FAMILY ARREST AND SEPARATION DURING THE HOLOCAUST IN ITALY

by

Maël Le Noc, Licencié d'Histoire, Licencié de Géographie

A thesis submitted to the Graduate College of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Geography
May 2016

Committee Members:

Alberto Giordano, Chair

Ronald R. Hagelman

Sarah A. Blue

COPYRIGHT

by

Maël Le Noc

2016

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

This work is protected by the Copyright Laws of the United States (Public Law 94-553, section 107). Consistent with fair use as defined in the Copyright Laws, brief quotations from this material are allowed with proper acknowledgement. Use of this material for financial gain without the author's express written permission is not allowed.

Duplication Permission

As the copyright holder of this work I, Maël Le Noc, authorize duplication of this work, in whole or in part, for educational or scholarly purposes only.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor and committee chair, Dr. Alberto Giordano, for his advice, guidance, and support throughout these past two years. His pioneer work was, and still is, an inspiration to me, and I cannot thank him enough for providing me the opportunity to explore the geographies of the Holocaust with him.

I would like to thank my committee members, Drs. Ronald R. Hagelman and Sarah A. Blue, whose suggestions, comments, and edits truly strengthened this research.

I would also like to thank Dr. Russell Weaver, for his course on qualitative methods and his availability to discuss my statistical analyses, as well as Ryan Schuermann, whose script was the very first step on the journey that has been this research.

I would like to express my gratitude to Drs. Anadelia Romo, Jennifer Forrest, and Lawrence E. Estaville, who provided me with their full support as I was going through the rocky process of applying to the master's program I am now about to complete.

A special thank you to my roommate and friend Eric Bruton who listened to me ramble about my research for the past two years and offered valuable remarks and edits for this thesis.

Finally, I would like to thank my family, in particular my mother and father, Béatrice and Yann Le Noc, who always encouraged me to pursue my dreams and have provided me with the necessary support and resources to do so along the way, even though I chose to study more than 15,000 km away from home.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF ABBREVIATIONS

Abbreviation	Description
AFG	Arrested Family Group
FSI	Family Separation Index
FSIS	Family Separation Index Score
GEE	Generalized Estimating Equation
GLM	Generalized Linear Model
ID	Unique identification code
NELM	New Economics of Labor Migration

ABSTRACT

Despite the fact that families are specifically targeted by genocide perpetrators, genocide and Holocaust researchers have paid relatively little attention to how patterns of victimization of individuals might differ from those of families. This thesis contributes to the literature by examining Jewish families' victimization during the Holocaust in Italy from a GIScience and historical geographical perspective. Starting from a large GIS database of individual victims of the Holocaust in Italy, a methodology was devised to identify family groups from individuals' lists and to determine if and when families were separated. Then individuals' and families' experiences were compared, focusing on spatio-temporal patterns, the frequency and type of family separations, and the effect explanatory of variables—such as Italians or foreign born families, nationality of perpetrators, gender, and age—on the degree and frequency of family separation and overall vulnerability. Although aggregate statistics do not show considerable differences between individuals and families, marked spatio-temporal patterns arose, especially as it concerns the nationality of both perpetrators and victims. Those patterns also suggest a high vulnerability of family to round-ups. Moreover, statistical analysis revealed that some individuals—such as children, Jews born in Italy, those arrested in 1944 or 1945, and those arrested by Italians or by Italians with Germans—were more likely to be separated than others, and that family separations tended to occur in medium or small camps and prisons.

I. INTRODUCTION

"Does it matter if they were from Kielce or Brno or Grodno or Brody or Lvov or Turin or Berlin? Or that the silverware or one linen tablecloth or the chipped enamel pot - the one with the red stripe, handed down by a mother to her daughter - were later used by a neighbour or someone they never knew? Or if one went first or last; or whether they were separated getting on the train or off the train; or whether they were taken from Athens or Amsterdam or Radom, from Paris or Bordeaux, Rome or Trieste, from Parczew or Bialystok or Salonika. Whether they were ripped from their dining-room tables or hospital beds or from the forest? Whether wedding rings were pried off their fingers or fillings from their mouths? None of that obsessed me; but - were they silent or did they speak? Were their eyes open or closed?

I couldn't turn my anguish from the precise moment of death. I was focused on that historical split second: the tableau of the haunting trinity - perpetrator, victim, witness.

- Anne Michaels, Fugitive Pieces

This quote from Anne Michaels' novel *Fugitive Pieces* exemplifies how the death of millions during the Nazi Holocaust might be so overwhelming that death is ultimately all one can think about. From the perspective of a geographer, however, the processes of the Holocaust, all *of that*, are of great interest. Where were *they* from? When, how and why did they get arrested, deported, separated? Beyond the simple pursuit of knowledge, remembering and understanding the past might help to recognize or anticipate similar events in the present.

Without a doubt, all *of that* was also of importance for the victims of the Holocaust while they were arrested, deported and executed. If one tries to understand how Jews endured their deportation, taking a look at what might have been their lived experience is one of the first steps. When listening to or reading testimonies from Holocaust survivors, family is a recurrent theme. For example, while exploring

microgeographies in oral histories of Holocaust survivors, Tim Cole presents several examples of narratives of familial separation (Cole 2015). Family separation during the Holocaust has also been studied in the field of psychology and shown to be a particularly traumatic experience, especially for children (Benz and Axelrod 2005), and geographers have shown that the family is a relevant unit of analysis for the study of human migration (Dumon 1989). Yet no extensive research has been done regarding family deportation and the extent of family separation during the Holocaust.

The unique HGIS (Historical GIS) created by Giordano and Holian (2014) based on Picciotto Fargion's (2002) work, which records various information for about 90% of the 10,000 victims of the Holocaust arrested in Italy, will make possible an analysis of the spatial and temporal patterns of the Holocaust at the scale of the family in the Italian case. While Giordano and Holian (2014) examine the pattern of arrest at the individual scale, the focus here is on the family group. The question asked is: how do family patterns of persecution during the Italian Holocaust differ from individual patterns ?

II. HISTORICAL BACKGROUND

In response to the September 8, 1943 armistice between the Kingdom of Italy and the Allies, who were in the slow process of liberating the southern part of the peninsula, Nazi Germany immediately seized control of northern and central Italy. Less than a week later, the Repubblica Sociale Italiana (RSI), a fascist puppet state officially led by Mussolini but actually subjugated to Germany, was created to rule over most of these territories. In addition, the German third Reich directly annexed the Operational Zone of the Alpine Foothills (*Operationszone Alpenvorland*) as well as the Operational Zone of the Adriatic Littoral (*Operationszone Adriatisches Küstenland*). The Nazis' taking of control marked the beginning of the Italian "hunt for Jews" (*caccia all'ebreo*).

While the systematic deportation and extermination of Jews on Italian territory began with German occupation, anti-Semitism and persecution of the Jewish population in Italy started years before. In 1938, the Italian fascist government introduced anti-Jewish racial laws, promulgated an expulsion decree of Jews who had immigrated to Italy after 1919, and set up a Department for Demography and Race (*Direzione Generale per la Demografia e la Razza*) in charge of managing and implementing anti-Jewish policies, as well as conducting a racial census.

Five years later, when Dannecker came to Italy with a SS unit in order to implement the "final solution," this census and the work of the Department for Demography and Race revealed themselves to be quite convenient. The principal roundups were carried out during fall 1943, especially in Rome on October 16; during November in Genoa, Milan, Florence and Trieste; and in Venice in December. After those first mass arrests by the Germans, the Italian police were put in charge of arresting Jews in the RSI territory and proceeded to more systematic small-scale roundups. Once arrested, Jews were mainly interned in Fossoli-Carpi, in Bolzano, and in Risiera di San Sabba before being deported to Germany, principally to Auschwitz (Picciotto Fargion 2000; Bensoussan and Marie 2014). Figure 1 is a

map situating the principal events and places of the Holocaust in Italy.

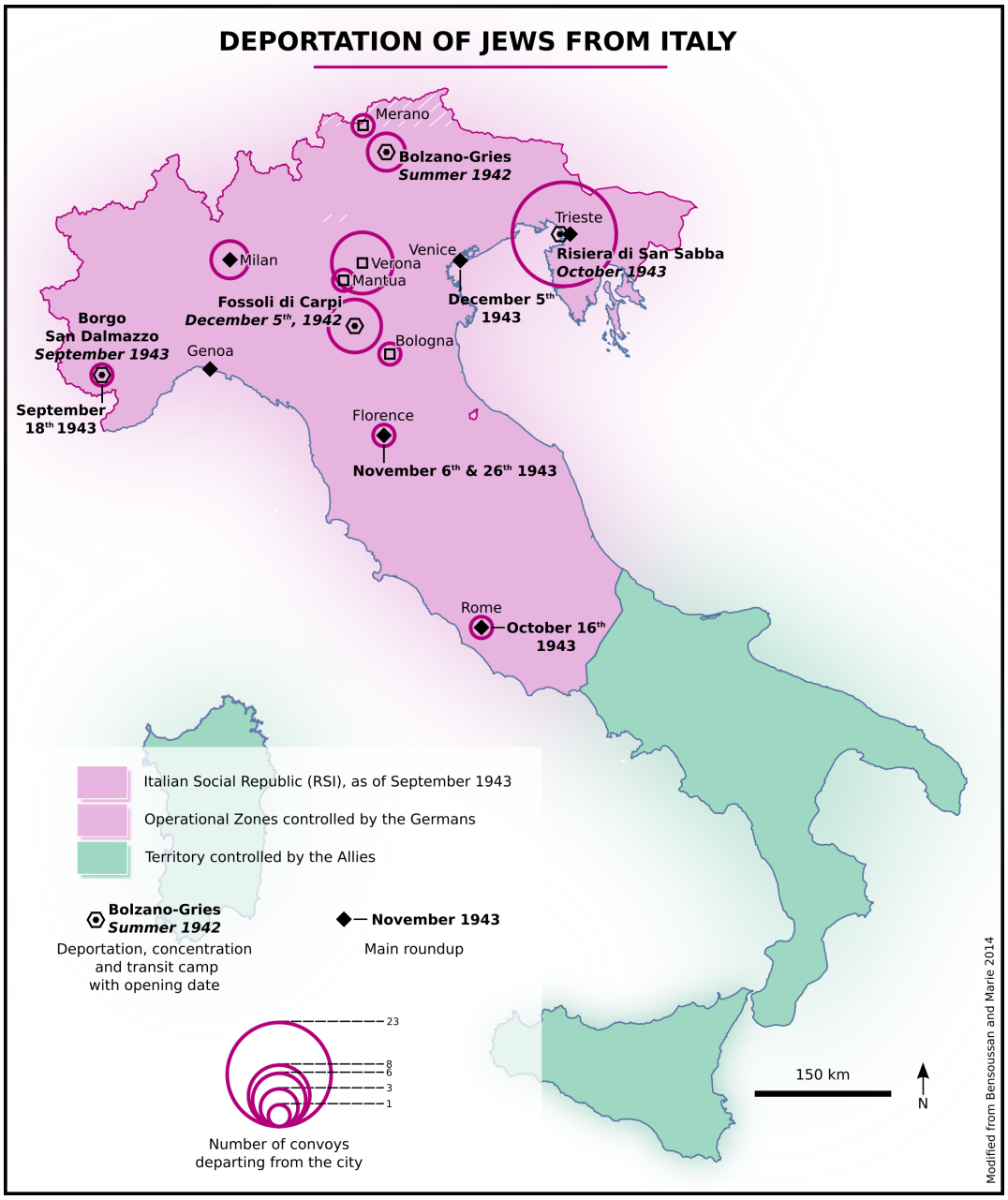

FIGURE 1: Deportation of Jews from Italy. *Modified from Bensoussan and Marie 2014 using data from Picciotto Fargion 2002*

When deportation began, the proportion of Jews in the Italian population was low, less than 0.1% in the late 1930s (Giordano and Holian 2014). In September 1943, an estimated 32,307 Jews (Italian or foreign) were present in the occupied part of Italy and thus subject to arrest and deportation. Of this number, about 10,000 were actually arrested (31%), and 8,500 were killed (26%). This constitutes

one of the lowest victimization rates of all the European countries under German occupation. Among the Jews deported from Italy were about 400 Jewish residents of Libya, who had been previously transported by the Italian authorities from Libya to the Italian mainland in 1942. Those Libyan Jews were mostly transferred to Bergen-Belsen, and virtually all of them survived (Picciotto Fargion 2002; Seibel 2002). Additionally, about 200 Jews died in Italy before being transferred to Eastern Europe Nazi camps, killed by perpetrators, often because they were trying to escape, or committing suicide. Finally, about 1,800 Jews were arrested in Rhodes and the neighboring island of Kos, two islands in the Aegean Sea that were then under Italian ruling and are today part of Greece. On July 20, 1944, they were deported to Auschwitz-Birkenau and only 153 survived (United States Holocaust Memorial Museum 2013).

While the identity, the origin, the destination, and the fate of the Jews deported from Italy between 1943 and 1945 is, in broad line, well established, there has been little systematic analysis of arrest and deportation proceedings. In order to gain awareness of what the victims of the "Final Solution" might have experienced, this study aims to shed light on family arrests and separations during the Holocaust in the specific context of Italy.

III. PURPOSE STATEMENT

The purpose of this research is to analyze the geographic patterns of deportation during the Holocaust at the scale of the family. Linking space and social structure, this study aspires to aid in the understanding of the variable effective implementation of the “Final Solution" and how criteria such as the age and the gender of the victims or the nationality of the perpetrators may have played a role. Researching family separation also aims to better understand the particular experiences of the victims.

More broadly, this study may prove family to be a relevant scale for the study of forced migration patterns. If this should be the case, it will provide new techniques and methods to extensively study family involuntary migration and separation. These techniques may be applied in a variety of research topics in the fields of migration studies and historical geography such as studies of refugees, forced migrations, displacements or genocides (Rwanda, Soviet Union, etc.).

In order to meet these objectives, the proposed research aims to answer the following questions:

- What were the spatio-temporal patterns of Jewish family arrests and deportations in Italy between September 1943 and February 1945?

- Are those patterns different from the spatio-temporal patterns of arrest at the individual scale presented in Giordano and Holian 2014 ?

- How often, and under what conditions, were family units maintained, separated, and/or reunited?

Figure 2 represents a model of the conceptual framework guiding this research. It hypothesizes that spatio-temporal patterns of family arrest might differ according to the nationality of the perpetrators and the gender, the age and the nationality of the victims. These factors are also likely to influence different types of family separations, particularly regarding the stage of deportation at which it occurs.

This framework recognizes that family separations might have occurred before or

during arrest; however, due to the specificities of the dataset on which the research is based, this study focuses mainly on separation after arrest. The examination of family separations that occurred prior to or at the point of arrest are only included in a few specific cases due to restrictions on how the database permits the identification those types of separations.

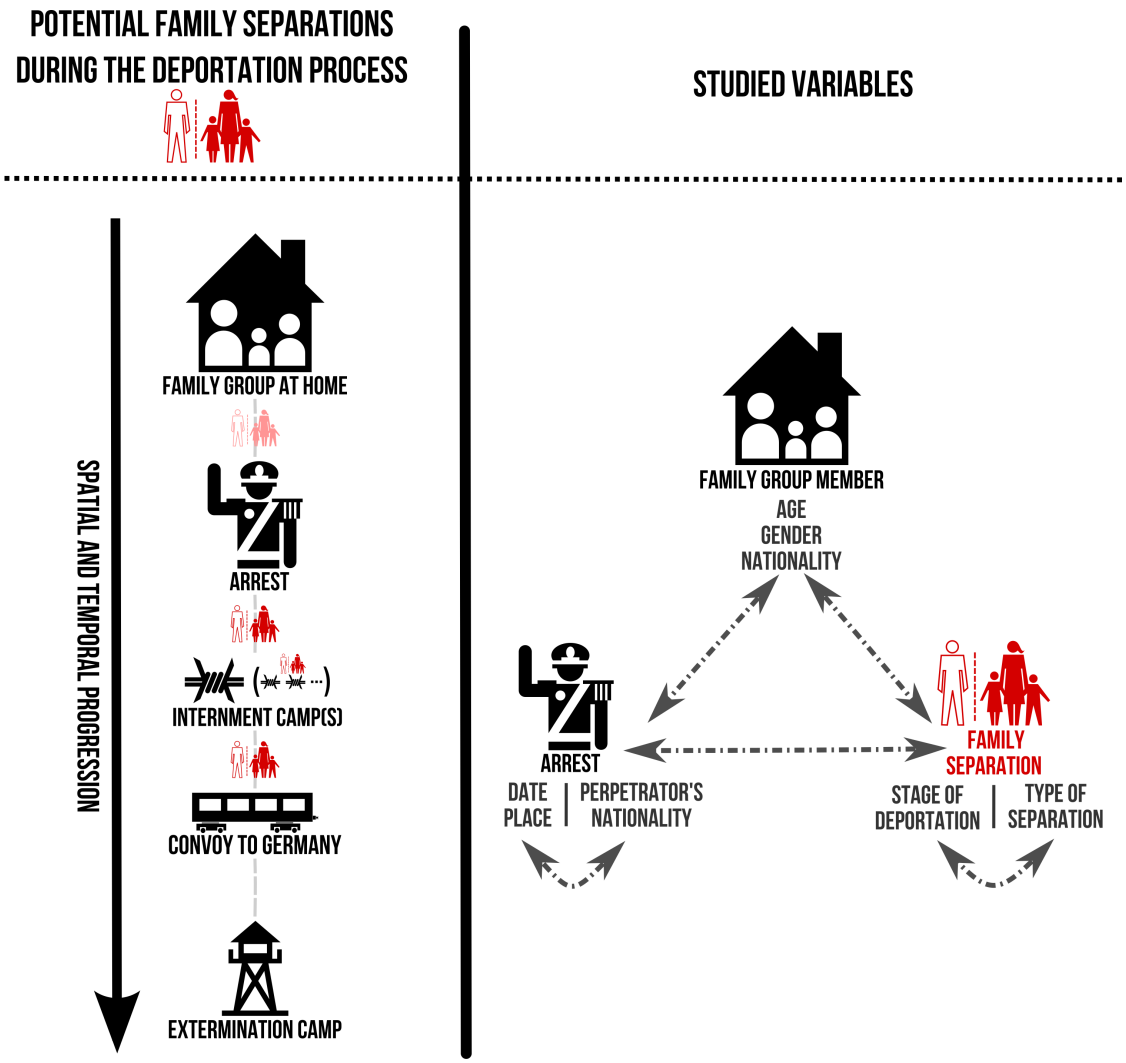

FIGURE 2: Conceptual Framework

IV. LITERATURE REVIEW

The literature relevant to family deportation is part of two broad fields of study: the field dealing with the Holocaust and genocide in general, and the field researching migrations and mobilities. Figure 3 illustrates the relation between the proposed research topic and those two fields of study.

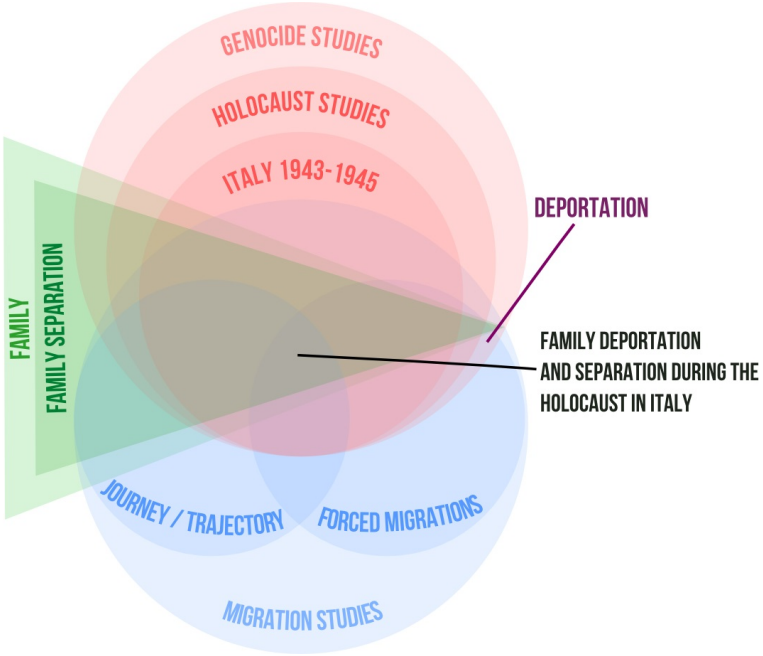

FIGURE 3: Situation of the proposed research topic within existing literature

Studies on the Holocaust in Italy are often part of a broad body of literature that includes not only the 1943-1945 period but also the preceding fascist period starting with the appointment of Mussolini as prime minister in 1922. These studies cover pre-World-War-II anti-Semitism, 1938 racial laws and their origins, Jews' life under fascism, Italian participation in the Holocaust, and the involvement of the Catholic Church (Zuccotti and Colombo 1987; Zimmerman 2005; Cooperman and Garvin 2000). The literature often emphasizes the specificity of the Holocaust in Italy, namely that it took place under German supervision. Some researchers have focused specifically on German perpetrators either in the early stages of the

Holocaust, such as the round-up of the Jews of Rome in October 1943 (Katz 2005), or in German occupied regions in the Italian Northeast (Villani 2005). Hence, the role of Italians, the extent of their involvement in the deportation process, and their collaboration with Germans are some of the recurrent topics. These questions are linked to the traditional and popular vision of Italians as *brava gente*, which tends to present Italians as rescuers and dissociate them from the German perpetrators: however, this interpretation has repeatedly been challenged and scholars have sometimes considered it as a myth (Herr 2014). Picciotto Fargion (2002) has written the most comprehensive and detailed work regarding quantitative aspects of the Holocaust in Italy. After compiling lists of about 90% of all Jews deported from Italy (Picciotto Fargion 2002), she published a series of book chapters and articles focusing on the chronology, the mechanisms and the proceedings of Jewish arrests and deportations in and from Italy. She presents the different stages of the Holocaust (large scale round-ups at first and the systematic search over the entire Italian territory subsequently) and the relations and cooperation between Germans and Italians, particularly the fact that during the second stage, Italians were mainly in charge of arrests and internment while German organized deportations (Picciotto Fargion 2005, 2000; Picciotto 2005)

Picciotto Fargion's writings from 2005 exhibit the premises of some considerations of space and time in the study of the Holocaust in Italy. Other early (and some more recent) geographical works on the Holocaust were generally limited to atlases mapping the principal locations of the Holocaust (camps, ghettos, train lines, etc.) associated with global numbers of victims but omitting many of the spatial facets of the Holocaust, especially those associated with human landscapes and people's experiences (Gilbert 2009; Bensoussan and Marie 2014). More recently, using Picciotto Fargion's data, an in depth spatio-temporal analysis of arrests of Jews in Italy has been conducted (Giordano and Holian 2014). In this inductive and quantitative study, statistics, cluster analysis, and the Knox index (a statistical index of spatial and temporal proximity between events) were used to find patterns that were

both spatial and temporal. Beyond these methodological innovations, results reveal that the fate of Jews in Italy was influenced by factors such as their age, their nationality, their location, and who arrested them. The study also outlines alternating stages of spatio-temporal dispersal and concentration and shows national patterns of arrest to be correlated with Jewish population centers, although a considerable proportion of Jews were arrested trying to escape. This work is part of the Holocaust Geographies Collaborative project, the first attempt to extensively look at spaces and places of the Holocaust. This interdisciplinary project is composed of a series of case studies using geovisualization and spatial analytical methods to study the geography and the chronology of the Holocaust at different scales (Knowles, Cole, and Giordano 2014; Beorn et al. 2009). It is worth mentioning that, according to their own conclusions, some of the limits of this project are the seeming insensitivity and lack of humanity of spatial analysis and scientific mapping with GIS. These pitfalls have to be carefully considered in the framework of this proposed research. Additionally, the topography of the Holocaust has been the subject of an essay by Charlesworth (2004), and space is a running theme in the work of Gigliotti (2009).

Contrary to the growing literature linking space and time in the Holocaust field of study, the literature dealing with family deportation and/or separation is still almost non-existent despite the fact that scholars studying migration as well as genocide have shown that family is a topic particularly relevant for each of their respective fields of study (Dumon 1989; von Joeden-Forgey 2010).

Within the genocide literature, there are several potential explanations for the lack of attention toward using the family unit as a study variable, despite its relevance: the assumption that families are studied “by default" because of their inherent ties to genocidal violence, the fact that that social sciences have traditionally not paid much attention to families, and the absence of families from the international legal documents relevant to genocide (von Joeden-Forgey 2010). Yet, “genocide [...] is a crime that is inextricably tied to families" (1). Indeed, family is targeted by perpetrators as the place of reproduction of human life and the place of cultural

perpetration through education, but also for practical reason as members of the targeted group can conveniently be found in households.

The relevance of family for Holocaust studies is also supported by the fact that family is a recurrent topic in Holocaust narratives and testimonies (von Joeden-Forgey 2010; Cole 2015). For instance, the work of Tim Cole on "(re)placing self and others in the past [as] one way of telling difficult stories" (Cole 2015, 30) presents several examples of family separation. In his focus on oral histories and microgeographies within survivors' narratives, he argues that survivor testimonies' inconsistencies in "placing" themselves and others while telling their stories of familial separation are part of a broader set of strategies of control exercised by individuals when retelling their past (Cole 2015). Cole's conclusion also echoes a broad body of literature in psychology dealing with the effects of family separation on Holocaust survivors, showing it to be a particularly traumatic experience, especially for children (Benz and Axelrod 2005). It is worth noting that most of those studies are dealing with separation due to or during the arrest, while the research proposed here does not examine only those cases but also family separation during the deportation process, after arrest. That being said, testimonies of those who might have actually been separated during the deportation process are rarely available due to the fact that most victims did not survive the Holocaust. A few attempts of representation of specific family trajectories displaying separation while being deported are available (Giordano and Holian 2014), but they tend to be limited to the illustration of family histories and the separation is not actually discussed.

The deportation of Jews to death camps during World War II, as well as other types of deportation, meet the criteria of forced migration as defined by Petersen (1958): impelled migration occurs "when the migrants retain some power to decide whether or not to leave," while forced migration occurs "when they do not have that power" (261). Yet, the Holocaust is rarely mentioned in forced migration studies (and even less in more general migration studies). Indeed, forced migration studies

and refugee studies, two closely associated fields (Hathaway 2007), have heavily focused on policy and legal issues, and especially forced migrant and refugee statuses and rights (Bakewell 2008). Similarly, deportation (as practiced by modern regimes) has been studied in term of international law, but almost never as political or historical practices (Walters 2002). In that sense, since the fate of Jews deported to death camps prevents any concerns regarding their rights or their status, studies of the Holocaust have been largely excluded from forced migration studies. Yet, the similarities of situations and experiences between refugees and deportees to death camps make part of the literature related to forced migration relevant for this study, at least from a methodological perspective.

The family has long been proven to be a relevant scale to study migrations. In 1989, the June issue of International Migration Journal had a section entitled “Effects Of Migration On Family Structure". The introductory article presented the history of research on migration dealing with family and made the case to continue to use that scale of analysis (Dumon 1989). More recently, Castles, Haas, and Miller (2014) summarized the New Economics of Labor Migration (NELM) and household approaches as seeing migration as a risk-sharing behavior among families or households or as a family or household strategy to provide resources for investment in economic activities (i.e. the family farm), and as a response to relative deprivation rather than absolute poverty. These approaches emphasize migrant agency, considered within the frame of households, to improve livelihoods and reduce financial instability. While such theories highlight the importance of taking family into consideration when studying migration, they, as virtually all migration theories, assume that migrants have a certain level of agency and, although their movement may be subject to a variety of constraints, that they can choose to move or not. This was clearly not the case for Jews deported during the Holocaust.

From the field of refugee studies, multiple research projects from a wide range of disciplines, such as psychology (Goldstein, Wampler, and Wise 1997; Luster et al. 2009; McGregor, Melvin, and Newman 2015), humane biology (Pesonen et al. 2008),

sociology and social science (Smit and Rugunanan 2015; Wilmsen 2013; Dreby 2015), as well as law and policy (Enchautegui and Menjívar 2015), have focused on family separation. Virtually all of these studies identify involuntary family separation as a trauma, regardless of the context of separation (war, deportation, etc...). Among all the cases studied, the vast majority are concerned with the separation of one or both parents from their children. This body of research shows that the experience to be emotionally difficult and to have psychological and physiological effects on both parents and children (Pesonen et al. 2008; Smit and Rugunanan 2015; Dreby 2015).

Most of these studies focus on long-term impacts of such separation, especially those having negative consequences on settlement, integration and well-being (Enchautegui and Menjívar 2015; Pesonen et al. 2008; McGregor, Melvin, and Newman 2015; Wilmsen 2013). For instance, young people that have been separated from their parents demonstrate posttraumatic stress disorder syndromes (McGregor, Melvin, and Newman 2015) and significant alteration in reproductive and marital traits (Pesonen et al. 2008). This focus on the post-migration situation of refugees echoes a more general problem with the global field of migration studies; the fact that very little attention has been paid to the actual journey between departure and arrival (Burrell 2008; Schapendonk and Steel 2014). The few exceptions to that observation emphasize the importance of the journey for the "travelers," or the deportee in the case of this study, during which one's experience leads to some personal transformation. They also state that social relations change through mobility (Burrell 2008; Schapendonk, Liempt, and Spierings 2015).

Nevertheless, some of the studies on family separation previously mentioned do indeed discuss the experience of refugees immediately after the separation, which is relevant to the study of Jewish deportees during the Holocaust. Among those impacts are strong negative emotions, including sadness, loneliness, fear, and worry (Pesonen et al. 2008), biomedical forms such as sleeplessness, nightmares, worry, guilt, depression, poor concentration, and physical symptoms such as headaches,

pain and difficulty of breathing (Wilmsen 2013).

Research that compare constrained choice separation to involuntary separation shows stronger emotional anxiety in the latter situation, especially due to a feeling of uncertainty (Dreby 2015). These circumstances, comparable to the situation of Holocaust deportees being split from their family without knowing what was to be their fate, have been called “ambiguous loss," a term describing a “situation of unclear loss resulting from not knowing whether a loved one is dead or alive, absent or present" (Boss 2004). Such ambiguous losses are particularly stressful for adults as well as children, leading to depression, hopelessness, and immobilization (Boss 2004; Luster et al. 2009).

Finally, a series of articles from the field of refugee studies emphasizes the importance of gender studies researching lived experiences and perceptions of forcibly dispersed migrants. (Smit and Rugunanan 2015; Dreby 2015). While both men and women may experience the physiological, emotional and psychological alterations listed above (Pesonen et al. 2008; Goldstein, Wampler, and Wise 1997), and while fathers as well as mothers fear involuntary separations, gender does structure family separation (Dreby 2015). Gendered family separation can have various causes (legislation, economic choice, culture, etc.), but they all tend to have similar outcomes. Separation of children from their mother tends to be the most distressing, while separation from their father is sometimes normalized (especially in the case of constrained choice separation). Moreover, mothers tend to experience very elevated levels of stress when becoming the head of families due to the separation from their male partners (Dreby 2015).

V. DATA AND METHODS

In 2002, Liliana Picciotto Fargion published a revised edition of her 1991 book comprising a list of about 9,000 Jews deported from Italy during the Holocaust (Picciotto Fargion 2002). Picciotto Fargion's work is the most authoritative on the quantitative aspects of Holocaust in Italy. Moreover, unlike records compiled by other scholars at the local level, her list is comprehensive and consistent in methodology, assumptions, and stated limitations. From this list, a database was derived and then acquired by Giordano and Holian (2014). During their project on spatio-temporal analysis of arrests during the Holocaust in Italy, Giordano and Holian (2014) revised and updated the database, adding latitude and longitude for destination camps and places of arrest and internment, as well as attaching new attributes such as the nationality of the perpetrators or the last place of residence of the victim. This updated dataset is the one used in this research in order to analyze family arrest, deportation and separation.

The database contains the names of the 6,116 deported victims for which the location of arrest was known, along with various information regarding their identity, their family, their arrest, their deportation and their fate. Those attributes are reported in Table 1. Some of those variables are unknown for some of the individuals in the dataset. For each victim, the dataset reports up to six locations of detention in Italy, not including their place of arrest nor their destination camp outside of Italy, and those locations are listed in order, allowing us to rebuild each victim's deportation trajectory. Finally, the dataset provides geographical attributes (longitude and latitude) for the places of arrest, the Italian concentration camp in which each deportee was detained immediately before being deported outside of Italy by convoy, and the destination camps.

This research includes extensive data treatment and improvement as well as statistical and spatio-temporal analysis of family arrest and family separation. Figure 4 presents these major steps and their articulations.

TABLE 1: Variables available in the original dataset

Variables in the original dataset
-Identity of the victim
Unique ID number
First name
Last name
Place of birth (city and country)
Date of birth (from which the age at the time of arrest is derived)
Stage of life at the time of arrest (child, adult, or elderly)
First name of father
First and last name of mother
First and last name of spouse
-Arrest and deportation of the victim
Last known city of residence
Place of arrest
Date of arrest (day, month and year)
Detention locations in Italy
Convoy number
Concentration or extermination camp outside of Italy
Fate
-Identity of the perpetrators of the arrest
Nationality of the perpetrators (Italians, Germans, or Italians with Germans)

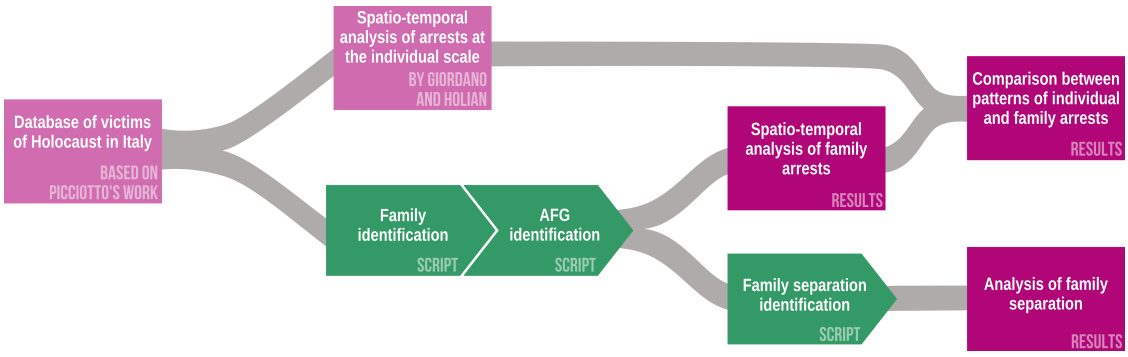

FIGURE 4: Major steps of the research

Data improvements and corrections

In order to study the Holocaust at the scale of the family, this project uses two different units of analysis: a) arrested family group (AFG) and b) individuals belonging to one of those groups. For the purpose of the proposed study, an AFG is defined as a group formed by two or more members of a unique family all arrested at the same time (same day) and in the same place (same city). The unit of analysis varies accordingly to the variable analyzed. For instance, AFGs are relevant when studying places of arrest or perpetrators, but the individual as a unit is needed to study gender or age as explanatory variables. Because individuals in the database are not grouped by family or by AFG, the first step of the research was to define family and to create two variables, a family number unique for each family (family ID), and AFG number unique for each AFG (AFGID). For practical reasons and because of the characteristics the data available, a family is defined here as all the persons related by blood or by marriage to any degree. Figure 5 shows the relations and differences between families and AFGs as defined for the purpose of the study. It especially reveals that individuals from a unique family might have very tenuous relations. For instance, Grazia Calo' is the sister in law of the sister in law of Pellegrino Vivanti's son and, despite the fact that their family tie seems to be relatively weak, they both have the same family ID. Without any supplemental information, it is difficult to know how such distant family members associate with each other, and therefore studying family separations based on family ID is likely to be irrelevant in many cases. This is why, based on the assumption that family members arrested at the same place at the same time probably live or frequently socialize with each other, most of the separation analysis was done using AFGIDs.

For each of the 6,116 individual in the initial database, two Python scripts successively derived a family ID and an AFGID from the names of their mother, father, and spouse, as well as their place and date of arrest.

The first script (Appendix A) was designed to spot possible inconsistencies (mis-

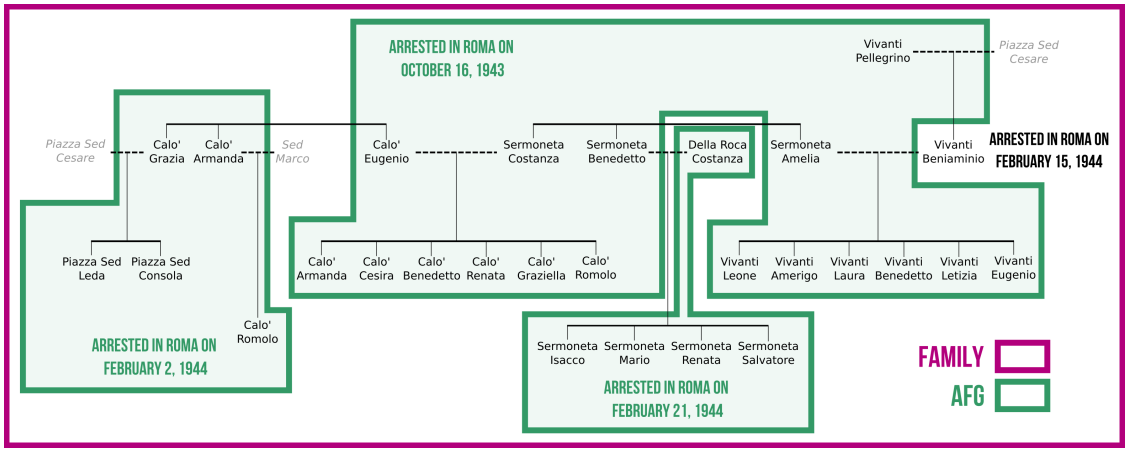

FIGURE 5: Comparison of family and AFG as defined for the purpose of the study *Individuals in light grey are individuals for which no entry exists in the database.*

spellings, different or incomplete information for members of the same family) and to assign a family ID number to each victim, if applicable. For any two individuals, the script then compares the names of their mother, father, and spouse, and assigns the same family ID to individuals that share the same two parents (siblings), are married to each other (spouses), or for which one's parent's spouse matches their second parent (parent and child). For instance, when trying to match a daughter and her father, the script checks if the name of the father's spouse matches the name of the daughter's mother. In all those cases, if one of the matched individual already has a family ID, the same ID is assigned to the second individual. In the case of an incomplete match between two individuals, for instance if the spouse of an individual has a different name of his or her spouse's spouse (himself) or if a name is missing, the script returns the inconsistency in the logfile as a possible issue (See Appendix B for a sample of possible errors returned). The script also creates a table of social network relations using code numbers to specify the relationship between any two individuals. This table was not used for analysis, but revealed itself useful to verify and understand the structure of specific families.

Once the script had run once, the logfile was carefully reviewed, and every possible issue was checked by hand, with the objective of correcting or completing the database when possible and appropriate. If names that should or may have

matched (individual and spouse's spouse for instance) were recorded but did not match across individuals, the reviewer looked for misspelling and uses of middle name or nicknames. Any such errors were corrected in the initial database in order for the two names to be identical. If the analysis was to be run again, this step might be partially automated by using an analysis based on Levenstein distances (the minimum number of edits needed to change one word to another), but the judgment of a human reviewer cannot be totally replaced by a script (for instance, an analysis based on the Levenstein distance would usually exclude nicknames). In the case of names that should or may have matched and were instead different, the inference was a false positive involving homonyms. For example, if John Doe's recorded parents are Jack Doe and Jane Roe while Jane Roe is married to John Smith, the Jane Roe in the database is unlikely to be John Doe's mother (especially since there was no divorce in Italy at the time). In that case, the false positive was identified as such and no changes were made to the database. In order to limit the number of false positives, a condition was added to the script stating that parents have to be at least 13 years older than their children. Lastly, when names were missing, for instance John Doe is married to Jane Roe, but Jane Roe has no spouse recorded, the reviewer first checked for homonyms that matched (perhaps there is another Jane Roe in the database that would be married to a John Doe), and if it was not the case, the reviewer exercised his best judgment to assess if the two individuals were likely to be actually related or not. This step especially involved the comparison of their age, their places of residence and arrest, their date of arrest, and their respective kinship with other people. Information was added in the database only if the reviewer estimated the relation between the two people likely enough. When all the possible errors were checked and the applicable modifications were made, the script was run again using the corrected database. The new logfile was analyzed and the procedure was repeated until no new possible errors were detected. Finally, the resulting family IDs of each individual were added to the database as a new variable.

The second script (Appendix C) was designed to create AFGs from the following variables: individual ID, family ID, place of arrest and date of arrest. The nationality of the perpetrator (Italian, German, Italian with German, or unknown), the nationality of the victims (Italian, non-Italian, or unknown), the places of internment in Italy, the convoy number in which they were transferred from Italy to Germany, and the place of concentration in Germany, were used by the script in order to characterized each AFG. The script assigns a unique AFGID to all the persons who have been arrested on the same day, at the same place, and who have the same family ID. This AFGID was manually added to the initial database as a new variable for each individual. The script also returns a list of AFGs along with their place and date of arrest, nationality (Italian, non-Italian, mixed or unknown), the nationality of perpetrator and family ID.

Finally some subsets of both the individual and the AFG listings were created for each relevant variable (such as gender and age for individuals, and nationality and period of arrest for AFG) in order to facilitate the analysis.

Patterns of family arrests

The analysis of spatio-temporal patterns of family arrests performed followed the same techniques and procedures adopted by Giordano and Holian (2014) in their study of spatio-temporal patterns at the individual scale in order to compare the results. The steps followed are represented in Figure 6.

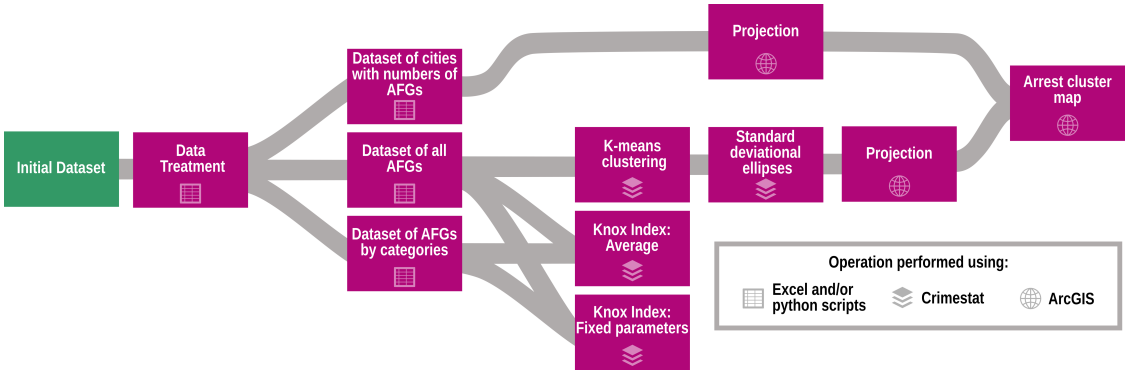

FIGURE 6: Flow diagram of performed analysis of family arrests

Before the analysis of arrest patterns per se, descriptive statistics and percentages regarding the composition of all AFG and the characteristics of AFG were calculated, including, but not limited to, the average number of members of an AFG and its standard deviation, the average proportion of men, women, children, adults and elderly, and the percentages of AFG for each perpetrator, places of arrest, period of arrest, and nationality. For the sake of facilitating comparison, most of those statistics were created to match those presented by Giordano and Holian (2014) and reproduced here in Table 2.

In order to assess the overall geography of AFG arrests, maps depicting the number of AFGs arrested at each place were created with ESRI ArcGIS, as well as maps of AFGs arrested at each place broken down by relevant variable value based on the subset created previously. The temporality of arrest was represented using histograms of number of AFG arrested broken down by perpetrator for each month between September 1943 and February 1945.

Subsequently, a map similar to the one produced by Giordano and Holian (2014) representing the spatial characteristics of AFG arrests clusters, reproduced in Figure 7, was created to analyze the geography of the Holocaust in Italy more closely. First, using Crimestat, the AFGs were split over five clusters, using K-Means clustering. In order to allow comparison, the number of clusters was chosen to match the one used by Giordano and Holian (2014) which was judged to be a " good compromise for the variables mapped and the size and shape of the area under study" (61). When using the K-Means clustering tool in Crimestat, standard deviational ellipses that represent the central tendency, dispersion, and directional trends of each cluster are automatically created. Those ellipses were then projected with ESRI ArcGIS and added to the map of number of AFG by city previously created.

In the following phase, spatio-temporal proximity was measured using the Knox index with the Crimestat software. The Knox index categorized any two events (two arrests of one AFG) as close in time and space, close in space but not in time, close in time but not in space, or not close in space nor in time, either according

TABLE 2: "The Numbers of the Holocaust in Italy" (Giordano and Holian 2014)

Place of arrest	Number	Perc.	Place of internment	Number	Perc.
Rome	1,692	27.7	Fossoli	2,174	35.5
Trieste	551	9.0	Trieste	1,053	17.2
Borgo San Dalmazzo	329	5.4	Rome	1,029	16.8
Florence	274	4.5	Milan	724	11.8
Milan	270	4.4	Verona	383	6.3
Turin	207	3.4	Borgo San Dalmazzo	329	5.4
Venice	204	3.3	Bolzano	185	3.0
Fiume/Rijeka	179	2.9	Florence	56	0.9
Civitella del Tronto	155	2.5	Mantova	42	0.7
Genoa	108	1.8	Merano	35	0.6
Other locations	2,147	35.1	Bologna	31	0.5
Total 10 highest	3,969	64.9	Turin	3	< 0.1
Total known	6,116	100	Suzzara	1	< 0.1
Not known	*682*		*Not known*	*71*	*1.2*
Total	6,798		Total	6,116	100

Gender	Number	Perc.	Fate	Number	Perc.
Male	2,916	47.7	Deceased	5,388	88.1
Female	3,194	52.2	Liberated	711	11.6
Not known	*6*	*0.1*	*Not known*	*17*	*0.3*
Total	6,116	100	Total	6,116	100

Place of birth	Number	Perc.	Destination camp	Number	Perc.
Born in Italy	3,902	63.8	Auschwitz	5,487	89.72
Born outside of Italy	2,114	34.6	Other camp	524	8.57
Not known	*100*	*1.6*	*Not known*	*105*	*1.72*
Total	6,116	100	Total	6,116	100

Age	Number	Perc.	Perpetrator	Number	Perc.
Children (16 or younger)	864	14.1	Italians	1,786	28.2
All adults (17 or older)	5,155	84.3	Germans	2,410	39.4
Of which:			Germans & Italians	319	5.2
Elderly (70 or older)	533	9.0	*Not known*	*1,601*	*26.2*
Not known	*97*	*1.6*	Total	6,116	100
Total	6,116	100			

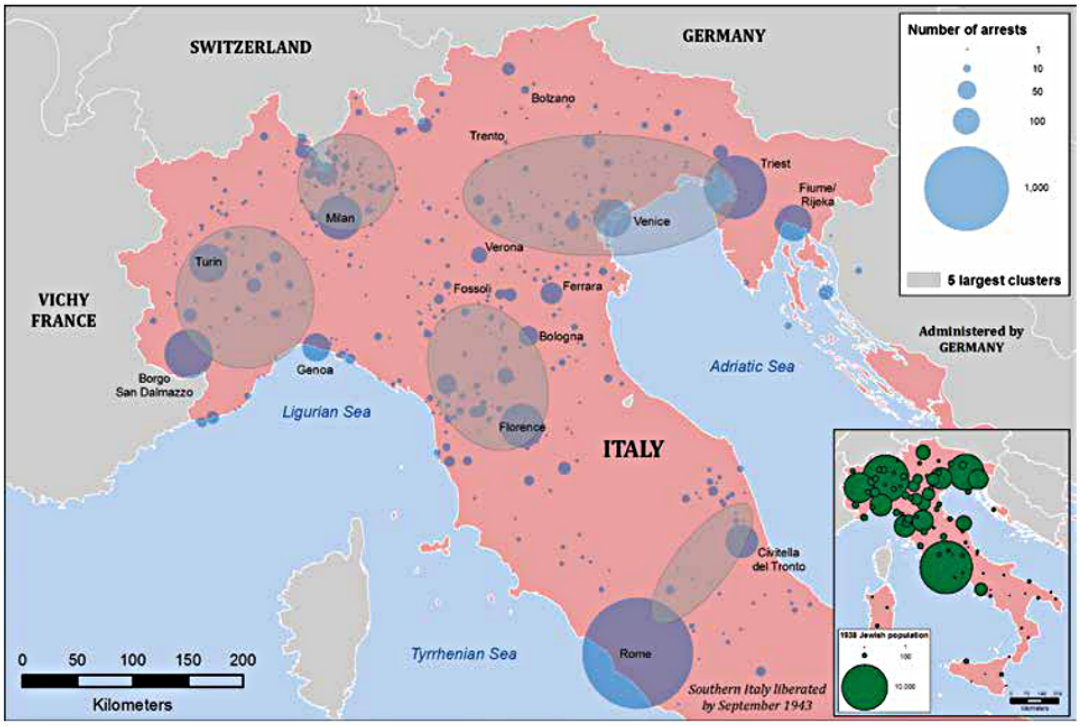

FIGURE 7: "Arrest clusters compared to 1938 Jewish population" from Giordano and Holian (2014)

to a definition of closeness given by the analyst or using average values. For this study, the analysis was run defining spatial closeness as less than 25km and temporal closeness as less than one month, once again to match Giordano and Holian's (2014) analysis. The percentage of any the four cases was collected and compared for all the AFGs and for subgroups broken down by perpetrators and nationalities. For each subgroup, as well as for the whole dataset, the average values returned by Crimestat were also collected.

All the numbers and results from the previously described analysis and operations were compared to those obtained by Giordano and Holian (2014) in order to assess any particularities of AFGs or detect biases related to the use of AFG as a unit.

Family separation

This step of the study aims to analyze family separation during the deportation process (i.e. after arrest). A third python script (Appendix D) was used to detect and record such family separations. This script compares the steps and places of deportation of every victim recorded in the database with the deportation sequence of every other member of his/her AFG. Since the database provides places of detention for each victims but no dates, the detection of separation is based on the assumption that two members of a same AFG remained together as long as they were detained in the same places in the same order.

In order to analyze the vulnerability of the victims to family separation during deportation, a Family Separation Index (FSI) was designed and each victim belonging to an AFG was assigned a Family Separation Index Score (FSIS) for each of the four deportation stages presented in Table 3. All FSISs are comprised between 0 and 1. A FSIS of 1 indicates that an individual has not been separated from any member of his/her AFG. Conversely, the lower one's score is, the higher the number and severity (in term of trauma and psychological effects) of separations experienced by this individual. At the time of arrest, the FSIS of each victim is 1. Indeed by definition, at the time of arrest, a victim is in the company of all the

people with whom he or she has been arrested. Then, the python script compares the trajectory of each victim with all the other member of his/her AFG and each time an individual is separated from one or several other members of his/her AFG, his/her FSIS is multiplied by a coefficient depending on the type of separation, which are presented in Table 4. Those coefficients have been assigned to separation situations based on the literature (cf Chapter IV, last five paragraphs) to reflect the emotional and psychological difficulty of the different types of separation. It is worth noting that this index is tentative and based on a subjective understanding of the literature. It is, however, particularly useful in comparative studies. If this index was to be reused, it might be appropriate to perform a sensitivity analysis of the model, particularly regarding the coefficients. For this study, since the present analysis showed very limited correlation between FSIS and any of the dependent variables used, no such sensitivity analysis has been performed.

TABLE 3: Deportation stages as defined for the family separation analysis

Stage	Definition
1 *	After arrest and before arrival to the first place of detention
2 **	Between the first place of detention and the second place of detention
3	Between the second and the last place of detention in Italy
4	Between the last camp of detention and deportation out of Italy by train
5 ***	After deportation out of Italy by train

* Stage 1 might not exist if the victim has been transferred from the place of arrest directly to the camp of detention from which they have been deported out of Italy.

** Stage 2 might not exist if the victim has been detained in only one place between the place of arrest and the camp of detention from which they have been deported out of Italy.

*** Stage 5 is not included in the family separation analysis since every convoy had a unique destination and thus no separation occurred during stage 5.

Since the types of separation examined are not all mutually exclusive, one separation event can lead to more than one type of separation and the FSIS of the victim can be multiplied by more than one coefficient for a single separation event. For instance, a mother being separated from her husband and two of her three children would see its FSIS multiplied by 0.8 (Victim separated from at least one other

Table 4: Family separation typology and FSI coefficients

Type of separation	Coefficient
Victim separated from at least one other member of their AFG but not all	0.8
Victim separated from all other members of their AFG	0.5
Child separated from his/her mother but not his/her father	0.5
Child separated from his/her father but not his/her mother	0.6
Child separated from both parents	0.3
Mother separated from at least one of her children, but not all	0.6
Mother separated from all of her children	0.5
Father separated from at least one of his children, but not all	0.6
Father separated from all of his children	0.5
Spouses separated from one another	0.7

member of their AFG but not all), 0.6 (Mother separated from at least one of her children, but not all) and 0.7 (Spouses separated from one another). In the case in which some of the stages do not apply to an individual, the FSIS of the individual for the irrelevant stages are simply reported from the previous stage.

In addition to assigning FSIS, the script also records the place at which the separation occurred (i.e. the last place where the separated members have been recorded together), as well as if one has been separated, if appropriate, from one's spouse, from one's mother, father or both, or from all or some of one's children, regardless of the stage of deportation during which this or these separation occurred.

It should be emphasized that the script has been carefully and iteratively designed to make sure that every separation case is rightfully identified and that the proper coefficients are applied. Moreover, before final validation, FSISs were calculated by hand for a random sample of 13 AFGs (10% of the 135 AFGs that were identified as separated) and compared to the output of the script.

In order to assess whether some victims were more vulnerable to family separation than others, this study proposes to research the effects of some demographic variables and some variables related to the arrest of the victims (the independent variables) on the FSIS of the victims (the dependent variable). To achieve this, two

analysis were performed. The first one aimed to analyze separation without any assessment of severity. For this analysis, the FSISs of the victim were converted into binary variables: 1 for the victims that were separated from other members of their family during or before the stage investigated, 0 for those who were not separated. The second analysis used the untransformed FSIS (1 for victims not separated, and the lower the score the more emotionally challenging the separation) with the objective of examining which categories of victim were more prone to severe separations. For both analyses, the independent variables were treated as dummy variables and are presented in Table 5.

Table 5: Independent variables

Variable	Possible value
x_1 Victim's nationality	Italian (0), Non-Italian (1)
x_2 Victim's gender	Female (0), Male (1)
x_3 Victim's age 1	Adult - over 18 (0), Children - under 18 (1)
x_4 Victim's age 2	Not elderly - under 70 (0), elderly - over 70 (1)
x_5 Perpetrators' nationality 1	Italian alone or with German (0), German alone (1)
x_6 Perpetrators' nationality 2	German alone or with Italian (0), Italian alone (1)
x_7 Period of arrest	1943 (0), 1944 - 1945 (1)
x_8 Victim's survival	Survived (0), Deceased (1)
x_9 Stage of deportation 1	Stage 1 (1), Not stage 1 (0)
x_{10} Stage of deportation 2	Stage 2 (1), Not stage 2 (0)
x_{11} Stage of deportation 3	Stage 3 (1), Not stage 3 (0)

The independent variables were chosen based on the theoretical framework presented in Chapter III. Moreover, the nationality of the victims, the nationality of the perpetrators and the period of arrest were shown to be of significance in the spatio-temporal patterns of arrest during the Holocaust in Italy (Giordano and Holian 2014). To determine if any correlation exists between separation during deportation and death in the camp, the victim's survival was also added as a variable. Based on the theoretical framework and the literature review previously introduced, the

null hypothesis that this study is testing is that none of those independent variables have an effect on the FSIS.

Before performing the analysis, the distribution of the FSIS was hardly predictable, especially since no previous study has researched family separation during the Holocaust. However, it was known that the dataset analyzed was longitudinal. Longitudinal data sets are comprised of repeated observations of an outcome and a set of covariates for each of many subjects, which is the case for types of information (FSIS) on the same subjects (Holocausts victims) at multiple points in time (deportation stage). The Generalized Estimating Equations (GEE) approach, an extension of the Generalized Linear Model (GLM), is recommended for those types of data, which breaks the assumption of independence among the dependent variable (Liang and Zeger 1986). Unlike GLMs, GEEs assume correlation of errors and dependent variable (indeed, the four FSIS of each victims at the four stages are correlated) and do not assume homogeneity of variance or normality of residuals.

For the first analysis of family separation, the dependent variable is binary. Hence the General Estimating Equation for Logistic Regression model was used to test the null hypothesis. After excluding all the victims in the database that don't belong to an AFG or for which at least one of the variable was missing, the number of victims on which the analysis could be performed was 2775 (and four times more FSIS). This number represents between 25% and 30% of all the Jews deported from Italy during the Holocaust, which permitted the production of results that should be meaningful and reliable.

For the second analysis, the one assessing the severity of family separation, the dependent variable is continuous between 0 and 1 (0 excluded). Thus the General Estimating Equation for Normal Regression model was used to test the hypothesis. Only the 479 victims that have been separated at some point during their deportation and for which all of the independent variables are known have been considered for that analysis.

Exploring separation due to arrest was slightly more complicated. Indeed, know-

ing what family ties were significant for the victim, and thus worth investigating in term of family separation, was impossible with the data at hand. The concept of AFG as meaningful family unit does not apply here, since an AFG regroup only the family members still together after arrest (so by definition, no AFG has been separated by arrest). Using the extended family as presented in Figure 5 is not satisfactory either, since one cannot assume how close kin members of a same family were (cousins, parents-in-law, grand-parents and grandchildren, or even parents and children over 18 years old) and thus whether being deported to different camps was effectively perceived as separation. Nonetheless, as mentioned before, the literature indicates quite expectedly that separations of spouses or separations of children under 18 from their parents, and vice versa, are traumatic events for the victims. Unfortunately, the database on which this research is based does not provide a list of children for the victims. For some of them, their children was identified using the first script, but there it was impossible to be certain that all of the children of one individual were identified or even present in the database. Therefore, in order not to focus on biased or partial data, this study did not investigate parents that were separated from their children. It is possible, however, to know the spouse of some of the victims, as well as the parents of many of the children deportees. For both of those categories of victims, SQL queries were used to identify whether or not the individuals for whom their spouse or both their parents were known were separated. This step could have been achieved using a python script, but SQL queries were chosen because a relational database with the data was available from a separate project and because the queries were very simple (simply comparing places and date of arrest for spouses or children and parents). It should be specified that in the case of children separated from their parents, the list includes children separated from both parents as well as children separated from only one of their parents, in order to have a number of case large enough to be used for statistical analysis. Finally, for both separations from spouse and separations from parents, a binary separation variable was added to each of the victims (1 if separation occurred, 0 if separation

did not occurred) and logistic regressions using the appropriate dependent variables from Table 5 were performed. Additionally, the lists reporting the cases of separations included places of arrest and places where many separation occurred were compared to the principal places of arrests.

Limitations of the proposed methods

The nature of the data induces some limitations that should be acknowledged. While about 10,000 Jews were deported from what was Italy in 1939, only 6,116 individuals, those for whom the place of arrest is known, are recorded in the database. Even though many of the remaining individuals were in fact excluded from the database due to the fact that they did not fit the scope of the analysis performed, for instance because they were not arrested in what is today Italy (about 1,800 victims), or because they died early during the deportation process (about 200 victims), there are approximately 1,500 Jewish victims of the Italian Holocaust for which no data were available. Moreover, some information is missing for some of the individuals recorded in the database, or might be incorrect or incomplete. The scripts cannot assign a family ID to missing individuals, or to individuals with missing information. Therefore, the analysis is probably slightly biased by the fact that some individuals may not have been included in their families. Overall, the number of families and the percentage of individuals arrested with their family is likely to be underestimated.

Additionally, it is worth noticing that the intended work is based on quantitative methods and includes all victims, and hence it might be difficult to see connections with the personal lived experiences of the victims. Future work might include the study of a subset of these families using qualitative methods, specifically those for which oral histories are available. The comparison of patterns of deportation found in this study with the victims' memories and testimonies will allow for a better understanding of the victim's perception of her/his own deportation.

VI. RESULTS

Patterns of family arrests

Before reporting the results of any analysis related to AFG characteristics, it should be noted that, because the family groups listing is a mere subset of the initial database, demographics of AFG were expected to be relatively similar to the corresponding demographics of the entirety of the 6,116 records of the original database. For similar reasons, spatial and temporal patterns of family arrest may be expected to be overall comparable to spatial and temporal patterns of individual arrest. Any major disparities would reveal either a special feature of family groups or a bias in the treatment of data and building of groups.

AFG characteristics and composition

From the 6,116 individuals in the initial database, 56.5% (3,456) were found to be arrested with other member of their family, forming 1,062 AFGs. Those AFGs were composed of 2 to 20 members, with an average of 3.25 members per AFGs and a standard deviation of 2.1. As presented in Figures 8 and 9, the age and gender distributions of AFG members are comparable to the age and gender distributions for the total number of victims. There is, however, a larger proportion of children among AFG members (22%) than among the overall victim population (14%). This fact is to be expected considering that children are likely to live with and frequently be around their parents, and thus be arrested with them. The similarity between statistics related to AFG members and to the total number of victims is also noticeable in Table 6, when analyzing perpetrator's nationality and victims' country of birth. As mentioned before, while the outcome of those comparisons were expected and do not provide any specific insights on the characteristics of AFG, it nevertheless suggests no strong bias has been introduced by grouping individuals into AFGs and that the followed methodology is likely to produce reliable results.

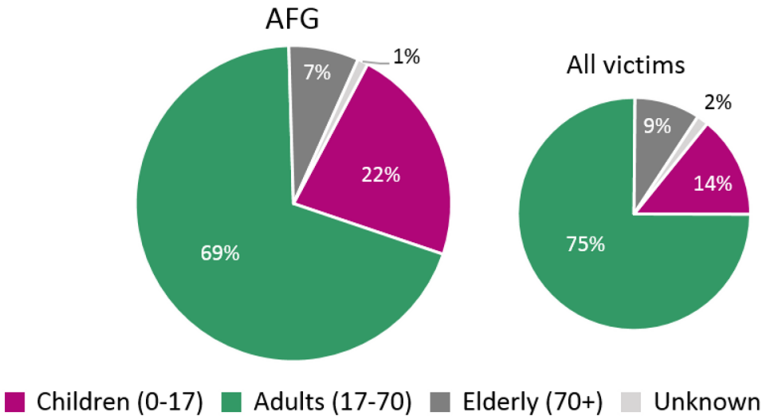

FIGURE 8: Age of victims belonging to an AFG compared to all victims

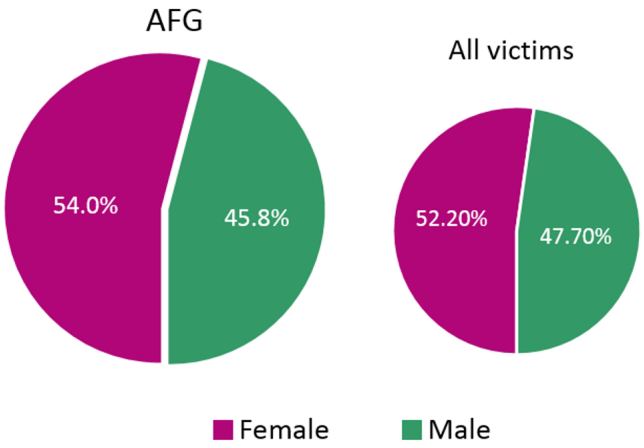

FIGURE 9: Gender of victims belonging to an AFG compared to all victims

Table 6: Comparison of AFG and overall perpetrators' nationality and victims' place of birth

	Number of AFGs	Percentage of AFG	Percentage of all victims
Nationality of perpetrators who arrested them			
- Italian	333	31.4%	29.2%
- German	426	40.1%	39.4%
- Italian with German	56	5.3%	5.2%
- Unknown	247	23.3%	26.2%
Total	1062	100%	100%
Victims place of birth			
- Italy	597	56.2%	63.8%
- Not Italy	336	31.6%	34.6%
- Mixed (possible only for AFG)	112	10.5%	N/A
- Unknown	17	1.6%	1.6%
Total	1062	100.0%	100%

Spatial Distribution of family group arrests

Unlike aggregate demographic statistics, spatial analysis does show differences between patterns of arrest at the family scale and those at the individual scale. Table 7 exhibits some of those differences, even though they are difficult to interpret. Indeed, divergent situations can lead to a same type of difference between the proportion of family groups arrested in one place over the total number of AFG and the proportion of the number of victims, members of an AFG or not, arrested in that same place over the total number of victims in the database. In Rome for instance, the former proportion, 23.4%, is lower than the latter, 27.7%, and this could be due either to a relatively low number of AFG member arrested in that city, or to an average size of family group arrested in that city bigger than the overall mean of AFG size (i.e. a large number of AFGs with more than 3 members). The equivalent statement is also true for higher than expected percentages of AFGs, such as in Borgo San Dalmazzo. In fact, in the light of the analysis of family vulnerability to round ups presented in a subsequent section that shows a high proportion of individual arrested with family members in those two cities, the latter explanation appears to be true for Rome while the former is appropriate for Borgo San Dalmazzo.

Mapping family arrests and analyzing AFG clusters (Figure 10) and comparing it to the patterns of arrests at the individual scale (Figure 11) allows one to see more clearly how spatial patterns of individual and family arrests differ. First, Figure 10 clearly shows that Rome, and to a lesser degree Trieste, Borgo San Dalmazzo, Florence, Milan, Venice, Civitella del Tronto, Turin and Fiume, concentrated a large number of family arrests. Regarding the spatial repartition of family arrests over the Italian territory, the overall location of the five clusters of arrest at the individual scale are preserved in the family scale analysis. The Rome cluster, the Florence-Bologna cluster and the Venice-Trieste cluster essentially represent arrests that happened at places of residence, most of them during round-ups. The Borgo San Dalmazzo cluster shed light on a quite different phenomenon. Indeed, many Jews arrested in that region were fleeing France. Finally, the cluster located in the

TABLE 7: Places of family arrests

	Number of AFGs	Percentage of AFG	Percentage of victims, in AFG or not
Place of arrest			
- Rome	248	23.4%	27.7%
- Triest	90	8.5%	9.0%
- Borgo San Dalmazzo	70	6.6%	5.4%
- Florence	46	4.3%	4.5%
- Milan	44	4.1%	4.4%
- Venice	29	2.7%	3.3%
- Civitella del Tronto	28	2.6%	2.5%
- Turin	28	2.6%	3.4%
- Fiume	22	2.1%	2.9%
- Genoa	18	1.7%	1.8%
- Other	431	41.3%	35.1%
Total	1062	100.0 %	

Milan region depicts a composite situation of arrests of Jews trying to escape the Holocaust by reaching Switzerland and arrests of Jews at their place of residence. The Rome cluster needs to be considered carefully as it is more a statistically artificial grouping of two cities where many arrests were recorded, Rome and Civitella del Tronto, with very few other arrest places than a spatially coherent cluster. While those observations are valid for both individual and family arrests, some of the characteristics of the clusters differs between the two scales of analysis. The Rome AFG cluster tends to be more centered over Rome than the individual cluster, and the Venice-Trieste cluster has is center closer to Trieste in the family arrests map than in the one mapping arrest at the individual scale. To some degree, the same observation can be made about the Turin-Borgo San Dalamzzo cluster, centered over Borgo San Dalmazzo, and the central cluster, centered over Florence. This indicates that for each of those regions regions—Rome, Trieste, Florence, and Borgio San Dalmazzo—are places where the proportion of family arrests were relatively larger than the proportion of individual arrests. This suggests that families might have been especially vulnerable to early round-ups in big cities such as the ones that occurred in Rome in October 1943 and in Trieste in November 1943. Additionally, the shape of the Milan cluster is more elongated in the family arrest map than in the map produced by Giordano and Holian (2014) and seems to follow the Italian-Switzerland border, which probably reflects a large number of families trying to escape the Holocaust by fleeing to Switzerland.

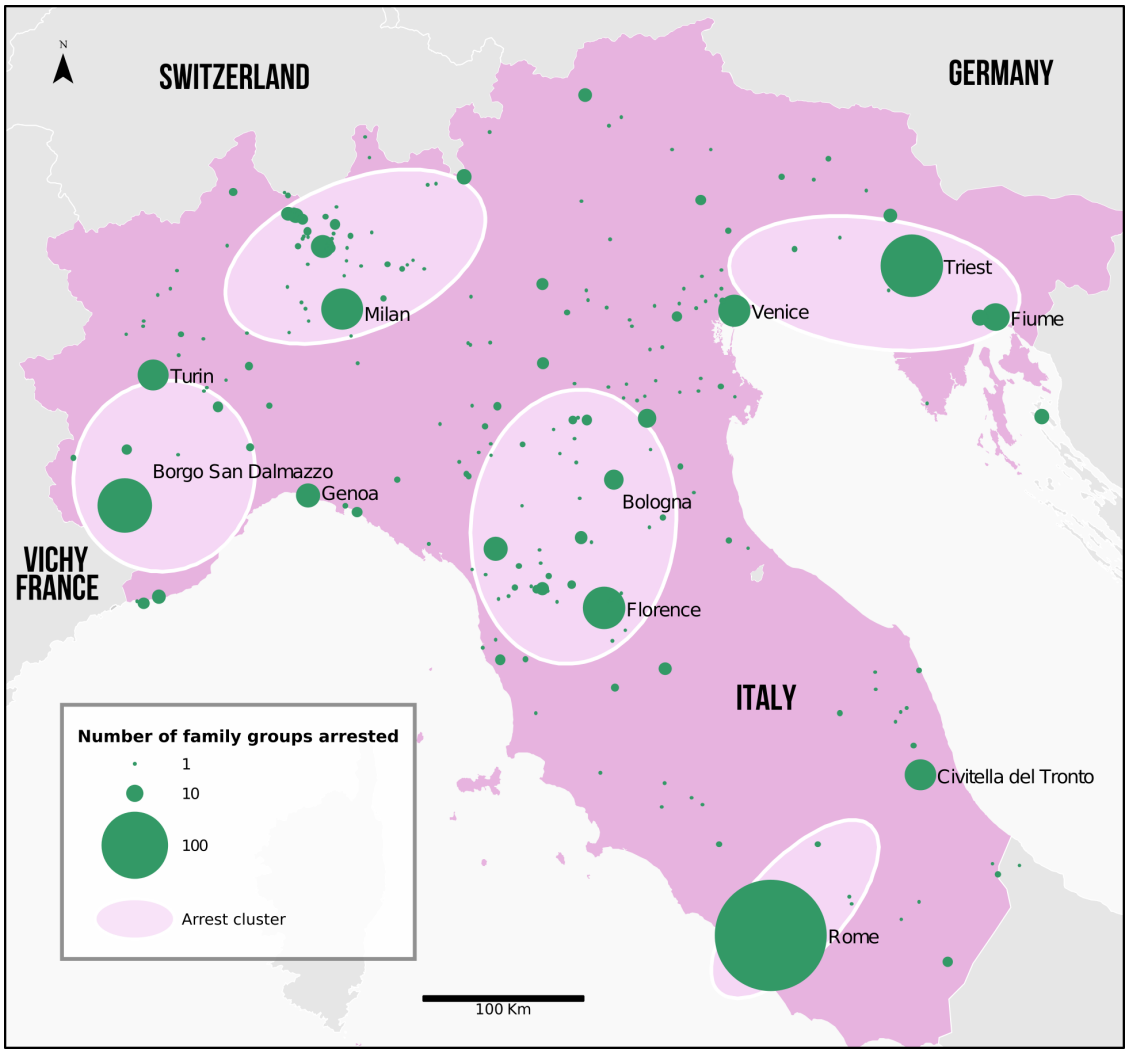

FIGURE 10: AFG arrest clusters

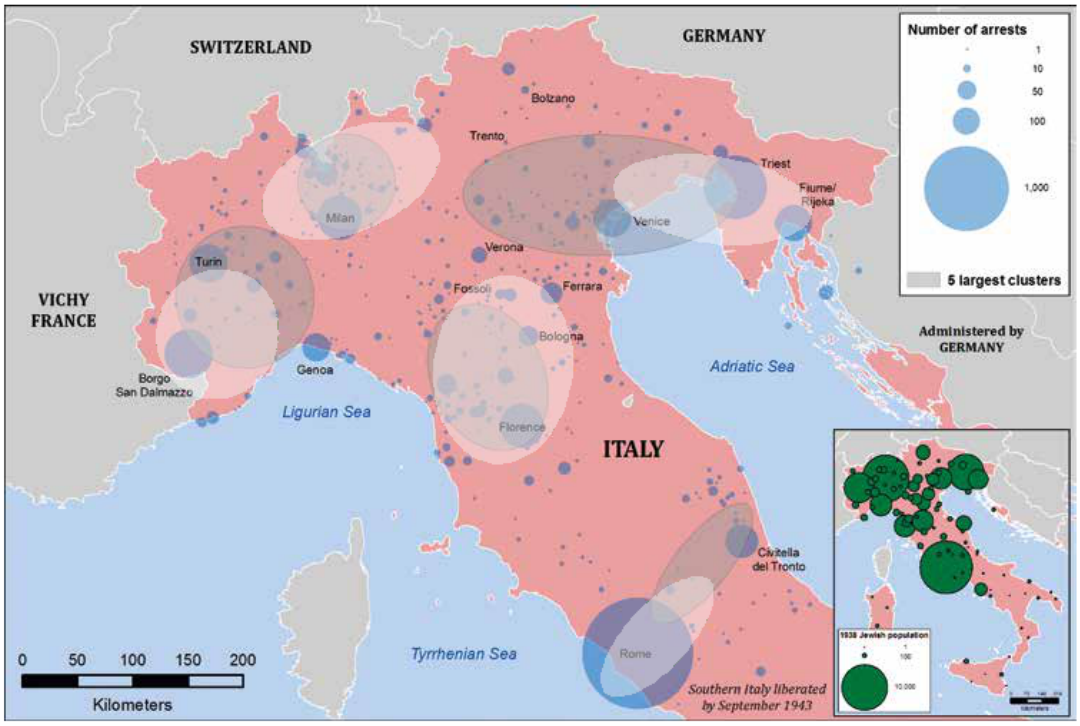

FIGURE 11: Overlay of AFG arrest clusters over individual arrest clusters
The AFG arrest clusters are the white ones

Spatio-temporal proximity of family group arrests

Figure 12 presents the spatio-temporal proximity of AFG arrests as measured by the Knox Index. Overall, almost 30% of all arrests occurred close in space (25km apart or less, which correspond more or less to an administrative region) and time (within a month). This number reaches 45% for some of the subsets. Arrests by Italian perpetrators, as well as arrests of mixed families to a lesser extent, tend to be concentrated in time, while arrests by German perpetrators, arrests of Italian families, and arrests of non-Italian families tend to be concentrated both in space and in time.

Figure 13 shows that the average time between AFG arrests is 3.3 months, and the average distance is 291 km. When broken down according to nationality of victims or perpetrators, average time and distance do not vary much, except for AFG arrested by German perpetrators which have average distance of 238 km and average time of 2 months, showing a greater spatio-temporal proximity of arrests than other group, as noticed previously.

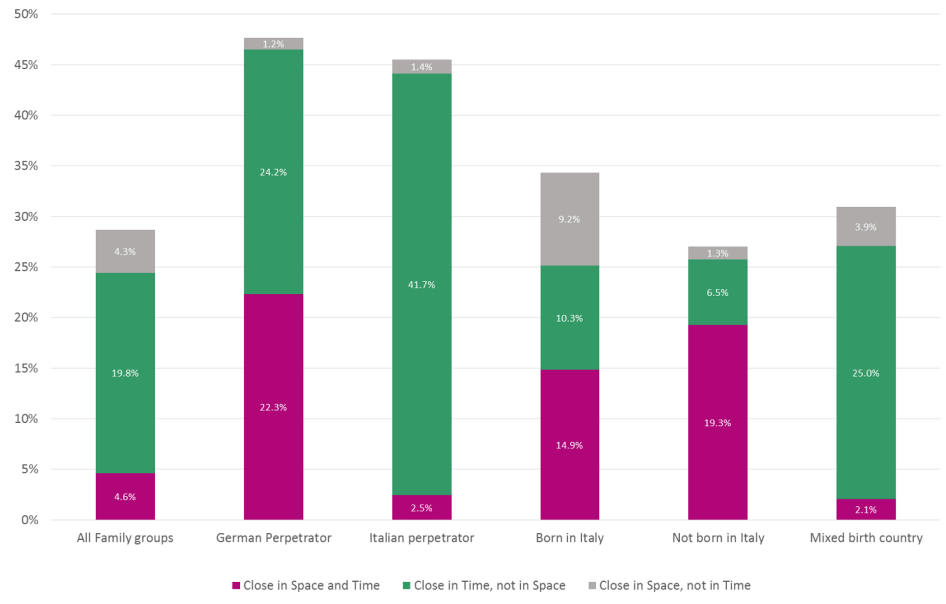

FIGURE 12: Spatio-temporal proximity of AFG as measured by the Knox Index

Before comparing the results of the above Knox index analysis with the results

FIGURE 13: Spatio-temporal proximity of AFG: Average distances

from Giordano and Holian (2014), it is important to detail the effects of family grouping on this type of analysis. As shown in Table 8, when data are grouped by family, the average distance is increased and the absolute value of the percentage of close events is decreased.

TABLE 8: Effect of family grouping on Knox Index results

	Knox Index for individuals belonging to an AFG	Knox Index when grouping by AFG
Average time between arrests	97.5 days	99 days
Average distance between arrests	286 km	291 km
Arrests close in time	26.9%	24.4%
Arrests close in distance	11.4%	8.9%

Arrests close in time defined as 30 days apart or less
Arrests close in distance defined as 25 km apart or less

Figure 13 shows that the average values of average spatial distance between arrests similar to Giordano and Holian's (2014) results (Figure 14) and average time is

about 10 fewer days. Because of the grouping effect discussed previously, this results reveals a slightly higher spatial proximity of family arrests than individual arrests, and a significantly higher temporal proximity. Nonetheless, the conclusions made by Giordano and Holian (2014), especially those regarding the specific concentration of arrest by German and Italian perpetrators, are corroborated.

Figure 15 confirms the previous observations. Overall spatio-temporal proximity, and specifically temporal proximity, of family arrest is clearly higher than the one of individual arrests, regardless of the subgroup. The particularly low spatial proximity of arrests by Italian illustrate their action over a large territory, while the high spatio-temporal proximity of arrest by Germans illustrates their role in the round-ups that took place in big city in September and October 1943, and particularly the one in Rome. Lastly, an interesting result is the high spatio-temporal proximity of arrest of non-Italian families. This pattern may be due to arrests of families trying to flee the persecutions in France.

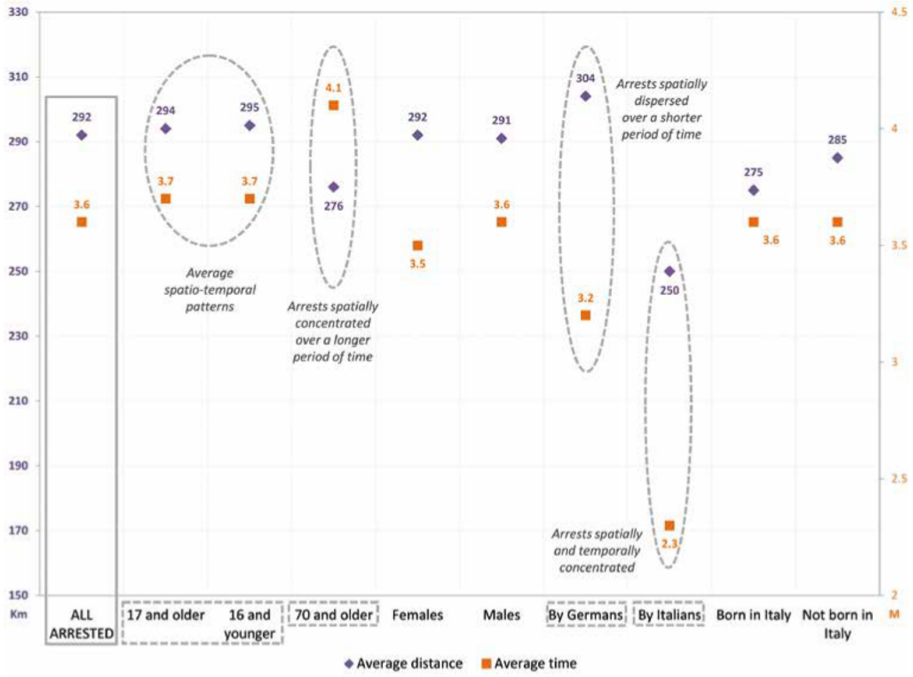

FIGURE 14: "Spatio-temporal proximity as measured by the Knox index" (Giordano and Holian 2014)

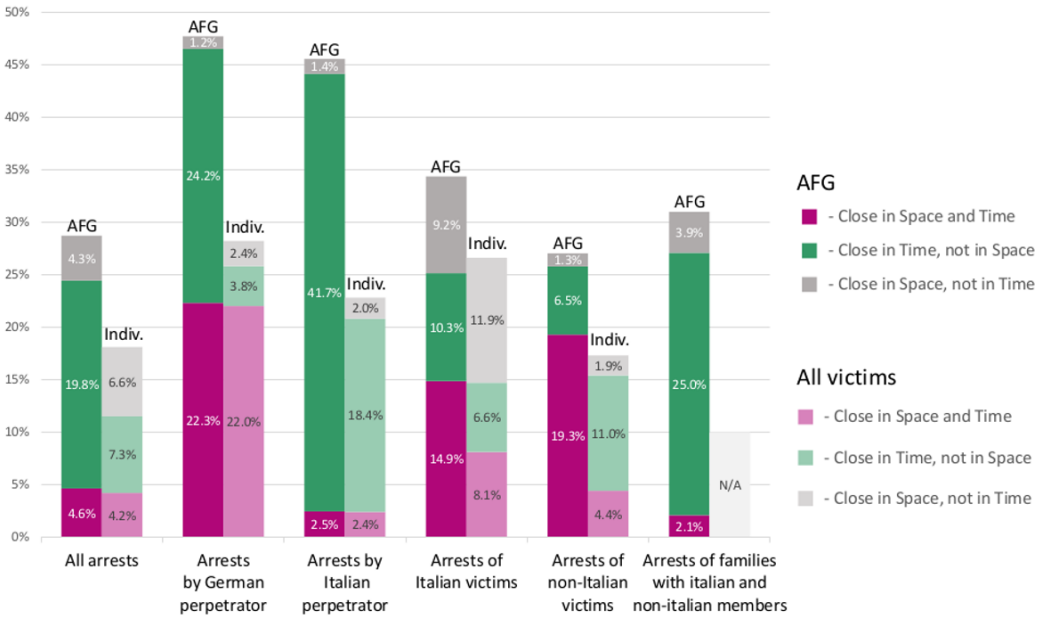

FIGURE 15: Spatio-temporal proximity: comparison with Giordano and Holian's (2014) results

Family vulnerability to round up

Both the the spatio-temporal proximity analysis and the cluster analysis suggested that family might have been arrested in high proportion during round-ups. In order to investigate this hypothesis, the percentage of individuals arrested with other members of their family during round ups in Rome, Florence, Venice and Borgo San Dalmazzo was calculated and compared to the overall percentage of individuals arrested with family members during the Holocaust in Italy (Figure 16). As expected, a higher proportion of Jews were arrested with family members during round ups in major cities than were the proportion of Jews arrested with family members between 1943 and 1945 over the entire Italian territory. This seems to indicate that families were more vulnerable to round-ups. This statement is reasonable if one considers that it is logistically easier to try escape alone than to plan a journey with other family members, in particular with children.

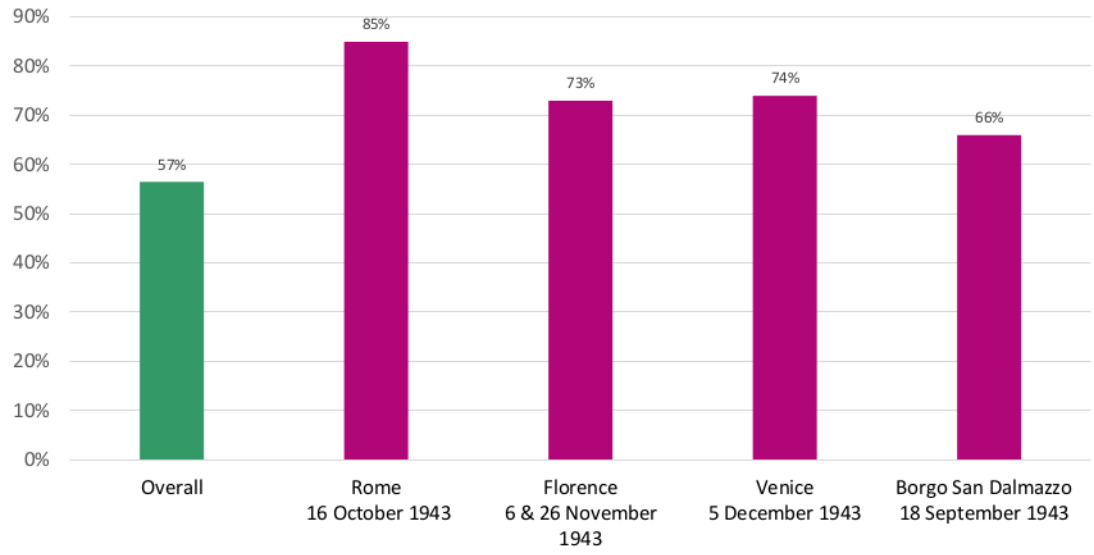

FIGURE 16: Percentage of individuals arrested with at least one member of their family during round-ups

Family separation

The family separation script recorded cases of separation for 16% of the AFGs (135 AFGs separated over 836 AFGs inspected) affecting 479 persons (17% of the 2775 Jews investigated). This alone establishes that, while not extremely frequent, family separation during deportation was far from uncommon and affected slightly more than 1 in 6 deportees.

The script also returned information about separation between spouses, and between parents and children. Of the 840 victims that were known to have been arrested with their spouse, 104 (13 %) were separated during deportation. Regarding the separation of parents from their child, 89% (204 fathers and 275 mothers) of the 538 parents that were arrested with their child remained together during the deportation process, 4% (8 men and 14 women) were separated from some of their children but not all, and 7 % (15 fathers and 21 mothers) were separated from all of their children. Finally, of the 645 children that were arrested with at least one of their parents, 89 % were not separated from them, 5% were separated from their mother (and remained with their father, or their father was not in the database), 2% were separated from their father, and 4% were separated from both parents. Overall, we noticed that parental and marital separation occurred slightly less often than other types of familial separation.

Probability of separation during deportation

The results of the GEE logistic regression, presented in Table 9, provide additional insights on family separation. First, those results show that, keeping all other variables constant, the odds of being separated for victims not born in Italy were 0.75 times the odds of being separated for victims born in Italy, meaning that Italians were more likely to be separated from members of their family than non-Italians. Similarly, odds of being separated were 33% higher for children than for adults, while being over 70 years old does not appear to have any effect on the likelihood of being separated. Gender does not appear to have correlation with separation

probability either, which is logical if one considers that separations often occurred between gender (for instance, only women were sent to Ravensbruck) which means that individuals from both genders would have been separated from each other.

Perhaps more significantly, odds of being separated for Jews arrested in 1944 or in 1945 were almost 150% higher than for Jews arrested in 1943. This observation seems consistent with the two-stage interpretation of the Italian Holocaust describing arrests of large number of Jews during massive round-ups in 1943 and smaller numbers of Jews were arrested in various locations in 1944 and 1945. Jews who were arrested in the 1943 round-ups were almost immediately sent to extermination camp by convoys, while in the following years, the time and number of detention places between arrest and deportation by convoy was long enough for the perpetrators to separate the victims according to various criteria, including whether they were supposed to be sent to work camp or death camp, and thus split families during the process.

Also of interest, the odds of being separated for Jews arrested by Germans alone were 0.26 times the odds for Jews arrested by Italians alone or Italians with Germans. Conversely, it means that Jews arrested by Italians, alongside with Germans or not, were way more likely to be separated from their family than Jews arrested by Germans alone. Although being arrested by Italians and having been separated does not automatically translate to being separated by Italians, since the data do not provide any information about who was in charge of the deportees at the time of separation, this clearly questions once again the myth of Italians as "brava gente."

Finally, there is a correlation between victims that have been separated and victims that survived the Holocaust. The odds of being separated for Jews who were liberated were 30% higher than the odds for Jews who did not survive. This suggests that few families survived entirely, perhaps because the likelihood of having all of the members of a family that were meeting the criteria not to be sent to death camp, and thus had better chances to survive, was relatively low.

In terms of stages of arrest, of course the later the stage, the higher the odds

of having been separated, but one can also notice that those odds grow quickly between stage 1 and stage 2, and increase rather slowly after stage 2, suggesting that a high proportion of separations occurred early in the deportation process, right after arrest of after the first place of detention.

TABLE 9: GEE logistic regression results

	Estimate	Standard Error	Statistical Significance
Intercept	-0.95	0.2596	***
Victim's Nationality	-0.30	0.1172	***
Victim's Gender	0.12	0.1095	
Children	0.29	0.1213	**
Elderly	-0.34	0.2641	
Perpetrator's Nationality 1	-1.34	0.2163	***
Perpetrator's Nationality 2	-0.19	0.2074	
Year of Arrest	0.90	0.1151	***
Fate	-0.26	0.1454	*
Stage 1	-0.89	0.0528	***
Stage 2	-0.32	0.0304	***
Stage 3	-0.28	0.0286	***

Note : * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$

Dependent variable : Separated = 1

Number of clusters (ie number of victims) : 2775

Severity of separation during deportation

The second GEE regression aimed to assess the severity of separation and its results are presented in Table 10. This analysis reveals that the severity of the separation is uncorrelated with the stage of deportation, the period of the Holocaust, the nationality of the perpetrators, the gender of the victims, or their fate. In fact, only the nationality of the victims seems to have an effect on the severity of their separation. Jews not born in Italy were likely to experience more severe separations than Italian Jews. This contrasts with the fact that non-Italian Jews were less likely separated from their family than Italian Jews. Overall, relatively few non-

Italian Jews were separated from their family but those who were separated had more extreme cases of separation, in terms psychological and emotional hardship. Additionally, children seemed to have been more likely to go through more severe separations than adults, but this is probably due to the fact that separation from parents, lived only by children, are already considered as the most difficult types of separation and coded with low coefficients.

TABLE 10: GEE normal regression results

	Estimate	Standard Error	Statistical Significance
Intercept	0.59	0.044	***
Victim's Nationality	-0.06	0.025	**
Victim's Gender	0.01	0.020	
Children	-0.04	0.024	*
Elderly	-0.04	0.040	
Perpetrator's Nationality 1	0.03	0.039	
Perpetrator's Nationality 2	-0.03	0.038	
Year of Arrest	0.02	0.022	
Fate	-0.02	0.027	
Stage 1	0.01	0.009	
Stage 2	0.01	0.005	
Stage 3	0.01	0.005	

Note : * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$

Dependent variable : Not separated = 1

Number of clusters (i.e. number of victims) : 479

Places of separation during deportation

Tables 11, 12 and 13 all display, for some places of detention, the percentage of victims that have been detained in the place (based on the 2775 victims in the dataset that belong to an AFG and for which all variables were known), the percentage of separation that occurred at the place (based on the 513 separations for which the place has been identified), and the separation rate of the place (i.e. the ratio between the number of persons separated at the location and the number of person

that transited through the location).

Table 11 displays only the main places of detention, those where more than 3% of all the deportees transited. All of those places, except Civitella del Tronto, have a lower separation rate than the overall 17% identified earlier (479 victims separated over the 2775 victims for which all the data were available), and ten of those 14 locations have a separation rate equal to or lower than 5%. This shows that the main prisons and camps in Italy were not places where separation systematically occurred, or major “sorting" centers as one might have expected.

To the contrary, Table 12 shows that, even though some of those principal places of detention are also places were many separation occurred simply because of the volume of Jews that transited through those camps and prisons, an important number of separation took place in locations through which a moderate number of victims transited. For instance, in Camugnano campo, in Chianni, and in Savigno, three medium-size places of detention, all the victims were separated from at least one family member. Similarly, in Bagni di Lucca campo, in Montecatini, and in Genoa, the average 17% separation rate was exceeded.

Finally, Table 13 lists the places of detention with a particularly high separation rate (more than 60%). Even though for most of those locations, the relevance of those numbers might be questioned due to the particularly low number of victims recorded, it nevertheless shows that even when only one family, or a very small number, was being deported by a group of perpetrators, they were not safe from being separated. In fact, it was even quite the opposite.

Overall, it is interesting to notice that victims seem to have been safer from family separation in locations that they shared with many other victims and families, perhaps because the number was difficult in itself for the perpetrator and they did not have time to focus on individual cases, perhaps because the number could offer some anonymity to the victims. To the contrary, when perpetrators were in charge of only a few Jews, they might have had more time to apply carefully the orders stating that some of the victims were supposed to end up in specific working

camps while other were to be sent to death camps, leading to separation of family members.

TABLE 11: Family separation in the main places of detention

Place of detention	Victims separated	Percentage of victims separated*	Victims detained	Percentage of victims detained**	Separation rate
Rome	67	13.06%	1806	65.08%	4%
Fossoli campo	10	2.0%	863	31.0%	1%
Rome collegio militare	30	5.9%	862	31.0%	3%
Fossoli	41	8.0%	823	29.7%	5%
Trieste	40	7.8%	445	16.0%	9%
Milan prison	7	1.4%	441	15.9%	2%
Borgo San Dalmazzo	5	1.0%	436	15.7%	1%
Milan	5	1.0%	400	14.4%	1%
Verona	15	2.9%	227	8.2%	7%
Florence	15	2.9%	141	5.1%	11%
Trieste prison	2	0.4%	139	5.0%	1%
Civitella del Tronto campo	22	4.3%	121	4.4%	18%
Venice prison	4	0.8%	98	3.5%	4%
Venice	4	0.8%	92	3.3%	4%

* Based on the 2775 victims that belong to an AFG and for which all variables were known

** Based on the 513 separations for which the place has been identified

TABLE 12: Places where most separation occurred

Place of detention	Victims separated	Percentage of victims separated*	Victims detained	Percentage of victims detained**	Separation rate
Rome	67	13.1%	1806	65.1%	4%
Fossoli	41	8.0%	823	29.7%	5%
Trieste	40	7.8%	445	16.0%	9%
Camugnano campo	31	6.0%	31	1.1%	100%
Rome collegio militare	30	5.8%	862	31.1%	3%
Civitella del Tronto campo	22	4.3%	121	4.4%	18%
Bagni di Lucca campo	19	3.7%	69	2.5%	28%
Verona	15	2.9%	227	8.2%	7%
Florence	15	2.9%	141	5.1%	11%
Montecatini	12	2.3%	20	0.7%	60%
Fossoli campo	10	1.9%	863	31.1%	1%
Chianni	10	1.9%	10	0.4%	100%
Milan	7	1.4%	441	15.9%	2%
Genoa	7	1.4%	23	0.9%	30%
Savigno	7	1.4%	7	0.3%	100%

* Based on the 2775 victims that belong to an AFG and for which all variables were known

** Based on the 513 separations for which the place has been identified

TABLE 13: Places with high separation rates

Place of detention	Victims separated	Percentage of victims separated*	Victims detained	Percentage of victims detained**	Separation rate
Camugnano campo	31	6.0%	31	1.1%	100%
Chianni	10	1.9%	10	0.4%	100%
Savigno	7	1.4%	7	0.3%	100%
Tizzano Val Parma	6	1.2%	6	0.2%	100%
Carsoli	5	1.0%	5	0.2%	100%
Chiavenna	5	1.0%	5	0.2%	100%
Issime	4	0.8%	4	0.1%	100%
Florence province	4	0.8%	4	0.1%	100%
Asti campo	3	0.6%	3	0.1%	100%
Casteldelfino	3	0.6%	3	0.1%	100%
Nembro	3	0.6%	3	0.1%	100%
Rapallo	3	0.6%	3	0.1%	100%
Succinto Canavese	3	0.6%	3	0.1%	100%
Torre Boldone	3	0.6%	3	0.1%	100%
Canischio	2	0.4%	2	0.1%	100%
Carpi	2	0.4%	2	0.1%	100%
Casola Valsenio	2	0.4%	2	0.1%	100%
Casoli di Camaiore	2	0.4%	2	0.1%	100%
L'Aquila prison	2	0.4%	2	0.1%	100%
Pescia	2	0.4%	2	0.1%	100%
Traversetolo	2	0.4%	2	0.1%	100%
Torino campo	2	0.4%	3	0.1%	67%
Montecatini	12	2.3%	20	0.7%	60%
Alessandria prison	3	0.6%	5	0.2%	60%

* Based on the 2775 victims that belong to an AFG and for which all variables were known

** Based on the 513 separations for which the place has been identified

Separation during arrest

Before analyzing separation due to arrest, it should be noted that perpetrators were likely to have less control on whether to separate a family or not during arrest. Of course they could have chosen to arrest only some members of one family, but in many cases they were only able to arrest the members that were present at the place of arrest; the separation being then due mainly to the fact that all of the members of the family were not physically at the same place during arrest, perhaps because some of them were at work, at school, or in any other place. This remark is not aimed to minimize the responsibility of the perpetrator of an arrest, since they always had the possibility not to arrest their victims, but to point out that different conditions applied during arrest and during deportation.

Regarding spousal separation during arrest, 28% of the 1051 victims for which the spouse and all other information necessary for the logistic regression was known were separated from their husband or wife. Table 14 presents the results of that analysis. Similar to separations during deportation, separations of spouses during arrest were more likely in 1944-1945 than in 1943. Such separation were also more likely among Italian families than among non-Italian families. That might be due to a high number of non-Italian families, particularly families coming from France, arrested together at the border when trying to flee the Holocaust in their country, while members of many Italian families might have been arrested separately in their place of work, residence and other. Finally, victims arrested by Italians, with or without Germans, were once again more likely to be separated than their counterpart arrested by Germans alone. As specified earlier, it would be unwise to interpret that as a deliberate practice from the Italian police, but it does suggest that keeping family together was not a priority for them, which challenges again the traditional image of Italian as "brava gente."

The results of the logistic analysis of separations of children from their parents are presented in Table 15, and they are very similar to those for spousal separations. The only major difference is that Italians, with or without Germans, did not separate

children more or less, statistically speaking, than Germans alone. Additionally, the proportion of children separated from one or both of their parents by arrest is 21% a number slightly lower than for spouse separation, which might indicate either that parents tended to be more unwilling to leave their child when trying to escape the Holocaust than to leave their spouse, or that perpetrators were somewhat reluctant to arrest children without their parents.

Tables 16 and 17 present the cities where most of the separation by arrest occurred (only the cities where more than 3% of the separation occurred for each case are reported). The pattern displayed here is comparable to the one regarding the main places of arrest presented in Table 2. One can, however, notice the high number of separations in the Italian-Swiss border, which suggests that families trying to escape to Switzerland were probably not traveling with all of the members of their family. One can also notice the relatively low number of separations at Trieste, which is consistent with the fact that Germans tended not to separate families as much as Italians, as well as the low number of separations at Borgo San Dalmazzo (less than 3%) compared to the relatively high number of arrest (5.4% of all arrest occurred in Borgo San Dalmazzo). Knowing that many families coming from France were arrested there, this is consistent with the previously stated hypotheses stating that many of the families that successfully crossed the French-Italian border did not leave members of their family behind.

Finally, separations during round-ups have been investigated, but the results were inconclusive, mainly because the number of cases available for analysis was too small.

TABLE 14: Spouses separation logistic regression results

	Estimate	Standard Error	Statistical Significance
Intercept	-0.91	0.435	**
Victim's Nationality	-0.72	0.199	***
Victim's Gender	-0.01	0.184	
Elderly	-0.38	0.357	
Perpetrator's Nationality 1	-1.35	0.328	***
Perpetrator's Nationality 2	-0.23	0.318	
Year of Arrest	0.88	0.190	***
Fate	-0.01	0.302	
Number of cases	2775		

Note : $^{*}p < 0.1$, $^{**}p < 0.05$, $^{***}p < 0.01$

Dependent variable: Separated = 1

Number of cases (i.e. number of victims): 1051

TABLE 15: Children separation from parents logistic regression results

	Estimate	Standard Error	Statistical Significance
Intercept	-1.70	0.706	**
Victim's Nationality	-0.99	0.304	***
Victim's Gender	0.02	0.259	
Perpetrator's Nationality 1	-0.38	0.598	
Perpetrator's Nationality 2	0.57	0.609	
Year of Arrest	1.52	0.304	***
Fate	0.35	0.38	

Note : $^{*}p < 0.1$, $^{**}p < 0.05$, $^{***}p < 0.01$

Dependent variable: Separated = 1

Number of cases (i.e. number of victims): 422

Table 16: Principal cities where spouses were separated by arrest

City	Number of cases	Percentage
Rome	31	20%
Venice	10	6%
Italian-Swiss border	9	6%
Ferrara	8	5%
Torino	8	5%
Florence	6	4%
Milan	5	3%
Cassano d'Adda	4	3%
Cremenaga	4	3%
Fiume	4	3%
Trieste	4	3%

Table 17: Principal cities where children were separated from their parents during arrest

City	Number of cases	Percentage
Rome	40	44%
Venice	8	9%
Civitella del Tronto	5	6%
Rieti	4	4%
Taglio di Po	4	4%
Cremenaga	3	3%
Italian-Swiss border	3	3%
Torino	3	3%

Overall, the various GIS and statistical analyses performed in this chapter proved that victimization of families during the Holocaust in Italy was different from victimization of individuals and highlighted specific patterns of family arrests and separation according to multiple factors. While this illustrates the fact that Jews may have experienced the deportation differently according to their nationality, the nationality of their perpetrators, or their time and place of arrest, explanations for those various patterns of victimization could not always be identified with certainty. Additional research using different sources would be needed in order to gain a deeper understanding of the Italian Holocaust.

VII. CONCLUSIONS

First and foremost, this study demonstrates that it is possible, and valuable, to study the Holocaust at the scale of the family. Not only did it permit an analysis of family separation, a previously unexplored phenomena, but it also provided results and insights that were sometimes different from those obtained from studies performed at the individual scale. Such methods using family as a unit could be reproduced in different contexts, particularly in genocide and forced migration studies.

Second, this research shows the value of geography for Holocaust and genocide studies, or any field of study that might be concerned with social networks. Indeed, GIS and spatio-temporal analysis were fundamental in uncovering the vulnerability of families to round-ups. Similarly, placed-based statistics was one of the methods that provided very insightful results when exploring family separation.

In terms of results, I found that more than half of the Jewish victims of the Holocaust in Italy were arrested with other members of their family, but also that families were often separated, either by arrest or during deportation.

Moreover, I was able to determine that families were particularly vulnerable to large-scale round ups, such as the one that occurred in Rome in October 1943, probably because fleeing with an entire family on short notice, especially with children, is rather complicated. Nevertheless, data suggests that a non negligible number of families were arrested at the border between Italy and Switzerland, probably while trying to escape arrests and persecutions. I discovered that many of those Italian families trying to escape were not complete nuclear families, which illustrates the fact that some victims of the Holocaust made the choice to flee even if that meant leaving part of their family behind, perhaps with the hope that they could be reunited later on, and that some families were separated during their flight. Quite interestingly, I also found that Jews trying to escape from the Vichy regime in France arrived, and were arrested, in Italy with their entire nuclear family.

Regarding family separations, this study showed that Italian Jews and victims arrested in 1944 or 1945 were more likely to be separated, either during arrest or during deportation, than non-Italian Jews or victims arrested in 1943. Children were also more likely to be separated from family members than adults during deportation, while gender did not seem to have a particular effect on separation likelihood. Interestingly, gender was not a significant variable for the study performed by Giordano and Holian (2014) either.

This analysis also establishes that victims arrested by Italians, alone or with Germans, were more likely to be separated from family members, both during deportation and in some cases during arrest, than victims arrested by Germans alone. Although this piece of information cannot be interpreted as a deliberate attempt of Italians to separate family, it nevertheless questions once again the already challenged "italiani brava gente" myth.

Additionally, this study suggests that family separation tended to be more likely in locations where a moderate or a small number of victims were handled by the perpetrators, in opposition to large detention centers and large-scale round ups where the likelihood of being separated from family members was lower.

Finally, as mentioned in the section discussing the limitation of the methods used, this study is merely quantitative. It allowed me to uncover specific phenomena, expose some of the perpetrators' practices, and glimpse the diverse experiences of deportation lived by Jews deported from Italy, but it does not take into account the narratives and the perspectives of the victims. It that sense, this research could later be completed using deportees' testimonies through a more qualitative approach.

APPENDIX SECTION

APPENDIX A
Assigning family IDs python script

```
#*******************************************************************************
# python 2.6.5 program to convert csv data concerning people and their relations,
    build a social network table, and assign an family Id number to individuals.
# based on a script by Ryan Schuermann - rs1571@txstate.edu - May-Dec 2012
# modified by Mael Le Noc - mael.lenoc@txstate.edu - Spring 2015
#
# !!!!!!!! WARNING !!!!!!!
# Your CSV files that you Save As from Excel or generate from some other source
# MUST BE IN THE EXACT FORMAT AS BELOW. I have provided an example of first line
# and first record (1 to 2 lines) for both files.
#
#Input file 1: personal information : Saved As CSV from Excel : (data\
    social_network_data.csv)
#ID,LAST NAME,FIRST NAME,BIRTHDAY,FATHER,MOTHER,SPOUSE
#I109,ALTMANN,FERDINANDO,1904.10.05,GUGLIELMO,SCHMIER GISELLA,HERSKOVITZ MARGHERITA
     (dep.)
#
# program details:
#
# creates 2D arrays concerning correlations between people. Array is [# people] by
    [# people]
# creates family ID number for each family (siblings, parents, children, parents'
    siblings,, parents' parents etc), and assign the appropriate family ID to each
    individual in a table.
#
#*******************************************************************************
import os, datetime, arcpy
#from operator import itemgetter
from numpy import *

infile1 = open(arcpy.GetParameterAsText(0),"r").readlines() # data input csv
outdir = arcpy.GetParameterAsText(1) # output folder/dir
outdir2 = arcpy.GetParameterAsText(3) # output folder/dir

#-------------------------------------------------------------------------------
# process output file name: ensure they will be txt files
#-------------------------------------------------------------------------------
rfile = root = ext = ""
rfile = arcpy.GetParameterAsText(2) # output filename txt
if ("." in rfile):
    (root,ext) = rfile.split(".")
else:
    root = rfile; ext = ""
if (ext != "txt"):
    rfile = root + ".txt"
outfile1= open(os.path.join(outdir,rfile),"w")
rfile2 = root2 = ext2 = ""
rfile2 = arcpy.GetParameterAsText(4) # output filename txt
if ("." in rfile):
    (root2,ext2) = rfile2.split(".")
else:
    root2 = rfile2; ext2 = ""
if (ext2 != "txt"):
    rfile2 = root2 + ".txt"
outfile2= open(os.path.join(outdir2,rfile2),"w")
del rfile; del root; del ext
del rfile2; del root2; del ext2
#-------------------------------------------------------------------------------
# declair variables
#-------------------------------------------------------------------------------
meValue = 122# self
siValue = 1 # sibling
faValue = 2 # father
```

```
moValue = 3 # mother
spValue = 4 # spouse
chValue = 5 # child
gpValue = 6 # grandparent
gcValue = 7 # grandchild
auValue = 8 # aunt/uncle
nnValue = 9 # neice/nephew
coValue = 10# cousin
# assign different relationship values for blood vs married aunts/uncles?
# grandparent's siblings?
# second, third cousins??

timeSuffix = datetime.datetime.now().strftime("_%Y_%m_%d__%H_%M_%S")
arcpy.env.overwriteOutput = True
idArray = []
oidArray = []
num_records = len(infile1) - 1 # -1 to account for header
relArray = zeros((num_records,num_records))
nineArray = zeros((num_records,1))
nineArray.fill(9999999999)
famArray = range(num_records)
famIDArray = range(num_records)
for y in range(num_records):
    famIDArray[y] = nineArray[y][0]
    famArray[y] = nineArray[y][0]
outstring = "Initialized 2D array of %d records..." % num_records
arcpy.AddMessage(outstring)
del outstring;
#-------------------------------------------------------------------------------
# build dictionary of IDs name, birthday, mother and father's name for quick data
    retrieval, and makes it easier to read
#-------------------------------------------------------------------------------
pInfo = {}
count = 0
for line in infile1:
    w = line.split(",")
    try:
        w[6] = w[6].rstrip("\n")
        w[6] = w[6].rstrip(" ")
    except:
        w[6] = w[6]
    #rough check to ensure we are on a personal line and not the header
    if not('ID' in w[0]):
        count += 1
        # build oid array forprocessing
        oidArray.append(w[0])
        #ensure ID is length 5, pad with zeros after the 'I' and build array, this
    is for output formatting
        newID = w[0]
        addZeros = 5 - len(w[0])
        if (addZeros > 0):
            newID = w[0][0]
            for x in range (0,addZeros):
                newID += "0"
            newID += w[0][(0-(len(w[0])-1)):]
        idArray.append(newID)
        del addZeros;
        #store: [id:name,id:bday,id:father,id:mother,id:spouse,id:spouse_dep]
        pInfo[w[0]+"name"] = w[1] + " " + w[2]
        if (('nessuna' in w[3].lower()) or ('sconosciuto' in w[3].lower()) or ('
    infante' in w[3].lower())):
            pInfo[w[0]+"bday"] = "0"
        else:
            pInfo[w[0]+"bday"] = w[3]
        if ('nessuna' in w[4].lower()):
            pInfo[w[0]+"father"] = ""
        else:
            pInfo[w[0]+"father"] = w[1] + " " + w[4]
        # mother has maiden and first name
        if ('nessuna' in w[5].lower()):
            pInfo[w[0]+"mother"] = ""
        else:
            pInfo[w[0]+"mother"] = w[5]
```

```
        # spouse has last and first name, plus deport text ...or one of three
    messages equalling no spouse/unknown
        if (('17' in w[6]) or ('nessuna' in w[6].lower()) or ('coniugato' in w[6].
    lower())):
            pInfo[w[0]+"spouse"] = ""
            pInfo[w[0]+"spouse_dep"] = ""
        else:
            if ('(dep.)' in w[6]):
                pInfo[w[0]+"spouse_dep"] = "Yes"
                try:
                    w[6] = w[6].replace("(dep.)","")
                    w[6] = w[6].rstrip(" ")
                except:
                    w[6] = w[6]
            else:
                pInfo[w[0]+"spouse_dep"] = "No"
            pInfo[w[0]+"spouse"] = w[6]
        # DEBUG
        #arcpy.AddMessage(line)
        #outstring = "%d)%s name[%s] bday[%s] father[%s] mother[%s] spouse[%s] dep
    [%s]\n" % (count,w[0],pInfo[w[0]+"name"],pInfo[w[0]+"bday"],pInfo[w[0]+"father"
    ],pInfo[w[0]+"mother"],pInfo[w[0]+"spouse"],pInfo[w[0]+"spouse_dep"])
        #arcpy.AddMessage(outstring)

del line;  del w
outstring = "Done with building personal dictionary of %d records..." % count
arcpy.AddMessage(outstring)
del outstring
#-------------------------------------------------------------------------------
# loop through records and determine matches
#-------------------------------------------------------------------------------
rowIndex = 0
for rows in infile1:
    row = rows.split(",")
    # ensure we are not on the header row and proceed
    if not('ID' in row[0]):
        # build the first row in the output file
        outfile1.write(idArray[rowIndex]+",")
        # loop through file again to look for matches
        potFather = (-1,-1,"","","")
        potMother = (-1,-1,"","","")
        potSpouse = (-1,-1,"","","")
        colIndex = 0
        for cols in infile1:
            col = cols.split(",")
            if not('ID' in col[0]):
                #-------------------------------------------------------------
                # found self/self
                #-------------------------------------------------------------
                if (row[0] == col[0]):
                    #on same record, do nothing
                    relArray[rowIndex][colIndex] = meValue
                    # DEBUG
                    #outstring = "Found Self: [%d] [%d]" % (rowIndex,colIndex)
                    #arcpy.AddMessage(outstring)
                    #del outstring
                #-------------------------------------------------------------
                # SIBLING: same last name, father, mother
                #-------------------------------------------------------------
                elif (row[1] == col[1]) and (pInfo[row[0]+"father"] == pInfo[col
    [0]+"father"]) and (pInfo[row[0]+"mother"] == pInfo[col[0]+"mother"]):
                    # found sibling
                    relArray[rowIndex][colIndex] = siValue
                    relArray[colIndex][rowIndex] = siValue
                    if famArray[rowIndex] == 9999999999:
                        famArray[rowIndex] = rowIndex
                    else:
                        trans = famArray[rowIndex]
                        for x in range (0,num_records):
                            if famArray[x] == trans:
                                famArray[x] = rowIndex
                    if famArray[colIndex] == 9999999999:
                        famArray[colIndex] = rowIndex
```

```
                else:
                    trans = famArray[colIndex]
                    for x in range (0,num_records):
                        if famArray[x] == trans:
                            famArray[x] = rowIndex
                #outstring = "Found sibling at [%d][%d] !" % (rowIndex,colIndex
)
                #arcpy.AddMessage(outstring)
                #del outstring
            #-------------------------------------------------------------
            # FATHER: father's spouse is same as mother and father is older
than child
            #-------------------------------------------------------------
            elif (pInfo[row[0]+"father"] == pInfo[col[0]+"name"]) and (int(
pInfo[row[0]+"bday"][:4]) > int(pInfo[col[0]+"bday"][:4])+13):
                # found potential father
                # verify that the father's spouse is row's mother, this
eliminates many same name father occurances
                # DEBUG
                # arcpy.AddMessage("Father!")
                if (pInfo[row[0]+"mother"] == pInfo[col[0]+"spouse"]):
                    if (potFather[0] == -1):
                        potFather = (rowIndex,colIndex,col[0],col[3],pInfo[col
[0]+"name"])
                        # DEBUG
                        # arcpy.AddMessage("Verified!!!!!!!!!!!!!")
                    else:
                        # found more than one possible father
                        # to do: process
                        outstring = "Serious Error: Found Multiple Fathers for
:(%s %s) Prev:%s %s, Current:%s %s" % (row[0],pInfo[row[0]+"name"],potFather
[2],potFather[4],col[0],pInfo[col[0]+"name"])
                        arcpy.AddMessage(outstring)
                        del outstring
                else:
                    outstring = "Possible Issue: Mother =/= Spouse for:(%s %s)
Mother: %s, Father's Spouse:%s" % (row[0],pInfo[row[0]+"name"],pInfo[row[0]+"
mother"],pInfo[col[0]+"spouse"])
                    arcpy.AddMessage(outstring)
                    del outstring
            #-------------------------------------------------------------
            # MOTHER: Mother's spouse is same as father, and mother is older
than child
            #-------------------------------------------------------------
            elif (pInfo[row[0]+"mother"] == pInfo[col[0]+"name"])and (int(pInfo
[row[0]+"bday"][:4]) > int(pInfo[col[0]+"bday"][:4])+13):
                # found potential mother
                # verify that the mother's spouse is row's father, this
eliminates many same name mother occurances
                # DEBUG
                # arcpy.AddMessage("Mother!")
                if (pInfo[row[0]+"father"] == pInfo[col[0]+"spouse"]):
                    if (potMother[0] == -1):
                        potMother = (rowIndex,colIndex,col[0],col[3],pInfo[col
[0]+"name"])
                        # DEBUG
                        # arcpy.AddMessage("Verified!!!!!!!!!!!!!")
                    else:
                        # found more than one possible mother
                        # to do: process
                        outstring = "Serious Error: Found Multiple Mothers for
:(%s %s) Prev:%s %s, Current:%s %s" % (row[0],pInfo[row[0]+"name"],potMother
[2],potMother[4],col[0],pInfo[col[0]+"name"])
                        arcpy.AddMessage(outstring)
                        del outstring
                else:
                    outstring = "Possible Issue: Father =/= Spouse for:(%s %s)
Father: %s, Mother's Spouse:%s" % (row[0],pInfo[row[0]+"name"],pInfo[row[0]+"
father"],pInfo[col[0]+"spouse"])
                    arcpy.AddMessage(outstring)
                    del outstring
            #-------------------------------------------------------------
            # SPOUSE: Spouse's spouse is same
```

```
            #-------------------------------------------------------------
            elif (pInfo[row[0]+"spouse"] == pInfo[col[0]+"name"]):
                # found potential spouse
                # first check to make sure jane doe is married to john smith
AND john smith is married to jane doe
                # backcheck spouse's spouse is the person we are on
                # DEBUG
                # arcpy.AddMessage("Spouse!")
                if (pInfo[row[0]+"name"] == pInfo[col[0]+"spouse"]):
                    # we have a spouse/spouse match
                    # check to see if there has not been a prev match, it is
possible that there are multiple couples with both same names
                    if (potSpouse[0] == -1):
                        potSpouse = (rowIndex,colIndex,col[0],col[3],pInfo[col
[0]+"name"])
                        # DEBUG
                        # arcpy.AddMessage("Verified!!!!!!!!!!!!!")
                    else:
                        # found more than one possible spouse
                        # to do: analyze potSpouse and col's birthdays and
choose closest one to row's birthday
                        outstring = "Serious Error: Found Multiple Spouse for
:(%s %s) Prev:%s %s, Current:%s %s" % (row[0],pInfo[row[0]+"name"],potSpouse
[2],potSpouse[4],col[0],pInfo[col[0]+"name"])
                        arcpy.AddMessage(outstring)
                        del outstring
                else:
                    outstring = "Possible Issue: Not same Spouse for:(%s %s)
Spouse: %s, Spouse's Spouse:%s" % (row[0],pInfo[row[0]+"name"],pInfo[row[0]+"
spouse"],pInfo[col[0]+"spouse"])
                    arcpy.AddMessage(outstring)
                    del outstring
            #-------------------------------------------------------------
            # increment column loop counters etc
            #-------------------------------------------------------------
            colIndex += 1
            #
            # End column loop
            #
    # hopfully we have established correct father/mother/spouse relations (if
any)
    # set father/child reltionship values
    if not(potFather[0] == -1):
        relArray[potFather[0]][potFather[1]] = faValue
        relArray[potFather[1]][potFather[0]] = chValue
        if famArray[potFather[0]] == 9999999999:
            famArray[potFather[0]] = rowIndex
        else:
            trans = famArray[potFather[0]]
            for x in range (0,num_records):
                if famArray[x] == trans:
                    famArray[x] = rowIndex
        if famArray[potFather[1]] == 9999999999:
            famArray[potFather[1]] = rowIndex
        else:
            trans = famArray[potFather[1]]
            for x in range (0,num_records):
                if famArray[x] == trans:
                    famArray[x] = rowIndex
    # set mother/child reltionship values
    if not(potMother[0] == -1):
        relArray[potMother[0]][potMother[1]] = moValue
        relArray[potMother[1]][potMother[0]] = chValue
        if famArray[potMother[0]] == 9999999999:
            famArray[potMother[0]] = rowIndex
        else:
            trans = famArray[potMother[0]]
            for x in range (0,num_records):
                if famArray[x] == trans:
                    famArray[x] = rowIndex
        if famArray[potMother[1]] == 9999999999:
            famArray[potMother[1]] = rowIndex
        else:
```

```
                trans = famArray[potMother[1]]
                for x in range (0,num_records):
                    if famArray[x] == trans:
                        famArray[x] = rowIndex
        # set spouse reltionship values
        if not(potSpouse[0] == -1):
            relArray[potSpouse[0]][potSpouse[1]] = spValue
            relArray[potSpouse[1]][potSpouse[0]] = spValue
            if famArray[potSpouse[0]] == 9999999999:
                famArray[potSpouse[0]] = rowIndex
            else:
                trans = famArray[potSpouse[0]]
                for x in range (0,num_records):
                    if famArray[x] == trans:
                        famArray[x] = rowIndex
            if famArray[potSpouse[1]] == 9999999999:
                famArray[potSpouse[1]] = rowIndex
            else:
                trans = famArray[potSpouse[1]]
                for x in range (0,num_records):
                    if famArray[x] == trans:
                        famArray[x] = rowIndex
        #---------------------------------------------------------------
        # increment row loop counters etc
        #---------------------------------------------------------------
        rowIndex += 1
    else:
        # build the first row in the output file (header row), this should be the
    literal text "ID", or whatever the first value of the data input file's first (
    header) row is
        outfile1.write(row[0]+",")
#
# End row loop
#
rowIndex = colIndex= 0
arcpy.AddMessage("Finished Assigning Nuclear Relationships...")

#
# output 2D array, first header line has already been written during the primary
    loop
#
outfile1.write("\n")
for y in range (0,num_records):
    outfile1.write(idArray[y]+",")
    for x in range (0,num_records):
         outfile1.write(str(int(relArray[y][x]))+",")
    outfile1.write("\n")

outfile1.close()
arcpy.AddMessage("Done writing output table...")

#
# output Family ID table
#

IDfam = 1
famSavArray = range(num_records)

for y in range (0,num_records):
    famSavArray[y]=famArray[y]

for y in range (0,num_records):
    if famArray[y] != 9999999999:
        trans = famArray[y]
        for z in range (0,num_records):
            if famArray[z] == trans:
                famIDArray[z] = IDfam
                famArray[z] = 9999999999
        IDfam += 1

for y in range (0,num_records):
    outfile2.write(idArray[y]+",")
    if famIDArray[y] != 9999999999:
```

```
            outfile2.write("fam"+str(int(famIDArray[y]))+",")
        else:
            outfile2.write(""+",")
        outfile2.write("\n")

# verification famille id #
#for y in range (0,num_records):
#    outfile2.write(idArray[y]+",")
#    try:
#        outfile2.write(str(idArray[int(famSavArray[y])])+","+"fam"+str(int(
    famIDArray[y]))+",")
#
#    except:
#        outfile2.write(" "+","+" "+",")
#    outfile2.write("\n")


outfile2.close()
arcpy.AddMessage("Done writing relation table...")

#
# go have a beer
#
```

APPENDIX B
Sample of possible inconsistencies returned by the first python script

```
Possible Issue: Father =/= Spouse for:(I3275 KREINER EDITH) Father: , Mother's
    Spouse:KREINER ?
Possible Issue: Not same Spouse for:(I3277 KROHN MARTIN ISRAELE) Spouse: ALEXANDER
    GERTRUDE SARA, Spouse's Spouse:KROHN MARTIN
Possible Issue: Mother =/= Spouse for:(I3278 KROO ALESSANDRO) Mother: VAMOS NELLY,
    Father's Spouse:VAMOS NELLY RACHELE
Possible Issue: Mother =/= Spouse for:(I3300 KUPFER MICHELE) Mother: KUPFER SALOMEE
    ', Father's Spouse:ELKAN SALOMEE'
Possible Issue: Mother =/= Spouse for:(I3302 KURTZ CARLOTTA) Mother: KOENIG NINA,
    Father's Spouse:GALANDAUER BELLA
Possible Issue: Not same Spouse for:(I3309 KWRADRATSTEIN DEBORA) Spouse: PANZER
    ARON, Spouse's Spouse:KWADRATSTEIN DEBORA
Possible Issue: Mother =/= Spouse for:(I3311 LABI ABRAMO) Mother: REGINIANO MISA,
    Father's Spouse:LABI MESSALA
Possible Issue: Mother =/= Spouse for:(I3316 LABI ARONNE) Mother: MAZZUS EMILIA,
    Father's Spouse:BENDAUD JOLE
Possible Issue: Mother =/= Spouse for:(I3316 LABI ARONNE) Mother: MAZZUS EMILIA,
    Father's Spouse:MISA ?
Possible Issue: Mother =/= Spouse for:(I3317 LABI ARONNE) Mother: RUBIN GIULIA,
    Father's Spouse:BENDAUD JOLE
Possible Issue: Mother =/= Spouse for:(I3317 LABI ARONNE) Mother: RUBIN GIULIA,
    Father's Spouse:MISA ?
Possible Issue: Mother =/= Spouse for:(I3319 LABI DIAMANTINA) Mother: NAHUM RUBINA,
     Father's Spouse:BUCABSA SARINA
Possible Issue: Not same Spouse for:(I3319 LABI DIAMANTINA) Spouse: LABI SCIALOM,
    Spouse's Spouse:
Possible Issue: Not same Spouse for:(I3319 LABI DIAMANTINA) Spouse: LABI SCIALOM,
    Spouse's Spouse:
Possible Issue: Father =/= Spouse for:(I3321 LABI DIAMANTINA) Father: LABI ISACCO
    GIUSEPPE, Mother's Spouse:LABI GIUSEPPE
Possible Issue: Father =/= Spouse for:(I3322 LABI DIANA) Father: LABI SALOMONE
    RENATO, Mother's Spouse:LABI RENATO
Possible Issue: Mother =/= Spouse for:(I3331 LABI GINO) Mother: BENDAUD JOLE,
    Father's Spouse:MISA ?
```

APPENDIX C
AFG detection python script

```
#*******************************************************************************
# python 2.6.5 program to group family members arrested together
# by Mael Le Noc - mael.lenoc@txstate.edu - Spring 2015
#
# !!!!!!!! WARNING !!!!!!!
# Your CSV files that you Save As from Excel or generate from some other source
# MUST BE IN THE EXACT FORMAT AS BELOW. I have provided an example of first line
# and first record (1 to 2 lines) for both files.
#
#Input file 1: personal information : Saved As CSV from Excel : (data\
  social_network_data.csv)
#ID,PLACE OF ARREST, X COORDINATE, Y COORDINATE, DATE OF ARREST,FAMILY ID, ARRESTED
   BY, NATIONALITY,
#I0002,CIVITELLA DEL TRONTO,13.667029,42.772396,1943.11.30,1943,fam1, Italiani, it,
#
# program details:
#
# creates table that createw arrest family groups and counts number of member in
  those groups as well as total number of family member (including himself/herself)
  . Output one table by family group and one table by family members
#
#*******************************************************************************
import os, datetime, arcpy
#from operator import itemgetter
from numpy import *

infile1 = open(arcpy.GetParameterAsText(0),"r").readlines() # data input csv
outdir = arcpy.GetParameterAsText(1) # output folder/dir
outdir2 = arcpy.GetParameterAsText(3) # output folder/dir

#-------------------------------------------------------------------------------
# process output file name: ensure they will be csv files
#-------------------------------------------------------------------------------
rfile = root = ext = ""
rfile = arcpy.GetParameterAsText(2) # output filename csv
if ("." in rfile):
        (root,ext) = rfile.split(".")
else:
        root = rfile; ext = ""
if (ext != "csv"):
        rfile = root + ".csv"
outfile1= open(os.path.join(outdir,rfile),"w")

rfile2 = root2 = ext2 = ""
rfile2 = arcpy.GetParameterAsText(4) # output filename csv
if ("." in rfile2):
        (root2,ext2) = rfile2.split(".")
else:
        root2 = rfile2; ext2 = ""
if (ext2 != "csv"):
        rfile2 = root2 + ".csv"
outfile2= open(os.path.join(outdir2,rfile2),"w")

del rfile; del root; del ext
del rfile2; del root2; del ext2

#-------------------------------------------------------------------------------
# declair variables
#-------------------------------------------------------------------------------
timeSuffix = datetime.datetime.now().strftime("_%Y_%m_%d__%H_%M_%S")
arcpy.env.overwriteOutput = True
idArray = []
oidArray = []
num_records = len(infile1) - 1 # -1 to account for header
arrestArray = zeros((num_records,4))
afgID = 0
afgdict = {}
outstring = "Initialized table of %d records..." % num_records
```

```
arcpy.AddMessage(outstring)
del outstring;

#-------------------------------------------------------------------------------
# build dictionary of IDs name, arrest place and date for quick data retrieval, and
   makes it easier to read
#-------------------------------------------------------------------------------
pInfo = {}
count = 0
for line in infile1:
        w = line.split(",")
        try:
                w[7] = w[7].rstrip("\n")
                w[7] = w[7].rstrip(" ")
        except:
                w[7] = w[7]
        #rough check to ensure we are on a personal line and not the header
        if not('ID' in w[0]):
                count += 1
                # build oid array for processing
                oidArray.append(w[0])
                #ensure ID is length 5, pad with zeros after the 'I' and build
  array, this is for output formatting
                newID = w[0]
                addZeros = 5 - len(w[0])
                if (addZeros > 0):
                        newID = w[0][0]
                        for x in range (0,addZeros):
                                newID += "0"
                        newID += w[0][(0-(len(w[0])-1)):]
                idArray.append(newID)
                del addZeros;
                #store: [id:place,id:X;id:Y,id:date,id:family,id:perpetrator,id:
  nationality]
                if (('nessuna' in w[1].lower()) or ('sconosciuto' in w[1].lower()))
  :
                        pInfo[w[0]+"place"] = ""
                else:
                        pInfo[w[0]+"place"] = w[1]
                pInfo[w[0]+"X"] = w[2]
                pInfo[w[0]+"Y"] = w[3]
                if (('nessuna' in w[4].lower()) or ('sconosciuto' in w[4].lower()))
  :
                        pInfo[w[0]+"date"] = ""
                else:
                        pInfo[w[0]+"date"] = w[4]
                pInfo[w[0]+"family"] = w[5]
                if (('nessuna' in w[6].lower()) or ('sconosciuto' in w[6].lower()))
  :
                        pInfo[w[0]+"perpetrator"] = ""
                else:
                        pInfo[w[0]+"perpetrator"] = w[6]
                if (('nk' in w[7].lower())):
                        pInfo[w[0]+"nationality"] = ""
                else:
                        pInfo[w[0]+"nationality"] = w[7]


del line;  del w
outstring = "Done with building personal dictionary of %d records..." % count
arcpy.AddMessage(outstring)
del outstring

#-------------------------------------------------------------------------------
# loop through records and determine matches
#-------------------------------------------------------------------------------
rowIndex = -1
for rows in infile1:
        row = rows.split(",")
        # ensure we are not on the header row and proceed
        if not('ID' in row[0]):
                #-----------------------------------------------------------
                # increment row loop counters etc
```

```
                #-------------------------------------------------------------
                rowIndex += 1
                if (pInfo[row[0]+"family"] != ""):
                        arrestArray[rowIndex][1] = 1
                        # loop through file again to look for matches
                        rowbisIndex = -1
                        for rowbiss in infile1:
                                rowbis = rowbiss.split(",")
                                if not('ID' in rowbis[0]):

#-------------------------------------------------------------
                                # increment row loop counters etc

#-------------------------------------------------------------
                                rowbisIndex += 1

#-------------------------------------------------------------
                                # check that it's not the same person, and
then that they have same family

#-------------------------------------------------------------
                                if (row[0] != rowbis[0]) :
                                        if (pInfo[row[0]+"family"] == pInfo
[rowbis[0]+"family"]) :
                                                arrestArray[rowIndex][1] +=
 1

                                                # DEBUG
                                                #outstring = " %s ..." % (pInfo[row
[0]+"family"])
                                                #arcpy.AddMessage(outstring)
                                                #del outstring

#-------------------------------------------------------------
                                                # check that place and date is not
empty and that they have same place, date

#-------------------------------------------------------------
                                                if (pInfo[row[0]+"place"]
!= "") and (pInfo[row[0]+"date"] != "") and (pInfo[row[0]+"place"] == pInfo[
rowbis[0]+"place"]) and (pInfo[row[0]+"date"] == pInfo[rowbis[0]+"date"]) :
                                                        # DEBUG
                                                        #outstring = " Same
 afg %s and %s, Indexes %d and %d " % (idArray[rowIndex], idArray[rowbisIndex],
rowIndex, rowbisIndex )
                                                        #arcpy.AddMessage(
outstring)
                                                        #del outstring


#-------------------------------------------------------------
                                                        # increment afg
size

#-------------------------------------------------------------
                                                        if arrestArray[
rowIndex][2] == 0 :
                                                                arrestArray
[rowIndex][2] += 1
                                                        arrestArray[
rowIndex][2] += 1


#-------------------------------------------------------------
                                                        # check if afg has
been assign to either one of them

#-------------------------------------------------------------
                                                        if arrestArray[
rowIndex][3] == 0 and arrestArray[rowbisIndex][3] == 0 :
                                                                afgID += 1
                                                                arrestArray
[rowIndex][3] = afgID
```

```
                                                                    arrestArray
[rowbisIndex][3] = afgID
                                                                    afgdict[str
(afgID)+"ID"] = "afg" + str(afgID,)
                                                                    afgdict[str
(afgID)+"famID"] = pInfo[row[0]+"family"]
                                                                    afgdict[str
(afgID)+"place"] = pInfo[row[0]+"place"]
                                                                    afgdict[str
(afgID)+"X"] = pInfo[row[0]+"X"]
                                                                    afgdict[str
(afgID)+"Y"] = pInfo[row[0]+"Y"]
                                                                    afgdict[str
(afgID)+"date"] = pInfo[row[0]+"date"]
                                                                    afgdict[str
(afgID)+"numafg"] = arrestArray[rowIndex][2]
                                                                    afgdict[str
(afgID)+"numfam"] = arrestArray[rowIndex][1]
                                                                    # DEBUG
                                                                    # outstring
 = " Cas 1 %s %s " % (idArray[rowIndex], idArray[rowbisIndex])
                                                                    # arcpy.
AddMessage(outstring)
                                                                    # del
outstring

#---------------------------------------------------------
                                                                    # compare
perpetrators and nationality
#---------------------------------------------------------
                                                                    if pInfo[
row[0]+"perpetrator"] == pInfo[rowbis[0]+"perpetrator"] :
afgdict[str(afgID)+"perpetrator"] = pInfo[row[0]+"perpetrator"]
                                                                    elif pInfo[
row[0]+"perpetrator"] == "" or pInfo[rowbis[0]+"perpetrator"] == "" :
afgdict[str(afgID)+"perpetrator"] = "partly unknown"
                                                                    else :
afgdict[str(afgID)+"perpetrator"] = "mixed"

                                                                    if pInfo[
row[0]+"nationality"] == pInfo[rowbis[0]+"nationality"] :
afgdict[str(afgID)+"nationality"] = pInfo[row[0]+"nationality"]
                                                                    elif pInfo[
row[0]+"nationality"] == "" or pInfo[rowbis[0]+"nationality"] == "" :
afgdict[str(afgID)+"nationality"] = "partly unknown"
                                                                    else :
afgdict[str(afgID)+"nationality"] = "mixed"

                                                                elif arrestArray[
rowbisIndex][3] == 0 :
                                                                    arrestArray
[rowbisIndex][3] = arrestArray[rowIndex][3]
                                                                    afgdict[str
(afgID)+"numafg"] = arrestArray[rowIndex][2]
                                                                    afgdict[str
(afgID)+"numfam"] = arrestArray[rowIndex][1]
                                                                    # DEBUG
                                                                    # outstring
 = " Cas 2 %s %s " % (idArray[rowIndex], idArray[rowbisIndex])
                                                                    # arcpy.
AddMessage(outstring)
                                                                    # del
outstring
```

```
#-------------------------------------------------------------
                                                    # compare
perpetrators and nationality
#-------------------------------------------------------------
                                                    if pInfo[
rowbis[0]+"perpetrator"] == afgdict[str(afgID)+"perpetrator"] :
pass
                                                    elif pInfo[
rowbis[0]+"perpetrator"] == "" :
afgdict[str(afgID)+"perpetrator"] = "partly unknown"
                                                    else :
afgdict[str(afgID)+"perpetrator"] = "mixed"

                                                    if pInfo[
rowbis[0]+"nationality"] == afgdict[str(afgID)+"nationality"] :
pass
                                                    elif pInfo[
rowbis[0]+"nationality"] == "" :
afgdict[str(afgID)+"nationality"] = "partly unknown"
                                                    else :
afgdict[str(afgID)+"nationality"] = "mixed"

                                                elif arrestArray[
rowIndex][3] == 0 :
                                                    arrestArray
[rowIndex][3] = arrestArray[rowbisIndex][3]
                                                    afgdict[str
(afgID)+"numafg"] = arrestArray[rowbisIndex][2]
                                                    afgdict[str
(afgID)+"numfam"] = arrestArray[rowbisIndex][1]
                                                    outstring =
 " Cas 3 %s %s - should not happen " % (idArray[rowIndex], idArray[rowbisIndex])
                                                    arcpy.
AddMessage(outstring)
                                                    del
outstring
                                                elif arrestArray[
rowIndex][3] != arrestArray[rowbisIndex][3] :
                                                    outstring =
 " Error : %s and %s not in same arrest group (afg %s and %s)" % (idArray[
rowIndex], idArray[rowbisIndex], arrestArray[rowIndex][3], arrestArray[
rowbisIndex][3])
                                                    arcpy.
AddMessage(outstring)
                                                    del
outstring
                                                elif arrestArray[
rowIndex][3] == arrestArray[rowbisIndex][3] :
                                                    # DEBUG
                                                    # outstring
 = " %s and %s already in same arrest group :afg %s " % (idArray[rowIndex],
idArray[rowbisIndex], arrestArray[rowIndex][3])
                                                    # arcpy.
AddMessage(outstring)
                                                    # del
outstring
                                                    pass
                                                else :
                                                    outstring =
 " Error : %s and %s other case (afg %s and %s), logically not possible" % (
idArray[rowIndex], idArray[rowbisIndex], arrestArray[rowIndex][3], arrestArray[
rowbisIndex][3])
                                                    arcpy.
AddMessage(outstring)
                                                    del
outstring
```

```
                # DEBUG
                #outstring = " %s ..." % idArray[rowIndex]
                #arcpy.AddMessage(outstring)
                #del outstring

#
# End row loop
#
rowIndex = colIndex= 0
arcpy.AddMessage("Finished checking arrests")
outstring = "%d group created " % (afgID)
arcpy.AddMessage(outstring)
del outstring

#
# output table
#
outfile1.write("ID, Family size, Arrested group size, Arrest family group ID")
outfile1.write("\n")

for y in range (0,num_records):
        outfile1.write(idArray[y]+",")
        outfile1.write(str(int(arrestArray[y][1]))+",")
        outfile1.write(str(int(arrestArray[y][2]))+",")
        if arrestArray[y][3] != 0 :
                outfile1.write("afg"+str(int(arrestArray[y][3]))+",")
        else :
                outfile1.write(",")
        outfile1.write("\n")

outfile1.close()
arcpy.AddMessage("Done writing output table for individuals...")

outfile2.write("Arrest group ID, Arrest group size, Place of arrest, X, Y, Date of
  arrest, Perpetrator, Nationality, Family ID, Family size")
outfile2.write("\n")

for x in range (1,afgID):
        outfile2.write(afgdict[str(x)+"ID"]+",")
        outfile2.write(str(int(afgdict[str(x)+"numafg"]))+",")
        outfile2.write(afgdict[str(x)+"place"]+",")
        outfile2.write(afgdict[str(x)+"X"]+",")
        outfile2.write(afgdict[str(x)+"Y"]+",")
        outfile2.write(afgdict[str(x)+"date"]+",")
        if afgdict[str(x)+"perpetrator"] == "" :
                outfile2.write("unknown"+",")
        else :
                outfile2.write(afgdict[str(x)+"perpetrator"]+",")
        if afgdict[str(x)+"nationality"] == "" :
                outfile2.write("unknown"+",")
        else :
                outfile2.write(afgdict[str(x)+"nationality"]+",")
        outfile2.write(afgdict[str(x)+"famID"]+",")
        outfile2.write(str(int(afgdict[str(x)+"numfam"]))+",")
        outfile2.write("\n")
outfile2.close()
arcpy.AddMessage("Done writing output table for arrest group...")

#
# Va boire une biere
#
```

APPENDIX D
Family separation python script

```
#*******************************************************************************

# python 2.6.5 program to convert csv data concerning people and their relations
  and explore family separation
# script by Mael Le Noc - mael.lenoc@txstate.edu - Spring 2016
#
# !!!!!!!! WARNING !!!!!!!
# Your CSV files that you Save As from Excel or generate from some other source
# MUST BE IN THE EXACT FORMAT AS BELOW. I have provided an example of first line
# and first record (1 to 2 lines) for the file.
#
#Input file 1: personal information : Saved As CSV from Excel : (data\
  social_network_data.csv)
#ID,PNC,fatherID,motherID,spouseID,ANNO_ARR,CHI_ARRES,Det1,det2,det3,det4,det5,
  RACCOLTA,LAGER,CONVOY,SESSO,ETA_ARR,famID,AFGID,arrest_place,fate
#I0003,ni,I0002,I2900,,1943,Italiani,CIVITELLA DEL TRONTO CAMPO,FOSSOLI CAMPO,,,,
  FOSSOLI,BERGEN BELSEN,11,FEMMINILE,17,fam1,afg1,ROMA,LIB.
#
# program details:
#
# compare detention locations with other members of AFGs and assign Family
  separation index score at different point in the deportation process
#
#
#*******************************************************************************

import os, datetime, arcpy
#from operator import itemgetter
from numpy import *

infile1 = open(arcpy.GetParameterAsText(0),"r").readlines() # data input csv
outdir = arcpy.GetParameterAsText(1) # output folder/dir
outdir2 = arcpy.GetParameterAsText(3) # output folder/dir


#-------------------------------------------------------------------------------

# process output file name: ensure they will be txt files
#-------------------------------------------------------------------------------

rfile = root = ext = ""
rfile = arcpy.GetParameterAsText(2) # output filename txt
if ("." in rfile):
    (root,ext) = rfile.split(".")
else:
    root = rfile; ext = ""
if (ext != "txt"):
    rfile = root + ".csv"
outfile1= open(os.path.join(outdir,rfile),"w")
rfile2 = root2 = ext2 = ""
rfile2 = arcpy.GetParameterAsText(4) # output filename txt
if ("." in rfile2):
    (root2,ext2) = rfile2.split(".")
else:
    root2 = rfile2; ext2 = ""
if (ext2 != "txt"):
    rfile2 = root2 + ".csv"
outfile2= open(os.path.join(outdir2,rfile2),"w")
del rfile; del root; del ext
del rfile2; del root2; del ext2


#-------------------------------------------------------------------------------

# declair variables
#-------------------------------------------------------------------------------

sepNotAll = 0.8 # Victim separated from at least one other member of their AFG but
  not all
```

```
sepAll = 0.5 # Victim separated from all other members of their AFG
sepChildFromMom = 0.5 # Child separated from his/her mother but not his/her father
  (or father unknown)
sepChildFromDad = 0.6 # Child separated from his/her father but not his/her mother
sepChildFromBoth = 0.3 # Child separated from both parents
sepMomFromChild = 0.6 # Mother separated from at least one of her children, but not
   all
sepMomFromAllChild = 0.5 # Mother separated from all of her children
sepDadFromChild = 0.6 # Father separated from at least one of her children, but not
   all
sepDadFromAllChild = 0.5 # Father separated from all of her childrengcValue = 7 #
  grandchild
sepFromSpouse = 0.7 # Spouses separated from one another


timeSuffix = datetime.datetime.now().strftime("_%Y_%m_%d__%H_%M_%S")
arcpy.env.overwriteOutput = True
idArray = []
oidArray = []
num_records = len(infile1) - 1 # -1 to account for header


#-------------------------------------------------------------------------------

# build dictionary of IDs name, birthday, mother and father's name for quick data
  retrieval, and makes it easier to read
#-------------------------------------------------------------------------------

pInfo = {}
count = 0
for line in infile1:
        w = line.split(",")
        try:
                w[18] = w[18].rstrip("\n")
                w[18] = w[18].rstrip(" ")
        except:
                w[18] = w[18]
    #rough check to ensure we are on a personal line and not the header
        if not('ID' in w[0]):
                count += 1
        # build oid array forprocessing
                oidArray.append(w[0])
        #ensure ID is length 5, pad with zeros after the 'I' and build array, this
  is for output formatting
                newID = w[0]
                addZeros = 5 - len(w[0])
                if (addZeros > 0):
                        newID = w[0][0]
                        for x in range (0,addZeros):
                                newID += "0"
                        newID += w[0][(0-(len(w[0])-1)):]
                idArray.append(newID)
                del addZeros;
        #store: [id:id, id:PNC...] ID,PNC,fatherID,motherID,spouseID,ANNO_ARR,
  CHI_ARRES,Det1,det2,det3,det4,det5,RACCOLTA,LAGER,CONVOY,SESSO,ETA_ARR,famID,
  AFGID,arrest,fate
                pInfo[w[0]+"id"] = w[0]
                pInfo[w[0]+"famid"] = w[17]
                pInfo[w[0]+"afgid"] = w[18]
                pInfo[w[0]+"father"] = w[2]
                pInfo[w[0]+"mother"] = w[3]
                pInfo[w[0]+"spouse"] = w[4]
                pInfo[w[0]+"arrest"] = w[19]
                pInfo[w[0]+"det1"] = w[7]
                pInfo[w[0]+"det2"] = w[8]
                pInfo[w[0]+"det3"] = w[9]
                pInfo[w[0]+"det4"] = w[10]
                pInfo[w[0]+"det5"] = w[11]
                pInfo[w[0]+"racc"] = w[12]
                pInfo[w[0]+"lager"] = w[13]
                pInfo[w[0]+"convoy"] = w[14]
                if ('it' in w[1].lower()):
                        pInfo[w[0]+"nat"] = "0"
```

```
                elif ('ni' in w[1].lower()):
                        pInfo[w[0]+"nat"] = "1"
                else:
                        pInfo[w[0]+"nat"] = ""
                if ('1943' in w[5]):
                        pInfo[w[0]+"anar"] = "0"
                elif (('1944' in w[5]) or ('1945' in w[5])):
                        pInfo[w[0]+"anar"] = "1"
                else:
                        pInfo[w[0]+"anar"] = ""
                if ('italiani con tedeschi' in w[6].lower()):
                        pInfo[w[0]+"perp1"] = "0"
                        pInfo[w[0]+"perp2"] = "0"
                elif ('tedeschi' in w[6].lower()):
                        pInfo[w[0]+"perp1"] = "1"
                        pInfo[w[0]+"perp2"] = "0"
                elif ('italiani' in w[6].lower()):
                        pInfo[w[0]+"perp1"] = "0"
                        pInfo[w[0]+"perp2"] = "1"
                else:
                        pInfo[w[0]+"perp1"] = ""
                        pInfo[w[0]+"perp2"] = ""
                if ('femminile' in w[15].lower()):
                        pInfo[w[0]+"gender"] = "0"
                elif ('maschile' in w[15].lower()):
                        pInfo[w[0]+"gender"] = "1"
                else:
                                pInfo[w[0]+"gender"] = ""
                if ('lib.' in w[20].lower()):
                        pInfo[w[0]+"fate"] = "0"
                elif ('dec.' in w[20].lower()):
                        pInfo[w[0]+"fate"] = "1"
                else:
                                pInfo[w[0]+"fate"] = ""
                if (int(w[16])<19):
                        pInfo[w[0]+"kid"] = "1"
                        pInfo[w[0]+"elderly"] = "0"
                elif (int(w[16])>69):
                        pInfo[w[0]+"kid"] = "0"
                        pInfo[w[0]+"elderly"] = "1"
                else:
                        pInfo[w[0]+"kid"] = "0"
                        pInfo[w[0]+"elderly"] = "0"
                pInfo[w[0]+"age"] = w[16]
                pInfo[w[0]+"sepfromspouse"] = ""
                pInfo[w[0]+"sepfromkids"] = ""
                pInfo[w[0]+"sepfromparents"] = ""
                pInfo[w[0]+"reunion"] = ""
                pInfo[w[0]+"sep1"] = ""
                pInfo[w[0]+"sep2"] = ""
                pInfo[w[0]+"more"] = ""

del line;  del w
outstring = "Done with building personal dictionary of %d records..." % count
arcpy.AddMessage(outstring)
del outstring

#-----------------------------------------------------------------------------

# loop through records and determine FSIS
#-----------------------------------------------------------------------------

rowIndex = 0
for rows in infile1:
    row = rows.split(",")
    # ensure we are not on the header row and proceed
    if not('ID' in row[0]):
                # setup lists of members of the AFG
                listafg =[]
                listadet1 =[]
                listadet2 = []
                listadet3 = []
                listadet4 = []
```

```
                listadet5 =[]
                listaracc =[]
                listaconvoy =[]
                listalagger =[]
                listchildren =[]
                listchafg = []
                listchadet1 = []
                listchadet2 = []
                listchadet3 = []
                listchadet4 = []
                listchadet5 = []
                listcharacc = []
                listchaconvoy = []
                listfam = []
                # loop through file again to look for matches
                colIndex = 0
                for cols in infile1:
                        col = cols.split(",")
                        if not('ID' in col[0]):

#-------------------------------------------------------------
                                #create lists of children

#-------------------------------------------------------------
                                if  (int(col[16]) <= 17) and ((row[0] == col[2]) or
 (row[0] == col[3])) :
                                        listchildren.append(col[0])

#-------------------------------------------------------------
                                # populate lists with ID of all AFG members

#-------------------------------------------------------------
                                if (row[18] == col[18]) :
                                        listafg.append(col[0])
                                        if ((row[7] == "") and (col[7] == "")):
                                                if ((row[12] == col[12]) or (row
[12] in col[12]) or (col[12] in row[12])):
                                                        listaracc.append(col[0])
                                        elif (((row[7] == col[7]) or (row[7] in col
[7]) or (col[7] in row[7])) and (row[7] != "") and (col[7] != "")) or ((row[6] !=
 "") and (row[7] == "") and (col[7] != "")):
                                                listadet1.append(col[0])
                                                if ((row[8] == "") and (col[8] ==
"")):
                                                        if ((row[12] == col[12]) or
 (row[12] in col[12]) or (col[12] in row[12])):
                                                                listaracc.append(
col[0])
                                                elif (((row[8] == col[8]) or (row
[8] in col[8]) or (col[8] in row[8])) and (row[8] != "") and (col[8] != "")) or
((row[7] != "") and (row[8] == "") and (col[8] != "")) :
                                                        listadet2.append(col[0])
                                                        if ((row[9] == "") and (col
[9] == "")):
                                                                if ((row[12] == col
[12]) or (row[12] in col[12]) or (col[12] in row[12])):
                                                                        listaracc.
append(col[0])
                                                        elif (((row[9] == col[9])
or (row[9] in col[9]) or (col[9] in row[9])) and (row[9] != "") and (col[9] !=
"")) or ((row[8] != "") and (row[9] == "") and (col[9] != "")):
                                                                listadet3.append(
col[0])
                                                                if ((row[10] == "")
 and (col[10] == "")):
                                                                        if ((row
[12] == col[12]) or (row[12] in col[12]) or (col[12] in row[12])):

listaracc.append(col[0])
                                                                elif (((row[10] ==
col[10]) or (row[10] in col[10]) or (col[10] in row[10])) and (row[10] != "") and
 (col[10] != "")) or ((row[9] != "") and (row[10] == "") and (col[10] != "")):
                                                                        listadet4.
```

```
append(col[0])
                                                                    if ((row
[11] == "") and (col[11] == "")):
                                                                        if
((row[12] == col[12]) or (row[12] in col[12]) or (col[12] in row[12])):

        listaracc.append(col[0])
                                                                    elif (((row
[11] == col[11]) or (row[11] in col[11]) or (col[11] in row[11])) and (row[11] !=
 "") and (col[11] != "")) or ((row[10] != "") and (row[11] == "") and (col[11] !=
 "")):

listadet5.append(col[0])
                                                                        if
((row[12] == col[12]) or (row[12] in col[12]) or (col[12] in row[12])):

        listaracc.append(col[0])
                                        if (row[14] == col[14]):
                                                listaconvoy.append(col[0])
                                                if ((row[12] == col[12]) or (row
[12] in col[12]) or (col[12] in row[12])) and (col[0] not in listaracc):
                                                        listaracc.append(col[0])
                                        if (row[13] == col[13]):
                                                listalagger.append(col[0])
                                if (row[17] == col[17]) and (row[0] != col[0]) and
(row[14] == col[14]):
                                        listfam.append(col[0])
                                if ((col[0] in listchildren) and (col[0] in listafg
)):
                                        listchafg.append(col[0])
                                if ((col[0] in listchildren) and (col[0] in
listaracc)):
                                        listcharacc.append(col[0])
                                if ((col[0] in listchildren) and (col[0] in
listaconvoy)):
                                        listchaconvoy.append(col[0])
                for reun in infile1:
                        reu = reun.split(",")
                        if not('ID' in reu[0]):
                                if (reu[0] in listfam) and (((((reu[0] not in
listadet1) or (row[0] not in listadet1)) and (len(listadet1) != 0)) or ((len(
listadet1) == 0) and (reu[7] != ""))) or ((((reu[0] not in listadet2) or (row[0]
not in listadet2)) and (len(listadet2) != 0)) or ((len(listadet2) == 0) and (reu
[8] != ""))) or ((((reu[0] not in listadet3) or (row[0] not in listadet3)) and (
len(listadet3) != 0)) or ((len(listadet3) == 0) and (reu[9] != ""))) or ((((reu
[0] not in listadet4) or (row[0] not in listadet4)) and (len(listadet4) != 0)) or
 ((len(listadet4) == 0) and (reu[10] != ""))) or ((((reu[0] not in listadet5) or
(row[0] not in listadet5)) and (len(listadet5) != 0)) or ((len(listadet5) == 0)
and (reu[11] != ""))) or (reu[0] not in listaracc)):
                                        pInfo[row[0]+"reunion"] = "withafg"
                                if ((reu[0] in listfam) and (reu[0] not in listafg
)):
                                                pInfo[row[0]+"reunion"] = pInfo[row
[0]+"reunion"] + "withfam"
                                if ((reu[0] in listchildren) and (reu[0] in
listadet1)):
                                        listchadet1.append(reu[0])
                                if ((reu[0] in listchildren) and (reu[0] in
listadet2)):
                                        listchadet2.append(reu[0])
                                if ((reu[0] in listchildren) and (reu[0] in
listadet3)):
                                        listchadet3.append(reu[0])
                                if ((reu[0] in listchildren) and (reu[0] in
listadet4)):
                                        listchadet4.append(reu[0])
                                if ((reu[0] in listchildren) and (reu[0] in
listadet5)):
                                        listchadet5.append(reu[0])
                #-----------------------------------------------------
                # check for separation and assign fsis
                #-----------------------------------------------------
                #
```

```
                # Stg 1 - 1
                # sep bw arrest and det 1
                pInfo[row[0]+"fsis1"] = 1.0
                if (len(listadet1) != 0):
                        if (set(listafg).issubset(set(listadet1))):
                                pInfo[row[0]+"fsis1"] = pInfo[row[0]+"fsis1"] * 1.0
                        elif ((len(listadet1) == 1) and (row[0] in listadet1)) or
((set(listafg) - set(listadet1)) == set((row[0],))):
                                pInfo[row[0]+"fsis1"] = pInfo[row[0]+"fsis1"] * 0.5
                                pInfo[row[0]+"sep1"] = pInfo[row[0]+"arrest"]
                        else :
                                pInfo[row[0]+"fsis1"] = pInfo[row[0]+"fsis1"] * 0.8
                                pInfo[row[0]+"sep1"] = pInfo[row[0]+"arrest"]
                        if (len(listchildren) != 0):
                                if (len(listchafg) != 0)  and (((len(listchadet1)
== 0) and (row[0] in listadet1)) or ((set(listchafg) == set(listchadet1)) and (
row[0] not in listadet1))):
                                        pInfo[row[0]+"fsis1"] = pInfo[row[0]+"fsis1
"] * 0.5
                                        pInfo[row[0]+"sepfromkids"] = pInfo[row
[0]+"sepfromkids"] + "all"
                                else :
                                        c1 = 0
                                        c2 = 0
                                        for c in listchafg :
                                                if c not in listchadet1 :
                                                        c1 = 1
                                        if c1 == 1 :
                                                pInfo[row[0]+"fsis1"] = pInfo[row
[0]+"fsis1"] * 0.6
                                                pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "notall"
                                                outstring = "id %s | col[0] %s |
set(listchafg)  %s | set(listchadet1) %s | set(listadet1) %s"  %(pInfo[row[0]+"id
"], col[0],', '.join(listchafg), ', '.join(listchadet1), ', '.join(listadet1))
                                                arcpy.AddMessage(outstring)
                                                del outstring
                                        for c in listchadet1 :
                                                if c not in listchafg :
                                                        c2 = 1
                                        if c2 == 1 :
                                                pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "reunion"
                        if ((pInfo[row[0]+"spouse"] != "") and (pInfo[row[0]+"
spouse"] in listafg) and (row[0] in listafg) and (((pInfo[row[0]+"spouse"] not in
 listadet1) and (row[0] in listadet1)) or ((pInfo[row[0]+"spouse"] in listadet1)
and (row[0] not in listadet1)))):
                                pInfo[row[0]+"fsis1"] =  pInfo[row[0]+"fsis1"] *
0.7
                                pInfo[row[0]+"sepfromspouse"] = "yes"
                        elif ((pInfo[row[0]+"spouse"] != "") and (pInfo[row[0]+"
spouse"] not in listafg)):
                                pInfo[row[0]+"sepfromspouse"] = "byarrest"
                        if (int(pInfo[row[0]+"age"]) <= 17):
                                if (pInfo[row[0]+"mother"] in listafg) and (row[0]
in listafg) and ((((pInfo[row[0]+"father"] == "") or (pInfo[row[0]+"father"] in
listadet1)) and (row[0] in listadet1) and (pInfo[row[0]+"mother"] not in
listadet1)) or (((pInfo[row[0]+"father"] == "") or (pInfo[row[0]+"father"] not in
 listadet1)) and (row[0] not in listadet1) and (pInfo[row[0]+"mother"] in
listadet1))):
                                        pInfo[row[0]+"fsis1"] =  pInfo[row[0]+"
fsis1"] * 0.5
                                        pInfo[row[0]+"sepfromparents"] = "mother"
                                elif (pInfo[row[0]+"father"] in listafg) and (row
[0] in listafg) and ((((pInfo[row[0]+"mother"] == "") or (pInfo[row[0]+"mother"]
in listadet1)) and (row[0] in listadet1) and (pInfo[row[0]+"father"] not in
listadet1)) or (((pInfo[row[0]+"mother"] == "") or (pInfo[row[0]+"mother"] not in
 listadet1)) and (row[0] not in listadet1) and (pInfo[row[0]+"father"] in
listadet1))):
                                        pInfo[row[0]+"fsis1"] =  pInfo[row[0]+"
fsis1"] * 0.6
                                        pInfo[row[0]+"sepfromparents"] = "father"
                                elif (pInfo[row[0]+"father"] in listafg) and (row
```

```
[0] in listafg) and (((pInfo[row[0]+"father"] not in listadet1) and (pInfo[row
[0]+"mother"] not in listadet1) and (row[0] in listadet1)) or (pInfo[row[0]+"
father"] in listadet1) and (pInfo[row[0]+"mother"] in listadet1) and (row[0] not
in listadet1)):
                                        pInfo[row[0]+"fsis1"] =  pInfo[row[0]+"
fsis1"] * 0.3
                                        pInfo[row[0]+"sepfromparents"] = "both"
                                elif (pInfo[row[0]+"mother"] in listafg) and (row
[0] in listafg) and (((pInfo[row[0]+"father"] not in listadet1) and (pInfo[row
[0]+"mother"] not in listadet1) and (row[0] in listadet1)) or (pInfo[row[0]+"
father"] in listadet1) and (pInfo[row[0]+"mother"] in listadet1) and (row[0] not
in listadet1)):
                                        pInfo[row[0]+"fsis1"] =  pInfo[row[0]+"
fsis1"] * 0.3
                                        pInfo[row[0]+"sepfromparents"] = "both"
            # Stg 1
            # sep bw arrest and raccolta  if d1 do not exist thus no stage 2 or
 3
             elif (len(listadet1) == 0):
                    if (set(listafg).issubset(set(listaracc))):
                            pInfo[row[0]+"fsis1"] = pInfo[row[0]+"fsis1"] * 1.0
                    elif (len(listaracc) == 1):
                            pInfo[row[0]+"fsis1"] = pInfo[row[0]+"fsis1"] * 0.5
                            pInfo[row[0]+"sep1"] = pInfo[row[0]+"arrest"]
                    else:
                            pInfo[row[0]+"fsis1"] = pInfo[row[0]+"fsis1"] * 0.8
                            pInfo[row[0]+"sep1"] = pInfo[row[0]+"arrest"]
                    if (len(listchildren) != 0):
                            if (len(listchafg) != 0) and (len(listcharacc) ==
0):
                                    pInfo[row[0]+"fsis1"] = pInfo[row[0]+"fsis1
"] * 0.5
                                    pInfo[row[0]+"sepfromkids"] =  pInfo[row
[0]+"sepfromkids"] +"all"
                            else :
                                    c1 = 0
                                    c2 = 0
                                    for c in listchafg :
                                            if c not in listcharacc :
                                                    c1 = 1
                                    if c1 == 1 :
                                            pInfo[row[0]+"fsis1"] = pInfo[row
[0]+"fsis1"] * 0.6
                                            pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "notall"
                                    for c in listcharacc :
                                            if c not in listchafg :
                                                    c2 = 1
                                    if c2 == 1 :
                                            pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "reunion"
                    if ((pInfo[row[0]+"spouse"] != "") and (pInfo[row[0]+"
spouse"] in listafg) and (pInfo[row[0]+"spouse"] not in listaracc) and (row[0] in
 listaracc)):
                            pInfo[row[0]+"fsis1"] =  pInfo[row[0]+"fsis1"] *
0.7
                            pInfo[row[0]+"sepfromspouse"] = "yes0"
                    elif ((pInfo[row[0]+"spouse"] != "") and (pInfo[row[0]+"
spouse"] not in listafg)):
                            pInfo[row[0]+"sepfromspouse"] = "byarrest"
                    if (int(pInfo[row[0]+"age"]) <= 17):
                            if ((pInfo[row[0]+"father"] == "") or (pInfo[row
[0]+"father"] in listaracc)) and (pInfo[row[0]+"mother"] in listafg) and (pInfo[
row[0]+"mother"] not in listaracc) and (row[0] in listaracc):
                                    pInfo[row[0]+"fsis1"] =  pInfo[row[0]+"
fsis1"] * 0.5
                                    pInfo[row[0]+"sepfromparents"] = "mother"
                            elif ((pInfo[row[0]+"mother"] == "") or (pInfo[row
[0]+"mother"] in listaracc)) and (pInfo[row[0]+"father"] in listafg) and (pInfo[
row[0]+"father"] not in listaracc) and (row[0] in listaracc):
                                    pInfo[row[0]+"fsis1"] =  pInfo[row[0]+"
fsis1"] * 0.6
                                    pInfo[row[0]+"sepfromparents"] = "father"
```

```
                                elif (pInfo[row[0]+"mother"] not in listaracc) and
(pInfo[row[0]+"father"] in listafg) and (pInfo[row[0]+"father"] not in listaracc)
 and (row[0] in listaracc):
                                        pInfo[row[0]+"fsis1"] =  pInfo[row[0]+"
fsis1"] * 0.3
                                        pInfo[row[0]+"sepfromparents"] = "both3"
                                elif (pInfo[row[0]+"father"] not in listaracc) and
(pInfo[row[0]+"mother"] in listafg) and (pInfo[row[0]+"mother"] not in listaracc)
 and (row[0] in listaracc):
                                        pInfo[row[0]+"fsis1"] =  pInfo[row[0]+"
fsis1"] * 0.3
                                        pInfo[row[0]+"sepfromparents"] = "both4"
                        pInfo[row[0]+"fsis2"] = pInfo[row[0]+"fsis1"]
                        pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"]
                # Stg2 (except if det 1 is empty)
                if (len(listadet1) != 0):
                        # Stg2 - 1
                        # Sep bw det1 and det2
                        if (len(listadet2) != 0):
                                if (set(listadet1).issubset(set(listadet2))) and (
row[0] in listadet1):
                                        pInfo[row[0]+"fsis2"] = pInfo[row[0]+"fsis1
"] * 1
                                elif ((len(listadet2) == 1) and (row[0] in
listadet2)) or ((set(listadet1) - set(listadet2)) == set((row[0],))):
                                        pInfo[row[0]+"fsis2"] = pInfo[row[0]+"fsis1
"] * 0.5
                                        if (pInfo[row[0]+"sep1"] != "") and (pInfo[
row[0]+"sep2"] != "") :
                                                pInfo[row[0]+"more"] = "yes"
                                        elif (pInfo[row[0]+"sep1"] != ""):
                                                pInfo[row[0]+"sep2"] = pInfo[row
[0]+"det1"]
                                        else :
                                                pInfo[row[0]+"sep1"] = pInfo[row
[0]+"det1"]
                                else:
                                        pInfo[row[0]+"fsis2"] = pInfo[row[0]+"fsis1
"] * 0.8
                                        if (pInfo[row[0]+"sep1"] != "") and (pInfo[
row[0]+"sep2"] != "") :
                                                pInfo[row[0]+"more"] = "yes"
                                        elif (pInfo[row[0]+"sep1"] != ""):
                                                pInfo[row[0]+"sep2"] = pInfo[row
[0]+"det1"]
                                        else :
                                                pInfo[row[0]+"sep1"] = pInfo[row
[0]+"det1"]
                                if (len(listchildren) != 0):
                                        if (len(listchadet1) != 0) and (((len(
listchadet2) == 0) and (row[0] in listadet2)) or ((set(listadet1) == set(
listadet2)) and (row[0] not in listadet2))):
                                                pInfo[row[0]+"fsis2"] = pInfo[row
[0]+"fsis1"] * 0.5
                                                pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "all"
                                        else :
                                                c1 = 0
                                                c2 = 0
                                                for c in listchadet1 :
                                                        if c not in listchadet2 :
                                                                c1 = 1
                                                if c1 == 1 :
                                                        pInfo[row[0]+"fsis2"] =
pInfo[row[0]+"fsis2"] * 0.6
                                                        pInfo[row[0]+"sepfromkids"]
 = pInfo[row[0]+"sepfromkids"] + "notall"
                                                for c in listchadet2 :
                                                        if c not in listchadet1 :
                                                                c2 = 1
                                                if c2 == 1 :
                                                        pInfo[row[0]+"sepfromkids"]
 = pInfo[row[0]+"sepfromkids"] + "reunion"
```

```
                                if ((pInfo[row[0]+"spouse"] != "") and (pInfo[row
[0]+"spouse"] in listadet1) and (row[0] in listadet1) and (((pInfo[row[0]+"spouse
"] not in listadet2) and (row[0] in listadet2)) or ((pInfo[row[0]+"spouse"] in
listadet2) and (row[0] not in listadet2)))):
                                        pInfo[row[0]+"fsis2"] =  pInfo[row[0]+"
fsis2"] * 0.7
                                        pInfo[row[0]+"sepfromspouse"] = "yes"
                                if (int(pInfo[row[0]+"age"]) <= 17):
                                        if (pInfo[row[0]+"mother"] in listadet1)
and (row[0] in listadet1) and ((((pInfo[row[0]+"father"] == "") or (pInfo[row
[0]+"father"] in listadet2)) and (row[0] in listadet2) and (pInfo[row[0]+"mother
"] not in listadet2)) or (((pInfo[row[0]+"father"] == "") or (pInfo[row[0]+"
father"] not in listadet2)) and (row[0] not in listadet2) and (pInfo[row[0]+"
mother"] in listadet2))):
                                                pInfo[row[0]+"fsis2"] =  pInfo[row
[0]+"fsis2"] * 0.5
                                                pInfo[row[0]+"sepfromparents"] = "
mother"
                                        elif (pInfo[row[0]+"father"] in listadet1)
and (row[0] in listadet1) and ((((pInfo[row[0]+"mother"] == "") or (pInfo[row
[0]+"mother"] in listadet2)) and (row[0] in listadet2) and (pInfo[row[0]+"father
"] not in listadet2)) or (((pInfo[row[0]+"mother"] == "") or (pInfo[row[0]+"
mother"] not in listadet2)) and (row[0] not in listadet2) and (pInfo[row[0]+"
father"] in listadet2))):
                                                pInfo[row[0]+"fsis2"] =  pInfo[row
[0]+"fsis2"] * 0.6
                                                pInfo[row[0]+"sepfromparents"] = "
father"
                                        elif (pInfo[row[0]+"father"] in listadet1)
and (row[0] in listadet1) and (((pInfo[row[0]+"father"] not in listadet2) and (
pInfo[row[0]+"mother"] not in listadet2) and (row[0] in listadet2)) or (pInfo[row
[0]+"father"] in listadet2) and (pInfo[row[0]+"mother"] in listadet2) and (row[0]
 not in listadet2)):
                                                pInfo[row[0]+"fsis2"] =  pInfo[row
[0]+"fsis2"] * 0.3
                                                pInfo[row[0]+"sepfromparents"] = "
both"
                                        elif (pInfo[row[0]+"mother"] in listadet1)
and (row[0] in listadet1) and (((pInfo[row[0]+"father"] not in listadet2) and (
pInfo[row[0]+"mother"] not in listadet2) and (row[0] in listadet2)) or (pInfo[row
[0]+"father"] in listadet2) and (pInfo[row[0]+"mother"] in listadet2) and (row[0]
 not in listadet2)):
                                                pInfo[row[0]+"fsis2"] =  pInfo[row
[0]+"fsis2"] * 0.3
                                                pInfo[row[0]+"sepfromparents"] = "
both"
                                # sep bw det2 and det3
                                if (len(listadet3) != 0):
                                        if (set(listadet2).issubset(set(listadet3))
) and (row[0] in listadet2):
                                                pInfo[row[0]+"fsis2"] = pInfo[row
[0]+"fsis2"] * 1.0
                                        elif ((len(listadet3) == 1) and (row[0] in
listadet3)) or ((set(listadet2) - set(listadet3)) == set((row[0],))):
                                                pInfo[row[0]+"fsis2"] = pInfo[row
[0]+"fsis2"] * 0.5
                                                if (pInfo[row[0]+"sep1"] != "") and
 (pInfo[row[0]+"sep2"] != "") :
                                                        pInfo[row[0]+"more"] = "yes
"
                                                elif (pInfo[row[0]+"sep1"] != ""):
                                                        pInfo[row[0]+"sep2"] =
pInfo[row[0]+"det2"]
                                                else :
                                                        pInfo[row[0]+"sep1"] =
pInfo[row[0]+"det2"]
                                        else:
                                                pInfo[row[0]+"fsis2"] = pInfo[row
[0]+"fsis2"] * 0.8
                                                if (pInfo[row[0]+"sep1"] != "") and
 (pInfo[row[0]+"sep2"] != "") :
                                                        pInfo[row[0]+"more"] = "yes
"
```

```
                                                elif (pInfo[row[0]+"sep1"] != ""):
                                                        pInfo[row[0]+"sep2"] =
pInfo[row[0]+"det2"]
                                                else :
                                                        pInfo[row[0]+"sep1"] =
pInfo[row[0]+"det2"]
                                        if (len(listchildren) != 0):
                                                if (len(listchadet2) != 0)  and (((
len(listchadet3) == 0) and (row[0] in listadet3)) or ((set(listadet2) == set(
listadet3)) and (row[0] not in listadet3))):
                                                        pInfo[row[0]+"fsis2"] =
pInfo[row[0]+"fsis2"] * 0.5
                                                        pInfo[row[0]+"sepfromkids"]
 = pInfo[row[0]+"sepfromkids"] + "all"
                                                else :
                                                        c1 = 0
                                                        c2 = 0
                                                        for c in listchadet2 :
                                                                if c not in
listchadet3 :
                                                                        c1 = 1
                                                        if c1 == 1 :
                                                                pInfo[row[0]+"fsis2
"] = pInfo[row[0]+"fsis2"] * 0.6
                                                                pInfo[row[0]+"
sepfromkids"] = pInfo[row[0]+"sepfromkids"] + "notall"
                                                        for c in listchadet3 :
                                                                if c not in
listchadet2 :
                                                                        c2 = 1
                                                        if c2 == 1 :
                                                                pInfo[row[0]+"
sepfromkids"] = pInfo[row[0]+"sepfromkids"] + "reunion"
                                        if ((pInfo[row[0]+"spouse"] != "") and (
pInfo[row[0]+"spouse"] in listadet2) and (row[0] in listadet2) and (((pInfo[row
[0]+"spouse"] not in listadet3) and (row[0] in listadet3)) or ((pInfo[row[0]+"
spouse"] in listadet3) and (row[0] not in listadet3)))):
                                                pInfo[row[0]+"fsis2"] =  pInfo[row
[0]+"fsis2"] * 0.7
                                                pInfo[row[0]+"sepfromspouse"] = "
yes"
                                        if (int(pInfo[row[0]+"age"]) <= 17):
                                                if (pInfo[row[0]+"mother"] in
listadet2) and (row[0] in listadet2) and ((((pInfo[row[0]+"father"] == "") or (
pInfo[row[0]+"father"] in listadet3)) and (row[0] in listadet3) and (pInfo[row
[0]+"mother"] not in listadet3)) or (((pInfo[row[0]+"father"] == "") or (pInfo[
row[0]+"father"] not in listadet3)) and (row[0] not in listadet3) and (pInfo[row
[0]+"mother"] in listadet3))):
                                                        pInfo[row[0]+"fsis2"] =
pInfo[row[0]+"fsis2"] * 0.5
                                                        pInfo[row[0]+"
sepfromparents"] = "mother"
                                                elif (pInfo[row[0]+"father"] in
listadet2) and (row[0] in listadet2) and ((((pInfo[row[0]+"mother"] == "") or (
pInfo[row[0]+"mother"] in listadet3)) and (row[0] in listadet3) and (pInfo[row
[0]+"father"] not in listadet3)) or (((pInfo[row[0]+"mother"] == "") or (pInfo[
row[0]+"mother"] not in listadet3)) and (row[0] not in listadet3) and (pInfo[row
[0]+"father"] in listadet3))):
                                                        pInfo[row[0]+"fsis2"] =
pInfo[row[0]+"fsis2"] * 0.6
                                                        pInfo[row[0]+"
sepfromparents"] = "father"
                                                elif (pInfo[row[0]+"father"] in
listadet2) and (row[0] in listadet2) and (((pInfo[row[0]+"father"] not in
listadet3) and (pInfo[row[0]+"mother"] not in listadet3) and (row[0] in listadet3
)) or (pInfo[row[0]+"father"] in listadet3) and (pInfo[row[0]+"mother"] in
listadet3) and (row[0] not in listadet3)):
                                                        pInfo[row[0]+"fsis2"] =
pInfo[row[0]+"fsis2"] * 0.3
                                                        pInfo[row[0]+"
sepfromparents"] = "both"
                                                elif (pInfo[row[0]+"mother"] in
listadet2) and (row[0] in listadet2) and (((pInfo[row[0]+"father"] not in
```

```
listadet3) and (pInfo[row[0]+"mother"] not in listadet3) and (row[0] in listadet3
)) or (pInfo[row[0]+"father"] in listadet3) and (pInfo[row[0]+"mother"] in
listadet3) and (row[0] not in listadet3)):
                                                        pInfo[row[0]+"fsis2"] =
pInfo[row[0]+"fsis2"] * 0.3
                                                        pInfo[row[0]+"
sepfromparents"] = "both"
                                        # sep bw det3 and det4
                                         if (len(listadet4) != 0):
                                                if (set(listadet3).issubset(set(
listadet4))) and (row[0] in listadet3):
                                                        pInfo[row[0]+"fsis2"] =
pInfo[row[0]+"fsis2"] * 1.0
                                                elif ((len(listadet4) == 1) and (
row[0] in listadet4)) or ((set(listadet3) - set(listadet4)) == set((row[0],))):
                                                        pInfo[row[0]+"fsis2"] =
pInfo[row[0]+"fsis2"] * 0.5
                                                        if (pInfo[row[0]+"sep1"] !=
 "") and (pInfo[row[0]+"sep2"] != "") :
                                                                pInfo[row[0]+"more
"] = "yes"
                                                        elif (pInfo[row[0]+"sep1"]
!= ""):
                                                                pInfo[row[0]+"sep2
"] = pInfo[row[0]+"det3"]
                                                        else :
                                                                pInfo[row[0]+"sep1
"] = pInfo[row[0]+"det3"]
                                                else:
                                                        pInfo[row[0]+"fsis2"] =
pInfo[row[0]+"fsis2"] * 0.8
                                                        if (pInfo[row[0]+"sep1"] !=
 "") and (pInfo[row[0]+"sep2"] != "") :
                                                                pInfo[row[0]+"more
"] = "yes"
                                                        elif (pInfo[row[0]+"sep1"]
!= ""):
                                                                pInfo[row[0]+"sep2
"] = pInfo[row[0]+"det3"]
                                                        else :
                                                                pInfo[row[0]+"sep1
"] = pInfo[row[0]+"det3"]
                                                if (len(listchildren) != 0):
                                                        if (len(listchadet3) != 0)
 and (((len(listchadet4) == 0) and (row[0] in listadet4)) or ((set(listadet3) ==
set(listadet3)) and (row[0] not in listadet3))):
                                                                pInfo[row[0]+"fsis2
"] = pInfo[row[0]+"fsis2"] * 0.5
                                                                pInfo[row[0]+"
sepfromkids"] = pInfo[row[0]+"sepfromkids"] + "all"
                                                        else :
                                                                c1 = 0
                                                                c2 = 0
                                                                for c in
listchadet3 :
                                                                        if c not in
 listchadet4 :
                                                                                c1
= 1
                                                                if c1 == 1 :
                                                                        pInfo[row
[0]+"fsis2"] = pInfo[row[0]+"fsis2"] * 0.6
                                                                        pInfo[row
[0]+"sepfromkids"] = pInfo[row[0]+"sepfromkids"] + "notall"
                                                                for c in
listchadet4 :
                                                                        if c not in
 listchadet3 :
                                                                                c2
= 1
                                                                if c2 == 1 :
                                                                        pInfo[row
[0]+"sepfromkids"] = pInfo[row[0]+"sepfromkids"] + "reunion"
```

```
                                                    if ((pInfo[row[0]+"spouse"] != "")
and (pInfo[row[0]+"spouse"] in listadet3) and (row[0] in listadet3) and (((pInfo[
row[0]+"spouse"] not in listadet4) and (row[0] in listadet4)) or ((pInfo[row[0]+"
spouse"] in listadet4) and (row[0] not in listadet4)))):
                                                        pInfo[row[0]+"fsis2"] =
pInfo[row[0]+"fsis2"] * 0.7
                                                        pInfo[row[0]+"sepfromspouse
"] = "yes"
                                                    if (int(pInfo[row[0]+"age"]) <= 17)
:
                                                        if (pInfo[row[0]+"mother"]
in listadet3) and (row[0] in listadet3) and ((((pInfo[row[0]+"father"] == "") or
(pInfo[row[0]+"father"] in listadet4)) and (row[0] in listadet4) and (pInfo[row
[0]+"mother"] not in listadet4)) or (((pInfo[row[0]+"father"] == "") or (pInfo[
row[0]+"father"] not in listadet4)) and (row[0] not in listadet4) and (pInfo[row
[0]+"mother"] in listadet4))):
                                                            pInfo[row[0]+"fsis2
"] =  pInfo[row[0]+"fsis2"] * 0.5
                                                            pInfo[row[0]+"
sepfromparents"] = "mother"
                                                        elif (pInfo[row[0]+"father
"] in listadet3) and (row[0] in listadet3) and ((((pInfo[row[0]+"mother"] == "")
or (pInfo[row[0]+"mother"] in listadet4)) and (row[0] in listadet4) and (pInfo[
row[0]+"father"] not in listadet4)) or (((pInfo[row[0]+"mother"] == "") or (pInfo
[row[0]+"mother"] not in listadet4)) and (row[0] not in listadet4) and (pInfo[row
[0]+"father"] in listadet4))):
                                                            pInfo[row[0]+"fsis2
"] =  pInfo[row[0]+"fsis2"] * 0.6
                                                            pInfo[row[0]+"
sepfromparents"] = "father"
                                                        elif (pInfo[row[0]+"father
"] in listadet3) and (row[0] in listadet3) and (((pInfo[row[0]+"father"] not in
listadet4) and (pInfo[row[0]+"mother"] not in listadet4) and (row[0] in listadet4
)) or (pInfo[row[0]+"father"] in listadet4) and (pInfo[row[0]+"mother"] in
listadet4) and (row[0] not in listadet4)):
                                                            pInfo[row[0]+"fsis2
"] =  pInfo[row[0]+"fsis2"] * 0.3
                                                            pInfo[row[0]+"
sepfromparents"] = "both"
                                                        elif (pInfo[row[0]+"mother
"] in listadet3) and (row[0] in listadet3) and (((pInfo[row[0]+"father"] not in
listadet4) and (pInfo[row[0]+"mother"] not in listadet4) and (row[0] in listadet4
)) or (pInfo[row[0]+"father"] in listadet4) and (pInfo[row[0]+"mother"] in
listadet4) and (row[0] not in listadet4)):
                                                            pInfo[row[0]+"fsis2
"] =  pInfo[row[0]+"fsis2"] * 0.3
                                                            pInfo[row[0]+"
sepfromparents"] = "both"
                                                    # sep bw det4 and det5
                                                    if (len(listadet5) != 0):
                                                        if (set(listadet4).issubset
(set(listadet5))) and (row[0] in listadet4):
                                                            pInfo[row[0]+"fsis2
"] = pInfo[row[0]+"fsis2"] * 1.0
                                                        elif ((len(listadet5) == 1)
 and (row[0] in listadet5)) or ((set(listadet4) - set(listadet5)) == set((row
[0],))):
                                                            pInfo[row[0]+"fsis2
"] = pInfo[row[0]+"fsis2"] * 0.5
                                                            if (pInfo[row[0]+"
sep1"] != "") and (pInfo[row[0]+"sep2"] != "") :
                                                                pInfo[row
[0]+"more"] = "yes"
                                                            elif (pInfo[row
[0]+"sep1"] != ""):
                                                                pInfo[row
[0]+"sep2"] = pInfo[row[0]+"det4"]
                                                            else :
                                                                pInfo[row
[0]+"sep1"] = pInfo[row[0]+"det4"]
                                                        else:
                                                            pInfo[row[0]+"fsis2
"] = pInfo[row[0]+"fsis2"] * 0.8
```

```
                                                                    if (pInfo[row[0]+"
sep1"] != "") and (pInfo[row[0]+"sep2"] != "") :
                                                                        pInfo[row
[0]+"more"] = "yes"
                                                                    elif (pInfo[row
[0]+"sep1"] != ""):
                                                                        pInfo[row
[0]+"sep2"] = pInfo[row[0]+"det4"]
                                                                    else :
                                                                        pInfo[row
[0]+"sep1"] = pInfo[row[0]+"det4"]
                                                                if (len(listchildren) != 0)
:
                                                                    if (len(listchadet4
) != 0) and (((len(listchadet5) == 0) and (row[0] in listadet5)) or ((set(
listadet4) == set(listadet5)) and (row[0] not in listadet5))):
                                                                        pInfo[row
[0]+"fsis2"] = pInfo[row[0]+"fsis2"] * 0.5
                                                                        pInfo[row
[0]+"sepfromkids"] = pInfo[row[0]+"sepfromkids"] + "all"
                                                                    else :
                                                                        c1 = 0
                                                                        c2 = 0
                                                                        for c in
listchadet4 :
                                                                            if
c not in listchadet5 :

        c1 = 1
                                                                        if c1 == 1
:

pInfo[row[0]+"fsis2"] = pInfo[row[0]+"fsis2"] * 0.6

pInfo[row[0]+"sepfromkids"] = pInfo[row[0]+"sepfromkids"] + "notall"
                                                                        for c in
listchadet5 :
                                                                            if
c not in listchadet4 :

        c2 = 1
                                                                        if c2 == 1
:

pInfo[row[0]+"sepfromkids"] = pInfo[row[0]+"sepfromkids"] + "reunion"
                                                                        #
outstring = "stage 4 *8  listchadet4 : %s , listchadet5 %s" %(', '.join(
listchadet4),', '.join(listchadet5))
                                                                        #
arcpy.AddMessage(outstring)
                                                                        #
del outstring
                                                                if ((pInfo[row[0]+"spouse"]
 != "") and (pInfo[row[0]+"spouse"] in listadet4) and (row[0] in listadet4) and
(((pInfo[row[0]+"spouse"] not in listadet5) and (row[0] in listadet5)) or ((pInfo
[row[0]+"spouse"] in listadet5) and (row[0] not in listadet5)))):
                                                                    pInfo[row[0]+"fsis2
"] =  pInfo[row[0]+"fsis2"] * 0.7
                                                                    pInfo[row[0]+"
sepfromspouse"] = "yes"
                                                                if (int(pInfo[row[0]+"age
"]) <= 17):
                                                                    if (pInfo[row[0]+"
mother"] in listadet4) and (row[0] in listadet4) and ((((pInfo[row[0]+"father"]
== "") or (pInfo[row[0]+"father"] in listadet5)) and (row[0] in listadet5) and (
pInfo[row[0]+"mother"] not in listadet5)) or (((pInfo[row[0]+"father"] == "") or
(pInfo[row[0]+"father"] not in listadet5)) and (row[0] not in listadet5) and (
pInfo[row[0]+"mother"] in listadet5))):
                                                                        pInfo[row
[0]+"fsis2"] =  pInfo[row[0]+"fsis2"] * 0.5
                                                                        pInfo[row
[0]+"sepfromparents"] = "mother"
                                                                    elif (pInfo[row
```

```
[0]+"father"] in listadet4) and (row[0] in listadet4) and ((((pInfo[row[0]+"
mother"] == "") or (pInfo[row[0]+"mother"] in listadet5)) and (row[0] in
listadet5) and (pInfo[row[0]+"father"] not in listadet5)) or (((pInfo[row[0]+"
mother"] == "") or (pInfo[row[0]+"mother"] not in listadet5)) and (row[0] not in
listadet5) and (pInfo[row[0]+"father"] in listadet5))):
                                                                    pInfo[row
[0]+"fsis2"] =  pInfo[row[0]+"fsis2"] * 0.6
                                                                    pInfo[row
[0]+"sepfromparents"] = "father"
                                                                elif (pInfo[row
[0]+"father"] in listadet4) and (row[0] in listadet4) and (((pInfo[row[0]+"father
"] not in listadet5) and (pInfo[row[0]+"mother"] not in listadet5) and (row[0] in
 listadet5)) or (pInfo[row[0]+"father"] in listadet5) and (pInfo[row[0]+"mother"]
 in listadet5) and (row[0] not in listadet5)):
                                                                    pInfo[row
[0]+"fsis2"] =  pInfo[row[0]+"fsis2"] * 0.3
                                                                    pInfo[row
[0]+"sepfromparents"] = "both"
                                                                elif (pInfo[row
[0]+"mother"] in listadet4) and (row[0] in listadet4) and (((pInfo[row[0]+"father
"] not in listadet5) and (pInfo[row[0]+"mother"] not in listadet2) and (row[0] in
 listadet5)) or (pInfo[row[0]+"father"] in listadet5) and (pInfo[row[0]+"mother"]
 in listadet5) and (row[0] not in listadet5)):
                                                                    pInfo[row
[0]+"fsis2"] =  pInfo[row[0]+"fsis2"] * 0.3
                                                                    pInfo[row
[0]+"sepfromparents"] = "both"
                    # Stg 2 -2
                    # sep bw det 1 and raccolta  if d2 do not exist and no
stage 3
                     elif (len(listadet2) == 0):
                            if (set(listadet1).issubset(set(listaracc))) or (
row[0] not in listadet1) :
                                    pInfo[row[0]+"fsis2"] = pInfo[row[0]+"fsis1
"] * 1.0
                            elif (len(listaracc) == 1):
                                    pInfo[row[0]+"fsis2"] = pInfo[row[0]+"fsis1
"] * 0.5
                                    pInfo[row[0]+"sep1"] = pInfo[row[0]+"arrest
"]
                            else:
                                    pInfo[row[0]+"fsis2"] = pInfo[row[0]+"fsis1
"] * 0.8
                                    pInfo[row[0]+"sep1"] = pInfo[row[0]+"arrest
"]
                            if (len(listchildren) != 0):
                                    if (len(listchadet1) != 0) and (len(
listcharacc) == 0) and (row[0] in listadet1):
                                            pInfo[row[0]+"fsis2"] = pInfo[row
[0]+"fsis2"] * 0.5
                                            pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "all"
                                    else :
                                            c1 = 0
                                            c2 = 0
                                            for c in listchadet1 :
                                                    if c not in listcharacc :
                                                            c1 = 1
                                            if c1 == 1 :
                                                    pInfo[row[0]+"fsis2"] =
pInfo[row[0]+"fsis2"] * 0.6
                                                    pInfo[row[0]+"sepfromkids"]
 = pInfo[row[0]+"sepfromkids"] + "notall"
                                            for c in listcharacc :
                                                    if c not in listchadet1 :
                                                            c2 = 1
                                            if c2 == 1 :
                                                    pInfo[row[0]+"sepfromkids"]
 = pInfo[row[0]+"sepfromkids"] + "reunion"
                                                    outstring = "stage 4 *8
listcharacc : %s , listchadet1 %s" %(', '.join(listcharacc),', '.join(listchadet1
))
                                                    arcpy.AddMessage(outstring)
```

```
                                                del outstring
                                if ((pInfo[row[0]+"spouse"] != "") and (pInfo[row
[0]+"spouse"] in listadet1) and (row[0] in listadet1) and (pInfo[row[0]+"spouse"]
 not in listaracc)):
                                        pInfo[row[0]+"fsis2"] =  pInfo[row[0]+"
fsis2"] * 0.7
                                        pInfo[row[0]+"sepfromspouse"] = "yes"
                                if (int(pInfo[row[0]+"age"]) <= 17):
                                        if ((pInfo[row[0]+"father"] == "") or (
pInfo[row[0]+"father"] in listaracc)) and (pInfo[row[0]+"mother"] in listadet1)
and (row[0] in listadet1) and (pInfo[row[0]+"mother"] not in listaracc):
                                                pInfo[row[0]+"fsis2"] =  pInfo[row
[0]+"fsis2"] * 0.5
                                                pInfo[row[0]+"sepfromparents"] = "
mother"
                                        elif ((pInfo[row[0]+"mother"] == "") or (
pInfo[row[0]+"mother"] in listaracc)) and (pInfo[row[0]+"father"] in listadet1)
and (row[0] in listadet1) and (pInfo[row[0]+"father"] not in listaracc):
                                                pInfo[row[0]+"fsis2"] =  pInfo[row
[0]+"fsis2"] * 0.6
                                                pInfo[row[0]+"sepfromparents"] = "
father"
                                        elif (pInfo[row[0]+"mother"] not in
listaracc) and (pInfo[row[0]+"father"] in listadet1) and (row[0] in listadet1)
and (pInfo[row[0]+"father"] not in listaracc):
                                                pInfo[row[0]+"fsis2"] =  pInfo[row
[0]+"fsis2"] * 0.3
                                                pInfo[row[0]+"sepfromparents"] = "
both"
                                        elif (pInfo[row[0]+"father"] not in
listaracc) and (pInfo[row[0]+"mother"] in listadet1) and (row[0] in listadet1)
and (pInfo[row[0]+"mother"] not in listaracc):
                                                pInfo[row[0]+"fsis2"] =  pInfo[row
[0]+"fsis2"] * 0.3
                                                pInfo[row[0]+"sepfromparents"] = "
both"
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"]
                # Stg 3
                # sep bw raccolta and d2, d3, d4 ou d5
                 if (len(listadet5) != 0):
                        if ((set(listadet5).issubset(set(listaracc))) and (row[0]
in listadet5)) or (row[0] not in listadet5):
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"] * 1.0
                        elif (len(listaracc) == 1):
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"] * 0.5
                                if (pInfo[row[0]+"sep1"] != "") and (pInfo[row[0]+"
sep2"] != "") :
                                        pInfo[row[0]+"more"] = "yes"
                                elif (pInfo[row[0]+"sep1"] != ""):
                                        pInfo[row[0]+"sep2"] = pInfo[row[0]+"det5"]
                                else :
                                        pInfo[row[0]+"sep1"] = pInfo[row[0]+"det5"]
                        else:
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"] * 0.8
                                if (pInfo[row[0]+"sep1"] != "") and (pInfo[row[0]+"
sep2"] != "") :
                                        pInfo[row[0]+"more"] = "yes"
                                elif (pInfo[row[0]+"sep1"] != ""):
                                        pInfo[row[0]+"sep2"] = pInfo[row[0]+"det5"]
                                else :
                                        pInfo[row[0]+"sep1"] = pInfo[row[0]+"det5"]
                        if (len(listchildren) != 0):
                                if (len(listchadet5) != 0) and (len(listcharacc) ==
 0) and (row[0] in listadet5):
                                        pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis3
"] * 0.5
                                        pInfo[row[0]+"sepfromkids"] = pInfo[row
[0]+"sepfromkids"] + "all"
                                else :
                                        c1 = 0
                                        c2 = 0
                                        for c in listchadet5 :
                                                if c not in listcharacc :
```

```
                                                c1 = 1
                                        if c1 == 1 :
                                                pInfo[row[0]+"fsis3"] = pInfo[row
[0]+"fsis3"] * 0.6
                                                pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "notall"
                                        for c in listcharacc :
                                                if (c not in listchadet5) or (c not
 in listchadet4) or (c not in listchadet3) or (c not in listchadet2) or (c not in
 listchadet1) :
                                                        c2 = 1
                                        if c2 == 1 :
                                                pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "reunion"
                        if ((pInfo[row[0]+"spouse"] != "") and (pInfo[row[0]+"
spouse"] in listadet5) and (row[0] in listadet5) and (pInfo[row[0]+"spouse"] not
in listaracc)):
                                pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"fsis3"] *
0.7
                                pInfo[row[0]+"sepfromspouse"] = "yes"
                        if (int(pInfo[row[0]+"age"]) <= 17):
                                if ((pInfo[row[0]+"father"] == "") or (pInfo[row
[0]+"father"] in listaracc)) and (pInfo[row[0]+"mother"] in listadet5) and (row
[0] in listadet5) and (pInfo[row[0]+"mother"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.5
                                        pInfo[row[0]+"sepfromparents"] = "mother"
                                elif ((pInfo[row[0]+"mother"] == "") or (pInfo[row
[0]+"mother"] in listaracc)) and (pInfo[row[0]+"father"] in listadet5) and (row
[0] in listadet5) and (pInfo[row[0]+"father"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.6
                                        pInfo[row[0]+"sepfromparents"] = "father"
                                elif (pInfo[row[0]+"mother"] not in listaracc) and
(pInfo[row[0]+"father"] in listadet5) and (row[0] in listadet5) and (pInfo[row
[0]+"father"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.3
                                        pInfo[row[0]+"sepfromparents"] = "both"
                                elif (pInfo[row[0]+"father"] not in listaracc) and
(pInfo[row[0]+"mother"] in listadet5) and (row[0] in listadet5) and (pInfo[row
[0]+"mother"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.3
                                        pInfo[row[0]+"sepfromparents"] = "both"
                elif (len(listadet4) != 0):
                        if ((set(listadet4).issubset(set(listaracc))) and (row[0]
in listadet4)) or (row[0] not in listadet4):
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"] * 1.0
                        elif (len(listaracc) == 1):
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"] * 0.5
                                if (pInfo[row[0]+"sep1"] != "") and (pInfo[row[0]+"
sep2"] != "") :
                                        pInfo[row[0]+"more"] = "yes"
                                elif (pInfo[row[0]+"sep1"] != ""):
                                        pInfo[row[0]+"sep2"] = pInfo[row[0]+"det4"]
                                else :
                                        pInfo[row[0]+"sep1"] = pInfo[row[0]+"det4"]
                        else:
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"] * 0.8
                                if (pInfo[row[0]+"sep1"] != "") and (pInfo[row[0]+"
sep2"] != "") :
                                        pInfo[row[0]+"more"] = "yes"
                                elif (pInfo[row[0]+"sep1"] != ""):
                                        pInfo[row[0]+"sep2"] = pInfo[row[0]+"det4"]
                                else :
                                        pInfo[row[0]+"sep1"] = pInfo[row[0]+"det4"]
                        if (len(listchildren) != 0):
                                if (len(listchadet4) != 0) and (len(listcharacc) ==
 0) and (row[0] in listadet4):
                                        pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis3
"] * 0.5
                                        pInfo[row[0]+"sepfromkids"] = pInfo[row
```

```
[0]+"sepfromkids"] + "all"
                                else :
                                        c1 = 0
                                        c2 = 0
                                        for c in listchadet4 :
                                                if c not in listcharacc :
                                                        c1 = 1
                                        if c1 == 1 :
                                                pInfo[row[0]+"fsis3"] = pInfo[row
[0]+"fsis3"] * 0.6
                                                pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "notall"
                                        for c in listcharacc :
                                                if (c not in listchadet4) or(c not
in listchadet3) or (c not in listchadet2) or (c not in listchadet1) :
                                                        c2 = 1
                                        if c2 == 1 :
                                                pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "reunion"
                        if ((pInfo[row[0]+"spouse"] != "") and (pInfo[row[0]+"
spouse"] in listadet4) and (row[0] in listadet4) and (pInfo[row[0]+"spouse"] not
in listaracc)):
                                pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"fsis3"] *
0.7
                                pInfo[row[0]+"sepfromspouse"] = "yes"
                        if (int(pInfo[row[0]+"age"]) <= 17):
                                if ((pInfo[row[0]+"father"] == "") or (pInfo[row
[0]+"father"] in listaracc)) and (pInfo[row[0]+"mother"] in listadet4) and (row
[0] in listadet4) and (pInfo[row[0]+"mother"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.5
                                        pInfo[row[0]+"sepfromparents"] = "mother"
                                elif ((pInfo[row[0]+"mother"] == "") or (pInfo[row
[0]+"mother"] in listaracc)) and (pInfo[row[0]+"father"] in listadet4) and (row
[0] in listadet4) and (pInfo[row[0]+"father"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.6
                                        pInfo[row[0]+"sepfromparents"] = "father"
                                elif (pInfo[row[0]+"mother"] not in listaracc) and
(pInfo[row[0]+"father"] in listadet4) and (row[0] in listadet4) and (pInfo[row
[0]+"father"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.3
                                        pInfo[row[0]+"sepfromparents"] = "both"
                                elif (pInfo[row[0]+"father"] not in listaracc) and
(pInfo[row[0]+"mother"] in listadet4) and (row[0] in listadet4) and (pInfo[row
[0]+"mother"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.3
                                        pInfo[row[0]+"sepfromparents"] = "both"
                elif (len(listadet3) != 0):
                        if ((set(listadet3).issubset(set(listaracc))) and (row[0]
in listadet3)) or (row[0] not in listadet3) :
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"] * 1.0
                        elif (len(listaracc) == 1):
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"] * 0.5
                                if (pInfo[row[0]+"sep1"] != "") and (pInfo[row[0]+"
sep2"] != "") :
                                        pInfo[row[0]+"more"] = "yes"
                                elif (pInfo[row[0]+"sep1"] != ""):
                                        pInfo[row[0]+"sep2"] = pInfo[row[0]+"det3"]
                                else :
                                        pInfo[row[0]+"sep1"] = pInfo[row[0]+"det3"]
                        else:
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"] * 0.8
                                if (pInfo[row[0]+"sep1"] != "") and (pInfo[row[0]+"
sep2"] != "") :
                                        pInfo[row[0]+"more"] = "yes"
                                elif (pInfo[row[0]+"sep1"] != ""):
                                        pInfo[row[0]+"sep2"] = pInfo[row[0]+"det3"]
                                else :
                                        pInfo[row[0]+"sep1"] = pInfo[row[0]+"det3"]
                        if (len(listchildren) != 0):
```

```
                                if (len(listchadet3) != 0) and (len(listcharacc) ==
 0) and (row[0] in listadet3):
                                        pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis3
"] * 0.5
                                        pInfo[row[0]+"sepfromkids"] = pInfo[row
[0]+"sepfromkids"] + "all"
                                else :
                                        c1 = 0
                                        c2 = 0
                                        for c in listchadet3 :
                                                if c not in listcharacc :
                                                        c1 = 1
                                                        outstring = "  c1"
                                                        arcpy.AddMessage(outstring)
                                                        del outstring
                                        if c1 == 1 :
                                                pInfo[row[0]+"fsis3"] = pInfo[row
[0]+"fsis3"] * 0.6
                                                pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "notall"

                                        for c in listcharacc :
                                                if (c not in listchadet3) or (c not
 in listchadet2) or (c not in listchadet1) :
                                                        c2 = 1
                                        if c2 == 1 :
                                                pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "reunion"
                        if ((pInfo[row[0]+"spouse"] != "") and (pInfo[row[0]+"
spouse"] in listadet3) and (row[0] in listadet3) and (pInfo[row[0]+"spouse"] not
in listaracc)):
                                pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"fsis3"] *
0.7
                                pInfo[row[0]+"sepfromspouse"] = "yes"
                        if (int(pInfo[row[0]+"age"]) <= 17):
                                if ((pInfo[row[0]+"father"] == "") or (pInfo[row
[0]+"father"] in listaracc)) and (row[0] in listadet3) and (pInfo[row[0]+"mother
"] in listadet3) and (pInfo[row[0]+"mother"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.5
                                        pInfo[row[0]+"sepfromparents"] = "mother"
                                elif ((pInfo[row[0]+"mother"] == "") or (pInfo[row
[0]+"mother"] in listaracc)) and (row[0] in listadet3) and (pInfo[row[0]+"father
"] in listadet3) and (pInfo[row[0]+"father"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.6
                                        pInfo[row[0]+"sepfromparents"] = "father"
                                elif (pInfo[row[0]+"mother"] not in listaracc) and
(pInfo[row[0]+"father"] in listadet3) and (row[0] in listadet3) and (pInfo[row
[0]+"father"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.3
                                        pInfo[row[0]+"sepfromparents"] = "both"
                                elif (pInfo[row[0]+"father"] not in listaracc) and
(pInfo[row[0]+"mother"] in listadet3) and (row[0] in listadet3) and (pInfo[row
[0]+"mother"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.3
                                        pInfo[row[0]+"sepfromparents"] = "both"
                elif (len(listadet2) != 0):
                        if ((set(listadet2).issubset(set(listaracc))) and (row[0]
in listadet2)) or (row[0] not in listadet2) :
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"] * 1.0
                        elif (len(listaracc) == 1):
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"] * 0.5
                                if (pInfo[row[0]+"sep1"] != "") and (pInfo[row[0]+"
sep2"] != "") :
                                        pInfo[row[0]+"more"] = "yes"
                                elif (pInfo[row[0]+"sep1"] != ""):
                                        pInfo[row[0]+"sep2"] = pInfo[row[0]+"det2"]
                                else :
                                        pInfo[row[0]+"sep1"] = pInfo[row[0]+"det2"]
                        else:
```

```
                                pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis2"] * 0.8
                                if (pInfo[row[0]+"sep1"] != "") and (pInfo[row[0]+"
sep2"] != "") :
                                        pInfo[row[0]+"more"] = "yes"
                                elif (pInfo[row[0]+"sep1"] != ""):
                                        pInfo[row[0]+"sep2"] = pInfo[row[0]+"det2"]
                                else :
                                        pInfo[row[0]+"sep1"] = pInfo[row[0]+"det2"]
                        if (len(listchildren) != 0):
                                if (len(listchadet2) != 0) and (len(listcharacc) ==
 0) and (row[0] in listadet2):
                                        pInfo[row[0]+"fsis3"] = pInfo[row[0]+"fsis3
"] * 0.5
                                        pInfo[row[0]+"sepfromkids"] = pInfo[row
[0]+"sepfromkids"] + "all"
                                else :
                                        c1 = 0
                                        c2 = 0
                                        for c in listchadet2 :
                                                if c not in listcharacc :
                                                        c1 = 1
                                        if c1 == 1 :
                                                pInfo[row[0]+"fsis3"] = pInfo[row
[0]+"fsis3"] * 0.6
                                                pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "notall"
                                        for c in listcharacc :
                                                if (c not in listchadet2) or (c not
 in listchadet1) :
                                                        c2 = 1
                                        if c2 == 1 :
                                                pInfo[row[0]+"sepfromkids"] = pInfo
[row[0]+"sepfromkids"] + "reunion"
                        outstring = "  listcharacc : %s | listchadet2 %s" %(', '.
join(listcharacc),', '.join(listchadet2))
                        arcpy.AddMessage(outstring)
                        del outstring
                        if ((pInfo[row[0]+"spouse"] != "") and (pInfo[row[0]+"
spouse"] in listadet2) and (row[0] in listadet2) and (pInfo[row[0]+"spouse"] not
in listaracc)):
                                pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"fsis3"] *
0.7
                                pInfo[row[0]+"sepfromspouse"] = "yes"
                        if (int(pInfo[row[0]+"age"]) <= 17):
                                if ((pInfo[row[0]+"father"] == "") or (pInfo[row
[0]+"father"] in listaracc)) and (pInfo[row[0]+"mother"] in listadet2) and (row
[0] in listadet2) and (pInfo[row[0]+"mother"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.5
                                        pInfo[row[0]+"sepfromparents"] = "mother"
                                elif ((pInfo[row[0]+"mother"] == "") or (pInfo[row
[0]+"mother"] in listaracc)) and (pInfo[row[0]+"father"] in listadet2) and (row
[0] in listadet2) and (pInfo[row[0]+"father"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.6
                                        pInfo[row[0]+"sepfromparents"] = "father"
                                elif (pInfo[row[0]+"mother"] not in listaracc) and
(pInfo[row[0]+"father"] in listadet2) and (row[0] in listadet2) and (pInfo[row
[0]+"father"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.3
                                        pInfo[row[0]+"sepfromparents"] = "both"
                                elif (pInfo[row[0]+"father"] not in listaracc) and
(pInfo[row[0]+"mother"] in listadet2) and (row[0] in listadet2) and (pInfo[row
[0]+"mother"] not in listaracc):
                                        pInfo[row[0]+"fsis3"] =  pInfo[row[0]+"
fsis3"] * 0.3
                                        pInfo[row[0]+"sepfromparents"] = "both"
                # Stg 4
                # sep bw raccolta and convoy
                if (set(listaracc).issubset(set(listaconvoy))):
                        pInfo[row[0]+"fsis4"] = pInfo[row[0]+"fsis3"] * 1
                elif (len(listaconvoy) == 1):
```

```
                        pInfo[row[0]+"fsis4"] =  pInfo[row[0]+"fsis3"] * 0.5
                        if (pInfo[row[0]+"sep1"] != "") and (pInfo[row[0]+"sep2"]
!= "") :
                                pInfo[row[0]+"more"] = "yes"
                        elif (pInfo[row[0]+"sep1"] != ""):
                                pInfo[row[0]+"sep2"] = pInfo[row[0]+"racc"]
                        else :
                                pInfo[row[0]+"sep1"] = pInfo[row[0]+"racc"]
                else:
                        pInfo[row[0]+"fsis4"] =  pInfo[row[0]+"fsis3"] * 0.8
                        if (pInfo[row[0]+"sep1"] != "") and (pInfo[row[0]+"sep2"]
!= "") :
                                pInfo[row[0]+"more"] = "yes"
                        elif (pInfo[row[0]+"sep1"] != ""):
                                pInfo[row[0]+"sep2"] = pInfo[row[0]+"racc"]
                        else :
                                pInfo[row[0]+"sep1"] = pInfo[row[0]+"racc"]
                if (len(listchildren) != 0):
                        if (len(listcharacc) != 0) and (len(listchaconvoy) == 0):
                                pInfo[row[0]+"fsis4"] = pInfo[row[0]+"fsis4"] * 0.5
                                pInfo[row[0]+"sepfromkids"] = pInfo[row[0]+"
sepfromkids"] + "all"
                        elif (set(listcharacc) == set(listchaconvoy)) and (pInfo[
row[0]+"sepfromkids"] == "") :
                                pInfo[row[0]+"sepfromkids"] = "no"
                        else :
                                c1 = 0
                                c2 = 0
                                for c in listcharacc :
                                        if c not in listchaconvoy :
                                                c1 = 1
                                if c1 == 1 :
                                        pInfo[row[0]+"fsis4"] = pInfo[row[0]+"fsis4
"] * 0.6
                                        pInfo[row[0]+"sepfromkids"] = pInfo[row
[0]+"sepfromkids"] + "notall"
                                for c in listchaconvoy :
                                        if c not in listcharacc :
                                                c2 = 1
                                if c2 == 1 :
                                        pInfo[row[0]+"sepfromkids"] = pInfo[row
[0]+"sepfromkids"] + "reunion"
                if ((pInfo[row[0]+"spouse"] != "") and (pInfo[row[0]+"spouse"] in
listaracc) and (pInfo[row[0]+"spouse"] not in listaconvoy)):
                        pInfo[row[0]+"fsis4"] =  pInfo[row[0]+"fsis4"] * 0.7
                        pInfo[row[0]+"sepfromspouse"] = "yes"
                elif ((pInfo[row[0]+"spouse"] != "") and (pInfo[row[0]+"spouse"] in
 listaracc) and (pInfo[row[0]+"sepfromspouse"] == "")):
                        pInfo[row[0]+"sepfromspouse"] = "no"
                elif ((pInfo[row[0]+"spouse"] != "") and (pInfo[row[0]+"spouse"] in
 listaracc) and (pInfo[row[0]+"sepfromspouse"] != "")):
                        pInfo[row[0]+"sepfromspouse"] = pInfo[row[0]+"sepfromspouse
"] + "butreunion"
                if (int(pInfo[row[0]+"age"]) <= 17):
                        if ((pInfo[row[0]+"father"] == "") or (pInfo[row[0]+"father
"] in listaconvoy)) and (pInfo[row[0]+"mother"] in listaracc) and (pInfo[row[0]+"
mother"] not in listaconvoy):
                                pInfo[row[0]+"fsis4"] =  pInfo[row[0]+"fsis4"] *
0.5
                                pInfo[row[0]+"sepfromparents"] = "mother"
                        elif ((pInfo[row[0]+"mother"] == "") or (pInfo[row[0]+"
mother"] in listaconvoy)) and (pInfo[row[0]+"father"] in listaracc) and (pInfo[
row[0]+"father"] not in listaconvoy):
                                pInfo[row[0]+"fsis4"] =  pInfo[row[0]+"fsis4"] *
0.6
                                pInfo[row[0]+"sepfromparents"] = "father"
                        elif (pInfo[row[0]+"mother"] not in listaconvoy) and (pInfo
[row[0]+"father"] in listaracc) and (pInfo[row[0]+"father"] not in listaconvoy):
                                pInfo[row[0]+"fsis4"] =  pInfo[row[0]+"fsis4"] *
0.3
                                pInfo[row[0]+"sepfromparents"] = "both"
                        elif (pInfo[row[0]+"father"] not in listaconvoy) and (pInfo
[row[0]+"mother"] in listaracc) and (pInfo[row[0]+"mother"] not in listaconvoy):
```

```
                                pInfo[row[0]+"fsis4"] =  pInfo[row[0]+"fsis4"] *
  0.3
                                pInfo[row[0]+"sepfromparents"] = "both"
                        elif (pInfo[row[0]+"sepfromparents"] == "") :
                                pInfo[row[0]+"sepfromparents"] = "no"
                        elif ((pInfo[row[0]+"mother"] in listaconvoy) and (pInfo[
  row[0]+"father"] in listaconvoy)) and (pInfo[row[0]+"sepfromparents"] == "both"):
                                pInfo[row[0]+"sepfromparents"] = pInfo[row[0]+"
  sepfromparents"] + "butreunionboth"
                        elif (pInfo[row[0]+"mother"] in listaconvoy) and ((pInfo[
  row[0]+"sepfromparents"] == "both") or (pInfo[row[0]+"sepfromparents"] == "mother
  ")) :
                                pInfo[row[0]+"sepfromparents"] = pInfo[row[0]+"
  sepfromparents"] + "butreunionmother"
                        elif (pInfo[row[0]+"father"] in listaconvoy) and ((pInfo[
  row[0]+"sepfromparents"] == "both") or (pInfo[row[0]+"sepfromparents"] == "father
  ")) :
                                pInfo[row[0]+"sepfromparents"] = pInfo[row[0]+"
  sepfromparents"] + "butreunionfather"

 #              if (len(listaconvoy) == len(listalagger)):
 #                      pInfo[row[0]+"fsis5"] = pInfo[row[0]+"fsis4"] * 1
 #              elif (len(listalagger) == 1):
 #                      pInfo[row[0]+"fsis5"] =  pInfo[row[0]+"fsis4"] * 0.5
 #              else:
 #                      pInfo[row[0]+"fsis5"] =  pInfo[row[0]+"fsis4"] * 0.8
                rowIndex += 1
                del listafg
                del listadet1
                del listadet5
                del listaracc
                del listaconvoy
                del listalagger
                del listchildren
                del listfam

#
# End row loop
#
rowIndex = colIndex= 0
arcpy.AddMessage("Finished Assigning FSIS...")

#-------------------------------------------------------------------------------

# write outputs
#-------------------------------------------------------------------------------



outfile1.write("ID,FSIS1,FSIS2,FSIS3,FSIS4,VicNat,VicGen,VicAge,Perp1,Perp2,AnArr,
  Fate,AFG,SepFromSpouse,SepFromChildren,SepFromParents,reunion,sep1,sep2,moresep")
outfile1.write("\n")
for y in range (0,num_records):
        outfile1.write(idArray[y]+",")
        outfile1.write(str(float(pInfo[idArray[y]+"fsis1"]))+",")
        outfile1.write(str(float(pInfo[idArray[y]+"fsis2"]))+",")
        outfile1.write(str(float(pInfo[idArray[y]+"fsis3"]))+",")
        outfile1.write(str(float(pInfo[idArray[y]+"fsis4"]))+",")
 #      outfile1.write(str(float(pInfo[idArray[y]+"fsis5"]))+",")
        outfile1.write(pInfo[idArray[y]+"nat"]+",")
        outfile1.write(pInfo[idArray[y]+"gender"]+",")
        outfile1.write(pInfo[idArray[y]+"age"]+",")
        outfile1.write(pInfo[idArray[y]+"perp1"]+",")
        outfile1.write(pInfo[idArray[y]+"perp2"]+",")
        outfile1.write(pInfo[idArray[y]+"anar"]+",")
        outfile1.write(pInfo[idArray[y]+"fate"]+",")
        outfile1.write(pInfo[idArray[y]+"afgid"]+",")
        outfile1.write(pInfo[idArray[y]+"sepfromspouse"]+",")
        outfile1.write(pInfo[idArray[y]+"sepfromkids"]+",")
        outfile1.write(pInfo[idArray[y]+"sepfromparents"]+",")
        outfile1.write(pInfo[idArray[y]+"reunion"]+",")
        outfile1.write(pInfo[idArray[y]+"sep1"]+",")
        outfile1.write(pInfo[idArray[y]+"sep2"]+",")
```

```
        outfile1.write(pInfo[idArray[y]+"more"]+",")
        outfile1.write("\n")
outfile1.close()


outfile2.write("ID,Sep,FSIS,DeportationStage,VicNat,VicGen,VicAge,Perp1,Perp2,AnArr
  ,Fate,kid,elderly,stg1,stg2,stg3,stg4")
outfile2.write("\n")
for y in range (0,num_records):
        outfile2.write(idArray[y]+",")
        if (pInfo[idArray[y]+"fsis1"] == 1) :
                outfile2.write("1,")
        else :
                outfile2.write("0,")
        outfile2.write(str(float(pInfo[idArray[y]+"fsis1"]))+",")
        outfile2.write("1,")
        outfile2.write(pInfo[idArray[y]+"nat"]+",")
        outfile2.write(pInfo[idArray[y]+"gender"]+",")
        outfile2.write(pInfo[idArray[y]+"age"]+",")
        outfile2.write(pInfo[idArray[y]+"perp1"]+",")
        outfile2.write(pInfo[idArray[y]+"perp2"]+",")
        outfile2.write(pInfo[idArray[y]+"anar"]+",")
        outfile2.write(pInfo[idArray[y]+"fate"]+",")
        outfile2.write(pInfo[idArray[y]+"kid"]+",")
        outfile2.write(pInfo[idArray[y]+"elderly"]+",")
        outfile2.write("1,")
        outfile2.write("0,")
        outfile2.write("0,")
        outfile2.write("0,")
        outfile2.write("\n")
        outfile2.write(idArray[y]+",")
        if (pInfo[idArray[y]+"fsis2"] == 1) :
                outfile2.write("1,")
        else :
                outfile2.write("0,")
        outfile2.write(str(float(pInfo[idArray[y]+"fsis2"]))+",")
        outfile2.write("2,")
        outfile2.write(pInfo[idArray[y]+"nat"]+",")
        outfile2.write(pInfo[idArray[y]+"gender"]+",")
        outfile2.write(pInfo[idArray[y]+"age"]+",")
        outfile2.write(pInfo[idArray[y]+"perp1"]+",")
        outfile2.write(pInfo[idArray[y]+"perp2"]+",")
        outfile2.write(pInfo[idArray[y]+"anar"]+",")
        outfile2.write(pInfo[idArray[y]+"fate"]+",")
        outfile2.write(pInfo[idArray[y]+"kid"]+",")
        outfile2.write(pInfo[idArray[y]+"elderly"]+",")
        outfile2.write("0,")
        outfile2.write("1,")
        outfile2.write("0,")
        outfile2.write("0,")
        outfile2.write("\n")
        outfile2.write(idArray[y]+",")
        if (pInfo[idArray[y]+"fsis3"] == 1) :
                outfile2.write("1,")
        else :
                outfile2.write("0,")
        outfile2.write(str(float(pInfo[idArray[y]+"fsis3"]))+",")
        outfile2.write("3,")
        outfile2.write(pInfo[idArray[y]+"nat"]+",")
        outfile2.write(pInfo[idArray[y]+"gender"]+",")
        outfile2.write(pInfo[idArray[y]+"age"]+",")
        outfile2.write(pInfo[idArray[y]+"perp1"]+",")
        outfile2.write(pInfo[idArray[y]+"perp2"]+",")
        outfile2.write(pInfo[idArray[y]+"anar"]+",")
        outfile2.write(pInfo[idArray[y]+"fate"]+",")
        outfile2.write(pInfo[idArray[y]+"kid"]+",")
        outfile2.write(pInfo[idArray[y]+"elderly"]+",")
        outfile2.write("0,")
        outfile2.write("0,")
        outfile2.write("1,")
        outfile2.write("0,")
        outfile2.write("\n")
        outfile2.write(idArray[y]+",")
```

```
        if (pInfo[idArray[y]+"fsis4"] == 1) :
                outfile2.write("1,")
        else :
                outfile2.write("0,")
        outfile2.write(str(float(pInfo[idArray[y]+"fsis4"]))+",")
        outfile2.write("4,")
        outfile2.write(pInfo[idArray[y]+"nat"]+",")
        outfile2.write(pInfo[idArray[y]+"gender"]+",")
        outfile2.write(pInfo[idArray[y]+"age"]+",")
        outfile2.write(pInfo[idArray[y]+"perp1"]+",")
        outfile2.write(pInfo[idArray[y]+"perp2"]+",")
        outfile2.write(pInfo[idArray[y]+"anar"]+",")
        outfile2.write(pInfo[idArray[y]+"fate"]+",")
        outfile2.write(pInfo[idArray[y]+"kid"]+",")
        outfile2.write(pInfo[idArray[y]+"elderly"]+",")
        outfile2.write("0,")
        outfile2.write("0,")
        outfile2.write("0,")
        outfile2.write("1,")
        outfile2.write("\n")
#       outfile2.write(idArray[y]+",")
#       outfile2.write(str(float(pInfo[idArray[y]+"fsis5"]))+",")
#       outfile2.write("5,")
#       outfile2.write(pInfo[idArray[y]+"nat"]+",")
#       outfile2.write(pInfo[idArray[y]+"gender"]+",")
#       outfile2.write(pInfo[idArray[y]+"age"]+",")
#       outfile2.write(pInfo[idArray[y]+"perp1"]+",")
#       outfile2.write(pInfo[idArray[y]+"perp2"]+",")
#       outfile2.write(pInfo[idArray[y]+"anar"]+",")
#       outfile2.write("\n")


outfile2.close()
arcpy.AddMessage("Done writing output table...")

#
# va boire une biere
#
```

REFERENCES

Bakewell, O. 2008. Research beyond the categories: the importance of policy irrelevant research into forced migration. *Journal of Refugee Studies* 21 (4): 432–453.

Bensoussan, G., and M. Marie. 2014. *Atlas de la Shoah: la mise à mort des Juifs d'Europe, 1939-1945.* Paris: Autrement.

Benz, U., and T. Axelrod. 2005. Traumatization through separation: loss of family and home as childhood catastrophes. *Shofar: An Interdisciplinary Journal of Jewish Studies* 23 (1): 85–99.

Beorn, W., T. Cole, S. Gigliotti, A. Giordano, A. Holian, P. B. Jaskot, A. K. Knowles, M. Masurovsky, and E. B. Steiner. 2009. Geographies of the Holocaust. *Geographical Review* 99 (4): 563–574.

Boss, P. 2004. Ambiguous loss research, theory, and practice: reflections after 9/11. *Journal of Marriage and Family* 66 (3): 551–566.

Burrell, K. 2008. Materialising the border: spaces of mobility and material culture in migration from post-socialist poland. *Mobilities* 3 (3): 353–373.

Castles, S., H. de Haas, and M. Miller. 2014. *The age of migration: international population movements in the modern world.* London: The Guilford Press.

Charlesworth, A. 2004. The topography of genocide. In *The historiography of the Holocaust,* ed. D. Stone, 216–252. Basingstoke: Palgrave Macmillan UK.

Cole, T. 2015. (re)placing the past: spatial strategies of retelling difficult stories. *The Oral History Review* 42 (1).

Cooperman, B. D., and B. Garvin, eds. 2000. *The Jews of Italy; memory and identity.* Bethesda, MD: University Press of Maryland.

Dreby, J. 2015. US immigration policy and family separation: the consequences for children's well-being. *Social Science & Medicine* 132:245–251.

Dumon, W. A. 1989. Family and migration. *International Migration* 27 (2): 251–270.

Enchautegui, M. E., and C. Menjívar. 2015. Paradoxes of family immigration policy: separation, reorganization, and reunification of families under current immigration laws. *Law & Policy* 37 (1-2): 32–60.

Gigliotti, S. 2009. *The train journey: transit, captivity, and witnessing in the Holocaust.* War and Genocide. Oxford: Berghahn Books.

Gilbert, M. 2009. *The Routledge atlas of the Holocaust.* Milan: Routledge.

Giordano, A., and A. Holian. 2014. Retracing the "Hunt for Jews". a spatio-temporal analysis of arrests during the Holocaust in Italy. In *Geographies of the Holocaust,* ed. A. K. Knowles, T. Cole, and A. Giordano, 53–86. Bloomington, In: Indiana University Press.

Goldstein, R. D., N. S. Wampler, and P. H. Wise. 1997. War experiences and distress symptoms of bosnian children. *Pediatrics* 100 (5): 873–878.

Hathaway, J. C. 2007. Forced migration studies: could we agree just to 'date'? *Journal of Refugee Studies* 20 (3): 349–369.

Herr, A. 2014. Fossoli di Carpi: The history and memory of the Holocaust in Italy. Ph.D. diss., Department of History, Clark University.

Katz, R. 2005. The Möllhausen telegram, the Kappler decodes, and the deportation of the Jews of Rome : The new CIA-OSS documents, 2000–2002. In *Jews in Italy under fascist and nazi rule, 1922–1945,* ed. J. D. Zimmerman, 224–242. Cambridge: Cambridge University Press.

Knowles, A. K., T. Cole, and A. Giordano, eds. 2014. *Geographies of the Holocaust.* Bloomington, In: Indiana University Press.

Knowles, A. K., P. B. Jaskot, C. W. Harvey, C. Hofmann, T. Patel, R. Vara, and A. Yule. 2014. Mapping the SS concentration camps. In *Geographies of the Holocaust,* ed. A. K. Knowles, T. Cole, and A. Giordano, 19–50. Bloomington, In: Indiana University Press.

Liang, K.-Y., and S. L. Zeger. 1986. Longitudinal data analysis using generalized linear models. *Biometrika* 73 (1): 13–22.

Luster, T., D. Qin, L. Bates, D. Johnson, and M. Rana. 2009. The lost boys of Sudan: coping with ambiguous loss and separation from parents. *American journal of Orthopsychiatry* 79 (2): 203.

McGregor, L. S., G. A. Melvin, and L. K. Newman. 2015. Familial separations, coping styles, and PTSD symptomatology in resettled refugee youth. *The Journal of Nervous and Mental Disease* 203 (6): 431–438.

Pesonen, A.-K., K. Räikkönen, K. Heinonen, E. Kajantie, T. Forsén, and J. G. Eriksson. 2008. Reproductive traits following a parent–child separation trauma during childhood: a natural experiment during World War II. *American Journal of Human Biology* 20 (3): 345–351.

Petersen, W. 1958. A general typology of migration. *American Sociological Review:* 256–266.

Picciotto Fargion, L. 2000. The persecution of Jews in Italy, 1943-1945 : a chronicle of events. In *The jews of italy; memory and identity,* ed. B. D. Cooperman and B. Garvin, 443–454. Bethesda, MD: University Press of Maryland.

———. 2002. *Il libro della memoria: Gli ebrei deportati dall'Italia (1943-1945).* Milan: Mursia.

———. 2005. The Shoah in Italy : History and characteristics. In *Jews in Italy under fascist and nazi rule, 1922–1945,* ed. J. D. Zimmerman, 209–223. Cambridge: Cambridge University Press.

Picciotto, L. 1998. Les archives de l'histoire de la Shoah en Italie. In *Les archives de la Shoah,* ed. J. Fredj, 131–156. Paris: L'Harmattan.

Picciotto, L. 2005. Statistical tables on the Holocaust in Italy with an insight on the mechanism of the deportations. *Yad Vashem Studies* 33:307–343.

Schapendonk, J., I. van Liempt, and B. Spierings. 2015. Travellers and their journeys: a dynamic conceptualization of transient migrants' and backpackers' behaviour and experiences on the road. *Migration Studies* 3 (1): 49–67.

Schapendonk, J., and G. Steel. 2014. Following migrant trajectories: the im/mobility of Sub-Saharan Africans en route to the European Union. *Annals of the Association of American Geographers* 104 (2): 262–270.

Seibel, W. 2002. The strength of perpetrators - The Holocaust in western Europe, 1940–1944. *Governance* 15 (2): 211–240.

Smit, R., and P. Rugunanan. 2015. Transnational forced migration and negotiating emotional well-being: the case of women refugees in South Africa. *Social Dynamics,* no. ahead-of-print: 1–20.

United States Holocaust Memorial Museum. 2013. *The Holocaust in Greece.* `https://www.ushmm.org/m/pdfs/20130305-holocaust-in-greece.pdf` (last accessed 8 March 2016).

Villani, C. 2005. The persecution of Jews in two regions of occupied Northern Italy, 1943–1945: Operationszone Alpenvorland and Operationszone Adriatisches Kustenland. In *Jews in Italy under fascist and nazi rule, 1922–1945,* ed. J. D. Zimmerman, 243–261. Cambridge: Cambridge University Press.

von Joeden-Forgey, E. 2010. The devil in the details:"life force atrocities" and the assault on the family in times of conflict. *Genocide Studies and Prevention* 5 (1): 1–19.

Walters, W. 2002. Deportation, expulsion, and the international police of aliens. *Citizenship Studies* 6 (3): 265–292.

Wilmsen, B. 2013. Family separation and the impacts on refugee settlement in Australia. *Australian Journal of Social Issues* 48 (2): 241.

Zimmerman, J. D., ed. 2005. *Jews in Italy under fascist and nazi rule, 1922–1945.* Cambridge: Cambridge University Press.

Zuccotti, S., and F. Colombo. 1987. *The Italians and the Holocaust: persecution, rescue, and survival.* New-York, NY: Halban.