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―

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もとに，世界の非上場企業に関する規模の大きな財務データのラングリングに関して述べる．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：財務データ（Financial 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では，これまで Bureau van Dijk[1]（ビューロー・ヴァン・ダイク，以下 BvD と略）社のデータベース Osiris から抽出された世界の「上場企業」（listed firms）に関する財務データを前処理，ラングリング[2]

1）BvD Web Page: https://www.bvdinfo.com/en-gb/

2）本稿では，データベースから抽出された粗データのファイルをコンピュータ（ソフトウェア）で利用できるファイル形式に変換する工程を「前処理」（preprocessing）と呼び，その後ファイルを R に読み込み，実際にデータ解析が行えるオブジェクト形式に変換する工程を，Wickham and Grolemund（2016）にならって「データラングリング」（data wrangling）と呼ぶことにする．なお，この工程は，データを扱う分野・業種・ニュアンス等の相違から，「抽出・変換・読み込み」（Extract, Transform, Load: ETL），「データパイプライン」（data pipeline），「データクレンジング」（data cleansing），「データクリーニング」（data cleaning），「データ操作」（data manipulation）

したものに対して要約，可視化，統計モデリングを行い探索的データ解析（Exploratory Data Analysis: EDA）（cf. Tukey（1977））を実行してきた（cf. 地道（2017-a, b），地道（2018-a, b），地道（2021-b），地道ら（2017, 2018），Jimichi *et al.*（2018），地道ら（2019, 2020-b），地道，阪（2021），地道ら（2021-a））．

本稿では，データ解析の対象をさらに拡大し，世界の「非上場企業」（unlisted firms and delisted firms）に対するデータラングリングについて検討する．具体的には，BvD 社のデータベース Orbis から抽出されたデータを利用し（cf. 地道（2020-a, b），地道ら（2020-a, b, c）），連結主体で抽出したデータと単体主体で抽出したデータをラングリングしたのち結合する．なお，本稿で行うラングリングには，主にデータ解析環境 R[3] を利用している．

本稿の構成は以下のようなものである．まず，本稿で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（II 節），今回扱うデータの属性について述べた後，東京大学情報基盤センター[4] のコンピュータ環境において実際に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，今後の研究に利用できる形式に変換（ラングリング）する工程を詳述し（IV 節），最終的に得られたデータに関する情報を要約と可視化によって調べる（V 節）．最後に，今後の課題などを与える（VI 節）．なお，本研究で行われるラングリングの全工程は，`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，本稿は Sweave（cf. Leisch（2002））による動的文書生成（dynamic documents）によって作成されている．この仕様によって，再現可能性[5] を確保している．なお，付録 A には，ラングリングに利用したコンピュータ環境に関する情報を与えている[6]．

等と呼ばれることもある（cf. 地道（2018-a））．

3） `https://www.r-project.org`

4） `https://www.itc.u-tokyo.ac.jp/`

5） 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

BvD 社のデータベース Orbis（2019年12月版）には，200カ国を超える世界における財務情報が入手可能な全企業を対象に収集されたデータが国際比較可能な統一のフォームで納められている[7]．これらの財務諸表は，連結（Consolidated）財務諸表として Orbis に収録されているものもあれば単体（Unconsolidated）財務諸表として収録されているものもあるため，本研究では，このデータベースからデータセットとして，主要財務情報を最長10年分抽出した以下のようなものを利用する：

(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）

データセットのファイルは，１個のサイズが５GB 程度の27個の TSV ファイル（２セット）からなり，表１，表２には，それぞれ，データセットの仕様とサイズを与えている．

表１：Orbis データセット：仕様

データセット名	版（年月）	データベース	上場情報	抽出主体	抽出期間	抽出指標数	国数（自治領含む）
DS-Orbis-C-2019	2019年12月	Orbis	上場・非上場	連結優先	10年	85	205
DS-Orbis-U-2019	2019年12月	Orbis	上場・非上場	単体優先	10年	85	203

表２：Orbis データセット：サイズ

データセット名	社数	総行数	粗データファイル	トータルサイズ
DS-Orbis-C-2019	26,353,934	289,893,274	SJ_OB_2019_1_1000000.asc,..., SJ_OB_2019_26000001_26353934.asc	約 142 GB
DS-Orbis-U-2019	26,352,382	289,876,202	SJ_OB_2019_U_1_1000000.asc,..., SJ_OB_2019_U_26000001_26352382.asc	約 142 GB

地道（2020-a）では，2018年に抽出された Orbis データセットを GNU parallel[8]（cf. Tange（2018））を用いて並列化することによって効率的に前処理

6)　ローカル環境での処理時間の計測は，iMac 2017 で行っている．
7)　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．

することが議論されており，地道ら（2020-a, b, c）では，2020年に新しく抽出された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，同様の方法で処理し，表3で与えられるCSVファイルを得ている．

表3：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

なお，抽出時の指標数が85に対して，処理後のファイルの列数が88になっているのは，前処理の工程で社名（`firm`），社名＋BvD ID（`firmID`），会計年度（`year`）を追加したためである．

地道（2020-b）では，2018年に抽出されたOrbisデータセットを前処理後，東京大学情報基盤センターに設置された専有利用型リアルタイムデータ解析ノード（以後，FENNELと略す）とGPGPU環境でデータベース管理システムPostgreSQL[9]と**PG-Strom**[10]を利用することによって，ラングリングを高速化することが検討されている．さらに，地道ら（2020-a, b, c）では，2019年12月版Orbisから抽出・前処理されたデータセットにもとづくCSVファイル`firmfinBC2019.csv`，`firmfinBU2019.csv`（表3参照）を東京大学のFENNEL環境でPostgreSQLを利用してデータベース化したものから，**PG-Strom**を利用して実際にラングリングし，データ可視化を行うことについて議論されている[11]．

次節では，東京大学のFENNEL環境のもとでデータを抽出することについて，データの属性とともに詳述する．

8） `https://www.gnu.org/software/parallel/`
9） `https://www.postgresql.org`
10） `https://heterodb.github.io/pg-strom/ja/`
11） 地道（2021-d）では，2019年12月版Orbisから抽出されたデータセットの約1％にあたる26万社の財務データを**PG-Strom**を利用してラングリングし，データ抽出システムSKWAD（cf. 地道（2021-a, b, c））に実装することについて議論されている．

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（財務情報・会計情報）は以下のようなものである：

`firmid`: 企業名＋BvD ID（例: "ITOHAM YONEKYU HOLDINGS INC. JP 6011001110293"; 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）

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

東京大学の FENNEL 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 1 に与える．

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

図 1 ：FENNEL 環境用スクリプトファイルのディレクトリ構成

FENNEL 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[12]．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 DWOrbis2019 の Makefile におけるターゲット DW-c を利用する（図 1，及び，スクリプト 1 参照）.

スクリプト 1 ：Makfile: ターゲット DW-c

```
1  DW-c:
2        date > start-DW-c.txt
3        Rscript DW-Orbis2019c-fennel-kg-01.R
4        date > end-DW-c.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-fennel-kg-01.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-fennel-kg-01.R

```
1  library(RPostgreSQL)
2  library(arrow)
3  drv <- dbDriver("PostgreSQL")
4  con <- dbConnect(drv, host="gpu-kg-01", user= ********, password=********,
       dbname="jhpcn")
5  # Data Wrangling from orbis2019c
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␣FROM␣orbis2019c␣
       WHERE␣(cons␣=␣'C1'␣OR␣cons␣=␣'C2')␣AND␣year␣=␣",i)
9  x <- fetch(dbSendQuery(con, sql), n = -1)
10 # dump parquet
11 write_parquet(x, sink = paste0("DWOrbis2019c-",i,".parquet"))
12 rm(sql, x)
13 }
```

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

12）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.

（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[13] ファイル DWOrbis2019c-2009.parquet 〜 DWOrbis2019c-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2に与える．

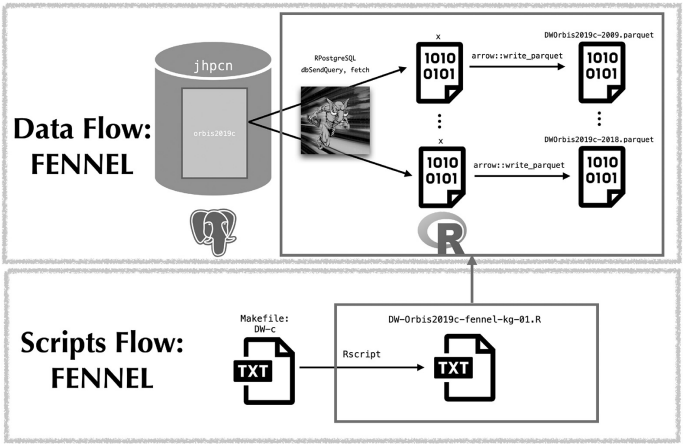

図2：FENNEL 環境のもとでの Orbis データ（連結ベース）抽出にともなうスクリプトファイルとデータファイルの流れ

以上の処理は，FENNEL 環境におけるディレクトリ DWOrbis2019（図1参照）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．

ターゲット DW-c の実行

```
$ make DW-c
```

13）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．

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.txt
2021 年 10 月 4 日 月曜日 13:00:41 JST
% cat end-DW-c.txt
2021 年 10 月 4 日 月曜日 13:11:50 JST
```

この結果から，11分 9 秒であることがわかる[14]．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 DWOrbis2019c-20xx.parquet に関する情報

```
% ls -la DWOrbis2019c-20*
-rw-r--r-- 1 masa masa 145200809 10 月 4 13:01 DWOrbis2019c-2009.parquet
-rw-r--r-- 1 masa masa 146295370 10 月 4 13:03 DWOrbis2019c-2010.parquet
-rw-r--r-- 1 masa masa 147046095 10 月 4 13:04 DWOrbis2019c-2011.parquet
-rw-r--r-- 1 masa masa 147928193 10 月 4 13:05 DWOrbis2019c-2012.parquet
-rw-r--r-- 1 masa masa 156193906 10 月 4 13:06 DWOrbis2019c-2013.parquet
-rw-r--r-- 1 masa masa 164452625 10 月 4 13:07 DWOrbis2019c-2014.parquet
-rw-r--r-- 1 masa masa 167645766 10 月 4 13:08 DWOrbis2019c-2015.parquet
-rw-r--r-- 1 masa masa 168174530 10 月 4 13:09 DWOrbis2019c-2016.parquet
-rw-r--r-- 1 masa masa 169257520 10 月 4 13:10 DWOrbis2019c-2017.parquet
-rw-r--r-- 1 masa masa 162249845 10 月 4 13:11 DWOrbis2019c-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 DWOrbis2019 の Makefile におけるターゲット DW-u を利用する（図 1，及び，スクリプト 3 参照）．

スクリプト 3 ：Makfile: ターゲット DW-u

```
1 DW-u:
2 	date > start-DW-u.txt
3 	Rscript DW-Orbis2019u-fennel-kg-01.R
4 	date > end-DW-u.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

14) 今回，**PG-Strom** を利用した場合しか計測していないが，このような短時間で処理されるのは，**PG-Strom** と FENNEL 環境の恩恵である．

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-fennel-kg-01.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-fennel-kg-01.R

```
1  library(RPostgreSQL)
2  library(arrow)
3  drv <- dbDriver("PostgreSQL")
4  con <- dbConnect(drv, host="gpu-kg-01", 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␣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3に与える．

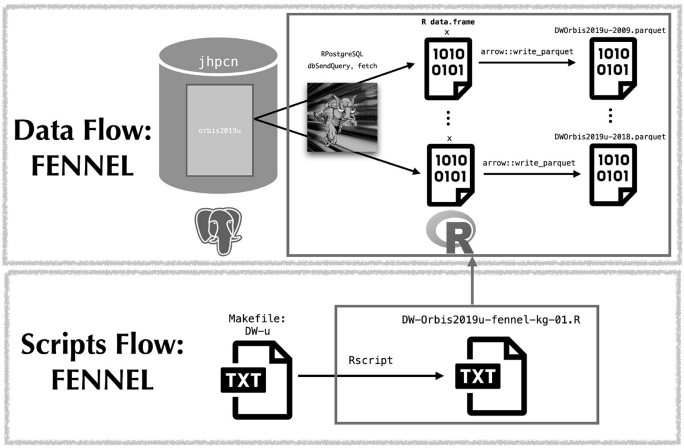

図 3 ：FENNEL 環境のもとでの Orbis データ（単体ベース）抽出にともなうスクリプトファイルとデータファイルの流れ

以上の処理は，FENNEL 環境におけるディレクトリ DWOrbis2019（図 1 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．

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.txt
2021 年 10 月 3 日 日曜日 15:57:11 JST
% cat end-DW-u.txt
2021 年 10 月 3 日 日曜日 16:40:34 JST
```

この結果から，43分23秒であることがわかる[15]．

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 DWOrbis2019u-20xx.parquet に関する情報

```
% ls -la DWOrbis2019u-20*
-rw-r--r-- 1 masa masa 1648546592 10月 3 16:03 DWOrbis2019u-2009.parquet
-rw-r--r-- 1 masa masa 1661934976 10月 3 16:07 DWOrbis2019u-2010.parquet
-rw-r--r-- 1 masa masa 1758045353 10月 3 16:11 DWOrbis2019u-2011.parquet
-rw-r--r-- 1 masa masa 1798218393 10月 3 16:15 DWOrbis2019u-2012.parquet
-rw-r--r-- 1 masa masa 1848457970 10月 3 16:20 DWOrbis2019u-2013.parquet
-rw-r--r-- 1 masa masa 1889544874 10月 3 16:23 DWOrbis2019u-2014.parquet
-rw-r--r-- 1 masa masa 1904674490 10月 3 16:27 DWOrbis2019u-2015.parquet
-rw-r--r-- 1 masa masa 1943625853 10月 3 16:32 DWOrbis2019u-2016.parquet
-rw-r--r-- 1 masa masa 2056603117 10月 3 16:36 DWOrbis2019u-2017.parquet
-rw-r--r-- 1 masa masa 1823171551 10月 3 16:40 DWOrbis2019u-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[16]，FENNEL 環境で抽出されたデータファイル DWOrbis2019c-2009.parquet 〜 DWOrbis2019c-2018.parquet（連結ベース），DWOrbis2019u-2009.parquet 〜 DWOrbis2019u-2009.parquet（単体ベース）を sftp コマンドでローカルに転送した（図 4 ）.

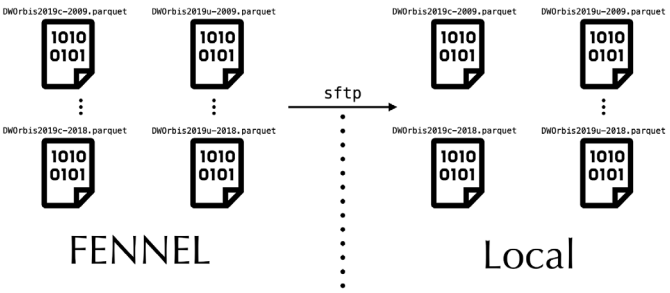

図 4 ：sftp コマンドによる FENNEL 環境からローカル環境への Orbis データファイルの転送

15）単体ベースのデータ抽出にはある程度の時間がかかっているようにみえるが，連結ベースの場合に比べて，抽出するデータの大きさが10倍程度の規模となっていることから，この場合も **PG-Strom** と FENNEL 環境の恩恵を受けていることがわかる.

16）当然，この後で述べるデータラングリングを FENNEL 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，FENNEL 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

1. データ利用における方針

非上場企業の財務データの利用に際しては以下の方針にしたがった：

（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[17]．

（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（`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．

2. データがもつ問題とその解決法

本研究の目的の一つは，非上場企業の財務データの国別情報にもとづく可視化を行うことである．これまでに，本稿で扱っているデータを調査した結果として国別情報とデータの提供元となるインフォメーションプロバイダに関して以下のような問題があることがわかった：

（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

17）本研究を開始した当初は，単体主体で抽出した企業の中から非上場のもののみのデータを利用することを考えていたが，ヴェトナム（Vietnam）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も非上場企業のものを抽出している．

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[18] 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．

これらの問題に対して，以下のような方法で対処した：

（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て，重複を排除する．

3. ラングリング手順

I 節でも述べたがデータを 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と呼ぶが，この工程には，データの読み込み

18）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`）．

(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．

ここでは，以下のように段階的に処理することによってラングリングを実行した．

(DW-L-1) 2018年度分の連結ベースデータをラングリングし，結果をファイル `Orbis2019c-2018preped.parquet` に出力

(DW-L-2) 2018年度分の単体ベースデータをラングリングし，結果をファイル `Orbis2019u-2018preped.parquet` に出力

(DW-L-3) 処理 (DW-L-1)，(DW-L-2) によって得られた2018年度分の連結・単体ベースの parquet ファイルをラングリングし，結果を結合しファイル `Orbis2019c-u1-2018preped.parquet` に出力

ローカル環境において，これらの処理を行うためのスクリプトファイルを格納したディレクトリ構成を図 5 に与える．

```
DW-Local-Orbis2019
├──── 0FENNEL
│        ├──── DWOrbis2019c-2018.parquet
│        └──── DWOrbis2019u-2018.parquet
├──── 1Cons
│        ├──── DW-Local-c.R
│        └──── Makefile
├──── 1Uncons
│        ├──── DW-Local-u.R
│        └──── Makefile
├──── 2Cons-Uncons
│        ├──── join.R
│        └──── Makefile
└──── Makefile
```

図 5 ：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

以下に，処理段階 (DW-L-1)，(DW-L-2)，(DW-L-3) の説明を与える．

連結ベースデータのラングリングと結果の出力

ディレクトリ `1Cons`（図 5 ）において連結ベースデータのラングリング

を行う．このディレクトリに配置されたMakefile（スクリプト5）にしたがって処理が行われる．

スクリプト5：Makfile: ターゲット DW-Local-c

```
DW-Local-c:
	date > start-DW-Local-c.txt
	Rscript DW-Local-c.R
	date > end-DW-local-c.txt
```

スクリプト5には，1行目でターゲットDW-Local-cが定義されており，(2, 4）行目は処理時間の計測を行うための設定である．なお，実際の処理は3行目によって行われ，RscriptコマンドでRスクリプトファイルDW-Local-c.R（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：DW-Local-c.R

```
library(tidyverse)
library(arrow)
library(stringr)
library(countrycode)
#
x <- read_parquet("../0FENNEL/DWOrbis2019c-2018.parquet") %>% tibble()
x2018c <- x %>% filter(listed != "Listed")
#
pattern.p <- x2018c %>% pull(country) %>% str_detect("\\(")
x2018c.p <- x2018c[pattern.p,]
x2018c.np <- x2018c[!pattern.p,]
#
y2018c <- bind_rows(
  x2018c.np %>% filter(!is.na(country)) %>% mutate(country.x = country),
  x2018c.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))
#
y2018c.sel <- y2018c %>% filter(iso2c.x == iso2c.y)
#
list.country.infoprov <- function(df = y2018c.sel)
{
  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)
```

```
31   }
32   names(info.list) <- countrycode(ISO2C, origin = 'iso2c', destination = '
        country.name')
33   info.list
34 }
35 all.list.cip <- list.country.infoprov()
36 cip <- function(x = all.list.cip)
37 {
38   require(countrycode)
39   require(dplyr)
40   require(tibble)
41   n <- length(x)
42   country <- names(x)
43   top.ip <- rep(0, n)
44   for(i in 1:n) top.ip[i] <- names(all.list.cip[i][[1]])[1]
45   iso2c <- countrycode(country, origin = 'country.name', destination = '
        iso2c')
46   x <- tibble(country, iso2c, top.ip) %>% na.omit() # for Kosovo
47   x
48 }
49 cip.frame <- cip()
50 #
51 z2018c <- y2018c.sel %>%
52   filter(country.x != "Kosovo", country.x != "Namibia") %>%
53   left_join(cip.frame[, c(2, 3)], by = c("iso2c.x" = "iso2c"))
54 #
55 z2018c.top.ip <- z2018c %>% filter(infoprov == top.ip)
56 #
57 write_parquet(z2018c.top.ip, "Orbis2019c-2018preped.parquet")
```

スクリプト6は以下のような処理を行う（なお，適宜，演算子 %>% を使ってパイプライン化している）：

1行目〜4行目：Rパッケージの読み込み[19)]

6行目：**arrow** パッケージに付属する関数 read_parquet を利用して2018年度分の連結ベースデータファイル（DWOrbis2019c-2018.parquet）の読み込んだものをオブジェクト x に付値

7行目：方針（Pol1）にしたがって，オブジェクト x から上場していない企業を抽出したものをオブジェクト x2018c に付値

9行目：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 に付値

10行目：オブジェクト x2018c から国別情報に自治領等（括弧 () を含むも

19）ここで読み込まれる **tidyverse** は，データラングリングのためのパッケージ群であり，オブジェクトの処理に対して必要となる便利な関数が用意されている．

の）のパターン pattern.p にしたがうものを抽出し，オブジェクト x2018c.p に付値

11行目：オブジェクト x2018c から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 x2018c.np に付値

13行目～16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 x2018c.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 x2018c.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 y2018c に付値

18行目：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 y2018c.sel に付値（対処法（Sol3）の実現）

20行目～34行目：オブジェクト y2018c.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

35行目：関数 list.country.infoprov の実行結果をオブジェクト all.list.cip に付値

36行目～48行目：オブジェクト all.list.cip 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

49行目：関数 cip の実行結果をデータ・フレーム・オブジェクト cip.frame に付値

51行目～53行目：オブジェクト y2018c.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.frame と結合（left_join）し，オブジェクト z2018c に付値

55行目：オブジェクト z2018c 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 z2018c.top.ip に付値（対処法（Sol4）の実現）

57行目：オブジェクト z2018c.top.ip をファイル Orbis2019c-2018preped.parquet（parquet 形式）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 6 に与える．

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．なお，両国ともデータが欠測しているため，実際の可視化や解析の結果に影響はない．

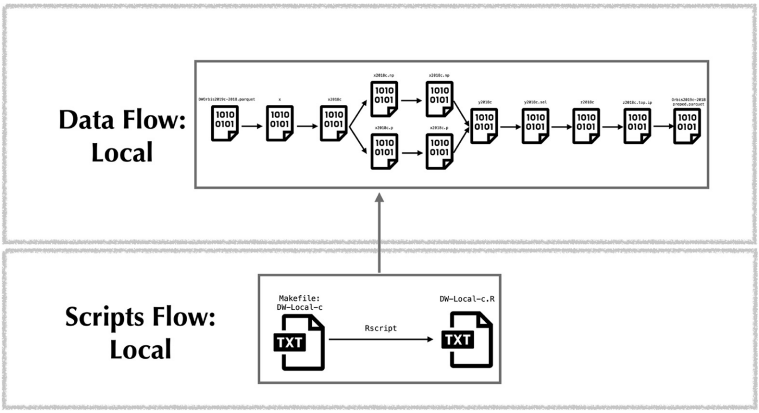

図 6 ：連結ベースデータのラングリング

以上の処理は，ディレクトリ 1Cons（図 5 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．

ターゲット DW-Local-c の実行

```
$ make DW-Local-c
```

この処理時間は，スクリプト 5 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-Local-u の処理時間の計測

```
% cat start-DW-Local-c.txt
2021 年 12 月 27 日 月曜日 22 時 51 分 46 秒 JST
% cat end-DW-local-c.txt
2021 年 12 月 27 日 月曜日 22 時 52 分 13 秒 JST
```

この結果から，27秒であることがわかる．

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 Orbis2019c-2018preped.parquet に関する情報

```
% ls -l Orbis2019c-2018preped.parquet
-rw-r--r--  1 masa  staff  91888684 12 27 22:52 Orbis2019c-2018preped.parquet
```

単体ベースデータのラングリングと結果の出力

ディレクトリ 1Uncons（図 5 参照）において単体ベースデータのラングリングを行う．このディレクトリに配置された Makefile（スクリプト 7）にしたがって処理が行われる．

スクリプト 7 ：Makfile: ターゲット DW-Local-u

```
1 DW-Local-u:
2       date > start-DW-Local-u.txt
3       Rscript DW-Local-u.R
4       date > end-DW-local-u.txt
```

スクリプト 7 には，1 行目でターゲット DW-Local-u が定義されており，(2, 4) 行目は処理時間の計測を行うための設定である．なお，実際の処理は 3 行目によって行われ，Rscript コマンドで R スクリプトファイル DW-Local-u.R（スクリプト 8）が実行されるように定義されている．

スクリプト 8 ：DW-Local-u.R

```
 1 library(tidyverse)
 2 library(arrow)
 3 library(stringr)
 4 library(countrycode)
 5 #
 6 x <- read_parquet("../0FENNEL/DWOrbis2019u-2018.parquet") %>% tibble()
 7 x2018u <- x %>% filter(listed != "Listed")
 8 #
 9 pattern.p <- x2018u %>% pull(country) %>% str_detect("\\(")
10 x2018u.p <- x2018u[pattern.p,]
11 x2018u.np <- x2018u[!pattern.p,]
12 #
13 y2018u <- bind_rows(
14   x2018u.np %>% filter(!is.na(country)) %>% mutate(country.x = country),
15   x2018u.p  %>% filter(!is.na(country)) %>% mutate(country.x = as.character
       (str_match(country, "(?<=\\().*?(?=\\))")))
16 ) %>% mutate(iso2c.x = countrycode(country.x, origin = 'country.name',
     destination = 'iso2c'), iso2c.y = substr(id, 1, 2))
17 #
18 y2018u.sel <- y2018u %>% filter(iso2c.x == iso2c.y)
```

```
19 list.country.infoprov <- function(df = y2018u.sel)
20 {
21   require(dplyr)
22   require(countrycode)
23   ISO2C <- df %>% pull(iso2c.x) %>% unique() %>% sort()
24   nc <- length(ISO2C)
25   info.list <- list()
26   for(i in 1:nc)
27   {
28     x <- df %>% filter(iso2c.x == ISO2C[i]) %>% pull(infoprov) %>% table()
         %>% sort(decreasing = TRUE) %>% list()
29     info.list <- c(info.list, x)
30   }
31   names(info.list) <- countrycode(ISO2C, origin = 'iso2c', destination = '
       country.name')
32   info.list
33 }
34 all.list.cip <- list.country.infoprov()
35 cip <- function(x = all.list.cip)
36 {
37   require(countrycode)
38   require(dplyr)
39   require(tibble)
40   n <- length(x)
41   country <- names(x)
42   top.ip <- rep(0, n)
43   for(i in 1:n) top.ip[i] <- names(all.list.cip[i][[1]])[1]
44   iso2c <- countrycode(country, origin = 'country.name', destination = '
       iso2c')
45   x <- tibble(country, iso2c, top.ip) %>% na.omit() # for Kosovo
46   x
47 }
48 cip.frame <- cip()
49 #
50 z2018u <- y2018u.sel %>%
51   filter(country.x != "Kosovo", country.x != "Namibia") %>%
52   left_join(cip.frame[, c(2, 3)], by = c("iso2c.x" = "iso2c"))
53 #
54 z2018u1 <- z2018u % >% filter(cons == "U1")
55 #
56 z2018u1.top.ip <- z2018u1 %>% filter(infoprov == top.ip)
57 #
58 write_parquet(z2018u1.top.ip, "Orbis2019u1-2018preped.parquet")
```

スクリプト 8 は，連結ベースデータのラングリングを行うスクリプト 6 において連結のコード（c）を単体（非連結）のコード（u）に置き換えることと，方針（Pol2）にしたがって単体財務情報のみを保有する企業に制限していること（54行目参照）以外は同様の処理を行っている．

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 6 に与える．

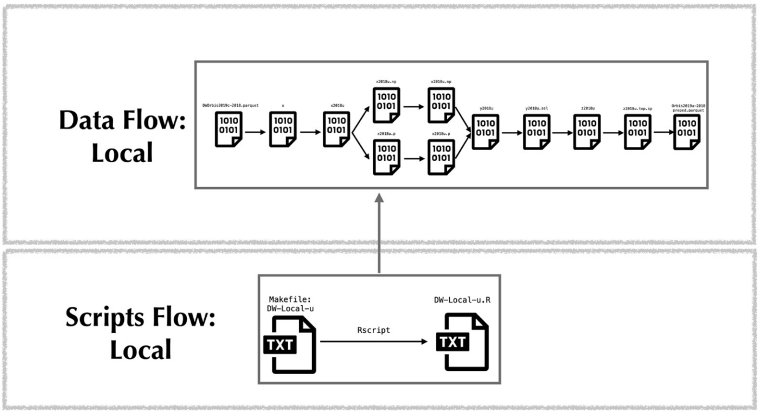

図 7：単体ベースデータのラングリング

以上の処理は，ディレクトリ 1Uncons（図 5 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．

ターゲット DW-Local-u の実行

```
$ make DW-Local-u
```

この処理時間は，スクリプト 7 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-Local-u の処理時間の計測

```
% cat start-DW-Local-u.txt
2021 年 12 月 27 日 月曜日 22 時 52 分 13 秒 JST
% cat end-DW-local-u.txt
2021 年 12 月 27 日 月曜日 22 時 58 分 32 秒 JST
```

この結果から， 6 分19秒であることがわかる．

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 Orbis2019u1-2018preped.parquet に関する情報

```
% ls -l Orbis2019u1-2018preped.parquet
-rw-r--r--  1 masa  staff  1147608520 12 27 22:58 Orbis2019u1-2018preped.parquet
```

連結ベースデータと単体ベースデータの結合と結果の出力

ディレクトリ 2Cons-Uncons（図 5 参照）において連結ベースデータと単体ベースデータの結合を行う．このディレクトリに配置された Makefile（スクリプト 9）にしたがって処理が行われる．

スクリプト 9 ：Makfile: ターゲット join

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

スクリプト 9 には，1 行目でターゲット join が定義されており，(2, 4）行目は処理時間の計測を行うための設定である．なお，実際の処理は 3 行目によって行われ，Rscript コマンドで R スクリプトファイル join.R（スクリプト10）が実行されるように定義されている．

スクリプト10：join.R

```
1  library(tidyverse)
2  library(arrow)
3  #
4  x2018c <- read_parquet("../1Cons/Orbis2019c-2018preped.parquet")
5  x2018u <- read_parquet("../1Uncons/Orbis2019u1-2018preped.parquet")
6  #
7  x2018cu <- bind_rows(x2018c, x2018u)
8  #
9  count.firms <- x2018cu %>% pull(firmid) %>% table()
10 dc.firms <- names(count.firms[count.firms >=2])
11 #
12 `%nin%` <- Negate(`%in%`)
13 #
14 x2018cu.ndc <- x2018cu %>% filter(firmid %nin% dc.firms)
15 #
16 x2018cu.dc <- x2018cu %>% filter(firmid %in% dc.firms) %>% arrange(firmid)
       %>% filter(cons == "U1")
17 #
18 y2018cu <- bind_rows(x2018cu.ndc, x2018cu.dc)
19 #
20 write_parquet(y2018cu, "Orbis2019c-u1-2018preped.parquet")
```

スクリプト10は以下のような処理を行う：

1行目～2行目：Rパッケージの読み込み

4行目：連結データファイル Orbis2019c-2018preped.parquet をディレクトリ 1Cons から読み込み，オブジェクト x2018c に付値

5行目：単体データファイル Orbis2019u1-2018preped.parquet をディレクトリ 1Uncons から読み込み，オブジェクト x2018u に付値

7行目：オブジェクト x2018c とオブジェクト x2018u を関数 bind_rows で行結合し，オブジェクト x2018cu に付値

9行目：オブジェクト x2018cu 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

10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

12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

14行目：オブジェクト x2018cu 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 x2018cu.ndc に付値

16行目：オブジェクト x2018cu 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 x2018cu.dc に付値（対処法（Sol5）の実現）

18行目：オブジェクト x2018cu.ndc とオブジェクト x2018cu.dc を行結合し，オブジェクト y2018cu に付値

20行目：オブジェクト y2018cu をファイル Orbis2019c-u1-2018preped.parquet（parquet 形式）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8に与える．

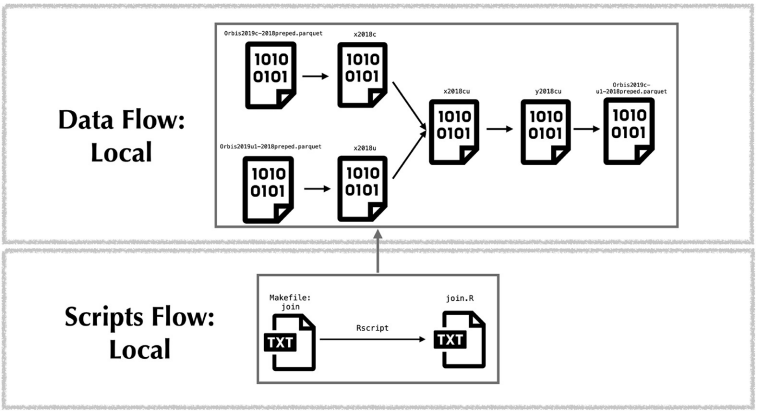

図 8 ：連結ベースデータと単体ベースデータの結合

以上の処理は，ディレクトリ 2Cons-Uncons（図 5 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．

ターゲット join の実行

```
$ make join
```

この処理時間は，スクリプト 9 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 join の処理時間の計測

```
% cat start-join.txt
2021 年 12 月 27 日 月曜日 22 時 58 分 32 秒 JST
% cat end-join.txt
2021 年 12 月 27 日 月曜日 23 時 07 分 06 秒 JST
```

この結果から， 8 分34秒であることがわかる．

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 Orbis2019c-u1-2018preped.parquet に関する情報

```
% ls -l Orbis2019c-u1-2018preped.parquet
-rw-r--r--  1 masa  staff  1252575487 12 27 23:07 Orbis2019c-u1-2018preped.parquet
```

ラングリング全体の自動実行

ここまでのラングリングの全工程を一括で処理することを考える．3段階ある処理は全て make コマンドを実行することによって行われるため，それらを順次実行するための Makefile を用意すればよい．ここでは，処理を行うトップディレクトリ DW-Local-Orbis2019（図5参照）に以下のような Makefile（スクリプト11）を用意した．

スクリプト11：Makefile: ターゲット all

```
1 all:
2 	date > start-all.txt
3 	(cd 1Cons; make DW-Local-c)
4 	(cd 1Uncons; make DW-Local-u)
5 	(cd 2Cons-Uncos; make join)
6 	date > end-all.txt
```

スクリプト11では，1行目でターゲット all が定義されており，(2, 6）行目は処理時間の計測を行うための設定である．実際の処理は，3行目で連結ベースデータのラングリングと結果のファイル出力が行われ，4行目で単体ベースデータのラングリングと結果のファイル出力が，さらに，5行目で連結ベースデータと単体ベースデータの結合と結果のファイル出力が行われる．Makefile における各指定とディレクトリ構成（図5）の対応は図9を参照されたい．

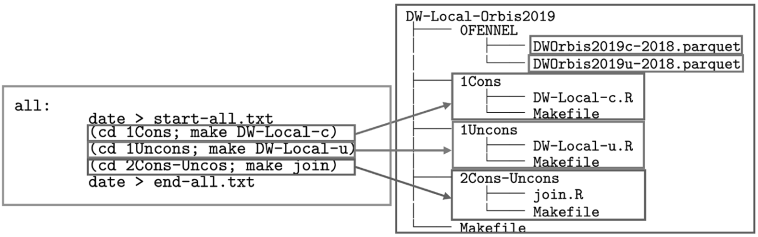

図9：Makefile の処理とディレクトリとの対応

データラングリングの全工程に関す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．

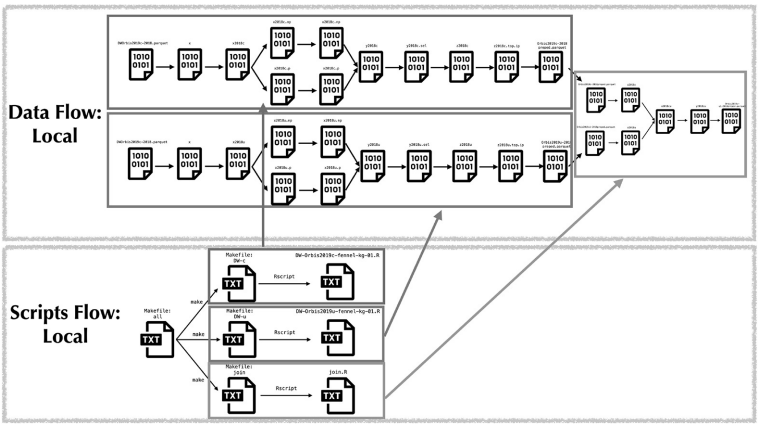

図10：全工程の自動実行

以上の処理は，ディレクトリ 1Uncons（図5参照）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．

ターゲット DW-Local-u の実行

```
$ make all
```

この処理時間は，スクリプト11における，(2, 6) 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
2021年 12月 27日 月曜日 22時 51分 46秒 JST
% cat end-all.txt
2021年 12月 27日 月曜日 23時 07分 06秒 JST
```

この結果から，15分20秒であることがわかる．

V　ラングリングされたデータに関する情報の要約と可視化

ここでは，前節で得られたラングリング結果のファイル Orbis2019c-u1-2018preped.parquet に納められたデータの要約と可視化を行う．まず，トップディレクトリ DW-Local-Orbis2019（図 5 参照）で R を起動し，以下のような入力によってファイルからデータを読み込み，その結果をオブジェクト x に付置する：

ファイル Orbis2019c-u1-2018preped.parquet の読み込みと付値

```
> library(arrow)
> x <- read_parquet("./2Cons-Uncons/Orbis2019c-u1-2018preped.parquet")
```

次に，オブジェクト x の要約を行う：

オブジェクト x の要約

```
> library(dplyr)
> summary.x <- x %>% select(month, assets_fix, assets_cur, assets_total,
                        shareholders, debt_long, liabilities_cur, employees,
                        operating_revenue, ebit, pl_before_tax, tax, net_income,
                        costs_employees) %>% summary()
```

ここでは，オブジェクト x における全列の要約ではなく，「量的」な変数（列）の要約を求めている．この要約の結果は以下のようなものである：

オブジェクト x の要約結果

```
> summary.x
     month            assets_fix          assets_cur
 Min.   : 0         Min.   :  -480975   Min.   :  -601243
 1st Qu.:12         1st Qu.:        0   1st Qu.:        8
 Median :12         Median :        6   Median :       55
 Mean   :12         Mean   :     4253   Mean   :     2628
 3rd Qu.:12         3rd Qu.:      113   3rd Qu.:      292
 Max.   :61         Max.   :1314984092   Max.   :1184670353
 NA's   :11442643   NA's   :11542011    NA's   :11499348
  assets_total         shareholders          debt_long
 Min.   :  -2071016   Min.   : -74145852   Min.   :  -436465
 1st Qu.:        14   1st Qu.:         1   1st Qu.:       0
 Median :        97   Median :        24   Median :       0
 Mean   :      6903   Mean   :      3079   Mean   :    1792
 3rd Qu.:       547   3rd Qu.:       172   3rd Qu.:      29
 Max.   :1431770781   Max.   :1314984092   Max.   :96047349
 NA's   :11470781     NA's   :11458019     NA's   :17161241
 liabilities_cur        employees        operating_revenue
 Min.   :   -365688   Min.   :     0     Min.   :  -959656
 1st Qu.:         2   1st Qu.:     1     1st Qu.:       10
 Median :        26   Median :     3     Median :       93
 Mean   :      2178   Mean   :    22     Mean   :     5391
 3rd Qu.:       169   3rd Qu.:     8     3rd Qu.:      534
 Max.   :1336680220   Max.   :842800     Max.   :182056642
 NA's   :11545236     NA's   :17300499   NA's   :15734957
      ebit           pl_before_tax            tax
 Min.   :-124574699   Min. :-124574723 Min.    :-3370000
 1st Qu.:         0   1st Qu.:       0   1st Qu.:       0
 Median :         2   Median :       2   Median :       0
 Mean   :       284   Mean   :     325   Mean   :      77
 3rd Qu.:        27   3rd Qu.:      26   3rd Qu.:       5
 Max.   :  25320317   Max.   :  25910117   Max.   :23450722
 NA's   :14847786     NA's   :14852273     NA's   :16681971
   net_income         costs_employees
 Min.   :-124574723   Min.   :  -37197
 1st Qu.:         0   1st Qu.:      14
 Median :         1   Median :      60
 Mean   :       256   Mean   :    1009
 3rd Qu.:        20   3rd Qu.:     236
 Max.   :  47419481   Max.   :50008000
 NA's   :14856933     NA's   :21138981
```

この結果から，欠測値（NA）が多く存在することがわかる．特に，今後の可視化やモデリングにおいては決算月数（month）が12ヶ月のものを利用する関係上，どの程度この条件を満たすデータが存在するかを調べる必要がある．また，ここにあるデータの中で完全データがどの程度存在するかを調べることも重要である．そこで，これらの情報を得るために以下の関数を用意した：

オブジェクト x に決算月数と欠測に関する条件を与えたオブジェクトの企業数を調べる関数 count.firms

```
> count.firms <- function(df1 = x)
{
  require(dplyr)
  df2 <- df1 %>% filter(month == 12)
  df3 <- df1 %>% na.omit()
  df4 <- df2 %>% na.omit()
  tab <- c(
    x = nrow(df1),
    x.12 = nrow(df2),
    x.NAomit = nrow(df3),
    x.12.NAomit = nrow(df4)
  )
  tab <- tibble(objects = names(tab), n.firms = tab)
  tab
}
```

この関数を実行し，その結果をオブジェクト xcf に付置した：

関数 count.firms の実行とオブジェクト xcf への付値

```
> xcf <- count.firms()
```

の実行結果は以下のように与えられる：

関数 count.firms の実行結果

```
> xcf
# A tibble: 4 × 2
  objects     n.firms
  <chr>         <int>
1 x           25584931
2 x.12        13855894
3 x.NAomit     2123300
4 x.12.NAomit  2113297
```

この結果から，データラングリングを実行した結果として得られたオブジェクト x の企業数が25,584,931社であるのに対して，決算月数（month）が12ヶ月のオブジェクト x.12 の企業数が13,855,894社となり，完全データのオブジェクト x.NAomit の企業数はさらに減って，2,123,300社である．また，決算月数が12ヶ月でかつ完全データの企業数にいたっては，2,113,297社となる．このことは，以下のように入力することによって可視化される（図11参照）．

関数 count.firms の実行結果 xcf の可視化

```
> library(ggplot2)
> xcf %>% ggplot(aes(n.firms, objects)) + geom_bar(stat = "identity")
```

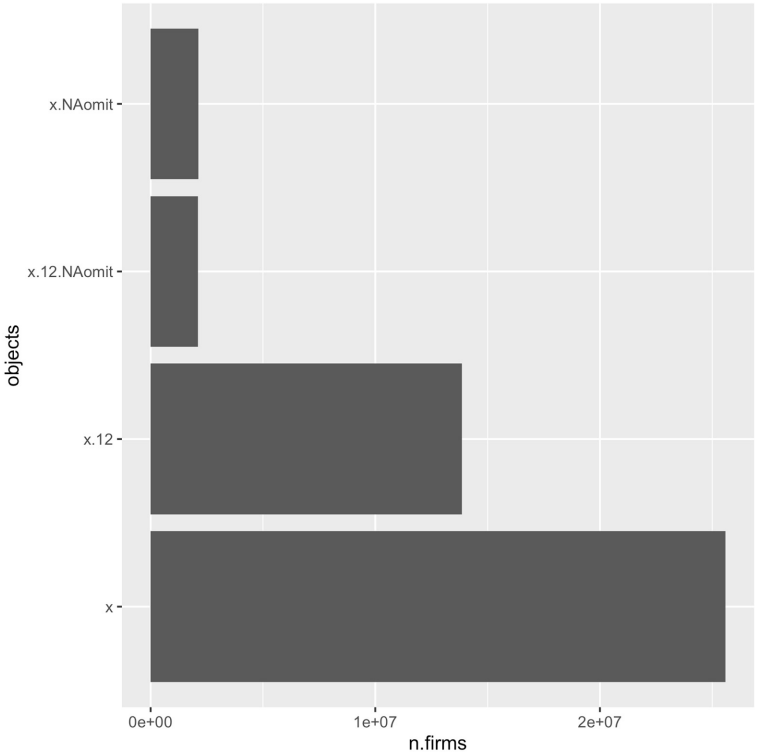

図11：関数 count.firms の実行結果 xcf の可視化

以上の考察から，決算月数が12ヶ月であり，かつ，III-1 項に挙げた24のデータ属性全てに欠測が存在しない完全データが200万社超利用できることがわかった[21)]．

21）今回抽出され，ラングリングされたデータの企業数（25,584,931社）から，この結果を考えると10分の 1 以下であるが，これまでこの種のデータを扱ってきた経験からは，逆に200万社以上のデータが完全に情報を持つことが，今後の可視化などの研究において貴重なものであると思われる．

VI おわりに

本稿では，Orbis データを利用して非上場企業の財務データのラングリングを行った．特に，make コマンドを利用して自動実行することによって，再現可能性を確保した．なお，本研究の目的である非上場企業の財務データの国別情報にもとづく可視化や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）

参考文献

［1］ Gandrud, C.（2020）*Reproducible Research with R and RStudio, Third Edition*, CRC Press.
［2］ 地道正行（2017-a）『Rによる対数非対称正規線形モデルによる財務データの統計モデリング』, 商学論究, 第64 巻, 第 5 号, pp. 159-185, 関西学院大学商学研究会.
［3］ 地道正行（2017-b）『R を利用した非対称分布族にもとづく財務データの統計モデリング』, 経済学論究, 第71巻, 第 2 号, pp. 141-174, 関西学院大学経済学部研究会.
［4］ 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.
［5］ 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.
［6］ 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.
［7］ 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.
［8］ 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.
［9］ 地道正行（2021-a）『財務データ抽出システムの再構築：NEEDS 企業財務データを中心に』, 商学論究, 第68巻, 第 3 号, pp. 1-78, 関西学院大学商学研究会.
［10］ 地道正行（2021-b）『財務データ抽出システムの再構築：Osiris データの利用』, 商学論究, 第69巻, 第 1 号, pp. 71-109, 関西学院大学商学研究会.
［11］ 地道正行（2021-c）『SKWAD ユーザマニュアル：NEEDS 企業財務データの抽出』, Ver. 1.0, pp. 1-88, 関西学院大学リポジトリ, http://hdl.handle.net/10236/00029654
［12］ 地道正行（2021-d）『財務データ抽出システムの再構築：Orbis データの利用』, 商学論究, 第69巻, 第 2 号, pp. 65-109, 関西学院大学商学研究会.

［13］ 地道正行, 宮本大輔, 阪智香, 永田修一（2017）『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 9 回シンポジウム, THE GRAND HALL（品川）, 2017年 7 月13 日（木）, 発表用ポスター. https://jhpcn-kyoten.itc.u-tokyo.ac.jp/abstract/jh171002-NWJ

［14］ 地道正行, 宮本大輔, 阪智香, 永田修一（2018）『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10回シンポジウム, THE GRAND HALL（品川）, 2018年 7 月12 日（木）, 発表用ポスター. https://jhpcn-kyoten.itc.u-tokyo.ac.jp/abstract/jh181001-NWJ

［15］ 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

［16］ 地道正行, 宮本大輔, 阪智香, 永田修一（2019）『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11回シンポジウム, THE GRAND HALL（品川）, 2019年 7 月11日（木）, 発表用ポスター. https://jhpcn-kyoten.itc.u-tokyo.ac.jp/abstract/jh191002-NWJ

［17］ 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.

［18］ 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

［19］ 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.

［20］ 地道正行, 阪智香（2021）『財務データと ESG レーティングデータによる株式時価総額の統計モデリング』, 商学論究, 第69巻, 第 2 号, pp. 1–64, 関西学院大学商学研究会.

［21］ 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

［22］ 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, 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

［23］ 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.

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

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

［26］ Tange, O.（2018）*GNU Parallel 2018*, ISBN：9781387509881, DOI：10.5281/zenodo.1146014, URL: https://doi.org/10.5281/zenodo.1146014

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

［28］ 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.）

［29］ 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.）

［30］ 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環境整備を，そして BvD 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．また，株式会社 ef-prime の鈴木了太氏からは，R のコーディングについて貴重なコメントを頂いた．ここに感謝の意を表する．

本研究の一部は以下の研究費より助成を得ている：

科研費科学研究費基盤研究 C：「グラフィカル・データ・アナリシスによる格差研究と社会環境会計による解決方法の提案」（2016年～2018年），課題番号：16K04022

科研費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

学際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～2021年度），課題番号：jh171002-NWJ, jh181001-NWJ, jh191002-NWJ, jh201003-NWJ, jh211001-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

付録 A　コンピュータ環境

ハードウェア環境

本研究を行うために，東京大学情報基盤センターに設置された専有利用型

リアルタイムデータ解析ノード（FENNEL）を利用させていただいた．以下に，1ノードの簡単な仕様を与える[22]：

OS: Ubuntu 16.04
CPU: Intell® Xeon® プロセッサー E5-2620 v4, 8 Cores
Main Memory: 16 GB
GPU Memory: 8 GB

また，本研究で利用した **PG-Strom** については以下のような環境を利用させていただいた：

OS: CentOS 7.8
CPU: Intel® Xeon® Silver 4214, 24 Cores
Main Memory: 376 GB
GPU Unit and Memory: NVIDIA® Tesla® V100, 16 GB

なお，ローカル環境は以下のようなものである：

- Machine: MacBook Pro 2018
  OS: macOS Big Sur（11.6）
  CPU: Intel® Core™ i9-8950HK, 6 Cores
  Main Memory: 32 GB

- Machine: iMac 2017
  OS: macOS Monterey（12.1）
  CPU: Intel® Core™ i7, 6 Cores
  Main Memory: 64 GB

- Machine: Mac mini 2020
  OS: macOS Monterey（12.1）
  CPU: Apple M1（arm64), 8 Cores
  Main Memory: 16 GB

22）今回利用しているノードは，GPU がある環境が2ノードであり，GPU がない環境が2ノードの合計4である．

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/）
  - **dplyr**（H. Wickham, http://dplyr.tidyverse.org/）
  - **ggplots2**（H. Wickham, https://ggplot2.tidyverse.org）
  - **magrittr**（S. M. Bache, H. Wickham, and L. Henry, https://magrittr.tidyverse.org）
- RStudio（RStudio, https://www.rstudio.com/）
- 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 4.1.2 (2021-11-01), 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 12.1
- Matrix products: default
- BLAS:
  /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRblas.0.dylib
- LAPACK:
  /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRlapack.dylib
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: dplyr 1.0.7, forcats 0.5.1, ggplot2 3.3.5, purrr 0.3.4, readr 2.1.1, stringr 1.4.0, tibble 3.1.6, tidyr 1.1.4, tidyverse 1.3.1
- Loaded via a namespace (and not attached): arrow 6.0.1, assertthat 0.2.1, backports 1.4.1, bit 4.0.4, bit64 4.0.5, broom 0.7.11, cellranger 1.1.0, cli 3.1.0, colorspace 2.0-2, compiler 4.1.2, crayon 1.4.2, DBI 1.1.2, dbplyr 2.1.1, digest 0.6.29, ellipsis 0.3.2, fansi 0.5.0, farver 2.1.0, fs 1.5.2, generics 0.1.1, glue 1.6.0, grid 4.1.2, gtable 0.3.0, haven 2.4.3, hms 1.1.1, httr 1.4.2, jsonlite 1.7.2, labeling 0.4.2, lifecycle 1.0.1, lubridate 1.8.0, magrittr 2.0.1, modelr 0.1.8, munsell 0.5.0, pillar 1.6.4, pkgconfig 2.0.3, R6 2.5.1, Rcpp 1.0.7, readxl 1.3.1, reprex 2.0.1, rlang 0.4.12, rstudioapi 0.13, rvest 1.0.2, scales 1.1.1, stringi 1.7.6, tidyselect 1.1.1, tools 4.1.2, tzdb 0.2.0, utf8 1.2.2, vctrs 0.3.8, withr 2.4.3, xml2 1.3.3