# Coding Protocols for "A Database of the United States Supreme Court's Shadow Docket, 1993-2025"

Jonathan P. Kastellec
Department of Politics
Princeton University
jkastell@princeton.edu

Anthony R. Taboni
Department of Government
University of Texas at Austin
anthony.taboni@austin.utexas.edu

November 26, 2025

# 1 Introduction

This document provides a description of the coding protocols used to create the Shadow Docket Database, including a description of all data sources and scripts used to create the data. To access the data itself and to view the codebook for the visit, please visit the project's website: www.shadowdocketdata.com.

# 2 Summary of Scripts and Data

This section summarizes the various scripts and datasets used and produced in the construction of the dataset. All scripts and data can be found in the zip file *** . If you are interested in just working with the final Shadow Docket Dataset, see www.shadowdocketdata.com/data.

## 2.1 Scripts

This subsection summarizes each of the Python and R scripts used in the data processing procedures.

- `01_Journals_to_Docket_Day.py` scrapes the pdfs of the Journal of the Supreme Court from 1993-2024 and separates the text into docket-days.
- `02_Separate_Actions.py` separates each docket-day into individual actions.
- `03_Scrape_Lower_Court_Dockets_SCOTUS.py` pulls the lower court information for dockets in the 2001-2024 terms.
- `04_Scrape_Dockets_National_Archives.py` pulls the lower court information for dockets in the 1997-2001 terms.
- `05_Process_Lower_Dockets.py` process the scraped docket information.
- `06_Extract_Details.R` categorizes actions and codes relief.
- `07_Merge_Dockets.R` merges together our docket-day-action dataset with information about lower court dockets—generating `shadow_docket_database_vX-X`.

## 2.2 Data

This subsection summarizes each of the data files used in the coding procedures. The specific subsections correspond to the folder names in the Data folder, described here:

- **Data folders**
  - `Data` The top data folder contains the final shadow docket data, which is named `shadow_docket_database_v2-0.csv`. (There are also `.xlsx` and `.dta` versions of the data.) It also contains a separate dataset of hand coded disagreements (`disagreements.xlsx`) and Martin-Quinn scores used to generate Figure 8 in the accompanying paper (`justices.xlsx`).
  - `SC_Journals` This sub-folder contains the raw PDFs files of the Supreme Court Journal, for every term between 1993 and 2024.
  - `Processed_Datasets` This sub-folder contains a variety of intermediate datasets that we post-process after the initial scraping of the PDF files.
    - The sub-folder `Lower_Court_Dockets` contains additional datasets related to linking the Supreme Court docket number with its respective lower court docket number, where applicable.

### 2.2.1 Data/

- `shadow_docket_database_vX-X`: The final Shadow Docket database. Available in .csv, .xlxs, and .dta formats.
- `disagreements.xlsx`: Hand coded votes on non-unanimous actions. Excludes dockets where Justice Stevens disagrees about IFP procedure
- `justices.xlsx`: Martin-Quinn ideology estimates for Supreme Court Justices. Obtained from `https://mqscores.wustl.edu/measures.php`

### 2.2.2 SC_Journals/

- `SCJ_XX.pdf`: Supreme Court journal for the XX term.

### 2.2.3 Processed_Data/

- `Categorized_Docket_Actions.xlsx`: Docket-day-actions with coded action class and relief. Also available as .csv.
- `Coding_Errors_Action_Class.xlsx`: Action classes to be hand coded
- `Coding_Errors_Action_Class_Fixed.xlsx`: Hand coded action classes
- `Coding_Errors_Relief.xlsx`: Reliefs to be hand coded
- `Coding_Errors_Relief_Relief_Fixed.xlsx`: Hand coded relief
- `Docket_Day_Action.xlsx`: Docket-day-actions with no action class or relief coded
- `Docket_Day.xlsx`: Docket-day entries
- `Docket_Information_1997_2024.xlsx`: Information about lower court dockets
- `SC_Bar_Admissions.xlsx`: Admissions to the Supreme Court bar
- Lower_Court_Dockets/
  - `archives_dockets.csv`: National Archive IDs (NAIDs) for dockets 1996-2001
  - `Supreme_Court_Dockets_Raw_Text_2017_2024.json`: Text of docket search website for 2017-2024 terms
  - `Supreme_Court_Dockets_Raw_Text_2001_2016.json`: Text of docket search website for 2001-2016 terms
  - `Supreme_Court_Dockets_1996_2001.json`: Text of docket search from National Archives for 1996-2001 terms
  - `Supreme_Court_Dockets_Missing.json`: Text of docket search for missed dockets from 2001-2024 terms

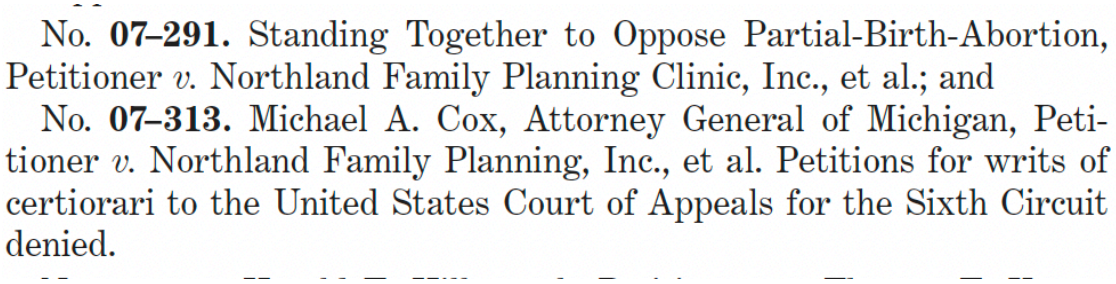
# 3 Parsing the Journal into Docket-Day-Actions

In this section we describe steps are performed in `01_Journals_to_Docket_Day.ipynb` to parse the Supreme Court journal into Docket-Day-Actions.

Before separating the journal into docket-day-actions, we do some initial cleaning and fixing of typos. We then separate the journal into individual days. We split each day into sections using headings assigned by the Court to groups of dockets. These include: Summary Disposition, Orders in Pending Cases, Certiorari Denied, Per Curiam, Habeas Corpus, Mandamus Denied, Prohibition Denied, Rehearings Denied, Attorney Discipline, Admission, Oral Argument, Opinion, Certiorari Granted, Motion Denied, Jurisdiction Postponed, and Order in Pending Case. We then separate each section into individual docket-days by splitting on "`No.`" This gives us `docket_number` and `text`. This generates `Docket_Day.xlsx`.

We then separately scrape each journal to pull out admissions to the bar. Attorney admissions are in a tabular format, allowing us to easily get each attorney, their city, and their sponsor. This information is saved as `SC_Bar_Admissions.xlsx`.

Next, using `02_Separate_Actions.ipynb`, we do additional text cleaning and typo fixes. We clean and standardize `docket_number`, producing `cleaned_docket`. Next, we make sure that each docket is associated with the action the court takes. In some cases, the Court lists multiple dockets and only describes the order after the last docket in the group. This can look something like this:



No. **07–291.** Standing Together to Oppose Partial-Birth-Abortion, Petitioner *v.* Northland Family Planning Clinic, Inc., et al.; and

No. **07–313.** Michael A. Cox, Attorney General of Michigan, Petitioner *v.* Northland Family Planning, Inc., et al. Petitions for writs of certiorari to the United States Court of Appeals for the Sixth Circuit denied.

Dockets are grouped together by either '`;`' or '`; and`'. We assign `text` from 07-313 to `action_text` for both 07-313 and 07-291. If there is only one docket associated with the text,

then action_text=text. This intermediate file is saved as SC_Docket_Day_Clenaed.xlsx.

# 4 Separating separate actions into individual actions

The following actions are also performed by 02_Separate_Actions.ipynb. We then separate each action_text into individual actions using the following regular expression:

1. The\s{0,1}motion
2. The\s{0,1}ap{2,3}lication
3. The\s{0,1}petitions{0,1}
4. The\s{0,1}judgement
5. The order
6. Order entered
7. Petitions{0,1}\s{0,1}for
8. Petitions{0,1}\s{0,1}of
9. Petitions{0,1}\s{0,1}to
10. Motions{0,1}
11. Applications{0,1}
12. Judgment\s*affirmed
13. Judgment\s*reversed
14. The Solicitor General
15. The Acting Solicitor General
16. Certiorari Granted

17. `[cC]ases\s*are\s*consolidated`

18. `directed to`

19. `and petition`

20. `and the petition`

If the text contains multiple instances of one pattern or multiple patterns, each accompanying block of text is classified as its own `specific_action`. If none of these patterns are found, `specific_action` is coded as "Missing."

# 5 Coding Action Classes

The next steps are done using `06_Extract_Details.R`. For each `specific_action`, we classify the `action_class`. The procedure is as follows:

1. First, we generate a docket-day counter for the number of actions taken on a given docket in a given day.

2. If a single action is taken and the Court's specified action is "Certiorari Granted," "Habeas Corpus," "Mandamus Denied," "Prohibition Denied," "Rehearings Denied," we set the `action_class` to the Court's specified action.[1] If there is only one action and it is "Certiorari Denied," we check for an "A" in the docket number. If no "A" is presented, `action_class` is coded as Certiorari. If the action is an admission to the bar, the `action_class` is coded as Admission. Finally, if there is one action and a "D" is present in the docket number, the action is coded as Attorney Discipline.

3. If the Court's action is "Oral Argument" and either 'specific_action" is "Missing" or contains the phrase "divided argument," `action_class` is coded as oral argument.

4. Next, we look at the following classes of action: Orders in Pending Cases, Prohibition Denied, Rehearings Denied, Mandamus Denied, Habeas Corpus, Certiorari Denied,

[1] We code the action as Certiorari or Mandamus, not as Certiorari Granted or Mandamus Denied.

Certiorari Granted, Admission. We use string searching to code `action_class`. Below are patterns used to identify `action_class`. Unless otherwise specified, these patterns are searched for in `specific_action`.[2]

5. Next, we turned to actions that the Court classifies as Opinions per Curiam or Summary Disposition

   All other actions were coded as "Other".

6. For actions that are coded as Other, `action_class` was coded manually. We began with the `specific_action` field. If this field is missing, we relied on `action_text`. The first question we asked is whether the action is simply an error. If `num_actions` $> 1$, we checked `action_text` to see if there were truly two independent actions. If only one action has been taken, we marked the action class as `DUPLICATE`. Next, we determined if the action falls into one of the existing classes.

   If the text of specific action was unclear and/or the text of `Action_Text` was unclear, we went the the relevant year in the Journal of the Supreme Court, searched for the docket number, and found the corresponding docket-date section. Reading this section helped us determine the coding of the action. Sometimes, the court resolved several dockets with one action. This can look something like this:

   No. **07–291.** Standing Together to Oppose Partial-Birth-Abortion, Petitioner *v.* Northland Family Planning Clinic, Inc., et al.; and
   No. **07–313.** Michael A. Cox, Attorney General of Michigan, Petitioner *v.* Northland Family Planning, Inc., et al. Petitions for writs of certiorari to the United States Court of Appeals for the Sixth Circuit denied.

   For most cases, the pre-processing code correctly assigned the text from the second docket as the first's `action_text`. Sometimes typos prevented this. When this oc-

[2]Each `specific_action` only corresponds to one action class. The order in which we search for patterns is non-random and helps minimize misclassification.

curred, we went to the journal and read the text from the last docket to determine `action_class`.

If in the process of coding `action_class`, manually, we found multiple actions listed in a single `specific_action`, we duplicated the row as necessary and coded each action separately. Rarely, we found instances where the pre-processing incorrectly split the text in the middle of the description. This occurred when there is an internal "`No.`" that starts a new line. Such as

> No. **05A578.** C. T. Hanft, United States Navy Commander, Consolidated Naval Brig, Applicant *v.* Jose Padilla. Jose Padilla has filed a petition for certiorari, seeking review of the Fourth Circuit's judgment upholding his detention by military authorities. See *Padilla* v. *Hanft*, No. 05–533 (filed Oct. 25, 2005). On November 22, 2005, the Government filed a motion before the Fourth Circuit, seeking approval to transfer

When this occurred, we first searched the internal docket number (No. 05-533 in this example) to make sure an action was not being attributed to the wrong docket. If this was the case, we noted this as an error. After checking for errors, we used the text from the Journal to manually code the action.[3]

# 6 Coding Relief

We use a similar procedure to above to code `relief`. If the Court specifies the whether it provides relief in its own action, i.e. Certiorari Granted, we use this coding for relief. The following `action_class` categories have no relief coding: Admission to the Bar, Affirmed, Affirmed in Part, DIG, Directed to Brief, Directed to File, GVR, Jurisdiction Postponed, Lack of Quorum, Probable Jurisdiction, Requested to File, Reversal, River Master, and Special Master. These are always coded as NA.

Otherwise, relief is coded as follows:

[3]Note that we relabel some oral argument/amicus actions as Solicitor General. This is done using a combination of the specified `action_class` and the relief. Similarly, we relabel some Certiorari actions as Dismissed as Improvidently Granted (DIG). We elaborate further on in this document.

```
action_class=='Amicus'&str_detect(specific_action,'Solicitor')~'SG',
action_class=='Oral Argument'&str_detect(action_text,'Solicitor')~'SG',
action_class=='Habeas Corpus'& str_detect(action_text,'dismiss')~'Dismissed',
action_class=='Mandamus'& str_detect(action_text,'dismiss')~'Dismissed',
action_class=='Prohibition'& str_detect(action_text,'dismiss')~'Dismissed',
action_class=='Certiorari'& str_detect(action_text,'dismiss')~'Dismissed',
action_class=='Amicus'& str_detect(action_text,'invited to brief and argue')~NA,
action_class=='Consolidate'& str_detect(specific_action,'[Cc]ases\\s{0,1}are\\s{0,1}consolidated')~NA,
action_class=='Consolidate'& str_detect(specific_action,' applications\\s{0,1}are\\s{0,1}consolidated')~NA,
action_class=='Oral Argument'&specific_action=='Missing'~NA,
str_detect(cleaned_docket,'D')~case_when(
  str_detect(action_text,'requested to resign')~'Resigned',
  str_detect(action_text,'suspend')~'Suspended',
  str_detect(specific_action,'reconsider')|str_detect(specific_action,'vacate')~'Challenge',
  str_detect(action_text,'Disbarment\\s{0,1}order\\s{0,1}entered')~'Disbarred',
  .default='Other'),
action_class %in% c('Reversal','Affirmed','GVR','River Master','Special Master','Admission')~NA,
.default=case_when(
  str_detect(action_text, 'pauperis\\s{0,1}and\\s{0,1}the\\s{0,1}petition\\s{0,1}for\\s{0,1}writ\\s{0,1}of\\s{0,1}certiorari')&str_detect(action_text,'granted')~'Granted',
  str_detect(action_text, 'pauperis\\s{0,1}and\\s{0,1}petition\\s{0,1}for\\s{0,1}writ\\s{0,1}of\\s{0,1}certiorari')&str_detect(action_text,'granted')~'Granted',
  str_detect(specific_action,' granted in part and denied in part')~'Granted/Denied',
  str_detect(specific_action,'granted is denied')~'Denied',
  str_detect(specific_action,'granted\\s{0,1}denied')~'Denied',
  str_detect(specific_action,'granted')~'Granted',
  str_detect(specific_action,'Granted')~'Granted',
  str_detect(specific_action,'denied')~'Denied',
  .default = 'Other'
`
```

For relief not captured by the `R` code, we again used manual coding. The `R` code was designed to also push some potential `action_class` errors to this file. Some entries in this file had an `action_class` assigned automatically, but no relief. In coding the relief, we looked for discrepancies between the action specified by the court and the assigned `action_class`. If we found a discrepancy, it was fixed.

Some stays of executions are included in Certiorari Denials. The description will deny the stay and then list "Petition for writ of certiorari to the XXX." without saying denied. These are marked as denied. Similarly, sometimes the description will just describe the request for stay and then deny the petition for cert or a petitions for habeas corpus. These stays are marked as denied.

> CERTIORARI DENIED
>
> No. **13–9776 (13A1053).** Robert Eugene Hendrix, Petitioner *v.* Florida. Application for stay of execution of sentence of death presented to Justice Thomas and by him referred to the Court. Petition for writ of certiorari to the Supreme Court of Florida denied.

## 6.1 Relief Codings

### 6.1.1 Coded as NA

The following do not have a coding for relief and are coded as NA:

- Admission to the Bar
- Affirmed

- Affirmed in Part
- DIG
- Directed to Brief
- Directer to File
- GVR
- Jurisdiction Postponed
- Lack of Quorum
- Probable Jurisdiction
- Requested to File
- Reversal
- River Master
- Special Master

#### 6.1.2 Granted, Denied, NA

The next set are coded as granted if the court grants the motion, denied if the court denies a motion, or NA if instead the court takes action directly. If the action is granted in part and dismissed in part code as Granted/Denied.

- Abeyance
- Amend
- Appointment of Counsel
- Bail
- Bill of Complaint
- Briefing
- Certificate of Appealability
- Consolidate
- Costs and Fees
- Damages
- Defer Action
- Enlarge Record

- Expedite Action
- File Under Seal
- Injunction
- Leave to File
- Oral Argument
- Printing
- Pro Hac Vice
- Release
- Remand
- Retax
- Sanctions
- Seamen
- Solicitor General
- Strike
- Substitution
- Vacate
- Vacate and Remand
- Vacate Injunction
- Vacate Stay
- Veteran

#### 6.1.3 Granted, Denied, Dismissed, NA

The next set are coded as granted if the court grants the motion, denied if the court denies a motion, or dismissed if the court dismissed the petition/motion, and NA if the court takes an action (not dismissing) directly. If the action is granted in part and dismissed in part code as Granted/Denied.

- Certiorari
- Dismiss
- Habeas Corpus

- In forma pauperis
- Leave to Intervene Mandamus
- Prohibition
- Rehearings
- Stay

#### 6.1.4 Unique Codings

**Amicus**

1. Granted
    - Granting petition for leave to file amicus
2. Denied
    - Denying petition for leave to file amicus
3. Missing
    - Missing relief

**Out of Time**

1. Granted
2. Denied
3. Missing

**Attorney Discipline**

Attorney discipline can be coded in the relief column. The processing code will create a new variable `disciplinary_action` and set the relief coding to `NA`.

1. Disbarment
    - Court disbars an attorney
2. Suspended
    - Attorney suspended as part of disbarment
3. Resigned
    - Attorney resigned from the bar
4. Challenge
    - Challenging disbarment
5. Misc
    - Other actions

# 7 Finishing up the Data

After coding `action_class` and `relief`, we made some slight modifications to restructure the data. First, we added a new `action_class`, Solicitor General, which captures actions involving the solicitor general. This was identified as follows:

```
mutate(action_class=case_when(
  action_class == 'Amicus'&relief=='SG'~'Solicitor General',
  action_class == 'Oral Argument' & relief=='SG'~'Solicitor General',
  action_class %in% c('Oral Argument','Amicus')&relief %in% c('Granted','Denied')&str_detect(action_text,'[sS]olicitor')~'Solicitor General',
  .default = action_class),
```

Next, we recoded recode certiorari petitions that dismiss as improvidently granted as their own cateogry.

```
mutate(action_class=case_when(
  action_class == 'Certiorari'&str_detect(action_text,'improvidently granted')~'DIG',
  .default = action_class),
  relief = ifelse(action_class=='DIG','NA',relief))
```

Next, we removed oral arguments. Actions relating to argument remain but oral arguments themselves are removed.

```
#Remove oral arguments
completed_dockets <- completed_dockets %>%
  mutate(action_class = ifelse((action_class=='Oral Argument'|action_class=='Solicitor General')&str_detect(action_text,'[aA]rgued by'),'Arguments',action_class))%>%
  filter(action_class!='Arguments')
```

Finally, we recoded some reversals as GVRs or orders to vacate and remand.

# 8 Coding Additional Variables

After finishing coding `action_class` and `relief`, we added several additional variables to the data set.

## 8.1 Relief Granted

`relief_granted` is equal to one if `relief`==“Granted,” 0 if `relief`==“Denied,” and NA otherwise.

## 8.2 Emergency Applications

We coded emergency applications as those whose `action_class` was in {Stay, Injunction, Vacate Stay, Vacate} **and** were associated with docket number contained “A.” To search for

```
mutate(action_class=case_when(
  action_class == 'Reversal'&cleaned_docket == '13-9085' ~'Reversal',
  action_class == 'Reversal'&str_detect(action_text,'U.\\s*S.')~'GVR',
  action_class == 'Reversal'&str_detect(action_text,'[pP]er\\s*curiam')~'Reversal',
  action_class == 'Reversal'&str_detect(action_text,'opinion of this Court')~'Reversal',
  action_class == 'Reversal'~'Vacate and Remand',
```

the docket number we used both `cleaned_docket` and `text`. Occasionally, the court gives two different docket numbers to the same application, only one of which contains "A". An example is presented below.

> No. **04–10166 (05A170).** Robert Alan Shields, Jr., Petitioner *v.* Doug Dretke, Director, Texas Department of Criminal Justice, Correctional Institutions Division. Application for stay of execution of sentence of death presented to Justice Scalia and by him referred to the Court denied. Petition for writ of certiorari to the United States Court of Appeals for the Fifth Circuit denied.

Before the 2003 term, the Court does not consistently denote applications with a docket containing "A." Therefore, emergency applications is equal to NA for all cases prior to the 2003 term. Emergency application is also coded as NA for docket-days with the `action_class` Attorney Discipline or Admission.

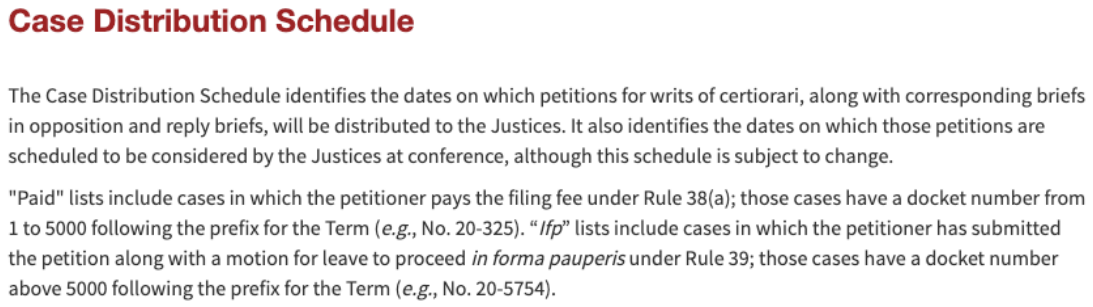
## 8.3 Cert Granted

`cert_granted` denotes whether the action grants or denies a petition for cert. It is equal to 1 if `action_class`=="Certiorari" and `relief`=='Granted," 0 if `action_class`=="Certiorari" and `relief`=='Denied," and NA otherwise.

## 8.4 Cert Type

`cert_type` denotes whether the petition for cert is paid or is filed `in forma pauperis`. As a reminder, most docket numbers are of the form YY-XXXX. If XXXX<5000, the petition is paid. If XXXX>5000, the application is filed *in forma pauperis*.

Note this approach prevents us from classifying dockets of the form YYAXXX as paid or `in forma pauperis`.



**Case Distribution Schedule**

The Case Distribution Schedule identifies the dates on which petitions for writs of certiorari, along with corresponding briefs in opposition and reply briefs, will be distributed to the Justices. It also identifies the dates on which those petitions are scheduled to be considered by the Justices at conference, although this schedule is subject to change.

"Paid" lists include cases in which the petitioner pays the filing fee under Rule 38(a); those cases have a docket number from 1 to 5000 following the prefix for the Term (*e.g.*, No. 20-325). "*Ifp*" lists include cases in which the petitioner has submitted the petition along with a motion for leave to proceed *in forma pauperis* under Rule 39; those cases have a docket number above 5000 following the prefix for the Term (*e.g.*, No. 20-5754).

## 8.5 Death Penalty

`death_penalty` is equal to 1 if '`execution`' appears in any `action_text` associated with the docket; otherwise it is equal to 0. It is coded as NA when `non_case`==1.

## 8.6 Dissent

`dissent` is equal to 1 for a docket-day if '`dissent`' appears in the `action_text`; otherwise it is equal to 0.

## 8.7 Disagreement

`disagreement` is equal to 1 for a docket-day if either `dissent`==1 or '`would grant`' or '`would deny`' appear in the `action_text`; otherwise it is equal to 0.

## 8.8 Non-case

`non_case` is coded as 1 if `action_class` is Attorney Discipline or Admission. Otherwise it is 0.

## 8.9 Discipline Action

`discipline_action` specifies the type of action taken for dockets falling into the Discipline Action `action_class`. It is coded as follows:

1. Disbarment
    - Court disbars an attorney
2. Suspended

- Attorney suspended as part of disbarment

3. Resigned
   - Attorney resigned from the bar
4. Challenge
   - Challenging disbarment
5. Misc
   - Other actions

For all other categories of `action_class` it is coded as NA.

## 8.10 Petitioner

`petitioner` lists the petitioner in the case. It is identified via string matching on the `text` variable. We look for patterns ending in "Petitioner," "Applicant," or starting with "Original."

```
mutate(petitioner = str_match(text, '[-\\d\\)]\\. (.*?), Petitioner')[,2],
       applicant =  str_match(text, '[-\\d\\)]\\. (.*?), Applicant')[,2],
       original = str_match(text, 'Original\\.(.*?)v\\.')[,2],
       petitioner = case_when(is.na(petitioner)&!is.na(applicant)~applicant,
                              is.na(petitioner)&!is.na(original)~original,
                              is.na(petitioner)~str_match(text, '[-\\d\\)]\\s{0,1}\\. (.*?)v\\.')[,2],
                              .default = petitioner),
```

If no petitioner can be identified, this variable is coded as NA.

## 8.11 Respondent

`respondent` is also coded using `text`. We search for patterns coming after the"v." as follows:

If no respondent can be identified, this is coded as NA.

The file is then saved as `Categorized_Docket_Actions.xlsx`.

```
respondent = str_match(text, 'Petitioners{0,1}\\s{0,1}v\\.(.*?)[\\.;]')[,2],
applicant = str_match(text, 'Applicants{0,1}\\s{0,1}v\\.(.*?)[\\.;]')[,2],
petition = str_match(text, 'Petitioners{0,1}\\s{0,1}v\\.(.*?)[\\.;] Petition')[,2],
motion = str_match(text, 'Petitioners{0,1}\\s{0,1}v\\.(.*?)[\\.;] The motion')[,2],
motion1 = str_match(text, 'Petitioners{0,1}\\s{0,1}v\\.(.*?)[\\.;] Motion')[,2],
naive = str_match(text,'v\\.(.{10}.*?)[\\.;]')[,2],
respondent = case_when(is.na(respondent)&!is.na(applicant)~applicant,
                       nchar(respondent)<15&!is.na(petition)&nchar(petition)<75~petition,
                       nchar(respondent)<15&!is.na(motion)&nchar(motion)<75~motion,
                       nchar(respondent)<15&!is.na(motion1)&nchar(motion1)<75~motion1,
                       (is.na(respondent)|nchar(respondent)<15)&nchar(naive)<75~naive,
                              .default = respondent)
```

# 9 Lower Court Dockets

To assist scholars who are interested utilizing the data to study the judicial hierarchy, we have merged in lower court dockets for appeals from the 1997 term onward. From the 2001 term onward, the Supreme Court's website provides digitalized versions of their docket book, which contains information about the lower court case and docket number(s). For each docket, we scraped this website for information about the identity of the lower court and the associated docket number. This relevant scraping code can be found in `03_Scrape_Lower_Court_Dockets_SCOTUS.ipynb`

For dockets between 1997 and 2001, the docket information is provided online through the National Archives. This code can be found in `04_Scrape_Dockets_National_Archives.ipynb`.

These dockets are processed and information about attorneys and parties is extracted using `05_Process_Lower_Dockets.ipynb`. The following variables are obtained using this data:

## 9.1 Lower Court

If the case is an appeal, `lower_court` gives the source of appeal

## 9.2 Lower Docket

If the case is an appeal, `lower_docket` gives the associated docket number from the lower court

### 9.3 Petitioner Counsel

`petitioner_counsel` provides information about who the counsel of record for the petitioner(s) is (are)

### 9.4 Government Petitioner

`gov_petitioner` is equal to 1 if variants of the phrase "United States Department of Justice" is listed among the petitioner counsel information. It is 0 if this phrase is not present. It is `NA` if we are unable to identify petitioner counsel.

### 9.5 Respondent Counsel

`respondent_counsel` provides information about who the counsel of record for the respondents(s) is (are)

### 9.6 Government Respondent

`gov_respondent` is equal to 1 if variants of the phrase "United States Department of Justice" is listed among the respondent counsel information. It is 0 if this phrase is not present. It is `NA` if we are unable to identify respondent counsel.

### 9.7 Other Parties

`other_parties` provides information about other parties and attorneys involved in the case

## 10 Creating the Final Dataset

With the lower court dockets in hand, we merge together information about lower courts with the shadow docket actions. `07_Merge_Dockets.R` performs this operation. The resulting data set is `shadow_docket_database_vX-X`. It is available in .csv, .xlsx, and .dta formats.