KIER DISCUSSION PAPER SERIES

KYOTO INSTITUTE
OF
ECONOMIC RESEARCH

Discussion Paper No.1601

“複数の企業データを用いたパネル・データ構築方法について”

行本　雅

2016 年 4 月

KYOTO UNIVERSITY

KYOTO, JAPAN

複数の企業データを用いたパネル・データ構築方法について

行本雅†

2016 年 4 月

要旨

近年では、大規模な個票データを用いたパネル・データ分析が広く行われるようになりつつある。また、単一の調査データだけでなく、複数の調査データを接続したパネル・データを構築して分析を行うことも増えてきている。しかしながら、こうした調査データは必ずしもパネル・データとして利用することを前提に整備されていないことが多い。

このため、個票データを研究目的で利用する際には、個々の研究プロジェクトごとにデータの状態の確認やデータ・クリーニングから開始しなければならない。とりわけ複数の調査データを接続してパネル・データを構築する場合には、かなりの手間と時間を要することになる。

こうしたデータの整備作業は使用するデータに依存する部分が大きいものの、データに関わらずある程度一般的に有用な処理も存在する。本稿は、比較的汎用性が高いと思われる処理についてまとめるとともにプログラム例を示すことで、今後の研究の用に供せんとするものである。

キーワード: パネル・データ構築、非標本誤差、企業データ
JEL: C81, C87, D22

† 京都大学経済研究所先端政策分析研究センター研究員

1.はじめに

近年では、パネル・データの分析手法が発達したこともあり、大規模な個票データを用いたパネル・データ分析が広く行われるようになりつつある。また、単一の調査データだけを用いるのではなく、複数の調査データを接続したパネル・データを構築して分析を行うことも増えてきている[1]。

しかしながら、こうした際に用いられる調査データは、必ずしもパネル・データとして利用することを前提に設計されていないことが多い。また、データのチェックもその調査の目的に問題が生じない範囲でなされていることが多いため、非標本誤差の問題に対処する必要が生じる[2]。この他、調査ごとに目的が異なるため複数の調査データを接続しようとするとしばしば整合性に問題が生じる。

このため、個票データを研究目的で利用する際には、個々の研究プロジェクトごとにデータの状態の確認やデータ・クリーニングから開始しなければならないことが多い。とりわけ複数の調査データを接続してパネル・データを構築する場合には、かなりの手間と時間を要することになる。したがって、同じデータを用いている研究であっても実際に分析に使用しているデータの精度は、個々の研究プロジェクトでデータ整備にどれだけの手間と時間をかけたかに大きく依存する。

こうしたデータの整備作業は使用するデータに依存する部分が大きいものの、ある程度一般的に有用な処理も存在する。そこで本稿は、比較的汎用性が高いと思われる処理についてまとめるとともに、現在ミクロ・データの分析に広く用いられている STATA の簡単なプログラム例を示すことで[3]、今後の研究の用に供せんとするものである。

2.本稿で想定するデータ

本稿では、現実の特定のデータに基づくのではなく、「main」と「sub」という架空の二つの調査データを想定する[4]。以下の処理に関わる主な仮定は、企業固有の番号が利用可能

---

[1] 企業名や住所情報等を利用して、異なるデータの同一企業を特定して接続する処理は名寄せと呼ばれる。これらの情報が用いることができない場合に、資本金や従業員数などの情報を利用して接続する方法がとられる場合もある。村田・伊藤(2015)では、照合方法による結果の比較を行い、企業名や住所情報等を用いない方法では規模が小さくなると精度が低下したとの結果を報告している。

[2] 非標本誤差は、照合によるデータの接続を行う上ではしばしば深刻な問題を引き起こす。実際の大規模な調査がどのように行われているかについては、松井(2008)が包括的で詳しい解説を行っており、実際の個票データを利用する上で参考となる。

[3] STATA は、統計パッケージの中ではデータ・マネジメント機能が充実しているため、ミクロ・データの分析に広く用いられている。ただし、分析用のデータ・セットを作成することに特化しており、SQL などのデータ・ベースソフトとは相違点も多い。

[4] 実際の個票データを用いて、企業名や住所情報等による名寄せ処理をともなうパネル・データ構築を行った研究としては周防・古隅・宮内(2009)がある。また、古隅(2011)では SAS でのプログラム例が示されている。

であること、企業単位のデータであること、企業名および住所情報等が利用可能であることである。

ほとんどの調査では、個々の企業に番号が振られている。この番号は、主に同一年の調査データ内で個々の企業を識別することを目的として振られており、必ずしも個々の企業固有の番号として時系列方向で固定されているわけではない。しかし、パネル・データとして利用可能な調査では、個々の企業ごとに時系列方向で固定された企業番号が振られていることや、事後的にコンバータが作成されておりこれを利用することで企業固有の番号を振ることができるように整備されていることが多い[5]。

本稿では、こうした時系列方向で固定された企業固有の番号が利用可能な二つの調査データを想定する。また、実際の調査データは調査の位置づけや性格によってデータ整備の状況に差があることが多い。そこで、整備状況のよい一方の調査データを主データ(以下、main データ)にしながら、これにもう一つの調査データを補助的なデータ(以下、sub データ)として接続して用いることを想定する。

この他、本稿ではプログラム例を簡単化するため、企業名や住所のクリーニングは簡単な処理しか行っていない。本稿の簡単な処理でも、主に大企業を中心とした調査データであればある程度有用であると思われるが、中小企業を対象とした調査データの場合には、これらについてより詳細かつ網羅的に行う必要が生じる[6]。

3.基本的な処理の方針と流れ

二つの調査データを用いたパネル・データの作成にはいくつかの方法があり、どの方法が適切かは使用するデータに依存する部分が大きい。本稿では、一方のデータ(main データ)をベースにして、これにもう一つのデータ(sub データ)を接続することでパネル化する方法を説明する。本稿の方法は、各調査データ内では企業番号を利用することでパネル化が容易であることを利用しようというものである[7]。

代替的な方法としては、二つの調査データをそれぞれ時系列方向でプールして集約し、照合を行うものがある。これは、照合に用いる変数が時系列方向で変化しない場合には効率的な方法である。しかし、実際には企業名や住所情報等は変更がありうる上に、変更がなくても表記の揺れがあれば、プールしたデータ内に同一企業のデータが複数生じること

---

[5] 企業番号が固定されていない調査データにおけるパネル・データ構築方法については、新保・高橋・大森(2005)や行本(2015)、周防・古隅・宮内(2009)、村田・伊藤(2015)を参照されたい。

[6] 中小企業を多く含む個票データにおける名寄せ処理をともなうパネル・データ構築方法については、周防・古隅・宮内(2009)および古隅(2011)を参照されたい。

[7] 企業番号が利用できない場合でも、同一調査内では企業名や住所情報等が一定期間そのまま利用されていることもあるので、まず同一調査内でパネル化を試みるのは有力な方法である。

になる。このため、個々のサンプルのアイデンティファイが難しくなるという問題がある。

また、個々の企業の企業番号が完全に固定されていない場合にも問題が生じる[8]。例えば、調査範囲に出入りがある調査では、一度調査対象から外れた企業が再度調査対象となったような場合には新たに企業番号が振られる運用はありうる。この場合、この企業は当該調査データの中で時期によって異なる企業番号を持つことになる。この際に、もう一方の調査データで一貫して同じ企業番号が振られていればアイデンティファイの問題が生じる。

本稿の処理の方針は、同一年の二つの調査データ間で企業名や住所情報等による照合を行い、パネル化した main データ側に sub データの企業番号(以下、ID)を付与し、これを外部キーとして sub データを接続することで、main データをベースとしてパネル化を行うというものである。ただし、実際の個票データは重複サンプルの問題や入力段階での表記の揺れなどによりそのままでは照合が上手くいかないため、照合作業の前段階でデータ・クリーニングを行う[9]。

なお、退出処理については、データが悉皆か否か、調査対象の範囲、操業状態や退出についてのフラグが利用可能かどうか、といった使用データに依存する部分が大きいため本稿では対象外とする[10]。この他、本稿ではひとまず同一年の main データと sub データ双方で観察されるサンプルによるパネル・データを作成する。ただし、main データ側で観察されるすべてのサンプルをふくめるように拡張することは容易である。

処理は大きく分けて、パネル・データの設計、データの変換とチェック、照合作業と接続したパネル・データの構築の三段階からなる。以下では、STATA での処理の流れについて簡単に説明する。バージョンは STATA13 である[11]。なお、各処理の詳細については補論のプログラム例を参照されたい。各段階の処理と対応するプログラム例は以下の通りである。なお、各節のタイトルの後の括弧内がプログラム例の対応する do ファイル名である[12]。

パネル・データの設計

3.1 フォルダ・システムの構築とデータの準備(01mkdir.do)

まず、フォルダ・システムを構築し、オリジナル・データを用意する。フォルダ・システムの構築には mkdir コマンドを使用する。また、提供されるデータ・ファイルに調査名

---

8 大規模な調査では、企業番号の管理を徹底するのは容易なことではない。

9 実際の個票データを用いて照合作業を行う際に、どのような点が障害となるかについては周防・古隅・宮内(2009)および古隅(2011)、村田・伊藤(2015)が詳しく報告している。

10 実際の退出処理の例としては、例えば行本(2015)を参照されたい。退出処理を行うには回答状況を元にフラグを作成して生存期間を調べることになる。これについては、本稿のプログラム例の補間処理が参考になると思われる。

11 STATA は STATA14 からユニコードに対応したため、それ以降のバージョンでは文字コードに起因する問題は減少するかもしれない。

12 本稿のようなデータ・マネジメントには、初歩的なデータベースやプログラミングの知識が必要である。データベースの入門書としては、増永(2006)などがある。また、STATA でのデータ・マネジメントについて実践的に解説したものとしては、Mitchell(2010)がある。

等が日本語表記でつけられていることがあるが、プログラミングが容易になるよう英語表記やローマ字表記に変更しておく。

プログラム例では、「00main2001.txt」、「00sub2001.txt」といったように調査名とデータ年を組み合わせた名前をつけている。また、冒頭に数字をつけているのは作業工程のどの段階のデータかわかりやすくするとともに、円記号問題を回避するためである[13]。

3.2 変数情報の取得とパネル・データの設計(02describe.do)

オリジナル・データの変数情報を取得し、最終的に作成するパネル・データの設計を行う。また、この段階で予備的なデータのチェックも行う。この際、必ず調査票や記入の手引き、コード表等を参照して、調査と回答内容がいつの時点なのか、また両者の時点に乖離はないか、変数の定義、単位等を確認しておく。

まず、提供されたオリジナル・データを読み込んで、describe コマンドでデータに格納されている変数の情報を取得する。データの読み込みはデータの形式に応じたコマンドを用いればよい。

プログラム例は、大規模なデータはカンマ区切りのテキスト形式で提供されることが多いため、import delimited コマンドで作成してある。なお、予約語を回避するためオプションで case を指定してある。

この段階では、データが正常に読み込めているか、変数の並び順、数値か文字列か、変数にデータが実際に入力されているかなどをチェックする。例えば、通常最初に格納されている企業番号等が異常に長い文字列になっているような場合には正常に読み込めていないので、テキスト・エディタ等でオリジナル・データを確認して原因を特定する[14]。また、桁数の大きい変数のデータ型が byte 型で読み込まれている場合は、データが入力されていないのでやはり確認しておく。

データが正常に読み込めていることが確認できれば、取得した変数情報に基づいて期間中の変数一覧表を作成し、最終的に作成するパネル・データの設計を行う。時系列方向や調査データ間で調整が必要かどうかを確認した上で、パネル・データ用の変数名とデータ型、単位などを決定する。

プログラム例では単純化のために、二つの調査データのすべての年のデータで処理に使用する変数(企業番号、企業名、都道府県、市区町村、住所、郵便番号、電話番号、資本金)が最初から同じ順に並んでいる非常に単純化したデータを想定している[15]。

---

[13] 「¥」と「＼」が識別できないことによって、プログラム上のエラーが生じる問題である。

[14] 例えば、一つ上の行のサンプルのレコードが途中で改行されている場合にこうしたことが起きる。

[15] 実際の個票データではこの段階で変数の調整が必要な場合が多いが、ほぼ使用データに依存するため本稿では扱わない。実際の個票データでの変数一覧表の作成例としては、例えば行本(2015)を参照されたい。

データの変換とチェック

3.3 データの変換(03import.do および 03_1importmain2001.do 以下の下位の do ファイル)

パネル・データの設計に沿って、データの STATA 形式への変換を行う。この際に、変数名をつけるとともに、データ型も指定する。また、データの読み込み時にエラーが生じている場合は原因を特定して対処する。この他、重複サンプルや欠損値などについて簡単なチェックを行いデータの状況を把握しておく[16]。

さらに、一度各データを縦に接続してパネル化し、時系列方向や二つの調査データ間の表記の揺れ等を目視で確認しておく。この目視での確認作業は、この後のデータ・クリーニングでどのような処理を行うかを判断するための材料となる。

プログラム例では、オリジナル・データがカンマ区切りのテキスト形式であることを想定して import delimited コマンドで作成してある。なお、プログラム全体の構造を見通しやすくするため個々のデータ・ファイルごとに個別に do ファイルを作成して入れ子状にしてある。このプログラムは、パネル・データを設計する際の変数一覧表にオリジナル・データの変数の並び順を含めておけば容易に作成できる[17]。

この他、オプションで読み込む範囲を指定するとともに、データ型はすべて文字列を指定し[18]、データの読み込みエラー対策として引用符のオプションを指定してある。また、通常、調査年はデータに含まれていないことが多いので作成している。さらに、この段階が最もエラーが生じやすいので、データ変換後に再度 describe コマンドでデータが正常に読み込めているか確認するようにしている。

3.4 データ・クリーニング(04cleaning.do)

実際の個票データでは表記の揺れが多く存在する。これは、同一調査データ内でもしばしば時期によって入力方針が変更される場合がある上、異なる調査データ間では入力方針が異なることが多いためである。この他にも、調査票の記入者やデータの入力者、OCR ソフトなどによっても揺れが生じることになる。このため、照合に先立ってデータ・クリー

---

16 本稿では扱わないが、コード化された変数がある場合には、この段階でコード表に存在しないコードが入力されていないかをチェックしておく。これには tabulate コマンドを使用すればよい。

17 実際の個票データは年によって調査項目や順番に変更があることが多いので、面倒なようでも変数一覧表に基づいてすべての年についてプログラムを作成するのが確実である。なお、import delimited コマンドの場合は、使用しない変数についても読み込む範囲にふくまれるものは変数名をつけておく必要がある。

18 通常、企業番号等は先頭部分に「0」をつけた桁数の固定された数値のことが多い。文字列でなく数値に変換した場合、先頭部分の「0」の情報が失われる。複数の変数を組み合わせた複合キーの場合には数値に変換してから結合すると問題が生じるので、最初の段階では文字列のまま読み込むのが無難である。

ニングを行う[19]。

本稿のデータ・クリーニングは、二つのデータ間での照合を可能にすることが主な目的である。すなわち、入力段階での表記の揺れ等の影響を受けないように表記などを統一するとともに、照合の妨げとなっている様々な要因を排除することを行う。ただし、データ・クリーニングによって失われる情報もあるため、データ・クリーニング後の住所データ等がより正確になるという性質のものではないことに留意されたい。

したがって、どのようなデータ・クリーニングが適切かは、最終的には使用データの状態を確認しながら適宜見極める必要がある。本稿では、比較的簡単なクリーニング処理について説明する[20]。

3.4.1 企業名のクリーニング(04cleaning_name.do)

企業名は照合作業の最も基礎的な情報となるが、実際の個票データでは入力段階での表記の揺れや入力ミスなどが存在することが多い。例えば、スペースの入力、株式会社や有限会社の表記方法や表記の位置、小文字と大文字、異体字、誤入力、などがある(周防・古隅・宮内:2009、古隅:2011)。これらのうち誤入力についてはどうにもならないが、それ以外についてはデータ・クリーニングで削除や表記の統一を行うことである程度照合が可能になる。

プログラム例では、スペースや株式会社、有限会社等の表記については削除している[21]。また、ア行とヤ行、促音の小文字は大文字に、アルファベットは 2 バイト文字を 1 バイト文字に統一するほか[22]、長音記号やハイフンなどについても統一している。この他、中点や句読点、ピリオド、カンマ等について削除している[23]。

3.4.2 都道府県・市区町村・住所の再構築(04address_rstr.do)

住所情報は、実際の個票データでは「都道府県」、「市区町村」、「住所」などに分けて入力されている場合が多い。ただし、区切り方は調査データによって異なることがあるし、同一の調査データ内でも揺れや処理に用いたプログラムによるエラーが生じていることがある(古隅:2011)。このため、一度「都道府県」、「市区町村」、「住所」をすべて結合し、再度分離することで再構築する。これは、二つの調査データに同一の処理を行うことで揺れをなくすことで照合を可能にするのが目的である。

再構築は、以下の手順で行う。まず、「都」、「府」、「県」が入力されていないものについ

---

19 データ・クレンジングともいう。

20 より一般的な企業名や住所情報のデータ・クリーニングについては、古隅(2011)を参照されたい。

21 株式会社や有限会社等の表記やその位置については、フラグを立てて情報を利用することも考えられる。

22 2 バイト文字と 1 バイト文字とは、いわゆる全角文字と半角文字のことである。

23 これらは、状態のよくない紙媒体をスキャンし OCR で処理した場合に入力されやすい。

ては補う。次に、「都道府県」、「市区町村」、「住所」をすべて結合する。最後に、「都道府県」、「市区町村」、「住所」を再び分離する。

このうち問題となるのは分離処理である。例えば「市区町村」を分離するには、「市」、「区」、「町」、「村」の文字の位置を調べて分離することになるが、これらの文字が「市区町村」名や「住所」に含まれることがある。そこで、実際の住所データに基づいて、こうしたエラーを回避するようプログラムを作成する必要がある。

例えば「市」であれば、「市川市」、「余市郡」、「下市町」、「市ヶ谷」のように市区町村名や郡名、住所等に「市」という文字を含む場合がある。このため、単純に「市」の位置を調べるだけでは適切に処理できない。そこで必要に応じて、こうした例外処理をあらかじめ個別に行った上で、「市」の位置による制約をかけるなどして分離する。

プログラム例は、日本郵便株式会社が公表している 2015 年 12 月時点の郵便番号データに記載されている住所情報に基づいて作成してある。また、過去についても公表されている範囲については遡及している。

3.4.3 住所のクリーニング(04cleaning_address.do)

住所は企業を特定する上では重要な情報であるが、入力段階での表記の揺れや入力ミスが生じやすい。特に町域よりあとの「丁目」、「番地」、「号」の部分は様々な表記方法がありうる(周防・古隅・宮内:2009、古隅:2011)。このため、データ・クリーニングで削除や表記の統一を行わなければ照合が上手くいかないことが多い。

プログラム例では、数字について 2 バイト文字を 1 バイト文字に統一した上で、町域より前の住所表記で数表現を含むもののうち数字（アラビア数字）で表記される可能性のある「番町」、「条」[24]について漢数字に置き換えている[25]。また、「上ル(る)」、「下ル(る)」、「入ル(る)」[26]のひらがなをカタカナに、「ケ(ヶ)」と「カ(ヵ)」の小文字を大文字に、漢字の「之」をカタカナの「ノ」に、「霞ケ(が)関」、「丸ノ(の)内」のひらがなをカタカナに統一し、「通(り)」の送り仮名を削除している。この他、照合の際に支障をきたしやすい町域より後の「丁目」、「丁」、「番地」、「番」、「号」、ハイフン等や、「大字」、「字」も削除している。さらに、町域より後の数字は入力ミスが起きやすいため、町域までを抽出することも行っている。この際、最後の「町」は省略されることもあるので削除している。

なお、プログラム例では町域よりあとの「丁目」、「番地」、「号」の部分については数字で入力されていることを想定している。漢数字で入力されている場合には、対応していな

---

24 「条」の数表現が数字で表記されるケースは北海道の住所にみられる。京都にも同様の住所があるが漢数字で表記される。

25 プログラム例では、1 から 10 までしか作成していない。実際にはもっと大きな数まで存在するが、二桁以上になると漢数字表記で「十」を表記するか否かといった問題が生じうるため、本稿ではひとまず処理を見送っている。また、「番町」、「条」以外にも数字で表記される可能性のある住所表記は存在するが、ここではすべてに対応はしていない。

26 これらは京都にみられる住所表記である。

いことに留意されたい[27]。

3.5 データのチェック(05data_check.do)

この段階でデータの最終的なチェックを行う。照合作業を行うためには、個々のサンプルのアイデンティファイがなされなければならないため、照合に用いる変数について重複しているサンプルを除外する処理を行う。これには、重複をチェックしたい変数を指定して、duplicates report および duplicates tag コマンドを使用する。

重複サンプルが生じる原因はいくつか考えられるが、企業名や住所情報等に変更や揺れが生じているために異なる企業として認識されている、同一企業の複数の拠点が回答している、M&A にともなうもの、などがある。このうち、M&A にともなうものは少し注意を要する。この問題は、調査と回答内容に時点の乖離がある場合に生じやすくなる。

例えば、回答内容の時点では別々の企業であったが調査時点では合併していたというような場合である。このとき、回答内容は合併前の個々の企業のものであるが、企業名や住所情報等は調査時点の合併後のものであるということが起きうる。もっとも、これだけであれば前年のデータを照合すれば対処は可能である。しかし、回答内容が合併後のものであるような可能性をも考慮すると、調査データのみでは判断ができなくなる。このため、こうしたサンプルは除外せざるをえない。

また、我が国の制度では、同一名称の異なる企業が存在しうることにも注意しなければならない。2005 年の商法改正までは、同一市町村内での類似商号が規制されていたため[28]、企業名と市町村が同じであれば重複サンプルの可能性が高い。しかし、市町村が異なれば同一企業名の異なる企業の可能性も考慮する必要があるし、法改正以降は同一市町村内であっても同一企業名の異なる企業ということもありえる。このため、同一企業名のサンプルを一括して除外すればよいということにはならない。

プログラム例では、照合に企業名と住所情報を主に用いるため、企業名・都道府県・市区町村が同一のサンプルを除外したデータを作成している。また、企業名と資本金による照合も行うため、これらが重複するサンプルについて除外したデータも作成している。ただし、企業名と資本金が一致する異なる企業も存在しうるため、一定規模以上のサンプルに絞っている[29]。この他、各調査の ID を数値に変換した場合に問題が生じないか、照合に用いる変数の欠損値の状況などをチェックしている。

なお、ここでの重複サンプルの処理は、以下で行う照合作業に支障をきたさないことを目的としていることに留意されたい。仮に一方の調査データ内で重複があったとしても、

---

27 もし、漢数字で入力されている場合には「丁」などの位置を調べて後方の文字を落とすことになろう。この他、古隅(2011)が提案している方法も参照されたい。

28 旧商法第十九条。

29 プログラム例では、資本金 2 億円以上に限定している。規模が小さくなれば企業数も多くなるため、同一企業名の異なる企業が存在する可能性も高くなる。この点については、村田・伊藤(2015)も参照されたい。

二つの調査データを接続するときにどちらかのサンプルとしか接続できなければ、最終的に二つの調査データを接続したデータ内では一意となるため問題は生じない。

ただし、二つの調査データで同じ企業について重複が生じていればやはり重複サンプルによる問題が生じる。このため、後で照合結果を統合する際に(3.9 節)、重複による不整合が生じていないかを再びチェックしなければならない。もし、この統合時のチェックで不整合が多く検出されるような場合には、この段階での重複サンプルのチェックをより厳しくする必要がある。

なお、本稿では直接扱わないが、各調査を単独で利用するような場合にはより厳しいチェックを行う必要があることに注意されたい。例えば、企業名と電話番号が重複するサンプル等は同一企業の情報が重複している可能性が高いために除外する、といった処理が考えられよう。

3.6 各パネル・データの構築(06panel.do)

チェックを行ったデータを整理して、照合作業に用いるクロスセクション・データを作成する。さらに、これらを縦に接続して、二つの調査データの接続作業に用いるパネル・データを作成する。

まず、照合用のクロスセクション・データを作成する。この後の照合作業には merge コマンドを使用するが、STATA の merge コマンドは二つのデータの同一名の変数間で照合を行い、照合に用いた各データの変数を一つにまとめる処理を行う。このため、照合に用いる変数は変数名を同じにしておく。ただし、後で接続処理に用いる各調査データの ID は区別できるように調査名を変数名につけておく。

次に、各調査データについて、接続に使用するパネル・データを作成する。本稿では二つの調査データで企業固有の番号が利用できることを想定しているため、これを利用して各調査データのパネル化を行う。また、接続用のデータでは、二つの調査のデータを区別できるようにすべての変数名に調査名をつけておく。

パネル化の手順は以下の通りである。まず、照合用に作成したクロスセクション・データを append コマンドで縦に接続する。次に、ID を数値に変換した変数を作成して xtset コマンドで STATA にパネル・データとして認識させる[30]。これで、long 形式のパネル・データが作成される。さらに、reshape コマンドを使用して wide 形式に変換する[31]。

照合作業と接続処理

3.7 照合作業(07match.do)

企業名と住所情報等を用いて、二つの調査データの同一年のクロスセクション・データ

[30] 数値に変換するのは、xtset コマンドでは文字列が使用できないためである。
[31] STATA の long 形式と wide 形式は、それぞれ縦持ち、横持ちのデータ形式のことである。

間で逐次照合作業を行う。照合は、厳しい条件のものからより緩い条件のものまで逐次行う。これには照合に用いる変数をキーに指定して、merge コマンドを一対一対応で使用すればよい。各条件での照合の結果、当該年の二つの調査データの ID 間の対応関係が得られる。ただし、照合に用いた変数が欠損値であったものについては、sub データの ID を欠損値に置き換えておく。その上で、これらを統合して当該年の対応表を作成する。これには、main データの ID をキーに一対一対応で merge コマンドのオプション update を指定して、各条件の照合結果を逐次接続すればよい。

また、企業名と資本金による照合も行う。これは、同一企業であっても二つの調査で異なる住所が記載されている場合を考慮するためである[32]。企業単位の調査の場合、本社の住所情報等を用いて照合を行うが、調査によって調査票での住所のたずね方が異なることがある。例えば、登記上の本社の住所をたずねる場合と、実質的な本社機能を有する場所をたずねる場合では、同一企業であっても回答される住所は異なりうる。さらに、後者の場合には当該調査の目的によっても、その本社機能を有する拠点は異なりうる[33]。

ただし、企業名と資本金による照合結果は、企業名と住所情報等による照合結果に比べるとやや確実性に欠ける。そこで、資本金が一定以上の場合に限定して行う。また、住所情報等による照合結果との統合はこの段階では行わない。

3.8 補正処理(08deploy.do)

照合結果に基づいて、二つの調査データのパネル・データを接続し、照合の基準年以外に展開した上で補間処理を行って補正を行う。

まず、wide 形式の二つの調査データを各年の照合結果に基づいて接続する。main データの wide 形式のパネル・データに、照合の結果得られた対応表を main データの ID をキーとして一対一対応で接続することで照合年の sub データの ID を付与し、これを外部キーとして sub データの wide 形式のパネル・データを一対一対応で接続する。

次に、照合年を含むすべての年について、二つの調査データの企業名が存在していて一致していれば照合の結果得られた sub データの ID を転記して展開する。この結果、前後で企業名が一致しているものについては、その間の期間中についても二つの調査データに企業名が存在していれば照合結果の sub データの ID を転記する補間処理を行う。

ただし、照合年のみ二つの調査データの企業名が一致しており、他の年で企業名の矛盾が生じているものについては問題が起きている可能性が高いため[34]、展開した照合年の sub データの ID を欠損に置き換える。

---

[32] こうした問題は、大企業の場合に起きやすい。

[33] 例えば、東京に本社が置かれているが、生産部門の統括機能は本社工場にあるようなケースはありえる。

[34] 例えば、M&A によって当該年のみ重複サンプルが生じているような場合にこうしたことが起きる。

3.9 照合結果の統合(09update.do)

各年の照合結果に基づいて他の年について補正を行ったものを統合する。ただし、この際に整合性に問題が生じるものについては、問題が生じている可能性が高いためサンプルから除外する。

照合結果の統合は、他の年への展開や補間による結果よりも当該年の照合結果が優先されるように二段階で行う。まず、補正処理を行った結果得られた対応表のうち照合の基準年のみを抽出したものを[35]、main データの ID をキーとして一対一対応で merge コマンドですべて接続して統合作業のベースとなる対応表を作成する。次に、各年の照合結果を基準とした対応表全体を、main データの ID をキーとして一対一対応で merge コマンドのオプションで update を指定して逐次接続する。

整合性のチェックは二通り行う。まず、照合結果の統合時にコンフリクトが生じているものを除外する[36]。例えば、本社の移転などによる住所情報の変更があった際に sub データ内でこの企業が異なる企業と認識され、時期によって異なる ID が振られており、さらにある時点で重複が生じている場合にこうした問題が起きうる[37]。このとき、照合の結果、時期によって main データの同一企業に対して sub データの異なる ID が対応づけられることになる。ここでさらに sub データ側のある時点で重複サンプルの問題が生じていれば、各時点での照合結果を基準として他の年に展開したことによって不整合が生じる[38]。

次に、各時点の sub データの ID に重複が生じているものについても除外する。例えば、sub データの同一 ID が時期によって異なる企業に振られている場合にこうした問題が起きる。この場合、main データの複数の異なる企業に sub データの同一 ID が対応づけられるために重複が生じる。

3.10 照合結果の追加(10add.do)

企業名と資本金など住所情報以外の情報を用いて行った照合結果の追加を行う。ただし、企業名と資本金などによる照合結果は、住所情報等による照合結果に比べて確実性に欠けるため、あくまでも住所情報等を用いた照合結果を優先してそれに追加する形で用いる。

まず、住所情報等を用いた照合結果を各年に分割した上で、当該年の企業名と資本金による照合結果を追加する。住所情報等による照合結果を優先させるため、merge コマンドのオプション update を使用して main データの ID をキーに一対一対応で接続する。この

---

35 これは、照合結果に他の年での整合性を考慮して修正を行ったものである。

36 これは、_merge のフラグが 5 になっているものである。

37 重複サンプルが生じるのは、住所情報等による照合に用いるデータの重複サンプルのチェックが企業名、都道府県、市区町村に基づいており、企業名が同じで都道府県、市区町村の異なるサンプルについては除外していないためである。

38 なお、sub データの同一企業に時期によって異なる企業番号が振られていても、同一年での重複が生じていない場合にはコンフリクトは生じない。

際、住所情報等による照合結果側に存在しないサンプルを落とすことで[39]、これまでの整合性チェックですでにはじかれているサンプルを除外する。次に 3.8 節と同様の補正処理を行う。最後に、住所情報等を用いた照合結果をベースに 3.9 節と同様に補正を行ったものを統合する。この際にも、整合性に問題が生じているものについてはサンプルから除外する。

これで、main データの ID に対して各年の sub データの ID を対応づける、パネル形式の対応表が作成されることになる。

3.11 データの接続(11connect.do)

最後に main データをベースにして sub データを接続する。まず、3.10 で作成した対応表を long 形式に変換する。次に、long 形式の main データに long 形式の対応表を main データの ID と year をキーとして一対一対応で接続することで、main データに対応する sub データの ID を付与する。さらに、これと year を外部キーとして long 形式の sub データを一対一対応で接続する。

プログラム例では、一対一対応で処理する際には欠損値があるとアイデンティファイできないため、同じ年に両方の調査データで観察されるサンプルのみを接続している。同じ年に対応する sub データのサンプルが観察されない main データのサンプルについてもデータに含めたい場合は、再度 main データと一対一対応で接続するか sub データを接続する際に重複サンプルがないことを確認した上で欠損値を許容するように多対一対応で接続すればよい[40]。

なお、最終的に正常に接続が行われているかは必ず目視での確認を行う。この他、異常が疑われるサンプルについては、google による検索なども用いて確認する。

4.むすび

近年、大規模な個票データが研究目的で広く利用されるようになってきているが、こうした大規模な個票データを分析に使用できる状態に整備するのは容易ではない。とりわけ複数の調査データを接続してパネル・データを構築する場合には、かなりの手間と時間を要することになる。

こうしたデータの整備作業は使用するデータに依存する部分が大きいものの、データに関わらずある程度一般的に有用な処理も存在する。そこで、本稿では今後の研究の参考となるように、これらについてまとめるとともに STATA の簡単なプログラム例を示した。本稿が、我が国のデータ整備の精度の向上ひいては実証研究の信頼性向上に寄与することになれば幸いである。

---

[39] これは、_merge のフラグが 2 になっているものである。

[40] ただし、この場合は重複サンプルのチェックをより厳しくする必要が生じる。

参考文献

新保一成・高橋睦春・大森民（2005）「工業統計パネルデータの作成―産業構造データベースの一環として―」, RIETI Policy Discussion Paper Series05-P-001.

周防節雄・古隅弘樹・宮内環（2009）「法人企業統計調査と事業所・企業統計調査の統合による企業データベース：1983～2005 年」，『統計数理』，第 57 巻，第 2 号, pp.277-303.

古隅弘樹（2011）「企業データを統合するための名寄せ処理技法」，『SAS ユーザー総会アカデミア/テクノロジー&ソリューションセッション論文集』, 2011 巻, pp.385-396.

増永良文（2006）『データベース入門』，サイエンス社.

松井博（2008）『公的統計の体系と見方』，日本評論社.

村田磨理子・伊藤伸介（2015）「賃金構造基本調査に対するデータリンケージの可能性について」，一橋大学経済研究所 Discussion paper Series A No.631.

行本雅（2015）「工業統計調査のパネル・データ整備の現状について」，京都大学経済研究所 Discussion paper No.1506.

Mitchell, Michael N.（2010）*Data Management Using Stata：A Practical Handbook*, Stata Press.

日本郵便株式会社 web ページ
http://www.post.japanpost.jp/zipcode/download.html

google web ページ
https://www.google.co.jp/

補論　プログラム例について

本稿のプログラム例は、以下の do ファイルから構成される。STATA13 で作成している。また、「main」と「sub」という架空の二つの調査データについて 2001 年から 2010 年までのデータが存在するものとして作成している。

なお、「00control.do」は「03import.do」以下の do ファイルの制御用である。

00control.do

01mkdir.do
02describe.do

03import.do
　03_1importmain2001.do, 03_1importmain2002.do, 03_1importmain2003.do,
　03_1importmain2004.do, 03_1importmain2005.do, 03_1importmain2006.do,
　03_1importmain2007.do, 03_1importmain2008.do, 03_1importmain2009.do,
　03_1importmain2010.do
　03_1importsub2001.do, 03_1importsub2002.do, 03_1importsub2003.do,
　03_1importsub2004.do, 03_1importsub2005.do, 03_1importsub2006.do,
　03_1importsub2007.do, 03_1importsub2008.do, 03_1importsub2009.do,
　03_1importsub2010.do

04cleaning.do
　04cleaning_name.do
　04address_restr.do
　04cleaning_address.do

05data_check.do
06panel.do
07match.do
08deploy.do
09update.do
10add.do
11connect.do

do ファイル使用方法

本稿のプログラム例は、あくまでも架空のデータを想定した例示であるが、使用データにあわせて適宜修正を行えば使用できるように作成している。使用する際は、基本的には以下の手順にしたがえばよい。

1.各 do ファイルの冒頭部分で各種設定を行っている、global 変数の宣言部分を使用データにあわせて調整する。
2.任意のドライブ直下で「01mkdir.do」を使用してフォルダ・システムを構築する。作成された「DM」フォルダ直下に、各 do ファイルを移動させる。
3.作成された「original」フォルダ内の各フォルダにオリジナル・データをコピーして、ファイル名を調整する。
4.「02describe.do」の import delimited コマンドを実際の使用データにあわせて変更する。
5.「03import.do」の下位の各 do ファイルを、実際の使用データから作成した変数一覧表に基づいて作成する。
6.「00control.do」を実行する。

この他、適宜使用データにあわせて修正していただきたい。

想定データ

本稿のプログラム例の架空の二つのデータは、ともに下記のような変数をふくんでいると想定している。

企業番号(ID)　　：桁数の固定された番号(1 バイト文字)
企業名(name)　　：仮名交じりの漢字表記の企業名
都道府県(state)　：都道府県名
市区町村(city)　　：郡名をふくむ市区町村名
住所(address)　　：住所表記(町域より後の数表現はアラビア数字)
郵便番号(zip)　　：郵便番号(1 バイト文字)
電話番号(tel)　　：電話番号(1 バイト文字)
資本金(capital)　：資本金(単位 100 万円)

```
*00control.do

set more off

*各種設定
*ログフォルダ
global logf "log"

*使用データ名
global dataA "main"
global dataB "sub"

global dlist = "${dataA} ${dataB}"

*接続開始・終了年
global startYear=2001
global endYear=2010

*doファイル実行
do 03import.do
do 04cleaning.do
do 05data_check.do
do 06panel.do
do 07match.do
do 08deploy.do
do 09update.do
do 10add.do
do 11connect.do
```

```
*01mkdir.do

*フォルダの展開

*設定(使用データ名)
global dataA "main"
global dataB "sub"


*フォルダ展開
mkdir DM

cd "DM"
mkdir original
mkdir stata
mkdir log

cd "original"
mkdir 00${dataA}
mkdir 00${dataB}

cd ..

cd "stata"
mkdir 03${dataA}
mkdir 03${dataB}

mkdir 04cleaning
mkdir 05data_check
mkdir 06cross
mkdir 06panel
mkdir 07match
mkdir 08deploy
mkdir 09update
mkdir 10add
```

```
*02describe.do

*各種設定
*ログフォルダ
global logf "log"

*使用データ名
global dataA "main"
global dataB "sub"

global dlist = "${dataA} ${dataB}"

*接続開始・終了年
global startYear=2001
global endYear=2010


capture log close
log using "${logf}¥02describe.log", replace

set more off

*変数情報の取得

*各データの変数情報取得
forvalues cnt = $startYear/$endYear{
	display "`cnt'年の${dataA}の変数情報の取得"
	import delimited "original¥00${dataA}¥00${dataA}`cnt'.txt", case
	describe
	clear

	display "`cnt'年の${dataB}の変数情報の取得"
	import delimited "original¥00${dataB}¥00${dataB}`cnt'.txt", case
	describe
	clear

}

log close
```

```
*03import.do

/*
*doファイル単独使用時用
global logf "log"

global dataA "main"
global dataB "sub"

global dlist = "${dataA} ${dataB}"

global startYear=2001
global endYear=2010
*/

capture log close
log using "${logf}¥03import.log", replace

set more off

*データのSTATA形式への変換
foreach data in $dlist{
	forvalues cnt = $startYear/$endYear{
		do 03_1import`data'`cnt'.do `data' `cnt'
		}
	}


*データの状態を確認するため一度縦に接続する
foreach data in $dlist{
	display "`data'のチェック用データ"
	use "stata¥03`data'¥03`data'${startYear}.dta", clear
		local cnt = $startYear + 1
	while `cnt' <= $endYear{
		display "`cnt'年のデータの追加"
		append using "stata¥03`data'¥03`data'`cnt'.dta"
		local cnt = `cnt' + 1

	}

	sort ID year
	save "stata¥03`data'¥03`data'_test.dta", replace
	clear

}
```

```
*03_1importmain2001.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importmain2002.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importmain2003.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
        display "`data'の`cnt'年のデータの変換"
        import delimited ID name state city address zip tel capital ///
                using"original¥00`data'¥00`data'`cnt'.txt", ///
                bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
                gen year = `cnt'

                keep ID year name state city address zip tel capital
                order ID year name state city address zip tel capital

                sort ID
                describe
                duplicates report ID
                duplicates report name
                duplicates report name state city
                duplicates report name state city address
                count if ID == ""
                count if name == ""
        save "stata¥03`data'¥03`data'`cnt'.dta", replace
        clear


log close
```

```
*03_1importmain2004.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importmain2005.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importmain2006.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importmain2007.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importmain2008.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importmain2009.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importmain2010.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importsub2001.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importsub2002.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importsub2003.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importsub2004.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importsub2005.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importsub2006.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importsub2007.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importsub2008.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importsub2009.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*03_1importsub2010.do

args data cnt

capture log close
log using "${logf}¥03_1import`data'`cnt'.log", replace

set more off

*Stata形式へのデータ変換
	display "`data'の`cnt'年のデータの変換"
	import delimited ID name state city address zip tel capital ///
		using"original¥00`data'¥00`data'`cnt'.txt", ///
		bindquotes(strict) colrange(1:8) rowrange(2) stringcols(_all)
		gen year = `cnt'

		keep ID year name state city address zip tel capital
		order ID year name state city address zip tel capital

		sort ID
		describe
		duplicates report ID
		duplicates report name
		duplicates report name state city
		duplicates report name state city address
		count if ID == ""
		count if name == ""
	save "stata¥03`data'¥03`data'`cnt'.dta", replace
	clear


log close
```

```
*04cleaning.do

/*
*doファイル単独使用時用
global logf "log"

global dataA "main"
global dataB "sub"

global dlist = "${dataA} ${dataB}"

global startYear=2001
global endYear=2010
*/

capture log close
log using "${logf}¥04cleaning.log", replace

set more off

*データクリーニング

*企業名のクリーニング
foreach data in $dlist{
	forvalues cnt = $startYear/$endYear{
		do 04_1cleaning_name.do `data' `cnt'
		}
	}

*都道府県名・市区町村名・住所の再構築
foreach data in $dlist{
	forvalues cnt = $startYear/$endYear{
		do 04_2address_restr.do `data' `cnt'
		}
	}

*住所のクリーニング
foreach data in $dlist{
	forvalues cnt = $startYear/$endYear{
		do 04_3cleaning_address.do `data' `cnt'
		}
	}
```

```
*04cleaning_name.do

args data cnt

capture log close
log using "${logf}¥04cleaning_name`data'`cnt'.log", replace

set more off

*企業名のクリーニング

use "stata¥03`data'¥03`data'`cnt'.dta", clear

*全般的なクリーニング
display "空白の削除"
local varlist = "name state city address zip tel"
foreach cnt2 in `varlist'{
        replace `cnt2' =  subinstr(`cnt2', "　", "", .)
        replace `cnt2' =  subinstr(`cnt2', " ", "", .)
        }


*企業名のクリーニング
display "株・有"
        local varlist = "（株） (株) ㈱ 株式会社 （有） (有) ㈲ 有限会社"
        foreach cnt2 in `varlist'{
                replace name =  subinstr(name, "`cnt2'", "", .)
        }

display "小文字"
        replace name =  subinstr(name, "ァ", "ア", .)
        replace name =  subinstr(name, "ィ", "イ", .)
        replace name =  subinstr(name, "ゥ", "ウ", .)
        replace name =  subinstr(name, "ェ", "エ", .)
        replace name =  subinstr(name, "ォ", "オ", .)
        replace name =  subinstr(name, "ャ", "ヤ", .)
        replace name =  subinstr(name, "ュ", "ユ", .)
        replace name =  subinstr(name, "ョ", "ヨ", .)
        replace name =  subinstr(name, "ッ", "ツ", .)

display "全角アルファベット文字"
        replace name =  subinstr(name, "Ａ", "A", .)
        replace name =  subinstr(name, "Ｂ", "B", .)
        replace name =  subinstr(name, "Ｃ", "C", .)
        replace name =  subinstr(name, "Ｄ", "D", .)
        replace name =  subinstr(name, "Ｅ", "E", .)
        replace name =  subinstr(name, "Ｆ", "F", .)
        replace name =  subinstr(name, "Ｇ", "G", .)
        replace name =  subinstr(name, "Ｈ", "H", .)
        replace name =  subinstr(name, "Ｉ", "I", .)
        replace name =  subinstr(name, "Ｊ", "J", .)
```

```
        replace name =  subinstr(name, "Ｋ", "K", .)
        replace name =  subinstr(name, "Ｌ", "L", .)
        replace name =  subinstr(name, "Ｍ", "M", .)
        replace name =  subinstr(name, "Ｎ", "N", .)
        replace name =  subinstr(name, "Ｏ", "O", .)
        replace name =  subinstr(name, "Ｐ", "P", .)
        replace name =  subinstr(name, "Ｑ", "Q", .)
        replace name =  subinstr(name, "Ｒ", "R", .)
        replace name =  subinstr(name, "Ｓ", "S", .)
        replace name =  subinstr(name, "Ｔ", "T", .)
        replace name =  subinstr(name, "Ｕ", "U", .)
        replace name =  subinstr(name, "Ｖ", "V", .)
        replace name =  subinstr(name, "W", "W", .)
        replace name =  subinstr(name, "Ｘ", "X", .)
        replace name =  subinstr(name, "Ｙ", "Y", .)
        replace name =  subinstr(name, "Ｚ", "Z", .)

display "長音"
        replace address =  subinstr(address, "ー", "―", .)
        replace address =  subinstr(address, "-", "―", .)
        replace address =  subinstr(address, "-", "―", .)
        replace address =  subinstr(address, "－", "―", .)

display "点等"
        replace name =  subinstr(name, "・", "", .)
        replace name =  subinstr(name, "･", "", .)
        replace name =  subinstr(name, "。", "", .)
        replace name =  subinstr(name, ".", "", .)
        replace name =  subinstr(name, "、", "", .)
        replace name =  subinstr(name, ",", "", .)

save "stata¥04cleaning¥04`data'`cnt'.dta", replace
clear

log close
```

```
*04address_restr.do

args data cnt

capture log close
log using "${logf}¥04address_restr`data'`cnt'.log", replace

set more off

*都道府県・市区町村・住所を再構築

use "stata¥04cleaning¥04`data'`cnt'.dta", clear

*都道府県のエラー処理
display "都府県が入力されていないエラー処理"
display "都・府"
        replace state="東京都" if state=="東京"
        replace state="京都府" if state=="京都"
        replace state="大阪府" if state=="大阪"

display "県"
        gen ken=strpos(state, "県")
        replace ken=0 if ken==.
        replace state=state+"県" ///
                if state!="" & ken==0 & state!="東京都" ///
                & state!="京都府" & state!="大阪府" & state!="北海道"

drop ken

*一度、都道府県以下をすべて結合して再度構築する
display "バックアップの作成"
        gen state_orig=state
        gen city_orig=city
        gen address_orig=address

display "一度結合する"
        replace address=state+city+address
        replace state=""
        replace city=""


*都道府県と東京特別区の処理
display "東京都"
        gen tokyo = strpos(address, "東京都")
        replace state = "東京都" if tokyo == 1
        replace address = subinstr(address, state, "", 1) if tokyo == 1
display "東京都特別区"
        gen t_ku = strpos(address, "区") if tokyo == 1
        replace t_ku =0 if t_ku >7
        replace city = substr(address, 1, t_ku+1) if t_ku != 0
        replace address = subinstr(address, city, "", 1)
```

```
drop tokyo

display "北海道"
        gen hokkaido = strpos(address, "北海道")
        replace state = "北海道" if hokkaido == 1
        replace address = subinstr(address, state, "", 1) if hokkaido == 1
        drop hokkaido

display "京都府・大阪府"
        gen hu = strpos(address, "府")
        replace state = substr(address, 1, hu+1) if hu == 5
        replace address = subinstr(address, state, "", 1) if hu == 5
        drop hu

display "県"
        gen ken = strpos(address, "県")
        replace state = substr(address, 1, ken+1) if ken <= 8 & ken >0
        replace address = subinstr(address, state, "", 1) if ken <= 8 & ken >0
        drop ken


*市の処理
display "市名が6文字(住所に「市」を含むケース対策)"
        gen shi_long1 = strpos(address, "かすみがうら市")
                replace city = "かすみがうら市" if shi_long1 == 1
        gen shi_long2 = strpos(address, "いちき串木野市")
                replace city = "いちき串木野市" if shi_long2 == 1
        gen shi_long3 = strpos(address, "つくばみらい市")
                replace city = "つくばみらい市" if shi_long3 == 1

        gen shi_long = 0
        forvalues cnt2 = 1/3{
                replace shi_long = 1 if shi_long`cnt2' == 1
        }


display "市名に「市」を含むケース"
        gen iti_shi1 = strpos(address, "市川市")
                replace city = "市川市" if iti_shi1 == 1
        gen iti_shi2 =  strpos(address, "市原市")
                replace city = "市原市" if iti_shi2 == 1
        gen iti_shi3 =  strpos(address, "四日市市")
                replace city = "四日市市" if iti_shi3 == 1
        gen iti_shi4 =  strpos(address, "廿日市市")
                replace city = "廿日市市" if iti_shi4 == 1
        gen iti_shi5 =  strpos(address, "野々市市")
                replace city = "野々市市" if iti_shi5 == 1

        gen iti_shi = 0
        forvalues cnt2 = 1/5{
                replace iti_shi = 1 if iti_shi`cnt2' == 1
```

```
}


display "郡名に「市」を含むケース"
	gen iti_gun1 = strpos(address, "余市郡")
	gen iti_gun2 = strpos(address, "高市郡")

display "町名に「市」を含むケース"
	gen iti_tyou1 = strpos(address, "吉野郡下市")
	gen iti_tyou2 = strpos(address, "神崎郡市川")
	gen iti_tyou3 = strpos(address, "芳賀郡市貝")
	gen iti_tyou4 = strpos(address, "中新川郡上市")
	gen iti_tyou5 = strpos(address, "西八代郡市川三郷")

	gen iti = 0
	forvalues cnt2 = 1/2{
		replace iti = 1 if iti_gun`cnt2' == 1
	}

	forvalues cnt2 = 1/5{
		replace iti = 1 if iti_tyou`cnt2' == 1
	}


display "過去の自治体で「市」を含むケース(市)"
	gen old_iti_shi1 = strpos(address, "今市市")
		replace city = "今市市" if old_iti_shi1 == 1
	gen old_iti_shi2 = strpos(address, "八日市場市")
		replace city = "八日市場市" if old_iti_shi2 == 1
	gen old_iti_shi3 = strpos(address, "八日市市")
		replace city = "八日市市" if old_iti_shi3 == 1

	forvalues cnt2 = 1/3{
		replace iti_shi = 1 if old_iti_shi`cnt2' == 1
	}


display "過去の自治体で「市」を含むケース(町)"
	gen old_iti_tyou1 = strpos(address, "石川郡野々市")
		replace iti = 1 if old_iti_tyou1>0
	gen old_iti_tyou2 = strpos(address, "九戸郡種市")
		replace iti = 1 if old_iti_tyou2>0
	gen old_iti_tyou3 = strpos(address, "西八代郡市川大門")
		replace iti = 1 if old_iti_tyou3>0
	gen old_iti_tyou4 = strpos(address, "氷上郡市島")
		replace iti = 1 if old_iti_tyou4>0
	gen old_iti_tyou5= strpos(address, "芦品郡新市")
		replace iti = 1 if old_iti_tyou5>0
	gen old_iti_tyou6 = strpos(address, "鹿足郡六日市")
		replace iti = 1 if old_iti_tyou6>0
	gen old_iti_tyou7 = strpos(address, "阿波郡市場")
```

```
		replace iti = 1 if old_iti_tyou7>0
	gen old_iti_tyou8 = strpos(address, "香美郡野市")
		replace iti = 1 if old_iti_tyou8>0
	gen old_iti_tyou9 = strpos(address, "日置郡東市来")
		replace iti = 1 if old_iti_tyou9>0
	gen old_iti_tyou10 = strpos(address, "日置郡市来")
		replace iti = 1 if old_iti_tyou10>0

	forvalues cnt2 = 1/10{
		replace iti = 1 if old_iti_tyou`cnt2' == 1
	}

	drop old_iti*

display "市の処理"
	gen shi = strpos(address, "市") if t_ku == 0
		replace shi = 0 if (shi == . | shi > 12 | shi_long == 1 | iti_shi == 1| iti == 1)
		replace city = substr(address, 1, shi+1) if shi!=0


*政令市特別区の処理
display "政令市フラグ"
	gen seireishi=0
		replace seireishi=1 if (city=="大阪市" | city=="名古屋市" | city=="京都市")
		replace seireishi=1 if (city=="横浜市" |city=="神戸市" | city=="北九州市")
		replace seireishi=1 if (city=="札幌市" | city=="川崎市" | city=="福岡市")
		replace seireishi=1 if (city=="広島市" | city=="仙台市" | city=="千葉市")
		replace seireishi=1 if (city=="さいたま市" | city=="静岡市" |city=="堺市")
		replace seireishi=1 if (city=="新潟市" | city=="浜松市" | city=="岡山市")
		replace seireishi=1 if (city=="相模原市" |city=="熊本市")

display "政令市の区の処理"
	gen ku = strpos(address, "区") if seireishi==1
		replace ku=0 if ku==.


*町・村の処理
display "町名に「町」を含む町の処理"
	gen mati_tyou = strpos(address, "杵島郡大町町")
		replace city = "杵島郡大町町" if mati_tyou == 1

display "住所に「町」を含む村の処理"
	gen mati_son1 = strpos(address, "大野郡白川村")
		replace city = "大野郡白川村" if mati_son1 == 1
	gen mati_son2 = strpos(address, "刈羽郡刈羽村")
		replace city = "刈羽郡刈羽村" if mati_son2 == 1
	gen mati_son3 = strpos(address, "河沼郡湯川村")
		replace city = "河沼郡湯川村" if mati_son3 == 1
	gen mati_son4 = strpos(address, "河西郡更別村")
		replace city = "河西郡更別村" if mati_son4 == 1
	gen mati_son5 = strpos(address, "宗谷郡猿払村")
```

```
		replace city = "宗谷郡猿払村" if mati_son5 == 1
	gen mati_son6 = strpos(address, "島牧郡島牧村")
		replace city = "島牧郡島牧村" if mati_son6 == 1

	gen mati_son = 0
	forvalues cnt2 = 1/6{
		replace mati_son = 1 if mati_son`cnt2' == 1
	}


display "町・村の処理"
	gen tyou = strpos(address, "町") ///
		if t_ku == 0 & shi == 0 & shi_long==0 & iti_shi==0 ///
		& ku == 0 & mati_son == 0 & mati_tyou==0
		replace tyou = 0 if tyou == .
	gen son = strpos(address, "村") ///
		if t_ku == 0 & shi == 0 & shi_long==0 & iti_shi==0 ///
		& ku == 0 & tyou == 0 & mati_son == 0 & mati_tyou==0
		replace son = 0 if son == .

	replace city = substr(address, 1, ku+1) if ku != 0
	replace city = substr(address, 1, tyou+1) if tyou != 0
	replace city = substr(address, 1, son+1) if son != 0
	replace address = subinstr(address, city, "", 1)

	drop t_ku seireishi shi ku tyou son
	drop shi_long*
	drop iti_shi*
	drop iti*
	drop mati_tyou
	drop mati_son*


save "stata¥04cleaning¥04`data'`cnt'.dta", replace

log close
```

```
*04cleaning_address.do

args data cnt

capture log close
log using "${logf}¥04cleaning_address`data'`cnt'.log", replace

set more off

*住所のクリーニング

use "stata¥04cleaning¥04`data'`cnt'.dta", clear


display "全角数字"
            replace address =  subinstr(address, "０", "0", .)
            replace address =  subinstr(address, "１", "1", .)
            replace address =  subinstr(address, "２", "2", .)
            replace address =  subinstr(address, "３", "3", .)
            replace address =  subinstr(address, "４", "4", .)
            replace address =  subinstr(address, "５", "5", .)
            replace address =  subinstr(address, "６", "6", .)
            replace address =  subinstr(address, "７", "7", .)
            replace address =  subinstr(address, "８", "8", .)
            replace address =  subinstr(address, "９", "9", .)

display "数字を含む住所(番町)"
            replace address =  subinstr(address, "1番町", "一番町", .)
            replace address =  subinstr(address, "2番町", "二番町", .)
            replace address =  subinstr(address, "3番町", "三番町", .)
            replace address =  subinstr(address, "4番町", "四番町", .)
            replace address =  subinstr(address, "5番町", "五番町", .)
            replace address =  subinstr(address, "6番町", "六番町", .)
            replace address =  subinstr(address, "7番町", "七番町", .)
            replace address =  subinstr(address, "8番町", "八番町", .)
            replace address =  subinstr(address, "9番町", "九番町", .)
            replace address =  subinstr(address, "10番町", "十番町", .)

display "数字を含む住所(条)"
            replace address =  subinstr(address, "1条", "一条", .)
            replace address =  subinstr(address, "2条", "二条", .)
            replace address =  subinstr(address, "3条", "三条", .)
            replace address =  subinstr(address, "4条", "四条", .)
            replace address =  subinstr(address, "5条", "五条", .)
            replace address =  subinstr(address, "6条", "六条", .)
            replace address =  subinstr(address, "7条", "七条", .)
            replace address =  subinstr(address, "8条", "八条", .)
            replace address =  subinstr(address, "9条", "九条", .)
            replace address =  subinstr(address, "10条", "十条", .)

display "表記の揺れを削除する"
```

```
		replace address =  subinstr(address, "大字", "", .)
		replace address =  subinstr(address, "字", "", .)
		replace address =  subinstr(address, "丁目", "", .)
		replace address =  subinstr(address, "丁", "", .)
		replace address =  subinstr(address, "番地", "", .)
		replace address =  subinstr(address, "番", "", .)
		replace address =  subinstr(address, "号", "", .)

display "ハイフンの削除"
		replace address =  subinstr(address, "ー", "", .)
		replace address =  subinstr(address, "-", "", .)
		replace address =  subinstr(address, "-", "", .)
		replace address =  subinstr(address, "－", "", .)
		replace address =  subinstr(address, "―", "", .)

display "表記の揺れの統一"
		replace address =  subinstr(address, "上る", "上ル", .)
		replace address =  subinstr(address, "下る", "下ル", .)
		replace address =  subinstr(address, "入る", "入ル", .)
		replace address =  subinstr(address, "通り", "通", .)

display "小文字"
		replace address =  subinstr(address, "ヶ", "ケ", .)
		replace address =  subinstr(address, "ヵ", "カ", .)

display "表記の揺れの統一"
		replace address =  subinstr(address, "之", "ノ", .)

display "表記の揺れの統一"
		replace address =  subinstr(address, "霞が関", "霞ケ関", .)
		replace address =  subinstr(address, "丸の内", "丸ノ内", .)


display "郵便番号と電話番号のハイフンの削除"
local varlist = "zip tel"
		foreach cnt2 in `varlist'{
		replace `cnt2' =  subinstr(`cnt2', "-", "", .)
		}

save "stata¥04cleaning¥04`data'`cnt'.dta", replace


*照合用住所(町域まで)の作成
*町域より後が数値の場合用(漢数字表記には対応していない)
display "住所の数字より前までのみ使用"
forvalues cnt2=0/9{
		gen num`cnt2' = strpos(address, "`cnt2'")
		replace num`cnt2'=. if num`cnt2'==0
}

		gen num = min(num0, num1, num2, num3, num4, num5, num6, num7, num8, num9)
```

```
        replace address =  substr(address, 1, num-1)

display "最後の「町」を落とす"
        gen len=length(address)
        gen tyou_posi=strpos(address,"町")
        replace address=substr(address, 1, len-2) if len-1==tyou_posi
        drop len tyou_posi

save "stata¥04cleaning¥04`data'`cnt'_shortadd.dta", replace
clear

log close
```

```
*05data_check.do

/*
*doファイル単独使用時用
global logf "log"

global dataA "main"
global dataB "sub"

global dlist = "${dataA} ${dataB}"

global startYear=2001
global endYear=2010
*/

capture log close
log using "${logf}¥05data_check.log", replace

set more off

*データの整理とエラーチェック
foreach data in $dlist{
	forvalues cnt = $startYear/$endYear{

	display "`data'の`cnt'年のチェック"
	use "stata¥04cleaning¥04`data'`cnt'.dta", clear

	display "ID重複サンプルの処理"
		duplicates report ID
		duplicates tag ID,gen(dup_ID)
		drop if dup_ID>0
		drop dup_ID

	display "ID桁数のチェック"
		gen len_ID = length(ID)
		tabulate len_ID

	display "IDが数値に変換可能かのチェック"
		gen id = ID
		destring id, replace force
		count if id == .
		list id ID if id == .

	display "企業名・都道府県・市区町村が同じサンプルの処理"
		duplicates report name state city
		duplicates tag name state city, gen(dup_name_state_city)
		drop if dup_name_state_city>0
		drop dup_name_state_city

	display "照合に使用する変数の空欄チェック"
		display "企業名"
```

```
                count if name == ""
        display "都道府県"
                count if state == ""
        display "市区町村"
                count if city == ""
        display "住所"
                count if address == ""
        display "zip"
                count if zip == ""
        display "tel"
                count if tel == ""

save "stata¥05data_check¥05`data'`cnt'.dta", replace
clear

*住所(町域まで)による照合用ファイルの作成
use "stata¥04cleaning¥04`data'`cnt'_shortadd.dta", clear

display "ID重複サンプルの処理"
        duplicates report ID
        duplicates tag ID,gen(dup_ID)
        drop if dup_ID>0
        drop dup_ID

display "ID桁数のチェック"
        gen len_ID = length(ID)
        tabulate len_ID

display "IDが数値に変換可能かのチェック"
        gen id = ID
        destring id, replace force
        count if id == .
        list id ID if id == .

display "企業名・都道府県・市区町村が同じサンプルの処理"
        duplicates report name state city
        duplicates tag name state city, gen(dup_name_state_city)
        drop if dup_name_state_city>0
        drop dup_name_state_city

display "照合に使用する変数の空欄チェック"
        display "企業名"
                count if name == ""
        display "都道府県"
                count if state == ""
        display "市区町村"
                count if city == ""
        display "住所"
                count if address == ""

save "stata¥05data_check¥05`data'`cnt'_shortadd.dta", replace
```

```
		clear

		*企業名・資本金による照合用ファイルの作成
		use"stata¥05data_check¥05`data'`cnt'.dta", clear

		display "資本金2億円以上に絞る(capitalの単位100万円)"
			destring capital, replace
			drop if capital < 200
			drop if capital == .

		display "重複サンプルの処理"
			duplicates report name capital
			duplicates tag name capital, gen(dup_name_cap)
			drop if dup_name_cap>0
			drop dup_name_cap

		save "stata¥05data_check¥05`data'`cnt'_cap.dta", replace

		}
}


log close
```

```
*06panel.do

/*
*doファイル単独使用時用
global logf "log"

global dataA "kikatsu"
global dataB "kaiji"

global dlist = "${dataA} ${dataB}"

global startYear=2001
global endYear=2010
*/

capture log close
log using "${logf}¥06panel.log", replace

set more off


*データの整理
foreach data in $dlist{
	forvalues cnt = $startYear/$endYear{

	display "`data'の`cnt'年のデータの整理"
	*一般的な照合用ファイルの整理
	use "stata¥05data_check¥05`data'`cnt'.dta", clear

	rename ID `data'ID
	drop *orig len_ID id

	save "stata¥06cross¥06`data'`cnt'.dta", replace
	clear

	*住所(町域まで)による照合用ファイルの整理
	use "stata¥05data_check¥05`data'`cnt'_shortadd.dta", clear

	rename ID `data'ID
	drop *orig len_ID id

	save "stata¥06cross¥06`data'`cnt'_shortadd.dta", replace
	clear

	*企業名・資本金による照合用ファイルの整理
	use "stata¥05data_check¥05`data'`cnt'_cap.dta", clear

	rename ID `data'ID
	drop len_ID id

	*照合に使用しない変数名の調整
```

```
		rename state `data'state
		rename city `data'city
		rename address `data'address
		drop *orig

		save "stata¥06cross¥06`data'`cnt'_cap.dta", replace
		clear

		}
}


*各パネル・データの構築
foreach data in $dlist{
		display "`data'のパネル・データ構築"
		use "stata¥06cross¥06`data'${startYear}.dta", clear
			local cnt = $startYear + 1
		while `cnt' <= $endYear{
			display "`cnt'年の追加"
			append using "stata¥06cross¥06`data'`cnt'.dta"
			local cnt = `cnt' + 1

		}

		sort `data'ID year

		rename * `data'*
		rename `data'`data'ID `data'ID
		rename `data'year year

		gen `data'id = `data'ID
		destring `data'id, replace

		duplicates report `data'id year
		duplicates tag `data'id year, gen(dup_id)
		drop if dup_id>0
		drop dup_id

		save "stata¥06panel¥06`data'_panel_long.dta", replace

		display "回答状況の確認"
		xtset `data'id year
		xtdescribe, patterns(100)

		reshape wide ///
			`data'ID `data'name `data'state `data'city ///
			`data'address `data'zip `data'tel `data'capital, ///
			i(`data'id) j(year)

		save "stata¥06panel¥06`data'_panel_wide.dta", replace
		clear
```

```
}

log close
```

```
*07match.do

/*
*doファイル単独使用時用
global logf "log"

global dataA "main"
global dataB "sub"

global dlist = "${dataA} ${dataB}"

global startYear=2001
global endYear=2010
*/

capture log close
log using "${logf}¥07match.log", replace

set more off

*照合作業
*各年ごとに逐次照合する
display "各年ごとに${dataA}と${dataB}を照合してコード対応表を作成"
forvalues cnt = $startYear/$endYear{
display "`cnt'年の企業名・都道府県・市区町村・住所・zip・telの照合"
use "stata¥06cross¥06${dataA}`cnt'.dta", clear
        merge 1:1 name state city address zip tel ///
        using "stata¥06cross¥06${dataB}`cnt'.dta"

        rename _merge merge1_`cnt'
        drop if merge1_`cnt' == 2

        gen ${dataA}id = ${dataA}ID
        destring ${dataA}id, replace
        gen ${dataB}id = ${dataB}ID
        destring ${dataB}id, replace

        replace ${dataB}id = . if (state=="" | city=="" | address=="" | zip=="" | tel=="")

        keep ${dataA}id ${dataB}id merge1_`cnt'
        count if ${dataB}id != .

save "stata¥07match¥01match1`cnt'.dta", replace
clear
}


forvalues cnt = $startYear/$endYear{
display "`cnt'年の企業名・都道府県・市区町村・住所・zipの照合"
use "stata¥06cross¥06${dataA}`cnt'.dta", clear
        merge 1:1 name state city address zip ///
```

```
		using "stata¥06cross¥06${dataB}`cnt'.dta"

		rename _merge merge2_`cnt'
		drop if merge2_`cnt' == 2

		gen ${dataA}id = ${dataA}ID
		destring ${dataA}id, replace
		gen ${dataB}id = ${dataB}ID
		destring ${dataB}id, replace

		replace ${dataB}id = .  if (state=="" | city=="" | address=="" | zip=="" )

		keep ${dataA}id ${dataB}id merge2_`cnt'
		count if ${dataB}id != .

save "stata¥07match¥01match2`cnt'.dta", replace
clear
}


forvalues cnt = $startYear/$endYear{
display "`cnt'年の企業名・都道府県・市区町村・住所・telの照合"
use "stata¥06cross¥06${dataA}`cnt'.dta", clear
		merge 1:1 name state city address tel  ///
		using "stata¥06cross¥06${dataB}`cnt'.dta"

		rename _merge merge3_`cnt'
		drop if merge3_`cnt' == 2

		gen ${dataA}id = ${dataA}ID
		destring ${dataA}id, replace
		gen ${dataB}id = ${dataB}ID
		destring ${dataB}id, replace

		replace ${dataB}id = . if (state=="" | city=="" | address=="" | tel=="")

		keep ${dataA}id ${dataB}id merge3_`cnt'
		count if ${dataB}id != .

save "stata¥07match¥01match3`cnt'.dta", replace
clear
}


forvalues cnt = $startYear/$endYear{
display "`cnt'年の企業名・都道府県・市区町村・住所の照合"
use "stata¥06cross¥06${dataA}`cnt'.dta", clear
		merge 1:1 name state city address  ///
		using "stata¥06cross¥06${dataB}`cnt'.dta"

		rename _merge merge4_`cnt'
```

```
		drop if merge4_`cnt' == 2

		gen ${dataA}id = ${dataA}ID
		destring ${dataA}id, replace
		gen ${dataB}id = ${dataB}ID
		destring ${dataB}id, replace

		replace ${dataB}id = . if (state=="" | city=="" | address=="" )

		keep ${dataA}id ${dataB}id merge4_`cnt'
		count if ${dataB}id != .

save "stata¥07match¥01match4`cnt'.dta", replace
clear
}


forvalues cnt = $startYear/$endYear{
display "`cnt'年の企業名・都道府県・市区町村・zip・telの照合"
use "stata¥06cross¥06${dataA}`cnt'.dta", clear
		merge 1:1 name state city zip tel ///
		using "stata¥06cross¥06${dataB}`cnt'.dta"

		rename _merge merge5_`cnt'
		drop if merge5_`cnt' == 2

		gen ${dataA}id = ${dataA}ID
		destring ${dataA}id, replace
		gen ${dataB}id = ${dataB}ID
		destring ${dataB}id, replace

		replace ${dataB}id = . if (state=="" | city=="" | zip=="" | tel=="")

		keep ${dataA}id ${dataB}id merge5_`cnt'
		count if ${dataB}id != .

save "stata¥07match¥01match5`cnt'.dta", replace
clear
}


forvalues cnt = $startYear/$endYear{
display "`cnt'年の企業名・都道府県・市区町村・zipの照合"
use "stata¥06cross¥06${dataA}`cnt'.dta", clear
		merge 1:1 name state city zip ///
		using "stata¥06cross¥06${dataB}`cnt'.dta"

		rename _merge merge6_`cnt'
		drop if merge6_`cnt' == 2

		gen ${dataA}id = ${dataA}ID
```

```
		destring ${dataA}id, replace
		gen ${dataB}id = ${dataB}ID
		destring ${dataB}id, replace

		replace ${dataB}id = . if (state=="" | city=="" | zip=="" )

		keep ${dataA}id ${dataB}id merge6_`cnt'
		count if ${dataB}id != .

save "stata¥07match¥01match6`cnt'.dta", replace
clear
}


forvalues cnt = $startYear/$endYear{
display "`cnt'年の企業名・都道府県・市区町村・telの照合"
use "stata¥06cross¥06${dataA}`cnt'.dta", clear
		merge 1:1 name state city tel ///
		using "stata¥06cross¥06${dataB}`cnt'.dta"

		rename _merge merge7_`cnt'
		drop if merge7_`cnt' == 2

		gen ${dataA}id = ${dataA}ID
		destring ${dataA}id, replace
		gen ${dataB}id = ${dataB}ID
		destring ${dataB}id, replace

		replace ${dataB}id = . if (state=="" | city=="" | tel=="")

		keep ${dataA}id ${dataB}id merge7_`cnt'
		count if ${dataB}id != .

save "stata¥07match¥01match7`cnt'.dta", replace
clear
}

forvalues cnt = $startYear/$endYear{
display "`cnt'年の企業名・都道府県・市区町村・住所(町域まで)の照合"
use "stata¥06cross¥06${dataA}`cnt'_shortadd.dta", clear
		merge 1:1 name state city address  ///
		using "stata¥06cross¥06${dataB}`cnt'_shortadd.dta"

		rename _merge merge8_`cnt'
		drop if merge8_`cnt' == 2

		gen ${dataA}id = ${dataA}ID
		destring ${dataA}id, replace
		gen ${dataB}id = ${dataB}ID
		destring ${dataB}id, replace
```

```
		replace ${dataB}id = . if (state=="" | city=="" | address=="" )

		keep ${dataA}id ${dataB}id merge8_`cnt'
		count if ${dataB}id != .

save "stata¥07match¥01match8`cnt'.dta", replace
clear
}


display "照合結果の確認"
forvalues cnt1 = $startYear/$endYear{
		forvalues cnt2 = 1/8{
		use  "stata¥07match¥01match`cnt2'`cnt1'.dta", clear
				display "`cnt1'年のmerge`cnt2'接続数'"
				count if ${dataB}id != .
		clear

		}
}

forvalues cnt = $startYear/$endYear{
		display "`cnt'年の照合結果の統合"
		use "stata¥07match¥01match1`cnt'.dta", clear
		forvalues cnt2 = 2/8{
				merge 1:1 ${dataA}id using "stata¥07match¥01match`cnt2'`cnt'.dta", update
					drop _merge
				}

		display "`cnt'年の照合による接続数の合計"
		count if ${dataB}id != .

		save "stata¥07match¥02match`cnt'.dta", replace
		clear
}


display "企業名・資本金の照合処理"
display "各年ごとに${dataA}と${dataB}を照合してコード対応表を作成"
forvalues cnt = $startYear/$endYear{
display "`cnt'年の企業名・資本金の照合"
use "stata¥06cross¥06${dataA}`cnt'_cap.dta", clear
		merge 1:1 name capital ///
		using "stata¥06cross¥06${dataB}`cnt'_cap.dta"

		rename _merge merge_cap`cnt'
		drop if merge_cap`cnt' == 2

		gen ${dataA}id = ${dataA}ID
		destring ${dataA}id, replace
		gen ${dataB}id = ${dataB}ID
```

```
        destring ${dataB}id, replace

        count if ${dataB}id != .
        count if ${dataB}id != . & ${dataA}state!=${dataB}state
        keep ${dataA}id ${dataB}id merge_cap`cnt'

save "stata¥07match¥02match_cap`cnt'.dta", replace
clear
}


log close
```

```
*08deploy.do

/*
*doファイル単独使用時用
global logf "log"

global dataA "main"
global dataB "sub"

global dlist = "${dataA} ${dataB}"

global startYear=2001
global endYear=2010
*/

capture log close
log using "${logf}¥08deploy.log", replace

set more off


*照合結果に基づいてwide形式のパネル・データを接続した上で補正を行う
forvalues cnt1 = $startYear/$endYear{
	display "`cnt1'年の照合結果を基準とした接続"
	use "stata¥06panel¥06${dataA}_panel_wide.dta", clear
		merge 1:1 ${dataA}id using "stata¥07match¥02match`cnt1'.dta"

		keep if _merge == 3
		drop _merge
		keep if ${dataB}id != .

	merge 1:1 ${dataB}id using "stata¥06panel¥06${dataB}_panel_wide.dta"

		keep if _merge == 3
		drop _merge


	display "`cnt1'年の照合結果を企業名が整合的な年に展開する"
	forvalues cnt2 = $startYear/$endYear{
		gen c_id`cnt2' = ${dataB}id ///
			if ${dataA}name`cnt2' == ${dataB}name`cnt2' ///
			& ${dataA}name`cnt2' != "" & ${dataB}name`cnt2' != ""

		display "`cnt2'年の企業名が整合的で展開された件数"
		 count if c_id`cnt2' != .

		gen inc`cnt2' = 1 ///
			if  ${dataA}name`cnt2' != ${dataB}name`cnt2' ///
			& ${dataA}name`cnt2' != "" & ${dataB}name`cnt2' != ""
		replace inc`cnt2' = 0 if inc`cnt2' == .
```

```
		display "`cnt2'年の企業名が整合的でなかった件数"
		count if inc`cnt2' != .

	}


	display "`cnt1'年基準の展開結果に基づく補間処理"
	display "展開結果のフラグ作成"
		gen match_total = 0
	forvalues cnt2 = $startYear/$endYear{
		gen match`cnt2' = 1 if c_id`cnt2' != .
		replace match`cnt2' = 0 if match`cnt2' == .
		replace match_total = match_total + match`cnt2'

	}

	display "前後で接続が行われている期間中は補間処理を行う"
		gen survive = 0
	forvalues cnt2 = $startYear/$endYear{
		replace survive = survive + match`cnt2'
		replace c_id`cnt2' = ${dataB}id ///
			if (survive < match_total & survive > 0 & c_id`cnt2' == . ///
			& ${dataA}name`cnt2' != "" & ${dataB}name`cnt2' != "")

	}


	display "`cnt1'年基準での他の年の不整合チェック"
		gen inc_total = 0
	forvalues cnt2 = $startYear/$endYear{
		replace inc_total = inc_total + inc`cnt2'

	}

	display "`cnt1'年しか接続できず他の年で不整合が起きているものを落とす"
	replace c_id`cnt1' = . if (match_total == 1 & inc_total > 0)

	save  "stata¥08deploy¥00deploy`cnt1'.dta", replace

	keep ${dataA}id c_id*
	save "stata¥08deploy¥01deploy`cnt1'.dta", replace

	keep ${dataA}id c_id`cnt1'
	save  "stata¥08deploy¥02deploy`cnt1'.dta", replace

	clear
}


log close
```

```
*09update.do

/*
*doファイル単独使用時用
global logf "log"

global dataA "main"
global dataB "sub"

global dlist = "${dataA} ${dataB}"

global startYear=2001
global endYear=2010
*/

capture log close
log using "${logf}¥09update.log", replace

set more off


*照合結果を基にベースを作成した上で補正結果を反映させる
display "ベースの作成"
use "stata¥06panel¥06${dataA}_panel_wide.dta", clear
                keep ${dataA}id
        forvalues cnt = $startYear/$endYear{
                merge 1:1 ${dataA}id using "stata¥08deploy¥02deploy`cnt'.dta"
                        drop _merge

        }

        display "補正結果の統合"
        forvalues cnt = $startYear/$endYear{
                merge 1:1 ${dataA}id using "stata¥08deploy¥01deploy`cnt'.dta", update
                        rename _merge update`cnt'

        }

        *確認用データの保存
        save  "stata¥09update¥00update.dta", replace


        display "不整合が生じるサンプルの処理"
        forvalues cnt = $startYear/$endYear{
                display "整合性のとれないサンプルを落とす"
                        list if update`cnt' == 5
                        drop if update`cnt' == 5

                display "重複が生じるサンプルを落とす"
                        duplicates tag c_id`cnt', gen(dup_${dataB}`cnt')
                        list if dup_${dataB}`cnt' > 0 & c_id`cnt' != .
```

```
                        drop if dup_${dataB}`cnt' > 0 & c_id`cnt' != .
                        drop dup_${dataB}`cnt'

                }


display "照合に基づく最終的な接続数の確認"
forvalues cnt = $startYear/$endYear{
        count if c_id`cnt' != .

        }

save  "stata¥09update¥01update.dta", replace

        keep ${dataA}id c_id*

save  "stata¥09update¥02update.dta", replace


log close
```

```
*10add.do

/*
*doファイル単独使用時用
global logf "log"

global dataA "main"
global dataB "sub"

global dlist = "${dataA} ${dataB}"

global startYear=2001
global endYear=2010
*/

capture log close
log using "${logf}¥10add.log", replace

set more off


*企業名・資本金による照合結果の追加
forvalues cnt = $startYear/$endYear{
	display "補正したものを再度分割する"
	use "stata¥09update¥02update.dta", clear
	keep ${dataA}id c_id`cnt'
	rename c_id`cnt' ${dataB}id

	merge 1:1 ${dataA}id using "stata¥07match¥02match_cap`cnt'.dta", update
	rename _merge merge_addcap`cnt'

	display "照合時にすでにはじかれているサンプルを落とす"
	drop if merge_addcap`cnt' == 2

	display "重複が生じてないかチェック"
	duplicates report ${dataB}id
	duplicates tag ${dataB}id, gen(dup_cap`cnt')

	display "重複が生じている場合は照合結果を優先"
	drop if dup_cap`cnt' > 0 & merge_addcap`cnt' == 4

	count if ${dataB}id != .

	save "stata¥10add¥00add_cap`cnt'.dta", replace
	clear

}


*照合結果に基づいてwide形式のパネル・データを接続した上で補正を行う
forvalues cnt1 = $startYear/$endYear{
```

```
display "`cnt1'年の照合結果を基準とした接続"
use "stata¥06panel¥06${dataA}_panel_wide.dta", clear
        merge 1:1 ${dataA}id using "stata¥10add¥00add_cap`cnt1'.dta"

        keep if _merge == 3
        drop _merge
        keep if ${dataB}id != .

merge 1:1 ${dataB}id using "stata¥06panel¥06${dataB}_panel_wide.dta"

        keep if _merge == 3
        drop _merge


display "`cnt1'年の照合結果を企業名が整合的な年に展開する"
forvalues cnt2 = $startYear/$endYear{
        gen c_id`cnt2' = ${dataB}id ///
                if ${dataA}name`cnt2' == ${dataB}name`cnt2' ///
                & ${dataA}name`cnt2' != "" & ${dataB}name`cnt2' != ""

        display "`cnt2'年の企業名が整合的で展開された件数"
         count if c_id`cnt2' != .

        gen inc`cnt2' = 1 ///
                if  ${dataA}name`cnt2' != ${dataB}name`cnt2' ///
                & ${dataA}name`cnt2' != "" & ${dataB}name`cnt2' != ""
        replace inc`cnt2' = 0 if inc`cnt2' == .

        display "`cnt2'年の企業名が整合的でなかった件数"
        count if inc`cnt2' != .

}


display "`cnt1'年基準の展開結果に基づく補間処理"
display "展開結果のフラグ作成"
        gen match_total = 0
forvalues cnt2 = $startYear/$endYear{
        gen match`cnt2' = 1 if c_id`cnt2' != .
        replace match`cnt2' = 0 if match`cnt2' == .
        replace match_total = match_total + match`cnt2'

}

display "前後で接続が行われている期間中は補間処理を行う"
        gen survive = 0
forvalues cnt2 = $startYear/$endYear{
        replace survive = survive + match`cnt2'
        replace c_id`cnt2' = ${dataB}id ///
                if (survive < match_total & survive > 0 & c_id`cnt2' == . ///
                & ${dataA}name`cnt2' != "" & ${dataB}name`cnt2' != "")
```

```
	}


	display "`cnt1'年基準での他の年の不整合チェック"
		gen inc_total = 0
	forvalues cnt2 = $startYear/$endYear{
		replace inc_total = inc_total + inc`cnt2'

	}

	display "`cnt1'年しか接続できず他の年で不整合が起きているものを落とす"
	replace c_id`cnt1' = . if (match_total == 1 & inc_total > 0)

	save  "stata¥10add¥00deploy`cnt1'.dta", replace

	keep ${dataA}id c_id*
	save "stata¥10add¥01deploy`cnt1'.dta", replace

	clear
}



display "補正結果の統合"
use "stata¥09update¥02update.dta", clear
	forvalues cnt = $startYear/$endYear{
		merge 1:1 ${dataA}id using "stata¥10add¥01deploy`cnt'.dta", update
			rename _merge add`cnt'

	}

	*確認用データの保存
	save  "stata¥10add¥00add.dta", replace


	display "不整合が生じるサンプルの処理"
	forvalues cnt = $startYear/$endYear{
		display "整合性のとれないサンプルを落とす"
			drop if add`cnt' == 5

		display "重複が生じるサンプルを落とす"
			duplicates tag c_id`cnt', gen(dup_${dataB}`cnt')
			drop if dup_${dataB}`cnt' > 0 & c_id`cnt' != .
			drop dup_${dataB}`cnt'

		}


display "最終的な接続数の確認"
forvalues cnt = $startYear/$endYear{
```

```
            count if c_id`cnt' != .

            }

save  "stata¥10add¥01add.dta", replace

            keep ${dataA}id c_id*

save  "stata¥10add¥02add.dta", replace


log close
```

```
*11connect.do

/*
*doファイル単独使用時用
global logf "log"

global dataA "main"
global dataB "sub"

global dlist = "${dataA} ${dataB}"

global startYear=2001
global endYear=2010
*/

capture log close
log using "${logf}¥11connect.log", replace

set more off


*データの接続処理

*コンバータのlong形式への変換

use  "stata¥10add¥02add.dta", clear

        reshape long c_id, i(${dataA}id) j(year)

save "stata¥cnv_long.dta", replace
clear

display "${dataA}をベースに${dataB}を接続する"
use "stata¥06panel¥06${dataA}_panel_long.dta", clear
        merge 1:1 ${dataA}id year using "stata¥cnv_long.dta"
                keep if _merge == 3
                drop _merge

                keep if c_id != .
                rename c_id ${dataB}id

        merge 1:1 ${dataB}id year using "stata¥06panel¥06${dataB}_panel_long.dta"
                keep if _merge == 3
                drop _merge

save "stata¥panel_long.dta", replace

log close
```