RANDOM ASSIGNMENT OF JUDGES - MODEL AND MEASUREMENT

January 16, 2021

Kailiang Fu
Linghai Liu
Jingxin Zhang [1]

[1]Authors contributed equally to this work.

Dear Attorney General Neronha,

Hope this letter finds you well!

We are a team of students at Brown University. We recently carried out a project on the judge assignment process in the judicial system. Random assignment is typically needed in order to guarantee fairness. However, in practice, such random assignment often does not occur, leading to issues such as the stop-and-frisk challenge in Southern District of New York. Thus, here we propose a random assignment model which could potentially improve the randomness of assignment.

We examined the JUSTFAIR data set of criminal sentencing decisions made in federal district courts from 2001-2019 and did notice certain districts that perform badly at random assignment. While Rhode Island has done an excellent job, ranking first among all districts by our measurement method, our random assignment model still has demonstrated noteworthy improvements of randomness during our experiments. In addition, our model is fully automatic and could be easily tailored for different circumstances if ever needed. Full details could be found in the attached technical report. We truly hope that the model we have implemented could be put into practice and make a difference for our judicial system.

We would appreciate it very much if you could kindly take our proposal into consideration.

Sincerely,
Kailiang Fu, Linghai Liu, Jingxin Zhang

1 Introduction

A judge is the voice of justice. Judges must be impartial and strive to properly interpret the meaning, significance, and implications of the law [7]. Only in this way, our liberties and our most fundamental and sacred rights will be protected as set forth in the Bill of Rights, as well as from unlawful and unwarranted intrusion into our lives from the government [9].

In a court, justice shall be extended to both sides (including the defendant). As pointed out, the purposes of the United States Sentencing Commission is to provide certainty and fairness in meeting the purposes of sentencing, avoiding unwarranted sentencing disparities among defendants with similar records who have been found guilty of similar criminal conduct, while preserving flexibility concerning individual cases [1]. Although judicial discretion has been constrained by sentencing guidelines and other means of structuring the sentencing decision since the 1970s, controversy for potential disparities in sentencing still exist [6]. Some studies have shown that those who are punished more severely, which might due to disparity, become more likely to reoffend [8]. Random assignment of judges can potentially address this issue, since the sentence that the criminal is to receive is a function of chance [5].

We believe that the random assignment of judges can alleviate the effects of disparity while potentially contribute to deterrence of future conducts. There exist practical instructions on randomization process [2], yet it is inefficient for its complicacy. There are legitimate constraints on random assignment of judges, including scheduling conflicts and conflict of interest. A judge would be excluded from the assignment of a case in which the judge's impartiality might reasonably be questioned [3]. We take these factors into consideration when designing our strategy for random assignment and measurement, as explained in detail in Section 2.

2 Model

2.1 Random Assignment

We intend to devise a strategy for random assignment of cases to judges that can be applied to all districts. Meanwhile, There are legitimate reasons that a judge should not be assigned to a case.

2.1.1 Initial Considerations

- **Time Conflict**: Before assigning a case to certain judge, we need to consider whether the judge is available at the given period. Possible reasons for time conflicts include but are not limited to period of leave and overlap with other cases.
- **Workload**: The number of cases a judge can be assigned to is limited, which can vary based on the judge's experience and the time commitment of each case.
- **Conflict of Interest**: A judge shall disqualify himself or herself in any proceeding in which the judge's impartiality might reasonably be questioned [3]. According to Rule 2.11, a judge is disqualified for a case when the judge knows the the accusers or defendants, individually or as a fiduciary, or the judge's spouse, domestic partner, parent, or child, or any other member of the judge's family residing in the judge's household, has an economic interest in the subject matter in controversy or in a party to the proceeding.
- **Personal Bias**: We should also exclude the judges who have personal bias or prejudice concerning a party or a party's lawyer, or personal knowledge of facts that are in dispute in the proceeding to guarantee the fairness of the trial.

2.1.2 Simplifying Assumptions and Set Variables

We made the following simplifying assumptions for creating this assignment strategy.

- The time conflict can be solved by registering each judge's unavailable time periods as a list of tuples where the first and second elements of each tuple represent the starting and ending of that unavailable

time slot, respectively. Each Case will also have the variable time to indicate when that case will be held. Therefore, by avoiding overlap between the busy periods of the judge and the time of the case, we are able to guarantee the assignment would not be in conflict with a judge's schedule.

- Considering each judge's experience varies from another, We indicate each judge's capability of hearing cases using max_cases defined in Judge class. We also have the total variable to indicate how many cases has been assigned to the judge for now. If assigning certain case will exceed the upper case limit of a judge, the assignment model will assign that case to another judge who can be assigned with more cases.

- To avoid conflict of interests and personal bias, we create a variable conflict_judges in the Defendant class to indicate which judges have relations or conflicts with the defendant. Since the case includes the defendant subject, we can check whether a judge is qualified for the case by checking whether he or she is on conflict_judges of the defendant.

2.1.3 Assignment Implementation

After converting restrictions in reality to manageable variables in classes, we can now implement the assignment strategy in Python:

1. The random assign model takes two inputs: the list of judges and the case to be assigned. The later also includes the information of the defendant.

2. We will first exclude all judges who are in the conflict_judges of the case's defendant. By doing which, we can avoid assigning the case to someone who has personal bias or interest conflict on this trial.

3. Then we randomly select a judge from the processed list and remove the judge from the list. Then, the model will check whether the judge is valid for this case or not.

4. The valid check is done by function valid, which will return true if and only if the judge has not reached his or her maximum case limit (total should be less than max_cases) and has no time conflict with the case (no overlap between busy_time of the judge and time of the case).

5. If valid, we return the id of the judge. Otherwise, we recursively call random_assign on the rest of the judges list until get a valid judge id or -1, indicating no judge on the list is available for the case.

2.2 Randomness Measurement

In a satisfactory result of the random assignment, we expect to observe a uniform distribution across all types of crime cases. Using the factors provided in the JUSTFAIR data set [4], we can enumerate over all types of cases, and investigate the proportion of each type of cases among all the cases heard by each judge.

2.2.1 Initial Considerations

Here are a list of factors selected from the JUSTFAIR data set [4] that we consider important to investigate and will be used in our implementation.

- **AGE**: age of the defendant at the time of sentencing (15-105, we categorized ages into $\leq$ 20, 21-30, 31-40, 41-50, 51-60, and $\geq$ 61)

- **MONSEX**: gender of the offender (0 = Male, 1 = Female, .= Missing, Indeterminable, or Inapplicable)

- **NEWRACE**: race of the defendant (1 = White, 2 = Black, 3 = Hispanic, 6 = Other, . = Missing, Indeterminable, or Inapplicable)

- **NEWEDUC**: highest level of education for offender (1 = Less Than H.S. Graduate, 3 = H.S. Graduate, 5 = Some College, 6 = College Graduate, . = Missing, Indeterminable, or Inapplicable)

- **CRIMHIST**: indication as to whether the defendant has any criminal history or law enforcement contacts (0 = No Criminal History, 1 = Yes, There is Criminal History, . = Missing, Indeterminable, or Inapplicable)
- **CASETYPE**: type of the case (1 = Felony, 2 = Misdemeanor A, 3 = Misdemeanor B/C (Should not exist), . = Missing or Indeterminable)

2.2.2 Proposed Method

For a given assignment in a given district, we propose a method of randomness measurement based on variance calculation. Suppose there are N combinations of the factors of interest and M judges in the given district. For each combination C_i and for each judge j, we find out the proportion of cases of this combination among all the cases heard by this judge, denoted as p_{ij}. For each combination C_i, we compute the variance of $p_{i1}, \cdots, p_{iM}$, denoted as v_i. We sum up all the v_i's weighted by proportion of cases of corresponding C_i among all the cases in the data set to obtain a score of randomness. The closer the score is to 0, the more random the assignment is. Full implementation of this method is attached in Appendix.

Algorithm 1 Randomness Measurement

Require: *dataset* (JUSTFAIR), F (factors of interest)
  Set *map*1, *map*2 as empty hash maps.
  for *case* in *dataset* **do**
    Get combination C of F of the *case*
    *map*1[C]++, *map*2[*case.judge*][C]++, *map*2[*case.judge*][*total*]++
  end for
  for *crimetype* in *map1.keys* **do**
    Calculate the variance v of the proportion of *crimetype* among all judges
  end for
  return *score* = sum up v using proportion of each *crimetype* among *dataset* as weight

3 Results

We apply our measurement method 2.2 to evaluate the randomness of judge assignment in each district where defendants were sentenced. In order to better compare and analyze the performance of all districts, we normalize the obtained scores using the following equation to scale randomness to a value in [0, 10], and the higher the value, the better the performance:

$$x' = 10 - 10 \cdot \frac{x - \min x}{\max x - \min x}$$

where x stands for each score in L.

The head of the result table 1 we have obtained are shown below. The fact that most of the districts have a randomness score close to 10 is due to the existence of outliers (e.g. Puerto Rico) to a great extent. For a majority of states, their relative randomness scores as shown are decent, while significant variations do exist. As we can see in the table, Rhode Island in particular has done an excellent job at randomly assigning cases to judges with a score of 9.999, which is the highest score among all districts.

We also test our random assignment model 2 on two districts, Rhode Island and New York North. Since our model is indeterministic when assigning cases to current judges in the districts, we run our random assignment model for 50 times and calculate the average randomness score as our statistic. The scores for both the original JUSTFAIR data set and our assignment model are demonstrated in table 2. From the data, we can observe a significant increase of randomness using the model we introduced.

index	state	randomness score
01	Dist of Columbia	9.413
02	Maine	9.935
03	Massachusetts	9.921
04	New Hampshire	10.000
05	Puerto Rico	0.000
06	Rhode Island	9.999
07	Connecticut	9.912
08	New York East	9.643
09	New York North	9.844
10	New York South	9.620

Table 1: Evaluated randomness of the first 10 districts

District	raw	generated
Rhode Island	1.39×10^{-4}	7.66×10^{-5}
New York North	1.62×10^{-3}	1.43×10^{-4}

Table 2: Comparison of randomness of raw and generated data

4 Discussion

We have introduced our model in section 2 and displayed the results we obtained in section 3. According to our proposed method for measuring randomness, we have observed that the assignments obtained by our model are significantly more random than those in the JUSTFAIR data set [4], which demonstrated the validity and feasibility of our model.

4.1 Advantages

- Our measurement of randomness is numerically tractable and comprehensible, as it is the weighted sum of variances across various crime types assigned to all the judges. By applying normalization to all randomness values corresponding to districts, we can compare their relative fairness of trial results based on readily available data sets such as [4].
- Our assignment model have demonstrated noteworthy improvements in the aspect of randomness. For districts with a small number of judges, such as Rhode Island and New York North, we have calculated the randomness of our results compared with that of the raw data, and it increases by approximately 10 times based on our proposed measurement of randomness.
- Our model is flexible in terms of implementation. In the implementation in the Appendix, we only take a few demographic variables into consideration. However, we can easily add more variables of interest and get more tailored implementation.
- Our model is efficient after implementation. Compared with the old-fashioned guidelines of drawing cards, ordering, and enclosing them with case files, our model is digitalized and automatic. Although it might elicit some inconvenience at the beginning, processing with computerized model can improve efficiency in the long run and help with data collection for further studies into trial data.

4.2 Disadvantages

- Our model only considers the demographic information of the defendants, while the information concerning the judge is not studied, neglecting the variability of judges in the judicial system.
- When evaluating the relative degree of randomness decomposing by districts, more information concerning that specific states could be incorporated, including the number of judges in that district, the

age and race constitution of the population of the district, etc. Based on our measurement, degree of randomness would appear worse if a district has more judges.

- We could improve our normalization algorithm. As several outlier districts do exist, the rest of the data would cluster on a small interval (in our case, [8, 10]), which weakens the differences among these districts.

4.3 Future Work

- More variables could be taken into consideration, such as crime type (in the usual sense). It is probable that the fields of expertise of judges vary. Assigning different types of cases to different judges in a specific circumstance might be reasonable since it helps reduce potential disparity due to unfamiliarity of the judge. At the same time, we need to evaluate the contribution of each variable towards the assignment. Variables with trivial weights may have to be chunked out to simplify the model and increase calculation speed.

- Factors of the district itself should be taken into consideration as well. As mentioned above, there exists the possibility that the number of judges or the constitution of the population would affect the randomness that we try to evaluate for the district. A more detailed framework could be built with the inclusion of these information.

References

[1] 28 u.s. code § 991 - united states sentencing commission; establishment and purposes. https://www.law.cornell.edu/uscode/text/28/991.

[2] Rule 8 assignment of judges and duty judge. https://www.washingtongov.org/DocumentCenter/View/206/Rule-8---Assignment-of-Judges-and-Duty-Judge-PDF?bidId=#:~:text=If%20a%20criminal%20defendant%20has,go%20through%20the%20selection%20process.

[3] American Bar Association. *Model Code of Judicial Conduct*. ABA Book Publishing, 2020. Canon 2, Rule 2.11: Disqualification.

[4] Maria-Veronica Ciocanel, Chad M Topaz, Rebecca Santorella, Shilad Sen, Christian M Smith, and Adam Hufstetler. Justfair: Judicial system transparency through federal archive inferred records, Jul 2020.

[5] Donald Green and Daniel Winik. Using random judge assignments to estimate the effects of incarceration and probation on recidivism among drug offenders. *Criminology*, 48, 10 2009.

[6] Charles W. Ostrom ; Brian J. Ostrom ; Matthew Kleiman. Judges and discrimination: Assessing the theory and practice of criminal sentencing. https://www.ncjrs.gov/App/Publications/abstract.aspx?ID=204024.

[7] Rhode Island Department of Education (RIDE). What is the role of a judge? https://www.ride.ri.gov/Portals/0/Uploads/Documents/Instruction-and-Assessment-World-Class-Standards/Transition/EIA-CCSS/BlanchetteC-Trial_Alexander-Role_of_a_Judge.pdf.

[8] Cassia Spohn and David Holleran. The effect of imprisonment on recidivism rates of felony offenders: A focus on drug offenders. *Criminology*, 40(2):329–358, May 2002.

[9] William K. Weisenberg. Why our judges and courts are important. https://www.abajournal.com/news/article/why_our_judges_and_courts_are_important#:~:text=Judges%20and%20courts%20exist%20to,justice%2C%20there%20is%20no%20freedom.

5 Appendix

```
class Judge:
    def __init__(self, jid: int, mc: int, tl: List[tuple], n:int):
        self.jid = jid # judge id
        self.max_cases = mc
        self.busy_time = tl
        self.total = n

class Case:
    def __init__(self, t: tuple, defendent: Defendent):
        self.time = t
        self.defen = defendent

class Defendant:
    def __init__(self, cid: int, jjs: List[Judge]):
        self.cid = cid # criminal id
        self.conflict_judges = jjs
```

Listing 1: class definitions

```
import random

def valid(j: Judge, c: Case)->bool:
    '''
    Check whether a selected judge assigned to c is valid
    :param j: the judge selected
    :param c: the case to which j is assigned
    '''
    if j.total+1>j.max_cases:
      return False  # cannot exceed max_cases
    for t in j.busy_time:
      if t[0]<=c.time[0]<=t[1] or t[0]<=c.time[1]<=t[1]:
        return False # cannot fall in busy_time intervals
    return True


def random_assign(jjs: List[Judge], c: Case)-> int:
  '''
  Assign randomly considering constraints
  :param jjs: a list of judges
  :param c: the case to be assigned
  :return : a judge id (jid) that represents the judge assigned to this case, or -1 if none
    is available
  '''
  judges = jjs.copy()
  ret = -1

  for j in c.defen.conflict_judges:
    if j in judges:
      judges.remove(j) # exclude those with personal bias of interest

  if not bool(judges):
    return ret # if the processed list of judges is empty
  else:
    ret = random.choice(judges) # randomly choose a judge from list
    judges.remove(ret) # remove the judge from the list

  if valid(ret, c):
    return ret.jid # return if this selected judge is random
  else:
    return random_assign(judges, c) # random assign again if not valid
```

Listing 2: random assignment

```
gender_map = {0:'m', 1:'f', 100:'#'}
race_map = {1:'w', 2:'b', 3:'h', 6:'o', 100:'#'}
```

```
edu_map = {1:'1', 3:'3', 5:'5', 6:'6', 100:'#'}
age_map = {1:'1', 2:'2', 3:'3', 4:'4', 5:'5', 6:'6', 100:'#'}
crimehist_map = {0:'0', 1:'1', 100:'#'}
def case2string(case):
  gender = case['MONSEX']
  race = case['NEWRACE']
  educ = case['NEWEDUC']
  age = case['AGE']
  crimehist = case['CRIMHIST']
  if not (gender >= 0 and gender < 2):
    gender = 100
  if not (race > 0 and race < 7):
    race = 100
  if not (educ > 0 and educ < 7):
    educ = 100
  if not (age >=15 and age <= 105):
    age = 100
  else:
    age = (age+1) // 10
  if age > 6 and age < 100:
    age = 6
  if not (crimehist >= 0 and crimehist < 2):
    crimehist = 100
  return gender_map[gender]+race_map[race]+edu_map[educ]+age_map[age]+crimehist_map[
    crimehist]

  def transformDF(c_data, flag):
  '''
  Transform the dataframe from 'by case' to 'by judge'.
  :param c_data: criminal data, given by JUSTFAIR dataset
  :param flag: true for original data, false for generated data
  '''
  dictS = {} # stores classifications
  dictL = {} # stores nested dictionaries
  for i in c_data.index:
    if flag:
      case = c_data.loc[i, ['MONSEX', 'NEWRACE', 'NEWEDUC', 'AGE', 'CRIMHIST', 'judge']]
    else:
      case = c_data.loc[i, ['MONSEX', 'NEWRACE', 'NEWEDUC', 'AGE', 'CRIMHIST', 'RANDOM_JUDGE
    ']]
    c = case2string(case)
    if c in dictS.keys():
      dictS[c] += 1
    else:
      dictS[c] = 1
    if flag:
      j = case['judge']
    else:
      j = case['RANDOM_JUDGE']
    if j in dictL.keys():
      dictL[j]['total'] += 1
      if c in dictL[j].keys():
        dictL[j][c] += 1
      else:
        dictL[j][c] = 1
    else:
      dictL[j] = {c:1, 'total':1}
  return dictS, dictL

  def getRandomness(c_data, dictS, dictL):
  '''
  Evaluate randomness given a dataset
  :param c_data: criminal data
  :param dictS: dictionary containing classifications
  :param dictL: dictionary using judge names as keys, dictionaries of counts of all cases
    the judge heard as values
  '''
  output = 0
```

```
  N = len(c_data)
  for crimetype in dictS.keys():
    v = []
    for judge in dictL.keys():
      if crimetype in dictL[judge].keys():
        v.append(dictL[judge][crimetype] / dictL[judge]['total'])
      else:
        v.append(0)
    output += (dictS[crimetype] / N) * (np.var(v))
  return output

  def getDistrictRandomness(d_data):
  '''
  Evaluate the randomness of the district.
  :param d_data: subset of dataset of the district
  '''
  d_dictS, d_dictL = transformDF(d_data, False)
  d_output = getRandomness(d_data, d_dictS, d_dictL)
  return d_output

  def normalize(district_randomness):
  '''
  Normalize randomness
  :param district_randomness: the list containing randomness of each district
  '''
  base = min([i for i in district_randomness if i != 0])
  ma = max(district_randomness)
  normalized_randomness = []
  for r in district_randomness:
    if r-0==0:
      normalized_randomness.append(None)
    else:
      normalized_randomness.append(round(-(20*(r-base)/(ma-base)-10), 3))
  return normalized_randomness
```

Listing 3: randomness measurement