粘土細工アプローチによる社会調査のデータクリーニングシステムの開発

―社会調査実習での活用を中心に―

羅　一等

Development of a Data Cleaning System for Social Surveys by the Clay Modeling Approach ―For the Use in Social Research Practicum―

NA, Ildeung

要旨：本論文の目的は、粘土細工アプローチという社会調査のデータクリーニング技法を具現するデータクリーニングシステムDCSSを開発し、それを社会調査実習で活用することを通して粘土細工アプローチおよびDCSSの利点を明らかにすることである。データクリーニングでは誤りの原因と正しい修正方法を突き止めるために、回答者の回答を、それを観察する者にとって理解可能なストーリーに再構成することが不可欠である。しかし、従来のデータクリーニングでは、チェックすべき項目別に作業を分担する、いわば変数中心のデータクリーニングが行われていたため、回答の文脈的理解と人物像の文脈的構成が妨げられてしまい、参照範囲と修正範囲が制限されるという問題があった。粘土細工アプローチはこのような弊害を克服するために提案されたものである。粘土細工アプローチは、異常値を検出する作業とそれを修正する作業とを区別して両者を独立的なプロセスにし、ケース単位で修正を施していくというのが特徴である。本稿では、2017年度と2018年度に専修大学で開講された社会調査実習で粘土細工アプローチとDCSSを導入してデータの整理作業を行った事例を紹介し、調査データの整理において粘土細工アプローチとDCSSを導入することの利点と、DCSSを利用して調査データの整理を行う方法について説明する。

キーワード：データクリーニング、粘土細工アプローチ、DCSS、社会調査実習

1　本稿の目的と構成

本稿の目的は、粘土細工アプローチ（clay modeling approach）という社会調査のデータクリーニング技法を具現するデータクリーニングシステムを開発し、それを社会調査実習で活用することを通して、粘土細工アプローチおよびそれを具現するシステムの利点を明らかにすることである。

粘土細工アプローチは、複雑な社会調査のデータクリーニングに対応するために、保田が開発したデータクリーニングの技法である（保田 2010、 2011、 2012、 2017、 2018；菅澤・保田 2018）。保田が言う「複雑な社会調査」とは、各種センサスといった公的統計と比べて、一般的な社会学研究で行われる社会調査の特徴を言い表したものである。それはつまり、社会学研究で行われる社会調査は公的統計に比べて、標本サイズが小さく、調査項目の数が多く、調査票の構造が入り組んでいることを意味する。このような複雑な社会調査では、データクリーニングにおける特有の困難がある。粘土細工アプローチはこの困難に対応するために提唱されたものである。そして粘土細工アプローチは、JGSS（日本版総合的社会調査）–2009ライフコース調査、NFRJ（全国家族調査パネルスタディ）–08Panel、2015年SSM（社会階層と社会移動全国調査）調査といった日本を代表する社会調査のデータクリーニングに適用され、その有効性が確かめられている（保田 2018）。粘土細工アプローチは、今後も発展を続けて、将来は社会調査のデータクリーニング方法の主流になることが期待される。

本研究では、「DCSS（Data Cleaning System for Social Surveys）」という粘土細工アプローチを具現するデータクリーニングシステムを開発して、粘土細工アプローチおよびDCSSを使用することが従来のデータクリーニングに比べてどのような利点があるのかを明らかにする。

DCSSは、簡単な操作で粘土細工アプローチによるデータクリーニングを実現することを目標にして開発されたアプリケーションである。「簡単な操作」とは具体的に、大学学部生が1時間ほどのインストラクションを受けて一通りの操作ができることを意味する。このような目標を設定した理由は、既存のシステムがその運用と

受稿日2018年11月26日　受理日2017年12月3日
専修大学人間科学部兼任講師

操作において一定の専門性を要求しており、社会調査実習科目の履修生のような社会調査の初心者がそれを使用することが難しかったからである。

例えば、保田は上述の日本の代表的な社会調査で粘土細工アプローチによるデータクリーニングを行うために、マイクロソフト社の表計算ソフトExcelをベースにしたツールを開発した。このツールは現在公開準備中であるが、その特徴と操作方法については文献にまとめられているので、文献を通して把握できる（保田 2018）。保田のツールはExcelをベースにしているので、ユーザーが必要に応じて機能をカスタマイズできるという特徴がある。つまり、データのいわば「くせ」に柔軟に対応できるという利点がある。しかし、こういった柔軟な対応は基本的にExcelの関数やVBAなどについて高度の専門知識を有している研究者を想定したものであり、そのような能力を大学学部生に期待することは難しい。筆者の教育経験上、学生の中には社会調査実習で初めてExcelを操作する人も少なくない。DCSSは、そのような学生でも粘土細工アプローチによるデータクリーニングが行えるように開発されたものである。

他にも、例えばCSPro（Census and Survey Processing System）という無償のアプリケーションを使用してデータクリーニングを行うことも可能である。CSProは、米国センサス局、ORC Macro International、Serpro S.A.が共同開発したアプリケーションで、CAPI（computer assisted personal interviewing）を含むデータ収集と整理過程で活用することができ、簡単な分析も行える優れたアプリケーションである。そして、CSProで粘土細工アプローチによるデータクリーニングを行うことも不可能ではない。しかし、ここでは具体的な説明は割愛するが、CSProの操作に慣れるためには訓練が必要であり、粘土細工アプローチの適用にも制約がある。

以上の問題意識から、筆者は2016年から粘土細工アプローチのための専用アプリケーションの開発に着手した。アプリケーションの開発は、ケースの数約6,500、変数の数約3,000のパネルデータのデータクリーニング作業と並行して進められた[1)]。当初は保田と同様にExcelをベースにしたツールを開発したが、上述した問題点から開発したツールは一般公開せず、研究グループ内でのみ使用される内部用のものとした。そして、2017年から一般公開を念頭に新しいJAVAアプリケーションの開発を行った。それがDCSSのプロトタイプである（羅 2017b）。以後、筆者が2017年度に担当した社会調査実習科目の演習でDCSSを導入するなどを経て（羅 2017a）、2018年にDCSSを一般公開した[2)]。

本稿では、粘土細工アプローチおよびDCSSの特徴と利点について説明し、社会調査実習でDCSSを活用する方法を説明する。本稿の構成は次の通りである。第2節では、従来のデータクリーニングの仕方を概略し、その弊害を指摘した上で、粘土細工アプローチの特徴と利点、具体的な手順を説明する。第3節では、DCSSの概要と特徴、データの読み込みと書き出しの方法、基本的な操作方法について説明する。第4節では、データ入力、コーディング、データクリーニングでDCSSを活用する方法を、社会調査実習での活用例とともに説明する。第5節では、実際に社会調査実習でDCSSを活用して調査データの整理を行った学生たちの感想に基づいて、粘土細工アプローチおよびDCSSの利点と課題を述べる。

2 データクリーニングの理論：粘土細工アプローチ

2.1 従来のデータクリーニング

従来のデータクリーニングでは、データの誤りを見つけて修正を施す作業を、いわゆる「単純集計チェック」と「論理チェック」の2つに大別して行う（盛山 2004；大谷ほか 2013）。単純集計チェックでは、存在しないコードが値になっていないかをチェックし、論理チェックでは、蓋然性の低い回答の組み合わせがないかをチェックする。これらの作業に用いるツールは、単純集計やクロス表を出力するための統計分析ソフトと、データファイルを修正するためのテキストエディタや表計算ソフトなどである。

従来のデータクリーニングのプロセスを、架空の例で説明しよう。

まず、単純集計チェックから説明する。単純集計チェックでは、最初に統計分析ソフトで全ての変数の度数分布表を出力し、存在しないコードが値になっていないかチェックする。問題が見つかった場合、統計分析ソフトのケース選択機能や度数分布表などを利用して誤った値が入力されたケースを突き止める。問題のケースと変数が特定できたら、値を修正する。値を修正する際に、必要であれば調査票や対象者リストにまで遡って、誤りの原因と修正方法を突き止める。表1は、そのプロセスを架空データで再現したものである。

変数*var_gakureki*は学歴変数で、そのコードは「1小卒、2中卒、3高卒、4大卒、9無回答」である。変数*var_id*はケースIDである。

表1　従来のデータクリーニングにおける単純集計チェックの例

var_gakureki

値	度数
0	1
1	195
2	27
3	19
4	41
9	17
合計	300

(a)
度数分布表を出力

→

var_id

値	度数
23	1
合計	1

(b)
ケース ID を見つける

→

var_gakureki

値	度数
1	195
2	28
3	19
4	41
9	17
合計	300

(c)
値を修正

表1（a）では、*var_gakureki* の度数分布表を出力し、存在しないコードが値になっていないかをチェックした。その結果、破線で囲われた箇所が示している通り、0という存在しないコードが値として入力されているケースが一つあることが分かった。表1（b）は、*var_gakureki* の値が0と入力されているケースを選択し、度数分布表を出力したものである。ここから問題のケースのIDが23であることが確認できる。次は、調査票を確認し、誤ったコードが入力された原因と修正方法を突き止める。表1（c）は、ケースID23の *var_gakureki* の値を0から2に修正し、*var_gakureki* の度数分布表を出力したものである。一般的な単純集計チェックはこのようなプロセスで行われる。

次に、論理チェックについて説明する。論理チェックでは、最初にチェックしたい変数の組み合わせからなるクロス表を出力する。そして、蓋然性の低い回答の組み合わせがないかをチェックする。問題が見つかった場合、統計分析ソフトのケース選択機能や度数分布表などを利用して誤った値が入力されたケースを突き止める。問題のケースと変数が特定できたら、値を修正する。値を修正する際に、必要であれば調査票や対象者リストにまで遡って、誤りの原因と修正方法を突き止める。表2は、そのプロセスを架空データで再現したものである。

変数 *var_kekkon* は婚姻状況変数で、そのコードは「1　非婚、2　現在配偶者がいる、3　離別、4　死別、9　無回答」である。変数 *var_kodomoninzu* は、子供の人数変数である。

表2（a）では、*var_kekkon* と *var_kodomoninzu* のクロス表を出力し、蓋然性の低い回答の組み合わせがないかをチェックした。その結果、破線で囲われた箇所が

表2　従来のデータクリーニングにおける論理チェックの例

		var_kodomoninzu			
		0	1	2	合計
var_kekkon	1	35	1	0	36
	2	18	99	98	215
	3	10	18	10	38
	4	3	2	6	11
	9	0	0	0	0
	合計	66	120	114	300

(a)
クロス表を出力

→

var_id

値	度数
23	1

(b)
ケース ID を見つける

→

		var_kodomoninzu			
		0	1	2	合計
var_kekkon	1	35	0	0	36
	2	18	100	98	216
	3	10	18	10	38
	4	3	2	6	11
	9	0	0	0	0
	合計	66	120	114	300

(c)
値を修正

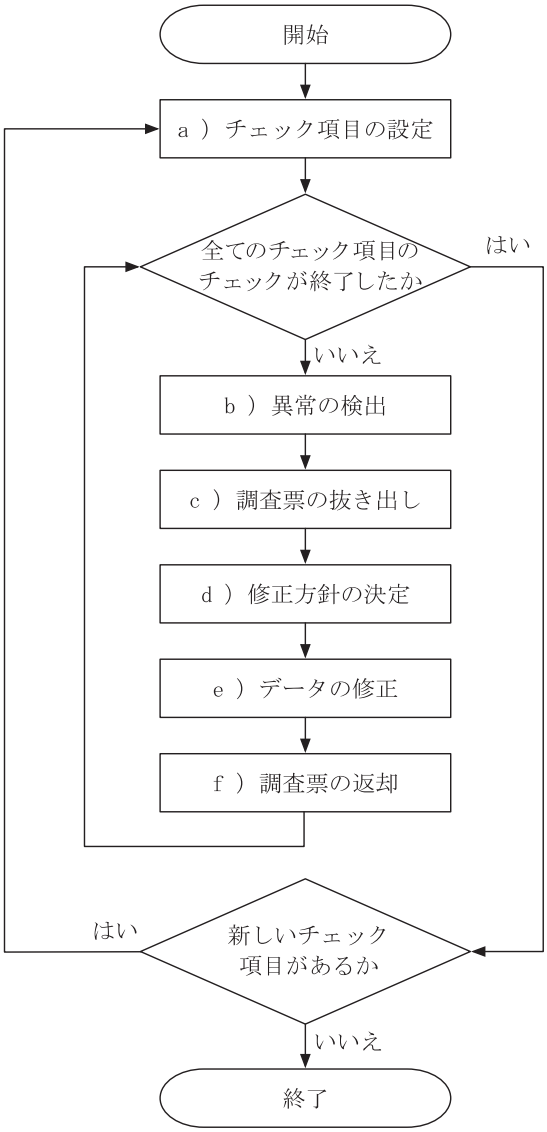


図１　従来のデータクリーニングのフローチャート

示している通り、非婚なのに子供が１人いるという回答になっているケースが一つあることが分かった。表２（b）は、*var_kekkon* の値が１、*var_kodomoninzu* の値が１のケースを選択し、度数分布表を出力したものである。ここから問題のケースのIDが23であることが確認できる。次は、調査票を確認し、誤ったコードが入力された原因と修正方法を突き止める。表２（c）は、ケースID23の *var_kekkon* の値を１から２に修正した後、*var_kekkon* と *var_kodomoninzu* のクロス表を出力したものである。一般的な論理チェックはこのようなプロセスで行われる。

そして、単純集計チェックと論理チェックからなるデータクリーニング作業は、基本的に次の手順で行われる（盛山 2004：153）。

ａ）チェックすべき項目を決定する。

ｂ）統計ソフトを用いて、チェックに引っかかるケースのＩＤ番号を抽出する。

ｃ）抽出されたケースの回収調査票を抜き出す。

ｄ）回収調査票を点検して、データをどのように修正すべきかを決定する。

ｅ）文書ソフトを用いてデータファイルを修正する。

ｆ）抜き出された回収調査票を元に戻す。

この手順を、ａ）チェック項目の設定、ｂ）異常の検出、ｃ）調査票の抜き出し、ｄ）修正方針の決定、ｅ）データの修正、ｆ）調査票の返却、とまとめよう。この手順で作業を進めていき、途中で新たにチェックすべき項目が見つかった場合は、それらをまとめて手順の最初から再び作業を行う。そして、新しいチェック項目が見つからなくなったらデータクリーニングを終了する。この一連のプロセスをフローチャートで表すと図１の通りである。

図１で明らかにされているように、従来のデータクリーニングでは、チェック項目の単位で作業を繰り返す。つまり、例えば年齢関連変数のチェックや配偶関係変数のチェックのように、変数または変数群を単位にして作業が進められる。このような作業方式を「変数中心のデータクリーニング」と呼ぶことにしよう。

一般的にデータクリーニング作業は１人で行うことは稀で、大人数による共同作業になることが多い。変数中心のデータクリーニングでは、作業の分担もおのずと変数（または変数群）単位で分けることになる。図１で言うと、「全てのチェック項目のチェックが終了したか」のループ作業を分担するわけである。そして、このような分担の仕方は、様々な困難を引き起こす。具体的にどのような困難が生じるのか、次に説明する。

2.2　変数中心のデータクリーニングの弊害

変数中心のデータクリーニングの弊害は、大きく分けて参照範囲の制限と修正範囲の制限の２つに整理できる。参照範囲の制限とは、参照する情報の範囲が制限されるため、自分が担当する変数の範囲では整合性が取れるが、他の人が担当する変数の範囲では整合性が取れないことを意味する。修正範囲の制限とは、修正すべき箇所があっても、それが自分の担当する変数ではない場合、簡単に修正できないことを意味する。具体例を挙げて説明しよう。

表３は、A さん、B さん、C さんがそれぞれ婚姻状況関連変数、所得関連変数、職歴関連変数を担当し、データクリーニングを行った架空の例である。クリーニングの対象となるのは、30歳女性で子供が１人いるが婚姻状況は非婚になっているケースである。

A さんは、婚姻状況関連変数をチェックし、回答者には子供が１人いるが婚姻状況は非婚になっていることを発見した。そして、子供がいるのに結婚したことがないというのはおかしいと考えて、婚姻状況変数の値を非婚から既婚に修正した。

B さんは、１人暮らしの回答者の本人年収と世帯年収を同じにする作業を行った。B さんも A さんと同様に、非婚でありながら子供が１人いるのはおかしいと考えた。しかし B さんは、回答者は非婚であり子供はいないはずだと考えて、世帯年収の値を本人年収と同じ200万円に修正した。

C さんは、A さんや B さんとは異なる回答者の人物像を描いた。C さんは、非婚でありながら子供が１人いるという回答は確かに蓋然性は低いが、非現実的ではないと考えた。C さんが回答者の職歴を確認すると、回答者は高校を中退していて初職に就いたのは16歳だった。このことから C さんは、回答者が高校時代に子供を産んだかもしれないと考えた。そして、婚姻状況変数と子供人数変数はそのままにして、配偶者の職業変数を無回答から非該当に修正した。

このように修正が施された３人のデータを一つに統合すると、表３に示されているように整合性の取れないデータができ上がってしまう。このような問題が起きる理由は、回答者の人物像を描くために参照した情報の範

表３　変数中心のデータクリーニングの例

作業分担者		A さん	B さん	C さん	
担当変数 回答者の人物像		婚姻状況関連 既婚の子あり	所得関連 非婚の子なし	職歴関連 非婚の子あり	
	元値	修正値	修正値	修正値	統合後のデータ
性別	女性				女性
年齢	30歳				30歳
婚姻状況	非婚	既婚			既婚
子供人数	１人		（０人）		１人
配偶者年齢	無回答		（非該当）	（非該当）	無回答
本人年収	200万				200万
世帯年収	無回答		200万		200万
学歴	高校中退				高校中退
初職就業年齢	16歳				16歳
配偶者職業	無回答			非該当	非該当

注１）括弧は修正希望

囲が、分担者間で異なるからである。C さんは、A さんが参照していない学歴、初職就業年齢、配偶者職業の情報を参照して、非婚でありながら子供を持つ母親という人物像を描いた。このように分担者間で異なる情報を参照することで起きるずれの問題が、参照範囲の制限の問題である。

参照範囲の制限の問題は、参照する情報の範囲を十分に広めれば対処できると思われるかもしれない。例えば、A さんも最初から学歴や職歴変数を参照していれば、C さんのような人物像を描いたかもしれないということである。しかし、人物像を描くのに役立つ情報を事前に選別することは不可能なため、それは現実的な方法ではない。一見チェック項目とは何の関係もない変数の回答からヒントを得ることが、実際のクリーニング作業では多々あるからである。結局、参照範囲の制限の問題を克服するためには、全ての変数を参照しなければならない。そうすると、全ての分担者が全ての変数に対してチェックを行うことになるので重複作業になってしまう。

仮に、労力を惜しまず全ての変数を参照してクリーニング作業を行ったとする。すると今度は、自分が担当していない変数の修正をどのように行えばいいかという困難に直面する。例えば、B さんは A さんと同じ情報を参照していたが、重きをおく情報が異なったため、A さんとは異なる人物像を描いた。そして、子供人数変数と配偶者年齢変数の修正を希望した。これらの変数は A さんが担当する変数であり、B さんが勝手に修正することはできない。理想としては、すぐに話し合いをして一つの修正方針を共有した後、各自の作業にとりかかるのが望ましい。しかし、実際の作業ではそれができない場合も多い。そのため、B さんは子供人数変数と配偶者年齢変数の修正を後回しにして、それらが修正されることを前提にして、自分が担当する変数の作業を進めることになる。そして、このような作業の進め方は大きなトラブルの引き金となる。例えば、後で話し合った結果、A さんの修正方針に従うことになった場合、B さんは（そして C さんも）それまで前提にしていたものが崩壊し、最初から作業をやり直さなければならないからである。これが修正範囲の制限の問題である。

整理すると、参照範囲の制限や修正範囲の制限の問題が起きる理由は、変数中心のデータクリーニングが回答の文脈的理解（contextual understanding of responses）と人物像の文脈的構成（contextual construction of the respondent's character）を妨げるからである。データクリーニングでは、回答者の一つ一つの回答を、それを観察する者にとって理解可能な形（すなわち、ストーリー）に再構成することが不可欠である。なぜならば、誤りの原因と正しい修正方法を突き止めるためには根拠が必要であり、回答者の回答に基づいて再構成したストーリーこそ、その根拠であるからである。しかし、変数中心のデータクリーニングは、チェックすべき項目別に作業を分担することで、必然的に変数の参照範囲と修正範囲を狭めて、回答の文脈的理解と人物像の文脈的構成を妨げてしまう。

2.3 粘土細工アプローチの提唱

粘土細工アプローチは、このような変数中心のデータクリーニングの弊害を克服するために提唱されたものである。以下では、粘土細工アプローチの要点と手順について説明する。

保田は、従来のデータクリーニングが因習的なものであり、統計学的なクリーニングの理論から逸脱していることを指摘する（保田 2018）。そして、統計学的なクリーニング研究で古典とされるフェレギ・ホルト原則（Fellegi and Holt 1976）に立ち返ることを提唱した[3)]。しかし、フェレギ・ホルト原則は国勢調査のような大規模のデータをコンピューターで自動クリーニングすることを想定したものであり、それをそのままの形で複雑な社会調査に適用することはできない。そこで保田は、フェレギ・ホルト原則に基づきながら、人の手で個別に判断してクリーニングを行う手続きを開発した。それが粘土細工アプローチである。

図 2 は、粘土細工アプローチの手続きをフローチャートで表したものである。各プロセスのタイトルは、理解のために従来のデータクリーニングのものを用いた。

図 2 から、従来のデータクリーニングと比較して粘土細工アプローチが持つ特徴を読み取ることができる。

一つ目の特徴は、エディティング（editing）と補定（imputation）[4)]のプロセスを区別し、両者を独立的なプロセスにしている点である。エディティングとは、要するに異常値を検出するプロセスのことで、図 2 で言うと、a）チェック項目の設定、全ての異常を検出したかの判定、b）異常の検出までがエディティングになる。補定とは、要するに異常値を修正するプロセスのことで、図 2 で言うと、全ての異常ケースの修正を終了したかの判定、c）調査票の抜き出し、d）修正方針の決定、e）データの修正、f）調査票の返却までが補定になる。エディティングと補定を独立的なプロセスにする

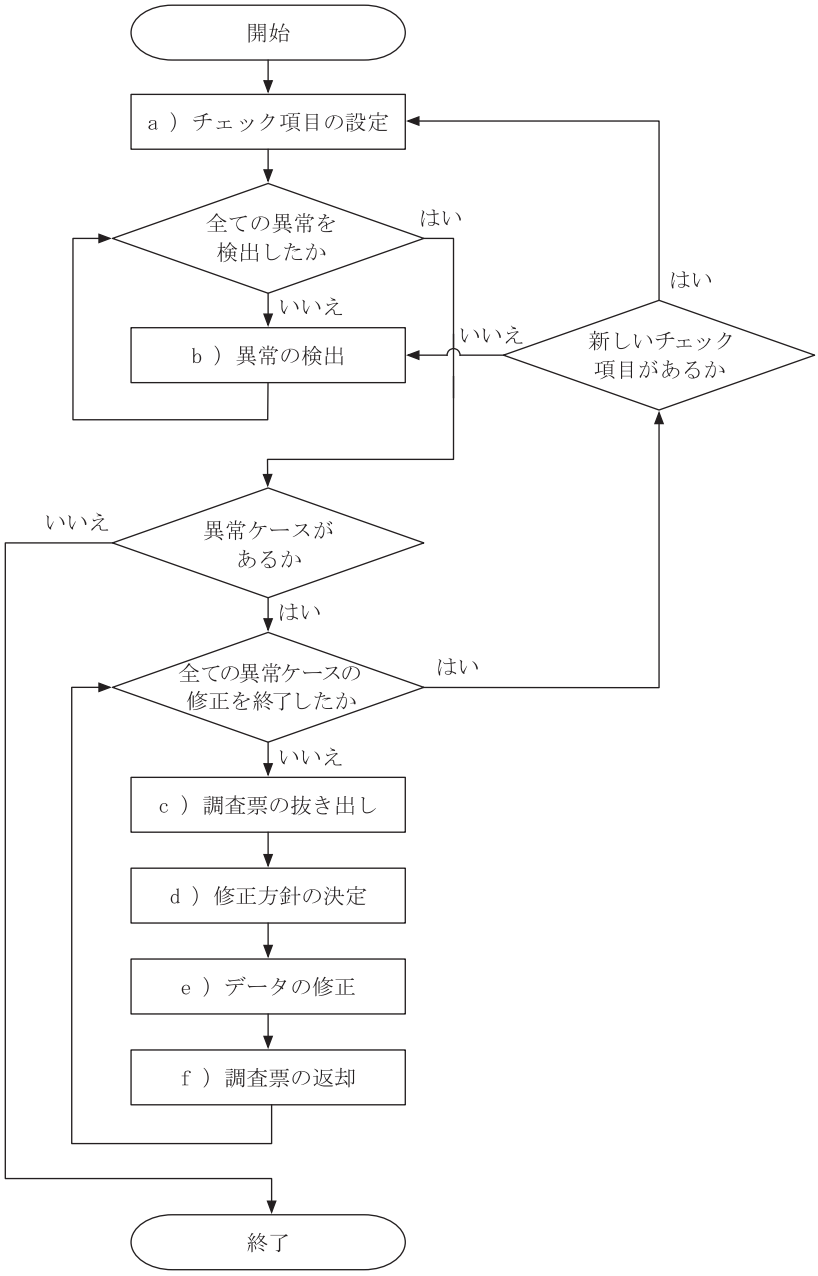


図２　粘土細工アプローチのフローチャート

ということは、エディティングが完全に終わってから補定に進むとし、プロセスの途中で両者を行き来することがないようにすることを意味する。図２を見ると、「全ての異常を検出したか」という判定条件を満たすまで、ｂ）異常の検出を繰り返していることが分かる。

２つ目の特徴は、修正作業がケース単位で行われる点である。まずエディティングで、設定した全てのチェック項目を適用して異常値を検出したら、次に補定では、検出された異常値をケース単位で一度に見渡しながら修正を施す。図２の「全ての異常ケースの修正を終了したか」のループがこの作業にあたる。このような作業の仕方を「ケース中心のデータクリーニング」と呼ぶことにしよう[5)]。ケース中心のデータクリーニングでは、作業の分担もおのずとケース単位で分けることになる。図２で言うと、「全ての異常ケースの修正を終了したか」のループ作業を分担するわけである。

このような特徴を持つ粘土細工アプローチを採用すれば、従来のデータクリーニングの弊害を克服することができる。先述したように、従来のデータクリーニングでは参照範囲の制限と修正範囲の制限の問題があった。これらの問題は、作業を変数単位で分担するがために生じる問題で、変数中心のデータクリーニングが回答の文脈的理解と人物像の文脈的構成を妨げるのが原因だった。しかし、ケース単位で作業を分担する粘土細工アプロー

チでは、回答の文脈的理解と人物像の文脈的構成を妨げることがないので、これらの問題は生じない。粘土細工アプローチではケース単位で作業を分担するので、１人の分担者が一つのケースの全回答を参照して一つの人物像を描く。したがって、自分が担当する変数の範囲では整合性が取れるが、他の人が担当する変数の範囲では整合性が取れないようなことは起きない。そして、粘土細工アプローチでは各分担者がケースを一つずつ担当して修正を施していくので、修正を保留した部分や自分が施した修正が他の分担者の作業に影響することもない。

このような利点の他にも、ケースを単位に作業を分担することで、従来にはなかった利点も生まれる。それはデータクリーニング作業の管理がしやすくなるという点である。データクリーニングの統括者は現在の作業進度を、クリーニングが終わったケースの数として明確に把握することができる。そして、その数字に基づいて作業スケジュールや作業の割り当てなどを決めることができる。作業の割り当てにおいても、統括者は任意の作業量（すなわち、ケースの数量）を随時、割り当てることができる。したがって、データクリーニングの進度に構わず、いつでも分担者の数を増やしたり減らしたりすることができる。このようにデータクリーニング作業の管理が柔軟にできる点は、従来のデータクリーニングでは期待できなかったものである。

それでは、粘土細工アプローチの具体的な手順について説明する。保田は粘土細工アプローチの手順を次の５つに区分した（保田 2018）。１）エディット規則[6]の作成、２）エディット規則の適用、３）ケースごとの修正、４）修正後のデータにエディット規則を再適用、５）完了の判断。表４は、これらの手順を従来のデータクリーニングの手順に対応させたものである。

１）エディット規則の作成では、異常値を検出するための論理式を書く。論理式はコンピューターが理解できる言語で書く。例えば、保田のツールでは、SPSS のシンタクスまたは Excel の関数式で論理式を書く。DCSS では、SPSS のシンタクスや Excel の関数式の文法が分からなくても、基本的な論理演算子を覚えれば通常の数式の形で論理式を書くことができる。その方法については第３節で説明する。

２）エディット規則の適用では、論理式をデータに適用して異常値を検出する。異常値の検出は、全ての論理式を全てのケースに適用して一括的に行う。この作業は自動化することができ、DCSS でも自動的に行われるようになっている。この作業までがエディティングの段階で、これが終わったら次に補定の段階に移る。

３）ケースごとの修正では、全てのエディット規則が適用された結果をケース単位で検討し、異常値を修正していく。異常値を修正する際は、当該ケースの全ての回答を参照して総合的な判断をする。この作業は大勢の人で分担して行うことができる。例えば、保田は10人程度までの分担者で作業を行うことを推奨している（保田 2018）。ただし、この数字はあくまでも一つの基準であって、調査データの特性に合わせて変えることもあり得る。後述するように、筆者は社会調査実習で13～14人の分担者で作業を行ったが、特に問題になることはなかった。作業量の分担は、まず数十または数百ケースの単位で分担して作業を行い、分担者の進度に合わせて割り当てを調整すると良い。

４）エディット規則を再適用では、修正後のデータを用いて再度エディティングと補定を行う。そうすることで、修正の漏れ、修正のミスを発見して対応することができ、新しいエディット規則の適用も可能になる。

５）完了の判断では、ケースごとの修正とエディット規則を再適用を繰り返して最終的に修正が必要な異常ケースがなくなったら、データクリーニングが完了したと判断する。修正が必要な異常ケースがないということ

表４　粘土細工アプローチの手順

作業段階の区分	粘土細工アプローチの手順	従来のデータクリーニングとの対応
エディティング	１）エディット規則の作成 ２）エディット規則の適用	ａ）チェック項目の設定 ｂ）異常の検出
補定	３）ケースごとの修正	ｃ）調査票の抜き出し ｄ）修正方針の決定 ｅ）データの修正 ｆ）調査票の返却
	４）エディット規則を再適用 ５）完了の判断	

図3　DCSS のデフォルト画面

の意味は、これ以上異常値が検出されないか、または許容できる異常値しか検出されないことを意味する。

3　データクリーニングシステムの開発：DCSS

3.1　DCSS の特徴

この節では、DCSS の概要と特徴、データの読み込みと書き出しの方法、基本的な操作方法について説明する。

DCSS は Java 8 で開発されたアプリケーションである。したがって、Java 8 のシステム要件を満たすコンピューターであれば、例えば Windows や macOS でもインストールして使用することができる[7)]。

DCSS は GUI（graphical user interface）を採用しているので、ユーザーは画面上に表示されるメニューやボタンを視覚的に捉えて直感的に操作することができる。粘土細工アプローチも直感的な操作で実現できる。例えば、DCSS には図3に示されているように、Report、Edit、Rules、Reference、Layout の5つのタブ（tab）があり、各タブにはそれぞれの目的と機能がある。そして、ユーザーは作業内容に応じてタブを移動しながら操作を行う。

各タブの目的と主な機能は次の通りである。

Report タブは、データクリーニング作業の進捗状況を把握するためのものである。Report タブでは、ケース ID、各ケースの修正を担当するエディターの名前、各ケースの修正記録などを書いたメモ、ケースごとの作業の進捗状況を確認することができる。

Edit タブは、ケースごとの修正を行うためのものである。Edit タブでは、変数名、変数ラベル、生値（raw value）をケースごとに表示させて、それらの情報を参照しながら編集値（edit value）を入力することができる[8)]。また、各ケースの修正記録などをメモで書き残すことができ、参考資料を確認することもできる。

Rules タブは、エディット規則を入力、管理するためのものである。Rules タブでは、変数名と変数ラベル、値と値ラベルを表示させて、それを見ながらエディット規則を入力、編集、削除することができる。また、各変数の値と値ラベルを入力、編集、削除することもできる。

Reference タブは、データクリーニングの作業時に参照する参考資料を入力、参照、管理するためのものである。Reference タブでは、XML ファイルで作成した参考資料を読み込ませて、それを参照、管理することができる。

Layout タブは、Edit タブで表示される変数の配置を決めるためのものである。Layout タブでは、変数名と変数ラベルを見ながら Edit タブのレイアウトを決めることができる。

3.2　データの読み込みと書き出し

DCSS では、クリーニングを行う生データ（raw data）の読み込みと、修正後のデータの書き出しを CSV 形式のファイルで行う。CSV 形式は、データの要素がカンマ（,）で区切られたデータ形式の一つであり、Windows のメモ帳などのテキストエディタ、または Excel などの表計算ソフトで作成、編集することができる。

生データの行列を CSV ファイルで作成する際に、先頭行は「変数名」を、先頭列は「ケース ID」を表すようにする。例えば、図4のように CSV ファイルを作成する。

生データの CSV ファイルを用意したら、DCSS でプ

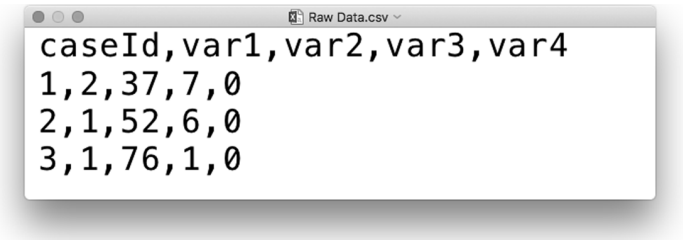


図 4　CSV 形式の生データの例

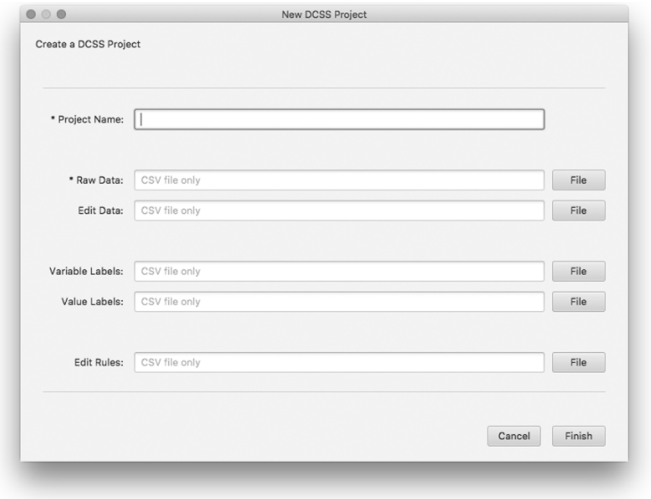


図 5　新しいプロジェクトの作成ウィンドウ

ロジェクトを作成する。DCSS では、データクリーニング作業に関する全ての情報を一つのファイルにまとめて管理する。そのファイルのことを「プロジェクト」と呼ぶ。プロジェクトは XML 形式のファイルである。以下ではプロジェクトの作り方について説明する。

まず DCSS を起動し、メニューの［File］→［New…］をクリックするか、または Windows では Ctrl+N キーを、macOS では Command+N キーを押す。すると図 5 のような新しいプロジェクトの作成ウィンドウが表示される。

新しいプロジェクトの作成ウィンドウの最上部には、「* Project Name : 」欄がある。この欄に新しいプロジェクトの名前を入力する。次に「* Raw Data : 」欄に、CSV ファイルで作成した生データのパス（path）を入力する。パスとは、ファイルが保存された場所のことである。パスは直接入力するか、欄の右側にある［File］ボタンを押してファイルを選択して入力する。新しいプロジェクトの名前と読み込ませたい生データのパスを入力したら、右下隅の［Finish］ボタンを押す。すると、新しいプロジェクトを保存する場所を指定するウィンドウが表示される。適当な場所を指定して［保存］ボタンを押すと、新しいプロジェクトが作成、保存される。プロジェクトが正しく作成されたら、図 6 のように Report タブの Case ID 列にケース ID の情報が表示される。

以上の方法で、DCSS のプロジェクトを作成することができる。ただし、このようにして作成したプロジェクトには、変数ラベルや値ラベルの情報は入っていない。変数ラベルや値ラベルの情報がなくてもプロジェクトを作成することはできるが、クリーニング作業を円滑に行うためには、もちろん変数ラベルや値ラベルの情報はあった方が良い。

変数ラベルや値ラベルのデータは、プロジェクトを作成する際に生データと一緒に読み込ませることができる。変数ラベルや値ラベルのデータも、生データの場合と同じく CSV ファイルで用意する。その際に、変数ラベルデータの先頭列は「変数名」、2 番目の列は「変数ラベル」を表すようにする。図 7 は Excel で変数ラベルデータを作成した例である。

値ラベルデータの場合は、先頭列は「変数名」、2 番目の列は「値」、3 番目の列は「値ラベル」を表すようにする。図 8 は Excel で値ラベルデータを作成した例である。

ケース ID 変数には、変数ラベルや値ラベルをつける必要はない。ケース ID 変数は、DCSS が生データを読み込む際に自動的に処理するからである。もし、ケース ID 変数の変数ラベルや値ラベルを図 7 や図 8 のような

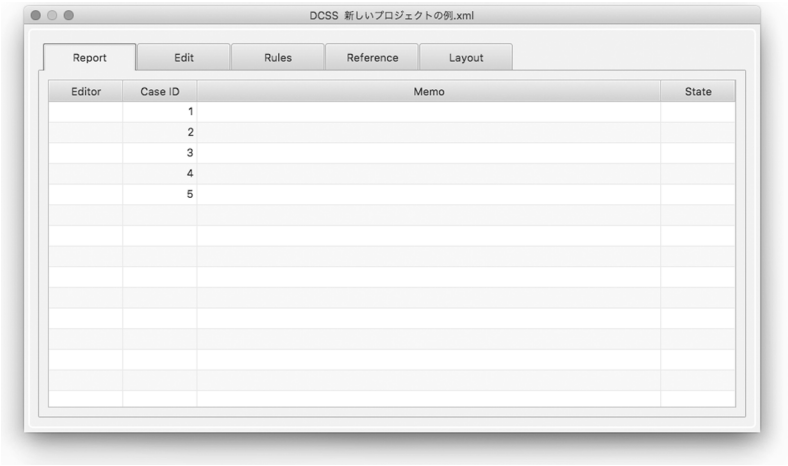


図 6　新しいプロジェクトを作成し、ケース ID が表示された画面

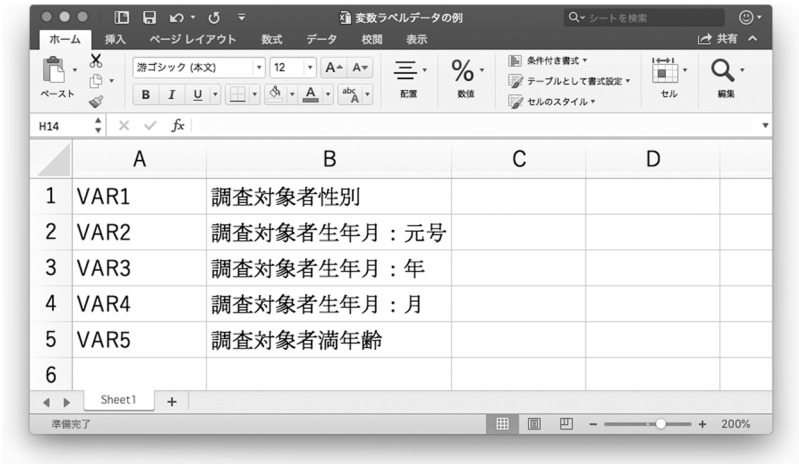


図 7　Excel で変数ラベルデータを作成した例

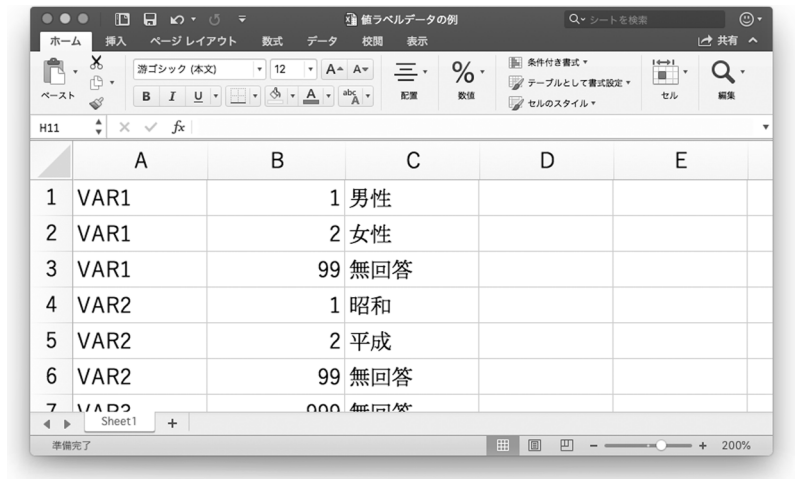


図 8　Excel で値ラベルデータを作成した例

データで作成して DCSS に読み込ませると、データの書き出しができなくなるので注意が必要である。

変数ラベルや値ラベルデータの CSV ファイルを用意し、それを DCSS に読み込ませる。変数ラベルや値ラベルデータの読み込みは、プロジェクトの作成時に行う。図 5 の新しいプロジェクトの作成ウィンドウの「Variable Labels : 」欄に変数ラベルデータが保存されたパスを、「Value Labels : 」欄に値ラベルデータが保存されたパスを入力する。そして、右下隅の［Finish］ボタンを押して新しいプロジェクトを作成すると、変数ラベルと値ラベルの情報が読み込まれたプロジェクトが作成できる。

DCSS では、データクリーニングの全過程を一つのプ

ロジェクトファイル内で行う。したがって、作業の途中で作業内容をこまめに保存したり、プロジェクトファイルをバックアップしておいたりすることが重要である。DCSS でプロジェクトを保存する方法は、メニューの［File］→［Save…］をクリックするか、または Windows では Ctrl+S キーを、macOS では Command+S キーを押すと、プロジェクトを保存することができる。そして、保存したプロジェクトを再度開きたい場合は、メニューの［File］→［Open…］をクリックするか、または Windows では　Ctrl+O キーを、macOS では　Command+O キーを押して、開きたいプロジェクトファイルを選択する。

プロジェクトは、DCSS でデータクリーニングを行うためのファイルであって、それがそのまま社会調査データになるわけではない。DCSS で生データを修正してプロジェクトファイルに保存しても、そのファイルは DCSS で開くことはできても、他の統計分析ソフトで開くことはできない。他のアプリケーションでも使用可能なファイルを作成するためには、DCSS のプロジェクトからデータを書き出さなければならない。

DCSS のデータの書き出し方は 3 つある。生データのみを書き出す、編集値のみを書き出す、生データに編集値が反映されたデータを書き出す、である。生データのみを書き出すは、プロジェクトを作成する際に読み込んだ生データをそのまま書き出す機能であり、メニューの［File］→［Export］→［Raw Data…］をクリックして実行する。編集値のみを書き出すは、DCSS で入力した編集値のみを書き出す機能であり、メニューの［File］→［Export］→［Edit Data…］をクリックして実行する。生データに編集値が反映されたデータを書き出すは、生データに DCSS で入力した編集値が上書きされたデータを書き出す機能である。つまり、編集値が入力されていない変数は生データのまま、編集値が入力された変数は編集値に置き換えられたデータが書き出される。この書き出しは、メニューの［File］→［Export］→[Edited Data…］をクリックして実行する。

書き出されたデータは CSV 形式のファイルなので、統計分析ソフトや表計算ソフトなどで開くことができる。

3.3　DCSS の基本操作

それでは、DCSS の基本的な操作方法について説明する。プロジェクトを作成し、データを書き出すまでの作業は、基本的に次の手順で進む。①エディット規則を入力する。②変数をレイアウトする。③エディターの名前を入力する。④ケースごとにデータを呼び出す。⑤編集値を入力する。⑥修正内容を記録する。⑦作業の進捗状態を記録する。以下では、この手順に沿って DCSS の操作方法を説明する。

①エディット規則を入力する

まず、エディット規則、すなわち異常値を検出するための論理式を入力する。論理式の入力は Rules タブで行う。Rules タブは、Variable List、Value List、Equation List、Keypad で構成される。図 9 は、その画面構成である。

Variable List は、変数名、変数ラベル、値のリストを表示する部分である。Value List は、変数の値と値ラベルを表示する部分である。Value List に変数の値と値ラベルを表示するには、Variable List で変数を選択すれば良い。また Value List では、値や値ラベルを入力、修正、削除するなどの編集を行うことができる。Equation List は、論理式のリストを表示する部分である。Equation

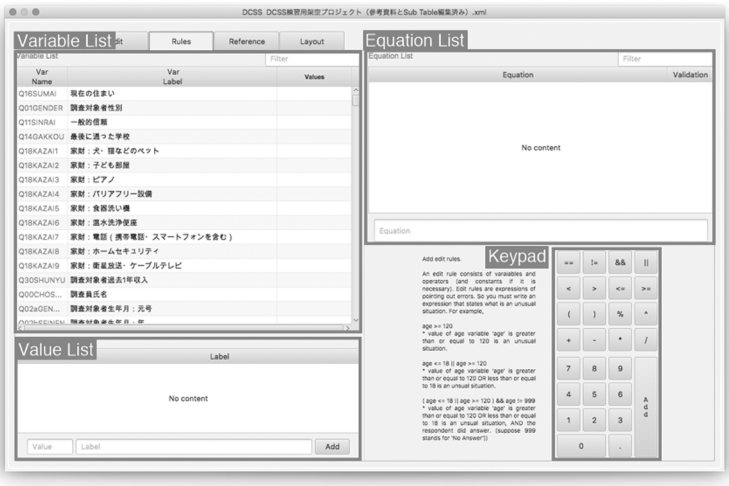


図 9　Rules タブの画面構成

List では、論理式を入力、修正、削除するなどの編集を行うことができる。Keypad は、論理式に数字や演算子を入力するためのものである。Keypad には論理式で使用可能な演算子が示されている。そのほとんどは一般的なものと同じだが、少し形が異なるものもある。等号は「==」、等号否定は「!=」、論理積は「&&」、論理和は「||」と表すので注意が必要である。

論理式は、Equation List の下部にある Equation 欄に入力する。論理式は Equation 欄に直接書き込むこともできるし、または Variable List を利用して変数名を、Keypad を利用して数字や演算子を入力することもできる。Variable List を利用して変数名を入力する場合、Variable List に表示された変数名を Equation 欄にドラッグアンドドロップする。数字や演算子は、Keypad のボタンを押して入力する。　論理式を Equation 欄に入力したら、エンターキー（または Keypad の「Add」ボタン）を押して Equation List に論理式を追加する。Equation List から論理式を削除したい場合は、Equation List で式を選択し、キーボードの「delete」キーを押す。Equation List の論理式を修正したい場合は、Equation List の論理式をダブルクリックすると編集可能状態になる。

論理式は、変数名と数字と演算子からなる。そして、その内容は「どのような値が異常値なのか」を表現したものである。式の内容が「どのような値が正常値なのか」を表現したものではない理由は、そのような内容で式を書く場合、変数の数が増えるにつれて書くべき式の数が指数関数的に増えるからである。

論理式の基本的な書き方は次の通りである。例えば、調査対象者の性別変数（*Q01GENDER*）に関する式を書くとする。変数 *Q01GENDER* の取り得る値は、1（男性）、2（女性）、99（無回答）とする。*Q01GENDER* の値が 1 より小さい場合、それは異常値である。これを式で表現すると（1）の通りである。

$$Q01GENDER < 1 \quad (1)$$

Q01GENDER の値が 2 より大きい、そして、値が99ではない場合、それは異常値である。これを式で表現すると（2）の通りである。

$$Q1GENDER > 2 \;\&\&\; Q01GENDER \;!= 99 \quad (2)$$

（1）と（2）を一つの式にまとめて、「値が 1 より小さい、または、値が 2 より大きい。そして、値が99ではない」という式を書くこともできる[9)]。

$$(Q1GENDER < 1 \| Q1GENDER > 2) \;\&\&\; Q01GENDER \;!= 99 \quad (3)$$

Q01GENDER の取り得る値は整数の離散値なので、剰余演算子（%）を利用して「値を 1 で割った剰余が 0 ではない」という式を付け加えることもできる。

$$(Q1GENDER < 1 \| Q1GENDER > 2) \;\&\&\; Q01GENDER \;!=99 \| Q1GENDER \;\%\; 1 \;!= 0 \quad (4)$$

図10は、これらの論理式を Equation List に追加した状態のものである。Equation 列には、入力した論理式が表示される。Validation 列には、Equation 列の論理式が正しく書かれているかどうかを簡易的にチェックした結果が表示される。チェック内容は変数名や演算子の書き間違いをチェックするもので、あくまでも簡易的なものである。変数名や演算子の書き間違いを検出しなかった場合は「validated」、検出した場合は「invalid」と表示される。

実際のデータクリーニングにおける論理式の類型とその例については、社会調査実習での活用例を紹介した第 4 節で説明する。

②変数をレイアウトする

次に、変数をレイアウトする。変数のレイアウトは Layout タブで行う。Layout タブは、Variable List、Main

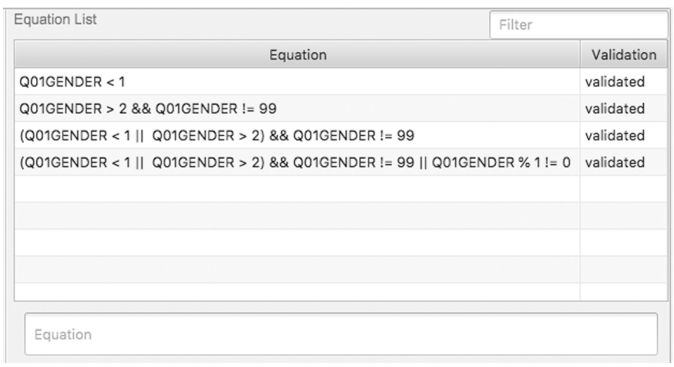


図10　Equation List に論理式を入力した例

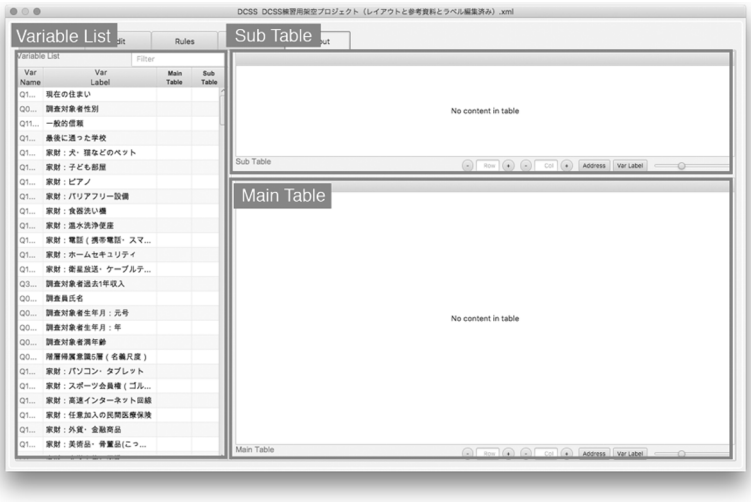


図11　Layout タブの画面構成

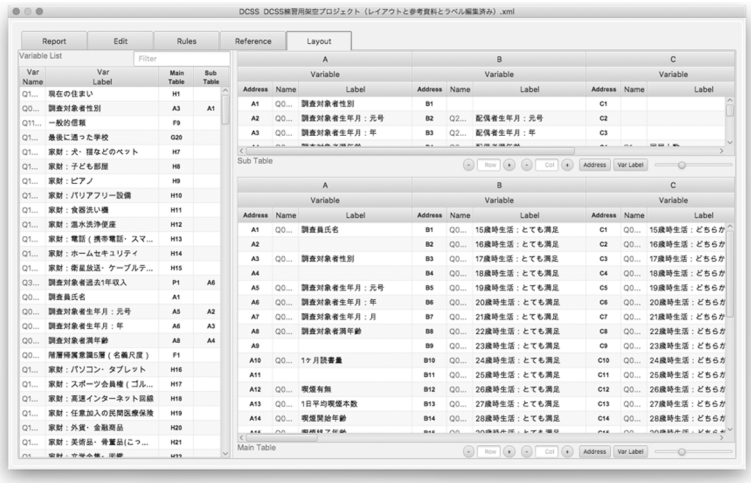


図12　変数のレイアウトの例

Table、Sub Table で構成される。図10は、その画面構成である。

Variable List は、変数名、変数ラベル、変数のアドレスを表示する部分である。変数のアドレスとは、Main Table や Sub Table における変数の位置のことである。DCSS では、生値や編集値をケースごとに表示させるためのテーブルが２つあり、Main Table と Sub Table である。簡単に説明すると、Main Table は編集値を入力するためのもので、Sub Table は生値や編集値を参照するためのものである。これらのテーブルは Edit タブで表示されるものなので、詳しい説明は Edit タブの時に説明する。Layout タブの Main Table と Sub Table は、それぞれ Main Table と Sub Table のレイアウトを調整する部分である。ここで Main Table と Sub Table の行と列の数を設定し、変数の配置を決める。

変数の配置を行うためには、まず行と列を追加してテーブルを生成しなければならない。行と列の追加や削除は、Main Table と Sub Table の下部にある Row 欄や Col 欄、または「+」や「−」ボタンで行う。行と列を生成する際に、先に生成するのは列である。列が生成されていない状態で行を生成しようとするとエラーメッセージが表示される。列の生成は、生成したい列の数を Col 欄に入力してエンターキーを押すか、Col 欄の右側にある+ボタンを生成したい列の数と同じ回数押して生成する。列を生成した後に行を生成する。行の生成も列の生成と同じ要領で、生成したい列の数を Row 欄に入力するか、Row 欄の右側にある+ボタンを押して生成する。

列は英語アルファベット順に A、B、C…と増えていき、行は数字順に１、２、３…と増えていく。これらを組み合わせるとアドレスになる。例えば、A１は１番目の列の１番目の行のアドレスである。

行と列を追加してテーブルを生成したら、変数を配置する。変数の配置は、Variable List の変数名を Main Table や Sub Table にドラッグアンドドロップして行う。ド

ラッグするのは「変数名」であって、変数ラベルではないことに注意する。変数ラベルをドラッグアンドドロップしても変数は配置されない。ドロップする場所は、配置したいアドレスの Name 列である。Address 列や Label 列にドロップしても変数は配置されない。

一旦配置した変数を、違うアドレスへ移動させたい場合は、Name 列の変数名をドラッグアンドドロップすれば良い。配置した変数を削除したい場合は、Name 列の変数名を選択し、キーボードの「delete」キーを押す。

③エディターの名前を入力する

エディット規則を入力し、変数をレイアウトした後は、作業の分担を決める。そして、分担者の名前を DCSS プロジェクトに記入する。DCSS では、各ケースの修正を担当する人のことをエディターと呼ぶ。エディターの名前の記入は、Report タブで行う。図 6 に示されているように、Report タブの最左列は Editor 列であり、2 番目の列は Case ID 列である。Case ID 列からケース ID を確認し、そのケースを担当するエディターの名前を Editor 列に入力する。入力は、まず入力を行うセルを選択し、ダブルクリックまたはエンターキーを押す。そしてセルが編集可能状態になったら、エディターの名前を入力する。名前を入力したら、エンターキーを押す。エンターキーを押さないと入力した内容が反映されないので注意する。

ここまでの作業は統括者の仕事である。つまり、エディット規則を入力し、変数をレイアウトし、エディターの名前を入力するまでの作業は、1 人（または少人数）が一つのプロジェクトファイル上で行う。分担者には、エディット規則が入力され、変数がレイアウトされ、エディターの名前が入力されたプロジェクトファイルを配布する。

④ケースごとにデータを呼び出す

プロジェクトファイルを受け取った分担者たちは、それぞれのコンピューターで DCSS を起動し、プロジェクトファイルを開いてケースごとのクリーニング作業を開始する。ケースごとのクリーニング作業は、Edit タブで行う。Edit タブの基本画面は、Main Table のみで構成される。図13は、その画面構成である。

Edit タブの Main Table には、Layout タブでレイアウトした通りに変数がレイアウトされる。作業者は、必要に応じて Layout タブで随時 Main Table のレイアウトを変更することができる。

Main Table（そして後に説明する Sub Table）の列は、図14のように階層構造になっている。A、B、C…と増えていく列の下位階層には「Variable」と「Value」の 2 つの列がある。Variable 列は変数に関する情報を表示する列であり、Value 列は値に関する情報を表示する列である。Variable 列の下位階層には「Address」と「Name」と「Label」の 3 つの列がある。Address 列はセルのアドレスを表示する列である。Name 列は変数名を表示す

図13　Edit タブの画面構成

図14　列の階層構造

る列である。Label 列は変数ラベルを表示する列である。Value 列の下位階層には「Raw」と「Edit」の２つの列がある。Raw 列は生値を表示する列である。Edit 列は編集値を入力、表示する列である。

⑤編集値を入力する

それでは、ケースごとにクリーニング作業を開始する。まず、ケースを選択し、当該ケースのデータ（生値と編集値など）を呼び出す。ケースの選択は、図15に示されているように、Edit タブの左下隅にあるケース ID を選択するチョイスボックス（choice box）で行う。このチョイスボックスからケース ID を選択すると、当該ケースのデータが呼び出され、Main Table に表示される。

ケースのデータを呼び出す際に、エディット規則の適用が行われる。そして、異常値の検出結果が Name 列に表示される。異常値を検出した場合、図16の A３のように変数名のセルが赤い枠線で囲われ、変数名が赤い太字で表示される。

Name 列の変数名を選択すると、その変数名が使用された論理式が図17のように画面下部に表示される。ここで異常値を検出した式を確認することができる。表示される論理式は黒字のものと赤字のものがあり、黒字は異常値を検出しなかったことを、赤字は異常値を検出したことを示す。つまり、赤字の式が異常値を検出した式である。

Edit タブは、回答の文脈的理解と人物像の文脈的構成に必要な全ての情報を参照できるようにデザインされている。Edit タブの基本画面は、Main Table のみで構成されているが、エディターは随時、ペイン（pane）を表示させて必要な情報を参照することができる。例えば、Sub Table がその一つである。Edit タブの右下に「Sub Table」ボタンがある。Sub Table ボタンを押すと、図18のように Edit タブの上部に Sub Table ペインが現れる。Sub Table では、Main Table とは別にエディターが参照したい変数の生値と編集値を自由にレイアウトして参照することができる。例えば、性別、年齢、学

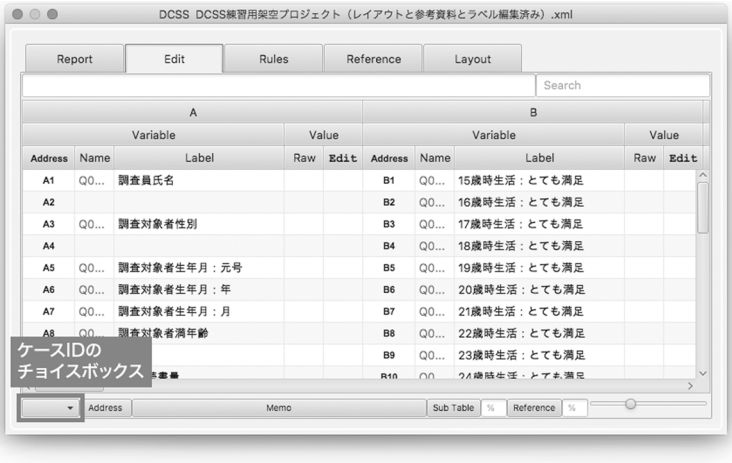


図15　ケース ID のチョイスボックス

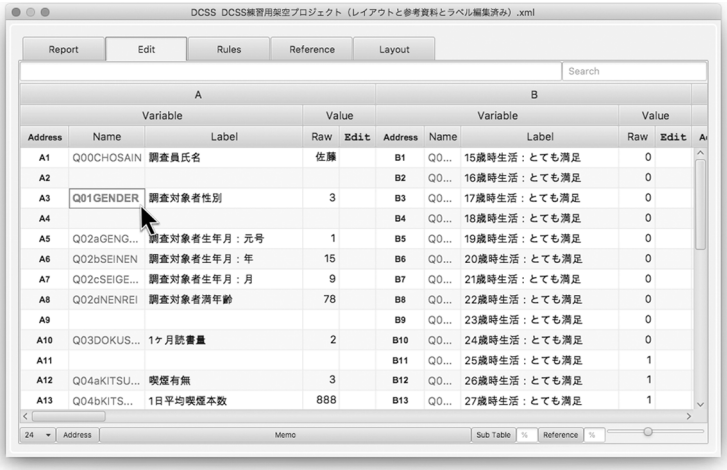


図16　Name 列に異常値の検出結果が表示される

図17　変数名を選択すると画面下部に論理式が表示される

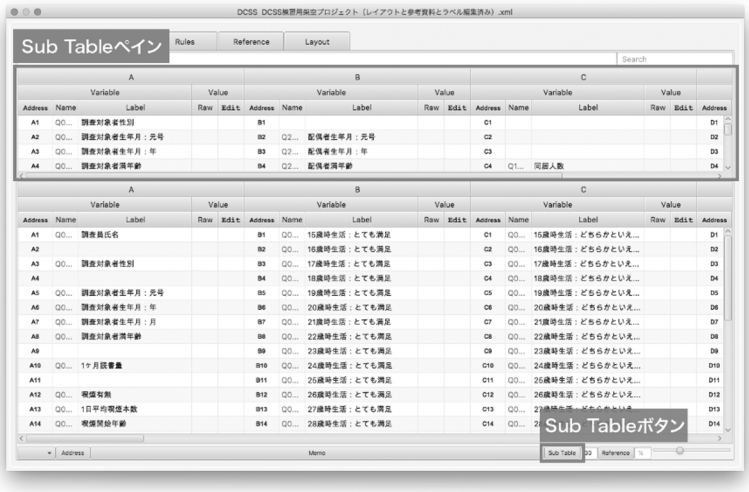


図18　Sub Table ペイン

歴、婚姻状況、現職など、回答者の人物像を構成する際に参考になる情報をまとめて Sub Table に表示させることができる。または、職歴の就業年齢と離職年齢の情報だけをまとめて年齢のずれを確認することもできる。Sub Table は、変数の数が多くて Main Table を一度に見渡せない場合に特に有用である。

このようにして、異常値の有無、異常値、異常値を検出した式、そしてその他の全ての変数の情報を参照しながら、エディターは回答の文脈的理解と人物像の文脈的構成の過程を経て、総合的な判断で編集値を決める。編集値が決まったら、それを Main Table の Edit 列に入力する。入力を行うセルを選択し、ダブルクリックまたはエンターキーを押して、セルを編集可能状態にする。そして編集値を入力し、エンターキーを押す。エンターキーを押さないと入力した内容は反映されないので注意する。入力された編集値は、赤字で表示される。

⑥修正内容を記録する

ケースの全ての異常値のチェックが終わったら、ケースの修正内容を記録する。修正内容の記録は Memo ペインで行う。Memo ペインは、Edit タブの下部にある「Memo」ボタンを押すと現れる。Memo ペインに記録した内容は、Edit タブまたは Report タブで読むことができる。

⑦作業の進捗状態を記録する

ケースの修正内容の記録が終わったら、作業の進捗状態を記録する。作業の進捗状態の記録は Report タブで行う。Report タブの最右列は State 列であり、ここに作業の進捗状態を記録する。進捗状態を記録したいセルをダブルクリック、またはセルを選択してエンターキーを押すと、セルの右側に逆三角形の選択ボタンが現れる。選択ボタンを押すと、進捗状態を Unreviewed、Shelved、Reviewed の３つから選ぶメニューが現れる。Unreviewed は未検討、Shelved は保留、Reviewed は検討済みを意味する。適切なものを選んで作業の進捗状態を記録する。

以上の手順と操作で DCSS によるデータクリーニングが行われる。

4　社会調査実習での活用

4.1　調査データの整理の仕方

DCSS は粘土細工アプローチを具現するシステムとして開発されたアプリケーションであるが、調査データの整理全般において活用することもできる。以下ではその方法を、社会調査実習での DCSS の活用例に基づいて説明する。

一般的に調査データの整理は、点検、コーディング、データ入力、データクリーニングの 4 段階の作業で行われる（盛山 2004 ; 大谷ほか 2013）。各作業の要点を、

A				
Variable			Value	
Address	Name	Label	Raw	Edit
A1	Q00CHOSAIN	調査員氏名	佐藤	
A2				
A3	Q01GENDER	調査対象者性別	3	2
A4				
A5	Q02aGENG...	調査対象者生年月：元号	1	

図19　セルを編集可能状態にしてから編集値を入力する

従来のやり方に基づいて整理すると次の通りである。点検は、回答者の回答を確定し、有効票と無効票を判別する作業である。点検作業は、調査票上で行われる。コーディングは、調査票の情報を電子ファイルに入力可能な数字や記号に置き換える作業である。コーディングは、コードの作成を調査前にするか調査後にするかによってプリコーディングとアフターコーディングに区別される。アフターコーディングの場合、自由回答の書き出しファイルを別途作成し、それを検討してコード表を作成する。プリコーディングもアフターコーディングも、作業は調査票上で行われる。データ入力は、調査票に記入されたコードをテキストデータまたは表形式データの形式で電子ファイルに入力する作業である。データ入力作業は、コンピューター上で行われる。データクリーニングは、データの誤りを見つけて修正を施す作業であり、作業はコンピューター上で行われる。

これが調査データの整理の基本だが、しかし近年では必ずしもこのようなやり方で調査データの整理が行われるわけではない。例えば、調査会社に調査を依頼してデータを納品してもらう場合、点検とデータ入力（自由回答の書き出しファイルの作成を含む）までの作業は調査会社側が担当してくれるので、調査を依頼する側はコーディングとデータクリーニングを担当することになる。つまり、表５のように点検、データ入力、コーディング、データクリーニングの順で調査データの整理が行われることになる。

そして、このような手順で作業が行われると、データ入力とコーディング作業の内容は自然と次のようになる。つまり、データ入力時には文字列データを含めて調査票上の全ての情報を原文のまま電子ファイルに入力することに専念し、コーディング時にはそのデータをもとにコンピューター上でコードの割り振りを行う。

先にデータ入力を行い、その後コンピューター上でコーディングを行うというのは、粘土細工アプローチの考え方にも通ずる。従来のように自由回答の書き出しファイルを用いてコーディングを行う場合、自由回答の書き出しファイルを見てコード表を作成すること自体は問題ではないが、自由回答の書き出しファイルだけを見

表５　近年における調査データの整理の手順と概要

	点検	データ入力	コーディング	データクリーニング
意味	回答者の回答を確定し、有効票と無効票を判別する作業	調査票に記入されたコードや文字列を、テキストデータまたは表形式データの形式で電子ファイルに入力する作業	調査票の情報を電子ファイルに入力可能な数字や記号に置き換える作業	回答者の誤答、調査員の誤記入や記入漏れ、コーディングミスやコーディング漏れ、データ入力における誤りを修正する作業
作業内容	誤記入、回答漏れ、つじつまが合わない回答などをチェックして、回答を修正・確定・破棄する	テキストデータ：カラム設計をし、それに従って任意のテキストエディタを利用してデータを入力 表形式データ：任意の表計算ソフトを利用してデータを入力	プリコーディング：調査前に用意されたコード表に従って回答にコードを割り振る アフターコーディング：調査後に回答を見てコードを作成し、回答にコードを割り振る	単純集計チェック：存在しないコードが値になっていないか 論理チェック：蓋然性の低い回答の組み合わせがないか
作業場所	調査票上	コンピューター上	コンピューター上	コンピューター上

てコードを割り振ることは問題になる。それは変数中心のデータクリーニングの場合と同じく、回答の文脈的理解と人物像の文脈的構成を考慮していないからである。その結果、自由回答の書き出しファイルだけを見て割り振ったコードをデータクリーニング時に見直さなければならないことが度々起きる。このような問題を回避するためにも、粘土細工アプローチの考え方に沿って、コーディングをデータクリーニングと同時に行うのが望ましい。

以上の理由から筆者が担当する社会調査実習では、点検、データ入力、コーディング、データクリーニングの順で調査データの整理を行い、コーディングとデータクリーニングは同時に行うこととした。そしてDCSSは、作業場所が調査票上である点検を除いて、データ入力、コーディング、データクリーニングにおいて活用された。

調査データの整理作業でDCSSが活用されたのは、2017年度と2018年度に専修大学で開講された社会調査実習においてである。社会調査実習で実施した調査および調査データの整理作業の概要は､表6の通りである｡2017年度の実習では、有効回収数152票の調査データの整理を、13人の作業分担者が休憩時間を含めて1日あたり約7時間の作業を3日間行った（羅 2018)。2018年度の実習では、有効回収数112票の調査データの整理を、14人の作業分担者が休憩時間を含めて1日あたり約7時間の作業を3日間行った。

4.2　DCSSによるデータ入力

DCSSによるデータ入力のポイントは、DCSSのレイアウト機能の活用である。DCSSのレイアウト機能は、作業者の疲労を軽減しデータ入力のミスを減らすことに役立つ。

データ入力における課題は、いかにミスを減らすかという問題である。ミスの種類には、調査票の誤読、キーのミスタッチ、カラムのずれなどが指摘されている（盛山 2004)｡このようなミスは基本的に人為的過誤(human error）であり、完全になくすことはできない。この種のミスは作業者の注意力に大きく左右されるものなので、作業者の疲労がたまらないようにするなどの注意力対策が重要である。DCSSのレイアウト機能は、この注意力対策として有効である。

DCSSのレイアウト機能とは、変数の配置をユーザーが自由に変えられる機能である。従来のデータ入力ではレイアウトの概念がなく、変数とケースからなる行列に直接、値を入力していた。そのため、調査票上の変数の配置とデータ行列上の変数の配置との間には関連性がなく、入力ミスを減らすためにはかなり神経を使わなければならなかった。しかしDCSSのレイアウト機能を利用すれば、調査票の配置と関連性を持たせて変数を配置し、データ入力画面をより直感的に構成することができる。例えば、Main TableのA列にはフェイスシートの変数を配置し、B列には現職関連変数を配置する、または子供関連変数を第1子と第2子とで列を変えて配置するといったことが可能である。このようにすれば、データを打ち込む作業がより直感的になり、また調査票の回答と入力データとを照合する作業も楽になる。そして、疲労の軽減はミスの軽減につながる。

それではDCSSでデータ入力作業を行う方法を説明する。簡単に説明すると、生値が空のプロジェクトを生成し、入力したいデータを編集値として入力し、編集値のみをCSVファイルで書き出す、という作業を行う。

まず、図20のように先頭行は「変数名」を、先頭列は

表6　社会調査実習の調査および調査データの整理作業の概要

	2017年度	2018年度
調査名	専修大学生のジェンダー意識に関する社会調査	2018年専修大学社会学科学部生のジェンダー意識に関する社会調査
母集団	2017年4月1日現在専修大学に在学する昼間部の学部学生1・2・3年次12,931名	2018年4月1日現在、専修大学人間科学部社会学科に在学する学部学生1・2・3年次424名
有効回収数	152票（回収率72.4%）	112票（回収率53.3%）
変数の総数	191個	111個
調査実施期間	2017年7月12日～9月13日	2018年7月11日～24日
データの整理期間	2017年9月14、19、20日（10：00～17：00）	2018年9月10、11、12日（10：00～17：00）
作業分担者の数	13人	14人

「ケースID」を表すようにし、変数の値は空白の行列をCSVファイルで作成する。

そして、図5のように新しいプロジェクトの作成ウィンドウを開き、「* Project Name :」欄にプロジェクトの名前を、「* Raw Data :」欄にデータ入力用CSVファイルのパスを入力し、新しいプロジェクトを作成する。次にLayoutタブに移動して、図12のようにMain Tableに行と列を追加してテーブルを生成し、変数をレイアウトする。次にEditタブに移動して、ケースIDのチョイスボックスからケースIDを選択し、Edit列にデータを入力していく。この作業を全ケースで繰り返す。全てのケースにデータを入力したら、メニューの［File］→［Export］→［Edit Data…］をクリックして、DCSSで入力した編集値のみを書き出し、データ入力作業を終了する。

2017年度の社会調査実習では、変数の総数191個のデータを、13人の作業者が1人あたり約12ケース担当し、1日半でデータ入力作業を終えた。2018年度の社会調査実習では、変数の総数111個のデータを、14人の作業者が1人あたり8ケース担当し、1日でデータ入力作業を終えた。作業の分担は個人単位で分けたが、作業は2人1組になって行った。調査票の誤読とキーのミスタッチを防ぐための措置である。一人は調査票を見ながら調査項目と回答を読み上げる役割を担当し、もう一人はDCSSの画面を見ながら調査項目と回答を復唱しキーボードで値を打ち込む役割を担当した。

ちなみに、入力ミスを減らすための工夫の一つとして、非該当と無回答のコードを全ての変数において統一した。一般的に非該当のコードには数字の8が使用され、無回答のコードには数字の9が使用される。そしてその桁数は、変数ごとに異なる。この慣例は、各変数の

図21　DCSSを利用したデータ入力の様子

占める桁数をきっちりと決めなければならなかったカラム設計時代の名残りと思われる。しかし、データの管理方法が変わった現代では、このような慣例は不必要であり、むしろ混乱を招きかねない。したがって、全ての変数において非該当のコードは8888、無回答のコードは9999に統一した。

4.3　DCSSによるコーディング

コンピューター上でコードの割り振りを行う際の課題は、参照にかかる手間をいかに減らすかという問題である。参照の手間には2種類ある。一つは参照したい情報にたどり着くまでの手間、もう一つは散在する情報をまとめて参照するための手間である。

参照したい情報にたどり着くまでの手間とは、例えば、回答とコードとの対応関係を説明したガイドブックを調べる際にかかる手間のことである。ガイドブックが印刷物の形で配布された場合、作業者は例えば紙の辞書を引くような手間をかけて回答とコードとの対応関係を確認しなければならない。

散在する情報をまとめて参照するための手間とは、

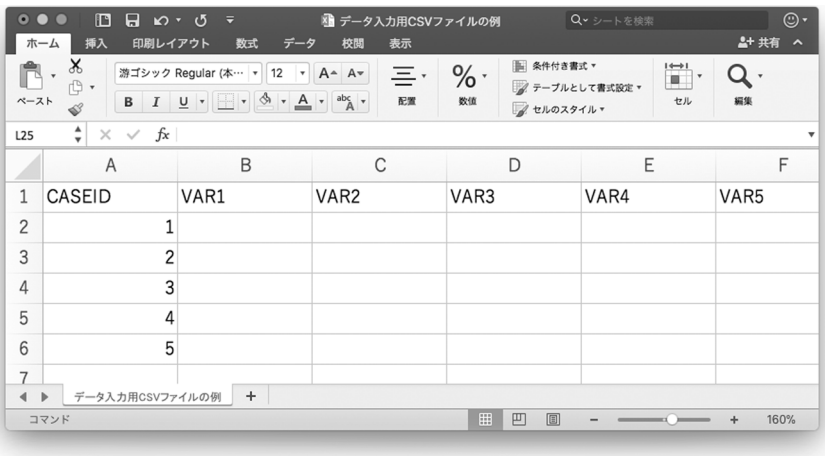


図20　データ入力用CSVファイルの作成例

コードを割り振る際に、回答の文脈的理解と人物像の文脈的構成のために諸変数の値を参照する際にかかる手間のことである。例えば、職業コーディングのために回答者の職歴の全体的な流れを把握したいとする。作業者は、調査票またはコンピューター画面上の生データの行列から、職歴の全体的な把握に必要な情報を取捨選択し、それらを一度に見渡せるように一箇所にまとめてメモする必要がある。まるで難しい英文の意味を理解するために、いくつもの単語の意味を調べてメモしていくような手間である。

これらの手間は、DCSS の Reference 機能と Sub Table 機能を利用して減らすことができる[10]。Sub Table 機能については、図18で説明した通りなので、ここでは説明を省略する。以下では、Reference 機能について説明する。

DCSS では、最大10点まで参考資料を保存することができる。そして、図22のようにデータの編集画面（Edit タブ画面）に Reference ペインを表示させて参考資料を閲覧、検索することができる。Reference ペインは、画面右下にある Reference ボタンを押すと表示される。作業者はデータ入力や編集の途中、別のアプリケーションを立ち上げることなく DCSS 内で参考資料を閲覧、検索することができる。

参考資料の読み込みと管理は Reference タブで行う。Reference タブは、Text Area、Reference Table、Page Navigation、Import Buttons で構成される。図23は、その画面構成である。

Text Area は、Reference Table に表示されるテーブルのセルの内容を確認する部分である。Reference Table は DCSS に読み込まれた参考資料を表示する部分である。参考資料は最大10点まで読み込むことができる。10点の参考資料は、画面下部の Page Navigation で一つずつ「めくる」ことができる。画面右側には、Import Buttons がある。Import ボタンは参考資料を読み込むボタンで、Clear ボタンは参考資料を削除するボタンである。

参考資料は XML 形式のファイルで用意する。参考資料の XML ファイルは Excel で簡単に作成できる。通常の Excel ファイルを作成して、それを XML 形式で保存すれば良い。Excel のメニューから［ファイル］→［名前を付けて保存］をクリックして「名前を付けて保存ウィンドウ」が開いたら、下の方にある「ファイルの種類（macOS 版の Excel では「ファイル形式」）」ボックスの一覧から、「XML スプレッドシート」と書かれた形式を選択して保存すると、Excel ファイルを XML 形式で保存することができる。

参考資料は Excel で作成するわけなので、行列の形になる。その際、行列の先頭行は「列の見出し」を表すようにし、列の総数は10を超えないようにする。そして、Excel のシート名が参考資料の名前として読み込まれるので、シート名に参考資料の名前を入力する。図24は、コード表を参考資料として作成した例である。

コード表の他にも、コーディング方針やメモ、または産業分類や職業分類など、様々な参考資料を作成して活用することができる。2017年度の社会調査実習では、回答者の学科と部活の団体名のコード表を作成し、Reference 機能を利用して学科変数と部活変数のコーディングを行った。2018年度の社会調査実習では、総務省の日本標準産業分類と日本標準職業分類を参考資料として作成し、Reference 機能を利用して回答者の15歳時の父親

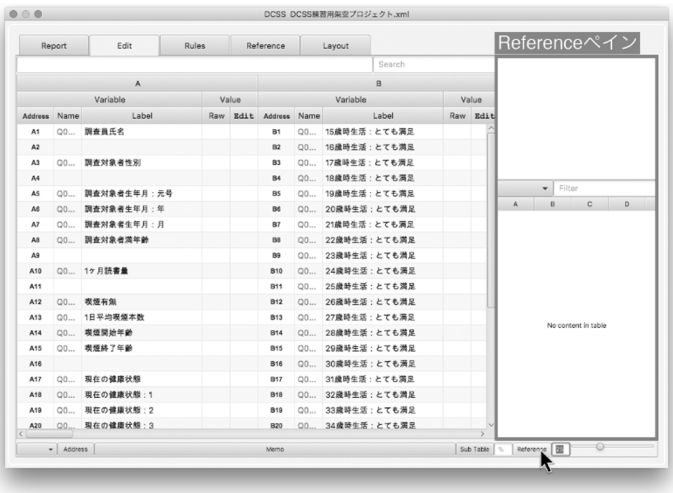


図22　Reference ペイン

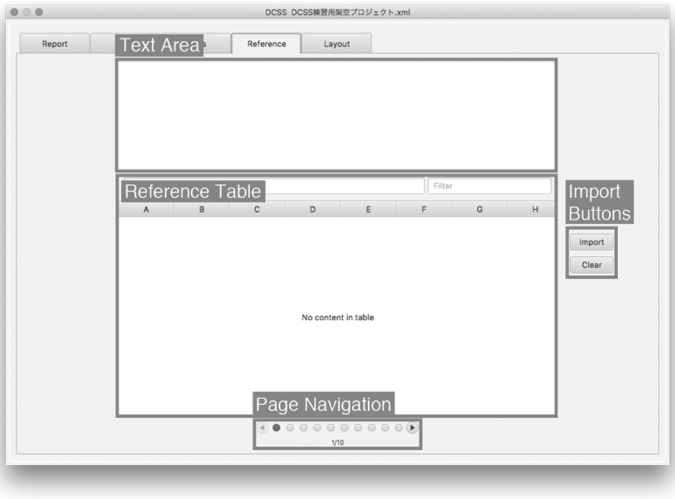


図23　Reference タブの画面構成

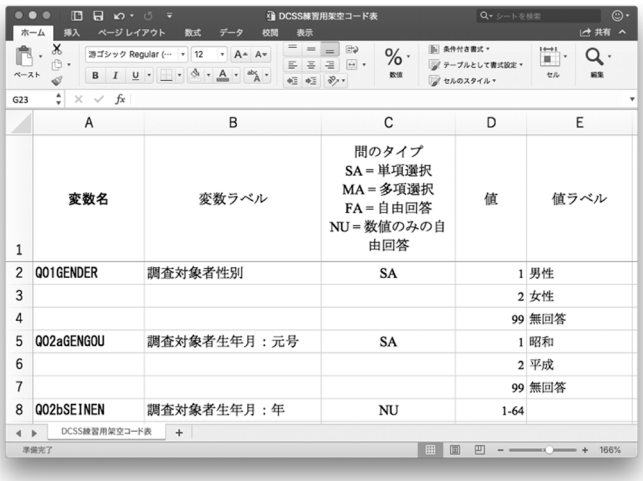
と母親の職業変数のコーディングを行った。

4.4　DCSS によるデータクリーニング

粘土細工アプローチでは、エディット規則の種類を range edit、filter edit、general edit の 3 つに分けて考える（保田 2018）。この分類は先行研究（Delgado-Quintero and Salazar-Gonzalez 2008）で提案されたものを保田が採用したもので、実践的な利便性を重視した分類法である。以下では、それぞれの分類と、その論理式を DCSS で書く方法について説明する。

range edit は、存在しないコードが値になっていないかをチェックするものであり、従来のデータクリーニングの単純集計チェックにあたる。range edit の書き方は、値の種類によって異なる。社会調査データの変数の取りうる値には 4 種類ある。一つ目は自由回答の文字列であり、これはデータクリーニングの対象外である。2 つ目は連続値の範囲で、長さや重さのようにいくらでも細かく測定できる有理数の値である。連続値の範囲の range edit は、例えば8888（非該当のコード）と9999（無回答のコード）を除いて、1 から10までの有理数の範囲に収まらない値は異常である、というような式を書く。3 つ目は離散値のリストで、連続しない整数の値である。離散値のリストの range edit は、例えば、1 でもない、5 でもない、10でもない、8888でもない、9999でもない値は異常である、というような式を書く。4 つ目は離散値の範囲で、年齢変数のように連続した整数の値である。離散値の範囲の range edit は、例えば、1 で割った剰余が 0 ではない値、または、8888と9999を除いて、1 から10までの範囲に収まらない値は異常である、というような式を書く。

filter edit は、非該当処理の不備をチェックするものであり、従来のデータクリーニングの論理チェックの中



	A	B	C	D	E
1	変数名	変数ラベル	問のタイプ SA＝単項選択 MA＝多項選択 FA＝自由回答 NU＝数値のみの自由回答	値	値ラベル
2	Q01GENDER	調査対象者性別	SA	1	男性
3				2	女性
4				99	無回答
5	Q02aGENGOU	調査対象者生年月：元号	SA	1	昭和
6				2	平成
7				99	無回答
8	Q02bSEINEN	調査対象者生年月：年	NU	1-64	

図24　参考資料の作成例

でフィルター質問とサブクエスチョンに関するものにあたる。非該当処理のチェックは、２つの方向でしなければならない点に注意する。フィルター質問で非該当になった調査対象者が、サブクエスチョンでは非該当になっていない場合と、フィルター質問で該当になった調査対象者が、サブクエスチョンでは非該当になっている場合の、両方のチェックが必要である。前者は、例えば、フィルター質問の値が２でもない、５でもない、６でもないのに、サブクエスチョンの値が8888ではないのは異常である、というような式を書く。後者は、例えば、フィルター質問の値が２、または５、または６なのに、サブクエスチョンの値が8888であるのは異常である、というような式を書く。

general edit は、filter edit 以外の論理的矛盾をチェックするものである。general edit は、蓋然性の低い回答の全ての組み合わせをチェックするものなので、変数の数と構成によってその数と類型は様々である。ここでは一般的なものとして、合計のチェックと大小関係のチェックを紹介する[11)]。合計のチェックは、例えば、通学時間の時間変数の値に60をかけて、通学時間の分変数の値に足したものが、通学時間の合計変数の値と同じでないのは異常である、というような式を書く。大小関係のチェックは、例えば、兄弟の順番変数の値が、兄弟人数変数の値より大きいのは異常である、というような式を書く。

表７は、以上の説明をまとめたものである。range edit、filter edit、general edit という呼び方は意味が伝わりづらいので、社会調査実習では検出内容をそのまま呼び方にして、存在しないコード、非該当処理の不備、論理的矛盾と呼んだ。

存在しないコード、非該当処理の不備を検出する式は、表７に示されたテンプレートを利用して作成した。実習で使用した調査票の構造はそれほど複雑ではなかったので、論理的矛盾の場合も表７のテンプレートで全ての論理式を書くことができた。

2017年度の実習では194のエディット規則を、2018年度の実習では138のエディット規則を作成してデータクリーニングを行った。

表７　DCSS のエディット規則の作成例

	式の類型	式のテンプレート	式の例
存在しないコード	連続値の範囲のチェック	(*変数名* < *最小値* || *変数名* > *最大値*) && *変数名* != *非該当コード* && *変数名* != *無回答コード*	(*var_weight* < 0 || *var_weight* > 200) && *var_weight* != 8888 && *var*_weight!= 9999
	離散値のリストのチェック	*変数名* != *値* && *変数名* != *値* && *変数名* != *値* && *変数名* != *値*	*var_syokugyo* != 11 && *var_syokugyo* != 22 && *var_syokugyo* != 103 && *var_syokugyo* != 325 && *var_syokugyo* != 8888 && *var_syokugyo* != 9999
	離散値の範囲のチェック	(*変数名* < *最小値* || *変数名* > *最大値*) && *変数名* != *非該当コード* && *変数名* != *無回答コード* || *変数名* % 1 = 0	(*var_nenrei* < 20 || *var_nenrei* > 100) && *var_nenrei* != 8888 && *var_nenrei* != 9999 || *var_nenrei* % 1 != 0
非該当処理の不備	フィルター質問で非該当になった調査対象者が、サブクエスチョンでは非該当になっていない	(*変数名 1* != *値* && *変数名 1* != *値* && *変数名 1* != *値*) && *変数名 2* != *非該当のコード*	(*var_kekkon* != 2 && *var_kekkon* != 5 && *var_kekkon* != 6) && *var_haigunenrei* != 8888
	フィルター質問で該当になった調査対象者が、サブクエスチョンでは非該当になっている	(*変数名 1* == *値* || *変数名 1* == *値* || *変数名 1* == *値*) && *変数名 2* == *非該当のコード*	(*var_kekkon* == 2 || *var_kekkon* == 5 || *var_kekkon* == 6) && *var_haigunenrei* == 8888
論理的矛盾	合計のチェック	*変数名 1* + *変数名 2* + *変数名 3* != *変数名 4*	*var_tsugakujikan_hour* * 60 + *var_tsugakujikan_minute* != *var_tsugakujikan_total*
	大小関係のチェック	*変数名 1* > *変数名 2*	*var_kyoudaijyunban* > *var_kyoudaininzu*

5 結論

社会調査実習でDCSSを活用して調査データの整理作業を行った後、学生たちに匿名で感想を聞いた。それに基づいて粘土細工アプローチおよびDCSSの利点と課題を整理すると次の通りである。

データ入力作業は集中力を要する大変な作業だが、DCSSのレイアウト機能は変数の配置を自由に変えられるので、作業者は自分にとって楽なレイアウトで作業をすることができ、作業の負担を減らすことができる。

論理式を書いて異常値を検出し、検出された異常値と全ての回答を読んで、ケース単位で修正を施していくという粘土細工アプローチの考え方は理にかなっていて、データクリーニングが初めての初心者でもすぐにその考え方とプロセスを理解し作業に取り掛かることができる。

DCSSを利用すれば、大勢の人で作業を分担することが容易である。そして、各分担者は、現在どの段階のプロセスまで進んでいて、自分がやるべき仕事は何なのかを明確に知ることができる。

課題としては、論理式を書くことが初めての学生が多く、正しい式が書けるまでに時間がかかったことである。ただし、一旦書き方を覚えれば、従来のように度数分布表やクロス表を利用して異常値を検出する方法よりも簡単に感じる場合が多かった。また、論理式を書く練習を通して、変数間の関連について注意深く考えることができ、以前よりも調査票の設計に注意を払うようになったという教育的効果もあった。

謝辞

DCSSの開発におけるGUIデザインおよび本論文の推敲作業において、学習院大学計算機センターの竹内俊子氏のご協力をいただきました。本論文の一部内容を第90回日本社会学会大会にて報告した際に、関西学院大学社会学部の大谷信介教授、関西大学社会学部の保田時男教授から有益なコメントをいただきました。記して感謝申し上げます。Portions of DCSS Software may utilize the following copyrighted material, the use of which is hereby acknowledged. EvalEx: Copyright 2012-2018 by Udo Klimaschewski（http://about.me/udo.klimaschewski, http://UdoJava.com）. The software is licensed under the MIT Open Source license. Twenty-three other contributors contributed to this software（https://github.com/uklimaschewski/EvalEx/graphs/contributors）.

注

1）Excelベースのツールである Inspector および Inspector 2、そしてDCSSのプロトタイプの開発にあたり、独立行政法人日本学術振興会の科学研究費助成事業特別推進研究事業「少子高齢化からみる階層構造の変容と格差生成メカニズムに関する総合的研究」（課題番号：JP 25000001）の支援を受けた。

2）DCSSの最新バージョンは、筆者のウェブサイト（https://www.hepokiki.com）からダウンロードできる。DCSSは無償のアプリケーションであるが、そのダウンロードと使用にはThe MIT License（https://opensource.org/licenses/mit-license.php）が適用される。その主な内容は、DCSSとDCSSと一緒に配布する関連文章は著作権者の許可を得ることなく自由に使用することができるが、それらの使用によって生じる問題について著作権者は責任を負わないというものである。

3）フェレギ・ホルト原則の具体的な内容については、保田の文献に分かりやすく整理されている（保田 2018）。

4）データクリーニング研究に関する用語の日本語訳は、まだ定訳がない状況である。実は「データクリーニング」も、英語では「data editing」という言葉を使うのが一般的である。本稿では「データクリーニング」以外の用語について、独立行政法人統計センター研究センターが2005年に作成した「統計データ・エディティングに関する用語集（対訳）」の訳を使用することにする。この用語集は、国連が1997年に刊行した『Statistical Data Editing、 Volume No. 2、 Methods and Techniques』の用語集（United Nations 1997）を増補改訂して2000年に刊行した「Glossary of Terms on Statistical Data Editing」を翻訳したものである。この用語集は、独立行政法人統計センターのウェブサイト（2018年10月25日取得。https://www.nstac.go.jp/services/words.html）で閲覧することができる。

5）粘土細工アプローチという名称も、「異常なケースについて一つずつ周辺情報を見ながら適切なデータに修正していくという手続きが、歪んだ粘土細工を一つずつ周辺の形に合わせて手直しする感覚に似ている」（保田 2011）ところから命名されたものである。

6）原典では「edit ルール」という言葉が使用されている（保田 2018）。この言葉は「edit rule」を訳したものである。独立行政法人統計センター研究センターの「統計データ・エディティングに関する用語集（対訳）」では、「edit rule」の訳語として「エディット規則」を使用しているので、本稿もそれに従う。

7）Java 8のシステム要件は、Oracle社のウェブサイトで確認できる。

8）DCSSでは、生値（raw value）と編集値（edit value）という用語を使用する。生値は、生データの変数の値のことで、データクリーニングがなされる前の値を意味する。保田の用語法では、元値にあたる。編集値は、DCSSで修正が施された値のことで、データクリーニング後の値を意味する。保田の用語法では、修正値または新値にあたる。

9）式 (3) と (4) に使用された括弧は可読性をよくするためのもので、括弧がなくても式の判定結果は変わらない。

10）コーディングを自動化することで、このような手間を省こうとする研究もある（高橋 2016）。しかし、コーディングの自動化について筆者は次の２つの理由で懐疑的である。第１に、自動コーディングの精度が100％で安定しない限り、人の手による確認作業を排除することはできない。少なくとも現状ではシステムの精度を100％まで上げることが難しく、機械によるコーディングと人によるコーディングの二重作業を回避できない。第２に、現在公開中の職業・産業コーディング自動化システムは、回答の文脈的理解と人物像の文脈的構成の面で課題がある。現在、東京大学社会科学研究所附属社会調査・データアーカイブ研究センターのウェブサイトを通して利用可能な職業・産業コーディング自動化システムでは、職業と産業のコーディングを行う際に、産業と職業の自由回答、そして学歴、従業上の地位・役職、従業先の規模の情報を利用している。しかし、職業と産業のコーディングを行う際に参照すべき情報の範囲はそれより広い。例えば、原は次のように指摘している。「従業先・事業内容・仕事内容についての記述、また、その他の選択式項目あるいは調査地点（従業地）・学歴・収入などとの組み合わせによって、回答者の職業活動像が具体的に浮かび上がってくる。さらには、両親の職業、学歴、職歴などを通して、回答者の生活歴についても同様だ」（原 2013）。参照すべき情報の範囲が広い上に、どの情報を参照すべきかはケースごとに異なってくるので、そのような作業を自動化することは非常に困難であろう。

11）他にも、比率（例えば、同居家族人数と世帯年収との比率関係）や歴史的比較（例えば、パネル調査データの第１波と第２波との比較）など、複雑性の高い組み合わせもある（Groves et al. 2004＝2011）。

参考文献

Delgado-Quintero, Sergio and Juan-Jose Salazar-Gonzalez, 2008, “A New approach for Data Editing and Imputation,” *Mathematical Methods of Operations Research*, 68 (3): 407-28.

Fellegi, Ivan P. and David Holt, 1976. “A Systematic Approach to Automatic Edit and Imputation,” *Journal of the American Statistical Association*, 71(353): 17-35.

Groves, Robert M., Floyd J. Fowler Jr., Mick P. Couper, James M. Lepkowski, Eleanor Singer and Roger Tourangeau, 2004, *Survey Methodology*, John Wiley & Sons.（＝2011、大隅昇監訳『調査法ハンドブック』朝倉書店。）

原純輔、2013、「職業自動コーディング」『社会と調査』11：3。

羅一等、2017a、「社会調査のデータクリーニングシステムの開発と応用――社会調査実習科目の演習への応用」『第64回数理社会学会大会報告要旨集』。

――――、2017b、「社会調査データの統合データクリーニングシステム開発の研究――DCSS の開発と試用」『第90回日本社会学会大会報告要旨集』。

――――編、2018、『専修大学2017年度「社会調査実習 A・B」（羅一等担当クラス）報告書――専修大学生のジェンダー意識に関する社会調査』専修大学人間科学部社会学科。

大谷信介・後藤範章・小松洋・木下栄二、2013、『新・社会調査へのアプローチ――論理と方法』ミネルヴァ書房。

盛山和夫、2004、『社会調査法入門』有斐閣。

菅澤貴之・保田時男、2018、「データ・クリーニング時期別にみたエラー検出傾向に関する基礎的分析」保田時男編『2015年 SSM 調査報告書 1 調査方法・概要』2015年 SSM 調査研究会、143-75。

高橋和子、2016、『職業・産業コーディング自動化システム』平成25～27年度　科研費補助金成果報告書。

United Nations, 1997, *Statistical Data Editing, Volume No. 2: Methods and Techniques*, United Nations.

保田時男、2010、「調査データのクリーニング方法に関する提言――Fellegi-Holt の原則に立ち返る」『第49回数理社会学会大会報告要旨集』。

――――、2011、「NFRJ-08Panel における調査票の設計――研究課題とクリーニングを視野に」『家族社会学研究』23（1）：89-95。

――――、2012、「パネルデータの収集と管理をめぐる方法論的な課題」『理論と方法』27（1）：85-98。

――――、2017、「回顧式家族調査 NFRJ－16R のねらいと経過」『家族社会学研究』29（2）：216-22。

――――、2018、「複雑な社会調査におけるデータ・クリーニング技法の開発」保田時男編『2015年 SSM 調査報告書 1 調査方法・概要』2015年 SSM 調査研究会、177-200。