Dealing With Historical US Election Data - Part 2

A comprehensive guide to processing, cleaning, and analyzing complex data

Roee Diler

2025-03-23

Introduction

Background and Importance

Welcome back to the second part of the guide on validating historical US election data. In the previous section, we covered the initial steps of data processing, including importing the raw ICPSR data and performing basic data cleaning tasks. In this section, we will continue the data cleaning process by addressing more complex issues such as missing values, duplicate entries, and data anomalies.

Working with historical election data presents several challenges:

1. Inconsistent formatting across different time periods
2. Missing or incorrectly coded values
3. Potential data entry errors in the original sources

This guide demonstrates a systematic approach to process, validate, and prepare this data for reliable analysis.

Purpose of This Guide

This document will walk through:

- Identifying and handling missing or problematic data points
- Validating data integrity through systematic checks
- Preparing clean, consistent datasets for further analysis
- Documenting data limitations and considerations for interpretation

By following these steps, researchers can ensure the highest possible data quality for their electoral analyses.

Required Packages

We load the necessary libraries using `pacman` for package management.

```
rm(list = ls())  # Clear workspace

library(pacman)
p_load(data.table, tidyverse, naniar, Hmisc, rlist, labelled)
```

Data Setup

Define Paths

We set up directory paths dynamically based on the script location.

```
# Get the directory where this script is located
script_path <- dirname(rstudioapi::getSourceEditorContext()$path)
basic_path <- script_path
data_path <- file.path(basic_path, "data")
icpsr_path <- file.path(data_path, "ICPSR/ICPSR_00001")
```

Loading Raw Data

We begin by importing the previously processed long-format ICPSR data. This file contains election results that have been extracted from the original ASCII files and converted to a more accessible format.

```
# Load the raw ICPSR data that was previously processed
icpsr_data <- fread(paste0(data_path, "/icpsr_long_raw.csv"))

# Display the first few rows of the dataset
head(icpsr_data)
```

variable <chr>	V1 <int>	V2 <chr>	V3 <int>	value <int>	source <chr>	column_name <chr>	values <chr>
V10	1	FAIRFIELD	10	98	DS0001	CONG_DIST_NUMBER_1835	0000099
V10	1	HARTFORD	30	98	DS0001	CONG_DIST_NUMBER_1835	0000099
V10	1	LITCHFIELD	50	98	DS0001	CONG_DIST_NUMBER_1835	0000099

variable <chr>	V1 <int>	V2 <chr>	V3 <int>	value <int>	source <chr>	column_name <chr>	values <chr>
V10	1	MIDDLESEX	70	98	DS0001	CONG_DIST_NUMBER_1835	0000099
V10	1	NEW HAVEN	90	98	DS0001	CONG_DIST_NUMBER_1835	0000099
V10	1	NEW LONDON	110	98	DS0001	CONG_DIST_NUMBER_1835	0000099

6 rows

Dataset dimensions: 4135212 rows and 8 columns

Data Cleaning and Manipulation

Remove Columns

First, we remove any congressional district columns as our main focus is on state and county-level election data.

```
# Remove congressional district columns
icpsr_data <- icpsr_data[!(str_detect(column_name, "CONG_DIST_NUMBER")), ]
```

Removed 211214 rows containing congressional district data

Rename Columns for Clarity

We rename columns to make the data more intuitive and easier to work with. We will rename key columns in the standard format of NHGIS data for advanced analysis in the future.

```
# Rename columns to more descriptive names
setnames(icpsr_data,
         old = c("variable", "V1", "V2", "V3", "value", "values"),
         new = c("column_number", "ICPSRST", "ICPSRNAM",
                 "ICPSRCTY", "VOTES", "missing_value"))
```

Identify Duplicate Entries

We create a unique identifier for each data point to help identify duplicates that might skew our analysis. Each data point is identified by a combination of data source (the original data set from ICPSR), state, county, and column number. If there is more than one entry for the same identifier, we flag it as a duplicate and investigate further.

```
# Create identifiers for each unique data point
icpsr_data <- icpsr_data[, ids := seq_len(.N),
                         by = .(source, ICPSRST, ICPSRNAM, ICPSRCTY, column_number)]

# Identify columns with duplicate entries
duplicate_columns <- icpsr_data[ids > 1, .(column_number, source)] %>% unique()

# Display columns with duplicates
duplicate_columns
```

column_number <chr>	source <chr>
V290	DS0042
V163	DS0046

2 rows

```
rm(duplicate_columns)
```

Examine Specific Problematic Cases

We examine specific columns that have been identified as problematic based on the duplicate analysis.

```
# Define conditions for problematic columns
cond1 <- (icpsr_data$column_number == "V290" &
            icpsr_data$source == "DS0042")
cond2 <- (icpsr_data$column_number == "V163" &
            icpsr_data$source == "DS0046")

# Display details about these problematic columns
problem_info <- icpsr_data[cond1 | cond2,
                           .(source, column_number, column_name, missing_value)] %>%
                unique()

problem_info
```

source <chr>	column_number <chr>	column_name <chr>	missing_value <chr>
DS0042	V290	X884_PRES_0100_VOTE	9999999
DS0042	V290	X884_PRES_0100_VOTE	9900000 thru highest
DS0046	V163	X938_GOV_0100_VOTE	9999999
DS0046	V163	X938_GOV_0100_VOTE	9900000 thru highest

4 rows

```
rm(problem_info, cond1, cond2)
```

As we can see, the identified problematic columns have duplicate entries that need to be resolved. Some of the missing values are coded as “9900000 thru highest,” which requires special handling. We will address these issues in the following section. Now, let’s examine the overall missing value patterns in the dataset.

The missing values represent the value of the votes that should be counted as missing. Usually, the missing values are coded as the maximum number that the data point can store (e.g. 9999999), but not always, so let’s check the unique missing values in the dataset.

```
# Display unique missing value codes
unique(icpsr_data$missing_value)
```

```
## [1] "9999999"              "9900000 thru highest" "999999"
```

So we have two numeric missing values options and one range, let’s handle the range first and see if any other columns beside what we already identified have this range.

```
# Check for values that exceed the "9900000 thru highest" missing value code
high_value_cases <- icpsr_data[missing_value == "9900000 thru highest",
                               .(source, column_number, column_name, missing_value)] %>%
                unique()
high_value_cases
```

source <chr>	column_number <chr>	column_name <chr>	missing_value <chr>
DS0042	V290	X884_PRES_0100_VOTE	9900000 thru highest
DS0046	V163	X938_GOV_0100_VOTE	9900000 thru highest

2 rows

```
rm(high_value_cases)
```

Alright, we have already identified those problematic columns, let’s remove them and convert the missing values to numeric.

Handle Missing Values

We examine and handle special missing value codes to ensure data quality.

```
# Check for values that exceed the "9900000 thru highest" missing value code
icpsr_data[missing_value == "9900000 thru highest" & VOTES >= 9900000, VOTES] %>% unique()
```

```
## [1] 9999999
```

As we can see, the only values of `VOTES` that exceed the missing value threshold 9900000 are 9999999 and we already have those missing values in other rows, so we can safely remove the rows with the missing value “9900000 thru highest”.

```
# Remove these problematic cases
icpsr_data <- icpsr_data[missing_value != "9900000 thru highest", ]

# Clean up and convert missing values to numeric
icpsr_data[, ":=" (ids = NULL,
                   missing_value = as.numeric(missing_value))]
```

Data Validation: Missing Value Checks

Now that we checked our missing values, let’s perform additional validation to ensure data quality.

```
icpsr_data[order(VOTES, decreasing = TRUE), ] %>%
  filter(VOTES != missing_value) %>%
  select(VOTES, missing_value) %>%
  unique()
```

VOTES <int>	missing_value <dbl>
99999999	9999999
99999990	9999999
9999990	9999999
5219597	9999999
4049514	9999999
3793582	9999999
3163181	9999999
3089502	9999999
3068269	9999999
2824196	9999999

1-10 of 10,000 rows Previous 1 2 3 4 5 6 … 1000 Next

We can see that there are some errors in the data where the votes exceed the missing value threshold and some cases that the votes are just slightly below the missing value threshold. These cases can be considered as missing values as well.

Let’s see how many errors like this we have in the data.

```
# Check for values that exceed missing value thresholds
exceeded_missing <- icpsr_data[(VOTES > missing_value),]
```

Found 49 rows with data points that exceed their missing value.

```
# Check for borderline values
borderline_values <- icpsr_data[(VOTES == 9999990),]
```

Found 2 rows with data points just below the missing value.

Now, let's examine the other missing value threshold - "999999".

```
icpsr_data[order(VOTES, decreasing = TRUE), ] %>%
  filter(VOTES != missing_value & between(VOTES, 999990, 999999)) %>%
  select(VOTES, missing_value) %>%
  unique()
```

VOTES <int>	missing_value <dbl>
999999	9999999

1 row

We can see that there are some cases that the missing value threshold is probably a mistake and should be 999999.

```
icpsr_data[VOTES > 999999 , .(missing_value)] %>% unique()
```

missing_value <dbl>
9999999

1 row

We don't have any cases where the missing value threshold is lower than the actual missing value.

Let's examine the percentage of total data points that should be converted to missing value.

```
# Calculate what percentage of data points are near missing value thresholds
near_missing_pct <- nrow(icpsr_data[(VOTES >= 9999990 | VOTES == 999999),]) / nrow(icpsr_da
ta)
near_missing_pct <- round(near_missing_pct * 100, 2)
```

Percentage of data points that will be converted to missing value: 19.24%

Apply Corrections to Borderline Values

We convert suspicious values that are near missing value thresholds to NA, as they likely represent missing data rather than actual vote counts.

```
# Convert suspicious values to NA
icpsr_data[(VOTES >= 9999990 | VOTES == 999999), VOTES := NA]

# Remove the missing_value column as it's no longer needed
icpsr_data[, missing_value := NULL]
```

Columns Names

Convert Column Names to Separate Fields

When the data was in wide format each column contain votes that a specific party got in a specific year in election for a specific office. From the ICPSR website we know how to interpret the column names:

> Some variable labels are cryptic.
>
> For example, the variable label V34="825 2 G GOV 9001 VOTE" is interpreted as:
>
> 1. year of election
> 2. code for elected office
> 3. type of election
> 4. elected office
> 5. party code.

Note: the "VOTE" in the end of the column name simply means that this column contain votes rather than other data so we can discard it.

Let's look at some columns' names in our data:

```
# Display a random sample of column names
sample(unique(icpsr_data$column_name), 5)
```

```
## [1] "X876_3_G_CONG_0616_VOTE" "X884_3_G_CONG_9001_VOTE"
## [3] "X908_3_S_CONG_9003_VOTE" "X861_3_S_CONG_9003_VOTE"
## [5] "X859_3_G_CONG_0329_VOTE"
```

We can see that the column names follow the format described above with underscore as a separator.

Split Column Names

Now if we split the column names by the underscore we expect to get a vector with 6 strings. Let's check if this is the case.

```
col_name_data <- tstrsplit(icpsr_data$column_name, "_")

col_name_data <- as.data.table(col_name_data)

# Display the number of columns:
ncol(col_name_data)
```

```
## [1] 7
```

The number of columns in the split data table is 7. Which is not according to our expectations. Let's investigate further.

```
# count the number of NA's in each column:
colSums(is.na(col_name_data))
```

```
##      V1      V2      V3      V4      V5      V6      V7
##       0       0       0       0   39828   49248 3662054
```

We have NA values only the the three last columns.

Let's examine the last column.

```
col_name_data[!is.na(V7) , ] %>% unique()
```

V1 <chr>	V2 <chr>	V3 <chr>	V4 <chr>	V5 <chr>	V6 <chr>	V7 <chr>
X829	3	M	H	AL	9003	VOTE
X829	3	M	H	AL	9004	VOTE
X829	3	M	H	AL	9005	VOTE
X829	3	M	H	AL	9006	VOTE
X829	3	M	H	AL	9007	VOTE
X829	3	M	H	AL	9008	VOTE
X829	3	M	H	AL	9009	VOTE
X829	3	M	H	AL	9010	VOTE
X829	3	M	H	AL	9011	VOTE
X829	3	M	H	AL	9012	VOTE

1-10 of 1,911 rows Previous 1 2 3 4 5 6 … 192 Next

As we can see, there is one additional underscore in some of the column names. It seems that we have the right components, the three first columns are year, office code and election type. The two last columns are party code and the final "VOTE" but the office name has an extra underscore. Let's fix this issue.

HAL is a common abbreviation for “House of Assembly” in historical election data. We will fix this issue to maintain consistency, but first let’s make sure that this is the only problem.

```
apply(col_name_data[!is.na(V7) , ], 2, unique)
```

```
## $V1
##  [1] "X829" "X831" "X833" "X834" "X835" "X836" "X825" "X827" "X906" "X932"
## [11] "X934" "X938" "X940" "X942" "X908" "X910" "X948" "X950" "X952" "X954"
## [21] "X956" "X958" "X960" "X944" "X962" "X946" "X882" "X837" "X839" "X841"
## [31] "X936" "X966" "X968" "X964" "X838" "X840" "X842" "X844" "X846" "X850"
## [41] "X852" "X854" "X856" "X858" "X860" "X862" "X863" "X864" "X866" "X868"
## [51] "X870" "X872" "X824" "X874" "X876" "X826" "X828" "X830" "X832" "X886"
## [61] "X888" "X890" "X892" "X896" "X898" "X900" "X902" "X904" "X912" "X914"
## [71] "X916" "X918" "X878" "X920" "X880" "X884" "X922" "X924" "X926" "X928"
## [81] "X930" "X894" "X921" "X889" "X891" "X848" "X851" "X845" "X843" "X873"
## [91] "X875" "X859" "X959"
##
## $V2
## [1] "3"
##
## $V3
## [1] "M" "W" "S" "G"
##
## $V4
## [1] "H"
##
## $V5
## [1] "AL"
##
## $V6
##   [1] "9003"  "9004"  "9005"  "9006"  "9007"  "9008"  "9009"  "9010"  "9011"
##  [10] "9012"  "9013"  "9014"  "9015"  "9016"  "9017"  "9018"  "9019"  "9020"
##  [19] "9021"  "9022"  "TOTAL" "0025"  "0100"  "9001"  "0103"  "9002"  "0026"
##  [28] "0029"  "0526"  "9999"  "2020"  "0200"  "0361"  "0380"  "0505"  "0331"
##  [37] "0543"  "0747"  "0745"  "0746"  "0501"  "0553"  "0748"  "0546"  "0341"
##  [46] "0328"  "0744"  "0742"  "0743"  "0320"  "0843"  "1193"  "1299"  "0370"
##  [55] "0570"  "1246"  "0646"  "1442"  "1447"  "0728"  "0300"  "0310"  "0037"
##  [64] "0604"  "0208"  "0605"  "0953"  "0326"  "0001"  "0012"  "0101"  "1101"
##  [73] "0524"  "1432"  "1012"  "1168"  "1169"  "0511"  "0591"  "0544"  "0759"
##  [82] "0512"  "0537"  "1104"  "0594"  "0631"  "0120"  "1105"  "1166"  "0013"
##  [91] "0020"  "1346"  "1387"  "0948"  "0402"  "0621"  "0793"  "0794"  "0795"
## [100] "0522"  "0700"  "0781"  "0796"  "0799"  "0800"  "0801"  "0802"  "0803"
## [109] "0804"  "0805"  "0112"  "0618"  "0667"  "0718"  "0867"  "0868"  "0869"
## [118] "0870"  "0877"  "0788"  "0855"  "0856"  "0857"  "0858"  "0859"  "0860"
## [127] "0861"  "0862"  "0863"  "0864"  "0865"  "0866"  "0871"  "0872"  "0873"
## [136] "0874"  "0876"  "0330"  "0650"  "0651"  "0652"  "1247"  "1252"  "1253"
## [145] "0211"  "0506"  "0620"  "0622"  "0655"  "1256"  "1257"  "1270"  "1271"
## [154] "1272"  "0572"  "0623"  "0624"  "0625"  "1204"  "0560"  "0562"  "0563"
## [163] "0564"  "1140"  "9023"  "9024"  "9025"  "0559"  "0565"  "0579"  "0583"
## [172] "0584"  "0585"  "0586"  "0587"  "0588"  "0589"  "0590"  "0592"  "0593"
## [181] "1263"  "1266"  "0519"  "0596"  "0597"  "1220"  "1265"  "0615"  "0617"
## [190] "0557"  "0721"  "1273"  "0538"  "0539"  "0540"  "0541"  "0542"  "0545"
## [199] "1269"  "1404"  "1411"  "0342"  "1013"  "1337"  "0880"  "1276"  "1277"
## [208] "0532"  "0897"  "0324"  "0930"  "0340"  "0909"  "0404"  "0114"  "1390"
## [217] "1391"  "1069"  "0048"  "1240"  "1095"  "1096"  "0908"  "1285"  "0354"
## [226] "0644"  "0346"  "0940"  "0608"  "1011"  "0517"  "1063"  "1111"  "0536"
## [235] "0573"  "1377"
```

```
##
## $V7
## [1] "VOTE"
```

It seems that the only issue is the extra underscore in the office name. Let's fix this issue.

```
col_name_data[!is.na(V7) , ":=" (V4 = "HAL", V5 = V6, V6 = V7, V7 = NA)]

col_name_data[!is.na(V7) , ] %>% nrow()
```

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

```
col_name_data[ , V7 := NULL]
```

Now that we have the correct number of columns, let's look into the unique values of each column.

First column - Year of Election

```
unique(col_name_data$V1)
```

```
##   [1] "X829"  "X830"  "X831"  "X832"  "X833"  "X834"  "X835"  "X836"  "X837"
##  [10] "X824"  "X838"  "X839"  "X840"  "X841"  "X842"  "X843"  "X844"  "X845"
##  [19] "X825"  "X846"  "X847"  "X848"  "X849"  "X850"  "X851"  "X852"  "X853"
##  [28] "X854"  "X855"  "X856"  "X857"  "X858"  "X859"  "X860"  "X826"  "X827"
##  [37] "X828"  "X912"  "X914"  "X916"  "X918"  "X920"  "X922"  "X906"  "X924"
##  [46] "X926"  "X928"  "X930"  "X932"  "X934"  "X936"  "X938"  "X908"  "X940"
##  [55] "X942"  "X910"  "X948"  "X950"  "X952"  "X944"  "X954"  "X956"  "X958"
##  [64] "X960"  "X962"  "X964"  "X966"  "X968"  "X946"  "X861"  "X862"  "X863"
##  [73] "X864"  "X865"  "X866"  "X867"  "X868"  "X869"  "X870"  "X871"  "X872"
##  [82] "X873"  "X880"  "X882"  "X884"  "X886"  "X888"  "X874"  "X890"  "X892"
##  [91] "X894"  "X896"  "X898"  "X900"  "X902"  "X875"  "X876"  "X877"  "X878"
## [100] "X879"  "X904"  "X881"  "X883"  "X885"  "X887"  "X889"  "X891"  "X893"
## [109] "X895"  "X901"  "X903"  "X905"  "X907"  "X909"  "X911"  "X913"  "X915"
## [118] "X917"  "X919"  "X897"  "X899"  "X823"  "X931"  "X937"  "X943"  "X949"
## [127] "X953"  "X925"  "X961"  "X965"  "X957"  "X929"  "X923"  "X933"  "X927"
## [136] "X947"  "X951"  "X941"  "X955"  "X959"  "X921"  "X939"  "X945"  "X935"
## [145] "X1827" "X963"  "X967"  "X92"
```

It seems that the first column contains the year of the election. Let's convert it to a numeric format for further analysis.

```
# make sure that the first character is "X" and replace it with "1"
col_name_data$V1 %>% str_detect("^X") %>% all()
```

```
## [1] TRUE
```

```
col_name_data[ , V1 := as.numeric(gsub("X", "1", V1))]

col_name_data$V1 %>% unique() %>% sort()
```

```
##   [1]   192  1823  1824  1825  1826  1827  1828  1829  1830  1831  1832  1833
##  [13]  1834  1835  1836  1837  1838  1839  1840  1841  1842  1843  1844  1845
##  [25]  1846  1847  1848  1849  1850  1851  1852  1853  1854  1855  1856  1857
##  [37]  1858  1859  1860  1861  1862  1863  1864  1865  1866  1867  1868  1869
##  [49]  1870  1871  1872  1873  1874  1875  1876  1877  1878  1879  1880  1881
##  [61]  1882  1883  1884  1885  1886  1887  1888  1889  1890  1891  1892  1893
##  [73]  1894  1895  1896  1897  1898  1899  1900  1901  1902  1903  1904  1905
##  [85]  1906  1907  1908  1909  1910  1911  1912  1913  1914  1915  1916  1917
##  [97]  1918  1919  1920  1921  1922  1923  1924  1925  1926  1927  1928  1929
## [109]  1930  1931  1932  1933  1934  1935  1936  1937  1938  1939  1940  1941
## [121]  1942  1943  1944  1945  1946  1947  1948  1949  1950  1951  1952  1953
## [133]  1954  1955  1956  1957  1958  1959  1960  1961  1962  1963  1964  1965
## [145]  1966  1967  1968 11827
```

We can see two issues here:

1. The smallest value is 192, which is incorrect. We should have years starting from 1824 (1823 for one specific case).
2. The largest value is 11827, which is also incorrect. We should have years up to 1968.

Let's investigate these issues further:

Incorrect Minimum Year

```
col_name_data[V1 == 192, ] %>% unique()
```

V1 <dbl>	V2 <chr>	V3 <chr>	V4 <chr>	V5 <chr>	V6 <chr>
192	G	PRES	0100	VOTE	*NA*

1 row

```
# look for this election in the original data
icpsr_data[column_name == "X92_G_PRES_0100_VOTE", .(source, column_number)] %>% unique()
```

source <chr>	column_number <chr>
DS0172	V65

1 row

```
icpsr_data[source == "DS0172" & column_number %in% paste0("V", 63:67),
           .(column_number, column_name)] %>% unique()
```

column_number <chr>	column_name <chr>
V63	X918_5_G_SEN_9999_VOTE
V64	X918_5_G_SEN_TOTAL_VOTE
V65	X92_G_PRES_0100_VOTE
V66	X920_1_G_PRES_0200_VOTE
V67	X920_1_G_PRES_0328_VOTE

5 rows

It seems that the issue here is a typo in the original data. The correct year should be 1920 by the order of the years in the previous and next columns. In addition, the elected office code is missing as well. Let's fix this issue.

```
col_name_data[V1 == 192, ":=" (V1 = 1920, V2 = 1, V3 = V2, V4 = V3, V5 = V4, V6 = V5)]
```

Incorrect Maximum Year

```
col_name_data[V1 == 11827, ] %>% unique()
```

V1 <dbl>	V2 <chr>	V3 <chr>	V4 <chr>	V5 <chr>	V6 <chr>
11827	2	G	GOV	9001	VOTE
11827	2	G	GOV	9999	VOTE
11827	2	G	GOV	TOTAL	VOTE

3 rows

```
# look for this election in the original data
icpsr_data[str_detect(column_name, "X1827"), .(source, column_number)] %>% unique()
```

source <chr>	column_number <chr>
DS0126	V49
DS0126	V50
DS0126	V51

3 rows

```
icpsr_data[source == "DS0126" & column_number %in% paste0("V", 45:55),
           .(column_number, column_name)] %>% unique()
```

column_number <chr>	column_name <chr>
V45	X825_2_G_GOV_TOTAL_VOTE
V46	X826_3_G_CONG_9001_VOTE
V47	X826_3_G_CONG_9002_VOTE
V48	X826_3_G_CONG_TOTAL_VOTE
V49	X1827_2_G_GOV_9001_VOTE
V50	X1827_2_G_GOV_9999_VOTE
V51	X1827_2_G_GOV_TOTAL_VOTE
V52	X827_3_S_H_AL_9001_VOTE
V53	X827_3_S_H_AL_9002_VOTE
V54	X827_3_S_H_AL_TOTAL_VOTE

1-10 of 11 rows Previous 1 2 Next

Once again, it seems that the issue is a typo in the original data. The correct year should be 1827. Let's fix this issue.

```
col_name_data[V1 == 11827, V1 := 1827]

col_name_data$V1 %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1823    1882    1916    1910    1942    1968
```

Now that is better!

Let's move on to the second column.

Second column - Elected Office Code

This column should contain the code for the elected office. There is no official codebook for this dataset but we can assume that the values should be numeric based on the comment above and the range should be like in other related dataset in the ICPSR.

So for example, there is the *Candidate Name and Constituency Totals, 1788-1990 (ICPSR 2)*, in which we found the following values for the office code:

- 1 - President
- 2 - Governor
- 3 - US Rep.
- 4 - Senate 1
- 5 - Senate 2
- 6 - Senate 3
- 7 - State Offices

```
unique(col_name_data$V2)
```

```
##  [1] "3"    "2"    "1"    "6"    "4"    "5"    "7"    "PRES" "GOV"  "SEN"
```

Not all values are numeric as expected, we have a values “PRES”, “GOV”, and “SEN” which should be on the forth column.

Let’s fix these issues.

Misplaced Values of Elected Office

```
col_name_data[V2 %in% c("PRES", "GOV", "SEN"), ] %>% apply(2, unique)
```

```
## $V1
##  [1] "1836" "1838" "1840" "1842" "1844" "1846" "1848" "1824" "1850" "1852"
## [11] "1854" "1826" "1828" "1830" "1832" "1834" "1866" "1868" "1870" "1872"
## [21] "1874" "1876" "1856" "1879" "1880" "1882" "1884" "1858" "1860" "1862"
## [31] "1864" "1892" "1894" "1896" "1898" "1900" "1885" "1902" "1904" "1906"
## [41] "1888" "1891" "1914" "1908" "1916" "1918" "1920" "1922" "1910" "1912"
## [51] "1928" "1930" "1932" "1924" "1926" "1934" "1936" "1938" "1944" "1946"
## [61] "1940" "1948" "1949" "1950" "1952" "1942" "1962" "1964" "1966" "1954"
## [71] "1956" "1958" "1960"
##
## $V2
## [1] "PRES" "GOV"  "SEN"
##
## $V3
##  [1] "0029"  "0100"  "OTHER" "TOTAL" "0526"  "0914"  "0300"  "0915"  "0918"
## [10] "0921"  "0310"  "1287"  "1288"  "1289"  "0025"  "0101"  "0916"  "0553"
## [19] "0200"  "0208"  "0361"  "0320"  "0115"  "1085"  "0107"  "0922"  "0341"
## [28] "0505"  "0340"  "1094"  "0524"  "0945"  "0511"  "0120"  "0826"  "0380"
## [37] "0754"  "0502"  "0517"  "9001"  "9002"  "0370"  "6"     "0741"  "0764"
## [46] "4"     "0537"  "0797"  "0798"  "0544"  "0832"  "0543"  "0793"  "0541"
## [55] "0631"  "0522"  "0618"  "0718"  "0328"  "0402"  "0646"  "0112"  "0637"
##
## $V4
##  [1] "VOTE"  "0100"  "0200"  "0361"  "0370"  "0380"  "0505"  "0754"  "OTHER"
## [10] "TOTAL" "0314"  "0766"  "0537"  "0797"  "0798"  "0631"  "0543"  "0793"
## [19] "0331"  "0522"  "0618"  "0718"  "0570"  "0402"  "0646"  "0112"  "0637"
##
## $V5
## [1] NA     "VOTE"
##
## $V6
## [1] NA
```

We can see that some of the rows are missing both the elected office code and the election type, while others have the elected office code in the wrong column. We can divide these cases using the “V5” column, if a column name is missing both the code and type - the “V5” column will be NA.

Let's make sure that this cases are completely differentiable.

```
# unique values in cases with missing values in the V5 column
col_name_data[V2 %in% c("PRES", "GOV", "SEN") & is.na(V5), ] %>% apply(2, unique)
```

```
## $V1
##  [1] "1836" "1838" "1840" "1842" "1844" "1846" "1848" "1824" "1850" "1852"
## [11] "1854" "1826" "1828" "1830" "1832" "1834" "1866" "1868" "1870" "1872"
## [21] "1874" "1876" "1856" "1879" "1880" "1882" "1884" "1858" "1860" "1862"
## [31] "1864" "1892" "1894" "1896" "1898" "1900" "1885" "1902" "1904" "1906"
## [41] "1888" "1891" "1914" "1908" "1916" "1918" "1920" "1922" "1910" "1912"
## [51] "1930" "1932" "1924" "1926" "1928" "1934" "1936" "1938" "1944" "1946"
## [61] "1940" "1948" "1950" "1952" "1942" "1964" "1966" "1954" "1956" "1958"
## [71] "1960" "1962"
##
## $V2
## [1] "PRES" "GOV"
##
## $V3
##  [1] "0029"  "0100"  "OTHER" "TOTAL" "0526"  "0914"  "0300"  "0915"  "0918"
## [10] "0921"  "0310"  "1287"  "1288"  "1289"  "0025"  "0101"  "0916"  "0553"
## [19] "0200"  "0208"  "0361"  "0320"  "0115"  "1085"  "0107"  "0922"  "0341"
## [28] "0505"  "0340"  "1094"  "0524"  "0945"  "0511"  "0120"  "0826"  "0380"
## [37] "0754"  "0502"  "0517"  "9001"  "9002"  "0370"  "0741"  "0764"  "0537"
## [46] "0797"  "0798"  "0544"  "0832"  "0543"  "0793"  "0541"  "0631"  "0522"
## [55] "0618"  "0718"  "0328"  "0402"  "0646"  "0112"  "0637"
##
## $V4
## [1] "VOTE"
##
## $V5
## [1] NA
##
## $V6
## [1] NA
```

We can see that all of the cases with missing values in the "V5" column are missing both the office code and the election type and the elected office are "PRES" or "GOV". We will adjust the data to the correct columns.

```
# unique values in cases with values in the V5 column
col_name_data[V2 %in% c("PRES", "GOV", "SEN") & !is.na(V5), ] %>% apply(2, unique)
```

```
## $V1
##  [1] "1914" "1916" "1920" "1922" "1928" "1932" "1926" "1934" "1938" "1944"
## [11] "1946" "1949" "1950" "1952" "1940" "1962" "1964" "1956" "1958"
##
## $V2
## [1] "SEN"
##
## $V3
## [1] "6" "4"
##
## $V4
##  [1] "0100"  "0200"  "0361"  "0370"  "0380"  "0505"  "0754"  "OTHER" "TOTAL"
## [10] "0314"  "0766"  "0537"  "0797"  "0798"  "0631"  "0543"  "0793"  "0331"
## [19] "0522"  "0618"  "0718"  "0570"  "0402"  "0646"  "0112"  "0637"
##
## $V5
## [1] "VOTE"
##
## $V6
## [1] NA
```

We can see that all of the cases with values in the “V5” column have the office code in the wrong column and the elected office is “SEN”. We will adjust the data to the correct columns.

```
# Change the order of the columns for cases with missing values in the V5 column
col_name_data[V2 %in% c("PRES", "GOV") ,
  ":=" (V2 = NA, V3 = NA, V4 = V2, V5 = V3, V6 = V4)]
```

```
# Change the order of the columns for cases with values in the V5 column
col_name_data[V2 %in% c("SEN") ,
  ":=" (V2 = V3, V3 = NA, V4 = V2, V5 = V4, V6 = V5)]
```

Now that we have fixed the misplaced values, let’s convert the column to numeric.

```
col_name_data[ , V2 := as.numeric(V2)]

col_name_data$V2 %>% unique()
```

```
## [1]  3  2  1  6  4  5  7 NA
```

Third column - Election Type

This column should contain the type of election. From the example above and the data we saw already we can assume that the values should be strings, specifically one character strings.

Here we found two related datasets in the ICPSR website that tell us what to expect in this column:

From the same dataset as before (ICPSR 2):

- G - General

- M - Multiple G election
- S - Special
- W - Multiple S election

From the *ICPSR 3371* dataset we found:

- G - General
- M - Multi-Member District Election
- S - Special
- W - Missing Data

Let's check the unique values in this column.

```
unique(col_name_data$V3)
```

```
##  [1] "M"    "G"    "W"    "S"    "G04"  "G11"  "S06"  "S11"  NA     "ATGN"
```

We can see that we have some values that are not as expected.

Specifically, we have the value "ATGN" which is not a valid election type, it is common abbreviation for "Attorney General" in historical election data which means that this is a data entry error.

Moreover, we have the values that we would expect like "G" and "S" with number attached to them, which is not expected. Let's investigate these issues further.

```
col_name_data[V3 == "ATGN" , ] %>% unique()
```

V1 <dbl>	V2 <dbl>	V3 <chr>	V4 <chr>	V5 <chr>	V6 <chr>
1968	7	ATGN	0100	VOTE	*NA*
1968	7	ATGN	0200	VOTE	*NA*
1968	7	ATGN	TOTAL	VOTE	*NA*

3 rows

It's look like a one specific case that we can find in the original data.

```
icpsr_data[str_detect(column_name, "_7_ATGN") , .(source, column_number, column_name)] %>%
unique()
```

source <chr>	column_number <chr>	column_name <chr>
DS0201	V419	X968_7_ATGN_0100_VOTE
DS0201	V420	X968_7_ATGN_0200_VOTE
DS0201	V421	X968_7_ATGN_TOTAL_VOTE

3 rows

This is the election for the Attorney General in 1968 in Washington.

We will adjust the data to the correct columns

```
col_name_data[V3 == "ATGN" , ":=" (V3 = NA, V4 = V3, V5 = V4, V6 = V5)]
```

Let's investigate the other issue.

```
col_name_data[str_detect(V3, "G\\d") | str_detect(V3, "S\\d"), ] %>% unique()
```

V1 <dbl>	V2 <dbl>	V3 <chr>	V4 <chr>	V5 <chr>	V6 <chr>
1831	2	G04	GOV	0025	VT
1831	2	G04	GOV	1346	VT
1831	2	G04	GOV	9999	VT
1831	2	G04	GOV	TOTAL	VT
1831	2	G11	GOV	0025	VT
1831	2	G11	GOV	0100	VT
1831	2	G11	GOV	0916	VT
1831	2	G11	GOV	9999	VT
1831	2	G11	GOV	TOTAL	VT
1842	3	S06	CNG	0029	VT

1-10 of 25 rows Previous 1 2 3 Next

Looks like the issue is in two different elections, one of the "G" from 1831, and the other of the "S" from 1842.

```
icpsr_data[str_detect(column_name, "X831_2_G04|X831_2_G11") , .(source, ICPSRST, column_nam
e)] %>% unique()
```

source <chr>	ICPSRST <int>	column_name <chr>
DS0009	3	X831_2_G04_GOV_0025_VT
DS0009	3	X831_2_G04_GOV_1346_VT
DS0009	3	X831_2_G04_GOV_9999_VT
DS0009	3	X831_2_G04_GOV_TOTAL_VT
DS0009	3	X831_2_G11_GOV_0025_VT
DS0009	3	X831_2_G11_GOV_0100_VT
DS0009	3	X831_2_G11_GOV_0916_VT

source	ICPSRST	column_name
<chr>	<int>	<chr>
DS0009	3	X831_2_G11_GOV_9999_VT
DS0009	3	X831_2_G11_GOV_TOTAL_VT

9 rows

This is a special case of two different general elections in 1831 in Massachusetts. Until 1831, the election in Massachusetts was conducted annually in April. This was the final regular Massachusetts election scheduled for April before the schedule changed to November.

Let's investigate the other issue.

```
icpsr_data[str_detect(column_name, "X842_3_S06|X842_3_S11") , .(source, ICPSRST, column_nam
e)] %>% unique()
```

source	ICPSRST	column_name
<chr>	<int>	<chr>
DS0009	3	X842_3_S06_CNG_0029_VT
DS0009	3	X842_3_S06_CNG_0100_VT
DS0009	3	X842_3_S06_CNG_0526_VT
DS0009	3	X842_3_S06_CNG_9001_VT
DS0009	3	X842_3_S06_CNG_9002_VT
DS0009	3	X842_3_S06_CNG_9003_VT
DS0009	3	X842_3_S06_CNG_9999_VT
DS0009	3	X842_3_S06_CNGTOTAL_VT
DS0009	3	X842_3_S11_CNG_0029_VT
DS0009	3	X842_3_S11_CNG_0100_VT

1-10 of 16 rows Previous 1 2 Next

Once again, Massachusetts is giving us a hard time. This time it's two different special elections in 1842 at the same congressional district. Twice in this year, the incumbent congressman resigned and a special election was held to fill the vacancy.

In order to be able to differentiate between these elections we will add another column to the data that will contain the month of the election.

```
col_name_data[str_detect(V3, "G\\d") | str_detect(V3, "S\\d"), ":=" (month = as.numeric(sub
str(V3, 2, 3)),
                                                          V3 = substr(V3, 1, 1))]

unique(col_name_data$V3)
```

```
## [1] "M" "G" "W" "S" NA
```

Before moving on to the next column, did you cache the other issue that came up from our investigation on the special elections in Massachusetts?

There is a typo in some cases where the elected office and the party code does not have an underscore between them. Let's fix this issue.

```
col_name_data[V3 == "S" & month == 6 & is.na(V6) , ":=" (V4 = "CNG", V5 = "TOTAL", V6 = V
5)]
```

Forth column - Elected Office

This column should contain the name of the elected office. We have already seen that there are some issues with this column, let's investigate further.

```
unique(col_name_data$V4)
```

```
##  [1] "HAL"   "GOV"   "PRES"  "CONG"  "SEN"   "CNG"   "GV03"  "GV11"  "ATGN"
## [10] "CG04"  "CGNOV" "CGJAN" "CG11"  "70CG"  "71CG"  "CG08"  "CG12"  "HAL1"
## [19] "HAL2"
```

We can see that we have some values that are not as expected. One issue is the values "CNG" that looks like a typo in the original data. We will fix this issue to maintain consistency.

```
col_name_data[ , V4 := ifelse(V4 == "CNG", "CONG", V4)]
```

The bigger issue is the values that contain numbers or an abbreviation of some month. Those are probably special cases that we need to investigate further.

```
col_name_data[str_detect(V4, "\\d") | str_detect(V4, "JAN|NOV"), -"V5"] %>% unique()
```

V1 <dbl>	V2 <dbl>	V3 <chr>	V4 <chr>	V6 <chr>	month <dbl>
1878	2	G	GV03	VOTE	*NA*
1878	2	G	GV11	VOTE	*NA*
1941	3	S	CG04	VOTE	*NA*
1859	3	S	CGNOV	VOTE	*NA*
1859	3	S	CGJAN	VOTE	*NA*
1923	3	S	CG04	VOTE	*NA*
1923	3	S	CG11	VOTE	*NA*
1929	3	S	70CG	VOTE	*NA*

V1 <dbl>	V2 <dbl>	V3 <chr>	V4 <chr>	V6 <chr>	month <dbl>
1929	3	S	71CG	VOTE	*NA*
1870	3	S	CG08	VOTE	*NA*

1-10 of 21 rows Previous **1** 2 3 Next

We will walk through each of these cases and fix them.

Special Cases

Special Case 1 - Governor Elections in 1878

```
source_1 <- icpsr_data[str_detect(column_name, "X878_2_G_GV03|X878_2_G_GV11") , .(source)]
%>% unique() %>% as.character()

icpsr_data[(str_detect(column_name, "X878_") & source == source_1), .(ICPSRST, column_nam
e)] %>% unique()
```

ICPSRST <int>	column_name <chr>
4	X878_2_G_GV03_0100_VOTE
4	X878_2_G_GV03_0200_VOTE
4	X878_2_G_GV03_0506_VOTE
4	X878_2_G_GV03_TOTAL_VOTE
4	X878_2_G_GV11_0100_VOTE
4	X878_2_G_GV11_0200_VOTE
4	X878_2_G_GV11_0361_VOTE
4	X878_2_G_GV11_0506_VOTE
4	X878_2_G_GV11_0728_VOTE
4	X878_2_G_GV11_9999_VOTE

1-10 of 18 rows Previous **1** 2 Next

Special case in New Hampshire in 1878 where the election date was changed from March to November.

```
col_name_data[V1 == 1878 & V2 == 2 & V3 == "G" & V4 %in% c("GV03", "GV11") ,
  ":=" (month = as.numeric(str_sub(V4, 3, 4)),
        V4 = "GOV")]
rm(source_1)
```

Special Case 2 - Special Elections in 1941

```
source_2 <- icpsr_data[str_detect(column_name, "X941_3_S_CG") , .(source)] %>% unique() %>%
as.character()

icpsr_data[(str_detect(column_name, "X941_") & source == source_2), .(ICPSRST, column_nam
e)] %>% unique()
```

ICPSRST <int>	column_name <chr>
13	X941_3_S_CONG_0100_VOTE
13	X941_3_S_CONG_0200_VOTE
13	X941_3_S_CONG_0522_VOTE
13	X941_3_S_CONG_TOTAL_VOTE
13	X941_3_S_CG04_0100_VOTE
13	X941_3_S_CG04_0200_VOTE
13	X941_3_S_CG04_0370_VOTE
13	X941_3_S_CG04_0522_VOTE
13	X941_3_S_CG04_0815_VOTE
13	X941_3_S_CG04_TOTAL_VOTE

1-10 of 14 rows Previous 1 2 Next

Special case in New York in 1941 where special elections were held several times in the same year. (specifically this elections refer to the 42rd congressional district, but there were other special elections in district 17 and 14 in the same year)

```
col_name_data[V1 == 1941 & V2 == 3 & V3 == "S" & V4 %in% c("CG04") , ":=" (month = as.numer
ic(str_sub(V4, 3, 4)),
                                                                          V4 = "CONG")]
rm(source_2)
```

Special Case 3 - Special Elections in 1859

```
source_3 <- icpsr_data[str_detect(column_name, "X859_3_S_CGNOV|X859_3_S_CGJAN") , .(sourc
e)] %>% unique() %>% as.character()

icpsr_data[(str_detect(column_name, "X859_") & source == source_3), .(ICPSRST, column_nam
e)] %>% unique()
```

ICPSRST <int>	column_name <chr>
21	X859_3_S_CGNOV_0100_VOTE
21	X859_3_S_CGNOV_0200_VOTE

ICPSRST <int>	column_name <chr>
21	X859_3_S_CGNOV_9999_VOTE
21	X859_3_S_CGNOV_TOTAL_VOTE
21	X859_3_S_CGJAN_0100_VOTE
21	X859_3_S_CGJAN_9001_VOTE
21	X859_3_S_CGJAN_9999_VOTE
21	X859_3_S_CGJAN_TOTAL_VOTE

8 rows

Special case in Illinois in 1859 where special elections were held in November and January in the same year.

```
col_name_data[V1 == 1859 & V2 == 3 & V3 == "S" & V4 %in% c("CGNOV") , ":=" (month = 11, V4
= "CONG")]
col_name_data[V1 == 1859 & V2 == 3 & V3 == "S" & V4 %in% c("CGJAN") , ":=" (month = 1, V4 =
"CONG")]

rm(source_3)
```

Special Case 4 - Special Elections in 1923

```
source_4 <- icpsr_data[str_detect(column_name, "X923_3_S_CG04|X923_3_S_CG11") , .(source)]
%>% unique() %>% as.character()

icpsr_data[(str_detect(column_name, "X923_") & source == source_4), .(ICPSRST, column_nam
e)] %>% unique()
```

ICPSRST <int>	column_name <chr>
21	X923_3_S_CG04_0100_VOTE
21	X923_3_S_CG04_0200_VOTE
21	X923_3_S_CG04_0380_VOTE
21	X923_3_S_CG04_9001_VOTE
21	X923_3_S_CG04_TOTAL_VOTE
21	X923_3_S_CG11_0100_VOTE
21	X923_3_S_CG11_0200_VOTE
21	X923_3_S_CG11_0380_VOTE
21	X923_3_S_CG11_9001_VOTE

ICPSRST <int>	column_name <chr>
21	X923_3_S_CG11_TOTAL_VOTE

1-10 of 10 rows

Special case in Illinois in 1923 where special elections were held in November and January in the same year.

```
col_name_data[V1 == 1923 & V2 == 3 & V3 == "S" & V4 %in% c("CG04", "CG11") ,
  ":=" (month = as.numeric(str_sub(V4, 3, 4)),
        V4 = "CONG")]
rm(source_4)
```

Special Case 5 - Special Elections in 1929

```
source_5 <- icpsr_data[str_detect(column_name, "X929_3_S_70CG|X929_3_S_71CG") , .(source)]
%>% unique() %>% as.character()

icpsr_data[(str_detect(column_name, "X929_") & source == source_5), .(ICPSRST, column_nam
e)] %>% unique()
```

ICPSRST <int>	column_name <chr>
34	X929_3_S_70CG_0100_VOTE
34	X929_3_S_70CG_0200_VOTE
34	X929_3_S_70CG_TOTAL_VOTE
34	X929_3_S_71CG_0100_VOTE
34	X929_3_S_71CG_0200_VOTE
34	X929_3_S_71CG_TOTAL_VOTE

6 rows

Special case in Missouri in 1929 where two special elections were held in the same day to select a congressman for the (rest of the) 70th and (the full term of the) 71st congresses.

```
col_name_data[V1 == 1929 & V2 == 3 & V3 == "S" & V4 %in% c("70CG", "71CG") ,
  ":=" (cong_num = as.numeric(str_sub(V4, 1, 2)),
        V4 = "CONG")]
rm(source_5)
```

Special Case 6 - Special Elections in 1870

```
source_6 <- icpsr_data[str_detect(column_name, "X870_3_S_CG08|X870_3_S_CG12") , .(source)]
%>% unique() %>% as.character()

icpsr_data[(str_detect(column_name, "X870_") & source == source_6), .(ICPSRST, column_nam
e)] %>% unique()
```

ICPSRST <int>	column_name <chr>
47	X870_3_G_CONG_0100_VOTE
47	X870_3_G_CONG_0112_VOTE
47	X870_3_G_CONG_0200_VOTE
47	X870_3_G_CONG_0331_VOTE
47	X870_3_G_CONG_9999_VOTE
47	X870_3_G_CONG_TOTAL_VOTE
47	X870_3_S_CG08_0100_VOTE
47	X870_3_S_CG08_0112_VOTE
47	X870_3_S_CG08_0200_VOTE
47	X870_3_S_CG08_0331_VOTE

1-10 of 15 rows Previous 1 2 Next

Special case in North Carolina in 1870 where special elections were held in August and December in the same year.

```
col_name_data[V1 == 1870 & V2 == 3 & V3 == "S" & V4 %in% c("CG08", "CG12") ,
  ":=" (month = as.numeric(str_sub(V4, 3, 4)),
        V4 = "CONG")]
rm(source_6)
```

Special Case 7 - Special Elections in 1827

```
source_7 <- icpsr_data[str_detect(column_name, "X827_3_S_CG11|X827_3_S_CG12") , .(source)]
%>% unique() %>% as.character()

icpsr_data[(str_detect(column_name, "X827_") & source == source_7), .(ICPSRST, column_nam
e)] %>% unique()
```

ICPSRST <int>	column_name <chr>
51	X827_3_S_CG12_9001_VOTE
51	X827_3_S_CG12_9002_VOTE

ICPSRST <int>	column_name <chr>
51	X827_3_S_CG12_TOTAL_VOTE
51	X827_3_G_CONG_0025_VOTE
51	X827_3_G_CONG_0029_VOTE
51	X827_3_G_CONG_0041_VOTE
51	X827_3_G_CONG_0101_VOTE
51	X827_3_G_CONG_9001_VOTE
51	X827_3_G_CONG_9002_VOTE
51	X827_3_G_CONG_TOTAL_VOTE

Special case in Kentucky in 1827 where special elections were held in November and December in the same year.

```
col_name_data[V1 == 1827 & V2 == 3 & V3 == "S" & V4 %in% c("CG11", "CG12") ,
  ":=" (month = as.numeric(str_sub(V4, 3, 4)),
        V4 = "CONG")]
rm(source_7)
```

Special Case 8 - Elections to HAL1 and HAL2

```
source_8 <- icpsr_data[str_detect(column_name, "HAL1|HAL2") , .(source)] %>% unique() %>% a
s.character()

icpsr_data[(str_detect(column_name, "HAL1|HAL2") & source == source_8), .(ICPSRST, column_n
ame)] %>% unique()
```

ICPSRST <int>	column_name <chr>
66	X960_3_G_HAL1_0100_VOTE
66	X960_3_G_HAL1_0200_VOTE
66	X960_3_G_HAL1_0361_VOTE
66	X960_3_G_HAL1_0505_VOTE
66	X960_3_G_HAL1_0646_VOTE
66	X960_3_G_HAL1_1404_VOTE
66	X960_3_G_HAL1_1453_VOTE
66	X960_3_G_HAL1_9999_VOTE
66	X960_3_G_HAL1_TOTAL_VOTE

ICPSRST <int>	column_name <chr>
66	X960_3_G_HAL2_0100_VOTE

1-10 of 72 rows Previous 1 2 3 4 5 6 … 8 Next

Special case in New Mexico. From 1942 to 1968, New Mexico had two at-large representatives, with the top two vote-getters winning the seats. The first seat was designated as HAL1 and the second as HAL2.

```
col_name_data[str_detect(V4, "HAL1|HAL2"), ":=" (V4 = "HAL")]
rm(source_8)
```

Now let's see the unique values again.

```
unique(col_name_data$V4)
```

```
## [1] "HAL"  "GOV"  "PRES" "CONG" "SEN"  "ATGN"
```

Now that we have fixed the issues, let's move on to the next column.

Fifth column - Party Code

This column should contain the party code. We have a codebook for the party codes in the ICPSR data, let's check the unique values in this column.

```
unique(col_name_data$V5) %>% length()
```

```
## [1] 965
```

We have 965 unique values in this column. So print them all won't be very helpful. Let's check the unique values that are not numeric.

```
col_name_data[is.na(as.numeric(V5)) , ]$V5 %>% unique()
```

```
## [1] "TOTAL" "OTHER" "TOTA"  "0594L"
```

We have issue of typo in the original data, the value "TOTA" should be "TOTAL" and the value "0594L" should be "0594". Let's fix this issue.

```
col_name_data[ , V5 := ifelse(V5 == "TOTA", "TOTAL", V5)]
col_name_data[ , V5 := ifelse(V5 == "0594L", "0594", V5)]
```

All other values are numeric and hopefully correct and can be merged with some party name.

Sixth column - “VOTE”

```
unique(col_name_data$V6)
```

```
## [1] "VOTE" "VT"   "TE"   NA
```

As we can see the values of this column are all “VOTE” (or some abbreviation of it) which is not relevant for our analysis. We can remove this column.

```
col_name_data[ , V6 := NULL]
```

Bind the Data

```
setnames(col_name_data, new = c("YEAR", "ELECT_OFFICE_CODE",
                    "ELECT_TYPE", "ELECT_OFFICE",
                    "PARTY_CODE", "MONTH", "CONG_NUM"))

icpsr_data <- cbind(icpsr_data, col_name_data)

icpsr_data[ , c("column_number", "column_name") := NULL]

rm(col_name_data)
```

Add Index

We know now that we have some duplicated rows in the data, let’s add an index to the data to help us identify these rows. We created some of them in the New Mexico elections but there are more.

In principle, every row in our data should be unique based on the combination of the state, county, year, data source and election details. If there is more than one row with the same combination of these variables, we flag it as a duplicate.

It is reasonable to assume that there were cases of two candidates from the same party, county, and year. But other cases should be investigated further and we will do it later.

```
icpsr_data[ , INDEX := seq_len(.N), by = .(ICPSRST, ICPSRNAM, ICPSRCTY,
                                           source, YEAR, ELECT_OFFICE_CODE,
                                           ELECT_TYPE, ELECT_OFFICE, PARTY_CODE, MONTH, CON
G_NUM)]

table(icpsr_data$INDEX) %>% as.data.frame()
```

Var1 <fct>	Freq <int>
1	3698113
2	156500

Var1 <fct>	Freq <int>
3	27757
4	16492
5	6245
6	4107
7	3629
8	3261
9	2868
10	536

1-10 of 25 rows Previous 1 2 3 Next

Let's see one example of that:

```
icpsr_data[INDEX == 25, .SD[1]]
```

ICPSRST <int>	ICPSRNA... <chr>	ICPSRCTY <int>	VOT... <int>	source <chr>	YE... <dbl>	ELECT_OFFICE_CODE <dbl>	ELECT_TYPE <chr>
49	ANDERSON	10	0	DS0149	1941	5	S

1 row | 1-8 of 13 columns

On this special election to Senate in 1941 Texas, we have 25 candidates from the Democratic party.

```
icpsr_data[ICPSRST == 49 & ICPSRCTY == 10 & YEAR == 1941 &
             ELECT_OFFICE_CODE == 5 & ELECT_TYPE == "S"]
```

ICPSRST <int>	ICPSRNA... <chr>	ICPSRCTY <int>	VOT... <int>	source <chr>	YE... <dbl>	ELECT_OFFICE_CODE <dbl>	ELECT_TYPE <chr>
49	ANDERSON	10	1	DS0149	1941	5	S
49	ANDERSON	10	0	DS0149	1941	5	S
49	ANDERSON	10	0	DS0149	1941	5	S
49	ANDERSON	10	0	DS0149	1941	5	S
49	ANDERSON	10	502	DS0149	1941	5	S
49	ANDERSON	10	0	DS0149	1941	5	S
49	ANDERSON	10	1	DS0149	1941	5	S
49	ANDERSON	10	0	DS0149	1941	5	S
49	ANDERSON	10	2	DS0149	1941	5	S

ICPSRST <int>	ICPSRNA... <chr>	ICPSRCTY <int>	VOT... <int>	source <chr>	YE... <dbl>	ELECT_OFFICE_CODE <dbl>	ELECT_TYPE <chr>
49	ANDERSON	10	0	DS0149	1941	5	S

1-10 of 31 rows | 1-8 of 13 columns Previous 1 2 3 4 Next

Now we can see the full details of this election on this specific county. We have 25 candidates from the Democratic party and two from the Republican party, however, most of them have 0 votes.

A quick Google search led us to this Wikipedia page (https://en.wikipedia.org/wiki/1941_United_States_Senate_special_election_in_Texas) which states that:

> In April 1941, incumbent Senator Morris Sheppard died in office. Governor Pappy O'Daniel appointed Andrew Jackson Houston to fill the seat until a successor could be duly elected, with the election scheduled for June 28. The winner finished Sheppard's term ending in 1943.

More over the table shows that there were four candidates from the Democratic party who got most of the votes exactly as in our data.

Data Validation

In this part we will validate the data to ensure that it is consistent and accurate.

ICPSR Notes

The first thing we need to do is to read the notes from the ICPSR website about the data. The notes states that:

> In datasets 126, 127, 128 and 129 for Georgia, the county identification code for Jackson county is 1510 in the data, however, it should be 1570. Users are advised when analyzing these data by county to either use the county names in variable V2 or recode the county identification in variable V3 from 1510 to 1570 for Jackson county. (from Data Collection Notes in ICPSR website)

```
icpsr_data[source %in% c("DS0126", "DS0127", "DS0128", "DS0129") &
             ICPSRNAM == "JACKSON" & ICPSRCTY == 1510, ICPSRCTY := 1570]
```

Verify Elected Office and Codes

Let's check if the elected office and the elected office code match according to the dictionary that we saw before.

- 1 - President

- 2 - Governor
- 3 - US Rep.
- 4 - Senate 1
- 5 - Senate 2
- 6 - Senate 3
- 7 - State Offices

```
table(icpsr_data$ELECT_OFFICE, icpsr_data$ELECT_OFFICE_CODE, useNA = "always")
```

```
##
##             1       2       3       4       5       6       7    <NA>
##   ATGN      0       0       0       0       0       0    6904       0
##   CONG    112       0 1607717       0       0       0       0       0
##   GOV       0  917635      89       0       0       0       0   25697
##   HAL       0       0  264192       0       0       0       0       0
##   PRES  622000     528       0       0       0       0       0   14131
##   SEN       0     552       0  132659  178865  152747      42       0
##   <NA>      0       0       0       0       0       0       0       0
```

We have some cases of mismatches between the elected office and the elected office code. Let's investigate these issues further.

Elected Office Congrass and Code 1

Those are results from election in Maryland, 1848 (DS155). According to Wikipedia (https://en.wikipedia.org/wiki/1848_United_States_presidential_election_in_Maryland) the results match the presidential election and not the congress election. This is a typo in the original data, the elected office should be "PRES" and not "CONG".

```
icpsr_data[YEAR == "1848" & ELECT_OFFICE_CODE == 1 & ELECT_OFFICE == "CONG", ELECT_OFFICE :
= "PRES"]
```

Elected Office Senate and Code 2

Those are results from election in Indiana, 1825 (DS061). According to Wikipedia (https://en.wikipedia.org/wiki/1825_Indiana_gubernatorial_election) the results match the gubernatorial election and not the senate election. This is a typo in the original data, the elected office should be "GOV" and not "SEN".

```
icpsr_data[YEAR == "1825" & ELECT_OFFICE_CODE == 2 & ELECT_OFFICE == "SEN", ELECT_OFFICE :=
"GOV"]
```

Elected Office President and Code 2

Those are results from election in Colorado, 1894 (DS170). According to Wikipedia (https://en.wikipedia.org/wiki/1894_Colorado_gubernatorial_election) the results match the gubernatorial election and not the presidential election. This is a typo in the original data, the elected office should be "GOV" and not "PRES".

```
icpsr_data[YEAR == "1894" & ELECT_OFFICE_CODE == 2 & ELECT_OFFICE == "PRES",
         ELECT_OFFICE := "GOV"]
```

Elected Office Governor and Code 3

Those are results from election in Ohio, 1850 (DS070). According to Wikipedia (https://en.wikipedia.org/wiki/1850_United_States_House_of_Representatives_elections_in_Ohio) the results match the congress election and not the gubernatorial election. This is a typo in the original data, the elected office should be “CONG” and not “GOV”.

we could not find any source with results or data on either the gubernatorial elections nor the congress election. However, all of the cases that labeled as ELECT_OFFICE - GOV, and ELECT_OFFICE_CODE - 3 are for unknown party “9999”. So we will discard those cases.

```
icpsr_data <- icpsr_data[!(ELECT_OFFICE_CODE == 3 & ELECT_OFFICE == "GOV") | is.na(ELECT_OF
FICE_CODE), ]
```

Elected Office Senate and Code 7

Those are results from election in Arizona, 1968 (DS169). According to Wikipedia (https://en.wikipedia.org/wiki/1968_United_States_Senate_election_in_Arizona) the results match the senate election but without those votes and so those votes are not part of the senate election. This is a typo in the original data, the elected office should be “ATGN” and not “SEN”.

Also we found that the results of these cases are match to the ratio of the GOP votes for the Attorney general as noted here: *Hall, D. R. (1969). The 1968 Election in Arizona. The Western Political Quarterly, 22(3), 463. https://doi.org/10.2307/446337 (https://doi.org/10.2307/446337)*

```
icpsr_data[ELECT_OFFICE_CODE == 7 & ELECT_OFFICE == "SEN", ELECT_OFFICE := "ATGN"]
```

Now let’s check the elected office and the elected office code again.

```
table(icpsr_data$ELECT_OFFICE, icpsr_data$ELECT_OFFICE_CODE, useNA = "always")
```

```
##
##             1       2       3       4       5       6       7    <NA>
##   ATGN      0       0       0       0       0       0    6946       0
##   CONG      0       0 1607717       0       0       0       0       0
##   GOV       0  918715       0       0       0       0       0   25697
##   HAL       0       0  264192       0       0       0       0       0
##   PRES  622112       0       0       0       0       0       0   14131
##   SEN       0       0       0  132659  178865  152747       0       0
##   <NA>      0       0       0       0       0       0       0       0
```

Delete No Data Elections

If we have cases of elections with missing votes in all entries, we can safely remove them. To check for such cases, we first need to create an identifier for each election.

```
# create a column with unique number for each group with in the specified columns

icpsr_data[ , ELEC_ID := .GRP, by = .(ICPSRST, ICPSRNAM, ICPSRCTY,
                                      source, YEAR,
                                      ELECT_OFFICE_CODE, ELECT_TYPE,
                                      ELECT_OFFICE, MONTH, CONG_NUM)]

summary(icpsr_data$ELEC_ID)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1  125479  271565  274393  421486  564752
```

```
#  Count the number of missing values in the votes and compare it to the number of votes
icpsr_data[ , ":=" (no_data_elections = sum(is.na(VOTES) | VOTES == 0) == .N), by = ELEC_I
D]
```

If all the votes are missing in an election, we will remove this election from the data.

```
icpsr_data <- icpsr_data[no_data_elections == F, ]
icpsr_data[ , no_data_elections := NULL]
```

Add States' and Parties' Names

Add States' Names

Now we will add the state's name to the data, for that we will use the ICPSR state codebook. But first, let's check if we have any missing state codes.

```
icpsr_data[is.na(ICPSRST) , ] %>% nrow()
```

```
## [1] 15
```

We have some missing state codes, let's fix this issue.

```
icpsr_data[is.na(ICPSRST) , source] %>% unique()
```

```
## [1] "DS0041" "DS0042"
```

Missing state code in two of the data sets of New York. Let's fix this issue.

```
icpsr_data[is.na(ICPSRST), ICPSRST := 13]
```

Now we will add the state's name to the data.

```
state_codebook <- fread(paste0(data_path, "/ICPSR/ICPSR_codebook_state_country.csv"))
state_codebook <- state_codebook[-1 , .SD[1], by = State, .SDcols = c("STATEICP")]
setnames(state_codebook, new = c("STATE", "ICPSRST"))

icpsr_data <- merge(icpsr_data, state_codebook, by = "ICPSRST", all.x = TRUE)

icpsr_data[ , .N, by = .(STATE, ICPSRST)]
```

STATE <chr>	ICPSRST <int>	N <int>
Connecticut	1	10212
Maine	2	22887
Massachusetts	3	25054
New Hampshire	4	10332
Rhode Island	5	8456
Vermont	6	20764
Delaware	11	3651
New Jersey	12	28609
New York	13	99352
Pennsylvania	14	128726

1-10 of 50 rows Previous 1 2 3 4 5 Next

```
rm(state_codebook)
```

Add Parties' Names

```
party_codebook <- fread(paste0(data_path, "/party_codebook.csv"))
setnames(party_codebook, new = c("PARTY_CODE", "PARTY_NAME"))

# Check for duplicates in the party codebook
nrow(party_codebook) == party_codebook$PARTY_CODE %>% unique() %>% length()
```

```
## [1] TRUE
```

```
party_codebook %>% str()
```

```
## Classes 'data.table' and 'data.frame':   1679 obs. of  2 variables:
##  $ PARTY_CODE: int  1 8 9 10 11 12 13 14 20 21 ...
##  $ PARTY_NAME: chr  "FEDERALIST" "ANTI-DEMOCRAT" "JEFFERSON REPUBLICAN" "ANTI-FEDERALIS
T" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

So we have 1679 unique party codes in the codebook and the party codes are all numeric. We know by now that the party codes in our data are not numeric because of the “TOTAL” values and the “OTHER” party. Let’s fix this issue before merging the party names.

A quick look at the values in the party names column discovered that there is no “OTHER” party in the codebook but there is an “UNIDENTIFIED” party. For our purposes, we will consider the “OTHER” party in the data as “UNIDENTIFIED”.

Handle “OTHER” Party

```
icpsr_data[PARTY_CODE == "OTHER" , PARTY_CODE := "9001"]
```

Handle “TOTAL” Party

Now we want to convert the total votes into a new column.

It should be that in each election there is only one row with the party code “TOTAL”. Let’s check if this is the case.

```
total_issue <- icpsr_data[, lapply(.SD, function(x){sum(PARTY_CODE == "TOTAL")}), by = ELEC
_ID, .SDcols = "PARTY_CODE"] %>% .[PARTY_CODE != 1, ]

total_issue %>% select(PARTY_CODE) %>% table() %>% as.data.table()
```

PARTY_CODE <chr>	N <int>
0	591
2	441

2 rows

We have elections where there is no “TOTAL” votes at all and cases where there is more than one row with the party code “TOTAL”.

No “TOTAL” Votes

```
no_total <- total_issue[PARTY_CODE == 0 , ELEC_ID]

icpsr_data[ELEC_ID %in%  no_total, source] %>% unique()
```

```
## [1] "DS0039" "DS0082" "DS0105" "DS0134" "DS0139" "DS0163" "DS0170" "DS0176"
## [9] "DS0190"
```

```
rm(no_total)
```

It seems that the issue is widespread in multiple datasets. From the ICPSR website we learn that:

> Counties may have been assigned a congressional district number of zero if they were not in existence at the time of the election, did not report vote totals, or did not participate in that election.

Which indicates that it is not uncommon to have missing data in the “TOTAL” votes.

Two “TOTAL” Votes

```
two_total <- total_issue[PARTY_CODE == 2 , ELEC_ID]

icpsr_data[ELEC_ID %in%  two_total, source] %>% unique()
```

```
##  [1] "DS0036" "DS0043" "DS0045" "DS0047" "DS0051" "DS0083" "DS0138" "DS0170"
##  [9] "DS0184" "DS0190"
```

The cases with two “TOTAL” votes are also widespread in multiple datasets.

Let’s compare the different values of “TOTAL” votes in each of those cases.

```
# Create a new data frame with the specific cases
two_total <- icpsr_data[ELEC_ID %in% two_total, ]

# create two more columns for the duplicated total votes
two_total[ , ":=" (TOTAL1 = VOTES, TOTAL2 = VOTES)]

two_total[!(PARTY_CODE == "TOTAL" & INDEX == 1) , ":=" (TOTAL1 = 0)]
two_total[!(PARTY_CODE == "TOTAL" & INDEX == 2) , ":=" (TOTAL2 = 0)]

two_total[ , ":=" (TOTAL1 = sum(TOTAL1), TOTAL2 = sum(TOTAL2)), by = ELEC_ID]

# Check how many cases have missing values in both "TOTAL" votes
two_total[is.na(TOTAL1) & is.na(TOTAL2) , ] %>% nrow()
```

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

There are two possible reasons to have one election with two records of “TOTAL” votes:

1. The first reason is that this election is in fact two different elections that we cant differentiate between them with the data we have. In this case the best course of action is to treat them as one election and to sum up the total votes.
2. The second reason is that this is a data entry error and the total votes were entered twice. In this case we should treat them as one election and to keep those total votes separated.

Since there are many cases like this and we cant investigate each one of them, we will use a different method to differentiate between the two cases. We will create a new column with the sum of the votes (without the total votes) and another column with the sum of the two “TOTAL” votes entries. If the sum of the votes is approximately equal to the sum of the two “TOTAL” votes entries, it is indication that this election is in fact two different elections. In that case we will sum the votes of the “TOTAL” entries (and also in case that one of the “TOTAL” votes is missing). Otherwise, it is indication that this is a data entry error.

```
two_total <- two_total[PARTY_CODE != "TOTAL" , ]
# create the new columns
two_total[ , ":=" (TOTAL3 = sum(VOTES, na.rm = T),
                   TOTAL4 = case_when(
                     is.na(TOTAL1) ~ TOTAL2,
                     is.na(TOTAL2) ~ TOTAL1,
                     .default = TOTAL1 + TOTAL2
                   )), by = ELEC_ID]

# create a logical column to indicate if this is two different elections
two_total[ , sum_total := case_when(
  is.na(TOTAL1) | TOTAL1 == 0 ~ TRUE,
  is.na(TOTAL2) | TOTAL2 == 0 ~ TRUE,
  TOTAL3 > TOTAL4*0.9 ~ TRUE,
  .default = FALSE
)]
```

Now we can sum the votes of the “TOTAL” entries or keep them seperated.

```
elec_to_sum <- two_total[sum_total == TRUE, ELEC_ID]

icpsr_data[ELEC_ID %in% elec_to_sum & PARTY_CODE == "TOTAL", VOTES := sum(VOTES, na.rm =
T), by = ELEC_ID]

icpsr_data <- icpsr_data[!(ELEC_ID %in% elec_to_sum & PARTY_CODE == "TOTAL" & INDEX == 2) ,
]
```

Now we will create a new column with the total votes in each election and another column with the second entry of the “TOTAL” votes in cases of data entry error.

```
icpsr_data[ , ":=" (TOTAL = VOTES, TOTAL2 = VOTES)]

icpsr_data[!(PARTY_CODE == "TOTAL" & INDEX == 1) , ":=" (TOTAL = 0)]
icpsr_data[!(PARTY_CODE == "TOTAL" & INDEX == 2) , ":=" (TOTAL2 = 0)]

icpsr_data[ , ":=" (TOTAL = sum(TOTAL), TOTAL2 = sum(TOTAL2)), by = ELEC_ID]

# convert the second column to total votes to NA in elections without two "TOTAL" votes
elec_no_sum <- two_total[sum_total == FALSE, ELEC_ID]

icpsr_data[!(ELEC_ID %in% elec_no_sum) , ":=" (TOTAL2 = NA)]

# delete the "TOTAL" votes entries
icpsr_data <- icpsr_data[PARTY_CODE != "TOTAL" , ]

# create a new column with the sum of the votes for comparison
icpsr_data[ , ":=" (TOTAL_VOTES = sum(VOTES, na.rm = T)), by = ELEC_ID]

# convert the party code to numeric to merge with the party codebook
icpsr_data[ , PARTY_CODE := as.numeric(PARTY_CODE)]

rm(two_total, elec_to_sum, elec_no_sum, total_issue)
```

Now we can merge the party names.

```
icpsr_data <- merge(icpsr_data, party_codebook, by = "PARTY_CODE", all.x = TRUE)

icpsr_data[is.na(PARTY_NAME) , ] %>% nrow()
```

```
## [1] 176788
```

```
rm(party_codebook)
```

Unfortunately, we have some missing party names.

Organize the Data

Simplify the Data

We will create a simplified version of some of the columns in the data but keep a version of the original columns as well.

```
icpsr_data[ , ":=" (ELECT_OFFICE_CODE_ORIG = ELECT_OFFICE_CODE,
                    ELECT_OFFICE_ORIG = ELECT_OFFICE,
                    ELECT_TYPE_ORIG = ELECT_TYPE)]

# Fill the missing values in the elected office code and delete redundant values
icpsr_data[ , ":=" (ELECT_OFFICE_CODE = case_when(ELECT_OFFICE_CODE %in% c(4:6) ~ 4,
                                                  ELECT_OFFICE_CODE == 7 ~ 5,
                                                  is.na(ELECT_OFFICE_CODE) &
                                                    ELECT_OFFICE == "PRES" ~ 1,
                                                  is.na(ELECT_OFFICE_CODE) &
                                                    ELECT_OFFICE == "GOV" ~ 2,
                                                  .default = ELECT_OFFICE_CODE))]
# Harmonize the elected office names
icpsr_data[ , ":=" (ELECT_OFFICE = ifelse(ELECT_OFFICE == "HAL", "CONG", ELECT_OFFICE))]
# Harmonize the elected office types
icpsr_data[ , ":=" (ELECT_TYPE = case_when(ELECT_TYPE == "M" ~ "G",
                                           ELECT_TYPE == "W" ~ "S",
                                           .default = ELECT_TYPE))]
```

Reorder Columns

```
icpsr_data <- icpsr_data[, c("ICPSRST", "STATE", "ICPSRCTY", "ICPSRNAM", "YEAR",
                             "ELECT_OFFICE_CODE", "ELECT_OFFICE", "ELECT_TYPE",
                             "PARTY_CODE", "PARTY_NAME", "VOTES", "TOTAL", "TOTAL2", "TOTAL
_VOTES",
                             "source", "MONTH", "CONG_NUM", "INDEX", "ELEC_ID",
                             "ELECT_OFFICE_CODE_ORIG", "ELECT_OFFICE_ORIG", "ELECT_TYPE_ORI
G")]

icpsr_data <- icpsr_data[order(YEAR, ICPSRST, ICPSRCTY, PARTY_CODE) , ]
```

Add Value Labels to Categorial Columns

```
categories_levels <- list(
  ELECT_OFFICE_CODE = c(President = 1, Governor = 2, Congress = 3,
                        Senate = 4, Attorney_General = 5),
  ELECT_OFFICE = c(President = "PRES", Governor = "GOV",
                   Congress = "CONG", Senate = "SEN", Attorney_General = "ATGN"),
  ELECT_TYPE = c(General = "G", Special = "S"),
  ELECT_OFFICE_CODE_ORIG = c(President = 1, Governor = 2, Congress = 3,
                             Senate = 4, Senate = 5, Senate = 6, Attorney_General = 7),
  ELECT_OFFICE_ORIG = c(President = "PRES", Governor = "GOV",
                        Congress = "CONG", Senate = "SEN",
                        House_of_assembly = "HAL", Attorney_General = "ATGN"),
  ELECT_TYPE_ORIG = c(General = "G", Special = "S",
                      Multiple_G_elections = "M",
                      Multiple_S_elections_or_missing_data = "W")
)

for (col in names(categories_levels)) {
  icpsr_data[[col]] <- labelled(icpsr_data[[col]], categories_levels[[col]])
}

rm(categories_levels)
```

Add Labels

```
# Add short explanation to the columns as labels:
labels <- c("ICPSR State Code", "State Name", "ICPSR County Code", "County Name", "Year of
Election",
            "Elected Office Code", "Elected Office", "Election Type",
            "Party Code", "Party Name", "Votes", "The Original TOTAL entry", "Second TOTAL
entry",
            "Sum of VOTES Per Election", "Original ICPSR Data Set", "Month of Election", "C
ongress Number",
            "Index", "Election ID",
            "Original Elected Office Code", "Original Elected Office", "Original Election T
ype")

var_label(icpsr_data) <- labels

rm(labels)
```

Explore the Data

```
icpsr_data
```

ICPSRST <int>	STATE <chr>	ICPSRCTY <int>	ICPSRNAM <chr>	YEAR <dbl>	ELECT_OFFICE_CODE <dbl+lbl>
11	Delaware	10	KENT	1823	2
11	Delaware	10	KENT	1823	2
11	Delaware	10	KENT	1823	2
11	Delaware	10	KENT	1823	2
11	Delaware	10	KENT	1823	2
11	Delaware	10	KENT	1823	2
11	Delaware	30	NEW CASTLE	1823	2
11	Delaware	30	NEW CASTLE	1823	2
11	Delaware	30	NEW CASTLE	1823	2
11	Delaware	30	NEW CASTLE	1823	2

1-10 of 10,000 rows | 1-6 of 22 columns Previous **1** 2 3 4 5 6 … 1000 Next

Export Data

Finally, we save the cleaned and organized data in RDS format.

```
saveRDS(icpsr_data, paste0(data_path, "/elections_returns/election_returns.rds"))
```

We will also save it in CSV format with a README file.

```
# Write to a CSV file
fwrite(icpsr_data, paste0(data_path, "/elections_returns/csv_version/election_returns.cs
v"))

# create a README file with the variable information
var_info <- look_for(icpsr_data)

var_info <- var_info %>% mutate(pos = NULL, levels = NULL)
var_info$value_labels <- lapply(var_info$value_labels, function(x) x = ifelse(is.null(x),
                                                                               NA,
                                                                               paste(names(x), x, collap

se = ", "))) %>%
  unlist()

fwrite(var_info, paste0(data_path, "/elections_returns/csv_version/README.csv"))

rm(var_info)
```