

Introduction

Data Import and Presentation

Define Functions for Data Processing

Process and Read All Files

Final Step: Merging All Processed Data

# Dealing With Historical US Election Data - Part 1

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

Roe Diler

2025-03-23

## Introduction

This notebook processes election returns data from the ICPSR dataset. It reads ASCII files, cleans the data, and generates CSV files for further analysis.

It was part of an economic history project that I worked on with a professor at the Hebrew University of Jerusalem. The goal was to analyze historical election data to understand voting patterns and political trends in the United States.

## ICPSR and Historical Election Data

The **Inter-university Consortium for Political and Social Research (ICPSR)** is a major archive of social science data, providing access to a wide range of datasets. One of its significant contributions is the **United States Historical Election Returns, 1824-1968 (ICPSR 1)**, a collection of county-level election returns spanning nearly 150 years.

This dataset includes over **90% of all elections** for the offices of **president, governor, U.S. senator, and U.S. representative** between 1824 and 1968. It covers both **regular and special elections**, documenting results for **all parties, candidates, and independent contenders**—including more than 1,000 unique political party names. The dataset provides a valuable resource for researchers studying **historical voting patterns, political party evolution, and electoral trends** across the United States.

These data sets were stored in **ASCII (American Standard Code for Information Interchange) format** which is a simple text-based format with accompanying **SAS setup files** that describe the structure of the data. The ASCII files contain raw data without metadata, while the SAS setup files provide information on **column positions, variable names, formats, and missing values**.

In R, these datasets can be imported using packages like **SAScii** and **asciiSetupReader**. We will examine both of these packages to determine the best approach for reading the ICPSR election data.

## Citation

Inter-university Consortium for Political and Social Research. *United States Historical Election Returns, 1824-1968*. Inter-university Consortium for Political and Social Research [distributor], 1999-04-26.

<https://doi.org/10.3886/ICPSR00001.v3> (<https://doi.org/10.3886/ICPSR00001.v3>).

## Required Packages

Load necessary libraries using `pacman` for package management.

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

library(pacman)
p_load(data.table, tidyverse, asciiSetupReader, SAScii, nanian, Hmisc, rlist)
```

## Define Paths

Set up directory paths dynamically based on the script location.

```
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")
```

## Data Import and Presentation

### Read the First Dataset

Let's start by reading the first dataset (DS0001) from the ICPSR election data. We will use the `read.SAScii` function from the `SAScii` package to import the data and review the first few rows.

```
txt_path <- file.path(icpsr_path, "/DS0001/00001-0001-Data.txt")
sas_path <- file.path(icpsr_path, "/DS0001/00001-0001-Setup.sas")

elec_data001 <- read.SAScii(txt_path, sas_path) %>% as.data.table()
```

```
elec_data001 %>% head()
```

V1 <dbl>	V2 <chr>	V3 <dbl>	V4 <dbl>	V5 <dbl>	V6 <dbl>	V7 <dbl>	V8 <dbl>	V9 <dbl>	V10 <dbl>
1	FAIRFIELD	10	98	98	98	98	98	98	98
1	HARTFORD	30	98	98	98	98	98	98	98

V1 <dbl>	V2 <chr>	V3 <dbl>	V4 <dbl>	V5 <dbl>	V6 <dbl>	V7 <dbl>	V8 <dbl>	V9 <dbl>	V10 <dbl>
1	LITCHFIELD	50	98	98	98	98	98	98	98
1	MIDDLESEX	70	98	98	98	98	98	98	98
1	NEW HAVEN	90	98	98	98	98	98	98	98
1	NEW LONDON	110	98	98	98	98	98	98	98

6 rows | 1-10 of 509 columns

We can see that the data is in a wide format with multiple columns representing different variables. The column names are not descriptive, and the data is not in a format suitable for analysis.

Let's try the `asciiSetupReader` package to read the data and extract variable names and missing values.

```
elec_data001 <- read_ascii_setup(txt_path, sas_path) %>% as.data.table()
elec_data001 %>% head()
```

ICPR_STATE_CODE <dbl>	COUNTY_NAME <chr>	IDENTIFICATION_NUMBER <dbl>	CONG_DIST_NUMBER_1825 <dbl>
1	FAIRFIELD	10	98
1	HARTFORD	30	98
1	LITCHFIELD	50	98
1	MIDDLESEX	70	98
1	NEW HAVEN	90	98
1	NEW LONDON	110	98

6 rows | 1-4 of 499 columns

The column names are correct, **but 10 columns are missing.**

Since neither package provides a complete solution, we will need to combine the functionality of both to read the data correctly.

We will create some functions that will help us process the data correctly and set up a pipeline to generate a well organized data set.

# Define Functions for Data Processing

## Outline of Data Processing Steps

To correctly process the ICPSR election data, we need a series of functions to:

1. **Define Paths:** Dynamically locate the ASCII data and setup files for each dataset.
2. **Read the Raw Data:** Load the dataset using the `read.SAScii` function and extract variable names and missing values dictionary using `parse_setup` function.
3. **Adjust Missing Data:** Identify and replace missing values in key columns.

4. **Reshape Data:** Transform the dataset into a long format for easier analysis.
5. **Merge Data:** Attach variable names and missing value dictionary to the main dataset.
6. **Automate Processing:** Wrap all the above steps into a single function to process multiple datasets efficiently.

## Define Functions

### Function: Define Paths

Generates file paths dynamically based on the dataset number.

```
define_paths <- function(file_num) {  
  list(  
    ascii_path = file.path(icpsr_path, paste0("DS0", file_num, "/00001-0", file_num, "-Data.  
txt")),  
    sas_path = file.path(icpsr_path, paste0("DS0", file_num, "/00001-0", file_num, "-Setup.s  
as"))  
  )  
}
```

### Function: Read Raw Data

Reads ICPSR data using `read.SAScii`, extracts variable names, and missing values dictionary using `parse_setup` function.

```
raw_read <- function(paths) {  
  icpsr_data <- read.SAScii(paths$ascii_path, paths$sas_path) %>% as.data.table()  
  setup_file <- parse_setup(paths$sas_path)  
  var_names <- setup_file[[1]] %>% as.data.table() %>% select(column_number, column_name)  
  missing <- setup_file[[3]] %>% as.data.table()  
  
  list(data = icpsr_data, var_names = var_names, missing = missing)  
}
```

### Function: Handle Missing IDs

Replaces missing values with `NA` in the `v1` and `v3` columns, representing the state and county identifiers.

```
change_missing_ids <- function(full_list) {
  icpsr_data <- full_list$data
  missing <- full_list$missing

  for (col in c("V1", "V3")) {
    if (col %in% missing$variable) {
      missing_values <- as.numeric(missing[variable == col, values])
      icpsr_data[get(col) %in% missing_values, (col) := NA]
    }
  }
  full_list$data <- icpsr_data
  return(full_list)
}
```

## Function: Reshape Data

Transforms data from wide to long format and adds a source identifier.

```
melt_icpsr <- function(full_list, file_num) {
  full_list$data <- melt(full_list$data, id.vars = c("V1", "V2", "V3"))
  full_list$data[, source := paste0("DS0", file_num)]
  return(full_list)
}
```

## Function: Merge Variables

Merges variable names and missing values into the dataset.

```
merge_icpsr <- function(full_list) {
  full_list$data <- merge(full_list$data, full_list$var_names,
                        by.x = "variable", by.y = "column_number", all.x = TRUE)
  full_list$data <- merge(full_list$data, full_list$missing,
                        by = "variable", all.x = TRUE)
  return(full_list$data)
}
```

## Function: Process ICPSR Data

Combines all previous steps into a single pipeline.

```
icpsr_process <- function(file_num) {
  file_num %>%
    define_paths() %>%
    raw_read() %>%
    change_missing_ids() %>%
    melt_icpsr(file_num = file_num) %>%
    merge_icpsr()
}
```

# Process and Read All Files

## Process ICPSR Data

We will now process all ICPSR datasets using the defined functions and store the cleaned data in a list. Exclude problematic datasets that require manual processing.

```
# Get ICPSR 0001 files list
filenames <- list.files(path = icpsr_path, full.names = FALSE)
filenames <- str_subset(filenames, "DS0")
filenames <- gsub("DS0", "", filenames)
# delete the problematic data sets that we will process manually and party codes file
filenames <- filenames[!filenames %in% c("001", "012", "091", "144", "194", "202", "204")]

# Read and clean ICPSR 0001 files
data_list <- lapply(filenames, icpsr_process)
```

## Manual Error Corrections for Multiple Datasets

### Data Set 1: Correcting Missing Variable Names

- The function `parse_setup` does not read all the vars names, we will add them manually.

```
# Define the file number for the dataset
file_num <- "001"

# Load the dataset and process missing IDs
full_list <- file_num %>% define_paths() %>%
  raw_read() %>%
  change_missing_ids()
```

```
# Extract the dataset and variable names from the processed list
icpsr_data <- full_list$data
var_names <- full_list$var_names

# Display the first 10 variable names to identify missing values
head(var_names, 10)
```

column_number <chr>	column_name <chr>
V1	ICPR_STATE_CODE
V2	COUNTY_NAME
V3	IDENTIFICATION_NUMBER
V4	CONG_DIST_NUMBER_1825
V5	CONG_DIST_NUMBER_1829

column_number <chr>	column_name <chr>
V7	CONG_DIST_NUMBER_1833
V9	CONG_DIST_NUMBER_1835
V11	CONG_DIST_NUMBER_1837
V13	CONG_DIST_NUMBER_1841
V15	CONG_DIST_NUMBER_1845
1-10 of 10 rows	

The function `parse_setup` does not correctly read all variable names. Specifically, it misses congressional district number variables between V6 and V24. We manually add these missing variable names.

```
# Define the missing variable names along with their respective identifiers
var_names_add <- data.table(
  column_number = c("V6", "V8", "V10",
                    "V12", "V14", "V16",
                    "V18", "V20", "V22", "V24"),
  column_name = var_names$column_name[5:14] # Extract the missing names from the dataset
)

# Manually assign the correct congressional district numbers to the corresponding variables
var_names[5:14, column_name :=
  c("CONG_DIST_NUMBER_1827", "CONG_DIST_NUMBER_1831",
    "CONG_DIST_NUMBER_1834", "CONG_DIST_NUMBER_1836",
    "CONG_DIST_NUMBER_1839", "CONG_DIST_NUMBER_1843",
    "CONG_DIST_NUMBER_1847", "CONG_DIST_NUMBER_1851",
    "CONG_DIST_NUMBER_1855", "CONG_DIST_NUMBER_1859")]

# Append the newly defined variables to the variable names dataset
var_names <- rbind(var_names, var_names_add)

# Ensure that variables are ordered numerically by their column number
var_names <- var_names[order(as.numeric(gsub("V", "", column_number))), ]

# Remove the temporary dataset to free memory
rm(var_names_add)

# Display the first 10 variable names to verify the changes
head(var_names, 10)
```

column_number <chr>	column_name <chr>
V1	ICPR_STATE_CODE
V2	COUNTY_NAME
V3	IDENTIFICATION_NUMBER

column_number <chr>	column_name <chr>
V4	CONG_DIST_NUMBER_1825
V5	CONG_DIST_NUMBER_1827
V6	CONG_DIST_NUMBER_1829
V7	CONG_DIST_NUMBER_1831
V8	CONG_DIST_NUMBER_1833
V9	CONG_DIST_NUMBER_1834
V10	CONG_DIST_NUMBER_1835
1-10 of 10 rows	

Next, we define missing value codes for the newly added variables and append them to the existing missing value definitions.

```
# Handle missing value definitions for the newly added variables
missing <- full_list$missing

# Define missing value codes for the new variables
missing_add <- data.table(
  variable = c("V6", "V8", "V10",
               "V12", "V14", "V16",
               "V18", "V20", "V22", "V24"),
  values = c(rep("0000099", 9), "9999999")
)

# Append missing value definitions to the existing missing dataset
missing <- rbind(missing, missing_add)

# Remove temporary dataset to free memory
rm(missing_add)
```

Finally, we update the full list with corrected variable names and missing value definitions, complete the data processing pipeline, and store the corrected dataset in the list for further analysis.

```
# Update the full list with corrected variable names and missing value definitions
full_list <- list(data = icpsr_data, var_names = var_names, missing = missing)

# Complete the data processing pipeline
d1 <- full_list %>% melt_icpsr(file_num = file_num) %>% merge_icpsr()

# Store the cleaned and processed dataset in a list for further analysis
data_list[[1]] <- d1

# Clean up the workspace by removing unnecessary objects
rm(d1, full_list, icpsr_data, var_names, missing)
```



## Data Set 12: Correcting Variable Case Sensitivity Issue

- The variable `v444` in the sas setup file was written as `v444` , we will fix it.

```
# Define the file number
file_num <- "012"

# Read and preprocess the dataset
full_list <- file_num %>% define_paths %>% raw_read() %>%
  change_missing_ids()

# Fix the variable name case issue (v444 -> V444)
full_list$var_names <- full_list$var_names[column_number == "v444", column_number := "V444"]

# Define missing values for the corrected variable
missing_add <- data.table(variable = "V444", values = "9999999")
full_list$missing <- rbind(full_list$missing, missing_add)
rm(missing_add)

# Process the dataset

d12 <- full_list %>%
  melt_icpsr(., file_num = file_num) %>%
  merge_icpsr()

# Store the processed dataset
data_list[[12]] <- d12

# Clean up workspace
rm(d12, full_list, file_num)
```

## Data Set 91: Handling Null Values to Prevent Data Corruption

- There are null values in the data set which causes the function to read all the data afterward as null values, we will use alternative data set without null values.

This is the warning message that we received when trying to read the data set with null values:

Warning: line 46 appears to contain an embedded nul:

```

# Define the file number
file_num <- "091"

# Define file paths to use an alternative dataset without null values
ascii_path = paste0(data_path, "/icpsr_no_nul/DS0", file_num, "_no_nul.txt")
sas_path = paste0(icpsr_path, "/DS0", file_num, "/00001-0", file_num, "-Setup.sas")

paths <- list(ascii_path = ascii_path, sas_path = sas_path)

# Read and preprocess the dataset using alternative files

d91 <- paths %>% raw_read() %>%
  change_missing_ids() %>%
  melt_icpsr(., file_num = file_num) %>%
  merge_icpsr()

# Store the processed dataset
data_list[[91]] <- d91

# Clean up workspace
rm(d91, ascii_path, sas_path, paths, file_num)

```

## Data Set 144: Removing Duplicate Variable (V4)

- The variable `v4` duplicates the variable `v2` which is the name of the county, we will remove it.

```

# Define the file number
file_num <- "144"

# Read and preprocess the dataset
full_list <- file_num %>% define_paths %>% raw_read() %>%
  change_missing_ids()

# Remove duplicate variable V4 (which duplicates county name from V2)
full_list$data <- full_list$data %>% select(-V4)

# Process the dataset

d144 <- full_list %>%
  melt_icpsr(., file_num = file_num) %>%
  merge_icpsr()

# Store the processed dataset
data_list[[144]] <- d144

# Clean up workspace
rm(d144, full_list, file_num)

```

## Data Set 194: Fixing Misabeled Variables in SAS Setup File

- The SAS setup file has typo errors in the LABEL section in the V3 version from 1999 that we used.

```
# Define the file number
file_num <- "194"

# Read and preprocess the dataset
full_list <- file_num %>% define_paths() %>%
  raw_read() %>%
  change_missing_ids()
```

```
# Extract dataset and variable names
icpsr_data <- full_list$data
var_names <- full_list$var_names

# Identify mislabeled variables
var_names[c(53:56, 123:127, 153:155) , ]
```

column_number <chr>	column_name <chr>
V53	X946_4_S_SEN_0200_VOTE
V54	X946_4_S_SEN_0361_VOTE
V56	X946_4_S_SEN_9002_VOTE
V57	X946_4_S_SEN_9003_VOTE
V124	X952_3_G_CONG_0328_VOTE
V125	X952_3_G_CONG_0618_VOTE
V127	X952_3_G_CONG_9001_VOTE
V128	X952_3_G_CONG_9999_VOTE
V129	X952_3_G_CONG_TOTAL_VOTE
V155	X956_1_G_PRES_0646_VOTE
1-10 of 12 rows	
Previous 1 2 Next	

As we can see, the variable names V55 and V126 are missing. We will manually fix these errors.

```

var_names_add <- data.table(column_number = c("V55", "V126"),
                           column_name = c("X946_4_S_SEN_9001_VOTE", "X952_3_G_CONG_0749_VO
TE"))
var_names[column_number == "V156", column_name := "X956_1_G_PRES_0913_VOTE"]
var_names <- rbind(var_names, var_names_add)
rm(var_names_add)

# Define missing values for corrected variables
missing <- full_list$missing
missing_add <- data.table(variable = c("V55", "V126"),
                          values = c(rep("9999999", 2)))
missing <- rbind(missing, missing_add)
rm(missing_add)

# Update full list with corrections
full_list <- list(data = icpsr_data, var_names = var_names, missing = missing)

d194 <- full_list %>% melt_icpsr(file_num = file_num) %>% merge_icpsr()

# Store the processed dataset
data_list[[194]] <- d194

# Clean up workspace
rm(d194, full_list, icpsr_data, var_names, missing, file_num)

```

## Data Set 202: Handling Incorrect Label Formatting in SAS Setup File

- The SAS setup file has typo errors in the LABEL section in the V3 version from 1999 that we used. The begging of the LABEL section was suppose to be \*/ , but it was /\* . In this case the other function read\_ascii\_setup read the data well so we used it.

```

# Define the file number
file_num <- "202"

# Define file paths
paths <- file_num %>% define_paths()

# Read dataset using an alternative parsing function due to SAS label formatting issue
icpsr_data <- read_ascii_setup(paths$ascii_path, paths$sas_path) %>% as.data.table()
setup_file <- parse_setup(paths$sas_path)

# Extract variable names and missing value definitions
var_names <- setup_file[[1]] %>% as.data.table() %>% select(column_number, column_name)
missing <- setup_file[[3]] %>% as.data.table()

# Rename key variables for consistency
setnames(icpsr_data,
         old = c("ICPR_STATE_CODE", "COUNTY_NAME", "IDENTIFICATION_NUMBER"),
         new = c("V1", "V2", "V3"))

# Combine dataset components
full_list <- list(data = icpsr_data, var_names = var_names, missing = missing)

# Process and reshape data
full_list <- full_list %>% change_missing_ids() %>% melt_icpsr(file_num = file_num)

# Merge metadata
full_list$data <- merge(full_list$data, full_list$var_names,
                       by.x = "variable", by.y = "column_name", all.x = TRUE)
full_list$data <- merge(full_list$data, full_list$missing,
                       by.x = "column_number", by.y = "variable", all.x = TRUE)

# Rename columns for clarity
setnames(full_list$data, old = c("column_number", "variable"), new = c("variable", "column_name"))

# Ensure consistency with other datasets
full_list$data <- full_list$data %>% select(names(data_list[[1]]))

# Store the processed dataset
data_list[[202]] <- full_list$data

# Clean up workspace
rm(icpsr_data, var_names, missing, paths, setup_file, full_list, file_num)

```

## Final Step: Merging All Processed Data

We will now merge all processed datasets into a single dataset and save it as a CSV file for further analysis.

```
icpsr_data <- rbindlist(data_list)

# Save the merged dataset to a CSV file
fwrite(icpsr_data, paste0(data_path, "/icpsr_long_raw.csv"))
```

A quick summary of the final dataset:

```
skimr::skim(icpsr_data)
```



#### Data summary

Name	icpsr_data
Number of rows	4135212
Number of columns	8
Key	NULL
<hr/>	
Column type frequency:	
character	5
numeric	3
<hr/>	
Group variables	None

#### Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
variable	0	1	2	4	0	517	0
V2	0	1	4	17	0	2090	0
source	0	1	6	6	0	198	0
column_name	0	1	18	25	0	14190	0
values	0	1	6	20	0	4	0

#### Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
V1	2029	1.00	38.46	15.94	1	24	40	49	82	
V3	53340	0.99	1030.71	1035.21	5	350	810	1350	8400	
value	3930	1.00	1830314.09	3878038.92	0	0	244	5268	999999999	