

DR. ALVIN'S PUBLICATIONS

# DATA WRANGLING A POPULATION OF COUNTRIES DATASET

---

WITH PYTHON  
BY DR. ALVIN ANG



---

1 | PAGE

COPYRIGHTED BY DR ALVIN ANG  
WWW.ALVINANG.SG

# CONTENTS

<b>I. Step 1: Import the Dataset .....</b>	<b>3</b>
<b>A. Take a peek at the Dataset .....</b>	<b>4</b>
<b>II. Step 2: Preview the Dataset .....</b>	<b>5</b>
<b>III. Step 3: Checking for Missing Data .....</b>	<b>6</b>
<b>IV. Step 4: Dealing with missing data .....</b>	<b>7</b>
<b>A. Drop NaNs .....</b>	<b>7</b>
1. Drop Off Columns That Have NaNs .....	7
2. Drop Off Rows that have NaNs .....	8
<b>B. Fill Up NaNs with Other Values.....</b>	<b>8</b>
1. Fill Up NaNs with 0.....	8
2. Forward Fill .....	9
3. Backward Fill .....	9
<b>V. Step 5: Renaming Columns .....</b>	<b>10</b>
<b>A. Using .Rename.....</b>	<b>10</b>
<b>VI. Step 6: Filtering a Row .....</b>	<b>11</b>
<b>VII. Step 7: Filtering a Column .....</b>	<b>12</b>
<b>VIII. Step 8: Searching using Regex.....</b>	<b>13</b>
<b>IX. Step 9: Searching using Like .....</b>	<b>15</b>
<b>About Dr. Alvin Ang .....</b>	<b>16</b>

---

## I. STEP 1: IMPORT THE DATASET

---

- You may find the file here: <https://www.alvinang.sg/s/Population-of-Countries-in-2000.csv>
- [https://www.alvinang.sg/s/Data Wrangling a Population of Countries Dataset by Dr Alvin Ang.ipynb](https://www.alvinang.sg/s/Data%20Wrangling%20a%20Population%20of%20Countries%20Dataset%20by%20Dr%20Alvin%20Ang.ipynb)

## A. TAKE A PEEK AT THE DATASET

	A	B	C	D	E	F	G	H	I
1		country	country_isocode	year	pop	xrat	tcgdp	cc	cg
2	0	Argentina	ARG	2000	37335.653	0.9995	295072.21869	75.716805379	5.5788042896
3	1	Australia	AUS	2000	19053.186	1.72483	541804.6521	67.759025993	6.7200975332
4	2	India	IND	2000	1006300.297	44.9416	1728144.3748	64.575551328	14.072205773
5	3	Israel	ISR	2000	6114.57	4.07733	129253.89423	64.436450847	10.266688415
6	4	Malawi	MWI	2000	11801.505	59.543808333	5026.2217836	74.707624181	11.658954494
7	5	South Africa	ZAF	2000	45064.098	6.93983	227242.36949	72.718710427	5.7265463933
8	6	United States	USA	2000	282171.957	1	9898700	72.347054303	6.0324539789
9	7	Uruguay	URY	2000	3219.793	12.099591667	25255.961693	78.978740282	5.108067988

- Legend:
  - country: country name
  - country iso code: Country code
  - year: 2000
  - pop: population in thousands
  - xrat: exchange rate to US dollar (national currency units per US dollar)
  - tcgdp: total PPP converted GDP, Geary-Khamis method, at current prices (in million International dollar)
  - cc: consumption share of PPP converted GDP per capita at current prices
  - cg: government consumption share of PPP converted GDP per capita at current prices

```

Step 1: Import the Dataset

import pandas as pd

df = pd.read_csv('https://www.alvinang.sg/s/Population-of-Countries-in-2000.csv')

df.sample()

```


Unnamed: 0	country	country_isocode	year	pop	xrat	tcgdp	cc	cg	
5	5	South Africa	ZAF	2000	45064.098	6.93983	227242.36949	72.71871	5.726546


---

## II. STEP 2: PREVIEW THE DATASET

---

### Step 2: Preview the Dataset


0s ✓  `df.shape`

 `(8, 9)`

0s ✓ [6] `df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Unnamed: 0            8 non-null     int64
1   country               8 non-null     object
2   country_isocode      8 non-null     object
3   year                  8 non-null     int64
4   pop                   8 non-null     float64
5   xrat                  8 non-null     float64
6   tcgdp                 8 non-null     float64
7   cc                    8 non-null     float64
8   cg                    8 non-null     float64
dtypes: float64(5), int64(2), object(2)
memory usage: 704.0+ bytes
```

 `df.describe()`



	Unnamed: 0	year	pop	xrat	tcgdp	cc	cg
count	8.00000	8.0	8.000000e+00	8.000000	8.000000e+00	8.000000	8.000000
mean	3.50000	2000.0	1.763826e+05	16.415811	1.606312e+06	71.404995	8.145477
std	2.44949	0.0	3.479223e+05	22.758175	3.397025e+06	5.318015	3.383397
min	0.00000	2000.0	3.219793e+03	0.999500	5.026222e+03	64.436451	5.108068
25%	1.75000	2000.0	1.037977e+04	1.543623	1.032544e+05	66.963157	5.689611
50%	3.50000	2000.0	2.819442e+04	5.508580	2.611573e+05	72.532882	6.376276
75%	5.25000	2000.0	1.043411e+05	20.310094	8.383896e+05	74.959919	10.614755
max	7.00000	2000.0	1.006300e+06	59.543808	9.898700e+06	78.978740	14.072206

## Step 3: Check for Missing Data

▶ `df.isna().any()`

`#there's no missing data!`

```
↳ Unnamed: 0      False
   country         False
   country_isocode False
   year            False
   pop             False
   xrat            False
   tcgdp           False
   cc              False
   cg              False
   dtype: bool
```

▶ `df.isnull().sum()`

`#there's no NaNs!`

```
↳ Unnamed: 0      0
   country         0
   country_isocode 0
   year            0
   pop             0
   xrat            0
   tcgdp           0
   cc              0
   cg              0
   dtype: int64
```

## Step 4: Deal with Missing Data

### 4a) Drop NaNs

#### 4a)(i) Drop off Columns that have NaNs

#### A. DROP NANS

##### 1. DROP OFF COLUMNS THAT HAVE NANS

```
df.dropna(axis = 'columns')
```

```
#drop off columns that have NaNs
```

Unnamed: 0	country	country_isocode	year	pop	xrat	tcgdp	cc	cg	
0	0	Argentina	ARG	2000	37335.653	0.999500	2.950722e+05	75.716805	5.578804
1	1	Australia	AUS	2000	19053.186	1.724830	5.418047e+05	67.759026	6.720098
2	2	India	IND	2000	1006300.297	44.941600	1.728144e+06	64.575551	14.072206
3	3	Israel	ISR	2000	6114.570	4.077330	1.292539e+05	64.436451	10.266688
4	4	Malawi	MWI	2000	11801.505	59.543808	5.026222e+03	74.707624	11.658954
5	5	South Africa	ZAF	2000	45064.098	6.939830	2.272424e+05	72.718710	5.726546
6	6	United States	USA	2000	282171.957	1.000000	9.898700e+06	72.347054	6.032454
7	7	Uruguay	URY	2000	3219.793	12.099592	2.525596e+04	78.978740	5.108068

## 2. DROP OFF ROWS THAT HAVE NANS

### 4a)(ii) Drop off Rows that have NaNs

```
[ ] df.dropna(axis = 'rows')
```

```
#drop off rows that have NaNs
```

Unnamed: 0	country	country_isocode	year	pop	xrat	tcgdp	cc	cg	
0	0	Argentina	ARG	2000	37335.653	0.999500	2.950722e+05	75.716805	5.578804
1	1	Australia	AUS	2000	19053.186	1.724830	5.418047e+05	67.759026	6.720098
2	2	India	IND	2000	1006300.297	44.941600	1.728144e+06	64.575551	14.072206
3	3	Israel	ISR	2000	6114.570	4.077330	1.292539e+05	64.436451	10.266688
4	4	Malawi	MWI	2000	11801.505	59.543808	5.026222e+03	74.707624	11.658954
5	5	South Africa	ZAF	2000	45064.098	6.939830	2.272424e+05	72.718710	5.726546
6	6	United States	USA	2000	282171.957	1.000000	9.898700e+06	72.347054	6.032454
7	7	Uruguay	URY	2000	3219.793	12.099592	2.525596e+04	78.978740	5.108068

## B. FILL UP NANS WITH OTHER VALUES

### 1. FILL UP NANS WITH 0

#### 4b) Fill Up NaNs with Other Values

##### 4b)(i) Fill Up NaNs with 0

```
▶ df.fillna(0)
```

```
#fill up all NaNs with 0
```

Unnamed: 0	country	country_isocode	year	pop	xrat	tcgdp	cc	cg	
0	0	Argentina	ARG	2000	37335.653	0.999500	2.950722e+05	75.716805	5.578804
1	1	Australia	AUS	2000	19053.186	1.724830	5.418047e+05	67.759026	6.720098
2	2	India	IND	2000	1006300.297	44.941600	1.728144e+06	64.575551	14.072206
3	3	Israel	ISR	2000	6114.570	4.077330	1.292539e+05	64.436451	10.266688
4	4	Malawi	MWI	2000	11801.505	59.543808	5.026222e+03	74.707624	11.658954
5	5	South Africa	ZAF	2000	45064.098	6.939830	2.272424e+05	72.718710	5.726546
6	6	United States	USA	2000	282171.957	1.000000	9.898700e+06	72.347054	6.032454
7	7	Uruguay	URY	2000	3219.793	12.099592	2.525596e+04	78.978740	5.108068



## 2. FORWARD FILL

### 4b)(ii) Forward Fill

```
df.fillna(method = 'ffill')
```

	Unnamed: 0	country	country_isocode	year	pop	xrat	tcgdp	cc	cg
0	0	Argentina	ARG	2000	37335.653	0.999500	2.950722e+05	75.716805	5.578804
1	1	Australia	AUS	2000	19053.186	1.724830	5.418047e+05	67.759026	6.720098
2	2	India	IND	2000	1006300.297	44.941600	1.728144e+06	64.575551	14.072206
3	3	Israel	ISR	2000	6114.570	4.077330	1.292539e+05	64.436451	10.266688
4	4	Malawi	MWI	2000	11801.505	59.543808	5.026222e+03	74.707624	11.658954
5	5	South Africa	ZAF	2000	45064.098	6.939830	2.272424e+05	72.718710	5.726546
6	6	United States	USA	2000	282171.957	1.000000	9.898700e+06	72.347054	6.032454
7	7	Uruguay	URY	2000	3219.793	12.099592	2.525596e+04	78.978740	5.108068

## 3. BACKWARD FILL

### 4b)(iii) Backward Fill

```
df.fillna(method = 'bfill')
```

	Unnamed: 0	country	country_isocode	year	pop	xrat	tcgdp	cc	cg
0	0	Argentina	ARG	2000	37335.653	0.999500	2.950722e+05	75.716805	5.578804
1	1	Australia	AUS	2000	19053.186	1.724830	5.418047e+05	67.759026	6.720098
2	2	India	IND	2000	1006300.297	44.941600	1.728144e+06	64.575551	14.072206
3	3	Israel	ISR	2000	6114.570	4.077330	1.292539e+05	64.436451	10.266688
4	4	Malawi	MWI	2000	11801.505	59.543808	5.026222e+03	74.707624	11.658954
5	5	South Africa	ZAF	2000	45064.098	6.939830	2.272424e+05	72.718710	5.726546
6	6	United States	USA	2000	282171.957	1.000000	9.898700e+06	72.347054	6.032454
7	7	Uruguay	URY	2000	3219.793	12.099592	2.525596e+04	78.978740	5.108068

---

## V. STEP 5: RENAMING COLUMNS

---

```
[6] df.columns
Index(['Unnamed: 0', 'country', 'country_isocode', 'year', 'pop', 'xrat',
      'tcgdp', 'cc', 'cg'],
      dtype='object')
```

```
df.columns = ['S/N', 'country', 'country_isocode', 'year', 'pop', 'xrat', 'tcgdp', 'cc', 'cg']
df
```

	S/N	country	country_isocode	year	pop	xrat	tcgdp	cc	cg
0	0	Argentina	ARG	2000	37335.653	0.999500	2.950722e+05	75.716805	5.578804
1	1	Australia	AUS	2000	19053.186	1.724830	5.418047e+05	67.759026	6.720098
2	2	India	IND	2000	1006300.297	44.941600	1.728144e+06	64.575551	14.072206
3	3	Israel	ISR	2000	6114.570	4.077330	1.292539e+05	64.436451	10.266688

### A. USING .RENAME

```
[12] df = df.rename(columns={"country_isocode": "blablabla"})
```

```
[13] df.columns
Index(['S/N', 'country', 'blablabla', 'year', 'pop', 'xrat', 'tcgdp', 'cc',
      'cg'],
      dtype='object')
```

## Step 6: Filtering a Row

```
[ ] #Filtering out United States
```

```
df[df['country'] == 'United States']
```

S/N	country	blablabla	year	pop	xrat	tcgdp	cc	cg	
6	6	United States	USA	2000	282171.957	1.0	9898700.0	72.347054	6.032454

```
▶ #Filtering out Australia
```

```
df[df['country'] == 'Australia']
```

```
↳
```

S/N	country	blablabla	year	pop	xrat	tcgdp	cc	cg	
1	1	Australia	AUS	2000	19053.186	1.72483	541804.6521	67.759026	6.720098

## Step 7: Filtering a Column

▶ #Filter out Country and Year

```
df.filter(items = ['country', 'year'])
```

↳

	country	year
0	Argentina	2000
1	Australia	2000
2	India	2000
3	Israel	2000
4	Malawi	2000
5	South Africa	2000
6	United States	2000
7	Uruguay	2000

▶ #OR

```
df[['year', 'country']]
```

↳

	year	country
0	2000	Argentina
1	2000	Australia
2	2000	India
3	2000	Israel
4	2000	Malawi
5	2000	South Africa
6	2000	United States
7	2000	Uruguay

## Step 8: Searching Out Using REGEX

```
[ ] #axis = 1 --> search via Columns  
    #axis = 0 --> search via Row (index column)
```

- regex = Regular Expression
- regex is a string of text that allows you to create patterns that help search and match text.
- the \$\$ means Matches end of line
- so r\$ means that 'r' has to match at the end of the line (in this case, year)
- if you put regex='y\$', it will search out 'country' because that's the only column name with ending 'y'

```
df.filter(regex = 'r$', axis = 1)  
#searching out column names ending with 'r'
```

	year
0	2000
1	2000
2	2000
3	2000
4	2000
5	2000
6	2000
7	2000

```
▶ df.filter(regex = 'y$', axis = 1)  
#searching out column names ending with 'y'
```

```
↳
```

	country
0	Argentina
1	Australia
2	India
3	Israel
4	Malawi
5	South Africa
6	United States
7	Uruguay

## Step 9: Searching Out Using LIKE

```
df.filter(like = 'p', axis = 1)
```

#like 'p' means to search for any column name that has 'p' in it

```
df
```

	pop	tcgdp
0	37335.653	2.950722e+05
1	19053.186	5.418047e+05
2	1006300.297	1.728144e+06
3	6114.570	1.292539e+05
4	11801.505	5.026222e+03
5	45064.098	2.272424e+05
6	282171.957	9.898700e+06
7	3219.793	2.525596e+04

---

THE END

---

---

## ABOUT DR. ALVIN ANG

---



Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at [www.AlvinAng.sg](http://www.AlvinAng.sg).