  arrange(iso2c.x, year)
#
plot.SAP <- function(country = "JP", DF = y)
{
  library(dplyr)
  library(ggplot2)
  library(reshape)
  df <- DF %>%
    filter(iso2c.x == country)
  n.obs <- df %>% pull(obs)
  p <- df %>%
    data.frame() %>%
    select(year, sum.interest.paid, sum.costs.employees, sum.tax, sum.net.
        income) %>%
    melt(id.vars = c("year")) %>%
    ggplot(aes(x = year, y = value, fill = variable)) +
    geom_area(position = "fill") +
    ggtitle(paste0("Country:␣", country, ";␣Range␣of␣n.obs;␣[",min(n.obs),"
        ,␣",max(n.obs),"]"))
  print(p)
}
#
plot.SAP.standard <- function(country = "JP", DF = y)
{
  library(dplyr)
  library(ggplot2)
  library(reshape)
  df <- DF %>%
    filter(iso2c.x == country)
  n.obs <- df %>% pull(obs)
  p <- df %>%
    data.frame() %>%
    select(year, sum.interest.paid, sum.costs.employees, sum.tax, sum.net.
        income) %>%
    melt(id.vars = c("year")) %>%
    ggplot(aes(x = year, y = value, fill = variable)) +
```

```
260     geom_area() +
261     ggtitle(paste0("Country:␣", country, ";␣Range␣of␣n.obs;␣[",min(n.obs),"
            ,␣",max(n.obs),"]"))
262   print(p)
263 }
264 #
265 plot.SAP.pos <- function(country = "JP", DF = y)
266 {
267   library(dplyr)
268   library(ggplot2)
269   library(reshape)
270   df <- DF %>%
271     filter(iso2c.x == country)
272   n.obs <- df %>% pull(obs)
273   p <- df %>%
274     data.frame() %>%
275     select(year, sum.interest.paid.pos, sum.costs.employees.pos, sum.tax.
            pos, sum.net.income.pos) %>%
276     melt(id.vars = c("year")) %>%
277     ggplot(aes(x = year, y = value, fill = variable)) +
278     geom_area(position = "fill") +
279     ggtitle(paste0("Country:␣", country, ";␣Range␣of␣n.obs;␣[",min(n.obs),"
            ,␣",max(n.obs),"]"))
280   print(p)
281 }
282 plot.SAP.pos.standard <- function(country = "JP", DF = y)
283 {
284   library(dplyr)
285   library(ggplot2)
286   library(reshape)
287   df <- DF %>%
288     filter(iso2c.x == country)
289   n.obs <- df %>% pull(obs)
290   p <- df %>%
291     data.frame() %>%
292     select(year, sum.interest.paid.pos, sum.costs.employees.pos, sum.tax.
            pos, sum.net.income.pos) %>%
293     melt(id.vars = c("year")) %>%
294     ggplot(aes(x = year, y = value, fill = variable)) +
295     geom_area() +
296     ggtitle(paste0("Country:␣", country, ";␣Range␣of␣n.obs;␣[",min(n.obs),"
            ,␣",max(n.obs),"]"))
297   print(p)
298 }
299 #
300 countries <- c("US", "JP", "DE", "FR", "GB", "IN", "RU", "ID", "IR")
301 #
302 for(i in countries)
303 {
304  ggsave(file = paste0("./figures/SAP-",i,".png"),
305         plot = plot.SAP(country = i),
306         width = 20, height = 10, units = "cm")
307 }
308 #
```

```
for(i in countries)
{
  ggsave(file = paste0("./figures/SAP-",i,"-standard.png"),
         plot = plot.SAP.standard(country = i),
         width = 20, height = 10, units = "cm")
}
#
for(i in countries)
{
  ggsave(file = paste0("./figures/SAP-pos-",i,".png"),
         plot = plot.SAP.pos(country = i),
         width = 20, height = 10, units = "cm")
}
#
for(i in countries)
{
  ggsave(file = paste0("./figures/SAP-pos-standard-",i,".png"),
         plot = plot.SAP.pos.standard(country = i),
         width = 20, height = 10, units = "cm")
}
#
ROA.ETR.png <- function(df, YEAR)
{
  require(ggplot2)
  require(dplyr)
  p <- df %>%
    filter(year == YEAR) %>%
    select(pl_before_tax, tax, assets_total) %>%
    filter(pl_before_tax > 0) %>%
    mutate(ETR = tax/pl_before_tax,
           ROA = pl_before_tax/assets_total) %>%
    ggplot(aes(ETR, ROA)) +
    geom_point(size = 0.01, alpha = 0.01) +
    xlim(-1, 1) + ylim(0, 1) +
    labs(title = YEAR)
    ggsave(paste0("./figures/ROA-ETR-", YEAR, ".png"),
      print(p), width = 20, height = 20, units = "cm")
}
for(i in 2009:2018) ROA.ETR.png(x, i)
#---------------------------------------------------------------
# Scatter Plot: (log(assets_total), log(operating_revenue))
#---------------------------------------------------------------
SP.png <- function(df, YEAR)
{
  require(ggplot2)
  require(dplyr)
  p <- df %>%
    filter(year == YEAR) %>%
    select(assets_total, operating_revenue) %>%
      ggplot(aes(log(assets_total), log(operating_revenue))) +
      geom_point(size = 0.002, alpha = 0.01) +
    ggtitle(paste("Year:", YEAR))
    ggsave(paste0("./figures/SP-log-", YEAR, ".png"),
      print(p), width = 20, height = 20, units = "cm")
```

```
}
for(i in 2009:2018) SP.png(x, i)
```

付録 C　動的文書生成と再現可能研究

C. 1 動的文書生成

本稿は，Sweave（cf. Leisch（2002））と make コマンド（cf. Mecklenburg（2005））及びシェルスクリプトを利用し，動的に文書を生成することによって作成している．具体的には，working/3paper ディレクトリ（図 6 参照）にあるシェル・スクリプト・ファイル makepaper.sh（スクリプト12）を実行することによって，本稿のソースファイル（Sweave ファイル）DataWrangling-Visualization-OrbisData.Rnw がシェル・スクリプト・ファイル Sweave-utf8.engine によって処理され，結果として本稿自体の PDF ファイル DataWrangling-Visualization-OrbisData.pdf が出力される．

スクリプト12：シェル・スクリプト・ファイル：makepaper.sh

```
#!/bin/sh
~/Library/TeXShop/Engines/Sweave-utf8.engine DataWrangling-Visualization-
    OrbisData.Rnw
```

シェル・スクリプト・ファイル Sweave-utf8.engine のソースコードは，スクリプト13を参照されたい．

スクリプト13：Sweave による処理と LaTeX によるコンパイルを実行するシェル・スクリプト・ファイル Sweave-utf8.engine

```
#!/bin/bash
export LANG=ja_JP.UTF-8 # この設定は重要!
export PATH=$PATH:/Library/TeX/texbin:/Library/Frameworks/R.framework/
    Resources/bin/
R CMD Sweave --encoding="utf8" "$1"
filename=${1%.*}
ptex2pdf -l -ot "-synctex=1␣-file-line-error" "$filename"
ptex2pdf -l -ot "-synctex=1␣-file-line-error" "$filename"
```

Sweave-utf8.engine ファイルの 4 行目で Rnw（R noweb）ファイルが

Sweave によって処理され，得られた LaTeX ファイルを 6，7 行目でコンパイルしている．なお，それ以外の行は環境設定に関するものである．

スクリプト14：make ファイル：working/3paper/Makefile

```
1 paper:
2       date > start-paper.txt
3       /bin/bash makepaper.sh
4       date > end-paper.txt
5 clean:
6       rm -r 0PDA-Osiris2021.tex *.log *.dvi *.aux *.out *.toc
7       rm 0PDA-Osiris2021-*.png
```

動的文書生成を実際に行うためには，working/3paper ディレクトリ（図 6 参照）におけるファイル Makefile（ファイル14）に記載されたターゲット paper に対して make コマンドを実行することによって行う．

make コマンドによるターゲット merge の実行

```
% make paper
```

この処理にかかる時間を計測した結果は以下のようなものである[27]：

make コマンドによるターゲット paper の実行時間

```
% cat start-paper.txt
2022 年 12 月 25 日 日曜日 18 時 59 分 22 秒 JST
% cat end-paper.txt
2022 年 12 月 25 日 日曜日 18 時 59 分 53 秒 JST
```

この結果から，31秒である．

動的文書生成における文書ファイル，画像ファイルの流れと，それを処理するスクリプトファイルの流れの対応を図15に与える．

27）Mac Studio 2022 のもとで計測

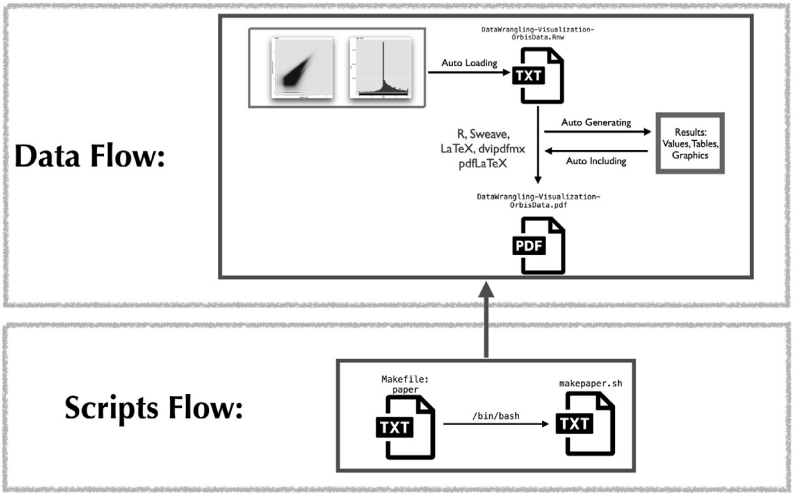

図15：動的文書生成における文書ファイル DataWrangling-Visualization-OrbisData.Rnw，画像ファイルの流れと，それを処理するスクリプトファイルの流れの対応

C. 2　再現可能研究

本研究の全工程を再現可能とするための方策としては，working ディレクトリ（図 6 参照）の直下にあるファイル Makefile（ファイル15）に記載されたターゲット all に対して make コマンドを実行することによって行う．

スクリプト15：make ファイル：working/Makefile

```
1 all:
2       date > start-all.txt
3       (cd 1DW-Local; make all);
4       (cd 2Visualization; make png)
5       (cd 3paper; make paper)
6       date > end-all.txt
```

working/Makefile ファイル（スクリプト15）で与えられているターゲット all の役割とディレクトリ構成（図 6 ）の対応を図16に与える．

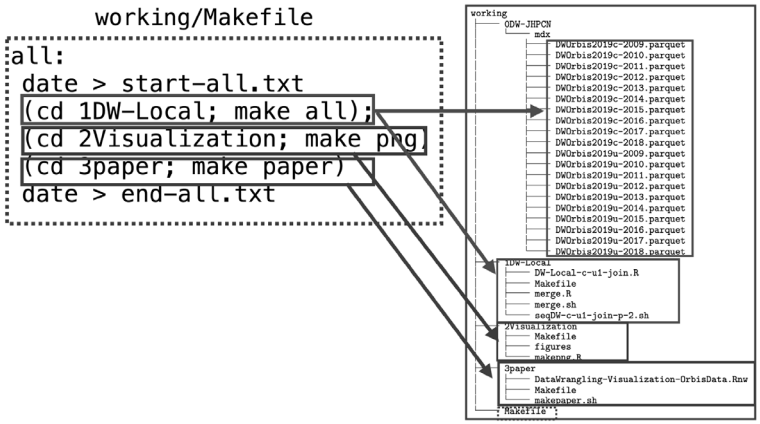

図16：working/Makefile ファイル（スクリプト15）で与えられているターゲット all の役割とディレクトリ構成（図 6 ）の対応

最終的に，working ディレクトリ（図 6 参照）において，ターゲット all に対して make コマンドを実行することによって，本稿を作成するための全工程が自動実行されることによって完全に再現される．

make コマンドによるターゲット all の実行

```
% make all
```

この全処理にかかる時間を計測した結果は以下のようなものである：

make コマンドによるターゲット all の実行時間

```
% cat start-all.txt
2022 年 12 月 25 日 日曜日 14 時 10 分 57 秒 JST
% cat end-all.txt
2022 年 12 月 25 日 日曜日 18 時 59 分 53 秒 JST
```

この結果から，全工程を処理する合計時間は 4 時間49分である[28)]．

28）この結果は，Mac Studio 2022 で計測したものである．

Makefile ファイル（スクリプト14）のターゲット all についての make コマンドによる実行によるデータファイルと文書ファイル，スクリプトファイルの流れ，及びスクリプトファイルから実行されるデータファイルと文書ファイルへの全処理の対応を図17に与える．

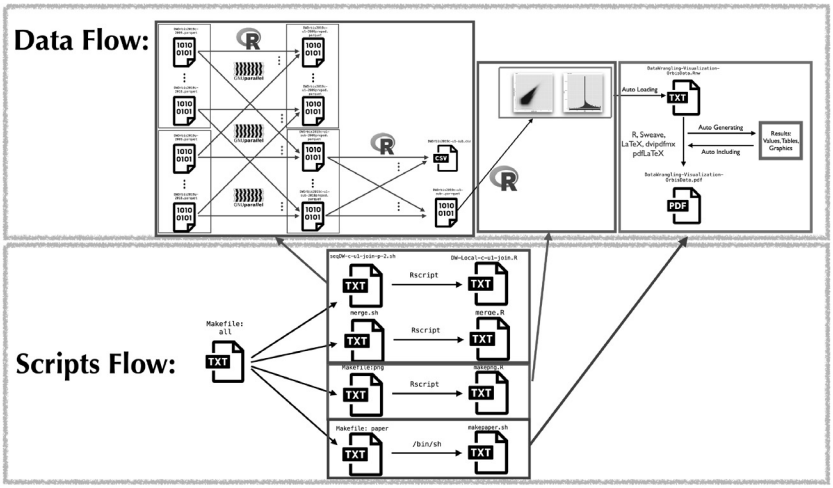

図17：Makefile ファイル（スクリプト14）のターゲット all についての make コマンドによる実行によるデータファイルと文書ファイル，スクリプトファイルの流れ，及びスクリプトファイルから実行されるデータファイルと文書ファイルへの全処理の対応

なお，本研究は，Peng（2011）による「再現可能性スペクトル」（reproducibility spectrum）に照らすと全工程を自動実行して再現性を確保しているため，「ゴールドスタンダード」に属すものと思われる（Peng（2011）の Fig. 1 を参照）．