Reading data

Resultados de las elecciones al Parlamento de Navarra 2019 agrupados por mesas electorales

DESCRIPCIÓN

Resultados definitivos de las elecciones celebradas el año 2019 agrupados por cada mesa electoral. Se incluye merindad, código de municipio, distrito, sección, mesa (distrito, sección, mesa), censo, participación, abstención, votos válidos, etc. así como los datos que obtuvo cada uno de los partidos políticos (votos y porcentaje).
El código de municipio 990 corresponde al registro de residentes ausentes.
Contiene línea de totales.

FICHA TÉCNICA

› **TEMA:** Administración pública

› **CATEGORÍA:** Elecciones

› **DEPARTAMENTO:** Departamento de Presidencia, Función Pública, Interior y Justicia

› **LUGAR:** Navarra

› **LICENCIA:** Creative Commoms by 4.0

› **FECHA DE CREACIÓN:** 11/06/2019

› **FECHA DE ACTUALIZACIÓN:** 11/06/2019

› **FRECUENCIA DE ACTUALIZACIÓN:** Datos inamovibles

› **ETIQUETAS:** Elecciones, Navarra

Envíanos tu opinión. Comenta este conjunto de datos

DESCARGAS

XML eXtensible Markup Language

JSON JavaScript Object Notation

CSV Comma-separated values

ODS OpenDocument Spreadsheet

XLS Microsoft Office Excel

Síguenos en Twitter @opendata_na

Suscríbete a los cambios de Open Data

*.xlsx

	A	B	C	D	E	F	G	H	I
1	Codcir	Codmun	Municipio	Mesa	Censo	Certif. Alta	Certif. Correc.	Censo Total	Votos Electores
2	31	1	Abáigar	1-001 -U	80	0	0	80	59
3	31	2	Abárzuza / Abartzuza	1-001 -U	431	0	0	431	290
4	31	3	Abaurregaina / Abaurrea Alta	1-001 -U	117	0	0	117	94
5	31	4	Abaurrepea / Abaurrea Baja	1-001 -U	32	0	0	32	29
6	31	5	Aberin	1-001 -U	281	0	0	281	177
7	31	6	Ablitas	1-001 -A	445	0	0	445	326
8	31	6	Ablitas	1-001 -B	451	0	0	451	341
9	31	6	Ablitas	2-001 -A	495	0	0	495	412
10	31	6	Ablitas	2-001 -B	481	0	0	481	399
11	31	7	Adiós	1-001 -U	126	0	0	126	96
12	31	8	Aguilar de Codés	1-001 -U	64	0	0	64	52
13	31	9	Aibar / Oibar	1-001 -U	669	0	0	669	501
14	31	11	Allín / Allin	1-001 -A	207	0	0	207	165
15	31	11	Allín / Allin	1-001 -B	245	0	0	245	181
16	31	11	Allín / Allin	1-001 -C	252	0	0	252	182
17	31	12	Allo	1-001 -U	765	0	0	765	558
18	31	10	Altsasu / Alsasua	1-001 -A	753	0	0	753	550
19	31	10	Altsasu / Alsasua	1-001 -B	728	0	0	728	517

*.CSV

```
Codcir;Codmun;Municipio;Mesa;Censo;Certif. Alta;Certif. Correc.;Censo Total;Votos Totales...
31;1;Abáigar;1-001 -U;80;0;0;80;59;0;59;0;0;18;10;0;4;19;0;0;0;1;0;7
31;2;Abárzuza / Abartzuza;1-001 -U;431;0;0;431;290;0;290;1;3;99;45;4;37;71;1;3;0;2;0;24
31;3;Abaurregaina / Abaurrea Alta;1-001 -U;117;0;0;117;94;0;94;0;1;21;33;1;12;21;2;0;0;0;0;3
31;4;Abaurrepea / Abaurrea Baja;1-001 -U;32;0;0;32;29;0;29;0;0;13;6;0;5;4;0;0;0;0;0;1
31;5;Aberin;1-001 -U;281;0;0;281;177;0;177;2;2;75;23;3;36;22;0;1;0;2;1;10
31;6;Ablitas;1-001 -A;445;0;0;445;326;0;326;5;4;134;9;10;133;20;0;5;0;0;0;6
31;6;Ablitas;1-001 -B;451;0;0;451;341;0;341;3;0;143;8;10;143;10;0;10;2;1;0;11
31;6;Ablitas;2-001 -A;495;0;0;495;412;0;412;7;6;204;3;8;128;26;0;6;4;1;0;19
31;6;Ablitas;2-001 -B;481;0;0;481;399;0;399;1;5;198;7;8;143;17;0;5;0;1;2;12
```

*.xml

```
<table:table-row table:style-name="ro2">
     <table:table-cell table:style-name="ce2" office:value-type="float"
                       office:value="31" calcext:value-type="float">
      <text:p>31</text:p>
     </table:table-cell>
     <table:table-cell table:style-name="ce2" office:value-type="float"
                       office:value="1" calcext:value-type="float">
      <text:p>1</text:p>
     </table:table-cell>
     <table:table-cell table:style-name="ce2" office:value-type="string"
                       calcext:value-type="string">
      <text:p>Abáigar</text:p>
     </table:table-cell>
     <table:table-cell table:style-name="ce2" office:value-type="string"
                       calcext:value-type="string">
      <text:p>1-001 -U</text:p>
     </table:table-cell>
     <table:table-cell table:style-name="ce5" office:value-type="float"
                       office:value="80" calcext:value-type="float">
      <text:p>80</text:p>
     </table:table-cell>
...
</table:table-row>
```

*.json


```
{
  "Codcir": "31",
  "Codmun": 1,
  "Municipio": "Abáigar",
  "Mesa": "1-001 -U",
  "Censo": 80,
  "Certif. Alta": 0,
  "Certif. Correc.": 0,
  "Censo Total": 80,
  "Votos Electores": 59,
  "Votos Interventores": 0,
  "Votos Totales": 59,
  "Votos Nulos": 0,
  "Votos Blancos": 0,
  "NA+": 18,
  "EH Bildu": 10,
  "I-E (n)": 0,
  "PSN-PSOE": 4,
  "GBAI": 19,
  "SAIN": 0,
  "VOX": 0,
  "EQUO": 0,
  "RCN-NOK": 1,
  "Ln": 0,
  "PODEMOS": 7
}
```

yaml

Name: Open City Model (OCM)

Description: |

  Open City Model is an initiative to provide cityGML data for all the buildings in the United States.

  By using other open datasets in conjunction with our own code and algorithms it is our goal to provide 3D geometries for every US building.

Documentation: https://github.com/opencitymodel/opencitymodel

Contact: https://github.com/opencitymodel/opencitymodel#contact

UpdateFrequency: Quarterly

Tags:

  - aws-pds

  - events

  - cities

  - geospatial

CSV

```
import csv
with open('some.csv', newline='') as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

CSV

```
import csv
with open('some.csv', newline='', encoding='utf-8') as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

CSV

```
import clevercsv
with open("imdb.csv", "r", newline="", encoding="utf-8") as fp:
    reader = clevercsv.reader(fp, delimiter=",", quotechar="", escapechar="\\")
    rows = list(reader)
```

https://towardsdatascience.com/handling-messy-csv-files-2ef829aa441d

Using pandas

df = pd.read_csv(filename)

df.to_csv(filename)

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S

Using pandas

```
df = pd.read_excel(filename)


# !pip install xlwt
df.to_excel(filename)
```

json

- JavaScript Object Notation
- Standardized format for passing data as text.
- Looks strikingly similar to Python's syntax for dictionaries, lists, strings and number types!
- …BUT… JSON is just text!

Structured data

```
alberto = {
  ‘name’: "Alberto",
  ‘age’: 25,
  ‘city’: "Pamplona" }

juan = {
  ‘name’: "Ana",
  ‘age’: 28,
  ‘city’: "Madrid" }

contacts = [alberto, juan]
```

```
[
  {
    "age": 25,
    "city": "Pamplona",
    "name": "Alberto"
  },
  {
    "age": 28,
    "city": "Madrid",
    "name": "Ana"
  }
]
```

https://towardsdatascience.com/an-introduction-to-json-c9acb464f43e

json

```
import json
data_string = json.dumps(contacts)
print(data_string)
new_contacts = json.loads(data_string)
# pretty
print(json.dumps(contacts, indent=4, sort_keys=True))
```

json

```
with open("contacts.json", "w") as file:
    json.dump(contacts, file)

with open("contacts.json") as file:
    contacts = json.load(file)
```

Using pandas

```
df = pd.read_json(filename)
df.to_json(filename)
```

```
[{
    "age": 25,
    "city": "Pamplona",
    "name": "Paula"
  },
  {
    "age": 28,
    "city": "Madrid",
    "name": "Ana"
  }]
```

	name	age	city
0	Paula	25	Pamplona
1	Ana	28	Madrid

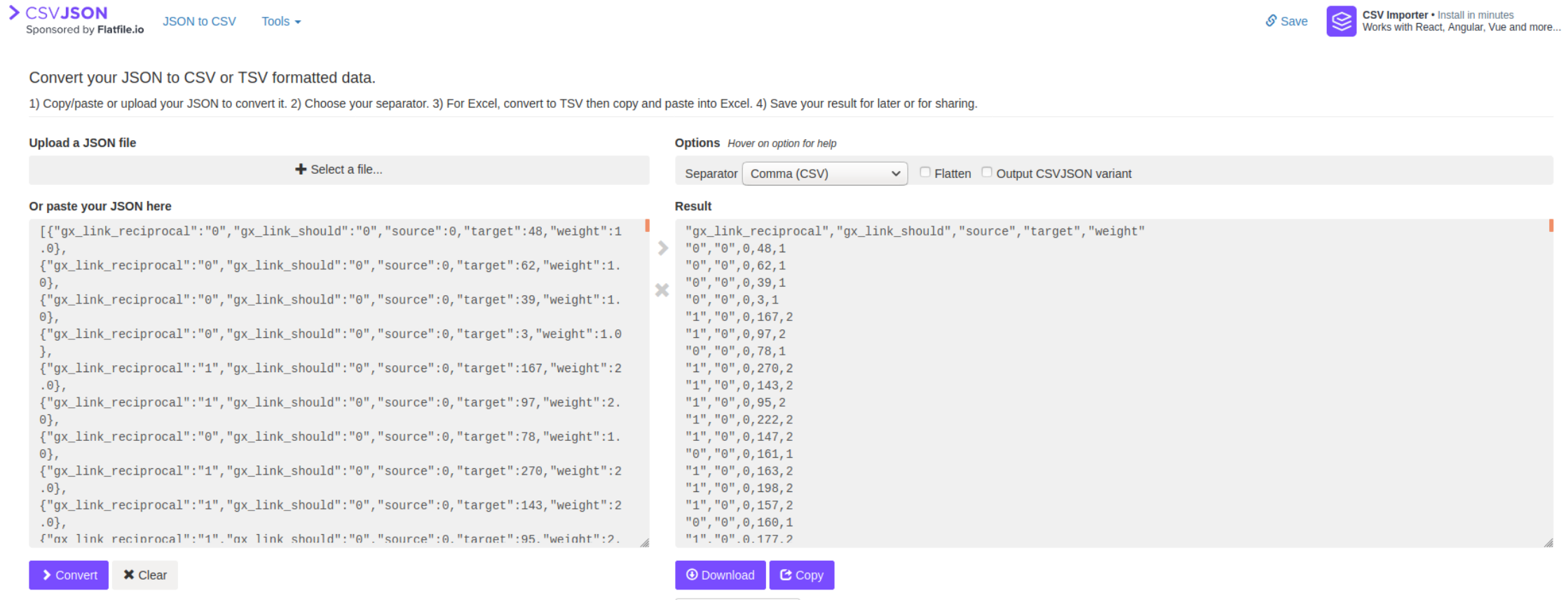
Using pandas

```
df = pd.read_json(filename)
df.to_json(filename)
```

```
[
  {
    "age": 25,
    "city": "Pamplona",
    "name": "Paula"
  },
  {
    "age": 28,
    "city": "Madrid",
    "name": "Ana"
  }
]
```

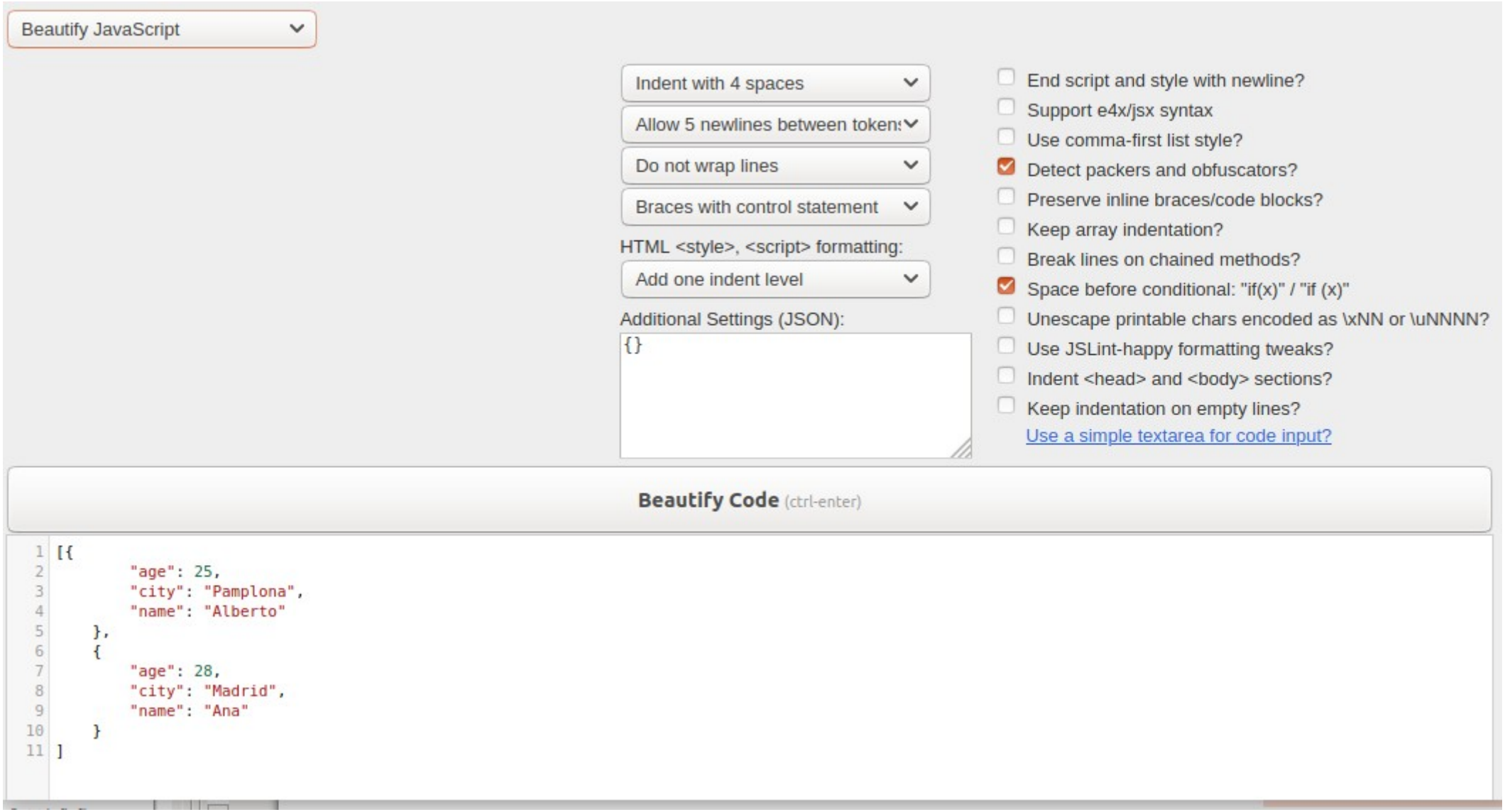
	name	age	city
0	Paula	25	Pamplona
1	Ana	28	Madrid

json2csv



https://www.csvjson.com/json2csv

beautifier



https://beautifier.io/

yaml

```
import yaml
with open('data/servers.yaml') as f:
    # The FullLoader parameter handles the conversion from YAML
    # scalar values to Python the dictionary format
    datos = yaml.load(f, Loader=yaml.FullLoader)
```

https://stackabuse.com/reading-and-writing-yaml-to-a-file-in-python/

xml

```
<catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>An in-depth look at creating applications
    with XML.</description>
  </book>
  …
</catalog>
```

https://towardsdatascience.com/parsing-xml-data-in-python-da26288280e1

xml

```
from xml.etree.ElementTree import parse
document = parse('data/books.xml')
for item in document.iterfind('book'):
    print(item.attrib[‘id’])
    print(item.findtext('title'))
```

Exercise

- Convert books.xml to books.json

```
[
  {
    "id": "bk101",
    "author": "Gambardella, Matthew",
    "title": "XML Developer's Guide",
    "genre": "Computer",
    "price": "44.95",
    "publish_date": "2000-10-01",
    "description": "An in-depth look at creating applications with XML."
  }, ...
]
```