15. CONFIANZA DEL PÚBLICO EN PERSONAJES E INSTITUCIONES

Para explicarle a usted los impactos o efectos de la ciencia y la tecnología, de la siguiente lista, diga en quién confía más por sus conocimientos; considere la escala: muy confiable, confiable, poco confiable y nada confiable.

ANOTE EL CÓDIGO QUE CORRESPONDA
MUY CONFIABLE = 1, CONFIABLE = 2, POCO CONFIABLE = 3,
NADA CONFIABLE=4, NO SABE =5

1 **Periodistas de revistas y periódicos** ..... ⌊__⌋
2 **Periodistas de televisión y radio** ..... ⌊__⌋
3 **Políticos** ..... ⌊__⌋
4 **Procuraduría Federal del Consumidor** (revista del consumidor) ..... ⌊__⌋
5 **Científicos trabajando en universidades o centros de investigación públicos** ..... ⌊__⌋
6 **Científicos trabajando en centros de investigación privados** ..... ⌊__⌋
7 **Asociaciones de protección ambiental** ..... ⌊__⌋
8 **Empresarios** ..... ⌊__⌋
9 **Militares** ..... ⌊__⌋
10 **Líderes o representantes religiosos** ..... ⌊__⌋
11 **Gobierno** ..... ⌊__⌋
12 **Médicos** ..... ⌊__⌋
13 **Escritores o intelectuales** ..... ⌊__⌋
14 **Consejo Nacional de Ciencia y Tecnología** ..... ⌊__⌋
15 **Profesores** (primaria, secundaria, etc.) ..... ⌊__⌋

200	S4P15_1	Periodistas de revistas y periódicos	Numérico	1	1-5	Periodistas de revistas y periódicos
					1	Muy confiable
					2	Confiable

(Continua)

Encuesta sobre la Percepción Pública de la Ciencia y la Tecnología en México 2017

Estructura de la tabla de datos del **CUESTIONARIO BÁSICO**

Nombre de la tabla de datos: **ENPECYT2017_CUESTIONARIO_BÁSICO.DBF**

Núm. Campo	Campo (Mnemónico)	Nombre del campo	Tipo	Longitud	Códigos válidos	Descripción del código válido
					3	Poco confiable
					4	Nada confiable
					5	No sabe
201	S4P15_2	Periodistas de televisión y radio	Numérico	1	1-5	Periodistas de televisión y radio
					1	Muy confiable
					2	Confiable
					3	Poco confiable
					4	Nada confiable

. Estructura de la base de datos. 2018.

In [12]:
```
import pandas as pd
import simpledbf
import matplotlib.pyplot as plt
```

In [3]:
```
dbf = simpledbf.Dbf5("enpecyt2017_cb1.dbf",codec="cp1252")
df = dbf.to_dataframe()
```

In [4]:
```
df[:5]
```

Out[4]:

	CD_A	PER	ENT	CON	V_SEL	N_HOG	N_REN	S3P1	S3P1_2	S3P2	...	S4P17_6	S4P17_7	S4P17_8	S4P17_9	S4P17_10	S4P17_11	S4P
0	14	1117	01	40018	3	1	01	3	3	NaN	...	2	3	3	2	2	2	
1	14	1117	01	40018	4	1	01	2	6	NaN	...	3	3	3	3	3	3	
2	14	1117	01	40018	2	1	01	2	6	NaN	...	3	3	3	3	3	3	
3	14	1117	01	40018	1	1	03	4	3	NaN	...	2	2	3	3	3	2	
4	14	1117	01	40018	5	1	02	3	3	NaN	...	3	3	3	3	3	3	

5 rows × 236 columns

In [5]:
```
df.keys()
```

Out[5]:
```
Index(['CD_A', 'PER', 'ENT', 'CON', 'V_SEL', 'N_HOG', 'N_REN', 'S3P1',
       'S3P1_2', 'S3P2',
       ...
       'S4P17_6', 'S4P17_7', 'S4P17_8', 'S4P17_9', 'S4P17_10', 'S4P17_11',
       'S4P17_12', 'S4P17_13', 'S4P17_14', 'FAC'],
      dtype='object', length=236)
```

In [10]:
```
prelista = pd.Series(df.keys())[199:214]
prelista
```

Out[10]:
```
199     S4P15_1
200     S4P15_2
201     S4P15_3
202     S4P15_4
203     S4P15_5
204     S4P15_6
205     S4P15_7
206     S4P15_8
207     S4P15_9
208    S4P15_10
209    S4P15_11
210    S4P15_12
211    S4P15_13
212    S4P15_14
213    S4P15_15
dtype: object
```

In [23]:
```
lista=list(prelista)
lista
```

Out[23]:
```
['S4P15_1',
 'S4P15_2',
 'S4P15_3',
 'S4P15_4',
 'S4P15_5',
 'S4P15_6',
 'S4P15_7',
 'S4P15_8',
 'S4P15_9',
 'S4P15_10',
 'S4P15_11',
 'S4P15_12',
 'S4P15_13',
 'S4P15_14',
 'S4P15_15']
```

In [24]:
```
lista.append("FAC")
lista
```

Out[24]:
```
['S4P15_1',
 'S4P15_2',
 'S4P15_3',
 'S4P15_4',
 'S4P15_5',
 'S4P15_6',
 'S4P15_7',
 'S4P15_8',
 'S4P15_9',
 'S4P15_10',
 'S4P15_11',
 'S4P15_12',
 'S4P15_13',
 'S4P15_14',
 'S4P15_15',
 'FAC']
```

In [27]:
```
nbase = df.loc[:,lista]
nbase[:5]
```

Out[27]:

	S4P15_1	S4P15_2	S4P15_3	S4P15_4	S4P15_5	S4P15_6	S4P15_7	S4P15_8	S4P15_9	S4P15_10	S4P15_11	S4P15_12	S4P15_13	S4P15_1
0	4	3	4	2	2	4	2	4	4	4	4	4	3	
1	2	2	4	4	2	5	4	2	2	5	4	2	2	
2	2	2	4	2	1	1	2	3	4	4	4	1	2	
3	4	4	4	2	2	2	4	2	4	4	4	2	2	
4	3	3	4	3	2	1	2	3	1	3	3	2	2	

In [29]:
```
per_imp=nbase.groupby("S4P15_1")
per_imp.groups
```

Out[29]:
```
{1: Int64Index([  16,   32,   42,  106,  126,  134,  156,  164,  222,  273,  295,
             380,  401,  428,  563,  589,  612,  643,  684,  692,  720,  735,
             746,  754,  755,  766,  809,  821,  942,  944, 1006, 1010, 1034,
            1141, 1147, 1184, 1274, 1315, 1524, 1527, 1589, 1597, 1703, 1743,
            1798, 1886, 1971, 2042, 2050, 2073, 2080, 2101, 2196, 2209, 2227,
            2230, 2246, 2267, 2294, 2297, 2370, 2428, 2430, 2452, 2467, 2483,
            2523, 2536, 2538, 2541, 2552, 2617, 2675, 2722, 2724],
           dtype='int64'),
 2: Int64Index([   1,    2,    5,    7,    8,    9,   12,   13,   17,   18,
             ...
            2824, 2827, 2830, 2835, 2837, 2840, 2845, 2846, 2849, 2852],
           dtype='int64', length=1032),
 3: Int64Index([   4,    6,   11,   15,   25,   26,   27,   29,   33,   36,
             ...
            2832, 2833, 2836, 2838, 2839, 2841, 2843, 2844, 2848, 2851],
           dtype='int64', length=1407),
 4: Int64Index([   0,    3,   10,   14,   28,   31,   38,   54,   62,   65,
             ...
            2784, 2794, 2796, 2805, 2815, 2818, 2825, 2842, 2847, 2850],
           dtype='int64', length=283),
 5: Int64Index([  60,   87,  135,  166,  253,  264,  287,  515,  546,  605,  607,
             613,  615,  691,  737,  751,  836,  839,  841,  843,  844,  882,
             889,  978,  986, 1151, 1208, 1457, 1637, 1664, 1725, 1731, 1747,
            1780, 1787, 1814, 1823, 1861, 1883, 1893, 1922, 2028, 2070, 2072,
            2112, 2355, 2516, 2529, 2630, 2655, 2677, 2683, 2705, 2721, 2729,
            2834],
           dtype='int64')}
```

In [31]:
```
per_imp["FAC"].sum()
```

Out[31]:
```
S4P15_1
1     1429324
2    15968722
3    16475788
4     2462385
5      721657
Name: FAC, dtype: int64
```

In [32]:
```
nbase["FAC"].sum()
```

Out[32]:
```
37057876
```

In [34]:
```
per_imp["FAC"].sum()/nbase["FAC"].sum()
```

Out[34]:
```
S4P15_1
1    0.038570
2    0.430913
3    0.444596
4    0.066447
5    0.019474
Name: FAC, dtype: float64
```

In [35]:
```
per_imp["FAC"].sum()/nbase["FAC"].sum()*100
```

Out[35]:
```
S4P15_1
1     3.857005
2    43.091304
3    44.459612
4     6.644701
5     1.947378
Name: FAC, dtype: float64
```

In [37]:
```
pre_imp_res = per_imp["FAC"].sum()/nbase["FAC"].sum()*100
pre_imp_res
```

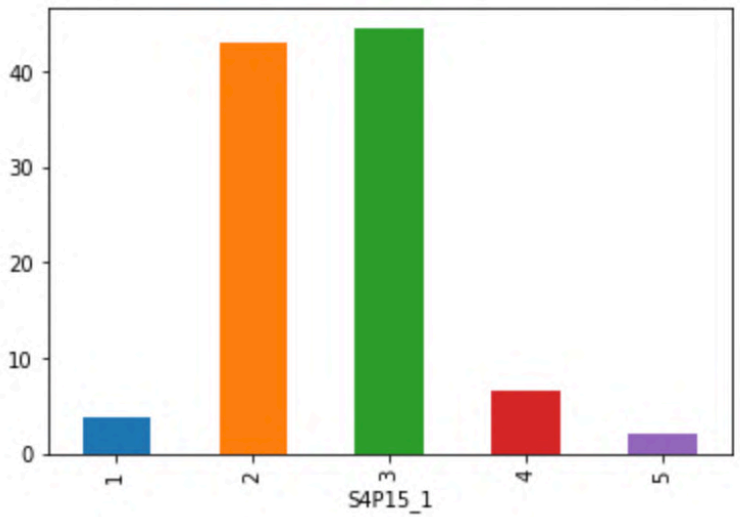
Out[37]:
```
S4P15_1
1     3.857005
2    43.091304
3    44.459612
4     6.644701
5     1.947378
Name: FAC, dtype: float64
```

In [39]:
```
pre_imp_res.name = "pre_imp_por"
pre_imp_res
```

Out[39]:
```
S4P15_1
1     3.857005
2    43.091304
3    44.459612
4     6.644701
5     1.947378
Name: pre_imp_por, dtype: float64
```

In [40]:
```
pre_imp_res.plot(kind="bar")
```

Out[40]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11200fc88>
```



In [49]:
```
opciones = ["Muy confiable","Confiable","Poco confiable","Nada confiable","No sabe"]
```

In [42]:
```
pre_imp_res.set_index(opciones)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-42-8ab6ba32baea> in <module>()
----> 1 pre_imp_res.set_index(opciones)

/anaconda3/lib/python3.6/site-packages/pandas/core/generic.py in __getattr__(self, name)
   4370             if self._info_axis._can_hold_identifiers_and_holds_name(name):
   4371                 return self[name]
-> 4372             return object.__getattribute__(self, name)
   4373
   4374     def __setattr__(self, name, value):

AttributeError: 'Series' object has no attribute 'set_index'
```

In [45]:
```
nframe=pre_imp_res.to_frame()
nframe
```

Out[45]:

S4P15_1	pre_imp_por
1	3.857005
2	43.091304
3	44.459612
4	6.644701
5	1.947378

In [45]:
```
nframe=pre_imp_res.to_frame()
nframe
```

Out[45]:

S4P15_1	pre_imp_por
1	3.857005
2	43.091304
3	44.459612
4	6.644701
5	1.947378

In [52]:
```
nuframe=nframe.set_index(pd.Series(opciones))
```

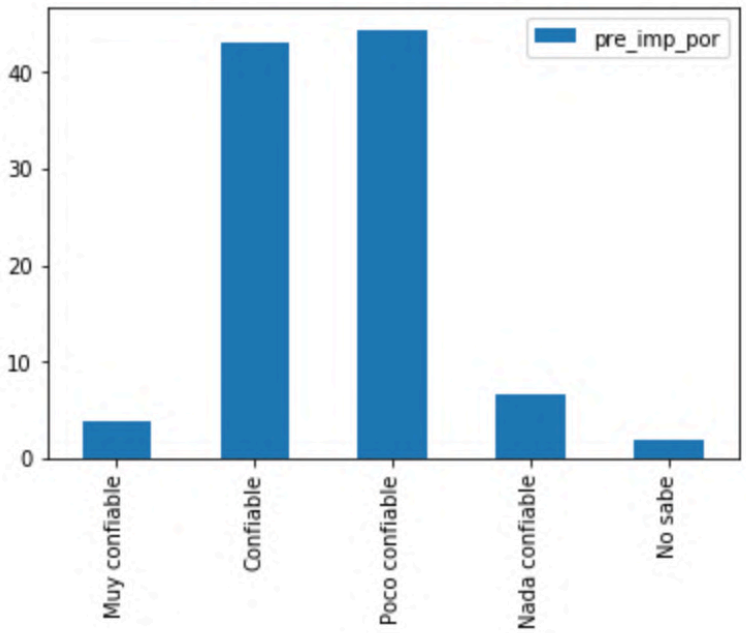
In [53]:
```
nuframe
```

Out[53]:

	pre_imp_por
Muy confiable	3.857005
Confiable	43.091304
Poco confiable	44.459612
Nada confiable	6.644701
No sabe	1.947378

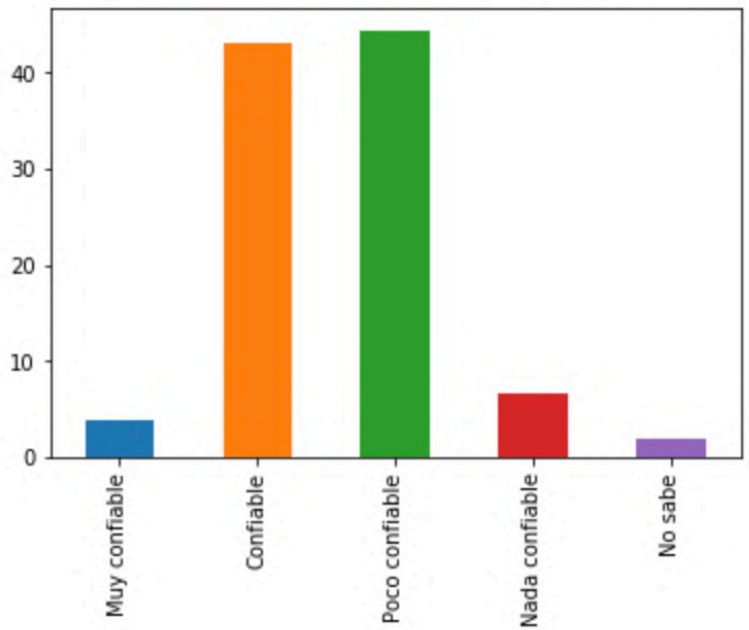
In [54]:
```
nuframe.plot(kind="bar")
```

Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x115ab3d30>



In [55]:
```
nuframe["pre_imp_por"].plot(kind="bar")
```

Out[55]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x115e3dac8>
```



15. CONFIANZA DEL PÚBLICO EN PERSONAJES E INSTITUCIONES

Para explicarle a usted los impactos o efectos de la ciencia y la tecnología, de la siguiente lista, diga en quién confía más por sus conocimientos; considere la escala: muy confiable, confiable, poco confiable y nada confiable.

ANOTE EL CÓDIGO QUE CORRESPONDA
MUY CONFIABLE = 1, CONFIABLE = 2, POCO CONFIABLE = 3,
NADA CONFIABLE=4, NO SABE =5

1 **Periodistas de revistas y periódicos** ..... ⌊__⌋
2 **Periodistas de televisión y radio** ..... ⌊__⌋
3 **Políticos** ..... ⌊__⌋
4 **Procuraduría Federal del Consumidor** (revista del consumidor) ..... ⌊__⌋
5 **Científicos trabajando en universidades o centros de investigación públicos** ..... ⌊__⌋
6 **Científicos trabajando en centros de investigación privados** ..... ⌊__⌋
7 **Asociaciones de protección ambiental** ..... ⌊__⌋
8 **Empresarios** ..... ⌊__⌋
9 **Militares** ..... ⌊__⌋
10 **Líderes o representantes religiosos** ..... ⌊__⌋
11 **Gobierno** ..... ⌊__⌋
12 **Médicos** ..... ⌊__⌋
13 **Escritores o intelectuales** ..... ⌊__⌋
14 **Consejo Nacional de Ciencia y Tecnología** ..... ⌊__⌋
15 **Profesores** (primaria, secundaria, etc.) ..... ⌊__⌋

In [62]:
```
g1=nbase.groupby("S4P15_5")
g2=nbase.groupby("S4P15_6")
g3=nbase.groupby("S4P15_10")
g4=nbase.groupby("S4P15_12")
```

In [72]:
```
p1=g1["FAC"].sum()/nbase["FAC"].sum()*100
p2=g2["FAC"].sum()/nbase["FAC"].sum()*100
p3=g3["FAC"].sum()/nbase["FAC"].sum()*100
p4=g4["FAC"].sum()/nbase["FAC"].sum()*100
```

In [73]:
```
forma1 = pd.DataFrame([p1,p2,p3,p4])
forma1
```

Out[73]:

	1	2	3	4	5
FAC	29.570499	54.027473	11.228825	1.613482	3.559721
FAC	28.365080	55.990246	10.455184	1.554636	3.634855
FAC	4.728196	27.904767	45.637429	19.120880	2.608727
FAC	23.567627	63.160841	10.932853	1.747550	0.591129

In [74]:
```
p1.name="p1"
p2.name="p2"
p3.name="p3"
p4.name="p4"
```

In [70]:
```
forma1 = pd.DataFrame([p1,p2,p3,p4])
forma1
```

Out[70]:

	1	2	3	4	5
p1	29.570499	54.027473	11.228825	1.613482	3.559721
p2	28.365080	55.990246	10.455184	1.554636	3.634855
p3	4.728196	27.904767	45.637429	19.120880	2.608727
p4	23.567627	63.160841	10.932853	1.747550	0.591129

In [79]:
```
forma1.columns=opciones
```

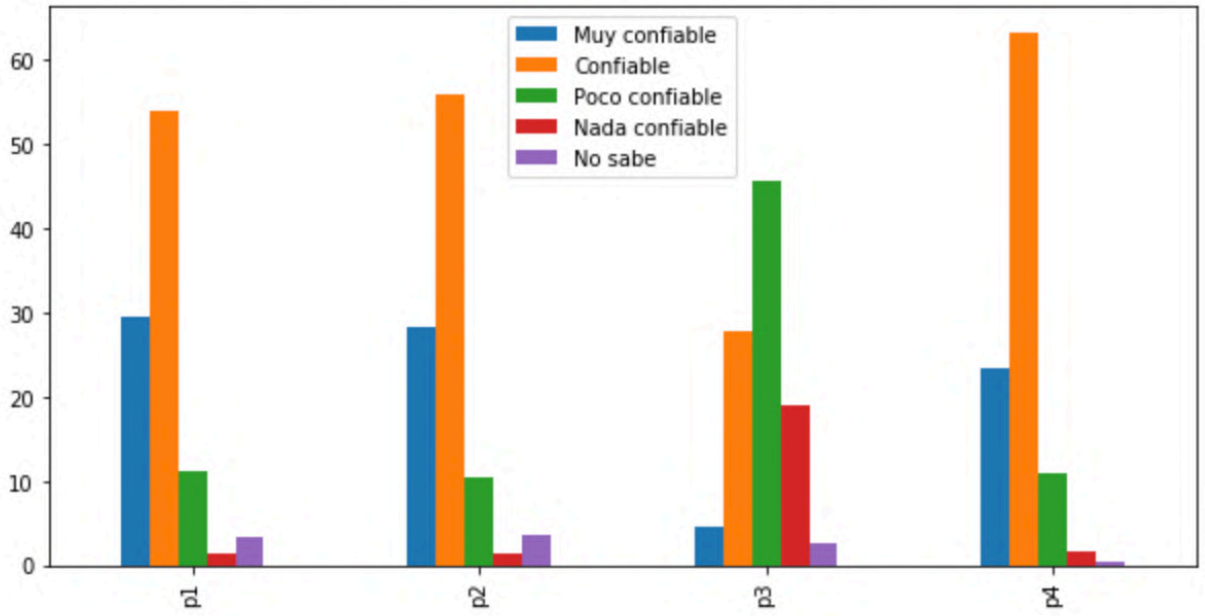
In [80]:
```
forma1
```

Out[80]:

	Muy confiable	Confiable	Poco confiable	Nada confiable	No sabe
p1	29.570499	54.027473	11.228825	1.613482	3.559721
p2	28.365080	55.990246	10.455184	1.554636	3.634855
p3	4.728196	27.904767	45.637429	19.120880	2.608727
p4	23.567627	63.160841	10.932853	1.747550	0.591129

In [84]:
```
forma1.plot(kind="bar",figsize=(10,5))
```

Out[84]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x115e63f28>
```



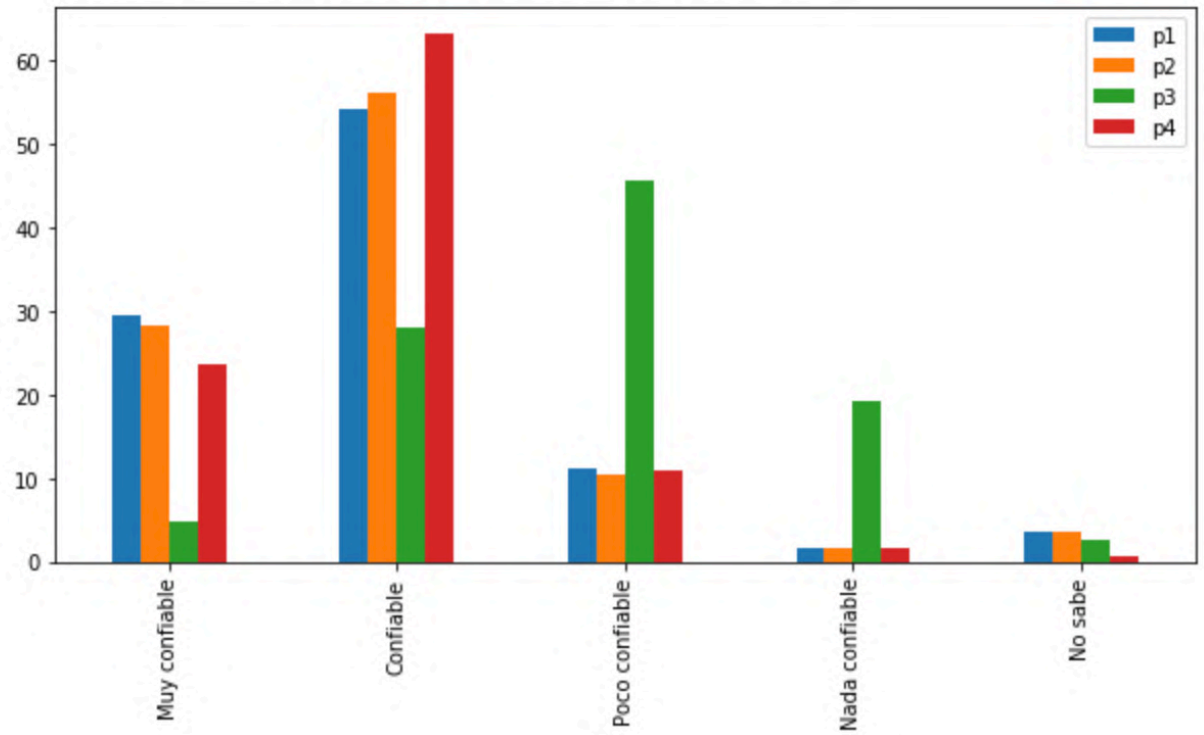
In [86]:
```
forma2=forma1.transpose()
forma2
```

Out[86]:

	p1	p2	p3	p4
Muy confiable	29.570499	28.365080	4.728196	23.567627
Confiable	54.027473	55.990246	27.904767	63.160841
Poco confiable	11.228825	10.455184	45.637429	10.932853
Nada confiable	1.613482	1.554636	19.120880	1.747550
No sabe	3.559721	3.634855	2.608727	0.591129

In [87]:
```
forma2.plot(kind="bar",figsize=(10,5))
```

Out[87]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11661b4e0>
```



In [88]:
```
nbase[:5]
```

Out[88]:

	S4P15_1	S4P15_2	S4P15_3	S4P15_4	S4P15_5	S4P15_6	S4P15_7	S4P15_8	S4P15_9	S4P15_10	S4P15_11	S4P15_12	S4P15_13	S4P15_1
0	4	3	4	2	2	4	2	4	4	4	4	4	3	
1	2	2	4	4	2	5	4	2	2	5	4	2	2	
2	2	2	4	2	1	1	2	3	4	4	4	1	2	
3	4	4	4	2	2	2	4	2	4	4	4	2	2	
4	3	3	4	3	2	1	2	3	1	3	3	2	2	

In [104]:
```
resultados=[]
for columna in nbase:
    print(columna)
```

```
S4P15_1
S4P15_2
S4P15_3
S4P15_4
S4P15_5
S4P15_6
S4P15_7
S4P15_8
S4P15_9
S4P15_10
S4P15_11
S4P15_12
S4P15_13
S4P15_14
S4P15_15
FAC
```

```
resultados=[]
for columna in nbase:
    temp=nbase.groupby(columna)
    temp0=temp["FAC"].sum()/nbase["FAC"].sum()*100
    print(temp0)
```

```
S4P15_1
1     3.857005
2    43.091304
3    44.459612
4     6.644701
5     1.947378
Name: FAC, dtype: float64
S4P15_2
1     3.683050
2    35.910623
3    48.847864
4    10.341632
5     1.216832
Name: FAC, dtype: float64
S4P15_3
1     0.358156
2     5.200487
3    31.333550
4    61.973061
5     1.134747
Name: FAC, dtype: float64
S4P15_4
1     5.655421
2    50.281120
3    26.882024
```

In [109]:
```
resultados=[]
for columna in nbase:
    temp=nbase.groupby(columna)
    temp0=temp["FAC"].sum()/nbase["FAC"].sum()*100
    temp0.name = columna
    resultados.append(temp0)
```

In [111]:
```
resultados
```

Out[111]:
```
[S4P15_1
 1     3.857005
 2    43.091304
 3    44.459612
 4     6.644701
 5     1.947378
 Name: S4P15_1, dtype: float64, S4P15_2
 1     3.683050
 2    35.910623
 3    48.847864
 4    10.341632
 5     1.216832
 Name: S4P15_2, dtype: float64, S4P15_3
 1     0.358156
 2     5.200487
 3    31.333550
 4    61.973061
 5     1.134747
 Name: S4P15_3, dtype: float64, S4P15_4
 1     5.655421
 2    50.281120
```

In [112]:
```
resultados.pop()
resultados
```

Out[112]:
```
[S4P15_1
 1     3.857005
 2    43.091304
 3    44.459612
 4     6.644701
 5     1.947378
 Name: S4P15_1, dtype: float64, S4P15_2
 1     3.683050
 2    35.910623
 3    48.847864
 4    10.341632
 5     1.216832
 Name: S4P15_2, dtype: float64, S4P15_3
 1     0.358156
 2     5.200487
 3    31.333550
 4    61.973061
 5     1.134747
 Name: S4P15_3, dtype: float64, S4P15_4
 1     5.655421
 2    50.281120
 3    26.882024
```

In [114]:
```
pd.DataFrame(resultados)
```

Out[114]:

S4P15_15	1	2	3	4	5
S4P15_1	3.857005	43.091304	44.459612	6.644701	1.947378
S4P15_2	3.683050	35.910623	48.847864	10.341632	1.216832
S4P15_3	0.358156	5.200487	31.333550	61.973061	1.134747
S4P15_4	5.655421	50.281120	26.882024	12.543401	4.638034
S4P15_5	29.570499	54.027473	11.228825	1.613482	3.559721
S4P15_6	28.365080	55.990246	10.455184	1.554636	3.634855
S4P15_7	20.758132	59.277523	14.551017	1.905695	3.507632
S4P15_8	3.878444	37.769491	46.789535	8.198916	3.363614
S4P15_9	5.905738	37.986386	45.792268	8.508288	1.807319
S4P15_10	4.728196	27.904767	45.637429	19.120880	2.608727
S4P15_11	0.669358	8.720883	30.243725	59.522270	0.843764
S4P15_12	23.567627	63.160841	10.932853	1.747550	0.591129
S4P15_13	16.907715	59.492695	17.577556	2.349849	3.672186
S4P15_14	24.627367	57.025071	7.265236	1.328438	9.753889
S4P15_15	17.681844	61.460190	16.810982	2.646560	1.400423

In [119]:
```
resultados_df=pd.DataFrame(resultados)
```

In [123]:
```
resultados_df.columns=opciones
```

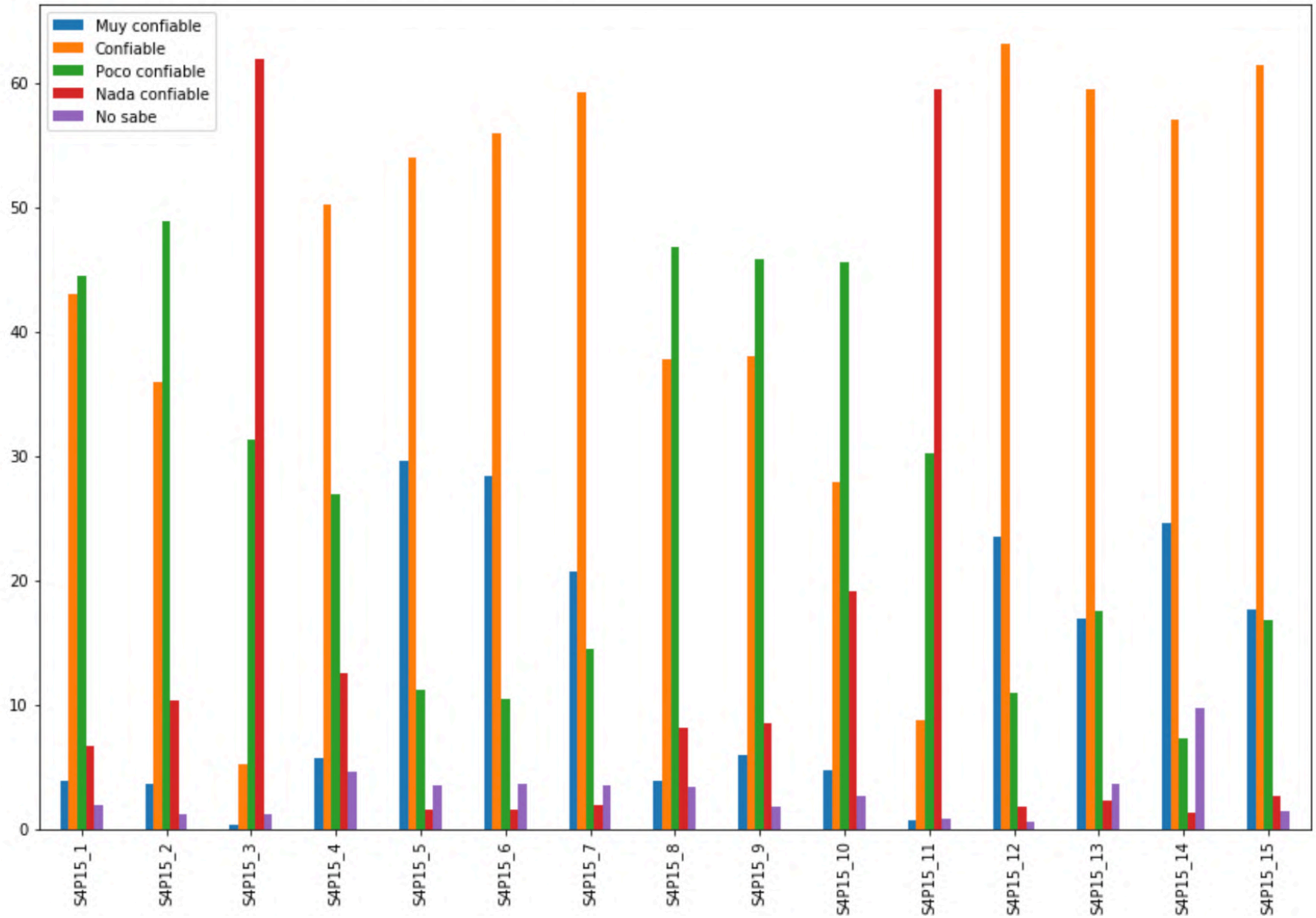
In [124]:
```
resultados_df
```

Out[124]:

	Muy confiable	Confiable	Poco confiable	Nada confiable	No sabe
S4P15_1	3.857005	43.091304	44.459612	6.644701	1.947378
S4P15_2	3.683050	35.910623	48.847864	10.341632	1.216832
S4P15_3	0.358156	5.200487	31.333550	61.973061	1.134747
S4P15_4	5.655421	50.281120	26.882024	12.543401	4.638034
S4P15_5	29.570499	54.027473	11.228825	1.613482	3.559721
S4P15_6	28.365080	55.990246	10.455184	1.554636	3.634855
S4P15_7	20.758132	59.277523	14.551017	1.905695	3.507632
S4P15_8	3.878444	37.769491	46.789535	8.198916	3.363614
S4P15_9	5.905738	37.986386	45.792268	8.508288	1.807319
S4P15_10	4.728196	27.904767	45.637429	19.120880	2.608727
S4P15_11	0.669358	8.720883	30.243725	59.522270	0.843764
S4P15_12	23.567627	63.160841	10.932853	1.747550	0.591129
S4P15_13	16.907715	59.492695	17.577556	2.349849	3.672186
S4P15_14	24.627367	57.025071	7.265236	1.328438	9.753889
S4P15_15	17.681844	61.460190	16.810982	2.646560	1.400423

In [127]:
```
resultados_df.plot(kind="bar",figsize=(15,10))
```

Out[127]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x116bfdf60>
```



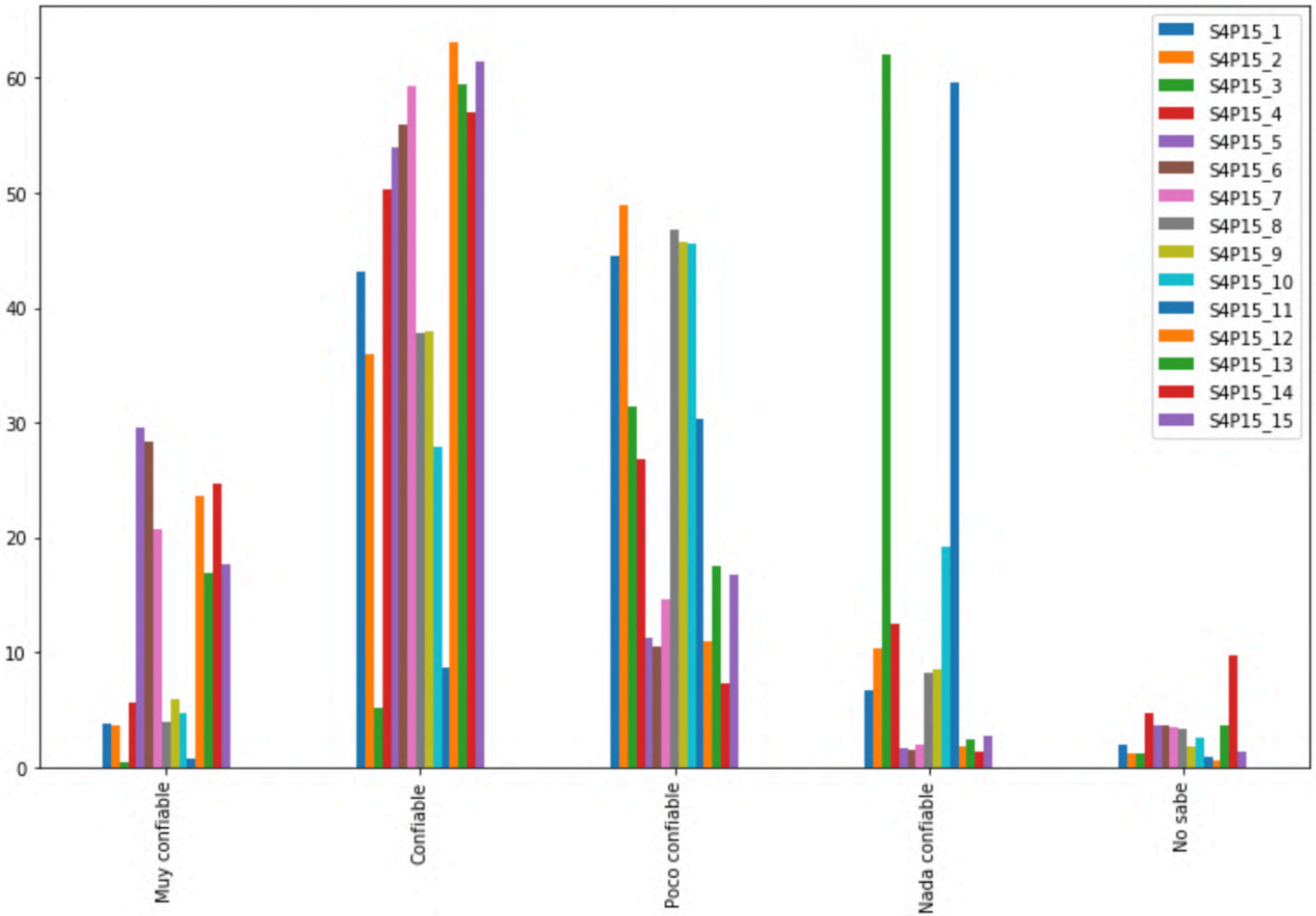
In [128]:
```
resultados_df.transpose()
```

Out[128]:

	S4P15_1	S4P15_2	S4P15_3	S4P15_4	S4P15_5	S4P15_6	S4P15_7	S4P15_8	S4P15_9	S4P15_10	S4P15_11	S4
Muy confiable	3.857005	3.683050	0.358156	5.655421	29.570499	28.365080	20.758132	3.878444	5.905738	4.728196	0.669358	23.
Confiable	43.091304	35.910623	5.200487	50.281120	54.027473	55.990246	59.277523	37.769491	37.986386	27.904767	8.720883	63.
Poco confiable	44.459612	48.847864	31.333550	26.882024	11.228825	10.455184	14.551017	46.789535	45.792268	45.637429	30.243725	10.
Nada confiable	6.644701	10.341632	61.973061	12.543401	1.613482	1.554636	1.905695	8.198916	8.508288	19.120880	59.522270	1.
No sabe	1.947378	1.216832	1.134747	4.638034	3.559721	3.634855	3.507632	3.363614	1.807319	2.608727	0.843764	0.

```
resultados_df.transpose().plot(kind="bar",figsize=(13,8))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1171d4d30>
```



In [163]:
```
resultados=[]
for columna in nbase:
    temp=nbase.groupby(columna)
    temp0=temp["FAC"].sum()/nbase["FAC"].sum()*100
    temp0.name = columna
    resultados.append(temp0)
```

In [166]:
```
def porcentajes(columna):
    temp = nbase.groupby(columna)
    temp0 = temp["FAC"].sum()/nbase["FAC"].sum()*100
    temp0.name = columna
    return temp0
```

In [167]:
```
porcentajes("S4P15_1")
```

Out[167]:
```
S4P15_1
1     3.857005
2    43.091304
3    44.459612
4     6.644701
5     1.947378
Name: S4P15_1, dtype: float64
```

In [173]:
```
resultados=[]
for columna in nbase:
    porc = porcentajes(columna)
    resultados.append(porc)
```

In [175]:
```
resultados.pop()
resultados
```

Out[175]:
```
[S4P15_1
 1     3.857005
 2    43.091304
 3    44.459612
 4     6.644701
 5     1.947378
 Name: S4P15_1, dtype: float64, S4P15_2
```

```
nbase[:5]
```

	S4P15_1	S4P15_2	S4P15_3	S4P15_4	S4P15_5	S4P15_6	S4P15_7	S4P15_8	S4P15_9	S4P15_10	S4P15_11	S4P15_12	S4P15_13	S4P15_1
0	4	3	4	2	2	4	2	4	4	4	4	4	3	
1	2	2	4	4	2	5	4	2	2	5	4	2	2	
2	2	2	4	2	1	1	2	3	4	4	4	1	2	
3	4	4	4	2	2	2	4	2	4	4	4	2	2	
4	3	3	4	3	2	1	2	3	1	3	3	2	2	

```
nbase.apply(sum)
```

```
S4P15_1          7772
S4P15_2          7676
S4P15_3         10156
S4P15_4          7500
S4P15_5          5897
S4P15_6          5900
S4P15_7          6450
S4P15_8          7648
S4P15_9          7099
S4P15_10         7985
S4P15_11         9754
S4P15_12         5725
S4P15_13         6393
S4P15_14         6636
S4P15_15         6203
FAC          37057876
dtype: int64
```

```
nbase.apply(lambda x: print(x[:5]))
```

```
0    4
1    2
2    2
3    4
4    3
Name: S4P15_1, dtype: int64
0    3
1    2
2    2
3    4
4    3
Name: S4P15_2, dtype: int64
0    4
1    4
2    4
3    4
4    4
Name: S4P15_3, dtype: int64
0    2
1    4
2    2
3    2
4    3
Name: S4P15_4, dtype: int64
0    2
1    2
2    1
3    2
4    2
Name: S4P15_5, dtype: int64
```

In [188]:
```
salida=nbase.apply(lambda x: (print(type(x))))
```

```
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
```

In [189]:
```
salida
```

Out[189]:
```
S4P15_1     None
S4P15_2     None
S4P15_3     None
S4P15_4     None
S4P15_5     None
S4P15_6     None
S4P15_7     None
S4P15_8     None
S4P15_9     None
S4P15_10    None
S4P15_11    None
S4P15_12    None
S4P15_13    None
S4P15_14    None
S4P15_15    None
FAC         None
dtype: object
```

In [192]:
```
nbase.apply(lambda x: x.name)
```

Out[192]:
```
S4P15_1      S4P15_1
S4P15_2      S4P15_2
S4P15_3      S4P15_3
S4P15_4      S4P15_4
S4P15_5      S4P15_5
S4P15_6      S4P15_6
S4P15_7      S4P15_7
S4P15_8      S4P15_8
S4P15_9      S4P15_9
S4P15_10    S4P15_10
S4P15_11    S4P15_11
S4P15_12    S4P15_12
S4P15_13    S4P15_13
S4P15_14    S4P15_14
S4P15_15    S4P15_15
FAC              FAC
dtype: object
```

In [208]:
```
nbase.apply(lambda x: porcentajes(x.name))
```

Out[208]:

	S4P15_1	S4P15_2	S4P15_3	S4P15_4	S4P15_5	S4P15_6	S4P15_7	S4P15_8	S4P15_9	S4P15_10	S4P15_11	S4P1
1	3.857005	3.683050	0.358156	5.655421	29.570499	28.365080	20.758132	3.878444	5.905738	4.728196	0.669358	23.56
2	43.091304	35.910623	5.200487	50.281120	54.027473	55.990246	59.277523	37.769491	37.986386	27.904767	8.720883	63.16
3	44.459612	48.847864	31.333550	26.882024	11.228825	10.455184	14.551017	46.789535	45.792268	45.637429	30.243725	10.93
4	6.644701	10.341632	61.973061	12.543401	1.613482	1.554636	1.905695	8.198916	8.508288	19.120880	59.522270	1.74
5	1.947378	1.216832	1.134747	4.638034	3.559721	3.634855	3.507632	3.363614	1.807319	2.608727	0.843764	0.59
629	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
637	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
687	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
703	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
720	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
728	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
740	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
742	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
746	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
750	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
760	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [199]:
```
prel=list(nbase.apply(lambda x: x.name))
prel
```

Out[199]:
```
['S4P15_1',
 'S4P15_2',
 'S4P15_3',
 'S4P15_4',
 'S4P15_5',
 'S4P15_6',
 'S4P15_7',
 'S4P15_8',
 'S4P15_9',
 'S4P15_10',
 'S4P15_11',
 'S4P15_12',
 'S4P15_13',
 'S4P15_14',
 'S4P15_15',
 'FAC']
```

In [200]:
```
prel.pop()
prel
```

Out[200]:
```
['S4P15_1',
 'S4P15_2',
 'S4P15_3',
 'S4P15_4',
 'S4P15_5',
 'S4P15_6',
 'S4P15_7',
 'S4P15_8',
 'S4P15_9',
 'S4P15_10',
 'S4P15_11',
 'S4P15_12',
 'S4P15_13',
 'S4P15_14',
 'S4P15_15']
```

In [203]:
```
pd.Series(prel).apply(lambda x: x)
```

Out[203]:
```
0      S4P15_1
1      S4P15_2
2      S4P15_3
3      S4P15_4
4      S4P15_5
5      S4P15_6
6      S4P15_7
7      S4P15_8
8      S4P15_9
9     S4P15_10
10    S4P15_11
11    S4P15_12
12    S4P15_13
13    S4P15_14
14    S4P15_15
dtype: object
```

In [204]:
```
pd.Series(prel).apply(lambda x: porcentajes(x))
```

Out[204]:

S4P15_15	1	2	3	4	5
0	3.857005	43.091304	44.459612	6.644701	1.947378
1	3.683050	35.910623	48.847864	10.341632	1.216832
2	0.358156	5.200487	31.333550	61.973061	1.134747
3	5.655421	50.281120	26.882024	12.543401	4.638034
4	29.570499	54.027473	11.228825	1.613482	3.559721
5	28.365080	55.990246	10.455184	1.554636	3.634855
6	20.758132	59.277523	14.551017	1.905695	3.507632
7	3.878444	37.769491	46.789535	8.198916	3.363614
8	5.905738	37.986386	45.792268	8.508288	1.807319
9	4.728196	27.904767	45.637429	19.120880	2.608727
10	0.669358	8.720883	30.243725	59.522270	0.843764
11	23.567627	63.160841	10.932853	1.747550	0.591129
12	16.907715	59.492695	17.577556	2.349849	3.672186
13	24.627367	57.025071	7.265236	1.328438	9.753889
14	17.681844	61.460190	16.810982	2.646560	1.400423

In [212]:
```
def porcentajes(columna):
    if(columna == "FAC"):
        return
    temp = nbase.groupby(columna)
    temp0 = temp["FAC"].sum()/nbase["FAC"].sum()*100
    temp0.name = columna
    return temp0
```

In [213]:
```
porcentajes("FAC")
```

In [215]:
```
res2=nbase.apply(lambda x: porcentajes(x.name))
```

In [219]:
```
res2
```

Out[219]:

S4P15_15	S4P15_1	S4P15_2	S4P15_3	S4P15_4	S4P15_5	S4P15_6	S4P15_7	S4P15_8	S4P15_9	S4P15_10	S4P15_11	S4
1	3.857005	3.683050	0.358156	5.655421	29.570499	28.365080	20.758132	3.878444	5.905738	4.728196	0.669358	23.5
2	43.091304	35.910623	5.200487	50.281120	54.027473	55.990246	59.277523	37.769491	37.986386	27.904767	8.720883	63.1
3	44.459612	48.847864	31.333550	26.882024	11.228825	10.455184	14.551017	46.789535	45.792268	45.637429	30.243725	10.9
4	6.644701	10.341632	61.973061	12.543401	1.613482	1.554636	1.905695	8.198916	8.508288	19.120880	59.522270	1.7
5	1.947378	1.216832	1.134747	4.638034	3.559721	3.634855	3.507632	3.363614	1.807319	2.608727	0.843764	0.

In [220]:
```
def porcentajesS(columna):
    if(columna.name == "FAC"):
        return
    temp = nbase.groupby(columna.name)
    temp0 = temp["FAC"].sum()/nbase["FAC"].sum()*100
    temp0.name = columna.name
    return temp0
```

In [223]:
```
nuevo = nbase.apply(porcentajesS)
nuevo
```

Out[223]:

S4P15_15	S4P15_1	S4P15_2	S4P15_3	S4P15_4	S4P15_5	S4P15_6	S4P15_7	S4P15_8	S4P15_9	S4P15_10	S4P15_11	S4
1	3.857005	3.683050	0.358156	5.655421	29.570499	28.365080	20.758132	3.878444	5.905738	4.728196	0.669358	23.5
2	43.091304	35.910623	5.200487	50.281120	54.027473	55.990246	59.277523	37.769491	37.986386	27.904767	8.720883	63.1
3	44.459612	48.847864	31.333550	26.882024	11.228825	10.455184	14.551017	46.789535	45.792268	45.637429	30.243725	10.9
4	6.644701	10.341632	61.973061	12.543401	1.613482	1.554636	1.905695	8.198916	8.508288	19.120880	59.522270	1.7
5	1.947378	1.216832	1.134747	4.638034	3.559721	3.634855	3.507632	3.363614	1.807319	2.608727	0.843764	0.

In [227]:
```
nuevo2 = nuevo.drop(["FAC"],axis=1)
```

In [228]:
```
nuevo2
```

Out[228]:

S4P15_15	S4P15_1	S4P15_2	S4P15_3	S4P15_4	S4P15_5	S4P15_6	S4P15_7	S4P15_8	S4P15_9	S4P15_10	S4P15_11	S4P
1	3.857005	3.683050	0.358156	5.655421	29.570499	28.365080	20.758132	3.878444	5.905738	4.728196	0.669358	23.5
2	43.091304	35.910623	5.200487	50.281120	54.027473	55.990246	59.277523	37.769491	37.986386	27.904767	8.720883	63.1
3	44.459612	48.847864	31.333550	26.882024	11.228825	10.455184	14.551017	46.789535	45.792268	45.637429	30.243725	10.9
4	6.644701	10.341632	61.973061	12.543401	1.613482	1.554636	1.905695	8.198916	8.508288	19.120880	59.522270	1.7
5	1.947378	1.216832	1.134747	4.638034	3.559721	3.634855	3.507632	3.363614	1.807319	2.608727	0.843764	0.5

In [229]:
```
nuevo2.index.name
```

Out[229]:
```
'S4P15_15'
```

In [230]:
```
nuevo2.index.name = "respuestas"
```

In [231]:
```
nuevo2
```

Out[231]:

respuestas	S4P15_1	S4P15_2	S4P15_3	S4P15_4	S4P15_5	S4P15_6	S4P15_7	S4P15_8	S4P15_9	S4P15_10	S4P15_11	S
1	3.857005	3.683050	0.358156	5.655421	29.570499	28.365080	20.758132	3.878444	5.905738	4.728196	0.669358	23
2	43.091304	35.910623	5.200487	50.281120	54.027473	55.990246	59.277523	37.769491	37.986386	27.904767	8.720883	63
3	44.459612	48.847864	31.333550	26.882024	11.228825	10.455184	14.551017	46.789535	45.792268	45.637429	30.243725	10
4	6.644701	10.341632	61.973061	12.543401	1.613482	1.554636	1.905695	8.198916	8.508288	19.120880	59.522270	1
5	1.947378	1.216832	1.134747	4.638034	3.559721	3.634855	3.507632	3.363614	1.807319	2.608727	0.843764	0

14. DESEMPEÑO DE PROFESIONES Y ACTIVIDADES

En una escala del 1 al 10, donde 10 equivale a "Muy respetable", ¿en México, cómo califica usted el desempeño de un. . .

ANOTE EL CÓDIGO SEGÚN CORRESPONDA, SI NO SABE, ANOTE 11

1 juez? ..... ⌊_⌋⌊_⌋

2 médico? ..... ⌊_⌋⌊_⌋

3 abogado? ..... ⌊_⌋⌊_⌋

4 hombre de negocios? ..... ⌊_⌋⌊_⌋

5 periodista? ..... ⌊_⌋⌊_⌋

6 banquero? ..... ⌊_⌋⌊_⌋

7 ingeniero? ..... ⌊_⌋⌊_⌋

8 arquitecto? ..... ⌊_⌋⌊_⌋

9 profesor? ..... ⌊_⌋⌊_⌋

10 deportista? ..... ⌊_⌋⌊_⌋

11 artista? ..... ⌊_⌋⌊_⌋

12 bombero? ..... ⌊_⌋⌊_⌋

13 enfermera? ..... ⌊_⌋⌊_⌋

14 oficial de policía? ..... ⌊_⌋⌊_⌋

15 sacerdote o ministro de culto? ..... ⌊_⌋⌊_⌋

16 investigador? ..... ⌊_⌋⌊_⌋

17 inventor? ..... ⌊_⌋⌊_⌋