Basic statistics.

Let's load data from country.csv file in jupyter, we are going to obtain statistical information about the data damain. We will explain the process step by step using several Pharo frameworks.

First of all, be sure NeoCSV, DataFrame and Roassal are installed in your Image, next step, before loading country.csv file, examine this file opening in your fabourite text editor, check the character encoding, its character separator and decimal point character. As you will see, it's windows-1258 encoded, separated by ';' and it uses european floating point format, ',' is the decimal point.
We use Zinc Streams and NeoCSV to encode the content and load it into an Array.
Let's create a DataFrame, due to this Array has a first row containing the column headers, we need to create the DataFrame from rows starting from two up to the end. We will assing the columns names from the first row in the array.

In [1]:
```
"this file is windows-1258 encoded so we have to load into Pharo kern
el using the correct encoding"

stream := ZnCharacterReadStream
                on:'/Users/Cat/Dropbox/Master Ciencia de les dades/S1
.1.mineria de dades/PAC2/countries.csv'
                        asFileReference binaryReadStream
                encoding: #windows1258.


arrayOfRows := (NeoCSVReader on: stream)
                            separator: $;;
                            upToEnd.

paisos := DataFrame fromRows:(arrayOfRows copyFrom:2 to:arrayOfRows s
ize).
paisos columnNames: (arrayOfRows at:1)
```

Out[1]:

Let's show rows from one to thirty three including its heading. Note that encoding is correct

In [2]:
```
"JupyterTalk will transform those Strings to utf-8, look at:#32 Camer
oon | Yaoundé"
self display openInJupyter: (OrderedCollection new
                                    add:paisos columnNames;
                                    addAll: (paisos asArrayOfRows cop
yFrom:1 to:33);
                                    yourself) .
```

Out[2]:

NAME	CAPITAL	TOTAL_AREA_KM2	POPULATION	DENSITY_KM2	CONTI
Afghanistan	Kabul	647500	24326959	37,6	Asia
Albania	Tirane	28750	3453505	120,1	Europe

Algeria	Algiers	2381740	29181456	12,3	Africa
Andorra	Andorra la Vella	450	67569	150,2	Europe
Angola	Luanda	1246700	10339364	8,3	Africa
Antigua & Barbuda	Saint John's	440	65619	149,1	America
Argentina	Buenos Aires	2766890	34673391	12,5	America
Armenia	Yerevan	29800	3590722	120,5	Asia
Australia	Canberra	7686850	18562252	2,4	Oceania
Austria	Vienna	83850	8014617	95,6	Europe
Azerbaijan	Baku	86600	7892712	91,1	Asia
Bahamas; The	Nassau	13940	259413	18,6	America
Bahrain	Manama	620	590784	952,9	Asia
Bangladesh	Dhaka	144000	131066751	910,2	Asia
Barbados	Bridgetown	430	257010	597,7	America
Belarus	Minsk	207600	10468730	50,4	Europe
Belgium	Brussels	30510	10099019	331	Europe
Belize	Belmopan	22960	219241	9,5	America
Benin	Porto-Novo	112620	5706582	50,7	Africa
Bhutan	Thimphu	47000	1822305	38,8	Asia
Bolivia	Sucre / La Paz	1098580	8073920	7,3	America
Bosnia - Herzegovina	Sarajevo	51233	3222635	62,9	Europe
Botswana	Gaborone	600370	1425275	2,4	Africa
Brazil	Brasilia	8511965	162698486	19,1	America
Brunei	Bandar Seri Begawan	5770	299953	52	Asia
Bulgaria	Sofia	110910	8753260	78,9	Europe
Burkina	Ouagadougou	274200	10713625	39,1	Africa
Burma	Rangoon	678500	45933719	67,7	Asia
Burundi	Bujumbura	27830	6398950	229,9	Africa
Cambodia	Phnom Penh	181040	10860260	60	Asia
Cameroon	Yaoundé	475440	13915813	29,3	Africa

Canada	Ottawa	9976140	28744482	2,9	America
Cape Verde	Praia	4030	448975	111,4	Africa

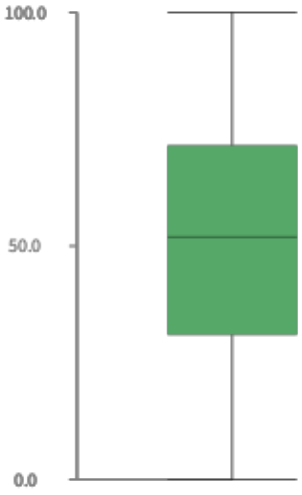
Now, let's get a basic boxplot from 'URBAN_POPULATION' column. We need to change the column type from String to Integer.
The BoxPlot is part of the Roassal Package. DataFrame has helper methods to draw basic statistical drawings using Roassal.

In [3]:
```
"CONVERT URBAN_POPULATION TO INTEGER"

newCol := (paisos column:#URBAN_POPULATION) collect:[:v| v ifNil:[0]
ifNotNil:[v asInteger]].
paisos column:#URBAN_POPULATION put:newCol.
```

Out[3]:

In [4]:
```
b := (paisos column:#URBAN_POPULATION) boxplot.
self display openInJupyter: b
```

Out[4]:



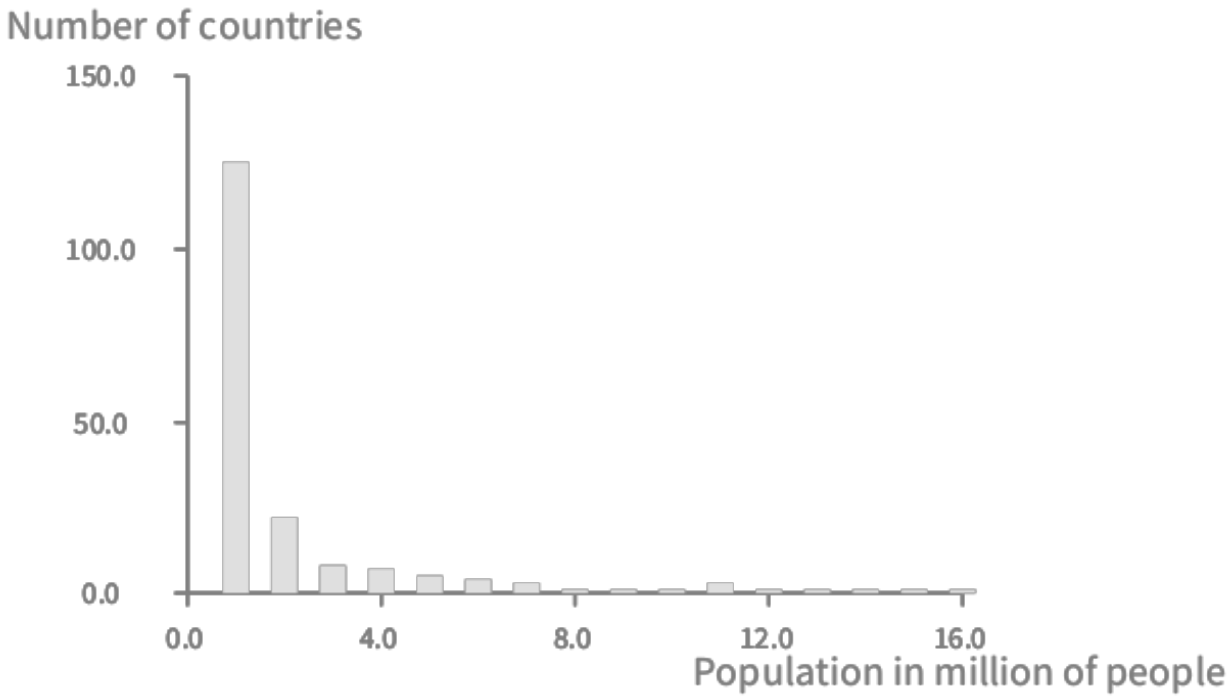
Now, let's create a histogram from a quantitative column. We will use POPULATION column, first we need to change column type to Integer, calculate maximun and minimun using DataSerie methods min and max. We will use a Bag in order to categorize the POPULATION column

In [5]:
```
"CONVERT POPULATION TO INTEGER"

newCol := (paisos column:#POPULATION) collect:[:v| v ifNil:[nil] ifNo
tNil:[v asInteger]].
paisos column:#POPULATION put:newCol.
maxPopulation := (paisos column:#POPULATION) max.
minPopulation := (paisos column:#POPULATION) min.
domainSize := maxPopulation - minPopulation.
bag := Bag new.
newCol do:[:each| bag add:((((each - minPopulation)/ domainSize*100)
asInteger) ) ]
```

Out[5]:

In [14]:
```
b:= RTGrapher new.
ds := RTData new.
ds barShape width: 10.
ds points: bag valuesAndCounts.
b add: ds.

uuid :=self display openInJupyter: b  extent:600@650
```

Out[14]:



We can modify an object shown in other response. #openInJupyter returns an uuid for each displayed object. we can use this uuid to refresh the picture after we change any of its properties.

In [15]:
```
"we can modify an object shown in other response."
b axisX title:'Population in million of people'.
b axisY title:'Number of countries'.
self display refresh: uuid
```

Out[15]:

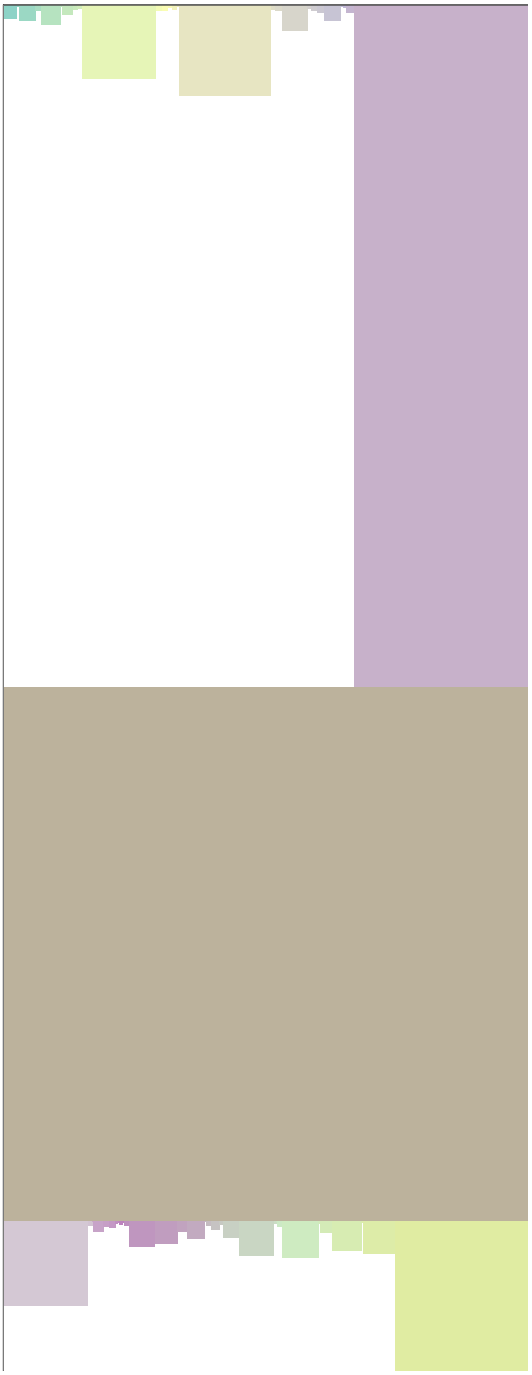
Let's visualize a nice drawing using Roassal. We can interact with our drawing.
We need to include the Roassal javascript libraey in our jupyter markdown document.

In [8]:
```
self loadScript: IPRoassal js
```

Out[8]:

In [17]:
```
| view coll col2 n b |
view := RTView new.
coll := (paisos column:#POPULATION).
n := RTMultiLinearColorForIdentity new objects: coll.
coll
    doWithIndex: [ :r :index |
        view
            add:
            ((RTBox new
                color: [ :value | n rtValue: r ];
                size: r/ 1000)
            elementOn: index) ].
col2 := (paisos column:#NAME).
RTFlowLayout new applyOn: view elements.
view elements do: [ :e | e @ (RTPopup text: [ :el | col2 at:el ]) ].
view @ RTDraggableView.
b := RTAxisAdaptedBuilder new.
b view: view.
b margin: 20.
b objects: view elements.
b build.

self display
        interactionOn;
        openInJupyter: b extent:650@640
```

Out[17]:



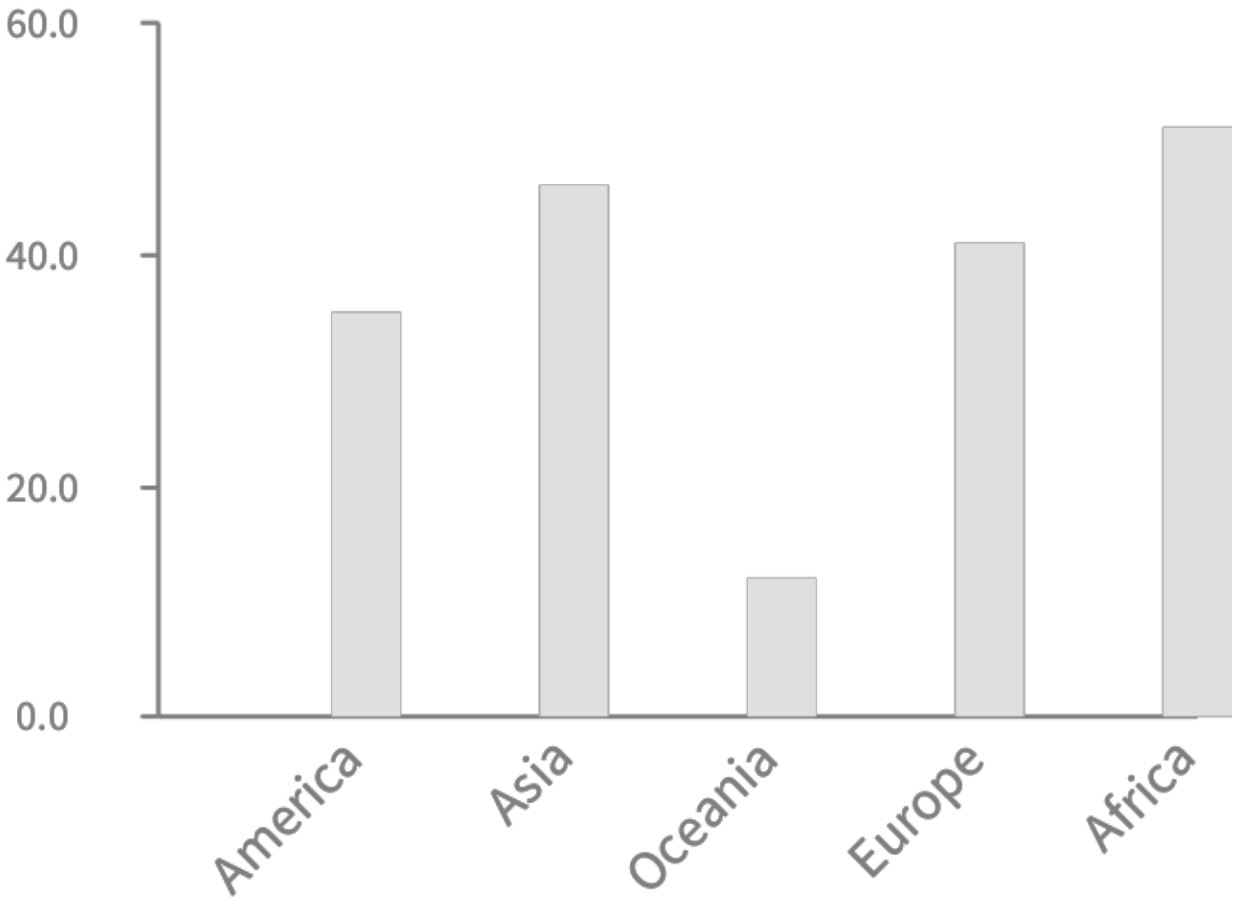
Now, let's get a barchar showing the number of countries in each continent. We use a Bag to get a count for each continent.

In [43]:
```
bag := Bag new.
(paisos column:#CONTINENT) do:[:each| bag add:each ]
```

Out[43]:

In [44]:
```
b:= RTGrapher new.
ds := RTData new.
ds barShape width: 20.
ds interaction highlight.
ds points: bag valuesAndCounts associations.
ds y: [:each| each value].
ds barChartWithBarTitle: [:each|each key].
b add: ds.
b axisX
        noTick;
        noLabel.

self display openInJupyter: b extent:640@600
```

Out[44]:



We will finish this tutorial getting basic statistcs from the column population.

In [46]:
```
self display openInJupyter:(paisos column:#POPULATION) summary asStri
ngTable
```

Out[46]:

	POPULATION
Min	16954
1st Qu.	1952975
Median	5989065
Mean	3.122843657e7
3rd Qu.	20013660
Max	1215609480