Geocoding and BISG Manual

John Curiel

August 6, 2020

Introduction

This is an instruction manual on how to geocode and predict the race of individuals by using Bayesian Inference with Surname and Geography (BISG). The document covers how to read in voterfiles, produce tables ammenable to read into an ArcGIS geocoder program, and then import these results back into R so as to impute race. By using BISG, it is possible to greatly improve upon ecological inference and estimate electoral participation given an individual's race.

Step 1: Cleaning and exporting the voterfile

The first step to geocoding a voterfile is to ensure that there are the appropriate fields necessary to do so, the full address field. Most frequently, the state will provide the relevant address fields, but usually not in the precise manner. Therefor, it will be necessary to read in the data and see what is available.

```
library(foreign)
options(stringsAsFactors = FALSE)
wisconsin_vf <- read.csv("F:/voterfile/wi/wisconsin_voterfile_sample.csv")
nrow(wisconsin_vf)
```

```
## [1] 1000
```

```
names(wisconsin_vf)
```

```
##  [1] "voterregnumber"       "lastname"             "firstname"
##  [4] "voterstatus"          "voterstatusreason"    "address1"
##  [7] "address2"             "ballotdeliverymethod" "ballotstatusreason"
## [10] "ballotreasontype"     "electionname"         "county"
```

```
head(wisconsin_vf)
```

```
##   voterregnumber lastname firstname voterstatus voterstatusreason
## 1      710556760   ANDREW    HILARY      Active        Registered
## 2      713639860   WALLIN     Craig      Active        Registered
## 3      700674106   PITLIK    Kelvin      Active        Registered
## 4       12124943     HEIN     James      Active        Registered
## 5       51667964   FULMER   Theresa      Active        Registered
## 6      700493498    SMITH      Ryan      Active        Registered
##               address1                address2 ballotdeliverymethod
## 1      1912 WEBSTER AVE  EAU CLAIRE WI 54701-6647                Mail
## 2          2089 COOK DR    SOMERSET WI 54025-7514                Mail
## 3 4710 SILENT SHORES DR RHINELANDER WI 54501-8649                Mail
## 4      2189 IRONWOOD DR   GREEN BAY WI 54304-1972                Mail
## 5       20 S CONCORD RD  OCONOMOWOC WI 53066-2737                Mail
```

```
## 6 1670 S HURON RD APT 2   GREEN BAY WI 54311-8008                Mail
##   ballotstatusreason ballotreasontype
## 1           Returned
## 2           Returned
## 3           Returned
## 4           Returned
## 5           Returned
## 6           Returned
##                                          electionname     county
## 1 2020 Spring Election and Presidential Preference Vote EAU CLAIRE
## 2 2020 Spring Election and Presidential Preference Vote  ST. CROIX
## 3 2020 Spring Election and Presidential Preference Vote     ONEIDA
## 4 2020 Spring Election and Presidential Preference Vote      BROWN
## 5 2020 Spring Election and Presidential Preference Vote   WAUKESHA
## 6 2020 Spring Election and Presidential Preference Vote      BROWN
```

In the above example, we read in the Wisconsin voterfile, with a unique observation for every registered voter. There are a total of 1,000 observations. It is apparent that the address information is located in two fields, address1 and address2. Therefore, we will want to combine them into a full address field. Additionally, we will want to ensure that said field is all uppercase, which will make the geocoding string matching later on more accurate.

```
library(stringr)
wisconsin_vf$full_addrs <- paste0(wisconsin_vf$address1, sep=", ", wisconsin_vf$address2)
wisconsin_vf$full_addrs <- str_to_upper(wisconsin_vf$full_addrs)
head(wisconsin_vf$full_addrs)
```

```
## [1] "1912 WEBSTER AVE, EAU CLAIRE WI 54701-6647"
## [2] "2089 COOK DR, SOMERSET WI 54025-7514"
## [3] "4710 SILENT SHORES DR, RHINELANDER WI 54501-8649"
## [4] "2189 IRONWOOD DR, GREEN BAY WI 54304-1972"
## [5] "20 S CONCORD RD, OCONOMOWOC WI 53066-2737"
## [6] "1670 S HURON RD APT 2, GREEN BAY WI 54311-8008"
```

It appears to be the case that we now have the necessary full address field. However, it would be a waste to export the entire table as is. It might be the case that there are fewer than 1,000 unique addresses. Additionally, there is no need to export all of the fields. Therefore, let's slim down the data.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
wisconsin_vf <- subset(wisconsin_vf, select=c(full_addrs, county))
wisconsin_vf <- wisconsin_vf[!duplicated(wisconsin_vf$full_addrs), ]
nrow(wisconsin_vf)
```

```
## [1] 1000
```

```
head(wisconsin_vf)
```

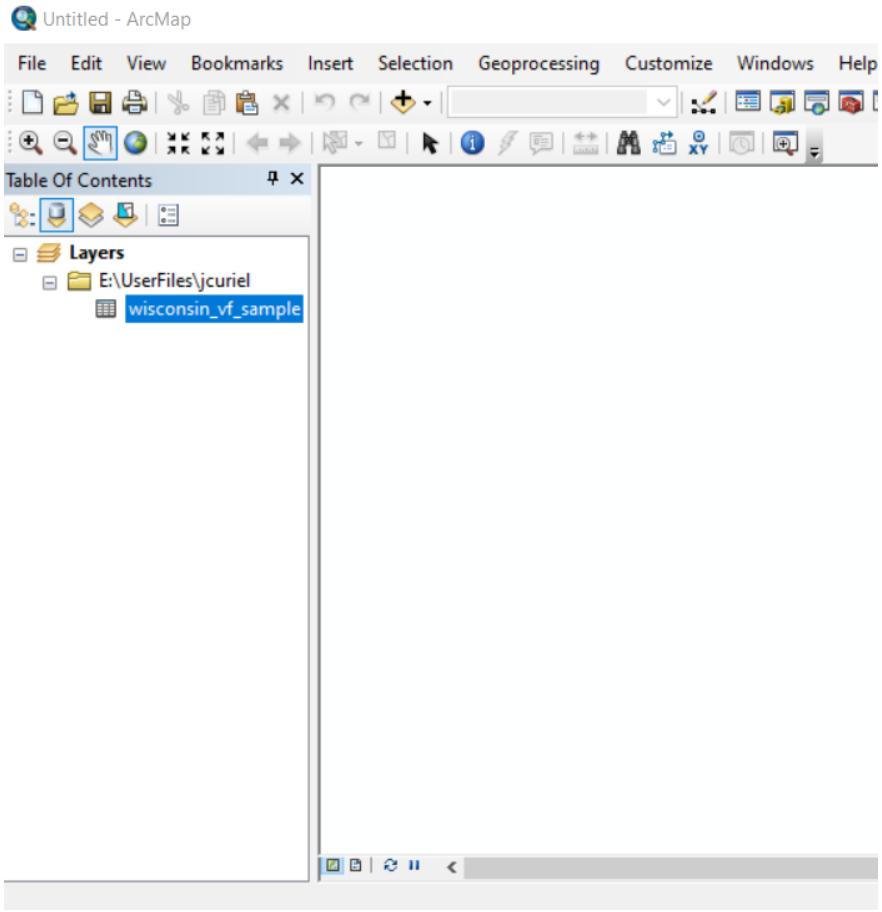


Figure 1: Adding the DBF to ArcMap

```
##                                         full_addrs     county
## 1       1912 WEBSTER AVE, EAU CLAIRE WI 54701-6647 EAU CLAIRE
## 2             2089 COOK DR, SOMERSET WI 54025-7514  ST. CROIX
## 3 4710 SILENT SHORES DR, RHINELANDER WI 54501-8649     ONEIDA
## 4        2189 IRONWOOD DR, GREEN BAY WI 54304-1972      BROWN
## 5        20 S CONCORD RD, OCONOMOWOC WI 53066-2737   WAUKESHA
## 6   1670 S HURON RD APT 2, GREEN BAY WI 54311-8008      BROWN
```

```
write.dbf(wisconsin_vf, "wisconsin_vf_sample.dbf")
```

In this case, we see that there were 1,000 unique addresses, so the number of rows are the same. However, we now only have 2 columns, which is more than sufficient to merge the coordinate data later on. Now we can export the file with the write.dbf() command, which exports the table as as dbf, which is the table formatted file for ArcGIS.

Step 2: Geocoding in ArcGIS

MIT has a license to ESRI ArcGIS products that any student, faculty, or staff member can make use of. Upon opening up ArcMaps, add the dbf via the add data command, as seen in Figure 1. This will allow us to make use of the file and geocode.

After adding the data, the next step is to open the geocoding tool, which can be found in the ArcGIS toolbox.

The path is, Geocoding Tools > Geocoding Addresses, as presented in Figure 2.

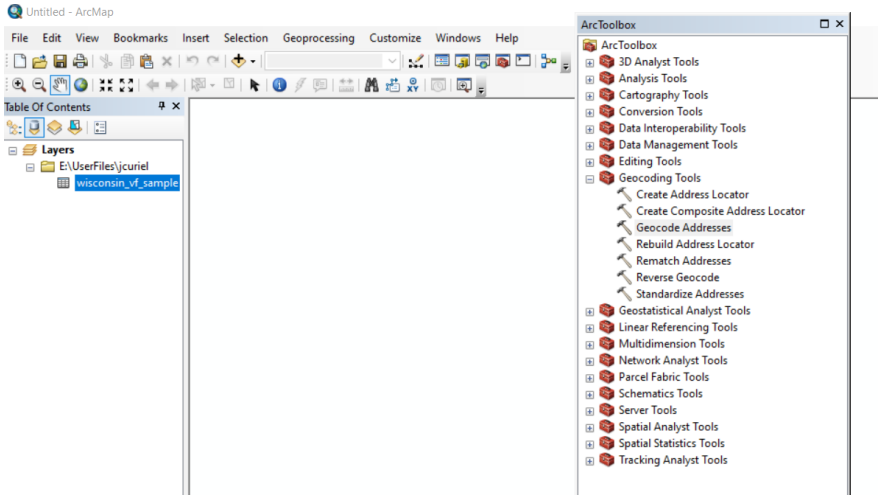


Figure 2: Geocoding Toolbox location

In order to use the geocoding tool, however, it will be necessary to have an ESRI address locator. There are a variety of types of locator files that can be used, to varying degrees of accuracy. Like most ESRI files, these locator files have two component files. All must be present in order to use. The MIT GIS Lab has these locator files available, and additional locators can be founnd in the Healthy Elections Dropbox. The locator file structure is such that it includes a .loc/.lox and .xml file. With these available, it will be possible to load in the locator file for the purpose of matching addresses to points. An example of the file structure can be seen in Figure 3.

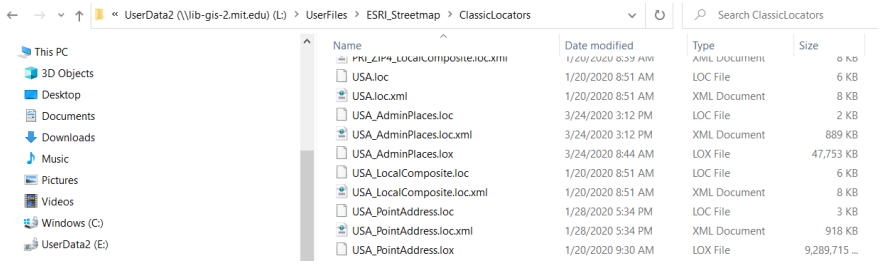


Figure 3: ESRI Locator File Structure

After confirming the presence of the necessary locator files, double click on the geocoding addresses toolbox in order to open the graphic user interface. The interface can be seen in Figures 4 and 5. There will be four fields of interest that the user needs to specify: (1) input table, (2) input address locator, (3) input address fields, and (4) output feature class. The input table will be the dbf table of the voterfile addresses that we added in earlier on into ArcMap. The input address locator specifies the locator file that we wish to use. In this case, we are using the USA locator file. The input address fields requests that the user specify whether there is a single or multiple fields containing the relevant address information from the dbf. In our case, we combined the address into a single field, therefore we will click single address, then under the right column, Alias Name, select a drop down arrow to select the full_addrs column from the dbf that we created earlier.

Upon specifying these input fields, the final step is to scroll down the graphic user interface and select where we wish to save the data. The output should be saved to a file geodatabase, which will ensure efficiency of memory and communication between the several types of ESRI files that make up a full shape file. Upon filling out all of the fields, click ok and run the geocoder. Note that while this data set has only 1,000 observations and will thus run quickly, voterfiles often contain millions of observations. This can easily take several hours, so move onto other tasks and check back in later.

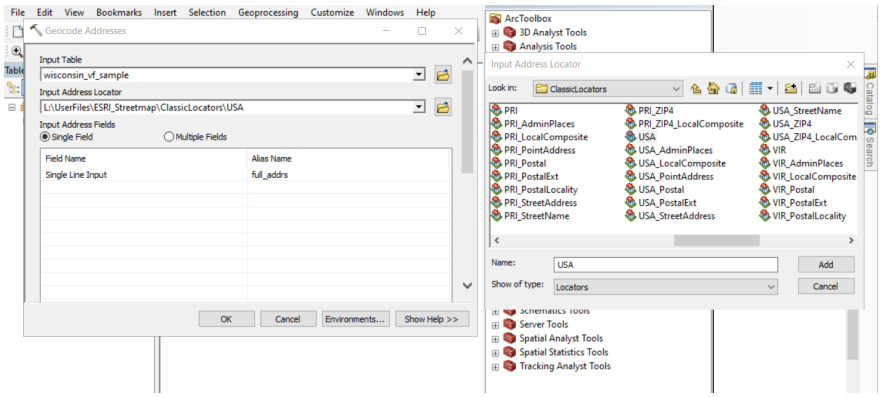


Figure 4: Geocoding Interface top half

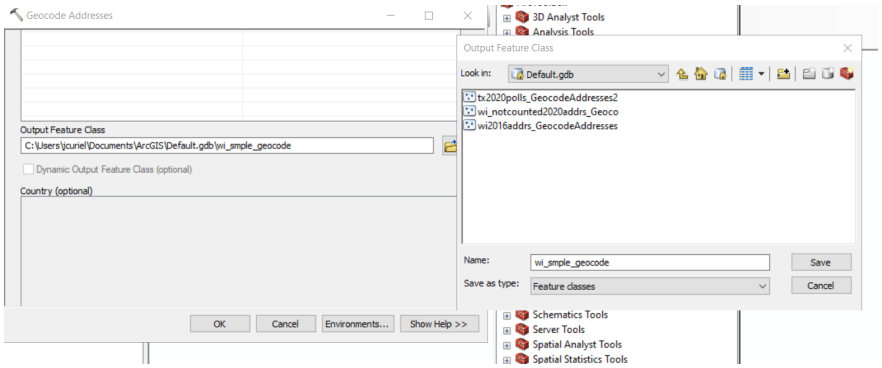


Figure 5: Geocoding interface bottom half

Upon the completion of the geocoding, we now see the addresses successfully geocoded and converted into a shape file stored in the geodatabase, with the points mapped onto the interactive map, as presented in Figure 6. While it is technically possible to use the output as is, what we have is inefficient and requires the loading of too many packages in R. Additionally, even if we read in these data, reading in too large of an ESRI shape file can easily break R. Therefore, we instead need to export the table, which can then easily be converted into a spatial dataframe in R.

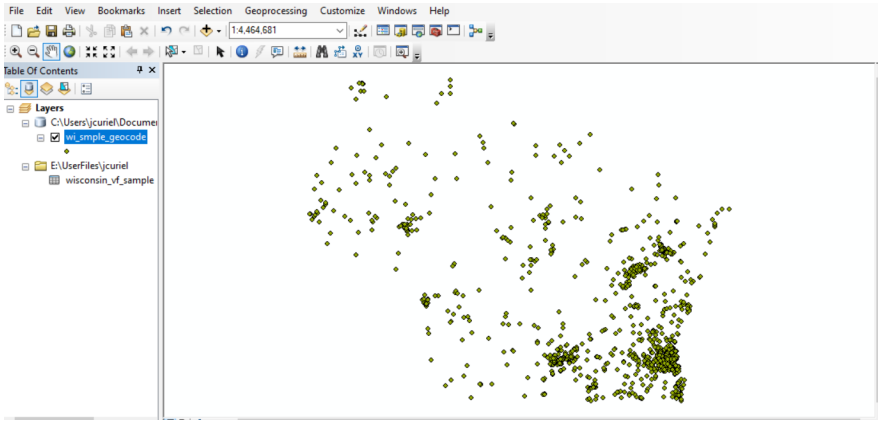
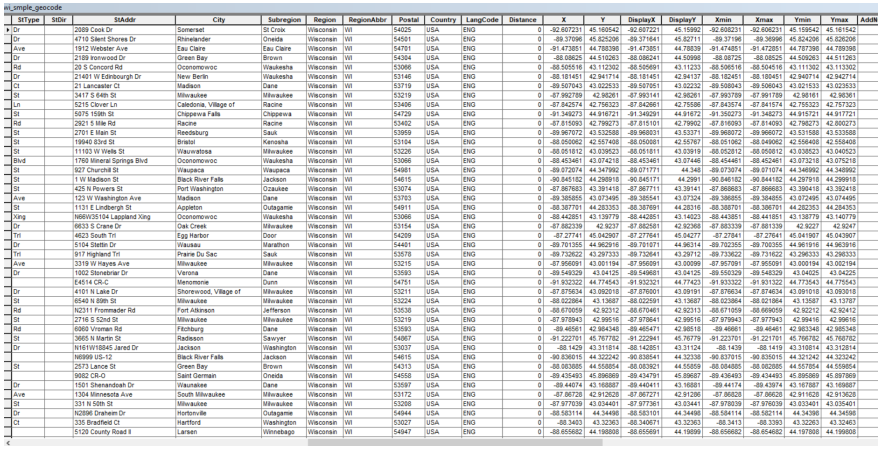


Figure 6: Geocoded output

We will first check to see which fields are present in the attribute table, which we can reach by right-clicking

the data on the left data panel, the select open attribute table. We only really need the X and Y coordinate fields, in addition to some meta info on which locator geocoded the address, accuracy score, etc. However, as seen in Figure 7, upon opening up the attribute table, we instead see dozens of fields as the geocoder decided to merge all of the meta information onto our exported table. We will want to reduce the number of fields present before exporting the table.



	StType	StDir	StAddr	City	Subregion	Region	RegionAbbr	Postal	Country	LangCode	Distance	X	Y	DisplayX	DisplayY	Xmin	Xmax	Ymin	Ymax	AddN
	Dr		2989 Cook Dr	Somerset	St Croix	Wisconsin	WI	54025	USA	ENG	0	-92.607231	45.160542	-92.607221	45.15992	-92.608231	-92.606231	45.159542	45.161542	
	Dr		4710 Silent Shores Dr	Rhinelander	Oneida	Wisconsin	WI	54501	USA	ENG	0	-89.37096	45.825286	-89.371641	45.82711	-89.37196	-89.36996	45.824286	45.826286	
	Ave		1912 Webster Ave	Eau Claire	Eau Claire	Wisconsin	WI	54701	USA	ENG	0	-91.473851	44.768396	-91.473851	44.76839	-91.474851	-91.472851	44.767396	44.769396	
	Dr		2189 Ironwood Dr	Green Bay	Brown	Wisconsin	WI	54304	USA	ENG	0	-88.08625	44.512263	-88.086241	44.50998	-88.08725	-88.08525	44.509263	44.511263	
	Rd		28 S Concord Rd	Oconomowoc	Waukesha	Wisconsin	WI	53066	USA	ENG	0	-88.505516	43.112302	-88.505691	43.11233	-88.506516	-88.504516	43.111302	43.113302	
	Dr		21401 W Edinbourgh Dr	New Berlin	Waukesha	Wisconsin	WI	53146	USA	ENG	0	-88.181451	42.941714	-88.181451	42.94137	-88.182451	-88.180451	42.940714	42.942714	
	Ct		21 Lancaster Ct	Madison	Dane	Wisconsin	WI	53719	USA	ENG	0	-89.507043	43.022533	-89.507051	43.02232	-89.508043	-89.506043	43.021533	43.023533	
	St		3417 S 64th St	Milwaukee	Milwaukee	Wisconsin	WI	53219	USA	ENG	0	-87.962709	42.962261	-87.960141	42.96261	-87.963709	-87.961709	42.96161	42.96361	
	Ln		5215 Clover Ln	Caledonia, Village of	Racine	Wisconsin	WI	53406	USA	ENG	0	-87.842574	42.756323	-87.842661	42.75586	-87.843574	-87.841574	42.755323	42.757323	
	St		5975 159th St	Chippewa Falls	Chippewa	Wisconsin	WI	54729	USA	ENG	0	-91.349273	44.916721	-91.349291	44.91672	-91.350273	-91.348273	44.915721	44.917721	
	Rd		2921 S Mile Rd	Racine	Racine	Wisconsin	WI	53402	USA	ENG	0	-87.816093	42.798273	-87.815101	42.79902	-87.816093	-87.814093	42.798273	42.800273	
	St		2701 E Main St	Reedsburg	Sauk	Wisconsin	WI	53959	USA	ENG	0	-89.967072	43.532588	-89.968031	43.53371	-89.968072	-89.966072	43.531588	43.533588	
	St		19940 83rd St	Bristol	Kenosha	Wisconsin	WI	53104	USA	ENG	0	-88.050082	42.557408	-88.050081	42.55767	-88.051082	-88.049082	42.556408	42.558408	
	St		11103 W Wells St	Wauwatosa	Milwaukee	Wisconsin	WI	53226	USA	ENG	0	-88.051812	43.039523	-88.051811	43.03919	-88.052812	-88.050812	43.038523	43.040523	
	Blvd		1760 Mineral Springs Blvd	Oconomowoc	Waukesha	Wisconsin	WI	53066	USA	ENG	0	-88.453461	43.074218	-88.453461	43.07448	-88.454461	-88.452461	43.073218	43.075218	
	St		927 Churchill St	Waupaca	Waupaca	Wisconsin	WI	54981	USA	ENG	0	-89.072074	44.347992	-89.071771	44.348	-89.073074	-89.071074	44.346992	44.348992	
	St		1 W Madison St	Black River Falls	Jackson	Wisconsin	WI	54615	USA	ENG	0	-90.846182	44.298918	-90.846171	44.2991	-90.846182	-90.844182	44.297918	44.299918	
	St		425 N Powers St	Port Washington	Ozaukee	Wisconsin	WI	53074	USA	ENG	0	-87.867683	43.399418	-87.867711	43.39141	-87.868683	-87.866683	43.390418	43.392418	
	Ave		123 W Washington Ave	Madison	Dane	Wisconsin	WI	53703	USA	ENG	0	-89.385855	43.073495	-89.385541	43.07324	-89.386855	-89.384855	43.072495	43.074495	
	St		1131 E Lindbergh St	Appleton	Outagamie	Wisconsin	WI	54911	USA	ENG	0	-88.387701	44.283353	-88.387691	44.28316	-88.388701	-88.386701	44.282353	44.284353	
	Xing		N88W35194 Lapsland Xing	Oconomowoc	Waukesha	Wisconsin	WI	53066	USA	ENG	0	-88.442851	43.139779	-88.442851	43.14023	-88.443851	-88.441851	43.138779	43.140779	
	Dr		6633 S Crane Dr	Oak Creek	Milwaukee	Wisconsin	WI	53154	USA	ENG	0	-87.882339	42.9237	-87.882581	42.92368	-87.883339	-87.881339	42.9227	42.9247	
	Trl		4623 South Trl	Egg Harbor	Door	Wisconsin	WI	54209	USA	ENG	0	-87.27741	45.042907	-87.277641	45.04277	-87.27841	-87.27641	45.041907	45.043907	
	Dr		5104 Stettin Dr	Wausau	Marathon	Wisconsin	WI	54401	USA	ENG	0	-89.701355	44.962916	-89.701571	44.96314	-89.702355	-89.700355	44.961916	44.963916	
	Trl		917 Highland Trl	Prairie Du Sac	Sauk	Wisconsin	WI	53578	USA	ENG	0	-89.732622	43.297333	-89.732641	43.29712	-89.733622	-89.731622	43.296333	43.298333	
	Ave		3319 W Hayes Ave	Milwaukee	Milwaukee	Wisconsin	WI	53215	USA	ENG	0	-87.956091	43.001194	-87.956091	43.00099	-87.957091	-87.955091	43.000194	43.002194	
	Dr		1002 Stonebriar Dr	Verona	Dane	Wisconsin	WI	53593	USA	ENG	0	-89.549329	43.04125	-89.549681	43.04125	-89.550329	-89.548329	43.04025	43.04225	
			E4514 CR-C	Menomonie	Dunn	Wisconsin	WI	54751	USA	ENG	0	-91.932322	44.774543	-91.932321	44.77423	-91.933322	-91.931322	44.773543	44.775543	
	Dr		4101 N Lake Dr	Shorewood, Village of	Milwaukee	Wisconsin	WI	53211	USA	ENG	0	-87.879534	43.092018	-87.879501	43.09191	-87.879634	-87.879434	43.091018	43.093018	
	St		6540 N 8th St	Milwaukee	Milwaukee	Wisconsin	WI	53224	USA	ENG	0	-88.022884	43.13687	-88.022591	43.13687	-88.023884	-88.021884	43.13587	43.13787	
	Rd		N2311 Frommader Rd	Fort Atkinson	Jefferson	Wisconsin	WI	53538	USA	ENG	0	-88.670059	42.92312	-88.670461	42.92313	-88.671059	-88.669059	42.92212	42.92412	
	St		2716 S 52nd St	Milwaukee	Milwaukee	Wisconsin	WI	53219	USA	ENG	0	-87.978943	42.99516	-87.979841	42.99516	-87.979943	-87.977943	42.99416	42.99616	
	Rd		6969 Vroman Rd	Fitchburg	Dane	Wisconsin	WI	53593	USA	ENG	0	-89.46541	42.964348	-89.465471	42.96518	-89.46681	-89.46481	42.963348	42.965348	
	St		3665 N Martin St	Radisson	Sawyer	Wisconsin	WI	54867	USA	ENG	0	-91.222701	45.767782	-91.222941	45.76779	-91.223701	-91.221701	45.766782	45.768782	
	Dr		N16W18845 Jared Dr	Jackson	Washington	Wisconsin	WI	53037	USA	ENG	0	-88.1429	43.311814	-88.142851	43.31124	-88.1439	-88.1419	43.310814	43.312814	
			N6899 US-12	Black River Falls	Jackson	Wisconsin	WI	54615	USA	ENG	0	-90.836615	44.322242	-90.838541	44.32238	-90.837515	-90.835515	44.321242	44.323242	
	St		2573 Lance St	Green Bay	Brown	Wisconsin	WI	54313	USA	ENG	0	-88.083685	44.558854	-88.083921	44.55859	-88.084685	-88.082685	44.557854	44.559854	
			9682 CR-O	Saint Germain	Oneida	Wisconsin	WI	54558	USA	ENG	0	-89.435493	45.896869	-89.434791	45.89687	-89.436493	-89.434493	45.895869	45.897869	
	Dr		1501 Shenandoah Dr	Waunakee	Dane	Wisconsin	WI	53597	USA	ENG	0	-89.44074	43.168807	-89.440411	43.16881	-89.44174	-89.43974	43.167807	43.169807	
	Ave		1304 Minnesota Ave	South Milwaukee	Milwaukee	Wisconsin	WI	53172	USA	ENG	0	-87.86728	42.912628	-87.867271	42.91286	-87.86828	-87.86628	42.911628	42.913628	
	St		331 N 58th St	Milwaukee	Milwaukee	Wisconsin	WI	53208	USA	ENG	0	-87.977039	43.034401	-87.977361	43.03441	-87.978039	-87.976039	43.033401	43.035401	
	Dr		N2896 Draheim Dr	Hortonville	Outagamie	Wisconsin	WI	54944	USA	ENG	0	-88.583114	44.34498	-88.583101	44.34498	-88.584114	-88.582114	44.34398	44.34598	
	Ct		335 Bradfield Ct	Hartford	Washington	Wisconsin	WI	53027	USA	ENG	0	-88.3403	43.32363	-88.340671	43.32363	-88.3413	-88.3393	43.32263	43.32463	
			5129 County Road II	Larsen	Winnebago	Wisconsin	WI	54947	USA	ENG	0	-88.655682	44.198886	-88.655691	44.19899	-88.656682	-88.654682	44.197886	44.199886	

Figure 7: Geocoded output

In order to reduce the number of fields, right-click the data in the data panel, select properties, then select the fields column. The fields interface can be seen in Figure 8. We now see that all of the fields are selected, which means that they are visible when opening up the attribute table. We instead only need a few fields, therefore simply de-select the fields that are not necessary. For our purposes, we want the following:

Necessary fields:

- ObjectID - Not possible to export without this field present; will get an error otherwise.
- Loc_name - Names the geocoder used to geocode the address. This is especially useful when using composite locators and reporting procedures for purposes of replication.
- Score - The degree of confidence in teh accuracy of the geocoding, with 100 as complete certainty, and 0 uncertainty.
- StName - A field that when empty, informs us that a postal geocoder was used, which is less precise relative to other geocoders.
- X - The reported longitude.
- Y - The reported latitude.
- full_addrs - This is the primary field that will be used to merge the coordinates onto the voterfile.
- county - In the event that the data is too large for a merge later on, this field will be used to subset and merge the data via a loop.

After selecting the fields then clicking ok, we can go back to open the attribute table for the geocoded data. We see that the table is now slimmed down with only the necessary data, as seen in Figure 9.

The final step is to export the data. To do so, after opening the attribute table, select the table options button on the top left corner of the screen, then click the export option. The necessary windows are presented in Figure 10. From there, click the folder icon in order to select the save location and name of the exported file. Once in the saving data window, select the drop down arrow for save as type, and select text file. From there, type out the name for the file, and be certain to type out .csv at the end in order to save the file as a

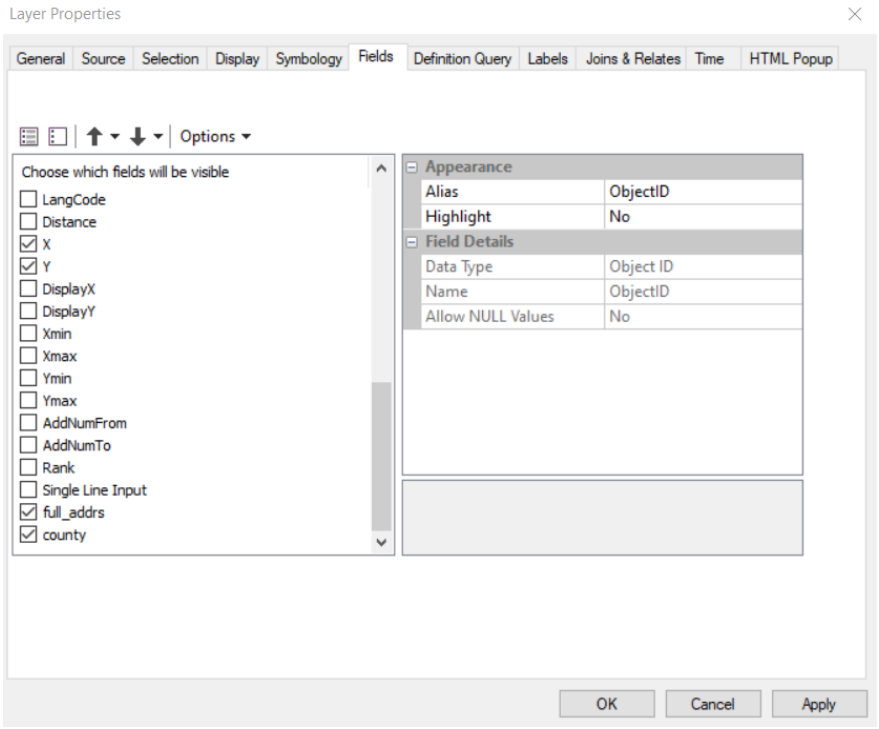


Figure 8: Geocoded output

wi_smple_geocode

ObjectID *	Loc_name	Score	StName	X	Y	full_addrs	county
1	PointAddress	100	Cook	-92.607231	45.160542	2089 COOK DR, SOMERSET WI 54025-7514	ST. CROIX
2	PointAddress	100	Silent Shores	-89.37096	45.825206	4710 SILENT SHORES DR, RHINELANDER WI 54501-8649	ONEIDA
3	PointAddress	100	Webster	-91.473851	44.788398	1912 WEBSTER AVE, EAU CLAIRE WI 54701-6647	EAU CLAIRE
4	PointAddress	100	Ironwood	-88.08625	44.510263	2189 IRONWOOD DR, GREEN BAY WI 54304-1972	BROWN
5	PointAddress	100	Concord	-88.505516	43.112302	20 S CONCORD RD, OCONOMOWOC WI 53066-2737	WAUKESHA
6	PointAddress	100	Edinbourgh	-88.181451	42.941714	21401 W EDINBOURGH DR, NEW BERLIN WI 53146-4815	WAUKESHA
7	PointAddress	100	Lancaster	-89.507043	43.022533	21 LANCASTER CT, MADISON WI 53719-1433	DANE
8	PointAddress	100	64th	-87.992789	42.98261	3417 S 64TH ST, MILWAUKEE WI 53219-4206	MILWAUKEE
9	PointAddress	98.98	Clover	-87.842574	42.756323	5215 CLOVER LN, CALEDONIA WI 53406-1550	RACINE
10	PointAddress	100	159th	-91.349273	44.916721	5075 159TH ST, CHIPPEWA FALLS WI 54729-7104	CHIPPEWA
11	PointAddress	100	5 Mile	-87.815093	42.799273	2921 5 MILE RD, RACINE WI 53402-1668	RACINE
12	PointAddress	100	Main	-89.967072	43.532588	2701 E MAIN ST LOT 61, REEDSBURG WI 53959-9458	SAUK
13	PointAddress	100	83rd	-88.050062	42.557408	19940 83RD ST, BRISTOL WI 53104	KENOSHA
14	PointAddress	100	Wells	-88.051812	43.039523	11103 W WELLS ST, WAUWATOSA WI 53226-3749	MILWAUKEE
15	PointAddress	100	Mineral Springs	-88.453461	43.074218	1760 MINERAL SPRINGS BLVD, OCONOMOWOC WI 53066-4871	WAUKESHA
16	PointAddress	100	Churchill	-89.072074	44.347992	927 CHURCHILL ST APT W1, WAUPACA WI 54981	WAUPACA
17	PointAddress	100	Madison	-90.845182	44.298918	1 W MADISON ST, BLACK RIVER FALLS WI 54615	JACKSON
18	PointAddress	100	Powers	-87.867683	43.391418	425 N POWERS ST, PORT WASHINGTON WI 53074-1610	OZAUKEE
19	PointAddress	100	Washington	-89.385855	43.073495	123 W WASHINGTON AVE UNIT 1005, MADISON WI 53703-2787	DANE
20	PointAddress	100	Lindbergh	-88.387701	44.283353	1131 E LINDBERGH ST, APPLETON WI 54911-3050	OUTAGAMIE
21	PointAddress	100	Lappland	-88.442851	43.139779	N66W35104 LAPPLAND XING, OCONOMOWOC WI 53066-1886	WAUKESHA
22	PointAddress	100	Crane	-87.882339	42.9237	6633 S CRANE DR, OAK CREEK WI 53154-1211	MILWAUKEE
23	PointAddress	97.22	South	-87.27741	45.042907	4623 S TRAIL RD, EGG HARBOR WI 54209-8837	DOOR
24	PointAddress	100	Stettin	-89.701355	44.962916	5104 STETTIN DR, WAUSAU WI 54401-3824	MARATHON
25	PointAddress	100	Highland	-89.732622	43.297333	917 HIGHLAND TRL, PRAIRIE DU SAC WI 53578-2021	SAUK
26	PointAddress	100	Hayes	-87.956091	43.001194	3319 W HAYES AVE, MILWAUKEE WI 53215-2824	MILWAUKEE
27	PointAddress	100	Stonebriar	-89.549329	43.04125	1002 STONEBRIAR DR, VERONA WI 53593-7821	DANE
28	PointAddress	100	C	-91.932322	44.774543	E4514 COUNTY ROAD C, MENOMONIE WI 54751	DUNN
29	PointAddress	98.98	Lake	-87.875634	43.092018	4101 N LAKE DR, SHOREWOOD WI 53211-1720	MILWAUKEE
30	PointAddress	100	89th	-88.022864	43.13687	6540 N 89TH ST, MILWAUKEE WI 53224-5310	MILWAUKEE
31	PointAddress	100	Frommader	-88.670059	42.92312	N2311 FROMMADER RD, FORT ATKINSON WI 53538-9092	JEFFERSON
32	PointAddress	100	52nd	-87.978943	42.99516	2716 S 52ND ST, MILWAUKEE WI 53219-3263	MILWAUKEE
33	PointAddress	100	Vroman	-89.46561	42.984348	6060 VROMAN RD, FITCHBURG WI 53593-9209	DANE
34	PointAddress	100	Martin	-91.222701	45.767782	3665 N MARTIN ST, RADISSON WI 54867	SAWYER
35	PointAddress	100	Jared	-88.1429	43.311814	N161W18845 JARED DR, JACKSON WI 53037-8924	WASHINGTON
36	PointAddress	100	12	-90.836015	44.322242	N6999 US HIGHWAY 12, BLACK RIVER FALLS WI 54615-5859	JACKSON
37	PointAddress	100	Lance	-88.083885	44.558854	2573 LANCE ST, GREEN BAY WI 54313-6750	BROWN
38	PointAddress	100	O	-89.435493	45.896869	9082 COUNTY ROAD O, SAINT GERMAIN WI 54558	ONEIDA
39	PointAddress	100	Shenandoah	-89.44074	43.168887	1501 SHENANDOAH DR, WAUNAKEE WI 53597-2353	DANE
40	PointAddress	100	Minnesota	-87.86728	42.912628	1304 MINNESOTA AVE, SOUTH MILWAUKEE WI 53172-1921	MILWAUKEE
41	PointAddress	100	50th	-87.977039	43.034401	331 N 50TH ST, MILWAUKEE WI 53208-3633	MILWAUKEE
42	PointAddress	100	Draheim	-88.583114	44.34498	N2896 DRAHEIM DR, HORTONVILLE WI 54944-9750	OUTAGAMIE
43	PointAddress	100	Bradfield	-88.3403	43.32363	335 BRADFIELD CT, HARTFORD WI 53027-8302	WASHINGTON
44	PointAddress	100	County Road II	-88.655682	44.198808	5120 COUNTY ROAD II, LARSEN WI 54947-8734	WINNEBAGO
45	PointAddress	100	Parkwood	-91.204099	43.762148	3605 PARKWOOD PL, LA CROSSE WI 54601-8319	LA CROSSE

Figure 9: Geocoded output

csv. Select save in the Saving Data window, then select ok in the Export Data window. Now the table will be saved to the specified location, and is ready to read into R.

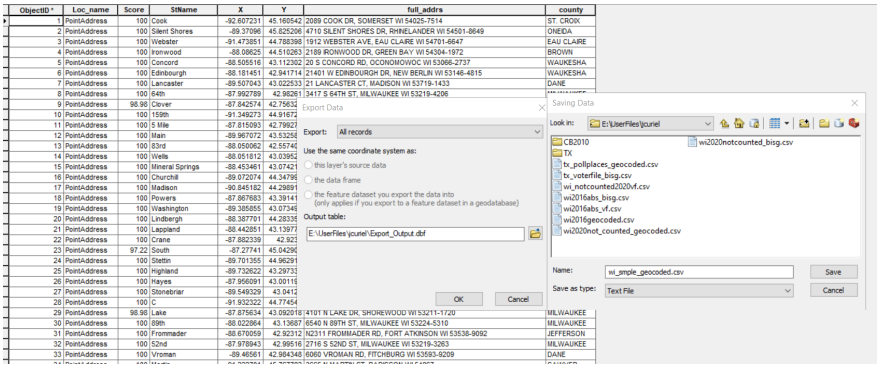


Figure 10: Geocoded output

Step 3: Identfying Census Geographies

Now that the points are geocoded, it is possible to load in the .csv exported from ArcMap. The table contains the relevant coordinate information that can be used to identify the racial geographic characteristics that can be used to impute an individual's race. Therefore, to start off with, read in the data.

```
wi_smpl_geocoded <- read.csv("F:/voterfile/wi/wi_smple_geocoded.csv")
class(wi_smpl_geocoded)
```

```
## [1] "data.frame"
```

```
head(wi_smpl_geocoded)
```

```
##   ObjectID     Loc_name Score        StName         X        Y
## 1        1 PointAddress   100          Cook -92.60723 45.16054
## 2        2 PointAddress   100 Silent Shores -89.37096 45.82521
## 3        3 PointAddress   100       Webster -91.47385 44.78840
## 4        4 PointAddress   100      Ironwood -88.08625 44.51026
## 5        5 PointAddress   100       Concord -88.50552 43.11230
## 6        6 PointAddress   100    Edinbourgh -88.18145 42.94171
##                                    full_addrs     county
## 1             2089 COOK DR, SOMERSET WI 54025-7514  ST. CROIX
## 2 4710 SILENT SHORES DR, RHINELANDER WI 54501-8649     ONEIDA
## 3       1912 WEBSTER AVE, EAU CLAIRE WI 54701-6647 EAU CLAIRE
## 4        2189 IRONWOOD DR, GREEN BAY WI 54304-1972      BROWN
## 5        20 S CONCORD RD, OCONOMOWOC WI 53066-2737   WAUKESHA
## 6  21401 W EDINBOURGH DR, NEW BERLIN WI 53146-4815   WAUKESHA
```

```
sum(is.na(wi_smpl_geocoded$X))
```

```
## [1] 0
```

```
length(which(wi_smpl_geocoded$X==0))
```

```
## [1] 0
```

We now have the data read in. Upon checking the coordinate field X via the is.na() and legnth(which()) commands, we see that there were no failures to match addresses to coordinates within our data. However, note that we have a normal datframe, not the necessary spatial object. However, thanks to the X and Y coordinates, we can easily convert these data to a spatial object with the SP package in R. First, store the X and Y columns into their own dataframe object, in this case titled wi_coords. Next, use the SpatialPointsDataFrame command to convert the data into a spatial dataframe. The required arguments are the coordinates object,

which will be used to create a mapped object, the data frame, which is the originally read in table, and the projection system, which as default should be the WGS84 as written out below. Note that there can be no NA coordinates present, or else the command will fail. Therefore, be certain to drop all missing information before hand.

```
library(sp)
```

```
## Warning: package 'sp' was built under R version 4.0.2
```

```
wi_smpl_geocoded <- subset(wi_smpl_geocoded, is.na(X)==FALSE & X!= 0)
wi_coords <- subset(wi_smpl_geocoded, select=c(X,Y))
wi_vf_spdf <-  SpatialPointsDataFrame(coords = wi_coords, data = wi_smpl_geocoded,
                                      proj4string = CRS("+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs8
class(wi_vf_spdf)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

We now see that we have the necessary spatial dataframe. The next step will be to overlay these data onto Census geographies. For the BISG process, we can make use of either tracts or census blocks for imputing race. For MEDSL purposes, we use tracts, given that there are fewer zero populated geographies, which can mess up the imputation. We will need a census polygon shape file, which can be found in the Healthy Elections Dropbox in the other data folder. Upon downloading the shape file, read it in with the readOGR command, which takes the arguments of the directory path, and the name of the component files that make up the shapefile. For memory saving purposes, let's also subset the geography to only Wisconsin, which can be done via the subset command and setting the STATE_FIPS to the Wisconsin FIPs code, 55. Additionally, we will want to ensure that the projection system for the cbgs object is the same as our voterfile point data, or else the eventual overlay will be impossible.

```
library(rgdal)
```

```
## rgdal: version: 1.4-8, (SVN revision 845)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.2.3, released 2017/11/20
##  Path to GDAL shared files: C:/Users/johna/Documents/R/win-library/4.0/rgdal/gdal
##  GDAL binary built with GEOS: TRUE
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: C:/Users/johna/Documents/R/win-library/4.0/rgdal/proj
##  Linking to sp version: 1.4-1
```

```
##commented out cmds are to read in the original shpfile
#cbgs <- readOGR("F:/shapefiles/cbgs", "CB2010")
#cbgs <- subset(cbgs, STATE_FIPS == "55")
cbgs <- readRDS("F:/voterfile/wi/cbgs_wi.Rdata")
cbgs<- spTransform(cbgs, CRS=CRS("+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0"))

class(cbgs)
```

```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

We now have the CBG data read in as a spatial dataframe, and the projection is the same as the wi_vf_spdf object. We can therefore finally move onto overlay the points onto the cbgs object and figure out the census demographics for each point. We will do this with the over() command from the sp package. This command

```
library(sp)
wi_vf_spdf$county <- over(wi_vf_spdf, cbgs)$STCOFIPS
wi_vf_spdf$county <- substr(wi_vf_spdf$county, 3,5)
head(wi_vf_spdf$county)
```

```
## [1] "109" "085" "035" "009" "133" "133"
```

```
sum(is.na(wi_vf_spdf$county)) #nothing missing
```

```
## [1] 0
```

```
wi_vf_spdf$tract <- over(wi_vf_spdf, cbgs)$TRACT
head(wi_vf_spdf$tract)
```

```
## [1] "120600" "970602" "000801" "940004" "204200" "201600"
```

```
sum(is.na(wi_vf_spdf$tract)) #nothing missing
```

```
## [1] 0
```

```
wi_vf_spdf$state <- "wi"
wi_vf_data <- wi_vf_spdf@data
```

We now have the tract and county ID fields from the census file, which gives us the necessary data to finally run BISG.

Step 4: BISG

For BISG to work, we need to load the wru package, and install it if not already. From there, create a census data object using the get_census_data() command, which makes use of the census API key to pull the necessary data. We only need the data for a single state, so specify the state abbreviation in upper case, and set age and sex to FALSE. From there, load in the full voterfile from earlier, specifying the full_addrs field if necessary for the purposes of merging. Given the internal logic of the command we will later run, make certain that the column with the last name information is renamed surname if not already. Then merge on the data from the spatial dataframe that contains the census ID info.

```
library(foreign)
library(wru)
library(stringr)
options(stringsAsFactors = FALSE)
###read in the census demographic data
###run this command to pull the data with the census API
#census.wi <- get_census_data(key = "b85306550d1fd788ddc045abfa6acf6ba7110abc",census.geo="tract",
  #                           state = c("WI"), age = FALSE, sex = FALSE)

census.wi <- readRDS("F:/voterfile/wi/census_wi.Rdata") #this is run for the purposes of the manual; us
wisconsin_vf <- read.csv("F:/voterfile/wi/wisconsin_voterfile_sample.csv")
colnames(wisconsin_vf)[colnames(wisconsin_vf)=="county"] <- "COUNTY_NAME"
class(wisconsin_vf)
```

```
## [1] "data.frame"
```

```
wisconsin_vf$full_addrs <- paste0(wisconsin_vf$address1, sep=", ", wisconsin_vf$address2)
wisconsin_vf$full_addrs <- str_to_upper(wisconsin_vf$full_addrs)
###change column name lastname to surname
colnames(wisconsin_vf)[colnames(wisconsin_vf)=="lastname"] <- "surname"
###merge on the census point data onto the full voterfile
```

```
wisconsin_vf <- merge(wisconsin_vf, wi_vf_data, by="full_addrs")
## check for the missing observations and subset
wi_leftover <- subset(wisconsin_vf, is.na(tract)==TRUE)
wisconsin_vf <- subset(wisconsin_vf, is.na(tract)==FALSE)
```

After checking to ensure that there is no missing information, finally run the command predict_race(), where the user will need to specify the census.data, level of geography, voter.file, and whether age or sex information will be included. If there is any data where there is missing census ID information due to geocoding problems, subset these into a leftover data frame, and use predict_race() with the surname.only set as TRUE. Upon running the predict_Race command, the output are several fields that are as follows:

predict_race() output

- pred.whi - The predicted probability (0 - 1 scale) that the individual is White.
- pred.bla - The predicted probability (0 - 1 scale) that the individual is Black.
- pred.his - The predicted probability (0 - 1 scale) that the individual is Hispanic.
- pred.asi - The predicted probability (0 - 1 scale) that the individual is Asian.
- pred.oth - The predicted probability (0 - 1 scale) that the individual is of some other race.

```
###run the BISG command

wisconsin_vf <- predict_race(voter.file = wisconsin_vf, census.geo = "tract", census.data = census.wi,
                      age = FALSE, sex = FALSE)
```

```
## [1] "Proceeding with Census geographic data at tract level..."
## [1] "Using Census geographic data from provided census.data object..."
```

```
## Warning in merge_surnames(voter.file): Probabilities were imputed for 105
## surnames that could not be matched to Census list.
```

```
## [1] "State 1 of 1: WI"
```

```
wisconsin_vf$surnameonly <- 0
head(wisconsin_vf)
```

```
##                                 full_addrs voterregnumber    surname
## 1  1389 RAIN DANCE TRL, NEKOOSA WI 54457-8696       51002116       GENZ
## 2          827 18TH LN, ARKDALE WI 54613-9779      700838332   LEPINSKI
## 3     305 E 3RD ST APT 2, FRIENDSHIP WI 53934      700866789     HARPER
## 4      1200 11TH AVE W, ASHLAND WI 54806-3744        4007087      SIREK
## 5 213 W MICHIGAN ST APT 3, BUTTERNUT WI 54514      700881049 WALTENOSKY
## 6         2730 26 1/2 AVENUE, MIKANA WI 54857        4033483      WEISS
##   firstname voterstatus voterstatusreason               address1
## 1    Roland      Active        Registered     1389 RAIN DANCE TRL
## 2   Kenneth      Active        Registered             827 18TH LN
## 3    Janene      Active        Registered      305 E 3RD ST APT 2
## 4     April      Active        Registered         1200 11TH AVE W
## 5      Leah      Active        Registered 213 W MICHIGAN ST APT 3
## 6     KELLY      Active        Registered      2730 26 1/2 AVENUE
##                address2 ballotdeliverymethod ballotstatusreason
## 1 NEKOOSA WI 54457-8696                Mail           Returned
## 2 ARKDALE WI 54613-9779                Mail           Returned
## 3   FRIENDSHIP WI 53934                Mail           Returned
## 4 ASHLAND WI 54806-3744                Mail           Returned
```

```
## 5    BUTTERNUT WI 54514      Voted In Person           Returned
## 6       MIKANA WI 54857                 Mail           Returned
##   ballotreasontype                                        electionname
## 1                  2020 Spring Election and Presidential Preference Vote
## 2                  2020 Spring Election and Presidential Preference Vote
## 3                  2020 Spring Election and Presidential Preference Vote
## 4                  2020 Spring Election and Presidential Preference Vote
## 5                  2020 Spring Election and Presidential Preference Vote
## 6                  2020 Spring Election and Presidential Preference Vote
##   COUNTY_NAME ObjectID     Loc_name Score     StName         X        Y county
## 1       ADAMS      346 PointAddress   100 Rain Dance -89.83567 44.20429    001
## 2       ADAMS       80 PointAddress   100       18th -89.93447 44.13847    001
## 3       ADAMS      523 PointAddress   100        3rd -89.81379 43.97125    001
## 4     ASHLAND      984 PointAddress   100       11th -90.88756 46.57751    003
## 5     ASHLAND      451 PointAddress   100   Michigan -90.49471 46.01330    003
## 6      BARRON      557       Postal   100            -91.61612 45.58600    005
##    tract state  pred.whi     pred.bla    pred.his     pred.asi    pred.oth
## 1 950100    wi 0.9878009 0.0000488547 0.002487171 0.0023885584 0.007274513
## 2 950202    wi 0.9861946 0.0003819906 0.007297621 0.0018384382 0.004287336
## 3 950400    wi 0.9016017 0.0527894087 0.014762806 0.0011751189 0.029670928
## 4 950300    wi 0.9614566 0.0003559463 0.001858334 0.0017048565 0.034624305
## 5 950700    wi 0.9379006 0.0015148643 0.016476912 0.0095794807 0.034528162
## 6 000100    wi 0.9905681 0.0001999805 0.003253083 0.0004630271 0.005515806
##   surnameonly
## 1           0
## 2           0
## 3           0
## 4           0
## 5           0
## 6           0
```

```
##run these is there is missing tract info
#wi_leftover <- predict_race(wi_leftover, surname.only = TRUE)
#wi_leftover$surnameonly <- 1

###binding the data together
#wisconsin_vf <- rbind(wisconsin_vf,wi_leftover)

saveRDS(wisconsin_vf, "wi_smpl_bisg.Rdata")
write.csv(wisconsin_vf, "wi_smpl_bisg.csv",row.names = FALSE)
```

We now finally have our results! From here, make certain to save the data, lest you have to rerun all of this again. Finally, when using the output, be certain to sum the predicted racial probabilities up to a level of geography at the county/jurisdiction level or higher. That is because the purpose of BISG is to correct for biases inherent to ecological inference. With these results, the user will be able to more accurately estimate racial turnout and its effect on election outcomes of interest. From there, run the requested analyses of interest.

Conclusion

While the BISG process is lengthy, it offers far higher quality data than the quicker alternatives. Note that the process entirely depends upon quality address information. Some states, especially addresses from college residence halls,homeless shelters, Native American reservations, or very rural areas, can lead to problems with imputation. That said, the end product is the best option compared to other alternatives. Several steps

to catch some of these issues were glossed over by instead offering the predict_race() command with the surname.only option set as TRUE. For more information, see the appendix to this manual for instructions on alternative geocoding steps, issues with large data and more. For any questions, please contact John A. Curiel at jcuriel.mit.edu. I hope that this manual as helpful in your BISG adventures!