Author: Anton Dementyev (2023638)
CSCM37 Coursework 1: Information Visualization

The growth of pleiades over the years (10000BC - 2100AD)

There are 3 data files from Pleiades. For this series of visualizations we use the Pleaides Locations data set. For simplicity, we have renamed it to 'pleiades.csv'.

Setup

```
import altair as alt
import pandas as pd
from vega_datasets import data
```

```
mp = data.world_110m.url
```

```
df = pd.read_csv('pleiades.csv')
```

Getting started

What can we learn from the visualization?

The goal of this visualization is to show the dynamics of the growth of the pleiades over the 12000-year period. For the date of emergence we take the minimal date (minDate) mentioned in the data set. As a result, we can see how the map is being filled with new locations upon interaction with time.

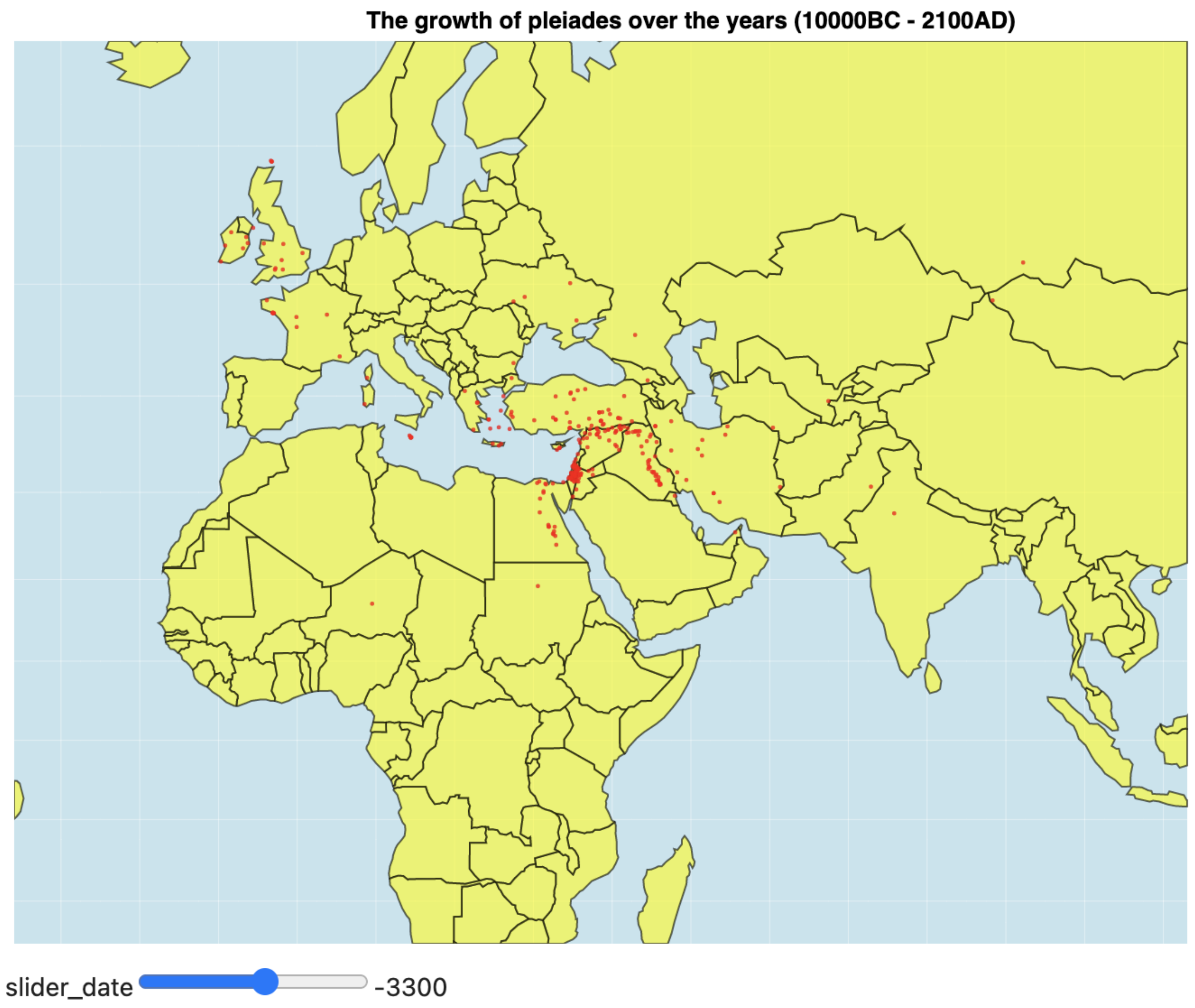
What is the name for the type of visualization(s) used?

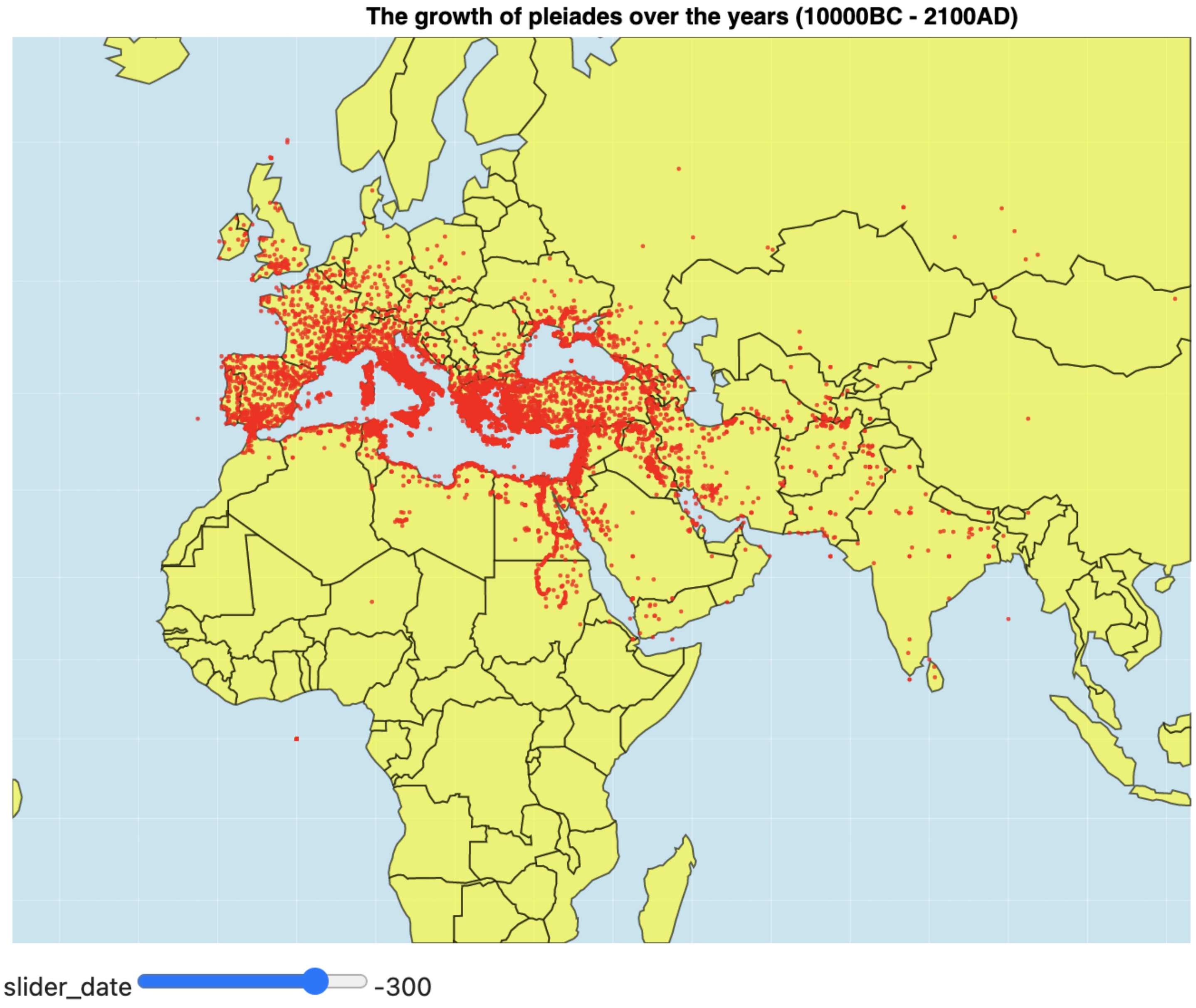
Geograhic map

Altair code

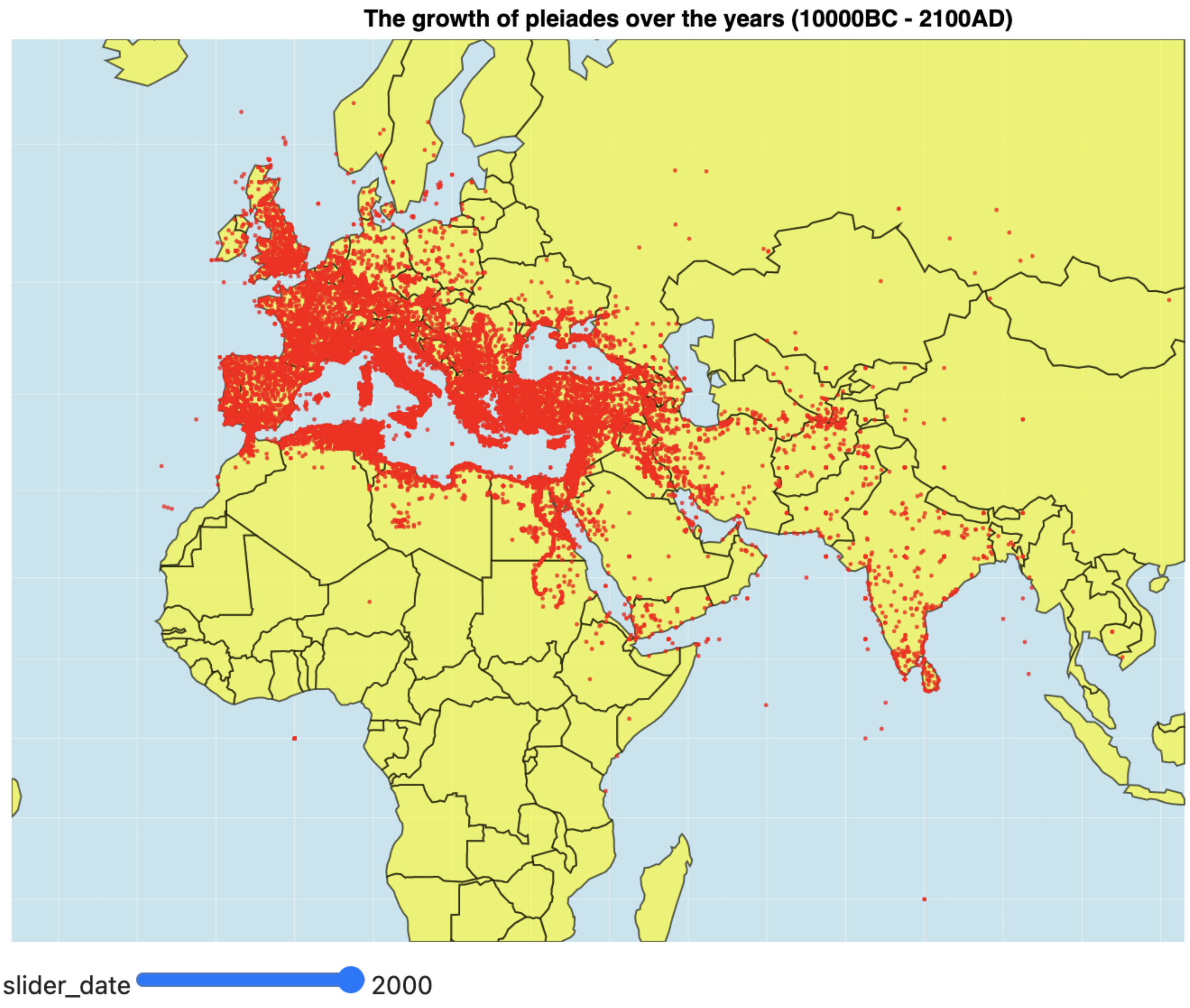
```python
alt.data_transformers.disable_max_rows()
alt.renderers.enable('html')

slider = alt.binding_range(
    step=100,
    min=-10000,
    max=df.maxDate.max()
)

select_date = alt.selection_single(
    name="slider",
    fields=['date'],
    bind=slider,
    init={"date": -10000}
)

sphere = alt.sphere()
graticule = alt.graticule()

source = alt.topo_feature(mp, 'countries')
data = alt.Chart(df).mark_circle(size=5, color='red').encode(
    longitude='reprLong:Q',
    latitude='reprLat:Q',
).add_selection(select_date).transform_filter(
    "datum.minDate <= slider.date[0]"
)

chart = alt.layer(
    alt.Chart(sphere).mark_geoshape(fill='lightblue', opacity=0.7),
    alt.Chart(graticule).mark_geoshape(stroke='white', strokeWidth=0.3, opacity=0.8),
    alt.Chart(source).mark_geoshape(fill='yellow', stroke='black', opacity=0.6),
    data
).project(
    'mercator',
    scale=250,
    center=[50, 30],
    clipExtent=[[0, 0], [650, 500]],
).properties(
    width=750,
    height=500,
    title='The growth of pleiades over the years (10000BC - 2100AD)'
).configure_view(stroke=None)


chart
```

Examples & interaction


The growth of pleiades over the years (10000BC - 2100AD)
slider_date
-3300


The growth of pleiades over the years (10000BC - 2100AD)

slider_date -300



What are all visual mappings used?

x position

reprLong - longitude

y position

reprLat - latitude

Was there any special data preparation done?

Cleaning from null values.

```
df = df[df.timePeriodsRange.notnull()]
```

What are the limitations of your design?

It is very memory consuming, since each point has to be rendered individually. However, merging points which are close to each other into one does not solve the issue due to unstable size. A solution could be to use a choropleth map and it is used further for one of the followup designs.

Pleiades and their connections across the UK

Setup

```
import altair as alt
import pandas as pd
from vega_datasets import data
```

```
mp = data.world_110m.url
```

```
df = pd.read_csv('pleiades.csv')
```

What can we learn from the visualization?

This visualization enables studying the connections of the Pleaides locations across the UK. We believe that with the aid of this visualization people should be able to retrace the links and origin of a majority of the locations. Exact coordinates of the locations are provided in the top left corner and a tooltip appears upon hovering over a location to fulfill "details on demand". This help reduce visual clutter and allows to point out interesting connections based on the expanded info.

What is the name for the type of visualization(s) used?

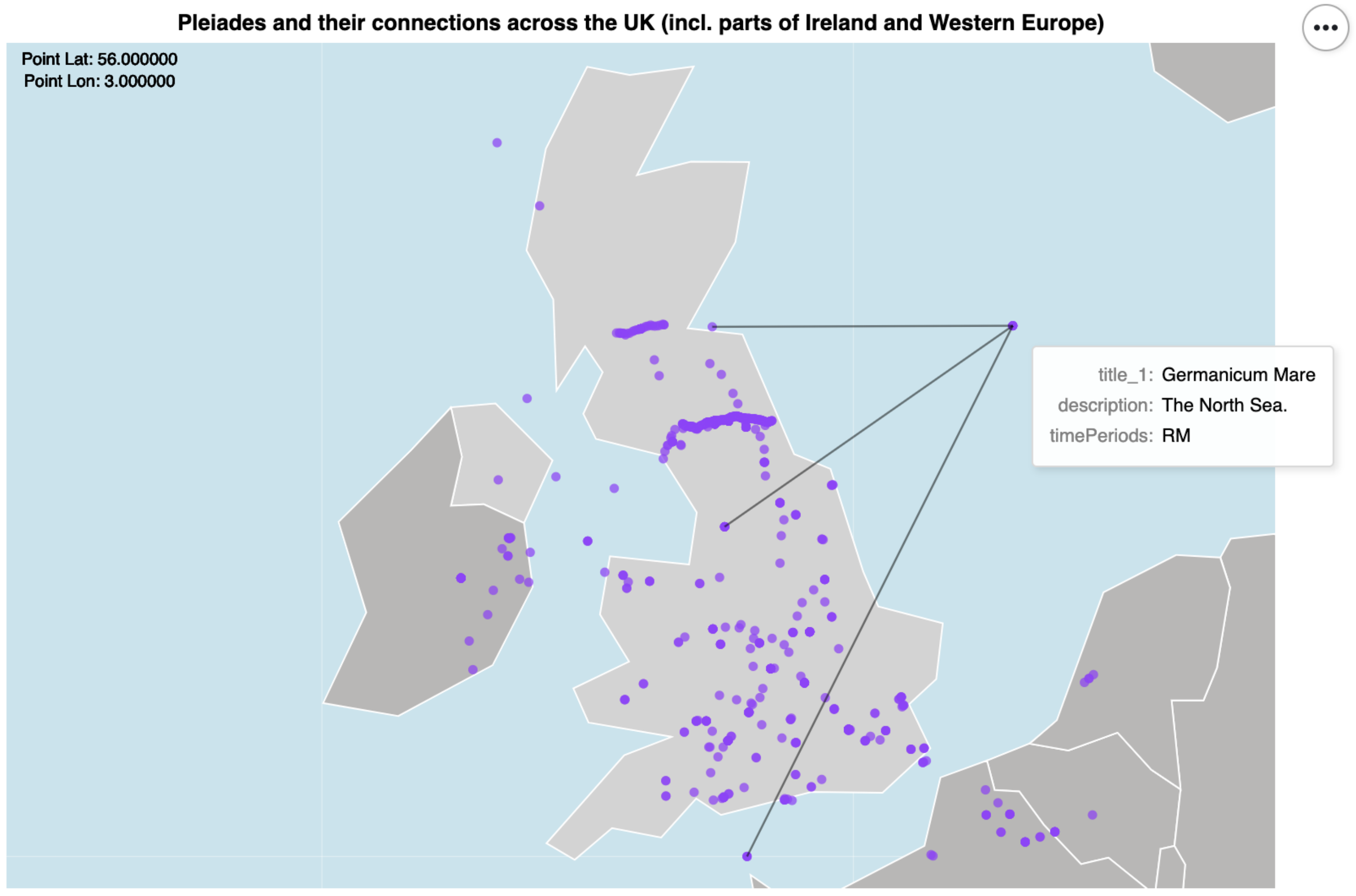
Geographic map

Altair code

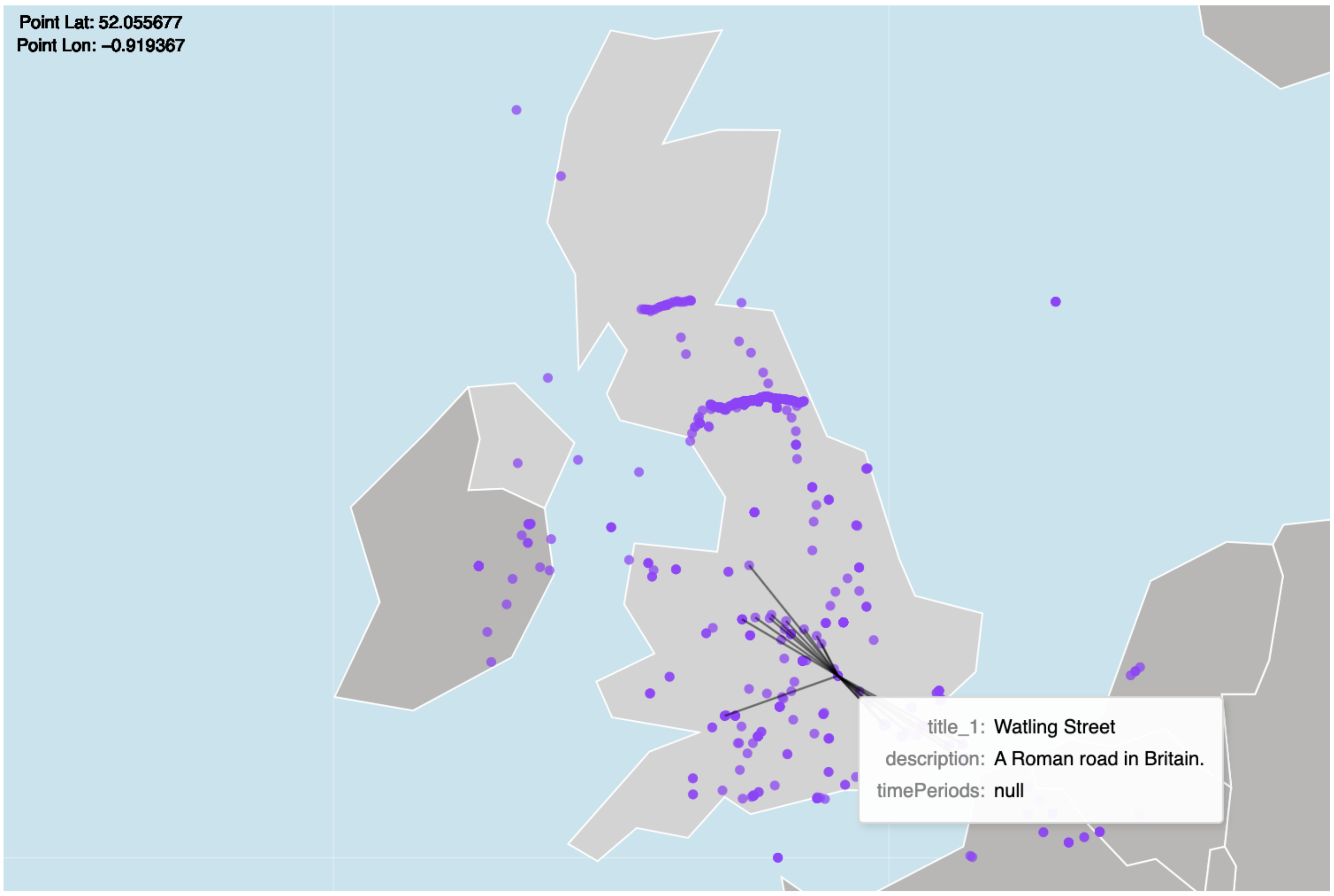
```
select_point = alt.selection_single(on="mouseover", nearest=True, fields=["origin"], emp

mp = data.world_110m.url
source = alt.topo_feature(mp, 'countries')

globe = alt.Chart(alt.sphere()).mark_geoshape(fill='lightblue', opacity=0.7)
meridian = alt.Chart(alt.graticule()).mark_geoshape(stroke='white', strokeWidth=0.3, opa

# the ID of the UK is 826 in "world_110m" => if not the UK, then apply a different colo
background = alt.Chart(source).mark_geoshape(
    stroke="white",
).encode(
    color=alt.condition(alt.datum.id == 826, alt.value('lightgray'), alt.value('#B8B7B6
)

# displaying latitude of a point (circle)
text_lat = alt.Chart(final).mark_text(dy=-227, dx=-320, size=10).encode(
    text='label:N',
).transform_filter(
    select_point
).transform_calculate(
    label=f'"Point Lon: " + format(datum.reprLong_1,".6f")'
)

# displaying longitude of a point (circle)
text_lon = alt.Chart(final).mark_text(dy=-240, dx=-320, size=10).encode(
    text='label:N',
).transform_filter(
    select_point
).transform_calculate(
    label=f'"Point Lat: " + format(datum.reprLat_1,".6f")'
)

# connecting points
connections = alt.Chart(final).mark_rule(opacity=0.5).encode(
    latitude="reprLat_1:Q",
    longitude="reprLong_1:Q",
    latitude2="reprLat_2:Q",
    longitude2="reprLong_2:Q",
    size=alt.value(1.2),
).transform_filter(
    select_point
)

points = alt.Chart(final).mark_circle(color='#9933FF', opacity=0.7, width=10).encode(
    latitude="reprLat_1:Q",
    longitude="reprLong_1:Q",
    tooltip=["title_1:N", "description:N", "timePeriods:N"]
).add_selection(
    select_point
)
```

```
chart = alt.layer(
    globe,
    meridian,
    background,
    text_lat,
    text_lon,
    points,
    connections
).properties(
    width=750,
    height=500,
    title='Pleiades and their connections across the UK (incl. parts of Ireland and Wes
).project(
    "mercator",
    scale=1800,
    center=[-4, 54.5],
).configure_view(stroke=None)


chart
```

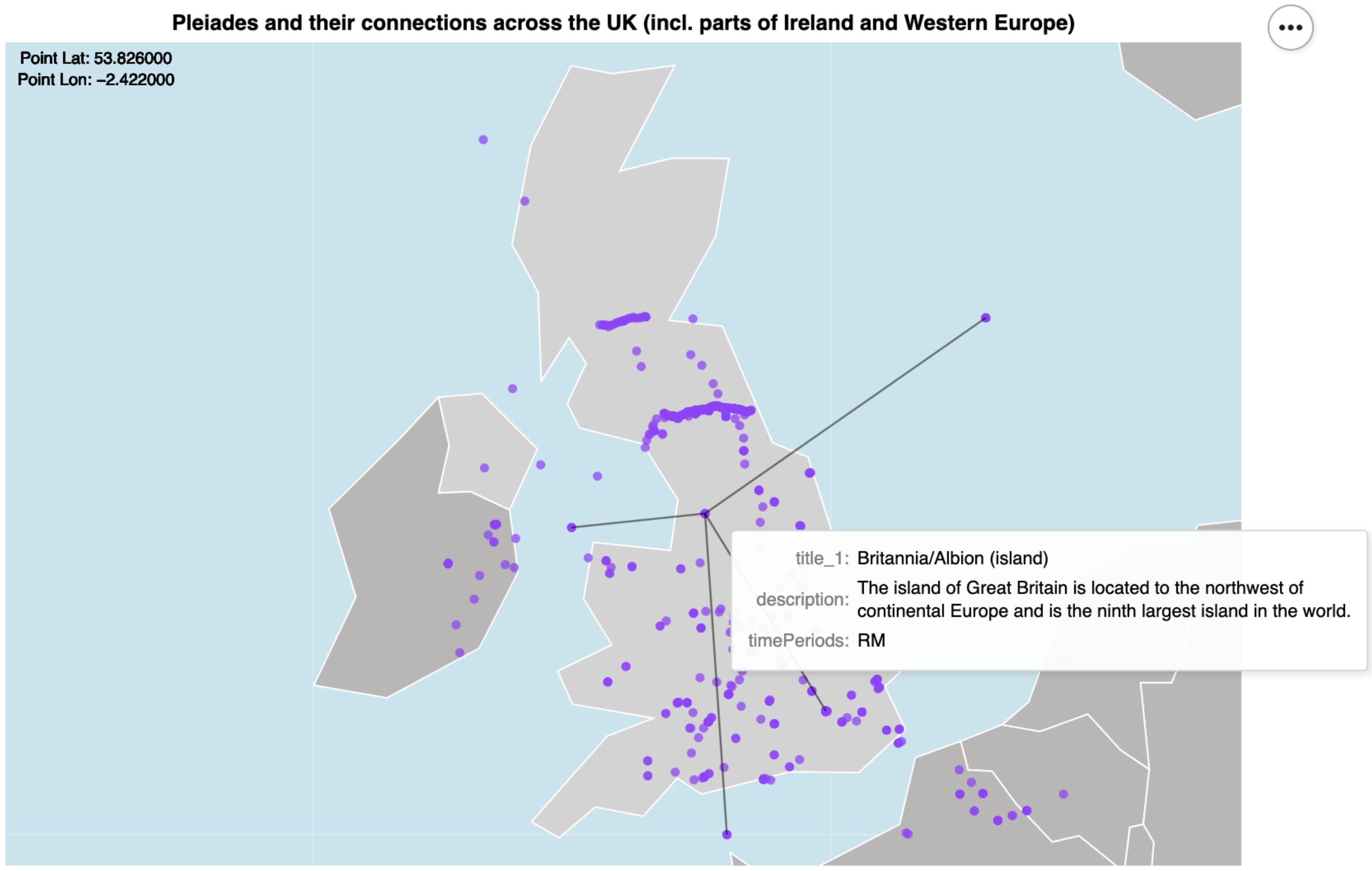
Examples & interaction


Pleiades and their connections across the UK (incl. parts of Ireland and Western Europe)
Point Lat: 56.000000
Point Lon: 3.000000
title_1: Germanicum Mare
description: The North Sea.
timePeriods: RM

Pleiades and their connections across the UK (incl. parts of Ireland and Western Europe)



title_1: Watling Street
description: A Roman road in Britain.
timePeriods: null


Pleiades and their connections across the UK (incl. parts of Ireland and Western Europe)
Point Lat: 53.826000
Point Lon: −2.422000
title_1: Britannia/Albion (island)
description: The island of Great Britain is located to the northwest of continental Europe and is the ninth largest island in the world.
timePeriods: RM

What are all visual mappings used?

x (x1, x2 for the lines) position

latitude of location

y (y1, y2 for the lines) position

longitude of location

Was there any special data preparation done?

Filtering to obtain only UK data (+ surrondings). Mapping connections.

```
# UK bounding rect
df = df[(df.reprLat > 49.959999905) & (df.reprLong > −7.57216793459) & (df.reprLat < 58.
# direct and backward connections
connections = []
for row in zip(df['id'], df['hasConnectionsWith'].astype(str)):
    spl = row[1].split(',')
    for x in spl:
        connections.append([row[0], x])

for row in zip(df['id'], df['connectsWith'].astype(str)):
    spl = row[1].split(',')
    for x in spl:
        connections.append([row[0], x])

connectionsDf = pd.DataFrame(connections, columns=["origin", "connection"])
```

```
connectionsDf[90:100]
```

	origin	connection
90	656280677	79595
91	437050931	nan
92	442436712	nan
93	452125610	592005824
94	82240559	nan
95	765102512	nan
96	504812865	nan
97	773739434	nan
98	747167617	nan
99	455597764	512867195

```
# left joining neccessary field to our lookup table
df2 = connectionsDf.merge(df[['id', 'title', 'reprLat', 'reprLong', 'description', 'time
df['id'] = df['id'].astype(str)
final = df2.merge(df[['id', 'title', 'reprLat', 'reprLong']], left_on='connection', rigl
```

What are the limitations of your design?

It can only be applied to areas with sparser distribution of points. This is, of course, unless map zooming is present, which is not the case in vega-lite.

Countries with most aqueducts (top 10)

Setup

```python
import altair as alt
import pandas as pd
import time
import requests
from vega_datasets import data
from geopy.geocoders import Nominatim
```

```python
df = pd.read_csv('pleiades.csv')
```

What can we learn from the visualization?

People travelling all over Southern Europe and its surrounding territories can often notice the remains of beautiful arch-shaped (not always) bridges called aqueducts, some of which are part of the pleiades. The utlimate goal of this visualization is to present countries with the most of such. This can help indentify the spread of the aqueducts beyond the well-know locations of their centralization (e.g., Italy, Greece) and enable comparison between different countries.

What is the name for the type of visualization(s) used?

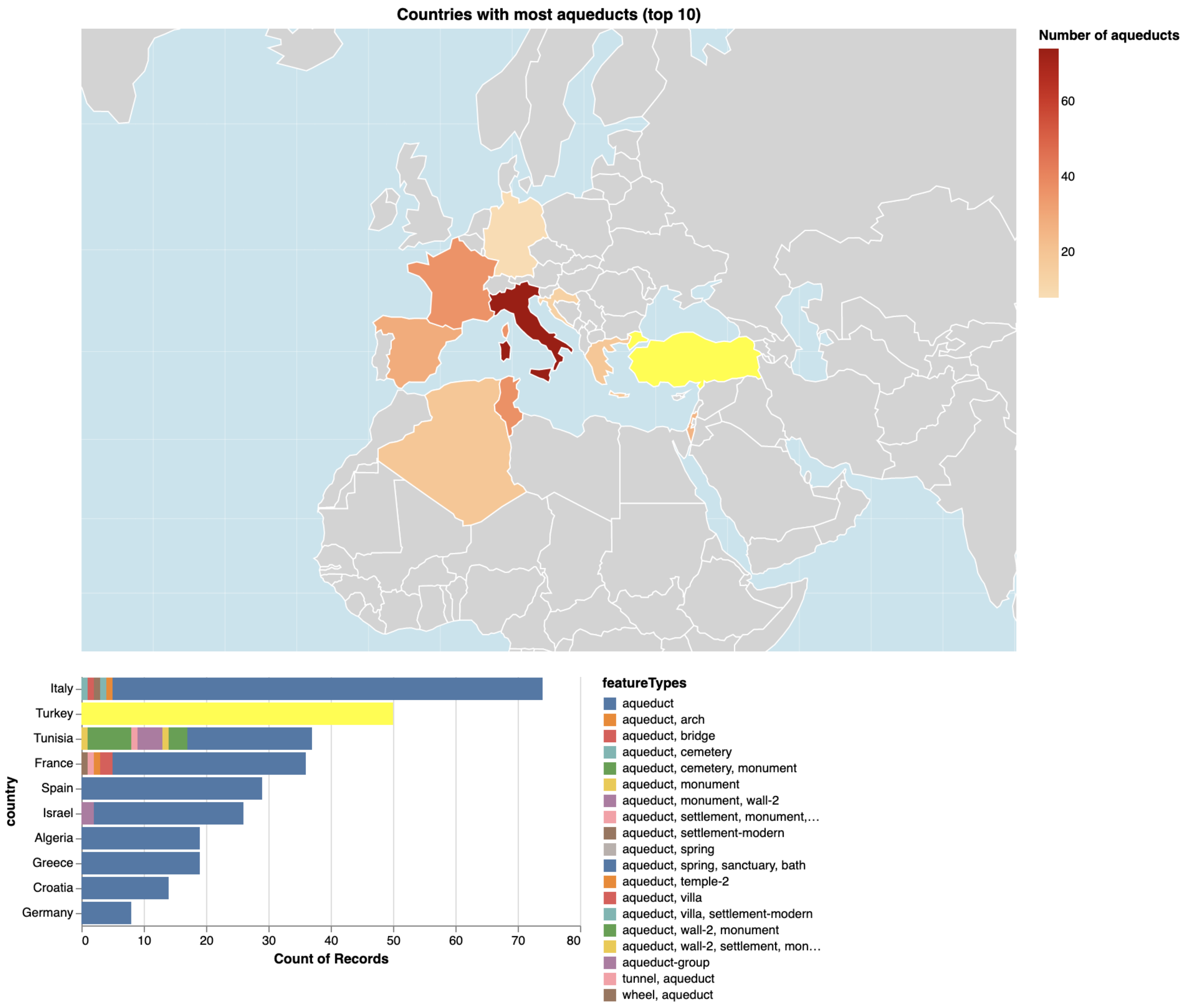
Choropleth geographic map & stacked bar chart

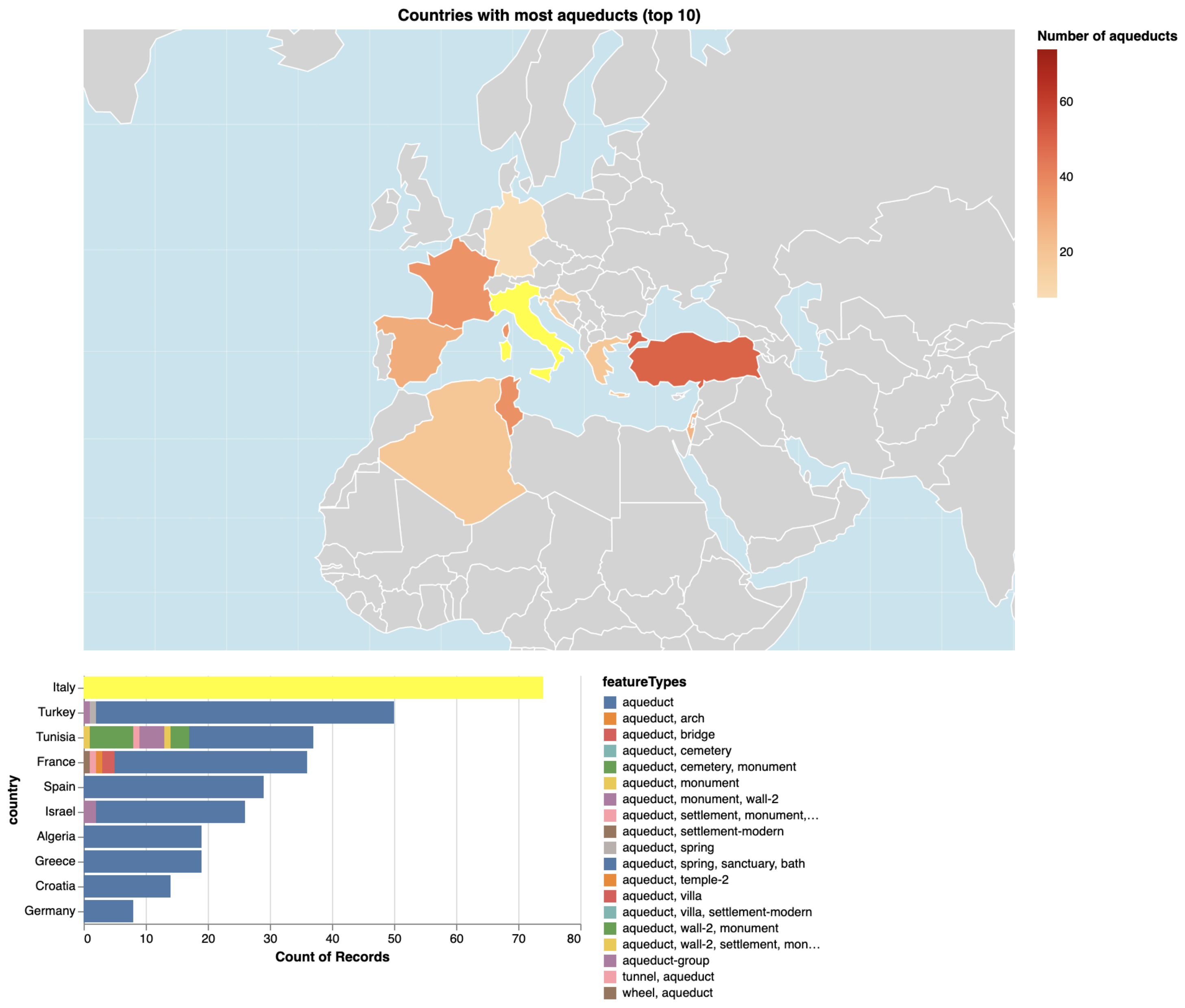
Altair code

```
selection = alt.selection(type='single', on='mouseover', fields=['country'], empty='none

bars = alt.Chart(kf).mark_bar().encode(
    x='count():Q',
    y=alt.Y('country:O', sort=alt.EncodingSortField(field="country", op="count", order=
    color=alt.condition(
        selection,
        alt.value('yellow'),
        'featureTypes:N'
    )
).add_selection(selection)

mp = data.world_110m.url
source = alt.topo_feature(mp, 'countries')

globe = alt.Chart(alt.sphere()).mark_geoshape(fill='lightblue', opacity=0.7)
meridian = alt.Chart(alt.graticule()).mark_geoshape(stroke='white', strokeWidth=0.3, opa

background = alt.Chart(source).mark_geoshape(
    stroke="white",
    fill="lightgray"
)

world_map = alt.Chart(source).mark_geoshape(
    stroke="white"
).encode(
     color=alt.condition(selection, alt.value('yellow'), alt.Color('counts:Q', legend=a
).transform_lookup(
    lookup='id',
    # left joining to get the count and country name
    from_=alt.LookupData(sorted_gr, 'country_id', ['country', 'counts'])
).add_selection(
    selection
)

layered_map = alt.layer(
    globe,
    meridian,
    background,
    world_map,
).properties(
    width=750,
    height=500,
    title='Countries with most aqueducts (top 10)'
).project(
    'mercator',
    scale=330,
    center=[-50, 66],
)

chart = (layered_map & bars).configure_view(stroke=None)
```

```
chart
```

Examples & interaction





What are all visual mappings used?

Map:

Color (hue + saturation of the 'orangered' palette): count of aqueducts. Previous mappings: count of aqueducts => country => country_id => country_id in the map 'world_110m' dataset.

Bar chart:

x position

count of aqueducts

y position

country (Nominal)

within a bar:

featureType => color

Was there any special data preparation done?

Country mapping (including country id mapping), grouping, aggregation by total count of aqueducts, sorting and filtering.

```
# filtering only those that contain "aqueduct"
df = df[(df['featureTypes'].str.contains("aqueduct"))]
```

```
# executing the cell below requires time, so you can skip ahead and uncomment the 3rd ce
# getting the country name from the given coordinates
def get_country(lat, long, geolocator):
    time.sleep(2)
    string = f'{lat}, {long}'
    locationString = geolocator.reverse(string, language='en')
    if locationString is None:
        return 'Unknown'
    locationArr = locationString.address.split(', ')
    if locationArr == []:
        return 'Unknown'
    else:
        return locationArr[-1]
```

```
# applying the function
geolocator = Nominatim(user_agent="vis_cw")
df['country'] = df.apply(lambda row: get_country(row.reprLat, row.reprLong, geolocator),
```

```
# the operation above takes quite a while, so we save it for the further use
df.to_csv('pleiades_vis3_withCountry.csv')
```

```
# uncomment the code below to read directly from the file (skip 3 cells above)
# df = pd.read_csv('pleiades_vis3_withCountry.csv')
```

```
# grouping, sorting, selecting top 10
country_gr = df.groupby(['country'])
sorted_gr = country_gr.size().reset_index(name='counts').sort_values('counts', ascendin
```

```
sorted_gr
```

	country	counts

	country	counts
13	Italy	74
23	Turkey	50
22	Tunisia	37
6	France	36
20	Spain	29
12	Israel	26
1	Algeria	19
8	Greece	19
4	Croatia	14
7	Germany	8

```
unique = sorted_gr.country.to_numpy()
```

```
unique
```

```
array(['Italy', 'Turkey', 'Tunisia', 'France', 'Spain', 'Israel',
       'Algeria', 'Greece', 'Croatia', 'Germany'], dtype=object)
```

```
# getting ids of each country in the list
r = requests.get('https://raw.githubusercontent.com/alisle/world-110m-country-codes/mas
d = r.json()
k = {}
for x in d:
    k[x['name']] = x['id']

dct = {}
for x in unique:
    dct[x] = k[x]
```

```
kf = df[df['country'].isin(unique)]
```

```
pd.options.mode.chained_assignment = None
kf['country_id'] = kf['country'].map(dct)
```

```
sorted_gr['country_id'] = sorted_gr['country'].map(dct)
```

```
sorted_gr
```

	country	counts	country_id
13	Italy	74	380
23	Turkey	50	792
22	Tunisia	37	788
6	France	36	250
20	Spain	29	724
12	Israel	26	376
1	Algeria	19	12
8	Greece	19	300
4	Croatia	14	191
7	Germany	8	276

What are the limitations of your design?
If we wanted to have a wider range of countries (e.g., 100), it would be very difficult to pick a country of interest from both the bar chart and the choropleth map. For the former it is evident due to limited space and for the latter it is hardly possible to pick a color map which can address subtle differencies in the value represented by color.

Emergence of pleiades from different categories over the last 4000 years (excl. settlements)

This visualization fulfills the concept of "overview first, zoom and filter" (in our case - overview => filter => zoom) from the Shneiderman's mantra.

Setup

```python
import altair as alt
import pandas as pd
```

```python
df = pd.read_csv('pleiades.csv')
```

What can we learn from the visualization?

We can figure out which types of pleiades were formed in the last 4000 years besides settlements (since this is by far the most common category) and determine their approximate date of emergence. Hence, we can establish whether there is a connection between certain time periods and categories. Also, we can examine each single location on the tick chart after selecting an interval of interest on the heatmap.

What is the name for the type of visualization(s) used?

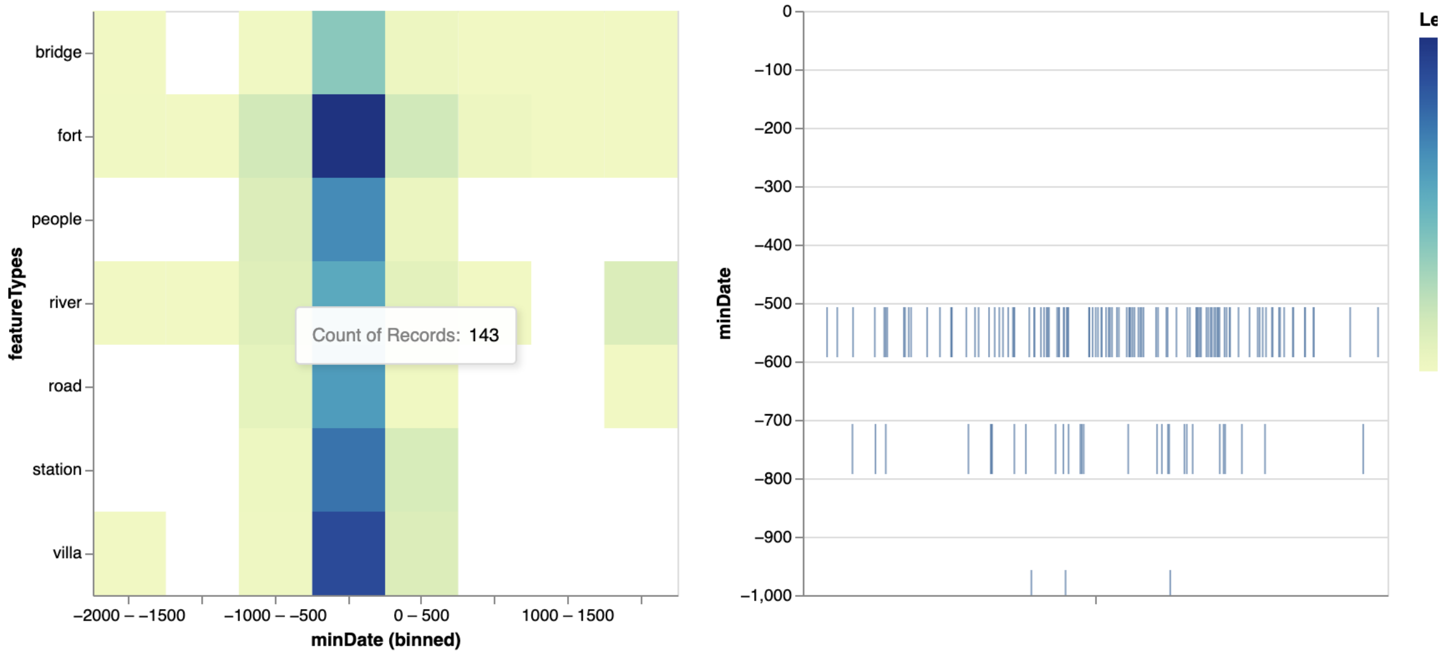
Heatmap & tick chart

Altair code

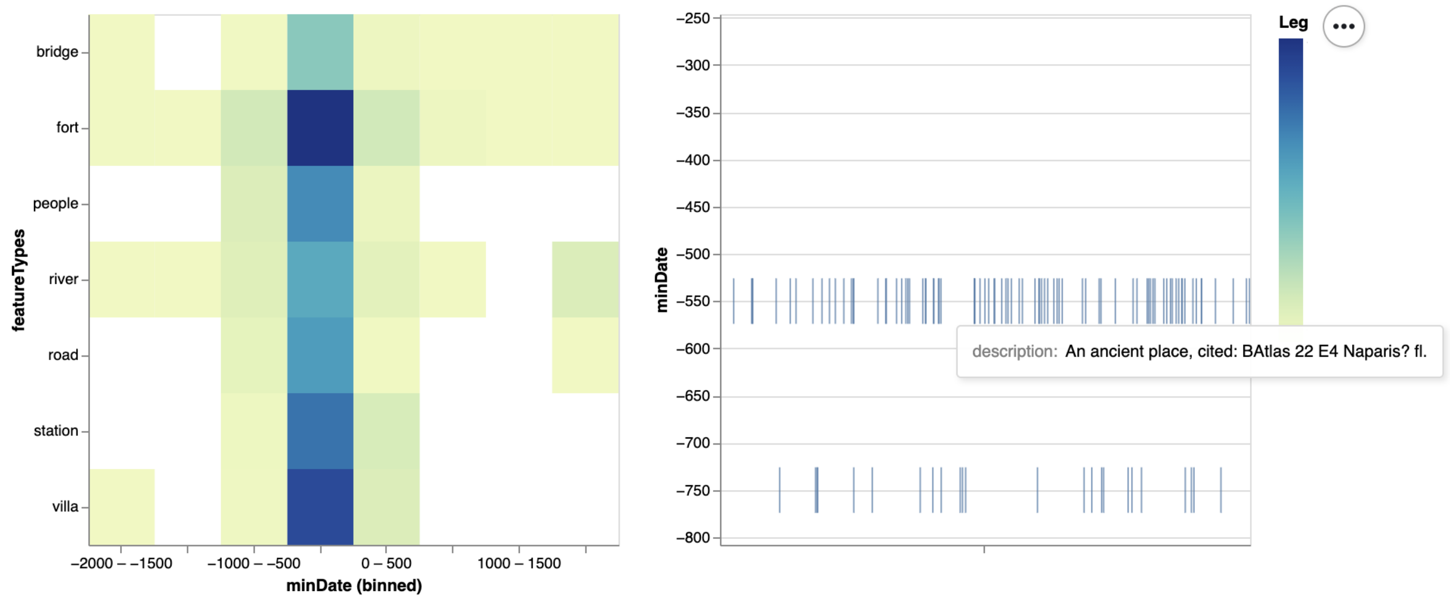
```
alt.data_transformers.disable_max_rows()
# selection a square of the heatmap
selection = alt.selection(type='single', encodings=['x', 'y'], empty='none')

heatmap = alt.Chart(df).mark_rect().encode(
    alt.X('minDate:O', axis=alt.Axis(labelAngle=0), bin=alt.Bin(step=500)),
    alt.Y('featureTypes:N'),
    # color is based on selection / this works only is one encoding (x | y) is specifie
    color=alt.condition(
        selection, alt.value('darkred'), alt.Color('count():Q', legend = alt.Legend(tit
    ),
    tooltip=["count():Q"]
).properties(
    width=350,
    height=350
).add_selection(selection)

tick = alt.Chart(df).mark_tick(height=30).encode(
    x=alt.X(
        'jitter:Q',
        title=None,
        axis=alt.Axis(values=[0], grid=False, labels=False),
        scale=alt.Scale(),
    ),
    y=alt.Y('minDate:Q'),
    tooltip=["description:N"]
).properties(
    width=320,
    height=320,
).transform_calculate(
    # a lot of ticks have the same position due to the nature of the data set, hence we
    jitter='sqrt(-2*log(random()))*cos(2*PI*random())'
).transform_filter(
    selection
).interactive()

chart = (heatmap | tick).properties(title='Emergence of pleiades from different categor


chart
```

Examples & interaction

Selecting a cell on the heatmap updates the chart showing ticks.



We can also zoom in on the ticks on the right chart.



What are all visual mappings used?

Heatmap:

Color (hue + saturation): count of records

x position

binned minDate (1 bin = 500 years)

y position

featureTypes (Nominal)

Tick chart:

x position

jitter - to avoid ticks overlaying each other

y position

minDate

Was there any special data preparation done?

Filtering by date of emergence, grouping by category and removing settlements / uncategorized data.

```
df = df[df['minDate'] >= -2000]


featureType_gr = df.groupby(['featureTypes'])
sorted_gr = featureType_gr.size().reset_index(name='counts').sort_values('counts', ascen


unique = sorted_gr['featureTypes'].to_numpy()


# we remove settlements due to their overwhelming supremacy
df = df[(df['featureTypes'].isin(unique)) & (~df['featureTypes'].str.contains("unknown"
```

What are the limitations of your design?

Selection on the heatmap is not highlighted due to the limitations of binning in vega-lite. Binning itself is also limited in order to accumulate fewer points in a bin to prevent from excessive memory consumption. The use of jitter might have a flip side - it might not be clear why the ticks have a different (uneven) horizontal spatial position. The use of minDate instead of mean(minDate, maxDate) might be questionable, however, it provides more reliablity.

Area occupied by the pleiades

Setup

```
import altair as alt
import pandas as pd
from vega_datasets import data
import numpy as np
import math
```

```
df = pd.read_csv('pleiades.csv')
```

What can we learn from the visualization?

The goal of this visualization is a bit abstract. We believe it can help identify interesting observations with respect to the improvised spatial dimensions of the locations represented by the box formed by coordinates (bbox field). The visualization is aimed to find single observations detached from the main cohort (outliers, but not too extreme - explained further). This can enable interesting analysis of these locations and establishing the causes of their differencies.

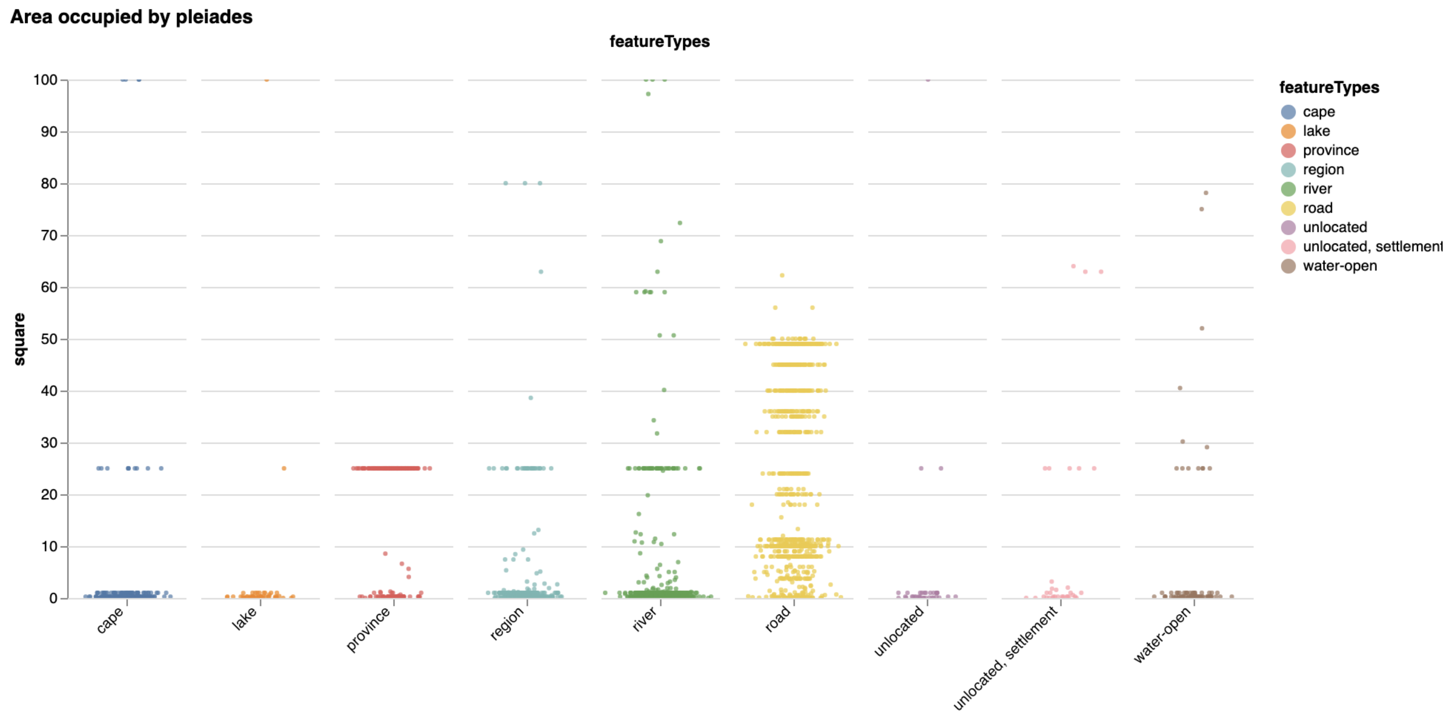
What is the name for the type of visualization(s) used?

Scatter plot (columned)

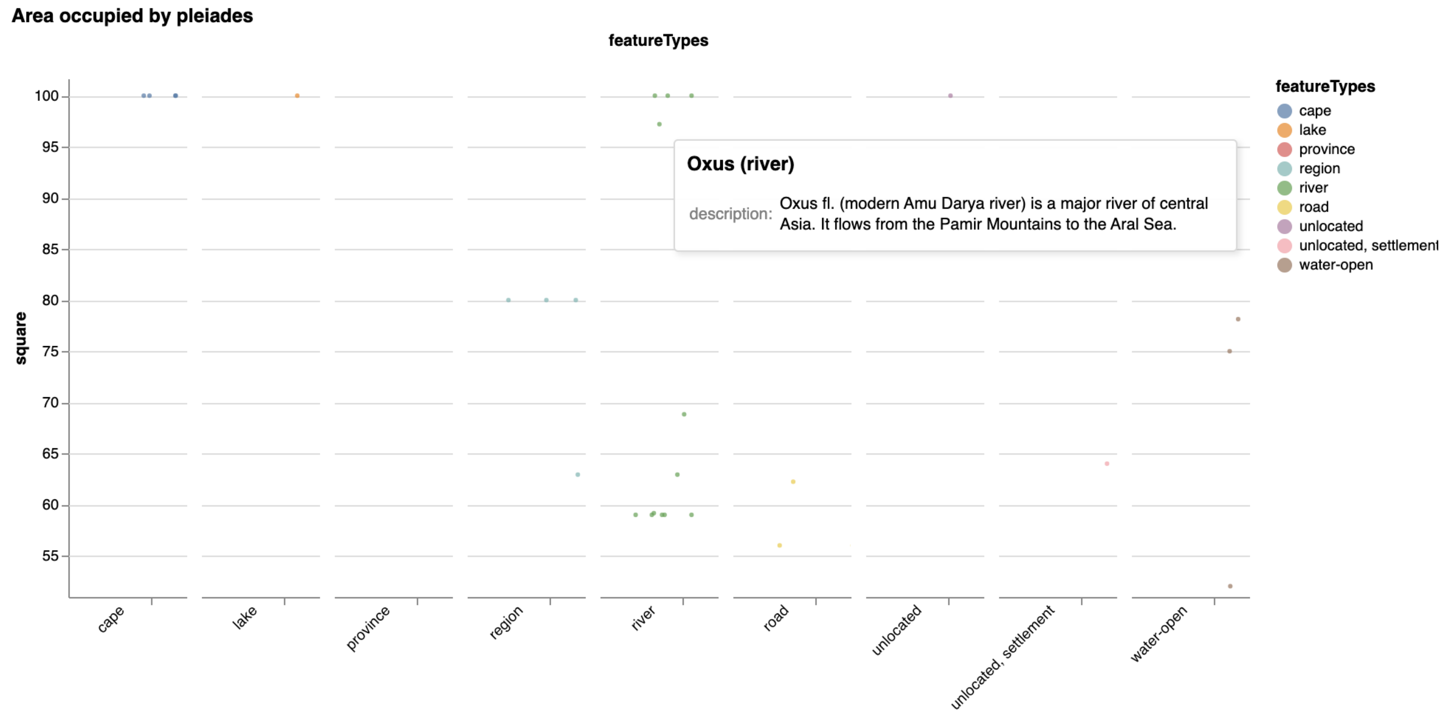
Altair code

```python
chart = alt.Chart(df).mark_circle(size=9).encode(
    x=alt.X(
        'jitter:Q',
        title=None,
        axis=alt.Axis(values=[0], ticks=True, grid=False, labels=False),
        scale=alt.Scale(),
    ),
    y='square:Q',
    color='featureTypes:N',
    tooltip=['title:N', 'description:N'],
    column=alt.Column(
        'featureTypes:N',
        header=alt.Header(
            labelAngle=-45,
            titleOrient='top',
            labelOrient='bottom',
            labelAlign='right',
            labelPadding=5,
        ),
    ),
).transform_calculate(
    jitter='sqrt(-2*log(random()))*cos(2*PI*random())'
).properties(
    width=80,
    height=350,
    title="Area occupied by pleiades"
).configure_facet(
    spacing=10
).configure_view(
    stroke=None
).interactive()


chart
```

Examples & interaction



If we zoom in on a column:



Oxus (river)

description: Oxus fl. (modern Amu Darya river) is a major river of central Asia. It flows from the Pamir Mountains to the Aral Sea.

What are all visual mappings used?

x position

featureTypes / category (Nominal)

y position

square (retrieved from the bounding coordinate box - "bbox" field)

within a column random jitter is applied to x-axis

Color: featureTypes (Nominal)

Was there any special data preparation done?
We need to calculate the square for each location, group the data set by category (featureTypes), then calculate the standard deviation within each category and sort from highest to lowest to maximize the chance of getting interesting results. Also, we remove extreme outliers and categories with a small count of observations (the latter is explained in the code).

```
def get_square(loc_str):
    arr = loc_str.split(',')
    if len(arr) == 4:
        res = (float(arr[2]) - float(arr[0])) * (float(arr[3]) - float(arr[1]))
        if res > 0:
            return res
        else:
            return 0
    else:
        return 0


df['bbox'] = df['bbox'].astype(str)
df['square'] = df.apply(lambda row: get_square(row.bbox), axis=1)
```

The upper boundary we are setting is aimed to delete extreme outliers (e.g., seas, islands, etc.) and **helps maintain overall coherence of the scatter plot.**

```
# removing 0 square and outliers (101 is an empirical value)
df = df[(df['square'] > 0) & (df['square'] <= 101)]


gb = df.groupby(['featureTypes'])
# sorting by the highest std of the square
sorted_gb = gb.agg(np.std, ddof=1).reset_index().sort_values('square', ascending=False)
most_std = sorted_gb['featureTypes'].to_numpy()


k = gb.size().reset_index(name='counts').sort_values('counts', ascending=False)
```

```
# this code can be used to set a value for the count of the location in a category
# since we are relying on std, we would want to have more condidence by increasing the
k = k[k['counts'] > 10]
count_pass = k['featureTypes'].to_numpy()
intersect = np.intersect1d(most_std, count_pass)
df = df[df['featureTypes'].isin(intersect)]
```

What are the limitations of your design?

Zooming in & out is required to pick a desired location. Hence, getting back to the initial position where all observations are visible can take some time. Also, due to the nature of the data set, (even with the use of jitter) a lot of points are merged together and require further zooming in. Finally, a lot of bounding boxes of the observations in the data set are in fact just a point (have no square), so the scope of this visualization is quite limited.