In [1]:
```
# setup notebook
# notebook formatting
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))

# pretty print all cell's output and not just the last one
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [2]:
```
import time
import numpy as np
# imports
import pandas as pd

import bokeh
from bokeh.layouts import gridplot
from bokeh.plotting import output_notebook
output_notebook() # set default; alternative is output_file()

import holoviews as hv
import streamz
import streamz.dataframe

from holoviews import opts
from holoviews.streams import Pipe, Buffer

hv.extension('bokeh')
```

(https://bokeh.org)

In [3]:
```
# usage of watermark lib to show env and versions
%load_ext watermark
%watermark -v -p jupyter,numpy,pandas,bokeh
```

```
CPython 3.7.5
IPython 7.9.0

jupyter 1.0.0
numpy 1.17.4
pandas 0.25.3
bokeh 1.4.0
```

In [4]:
```
# download bokeh sample data - will download to $HOME/.bokeh/data - and create directories if ne
cessary
bokeh.sampledata.download()
```

```
Using data directory: C:\Users\chris\.bokeh\data
Downloading: CGM.csv (1589982 bytes)
   1589982 [100.00%]
Downloading: US_Counties.zip (3171836 bytes)
   3171836 [100.00%]
Unpacking: US_Counties.csv
Downloading: us_cities.json (713565 bytes)
    713565 [100.00%]
Downloading: unemployment09.csv (253301 bytes)
    253301 [100.00%]
Downloading: AAPL.csv (166698 bytes)
    166698 [100.00%]
Downloading: FB.csv (9706 bytes)
      9706 [100.00%]
Downloading: GOOG.csv (113894 bytes)
    113894 [100.00%]
Downloading: IBM.csv (165625 bytes)
    165625 [100.00%]
Downloading: MSFT.csv (161614 bytes)
    161614 [100.00%]
Downloading: WPP2012_SA_DB03_POPULATION_QUINQUENNIAL.zip (4816256 bytes)
   4816256 [100.00%]
Unpacking: WPP2012_SA_DB03_POPULATION_QUINQUENNIAL.csv
Downloading: gapminder_fertility.csv (64346 bytes)
     64346 [100.00%]
Downloading: gapminder_population.csv (94509 bytes)
     94509 [100.00%]
Downloading: gapminder_life_expectancy.csv (73243 bytes)
     73243 [100.00%]
Downloading: gapminder_regions.csv (7781 bytes)
      7781 [100.00%]
Downloading: world_cities.zip (645274 bytes)
    645274 [100.00%]
Unpacking: world_cities.csv
Downloading: airports.json (6373 bytes)
      6373 [100.00%]
Downloading: movies.db.zip (5053420 bytes)
   5053420 [100.00%]
Unpacking: movies.db
Downloading: airports.csv (203190 bytes)
    203190 [100.00%]
Downloading: routes.csv (377280 bytes)
    377280 [100.00%]
```

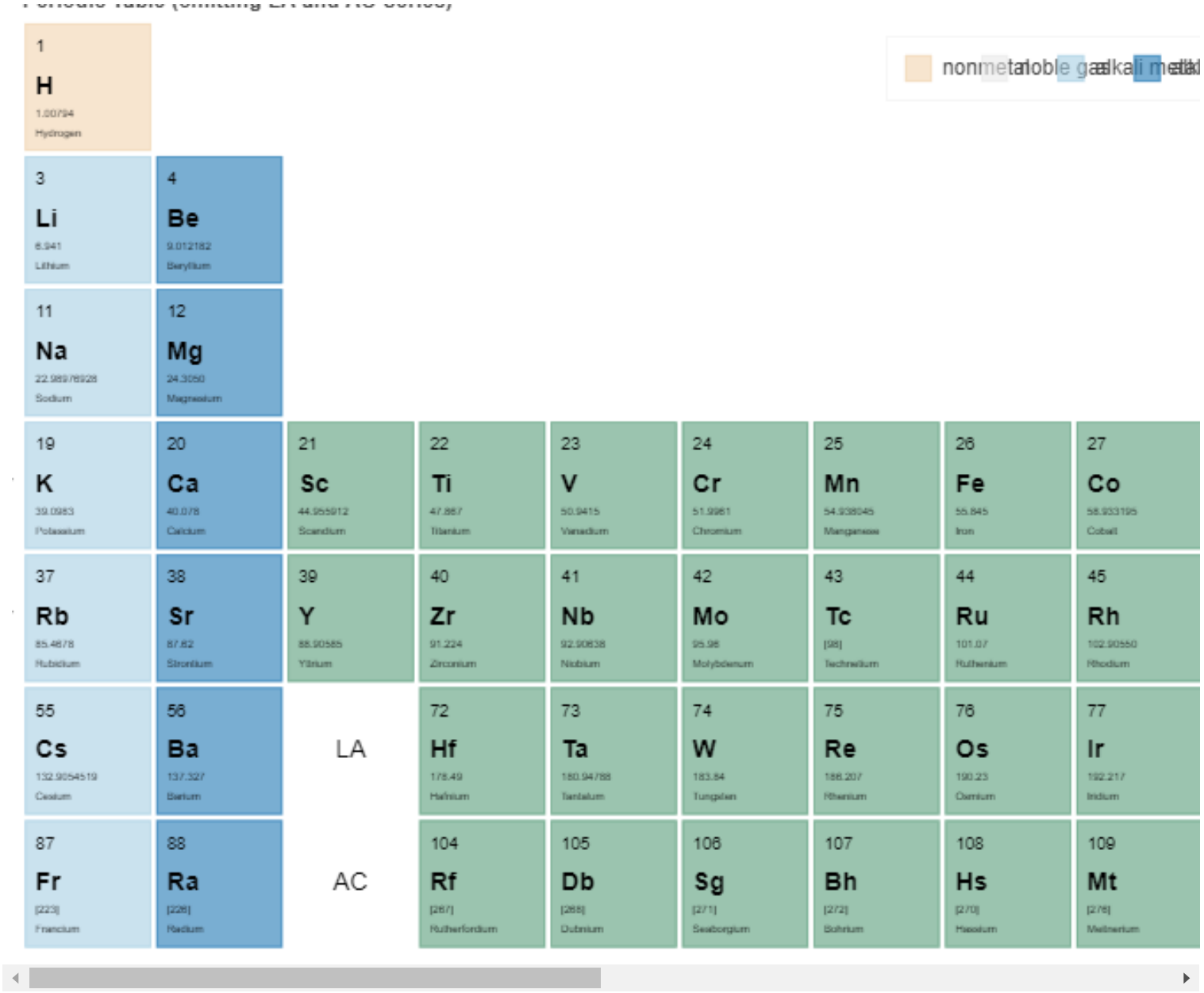
```
Downloading: haarcascade_frontalface_default.xml (930127 bytes)
    930127 [100.00%]
```

In [5]:
```
# example: periodic table
# credit:  http://docs.bokeh.org/en/1.4.0/docs/user_guide/categorical.html#userguide-categorical

from bokeh.io import output_file, show
from bokeh.models import ColumnDataSource
from bokeh.plotting import figure
from bokeh.sampledata.periodic_table import elements
from bokeh.transform import dodge, factor_cmap

# output_file("periodic.html")

periods = ["I", "II", "III", "IV", "V", "VI", "VII"]
groups = [str(x) for x in range(1, 19)]

df = elements.copy()
df["atomic mass"] = df["atomic mass"].astype(str)
df["group"] = df["group"].astype(str)
df["period"] = [periods[x-1] for x in df.period]
df = df[df.group != "-"]
df = df[df.symbol != "Lr"]
df = df[df.symbol != "Lu"]

cmap = {
    "alkali metal"         : "#a6cee3",
    "alkaline earth metal" : "#1f78b4",
    "metal"                : "#d93b43",
    "halogen"              : "#999d9a",
    "metalloid"            : "#e08d49",
    "noble gas"            : "#eaeaea",
    "nonmetal"             : "#f1d4Af",
    "transition metal"     : "#599d7A",
}

source = ColumnDataSource(df)

p = figure(plot_width=1500, title="Periodic Table (omitting LA and AC Series)",
           x_range=groups, y_range=list(reversed(periods)), toolbar_location=None, tools="hover"
)

p.rect("group", "period", 0.95, 0.95, source=source, fill_alpha=0.6, legend_field="metal",
       color=factor_cmap('metal', palette=list(cmap.values()), factors=list(cmap.keys())))
```

```
text_props = {"source": source, "text_align": "left", "text_baseline": "middle"}

x = dodge("group", -0.4, range=p.x_range)

r = p.text(x=x, y="period", text="symbol", **text_props)
r.glyph.text_font_style="bold"

r = p.text(x=x, y=dodge("period", 0.3, range=p.y_range), text="atomic number", **text_props)
r.glyph.text_font_size="8pt"

r = p.text(x=x, y=dodge("period", -0.35, range=p.y_range), text="name", **text_props)
r.glyph.text_font_size="5pt"

r = p.text(x=x, y=dodge("period", -0.2, range=p.y_range), text="atomic mass", **text_props)
r.glyph.text_font_size="5pt"

p.text(x=["3", "3"], y=["VI", "VII"], text=["LA", "AC"], text_align="center", text_baseline="mid
dle")

p.hover.tooltips = [
    ("Name", "@name"),
    ("Atomic number", "@{atomic number}"),
    ("Atomic mass", "@{atomic mass}"),
    ("Type", "@metal"),
    ("CPK color", "$color[hex, swatch]:CPK"),
    ("Electronic configuration", "@{electronic configuration}"),
]

p.outline_line_color = None
p.grid.grid_line_color = None
p.axis.axis_line_color = None
p.axis.major_tick_line_color = None
p.axis.major_label_standoff = 0
p.legend.orientation = "horizontal"
p.legend.location ="top_center"

show(p)
```

Out[5]: **GlyphRenderer**(id = '1029', …)

Out[5]: **GlyphRenderer**(id = '1067', …)



Bokeh glyphs (https://bokeh.pydata.org/en/latest/docs/reference/models/markers.html)

Asterisk

Circle

CircleCross

CircleX

Cross

Dash

Diamond

DiamondCross

Hex

InvertedTriangle

Square

SquareCross

SquareX

Triangle

X

Example - Sankey diagram

using data from holoviz example - on energy

Sankey diagrams are flow diagrams that show relative contributions of entities

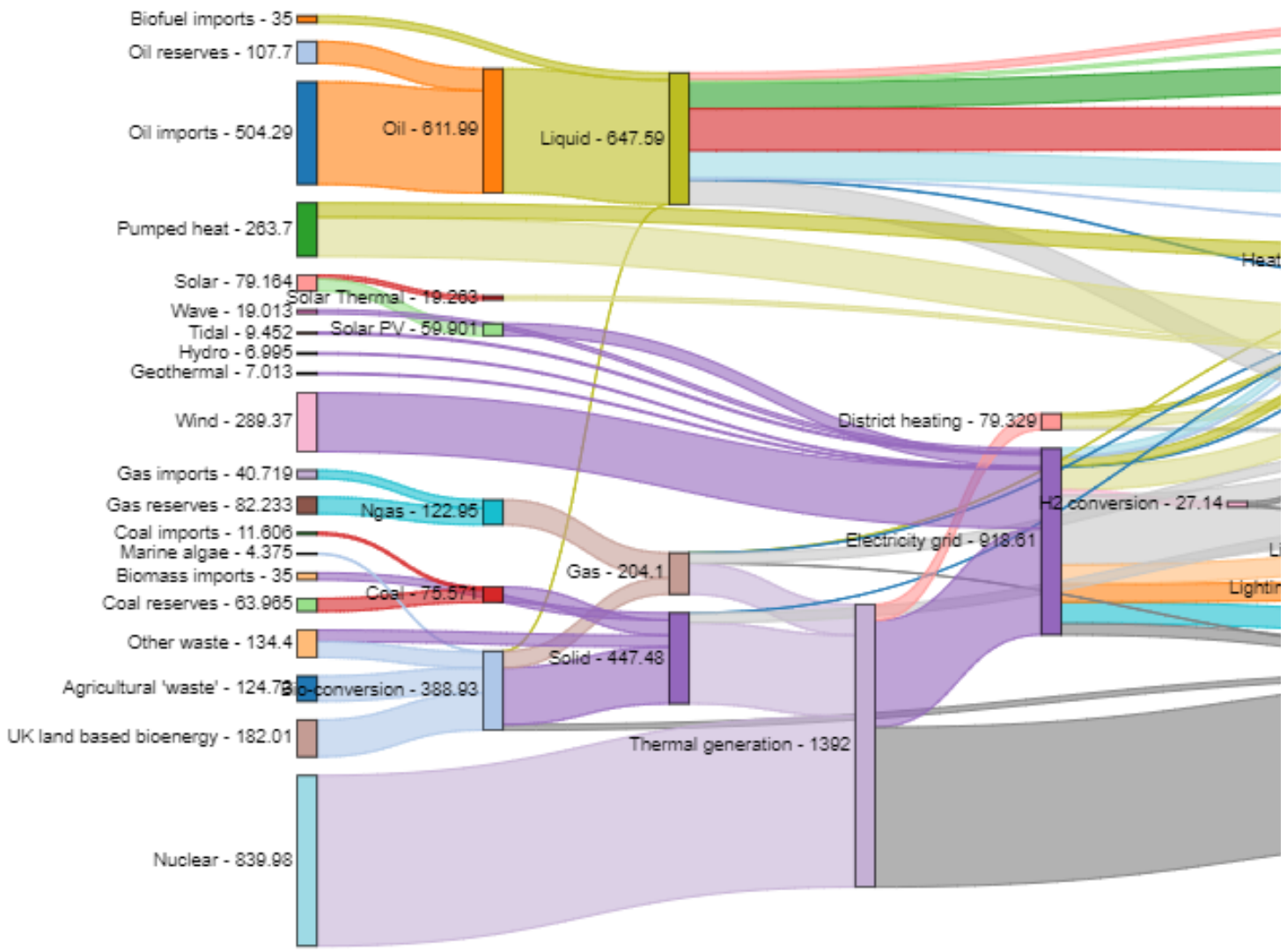
In [6]:
```
# data from github must be raw url
data_url = "https://raw.githubusercontent.com/holoviz/holoviews/master/examples/assets/energy.csv"
edges = pd.read_csv(data_url)
edges.head(5)
```

Out[6]:

	source	target	value
0	Agricultural 'waste'	Bio-conversion	124.729
1	Bio-conversion	Liquid	0.597
2	Bio-conversion	Losses	26.862
3	Bio-conversion	Solid	280.322
4	Bio-conversion	Gas	81.144

In [7]:
```
sankey = hv.Sankey(edges, label='Energy Diagram')
sankey.opts(label_position='left', edge_color='target', node_color='index', cmap='tab20')
```

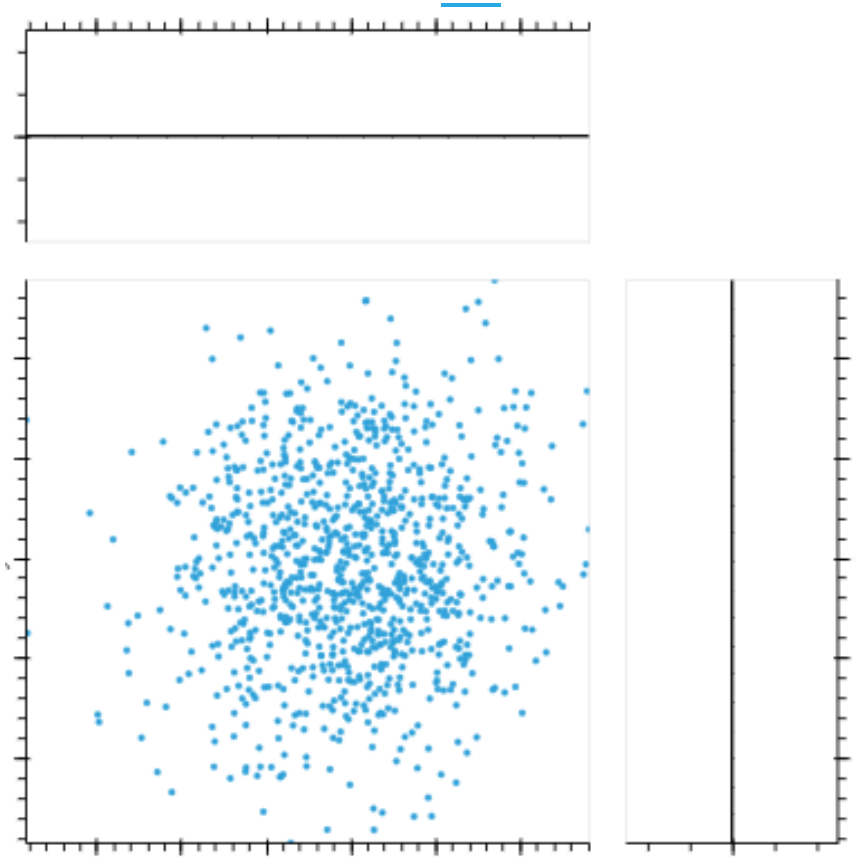
Out[7]:



In [8]:
```
import numpy as np
import holoviews as hv
from holoviews import opts
from holoviews import streams
hv.extension('bokeh')


opts.defaults(opts.Histogram(framewise=True))

# Declare distribution of Points
points = hv.Points(np.random.multivariate_normal((0, 0), [[1, 0.1], [0.1, 1]], (1000,)))

# Declare points selection selection
sel = streams.Selection1D(source=points)

# Declare DynamicMap computing mean y-value of selection
mean_sel = hv.DynamicMap(lambda index: hv.HLine(points['y'][index].mean() if index else -10),
                         kdims=[], streams=[sel])

# Declare a Bounds stream and DynamicMap to get box_select geometry and draw it
box = streams.BoundsXY(source=points, bounds=(0,0,0,0))
bounds = hv.DynamicMap(lambda bounds: hv.Bounds(bounds), streams=[box])

# Declare DynamicMap to apply bounds selection
dmap = hv.DynamicMap(lambda bounds: points.select(x=(bounds[0], bounds[2]),
                                                  y=(bounds[1], bounds[3])),
                     streams=[box])

# Compute histograms of selection along x-axis and y-axis
yhist = hv.operation.histogram(dmap, bin_range=points.range('y'), dimension='y', dynamic=True, n
ormed=False)
xhist = hv.operation.histogram(dmap, bin_range=points.range('x'), dimension='x', dynamic=True, n
ormed=False)

# Combine components and display
points * mean_sel * bounds << yhist << xhist
```

Out[8]:

(https://bokeh.org/)

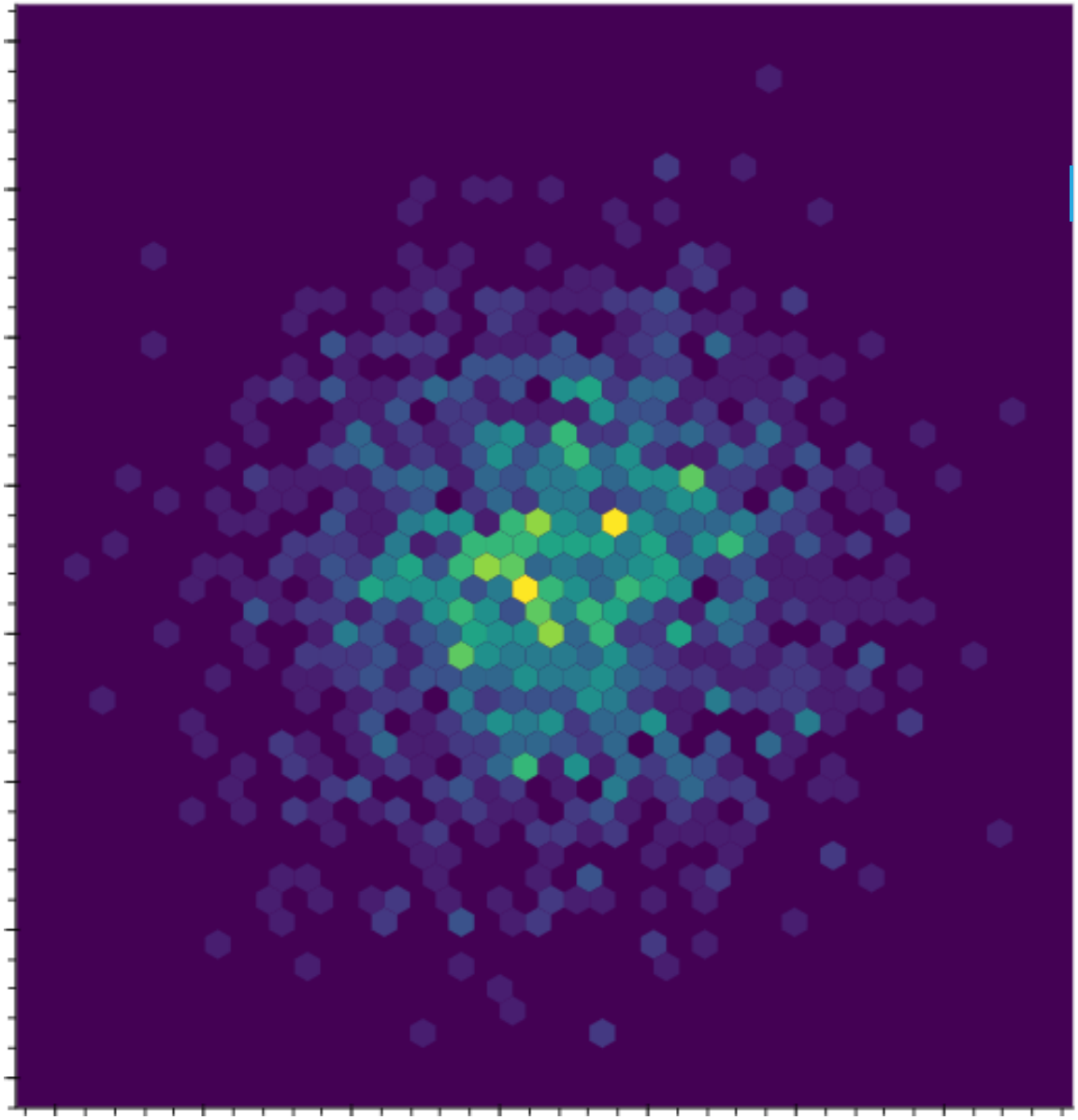


In [9]:
```
import numpy as np
from bokeh.models import HoverTool
from bokeh.plotting import figure, show

x = 2 + 5*np.random.standard_normal(1500)
y = 2 + 5*np.random.standard_normal(1500)

p = figure(match_aspect=True, tools="wheel_zoom,reset")
p.background_fill_color = '#440154'
p.grid.visible = False

p.hexbin(x, y, size=0.5, hover_color="pink", hover_alpha=0.8)

hover = HoverTool(tooltips=[("count", "@c"), ("(q,r)", "(@q, @r)")])
p.add_tools(hover)

show(p)
```

Out[9]: (GlyphRenderer(id='2152', ...),

	q	r	counts
0	-18	14	1
1	-17	3	1
2	-16	8	1
3	-15	-3	1
4	-15	12	1
..	..	..	...
568	21	-9	1
569	21	-7	1
570	22	-19	1
571	23	-25	1
572	25	-10	1

[573 rows x 3 columns])



(http

In [ ]: