What can we learn from the visualization?

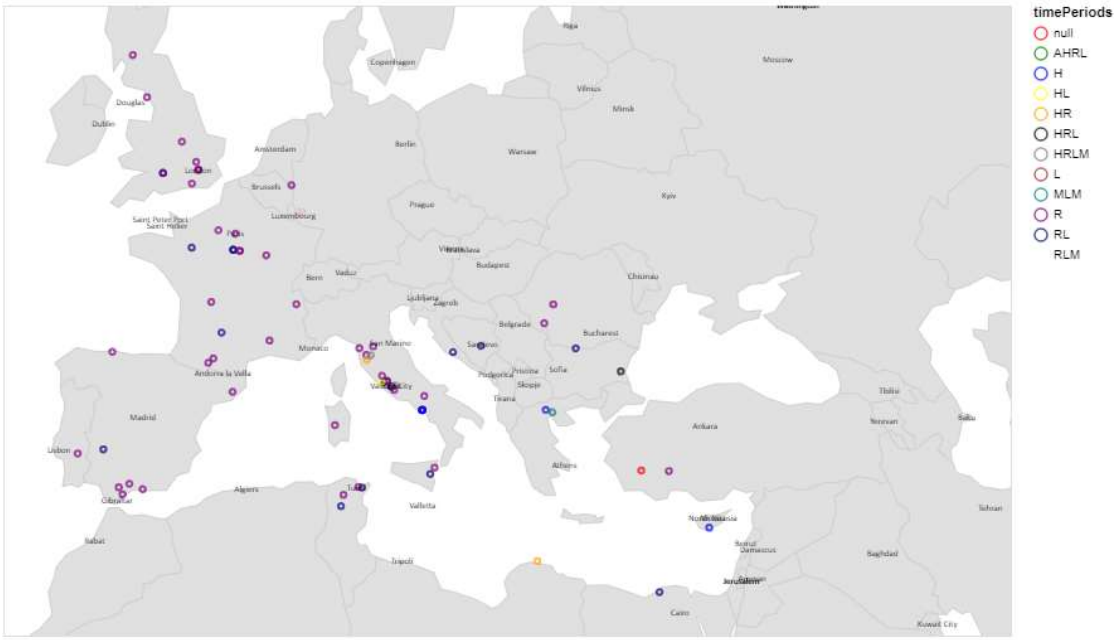
The aim of this visualisation is to show all the baths around Europe and how the roman empire built the vast majority of them.

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

map

```
import altair as alt
from altair import datum
from vega_datasets import data
import numpy as np
import pandas as pd
from geopy.geocoders import Nominatim

mapDataWithCountryNames= alt.topo_feature("https://raw.githubusercontent.com/johan/world.geo.json/master/c
mapData = alt.topo_feature("https://vega.github.io/vega/data/world-110m.json",'countries')

locs = pd.read_csv('./pleiades-locations-latest.csv')
places = pd.read_csv('./pleiades-places-latest.csv')
names = pd.read_csv('./pleiades-names-latest.csv')

capitals = pd.read_json('http://techslides.com/demos/country-capitals.json')



print("locs.shape = ",locs.shape)
print("places.shape = ",places.shape)
print("names.shape = ",names.shape)
print("capitals.shape = ",capitals.shape)

# Data Prep

locsTen= locs[1:40468:10]   #takes ten'th item from locs
bathFilter=((locs.featureType==("bath,"))|(locs.featureType==("bath")))
bathData = locs[bathFilter]
bathPoints = bathData[1:bathData.size:10]
print("bathData.shape = ",bathData.shape)
print("bathPoints.shape = ",bathPoints.shape,"\n")

alt.data_transformers.disable_max_rows()
```

```
#prerequisites
colorRange = ['red','green','blue','yellow','orange','black','grey','brown','teal','purple','navy','cream'
click = alt.selection_multi(encodings=['color'])
centerVal = [20,45]
scaleVal = 700

# world map
background = alt.Chart(mapData).mark_geoshape(
    fill = '#E0E0E0',
    stroke = '#D0D0D0').project(
    center = centerVal,
    scale=scaleVal,
).properties(
    width=800,
    height=500
)


# points on background
settlementP = alt.Chart(bathData).mark_point(
    size=20
).project(
    center = centerVal,
    scale=scaleVal,
).encode(
    longitude='reprLong',
    latitude='reprLat',
    color=alt.Color('timePeriods',scale=alt.Scale(range=colorRange)),
    tooltip=['featureType','reprLatLong','maxDate','timePeriods']
).transform_filter(
    datum.timePeriods!="null"
)


# capital text
capitalsText = alt.Chart(capitals).mark_text(
    size=7,
    font='calibri'
).project(
    center = centerVal,
    scale=scaleVal,
).encode(
    longitude='CapitalLongitude:Q',
    latitude='CapitalLatitude:Q',
    text = 'CapitalName'
)

mainMap = background + settlementP + capitalsText
mainMap
```



What are all visual mappings used?

x position
: latitude of location
.
y position
: longitude of location
.
Color: shows the time period the baths are from (most are purple which is Roman)
.
Tooltips: hovering over the points give more information as to where it is, its maxDate and more

Was there any special data preparation done?

I filtered the locations set for featureTypes that were 'bath' or 'bath,' which also had a time period containing 'R'

What are the limitations of your design?

This visualisation doesn't represent densely populated areas such as Rome very well.

What can we learn from the visualization?

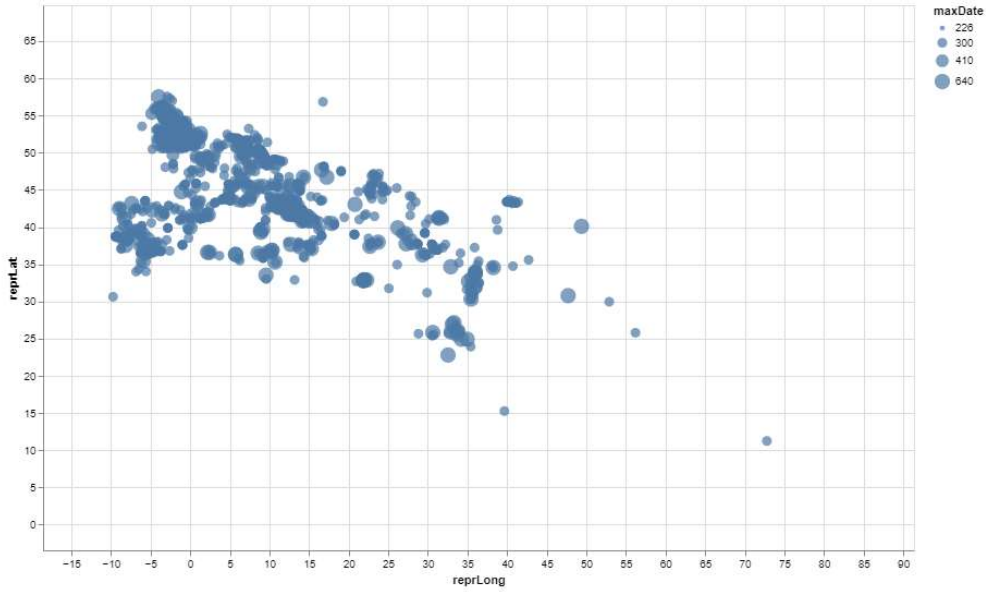
The aims of this visualisation is to encourage the user to explore around the interesting feature types within the dataset that are from the Roman time period. They can freely zoom into clusters and however over them (using tooltips (as is possible with all my visualisations)) to see how each area is arranged. This helps the user explore the empire as they can zoom into clusters such as the one on Rome and see the city layout as well as how old each point is.

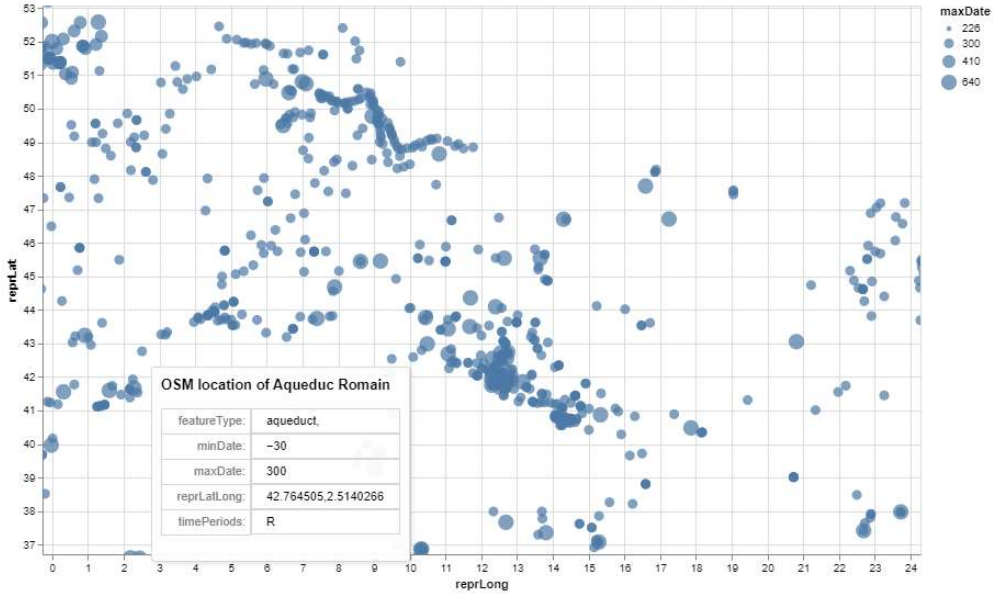
What is the name for the type of visualization(s) used?

Scatter Plot

```
import altair as alt
from altair import datum
from vega_datasets import data
import numpy as np
import pandas as pd
from geopy.geocoders import Nominatim

mapDataWithCountryNames= alt.topo_feature("https://raw.githubusercontent.com/johan/world.geo.json/master/countries.geo.
mapData = alt.topo_feature("https://vega.github.io/vega/data/world-110m.json",'countries')


locs = pd.read_csv('./pleiades-locations-latest.csv')
places = pd.read_csv('./pleiades-places-latest.csv')
names = pd.read_csv('./pleiades-names-latest.csv')

capitals = pd.read_json('http://techslides.com/demos/country-capitals.json')



print("locs.shape = ",locs.shape)
print("places.shape = ",places.shape)
print("names.shape = ",names.shape)
print("capitals.shape = ",capitals.shape)

# Data Prep

locsTen= locs[1:40468:10]   #takes ten'th item from locs

romanData =  (locs.timePeriods =="R") &  (locs.featureType!="unknown") & (locs.featureType!="unknown,")
romanPoints = locs[romanData]
#pointsTen= romePoints[1:romePoints.size:10]

print("locsTen.shape = ",locsTen.shape)
print("romeData.shape = ",romeData.shape)
print("romePoints.shape = ",romePoints.shape)
#print("pointsTen.shape = ",pointsTen.shape)

alt.data_transformers.disable_max_rows()
```

```
# prerequisites
click = alt.selection_multi(encodings=['color'])


# scatter plots of points
scatter = alt.Chart(romanPoints).mark_circle(
).encode(
    x='reprLong:Q',
    y='reprLat:Q',
    size=alt.Size('maxDate:O',
        scale=alt.Scale(range=(20,200)))
    ,
    tooltip=['featureType','minDate','maxDate','title','reprLatLong','timePeriods']
).properties(
    width=800,
    height=500
).interactive()

# legend


chart = ( scatter )

chart
```





What are all visual mappings used?

.
x position
: latitude of location
.
y position
: longitude of location
.
Bubble size shows how old the points are
.
Tooltips show some additional information such as featureType and a description of the data point.

Was there any special data preparation done?

This visualisation allows the user to explore around the interesting feature types within the dataset. They can freely zoom into clusters and however over them (using tooltips (as is possible with all my visualisations)) to see how each area is arranged. For example, they can easily find Athens and see how the city is arranged and how old each building is based on how small the bubbles are.

What are the limitations of your design?

The visualisation looks some what cluttered when fully zoomed out, I also wanted to show feature type by colouring the points however this cause more clutter and I eventually chose against it as it would take away from the visulisation.

What can we learn from the visualization?

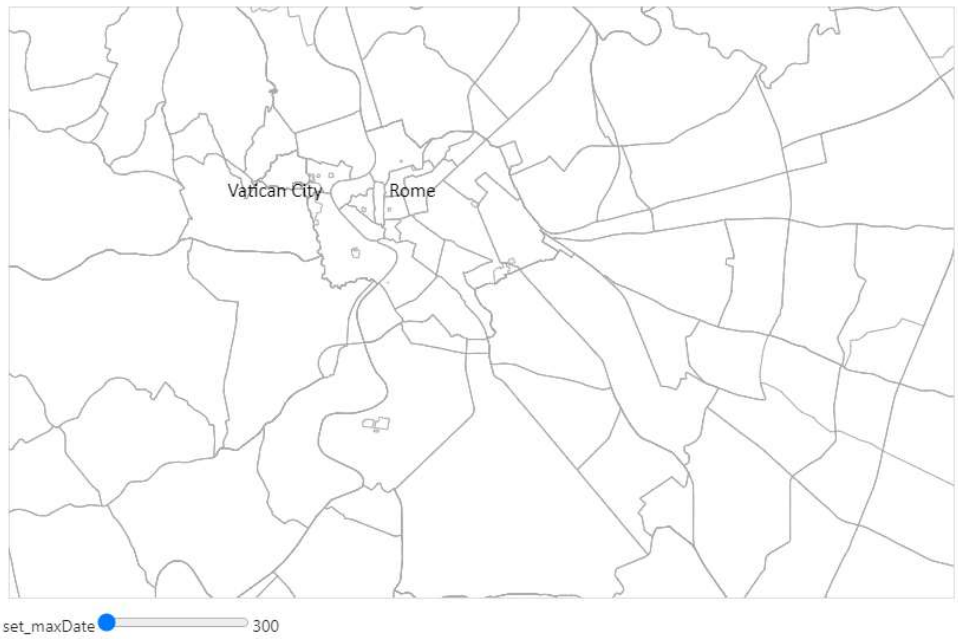
This visualisation shows how Rome expanded over time as the Roman empire grew from -30bc for the next several centuries. The map shows through the user of a slider how various buildings came into existence over time in Rome, with clusters of churches and tombs appearing and iconic Roman buildings such as the Basilica coming into existence.

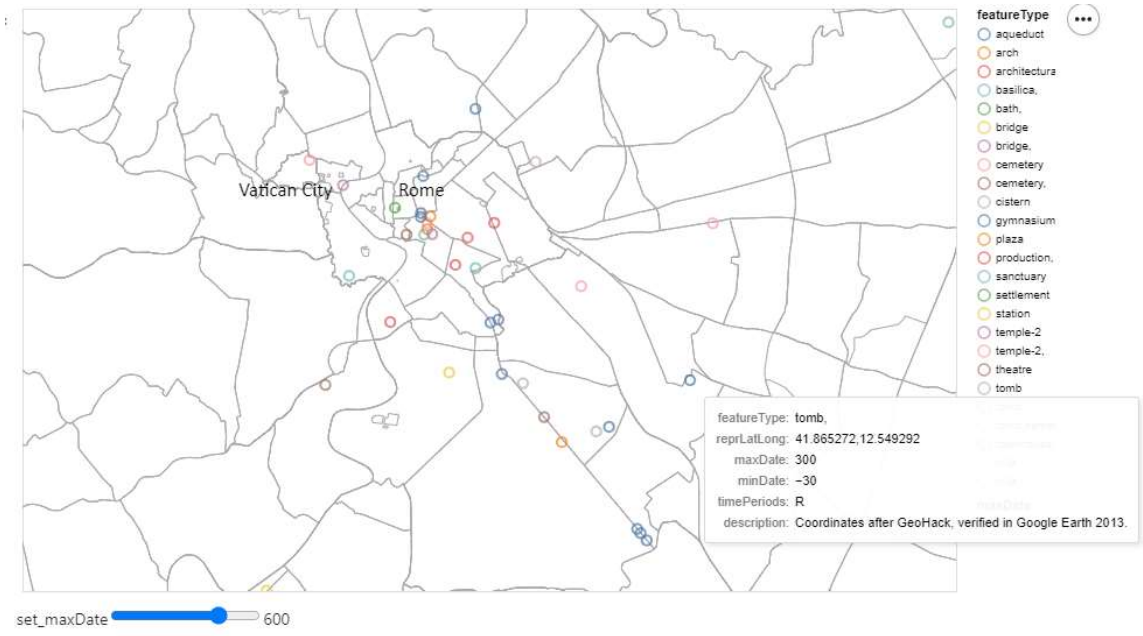
What is the name for the type of visualization(s) used?

Map with interactive slider

```
import altair as alt
import json
from altair import datum
from vega_datasets import data
import numpy as np
import requests
import pandas as pd

import json
from geopy.geocoders import Nominatim

mapDataWithCountryNames= alt.topo_feature("https://raw.githubusercontent.com/johan/world.geo.json/master/countries
mapData = alt.topo_feature("https://vega.github.io/vega/data/world-110m.json",'countries')

locs = pd.read_csv('./pleiades-locations-latest.csv')
places = pd.read_csv('./pleiades-places-latest.csv')
names = pd.read_csv('./pleiades-names-latest.csv')

capitals = pd.read_json('http://techslides.com/demos/country-capitals.json')


print("locs.shape = ",locs.shape)
print("places.shape = ",places.shape)
print("names.shape = ",names.shape)
print("capitals.shape = ",capitals.shape)

#Data Prep
romeData = ((locs.timePeriods =="R")|(locs.timePeriods =="HR")|(locs.timePeriods =="HRL")
            |(locs.timePeriods =="RL")|(locs.timePeriods =="R")) & ((locs.featureType != "unknown")&(locs.featureTy
romePoints = locs[romeData]


#prerequisites
click = alt.selection_multi(encodings=['color'])
centerVal = [12.5,41.88]
scaleVal = 200000

romeMap = alt.topo_feature('/tree/976335/rome.geojson','id')
```

```
# world map
background = alt.Chart(romeMap).mark_geoshape(
    fill = '#fffff',
    stroke = '#AAA9A9').project(
    center = centerVal,
    scale=scaleVal,
).properties(
    width=800,
    height=500
)

#slider
slider  = alt.binding_range(min=300, max=700, step=100)
selectorMaxDate  = alt.selection_single(name="set", fields=['maxDate'],
                                bind=slider, init={'maxDate': 300})

# points on background
romeP = alt.Chart(romePoints).mark_point(
    size=12
).project(
    center = centerVal,
    scale=scaleVal,
).encode(
    longitude='reprLong',
    latitude='reprLat',
    color=alt.Color('featureType',legend=None),
    tooltip=['featureType','reprLatLong','maxDate','minDate','timePeriods','description'],
    size=alt.Size('maxDate:N',scale=alt.Scale(range=(40,100)))
).add_selection(
    selectorMaxDate
).transform_filter(
    datum.maxDate < selectorMaxDate.maxDate
)

# capital text
capitalsText = alt.Chart(capitals).mark_text(
    size=17,
    font='calibri'
).project(
    center = centerVal,
    scale=scaleVal,
).encode(
    longitude='CapitalLongitude:Q',
    latitude='CapitalLatitude:Q',
    text = 'CapitalName'
)

mainMap = (background + romeP  + capitalsText)
mainMap
```





What are all visual mappings used?

x position
: latitude of location
.
y position
: longitude of location
.
Color shows the feature type which is detailed in the legend
.
Size represents the age of the points which is also detailed in its own legend.

This visualisation shows how Rome expanded over time as the Roman empire grew from -30bc for the next several centuries. The map shows through the user of a slider how various buildings came into existence over time in Rome, with clusters of churches and tombs appearing and iconic Roman buildings such as the Basilica coming into existence.

.

In terms of data prep the first thing I did was filter all points which had their latitude between 42.0277, 12.2990 and 41.8098, 12.6289. This meant only points in and around Rome would be shown. Then filtered this data for points which belonged to the roman time period.

.

The interactive slider filters the points based on how old they are, this shows how Rome expanded over time and gives the user complete control.

.

I also added a GeoJson of country capitals to help the user orient themselves.

.

The visulisation also makes use of an additional GeoJson which shows the road layouts within Rome, this gives context as to how ancient Rome was laid out as many of the points are along these roads that are thousands of years old.

What are the limitations of your design?

The visualisation is slightly cluttered towards the centre, it would be good for the user to be able to zoom in however unfortunately Altiair doesn't allow for zooming on GeoJson maps currently.

What can we learn from the visualization?

The aim of this visualisation is to allow the viewer to quickly see the spread of settlements throughout the European region and which countries have many or few within their borders.

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

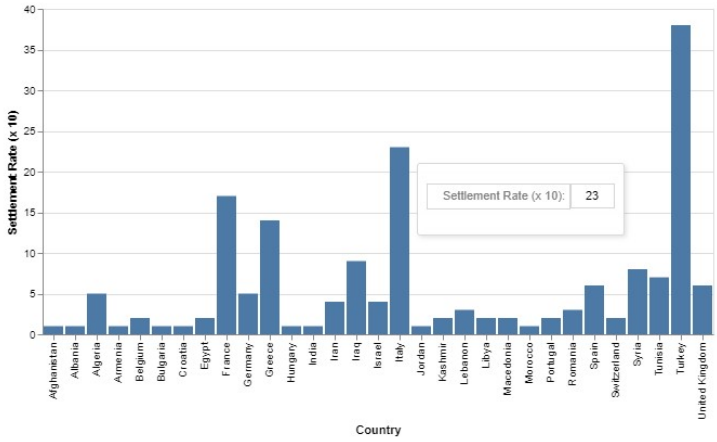
Histogram & Choropleth

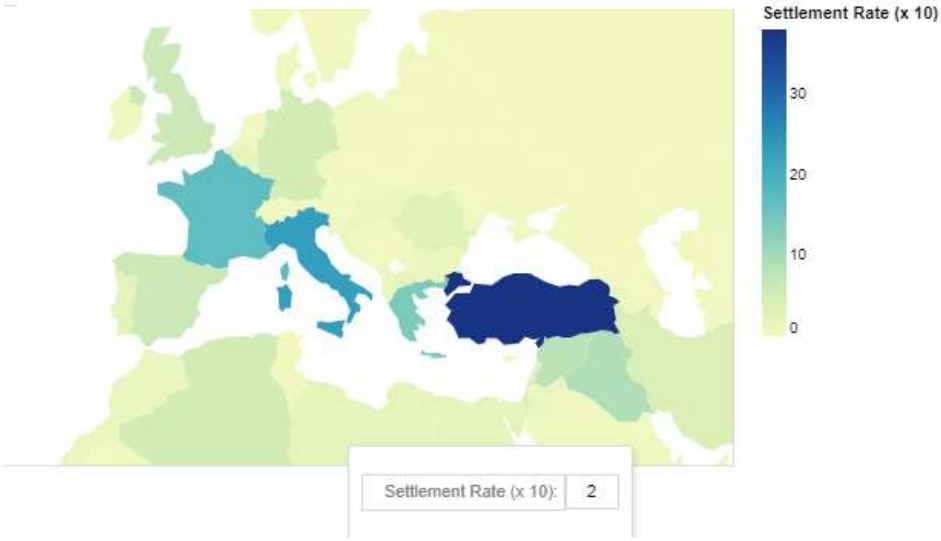
```
import altair as alt
from altair import datum
from vega_datasets import data
import numpy as np
import pandas as pd
from geopy.geocoders import Nominatim

mapDataWithCountryNames= alt.topo_feature("https://raw.githubusercontent.com/johan/world.geo.jsc
mapData = alt.topo_feature("https://vega.github.io/vega/data/world-110m.json",'countries')

locs = pd.read_csv('./pleiades-locations-latest.csv')
places = pd.read_csv('./pleiades-places-latest.csv')
names = pd.read_csv('./pleiades-names-latest.csv')

capitals = pd.read_json('http://techslides.com/demos/country-capitals.json')



print("locs.shape = ",locs.shape)
print("places.shape = ",places.shape)
print("names.shape = ",names.shape)
print("capitals.shape = ",capitals.shape)

#Data Prep
settlementFilter=locs.featureType==("settlement")
settlementData = locs[settlementFilter]
bathFilter=locs.featureType==("bath")
bathData = locs[bathFilter]
empireDataS = ((settlementData.timePeriods =="R")|(settlementData.timePeriods =="HR")|(settlemer
            |(settlementData.timePeriods =="RL")|(settlementData.timePeriods =="R"))
empireSettlementPoints = settlementData[empireDataS]
empireDataB = ((bathData.timePeriods =="R")|(bathData.timePeriods =="HR")|(bathData.timePeriods
            |(bathData.timePeriods =="RL")|(bathData.timePeriods =="R"))
empireBathPoints = bathData[empireDataB]


alt.data_transformers.disable_max_rows()
```

```
#prerequisites
centerVal = [30,45]
scaleVal = 500


# world map
background = alt.Chart(mapData).mark_geoshape(
    fill = '#fffff',
    stroke = '#AAA9A9').project(
    center = centerVal,
    scale=scaleVal,
).properties(
    width=800,
    height=500
)

#slider
slider  = alt.binding_range(min= 300, max=700, step=100)
selectorMaxDate  = alt.selection_single(name="set", fields=['maxDate'],
                                    bind=slider, init={'maxDate': 300})

# points on background
romePSettlement = alt.Chart(empireSettlementPoints).mark_point(
    color='#d1081f',
    size=12
).project(
    center = centerVal,
    scale=scaleVal,
).encode(
    longitude='reprLong',
    latitude='reprLat',
    tooltip=['reprLatLong','maxDate','minDate','timePeriods','description'],
    size=alt.Size('minDate:N',scale=alt.Scale(range=(6,30)))
).add_selection(
    selectorMaxDate
).transform_filter(
    datum.maxDate < selectorMaxDate.maxDate
)


# capital text
capitalsText = alt.Chart(capitals).mark_text(
    size=6.5,
    font='calibri'
).project(
    center = centerVal,
    scale=scaleVal,
).encode(
    longitude='CapitalLongitude:Q',
    latitude='CapitalLatitude:Q',
    text = 'CapitalName'
```

```
)

mainMap = (background + romePSettlement  + capitalsText)
mainMap

alt.data_transformers.disable_max_rows()

# get country name from coords
def getCountry(reprLatLong):
    geolocator = Nominatim(user_agent="my-appliation")
    location = geolocator.reverse(reprLatLong)
    local = location.address
    localCSV = local.split(',')
    return localCSV[-1]
    print(localCSV[-1])

settlementFilter=locs.featureType==("settlement")
settlementData = locs[settlementFilter]
settlementPoints = settlementData[1:settlementData.size:10]
settlementData.shape

# sendding data to file

settlementCoords = settlementPoints.reprLatLong
settlementCoords.tolist()[:]
pp.pprint (settlementCoords)

countryCoords = []
for i in settlementCoords:
    countryCoords.append(getCountry(i))

pp.pprint (countryCoords)

#data request (this may not work on your machine as i had to add a unique identifier)
with open('1301.1846', 'wt',encoding="utf-8") as myfile:
    wr = csv.writer(myfile, quoting=csv.QUOTE_ALL)
    wr.writerow(countryCoords)



# data manually copied in from outputted csv files using getCountry function with the settlemnet
apperance=[#1-400
'Spain','Morocco','Algeria','Algeria','Turkey','France','France','Algeria','Libya','Italy','Turk
#500-900
'Egypt','Greece','Kashmir','Greece','Turkey','Turkey','Turkey','Turkey','Italy','Italy','Italy',
#901-1300
'Turkey','Turkey','Turkey','Turkey','Algeria','Libya','Greece','Greece','Greece','Italy','Iran',
#1301-1846
'Turkey','Iraq','Syria','Iraq','Iran','Afghanistan','Lebanon','Turkey','Turkey','Syria','Syria',

#for manual hist
```

```
apperanceDist = list(dict.fromkeys(apperance))
apperanceCount = list(Counter(apperance).values())

#manual histogram
df = pd.DataFrame({'Country': apperanceDist,
                   'Settlement Rate (x 10)': apperanceCount})

histo = alt.Chart(df).mark_bar().encode(
    x='Country',
    y='Settlement Rate (x 10)',
    tooltip=['Settlement Rate (x 10):Q'],
)

matrix = np.c_[apperanceDist,apperanceCount]
matrix.shape
matrix
#matrix[:,[1]]
#matrix[:,[0]]



#prerequisites
centerVal = [-20,60]
scaleVal = 350

mapData = alt.topo_feature("https://vega.github.io/vega/data/world-110m.json",'countries')
mapDataWithCountryNames= alt.topo_feature("https://raw.githubusercontent.com/johan/world.geo.jsc
source = pd.read_csv('./choro.csv')



choro= alt.Chart(source).mark_geoshape().encode(
    shape='geo:G',
    color='Settlement Rate (x 10):Q',
    tooltip=['Settlement Rate (x 10):Q'],
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(data=mapData, key='id'),
    as_='geo'
).properties(
    width=500,
    height=300
).project(
    center = centerVal,
    scale=scaleVal,
)


histo | (choro)
```





What are all visual mappings used?

Histogram

x position : Country Name

y position : Number of settlements

Choropleth

Color : How many settlements are within the countries boarders

Was there any special data preparation done?

To create a choropleth map of settlements from the datasets provided required lots of data preparation, this was mainly because the dataset only provided lat/long coordinates for a location (no country name was given).

.

This meant I had to create a function in python to convert from coordinates of each point (provided by the dataset) into a country name. This was done through the use of an open-source library called ‘geopy’ that converts coordinates into an address. I would then convert this address into a CSV and extract the last value (the country name). This data was outputted as a csv file which I could then simply manually copy into the script to produce the 'Apperance' array which shows a list of country names for each time a point landed within that country.

.

Next, I could simply use a function to count the number of occurrences of each name and create a matrix of the names and the rate at which they appeared.
This data could then be used to create a histogram.

.

Then I was able to convert the country names to their ISO country codes (using a similar method) so it would align with the world-110m.json ID format (as it uses country codes rather than names).

.

This data could then be used to create the choropleth map.

.

It's worth noting that I took every 10th settlement point from the original data set, this helps cut down from >1800 points to around 180 which helps as geopy limits the amount of requests that can be made by one person but helps to retain the key information about what countries have more settlements than others.

.

What are the limitations of your design?

The visualisation creates hard boundaries for settlement points where there may be more nuance, for example there may be 20 settlements on a countries boarder and so it shows the entire country as having a high density when in fact it is in a small region.

What can we learn from the visualization?

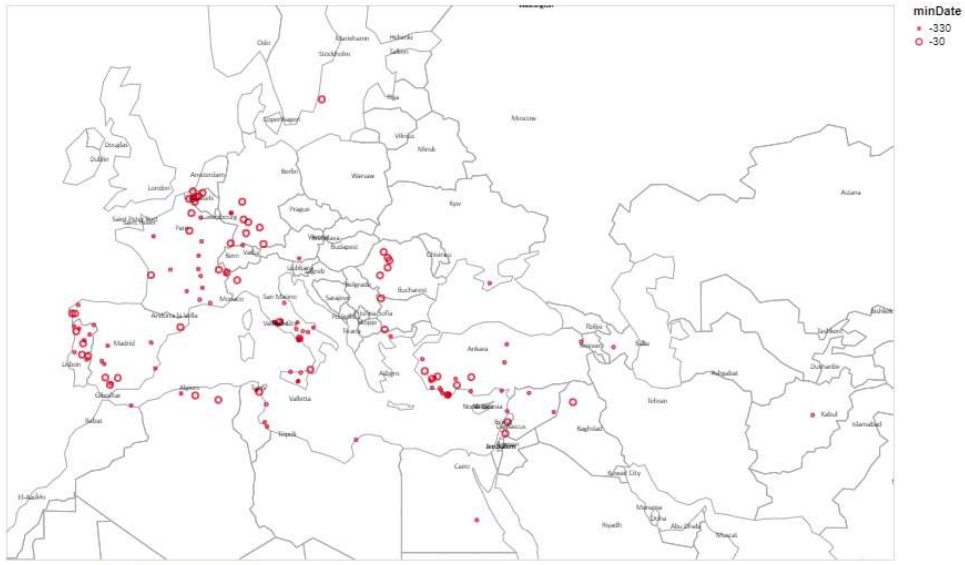
The aim of this visualisation is to show how the roman empire expanded over Europe over time and how the settlements expanded from central Italy through to northern Europe and eventually over to the United Kingdom. With newer settlements being shown as larger points.

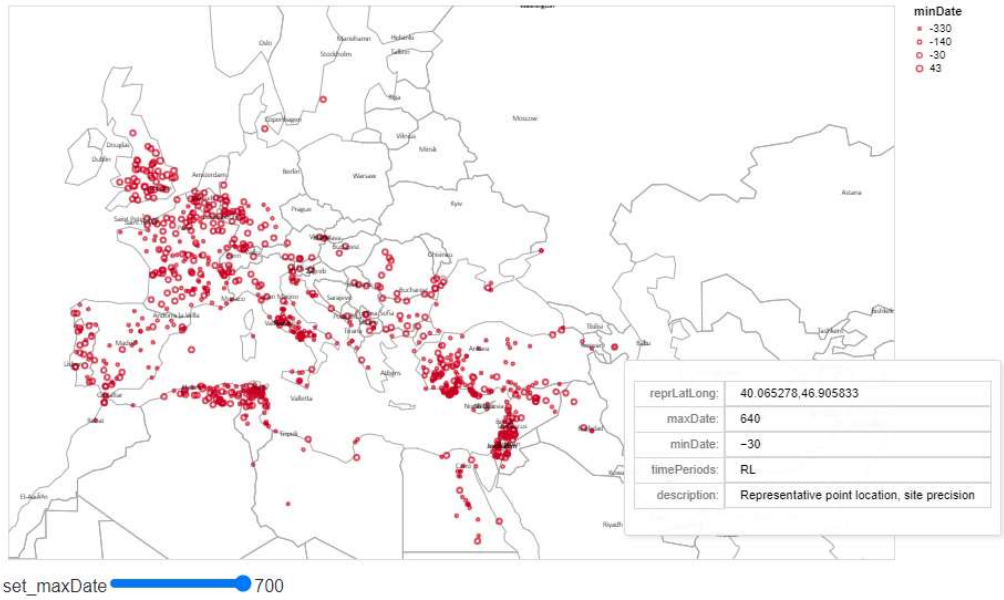
What is the name for the type of visualization(s) used?

Map with interactive slider

```
import altair as alt
from altair import datum
from vega_datasets import data
import numpy as np
import pandas as pd
from geopy.geocoders import Nominatim

mapDataWithCountryNames= alt.topo_feature("https://raw.githubusercontent.com/johan/world.geo.json/master/c
mapData = alt.topo_feature("https://vega.github.io/vega/data/world-110m.json",'countries')

locs = pd.read_csv('./pleiades-locations-latest.csv')
places = pd.read_csv('./pleiades-places-latest.csv')
names = pd.read_csv('./pleiades-names-latest.csv')

capitals = pd.read_json('http://techslides.com/demos/country-capitals.json')



print("locs.shape = ",locs.shape)
print("places.shape = ",places.shape)
print("names.shape = ",names.shape)
print("capitals.shape = ",capitals.shape)

#Data Prep
settlementFilter=locs.featureType==("settlement")
settlementData = locs[settlementFilter]

bathFilter=locs.featureType==("bath")
bathData = locs[bathFilter]

empireDataS = ((settlementData.timePeriods =="R")|(settlementData.timePeriods =="HR")|(settlementData.time
            |(settlementData.timePeriods =="RL")|(settlementData.timePeriods =="R"))
empireSettlementPoints = settlementData[empireDataS]

empireDataB = ((bathData.timePeriods =="R")|(bathData.timePeriods =="HR")|(bathData.timePeriods =="HRL")
            |(bathData.timePeriods =="RL")|(bathData.timePeriods =="R"))
empireBathPoints = bathData[empireDataB]


alt.data_transformers.disable_max_rows()
```

```
#prerequisites
centerVal = [30,45]
scaleVal = 500


# world map
background = alt.Chart(mapData).mark_geoshape(
    fill = '#fffff',
    stroke = '#AAA9A9').project(
    center = centerVal,
    scale=scaleVal,
).properties(
    width=800,
    height=500
)

#slider
slider  = alt.binding_range(min= 300, max=700, step=100)
selectorMaxDate  = alt.selection_single(name="set", fields=['maxDate'],
                                  bind=slider, init={'maxDate': 300})

# points on background
romePSettlement = alt.Chart(empireSettlementPoints).mark_point(
    color='#d1081f',
    size=12
).project(
    center = centerVal,
    scale=scaleVal,
).encode(
    longitude='reprLong',
    latitude='reprLat',
    tooltip=['reprLatLong','maxDate','minDate','timePeriods','description'],
    size=alt.Size('minDate:N',scale=alt.Scale(range=(6,30)))
).add_selection(
    selectorMaxDate
).transform_filter(
    datum.maxDate < selectorMaxDate.maxDate
)


# capital text
capitalsText = alt.Chart(capitals).mark_text(
    size=6.5,
    font='calibri'
).project(
    center = centerVal,
    scale=scaleVal,
).encode(
    longitude='CapitalLongitude:Q',
    latitude='CapitalLatitude:Q',
    text = 'CapitalName'
)

mainMap = (background + romePSettlement  + capitalsText)
mainMap
```





What are all visual mappings used?

.

x position

: latitude of location

.

y position

: longitude of location

.

Tooltips show some additional information such as a description of the data point.

.

Size represents how old the points are as shown in the legend.

First of all, I filtered for only settlements, and then from this data I filtered for only Roman time periods.

.

The interactive slider filters the points based on how old they are, this gives the desired effect of showing over time how the roman settlements expanded over time.

.

I also added a GeoJson of country capitals to help the user orient themselves.

What are the limitations of your design?

The visualisation only shows how settlements expanded, whilst there are other roman building types.