Heatmap using seabron

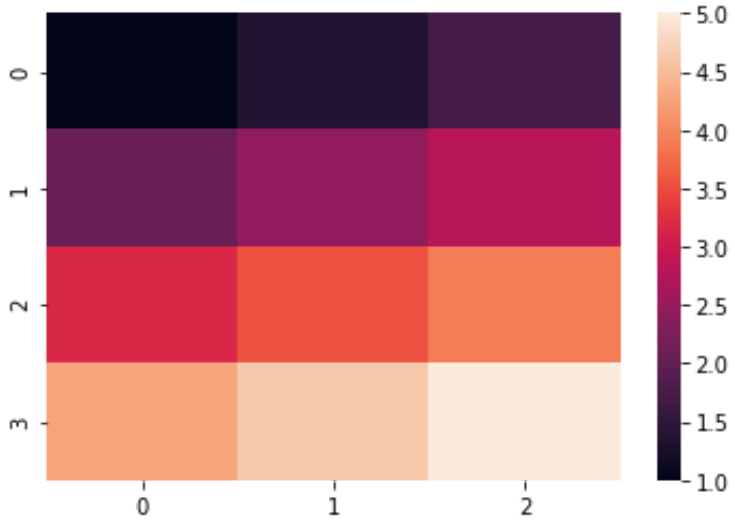
In [1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

In [2]:
```
data_2d = np.linspace(1,5 ,12).reshape(4,3)
data_2d
```

Out[2]:
```
array([[1.        , 1.36363636, 1.72727273],
       [2.09090909, 2.45454545, 2.81818182],
       [3.18181818, 3.54545455, 3.90909091],
       [4.27272727, 4.63636364, 5.        ]])
```

In [3]:
```
sns.heatmap(data_2d)
```

Out[3]:
```
<AxesSubplot:>
```



heatmap show only numeric valuse

In [4]:
```
data=pd.read_csv("who is responsible for global warming.csv")
data.head(2)
```

Out[4]:

	Country Name	Country Code	Indicator Name	Indicator Code	2000	2001	2002	2003	2004	2005	2006	20
0	United States	USA	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	20.178751	19.636505	19.613404	19.564105	19.658371	19.591885	19.094067	19.217
1	United Kingdom	GBR	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	9.199549	9.233175	8.904123	9.053278	8.989140	8.982939	8.898710	8.617

In [5]:
```
data1=data.drop(["Country Code","Indicator Name","Indicator Code"],axis=1).set_index("Country Name")
```

In [6]:
```
plt.figure(figsize=(16,9))
sns.heatmap(data1)
```

Out[6]: <AxesSubplot:ylabel='Country Name'>


```
sns.heatmap(
    data,
    vmin=None,
    vmax=None,
    cmap=None,
    center=None,
    robust=False,
    annot=None,
    fmt='.2g',
    annot_kws=None,
    linewidths=0,
    linecolor='white',
    cbar=True,
    cbar_kws=None,
    cbar_ax=None,
```

```
    square=False,
    xticklabels='auto',
    yticklabels='auto',
    mask=None,
    ax=None,
    **kwargs,
)
Docstring:
Plot rectangular data as a color-encoded matrix.

This is an Axes-level function and will draw the heatmap into the
currently-active Axes if none is provided to the ``ax`` argument.  Part of
this Axes space will be taken and used to plot a colormap, unless ``cbar``
is False or a separate Axes is provided to ``cbar_ax``.

Parameters
----------
data : rectangular dataset
    2D dataset that can be coerced into an ndarray. If a Pandas DataFrame
    is provided, the index/column information will be used to label the
    columns and rows.
vmin, vmax : floats, optional
    Values to anchor the colormap, otherwise they are inferred from the
    data and other keyword arguments.
cmap : matplotlib colormap name or object, or list of colors, optional
    The mapping from data values to color space. If not provided, the
    default will depend on whether ``center`` is set.
center : float, optional
    The value at which to center the colormap when plotting divergant data.
    Using this parameter will change the default ``cmap`` if none is
    specified.
robust : bool, optional
    If True and ``vmin`` or ``vmax`` are absent, the colormap range is
    computed with robust quantiles instead of the extreme values.
annot : bool or rectangular dataset, optional
    If True, write the data value in each cell. If an array-like with the
    same shape as ``data``, then use this to annotate the heatmap instead
    of the data. Note that DataFrames will match on position, not index.
fmt : string, optional
```

```
    String formatting code to use when adding annotations.
annot_kws : dict of key, value mappings, optional
    Keyword arguments for ``ax.text`` when ``annot`` is True.
linewidths : float, optional
    Width of the lines that will divide each cell.
linecolor : color, optional
    Color of the lines that will divide each cell.
cbar : boolean, optional
    Whether to draw a colorbar.
cbar_kws : dict of key, value mappings, optional
    Keyword arguments for `fig.colorbar`.
cbar_ax : matplotlib Axes, optional
    Axes in which to draw the colorbar, otherwise take space from the
    main Axes.
square : boolean, optional
    If True, set the Axes aspect to "equal" so each cell will be
    square-shaped.
xticklabels, yticklabels : "auto", bool, list-like, or int, optional
    If True, plot the column names of the dataframe. If False, don't plot
    the column names. If list-like, plot these alternate labels as the
    xticklabels. If an integer, use the column names but plot only every
    n label. If "auto", try to densely plot non-overlapping labels.
mask : boolean array or DataFrame, optional
    If passed, data will not be shown in cells where ``mask`` is True.
    Cells with missing values are automatically masked.
ax : matplotlib Axes, optional
    Axes in which to draw the plot, otherwise use the currently-active
    Axes.
kwargs : other keyword arguments
    All other keyword arguments are passed to
    :func:`matplotlib.axes.Axes.pcolormesh`.

Returns
-------
ax : matplotlib Axes
    Axes object with the heatmap.

See also
--------
```

```
clustermap : Plot a matrix using hierachical clustering to arrange the
             rows and columns.

Examples
--------

Plot a heatmap for a numpy array:

.. plot::
    :context: close-figs

    >>> import numpy as np; np.random.seed(0)
    >>> import seaborn as sns; sns.set()
    >>> uniform_data = np.random.rand(10, 12)
    >>> ax = sns.heatmap(uniform_data)

Change the limits of the colormap:

.. plot::
    :context: close-figs

    >>> ax = sns.heatmap(uniform_data, vmin=0, vmax=1)

Plot a heatmap for data centered on 0 with a diverging colormap:

.. plot::
    :context: close-figs

    >>> normal_data = np.random.randn(10, 12)
    >>> ax = sns.heatmap(normal_data, center=0)

Plot a dataframe with meaningful row and column labels:

.. plot::
    :context: close-figs

    >>> flights = sns.load_dataset("flights")
    >>> flights = flights.pivot("month", "year", "passengers")
    >>> ax = sns.heatmap(flights)
```

```
Annotate each cell with the numeric value using integer formatting:

.. plot::
    :context: close-figs

    >>> ax = sns.heatmap(flights, annot=True, fmt="d")

Add lines between each cell:

.. plot::
    :context: close-figs

    >>> ax = sns.heatmap(flights, linewidths=.5)

Use a different colormap:

.. plot::
    :context: close-figs

    >>> ax = sns.heatmap(flights, cmap="YlGnBu")

Center the colormap at a specific value:

.. plot::
    :context: close-figs

    >>> ax = sns.heatmap(flights, center=flights.loc["January", 1955])

Plot every other column label and don't plot row labels:

.. plot::
    :context: close-figs

    >>> data = np.random.randn(50, 20)
    >>> ax = sns.heatmap(data, xticklabels=2, yticklabels=False)

Don't draw a colorbar:
```

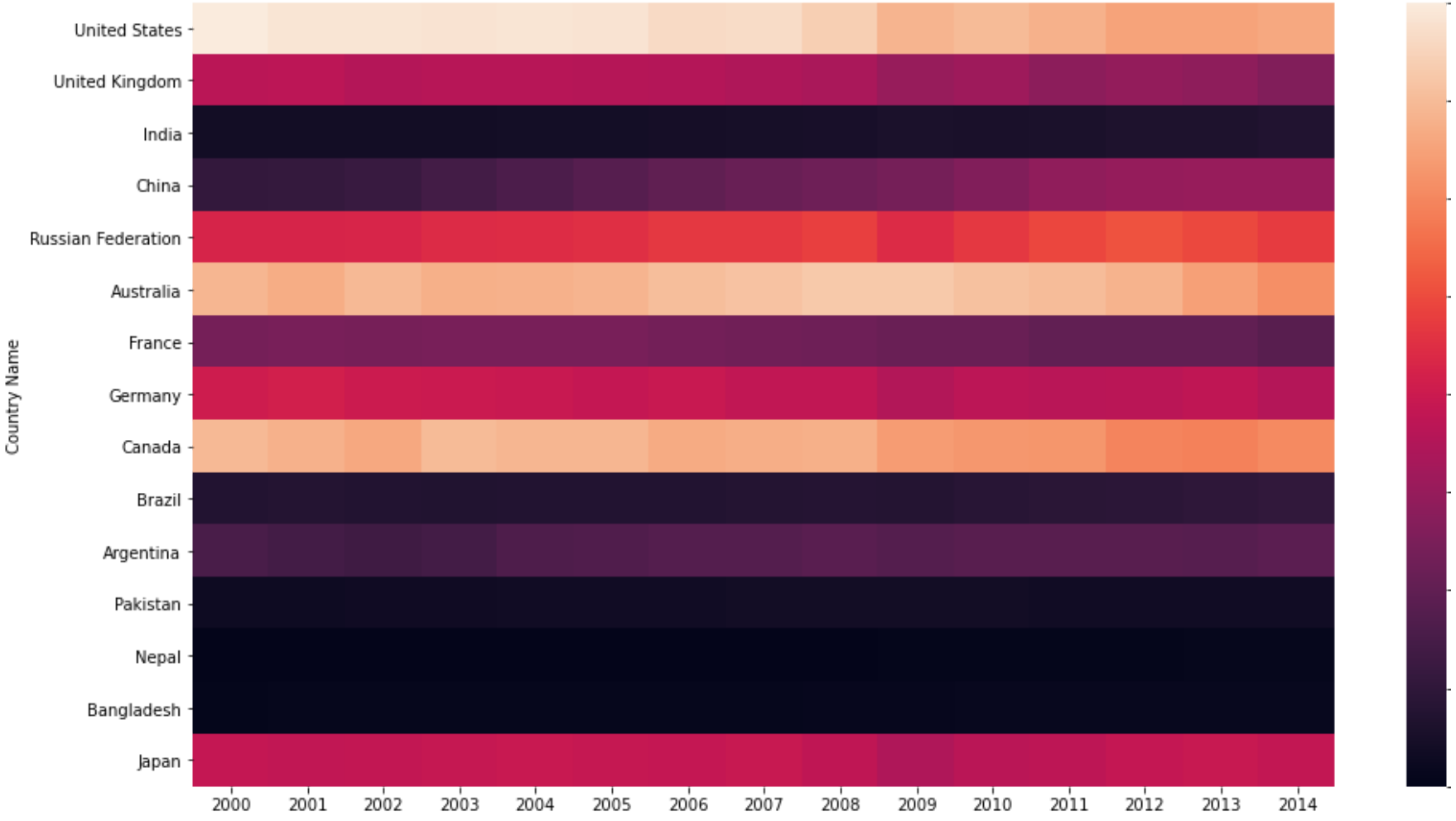
```
.. plot::
    :context: close-figs

    >>> ax = sns.heatmap(flights, cbar=False)

Use different axes for the colorbar:

.. plot::
    :context: close-figs

    >>> grid_kws = {"height_ratios": (.9, .05), "hspace": .3}
    >>> f, (ax, cbar_ax) = plt.subplots(2, gridspec_kw=grid_kws)
    >>> ax = sns.heatmap(flights, ax=ax,
    ...                  cbar_ax=cbar_ax,
    ...                  cbar_kws={"orientation": "horizontal"})

Use a mask to plot only part of a matrix

.. plot::
    :context: close-figs

    >>> corr = np.corrcoef(np.random.randn(10, 200))
    >>> mask = np.zeros_like(corr)
    >>> mask[np.triu_indices_from(mask)] = True
    >>> with sns.axes_style("white"):
    ...     f, ax = plt.subplots(figsize=(7, 5))
    ...     ax = sns.heatmap(corr, mask=mask, vmax=.3, square=True)
```

```
sns.heatmap(
    data,
    vmin=None,
    vmax=None,
    cmap=None,
    center=None,
    robust=False,
    annot=None,
    fmt='.2g',
    annot_kws=None,
```

```
    linewidths=0,
    linecolor='white',
    cbar=True,
    cbar_kws=None,
    cbar_ax=None,
    square=False,
    xticklabels='auto',
    yticklabels='auto',
    mask=None,
    ax=None,
    **kwargs,
)
```

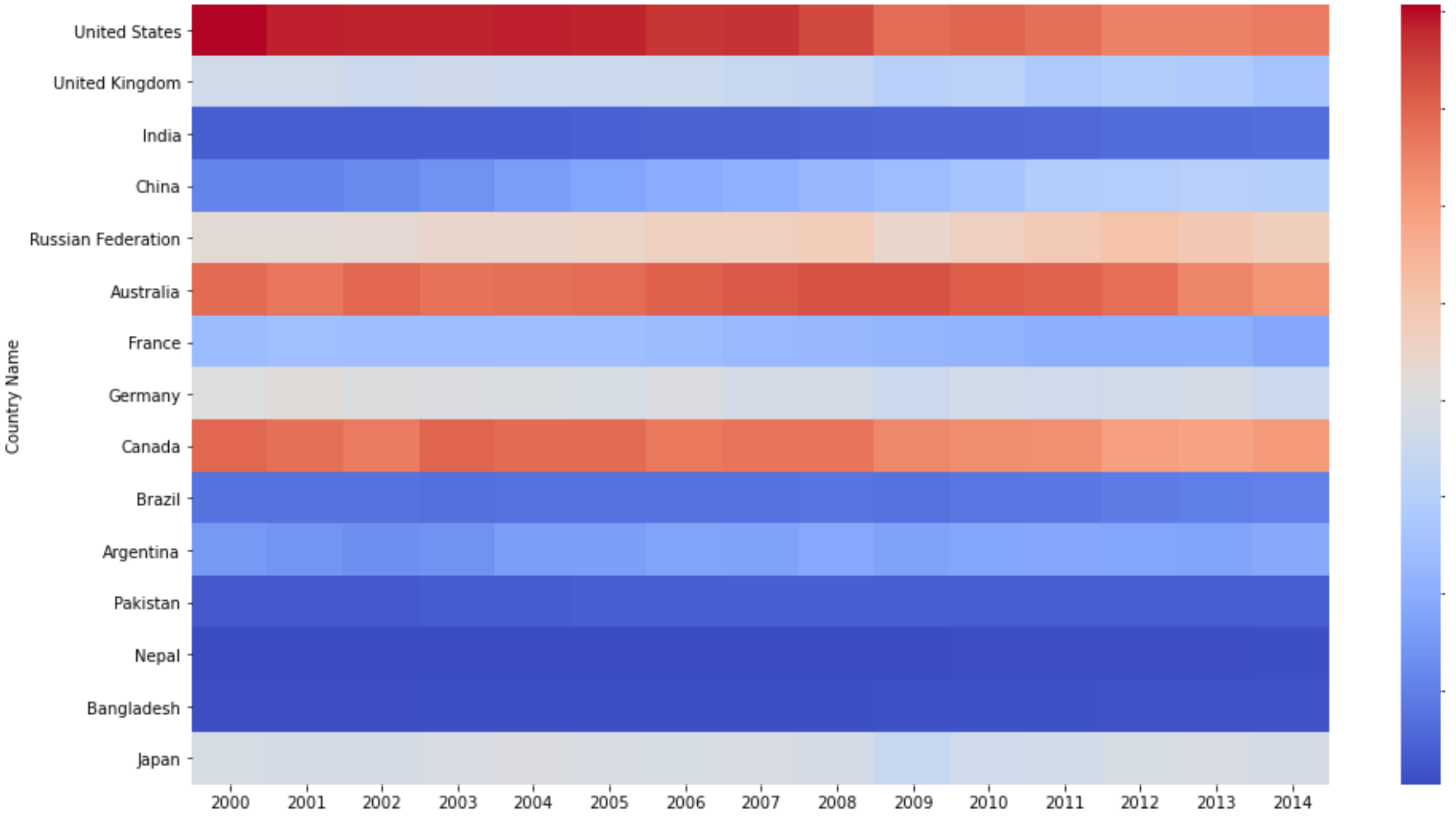
In [7]:
```
plt.figure(figsize=(16,9))
sns.heatmap(data1, vmin=0,vmax=20)
```

Out[7]: <AxesSubplot:ylabel='Country Name'>



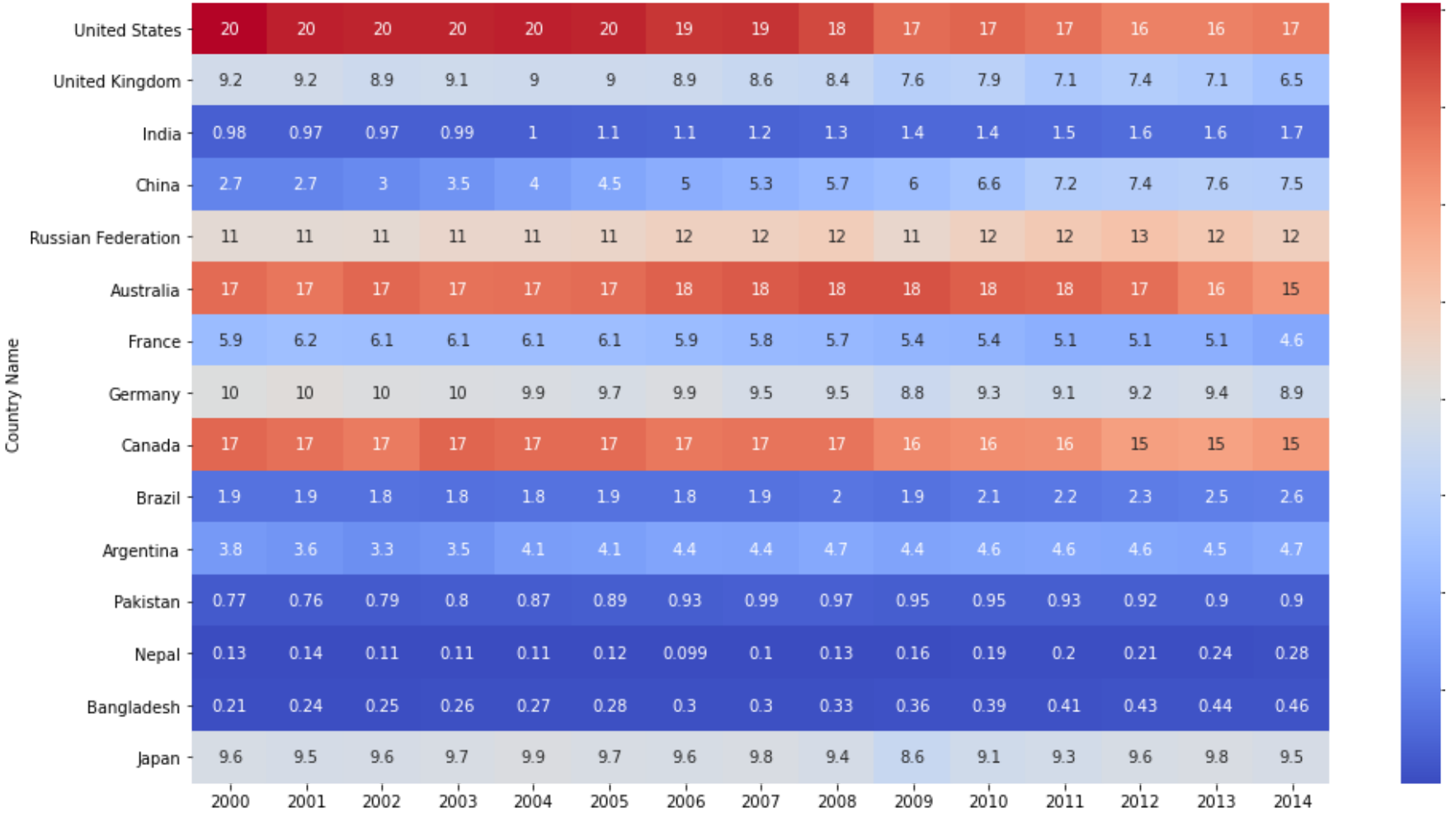
In [8]:
```
"""
cmap valuse= supported values are 'Accent', 'Accent_r', 'Blues',
'Blues_r', 'BrBG', 'BrBG_r', 'BuGn', 'BuGn_r', 'BuPu', 'BuPu_r',
'CMRmap', 'CMRmap_r', 'Dark2', 'Dark2_r', 'GnBu', 'GnBu_r', 'Greens',
'Greens_r', 'Greys', 'Greys_r', 'OrRd', 'OrRd_r', 'Oranges', 'Oranges_r',
'PRGn', 'PRGn_r', 'Paired', 'Paired_r', 'Pastel1', 'Pastel1_r', 'Pastel2',
'Pastel2_r', 'PiYG', 'PiYG_r', 'PuBu', 'PuBuGn', 'PuBuGn_r', 'PuBu_r', 'PuOr',
'PuOr_r', 'PuRd', 'PuRd_r', 'Purples', 'Purples_r', 'RdBu', 'RdBu_r', 'RdGy',
'RdGy_r', 'RdPu', 'RdPu_r', 'RdYlBu', 'RdYlBu_r', 'RdYlGn', 'RdYlGn_r', 'Reds',
'Reds_r', 'Set1', 'Set1_r', 'Set2', 'Set2_r', 'Set3', 'Set3_r', 'Spectral', 'Spectral_r',
'Wistia', 'Wistia_r', 'YlGn', 'YlGnBu', 'YlGnBu_r', 'YlGn_r', 'YlOrBr', 'YlOrBr_r', 'YlOrRd',
'YlOrRd_r', 'afmhot', 'afmhot_r', 'autumn', 'autumn_r', 'binary', 'binary_r', 'bone', 'bone_r',
'brg', 'brg_r', 'bwr', 'bwr_r', 'cividis', 'cividis_r', 'cool', 'cool_r', 'coolwarm',
'coolwarm_r', 'copper', 'copper_r', 'cubehelix', 'cubehelix_r', 'flag', 'flag_r',
'gist_earth', 'gist_earth_r', 'gist_gray', 'gist_gray_r', 'gist_heat', 'gist_heat_r',
'gist_ncar', 'gist_ncar_r', 'gist_rainbow', 'gist_rainbow_r', 'gist_stern', 'gist_stern_r',
'gist_yarg', 'gist_yarg_r', 'gnuplot', 'gnuplot2', 'gnuplot2_r', 'gnuplot_r', 'gray',
'gray_r', 'hot', 'hot_r', 'hsv', 'hsv_r', 'icefire', 'icefire_r', 'inferno', 'inferno_r',
'jet', 'jet_r', 'magma', 'magma_r', 'mako', 'mako_r', 'nipy_spectral', 'nipy_spectral_r',
'ocean', 'ocean_r', 'pink', 'pink_r', 'plasma', 'plasma_r', 'prism', 'prism_r', 'rainbow',
'rainbow_r', 'rocket', 'rocket_r', 'seismic', 'seismic_r', 'spring', 'spring_r', 'summer',
'summer_r', 'tab10', 'tab10_r', 'tab20', 'tab20_r', 'tab20b', 'tab20b_r', 'tab20c', 'tab20c_r',
'terrain', 'terrain_r', 'turbo', 'turbo_r', 'twilight', 'twilight_r', 'twilight_shifted',
'twilight_shifted_r', 'viridis', 'viridis_r', 'vlag', 'vlag_r', 'winter', 'winter_r'
"""
plt.figure(figsize=(16,9))

sns.heatmap(data1, cmap="coolwarm")
```

Out[8]: <AxesSubplot:ylabel='Country Name'>


United States
United Kingdom
India
China
Russian Federation
Australia
France
Germany
Canada
Brazil
Argentina
Pakistan
Nepal
Bangladesh
Japan
Country Name
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014

In [9]:
```
plt.figure(figsize=(16,9))

sns.heatmap(data1, cmap="coolwarm",annot=True)
```

Out[9]: <AxesSubplot:ylabel='Country Name'>



In [10]:
```
data_2d
```

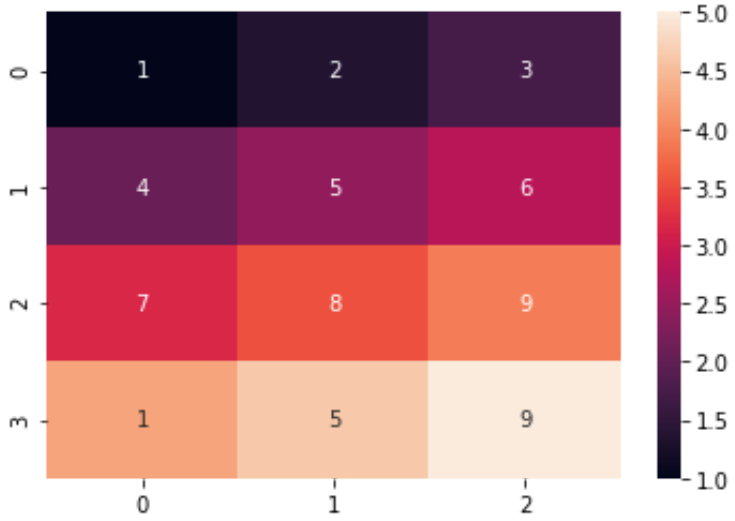
Out[10]:
```
array([[1.        , 1.36363636, 1.72727273],
       [2.09090909, 2.45454545, 2.81818182],
       [3.18181818, 3.54545455, 3.90909091],
       [4.27272727, 4.63636364, 5.        ]])
```

In [11]:
```
arr_2d=np.array([[1,2,3],[4,5,6],[7,8,9],[1,5,9]])
arr_2d
```

Out[11]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9],
       [1, 5, 9]])
```

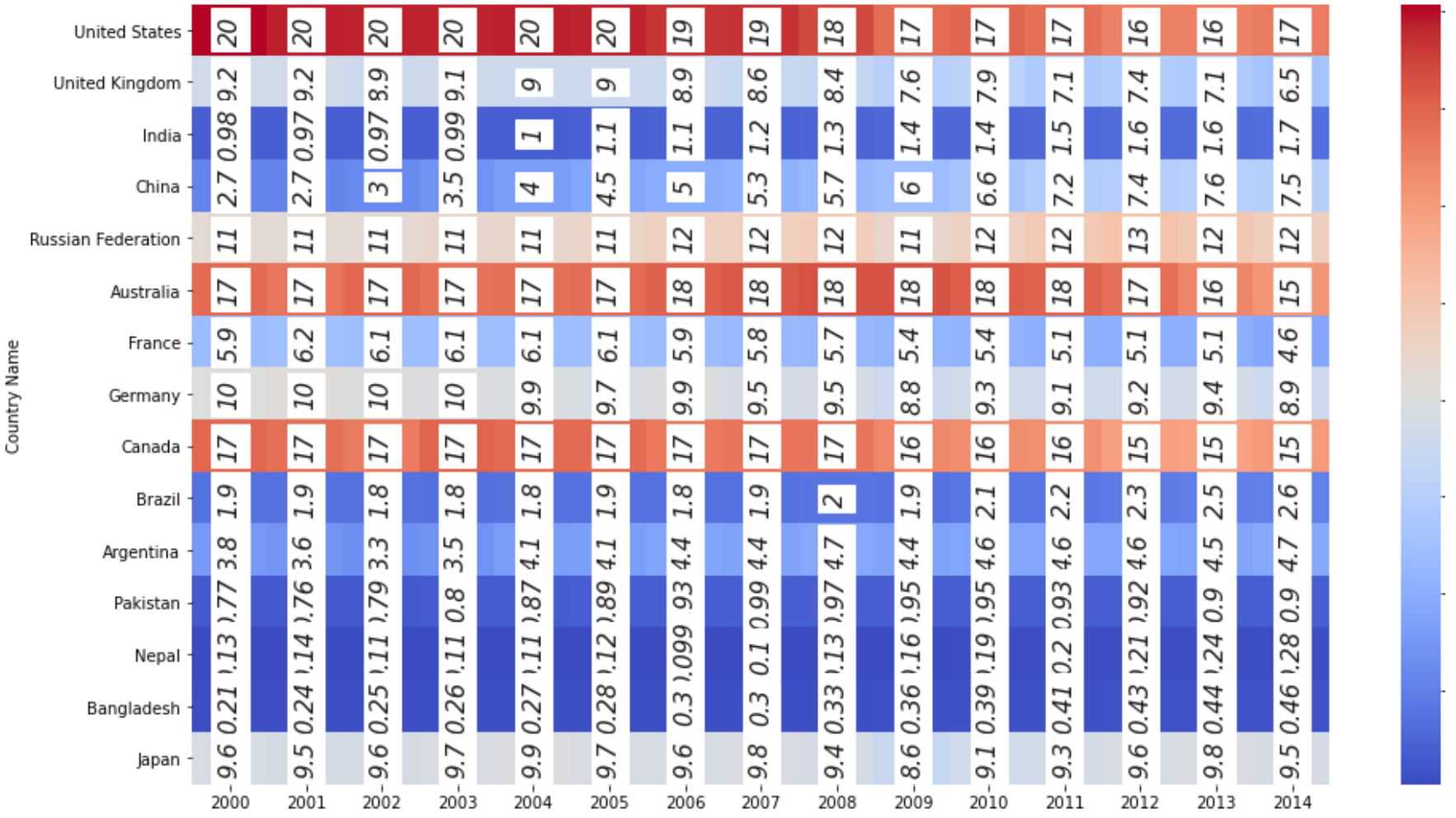
In [12]:
```
sns.heatmap(data_2d,annot=arr_2d,fmt="d")#  fmt----str="s", decimal="d"
```

Out[12]:
```
<AxesSubplot:>
```



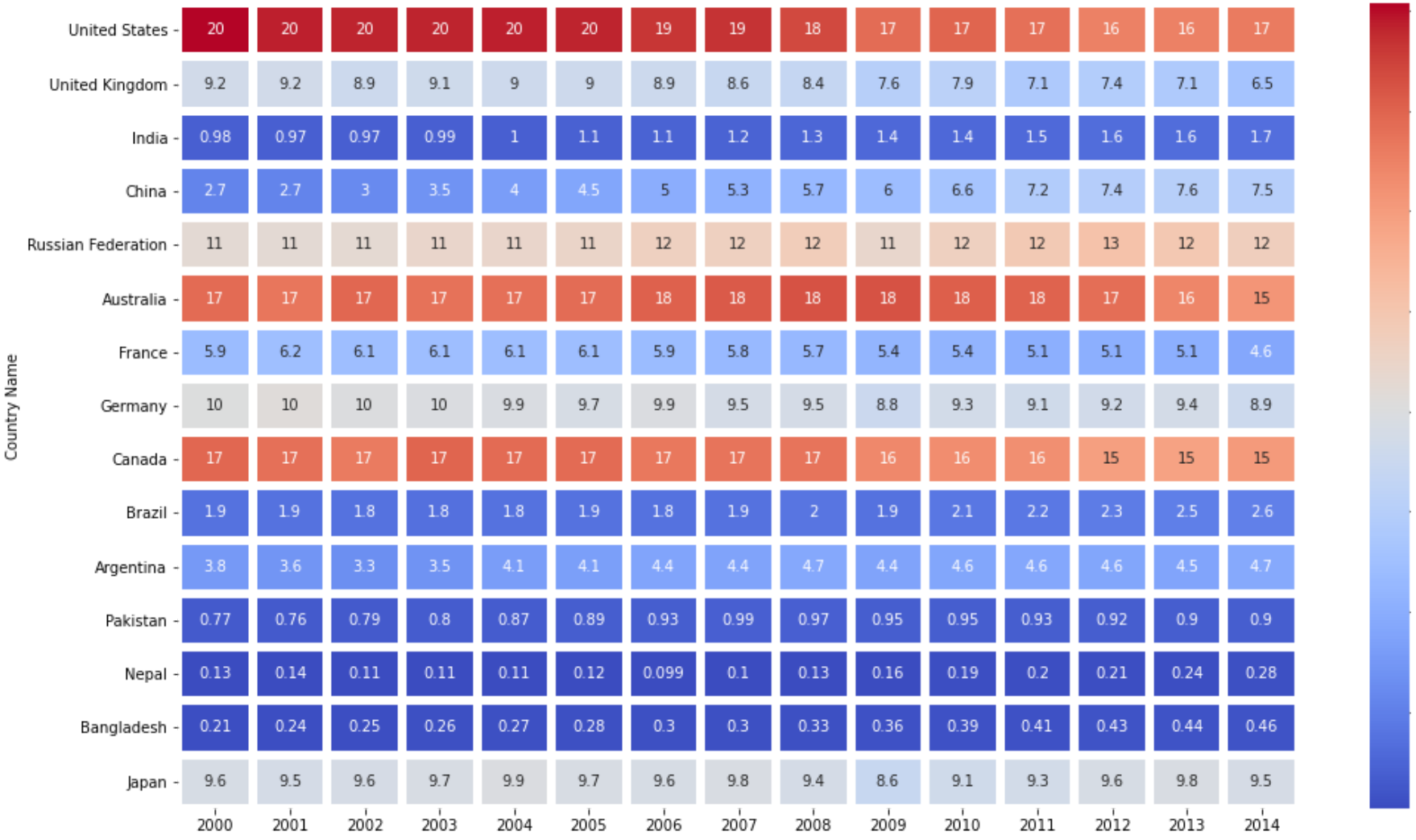
In [13]:
```
plt.figure(figsize=(16,9))
annot_kws1={
    "fontsize":15,
    "fontstyle":"italic",
   'color':'k',
    "alpha":0.9,
    "rotation":"vertical",
    "verticalalignment":"center",
    "backgroundcolor":"w"
}

sns.heatmap(data1, cmap="coolwarm",annot=True,annot_kws=annot_kws1)
```

Out[13]:
```
<AxesSubplot:ylabel='Country Name'>
```


Country Name
United States
United Kingdom
India
China
Russian Federation
Australia
France
Germany
Canada
Brazil
Argentina
Pakistan
Nepal
Bangladesh
Japan
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014

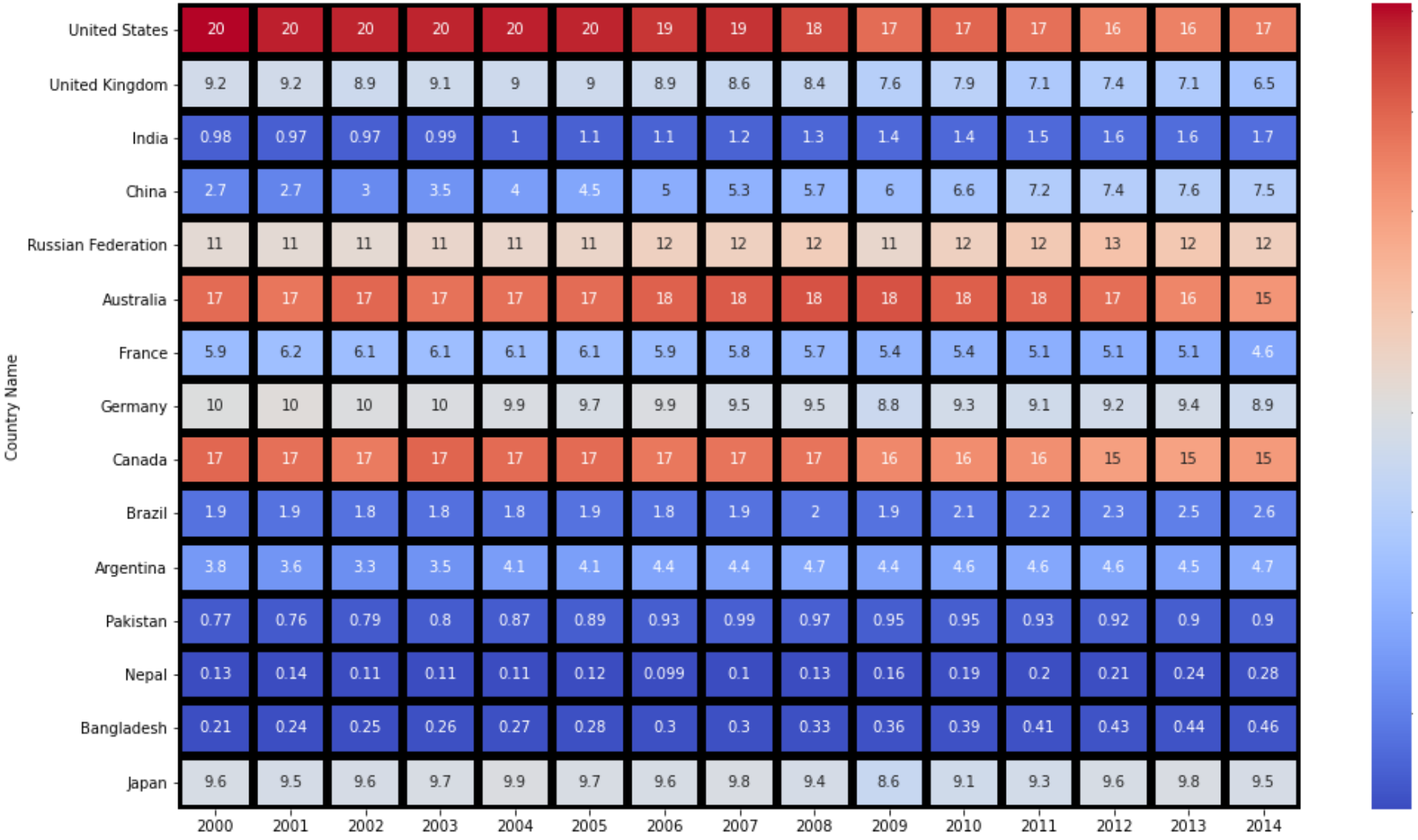
In [14]:
```
plt.figure(figsize=(17,10))

sns.heatmap(data1, cmap="coolwarm",annot=True,linewidths=5)
```

Out[14]: <AxesSubplot:ylabel='Country Name'>



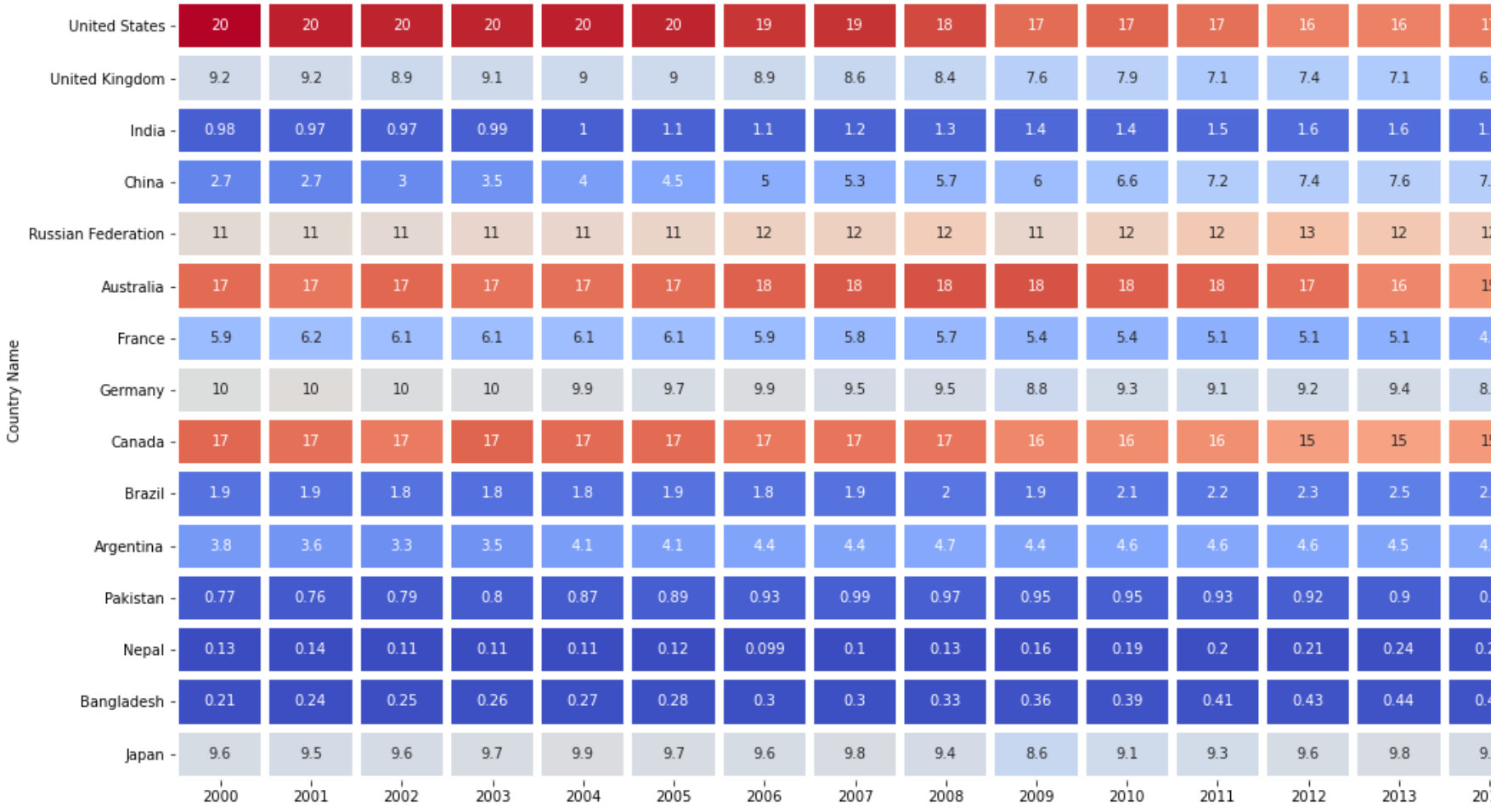
In [15]:
```
plt.figure(figsize=(17,10))

sns.heatmap(data1, cmap="coolwarm",annot=True,linewidths=5,linecolor="k")
```

Out[15]: <AxesSubplot:ylabel='Country Name'>



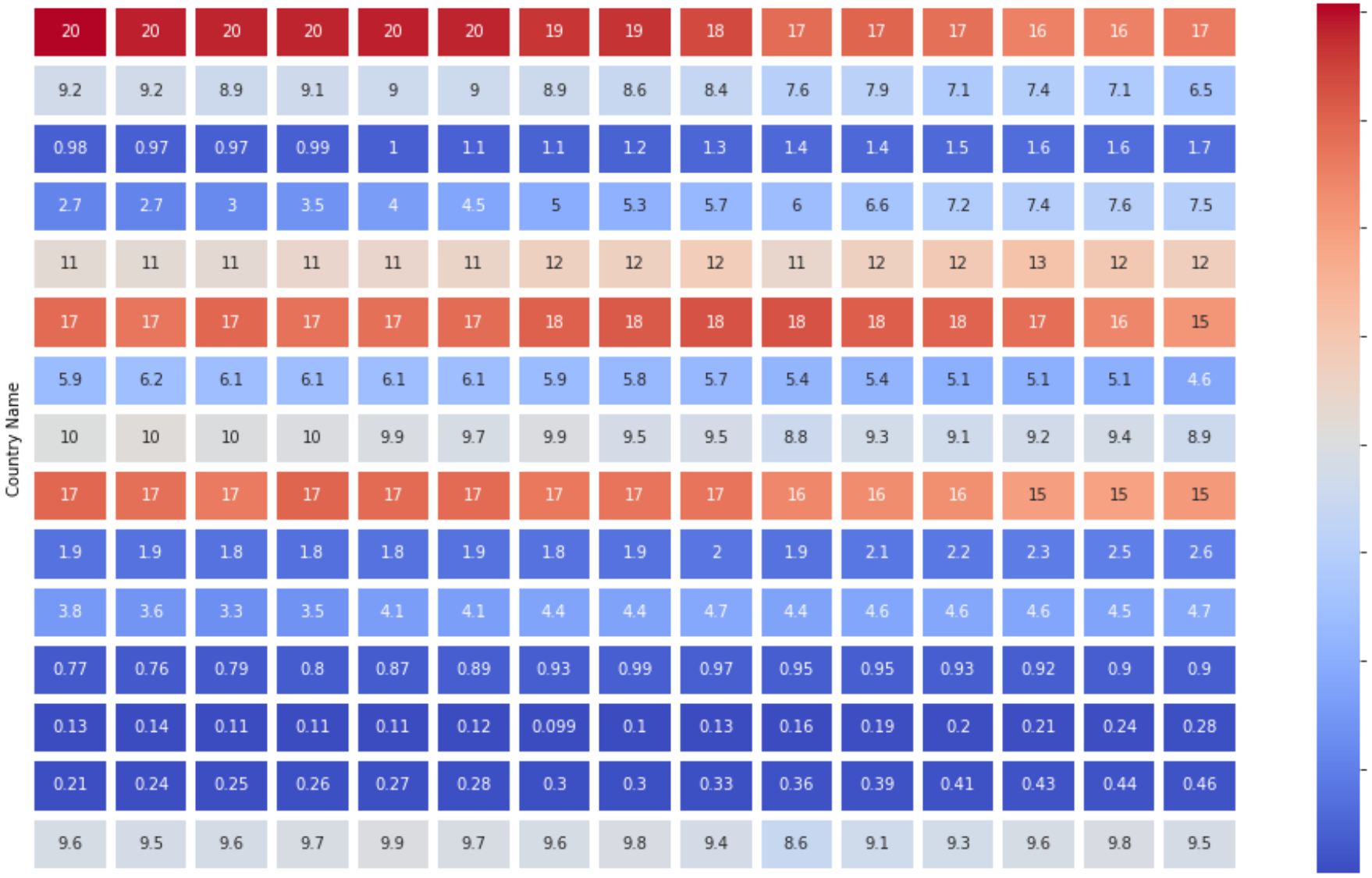
In [16]:
```
plt.figure(figsize=(17,10))

sns.heatmap(data1, cmap="coolwarm",annot=True,linewidths=5,cbar=False)
```

Out[16]: <AxesSubplot:ylabel='Country Name'>



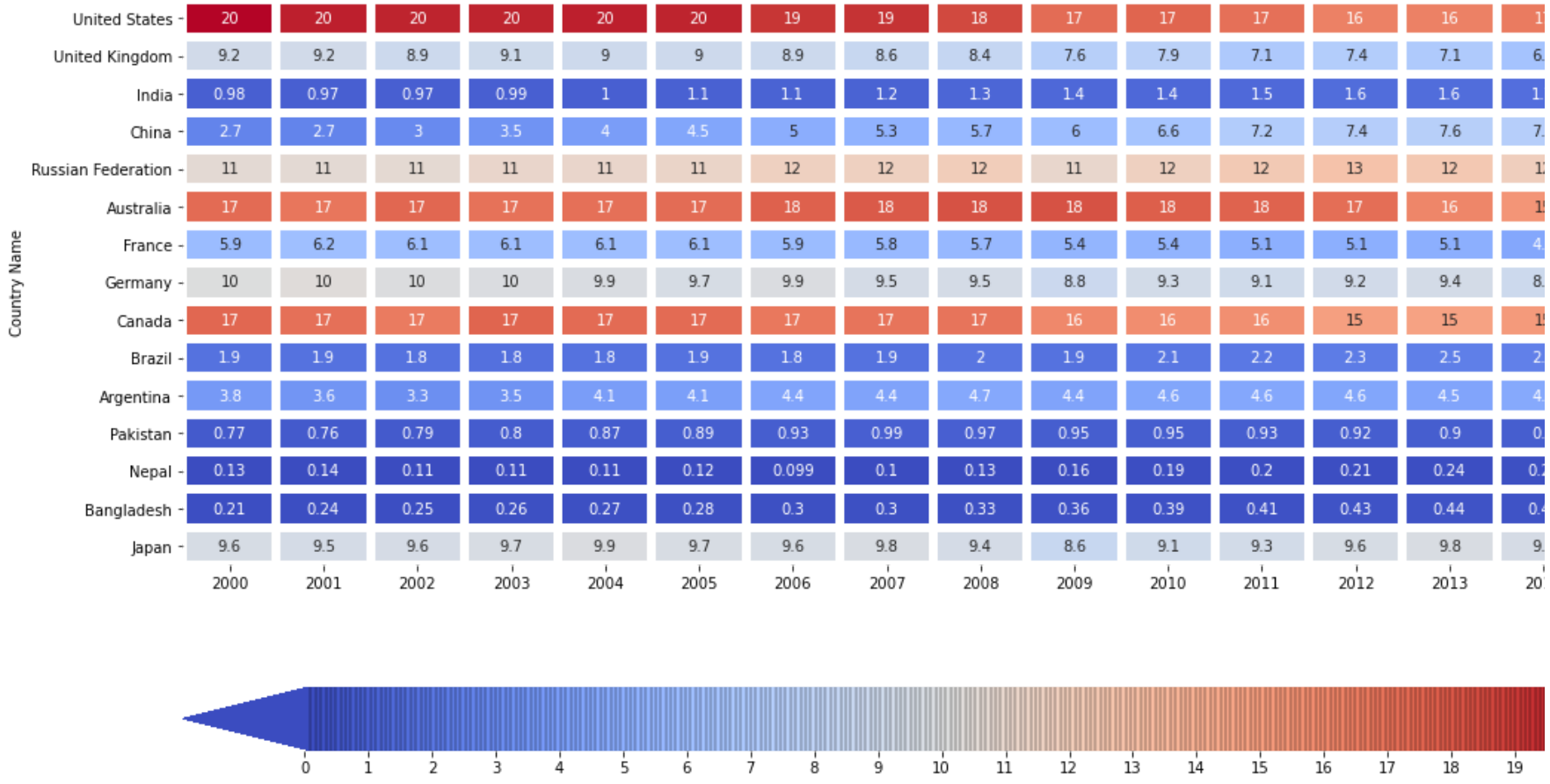
In [17]:
```
plt.figure(figsize=(17,10))

sns.heatmap(data1, cmap="coolwarm",annot=True,linewidths=5,xticklabels=False,yticklabels=False)
```

Out[17]: <AxesSubplot:ylabel='Country Name'>



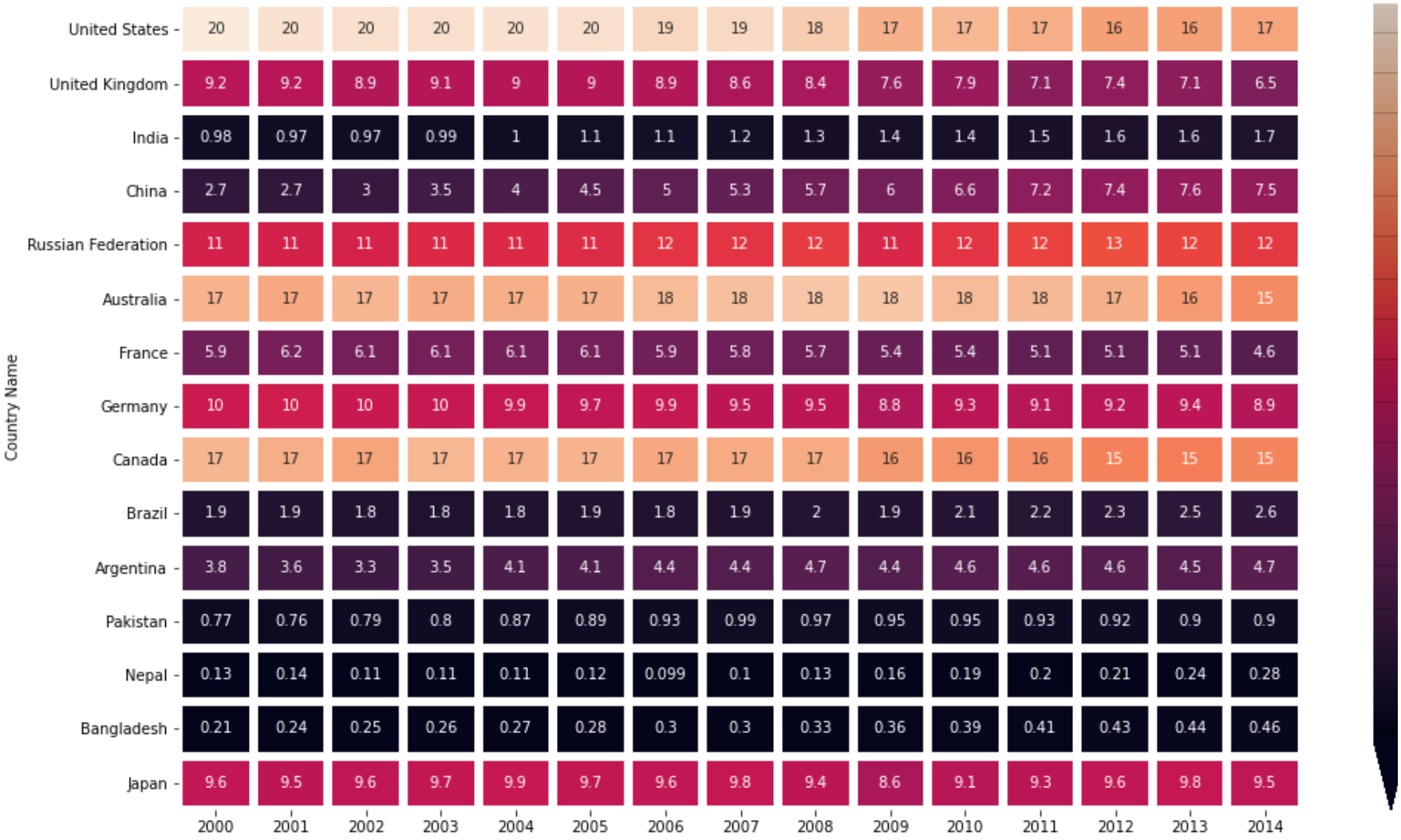
In [18]:
```
plt.figure(figsize=(17,10))

sns.heatmap(data1, cmap="coolwarm",annot=True,linewidths=5,cbar_kws={
    #"orientation":"vertical",
    "orientation":"horizontal",
    "shrink":1,
    "extend":"min",#min,max,both
    "extendfrac":0.1,
    "ticks":np.arange(0,22),
    "drawedges":True,
})
```

Out[18]: <AxesSubplot:ylabel='Country Name'>



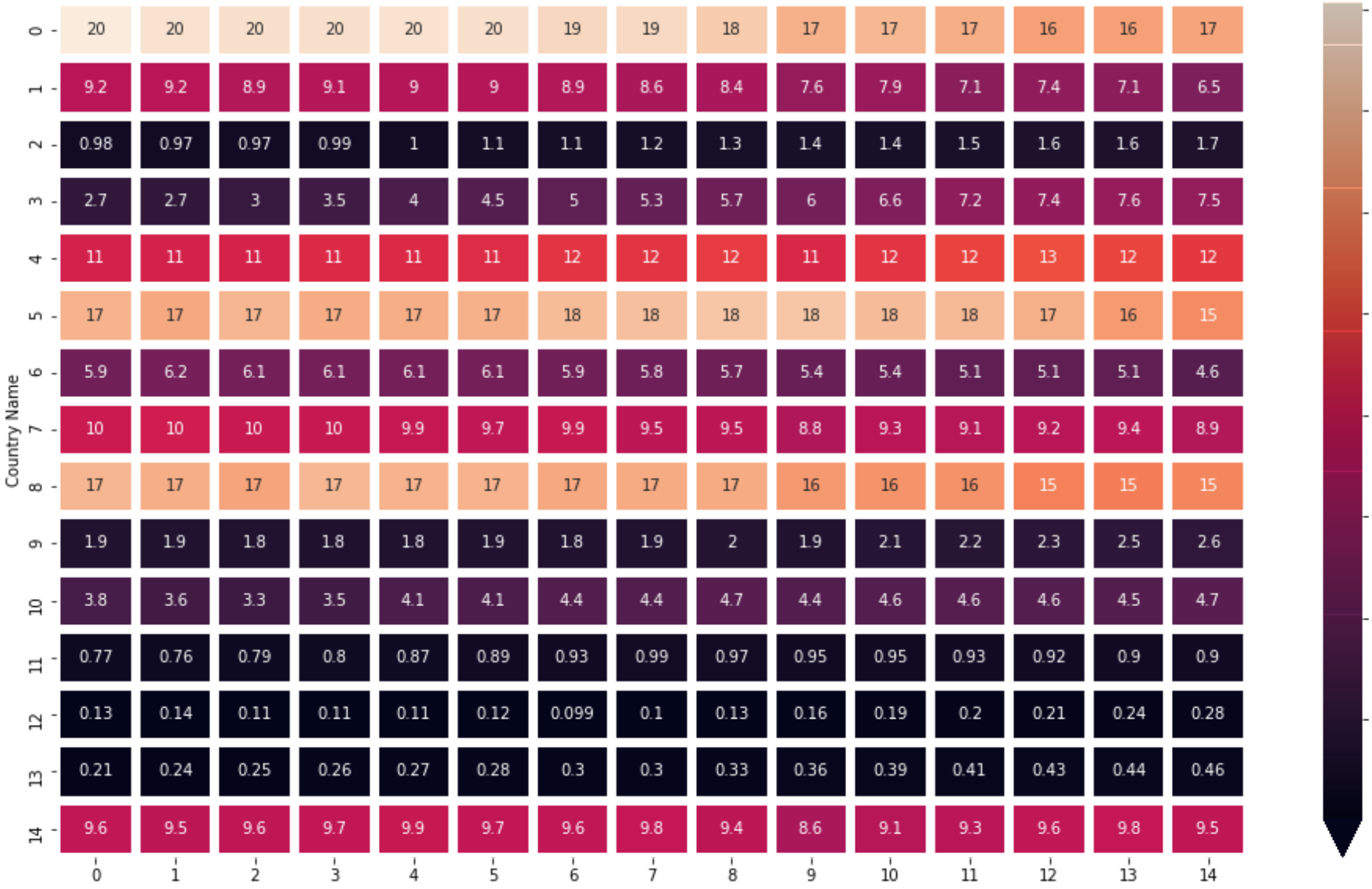
In [19]:
```
plt.figure(figsize=(17,10))

sns.heatmap(data1,annot=True,linewidths=5,cbar_kws={
    "orientation":"vertical",
#     "orientation":"horizontal",
    "shrink":1,
    "extend":"min", # min,max,both
    "extendfrac":0.1,
    "ticks":np.arange(0,22),
    "drawedges":True,
})
```

Out[19]:
```
<AxesSubplot:ylabel='Country Name'>
```


Country Name
United States 20 20 20 20 20 20 19 19 18 17 17 17 16 16 17
United Kingdom 9.2 9.2 8.9 9.1 9 9 8.9 8.6 8.4 7.6 7.9 7.1 7.4 7.1 6.5
India 0.98 0.97 0.97 0.99 1 1.1 1.1 1.2 1.3 1.4 1.4 1.5 1.6 1.6 1.7
China 2.7 2.7 3 3.5 4 4.5 5 5.3 5.7 6 6.6 7.2 7.4 7.6 7.5
Russian Federation 11 11 11 11 11 11 12 12 12 11 12 12 13 12 12
Australia 17 17 17 17 17 17 18 18 18 18 18 18 17 16 15
France 5.9 6.2 6.1 6.1 6.1 6.1 5.9 5.8 5.7 5.4 5.4 5.1 5.1 5.1 4.6
Germany 10 10 10 10 9.9 9.7 9.9 9.5 9.5 8.8 9.3 9.1 9.2 9.4 8.9
Canada 17 17 17 17 17 17 17 17 17 16 16 16 15 15 15
Brazil 1.9 1.9 1.8 1.8 1.8 1.9 1.8 1.9 2 1.9 2.1 2.2 2.3 2.5 2.6
Argentina 3.8 3.6 3.3 3.5 4.1 4.1 4.4 4.4 4.7 4.4 4.6 4.6 4.6 4.5 4.7
Pakistan 0.77 0.76 0.79 0.8 0.87 0.89 0.93 0.99 0.97 0.95 0.95 0.93 0.92 0.9 0.9
Nepal 0.13 0.14 0.11 0.11 0.11 0.12 0.099 0.1 0.13 0.16 0.19 0.2 0.21 0.24 0.28
Bangladesh 0.21 0.24 0.25 0.26 0.27 0.28 0.3 0.3 0.33 0.36 0.39 0.41 0.43 0.44 0.46
Japan 9.6 9.5 9.6 9.7 9.9 9.7 9.6 9.8 9.4 8.6 9.1 9.3 9.6 9.8 9.5
2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

In [20]:
```
plt.figure(figsize=(17,10))
#  yticklabels= # contry_lab=["a","b","c",.....] ap de sakte ho lebale to name
sns.heatmap(data1,annot=True,linewidths=5,xticklabels=np.arange(0,15),yticklabels=np.arange(0,15),cbar_kws=
```

Out[20]: <AxesSubplot:ylabel='Country Name'>



In [21]:
```
# # # plt.figure(figsize=(17,10))

# ax=sns.heatmap(data1, cmap="coolwarm",annot=True,linewidths=5)
# ax.set(title="vasim shaikh first heatmap ",
#       xlabel="co2 eemmistion ",
#       ylabel="Country name")
# sns.set(font_scale=3)
```

Correlation heatmap

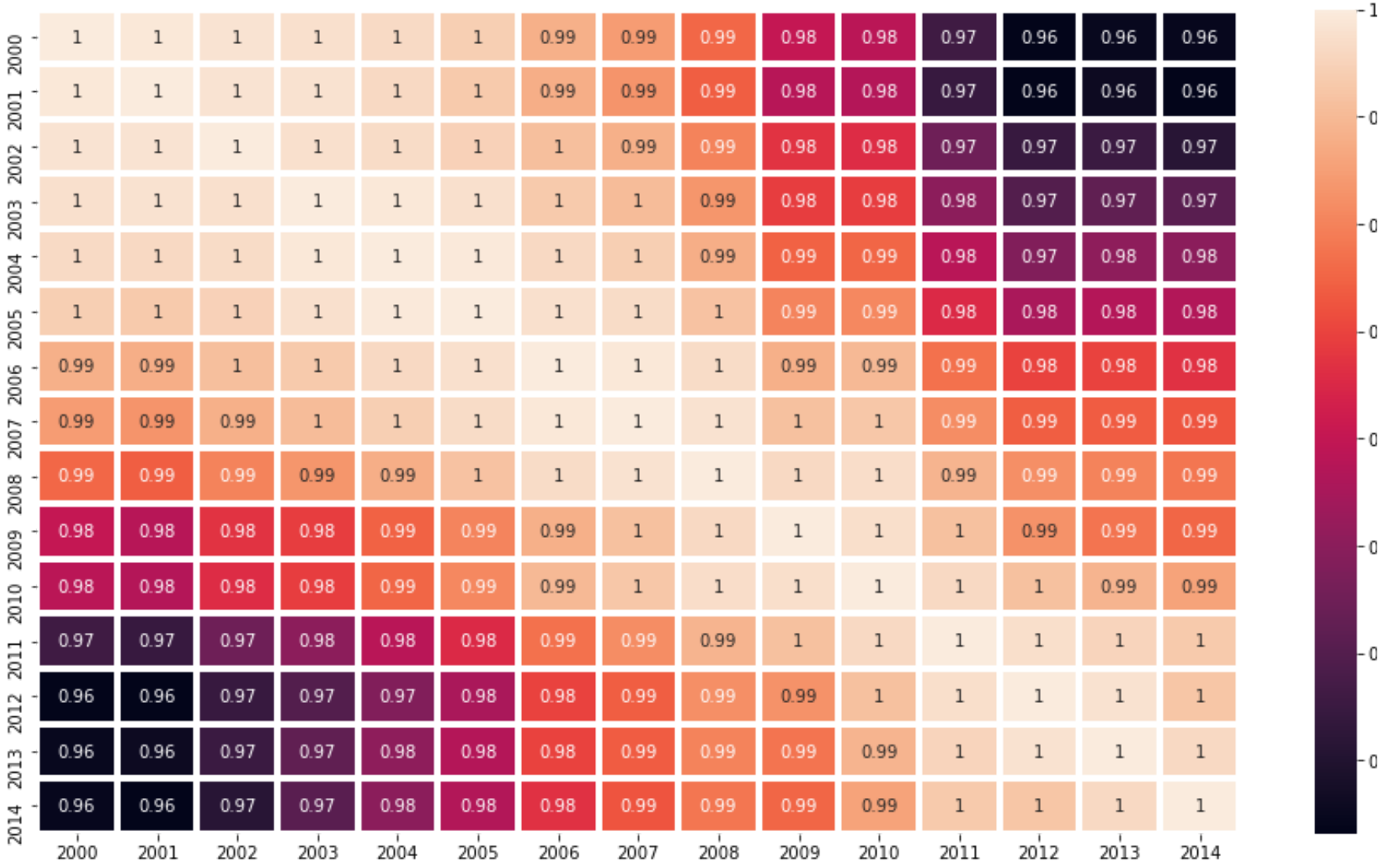
In [22]:
```
data1.corr()
```

Out[22]:

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	201
2000	1.000000	0.999632	0.999155	0.998911	0.998314	0.997008	0.994087	0.992283	0.987767	0.980143	0.979172	0.967887	0.96158
2001	0.999632	1.000000	0.999229	0.999026	0.998095	0.996628	0.993860	0.991532	0.987057	0.978912	0.978562	0.967206	0.96162
2002	0.999155	0.999229	1.000000	0.998907	0.998399	0.997391	0.995643	0.994017	0.990034	0.983584	0.982944	0.972479	0.96716
2003	0.998911	0.999026	0.998907	1.000000	0.999568	0.998887	0.996614	0.995277	0.991681	0.984511	0.984466	0.975128	0.96991
2004	0.998314	0.998095	0.998399	0.999568	1.000000	0.999701	0.998105	0.997144	0.993891	0.987300	0.987668	0.979061	0.97409
2005	0.997008	0.996628	0.997391	0.998887	0.999701	1.000000	0.998942	0.998420	0.995803	0.990125	0.990498	0.982646	0.97775
2006	0.994087	0.993860	0.995643	0.996614	0.998105	0.998942	1.000000	0.999570	0.998415	0.994104	0.994985	0.988553	0.98489
2007	0.992283	0.991532	0.994017	0.995277	0.997144	0.998420	0.999570	1.000000	0.999088	0.995724	0.996367	0.990928	0.98697
2008	0.987767	0.987057	0.990034	0.991681	0.993891	0.995803	0.998415	0.999088	1.000000	0.998145	0.998539	0.994593	0.99112
2009	0.980143	0.978912	0.983584	0.984511	0.987300	0.990125	0.994104	0.995724	0.998145	1.000000	0.998722	0.995657	0.99138
2010	0.979172	0.978562	0.982944	0.984466	0.987668	0.990498	0.994985	0.996367	0.998539	0.998722	1.000000	0.998182	0.99578
2011	0.967887	0.967206	0.972479	0.975128	0.979061	0.982646	0.988553	0.990928	0.994593	0.995657	0.998182	1.000000	0.99877
2012	0.961582	0.961625	0.967161	0.969919	0.974094	0.977758	0.984892	0.986978	0.991128	0.991382	0.995782	0.998778	1.00000
2013	0.962466	0.962827	0.967573	0.971053	0.975276	0.978611	0.984857	0.986819	0.989983	0.988844	0.994553	0.997744	0.99906
2014	0.962331	0.961622	0.965665	0.970508	0.975061	0.978521	0.983371	0.986199	0.988927	0.987571	0.992817	0.996681	0.99613

In [28]:
```
plt.figure(figsize=(16,9))
sns.heatmap(data1.corr(),annot=True,linewidth=3)
```

Out[28]: <AxesSubplot:>



In [29]:
```
plt.figure(figsize=(16,9))
ax=sns.heatmap(data1.corr(),annot=True,linewidth=3)
ax.tick_params(size=10,color="w",labelsize=10,labelcolor="w")
plt.title("heatmap using seaborn", fontsize=15)
plt.show()
```

In [30]:
```
from sklearn.datasets import load_breast_cancer
canncer_dataset=load_breast_cancer()
```

In [31]:
```
canncer_dataset
```

Out[31]:
```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
        ...,
        [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
        0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
        1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
        1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
```

```
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
 'frame': None,
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnostic) dataset\n--------------------
---------------------\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 569\n\n    :Number of A
ibutes: 30 numeric, predictive attributes and the class\n\n    :Attribute Information:\n        - radius (m
of distances from center to points on the perimeter)\n        - texture (standard deviation of gray-scale v
es)\n        - perimeter\n        - area\n        - smoothness (local variation in radius lengths)\n
compactness (perimeter^2 / area - 1.0)\n        - concavity (severity of concave portions of the contour)\n
- concave points (number of concave portions of the contour)\n        - symmetry\n        - fractal dimensi
("coastline approximation" - 1)\n\n        The mean, standard error, and "worst" or largest (mean of the th
\n        worst/largest values) of these features were computed for each image,\n        resulting in 30 fe
res.  For instance, field 0 is Mean Radius, field\n        10 is Radius SE, field 20 is Worst Radius.\n\n
- class:\n                - WDBC-Malignant\n                - WDBC-Benign\n\n    :Summary Statistics:\n\n
================================== ====== ======\n                                        Min    Max\n
================================== ====== ======\n    radius (mean):                      6.981  28.11
texture (mean):                    9.71   39.28\n    perimeter (mean):                   43.79  188.5\
area (mean):                       143.5  2501.0\n    smoothness (mean):                  0.053  0.163
compactness (mean):                0.019  0.345\n    concavity (mean):                   0.0    0.427\
concave points (mean):             0.0    0.201\n    symmetry (mean):                    0.106  0.304\
fractal dimension (mean):          0.05   0.097\n    radius (standard error):            0.112  2.873\
texture (standard error):          0.36   4.885\n    perimeter (standard error):         0.757  21.98\
area (standard error):             6.802  542.2\n    smoothness (standard error):        0.002  0.031\
compactness (standard error):      0.002  0.135\n    concavity (standard error):         0.0    0.396\
concave points (standard error):   0.0    0.053\n    symmetry (standard error):          0.008  0.079\
fractal dimension (standard error): 0.001  0.03\n    radius (worst):                    7.93   36.04\n
texture (worst):                   12.02  49.54\n    perimeter (worst):                  50.41  251.2\
area (worst):                      185.2  4254.0\n    smoothness (worst):                 0.071  0.223
compactness (worst):               0.027  1.058\n    concavity (worst):                  0.0    1.252\
concave points (worst):            0.0    0.291\n    symmetry (worst):                   0.156  0.664\
fractal dimension (worst):         0.055  0.208\n    ================================== ====== ======
\n    :Missing Attribute Values: None\n\n    :Class Distribution: 212 - Malignant, 357 - Benign\n\n    :Cre
r:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian\n\n    :Donor: Nick Street\n\n    :Date: No
ber, 1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.\nhttps://goo.gl/U2Uwz2
\nFeatures are computed from a digitized image of a fine needle\naspirate (FNA) of a breast mass.  They des
be\ncharacteristics of the cell nuclei present in the image.\n\nSeparating plane described above was obtain
using\nMultisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via Linear Programming
```

```
Proceedings of the 4th\nMidwest Artificial Intelligence and Cognitive Science Society,\npp. 97-101, 1992],
lassification method which uses linear\nprogramming to construct a decision tree.  Relevant features\nwere
ected using an exhaustive search in the space of 1-4\nfeatures and 1-3 separating planes.\n\nThe actual lin
program used to obtain the separating plane\nin the 3-dimensional space is that described in:\n[K. P. Benne
and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Two Linearly Inseparable Sets",\nOptim
tion Methods and Software 1, 1992, 23-34].\n\nThis database is also available through the UW CS ftp server:
\nftp ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. topic:: References\n\n   - W.N. S
et, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction \n     for breast tumor diagnosis. IS&T/S
1993 International Symposium on \n     Electronic Imaging: Science and Technology, volume 1905, pages 861-8
0,\n     San Jose, CA, 1993.\n   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis
\n     prognosis via linear programming. Operations Research, 43(4), pages 570-577, \n     July-August 1995
- W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques\n     to diagnose breast can
from fine-needle aspirates. Cancer Letters 77 (1994) \n     163-171.',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
 'filename': 'c:\\python3.8.3\\lib\\site-packages\\sklearn\\datasets\\data\\breast_cancer.csv'}
```

In [42]:
```
canner_df = pd.DataFrame(np.c_[canncer_dataset["data"],canncer_dataset["target"]],
                         columns=np.append(canncer_dataset["feature_names"],["targget"]))
canner_df
```

Out[42]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	17.33	184.60	2
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.80	1
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.50	1
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.87	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.20	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10	2
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00	1
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70	1
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60	1
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.16	

569 rows × 31 columns

In [47]:
```
plt.figure(figsize=(30,30))
ax=sns.heatmap(canner_df.corr(),annot=True,linewidth=3)
ax.tick_params(size=10,color="w",labelsize=10,labelcolor="w")
plt.title("canner_data using seaborn", fontsize=15)
plt.show()
```