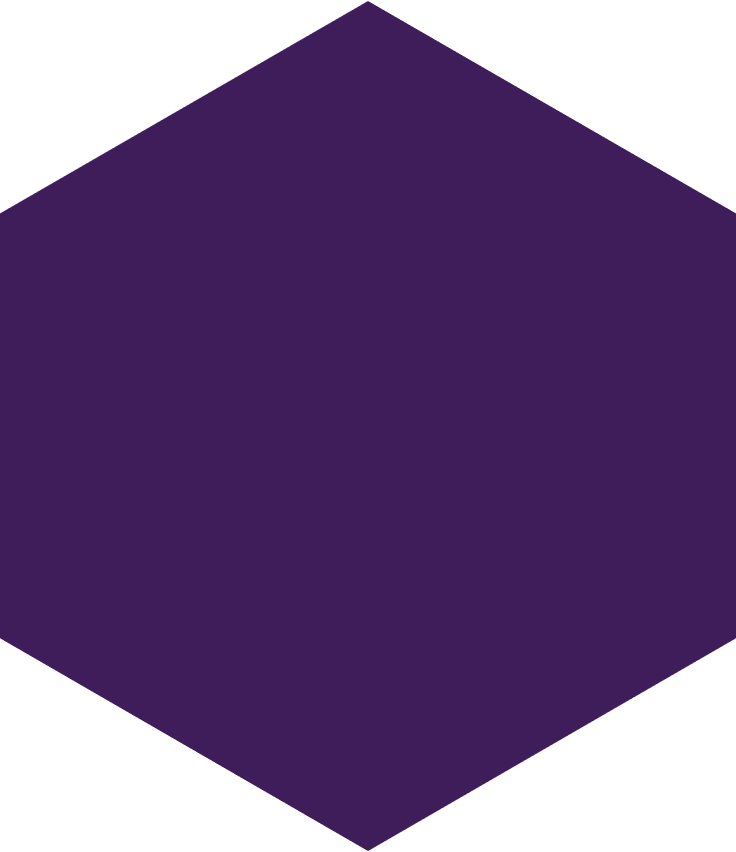
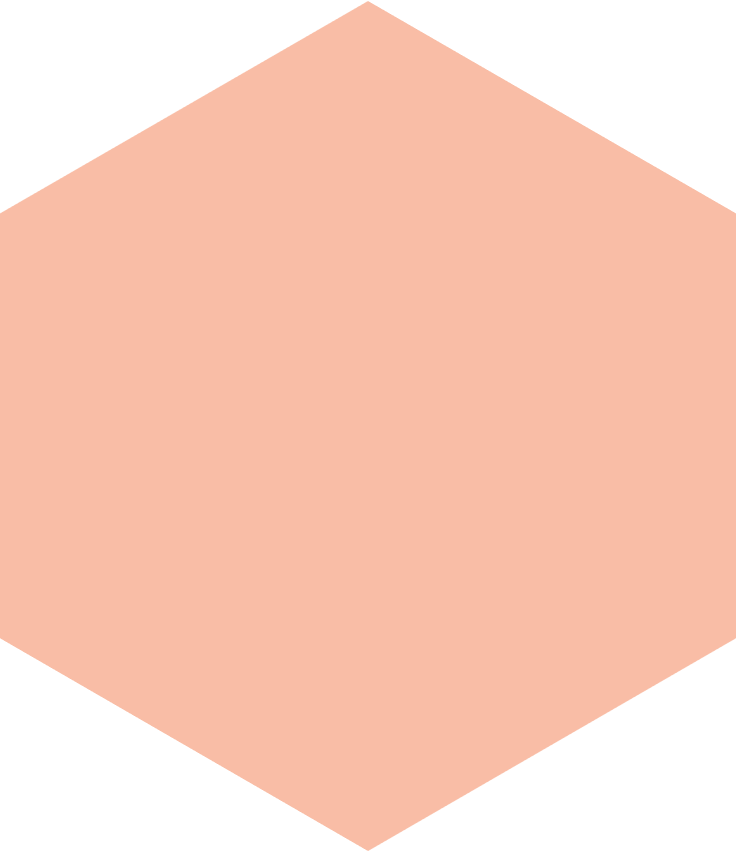


|  |
| --- |
| **Data Visualization of Historical Rainfall in India** |
| Vignesh Muthumani (10385771)  **Data Visualization – CA2** |
| Exploratory data analysis of the trend of rainfall across different regions of India from the year 1901-2015 |
|  |





**CONTENTS**

1. Introduction
2. Data Understanding
3. Methodology
   1. Python (Bokeh)
   2. Tableau
4. Key insights
5. Conclusion
6. References

**1. INTRODUCTION**

Rainfall is very crucial for a country like India, where agriculture is its backbone, which determines the GDP of the country. There has to be sufficient amount of rainfall for Agriculture production and to avoid the drought.

In this paper, we take a look at the previous history of rainfall in India by its regions (states) from 1901-2016. This could help us see the trend of rainfall since then, and provide some critical insights on which region is most affected, which region has it in surplus, etc.

**2. DATA UNDERSTANDING**

The dataset we are going to be working on for this project is “Actual rainfall in India from 1901 to 2016” obtained from the meteorological department of India through data.gov.in.

* The dataset has about 4416 observations and 15 variables.
* It possesses the information on the amount of rainfall (in mm) for each of the 36 states of India.
* It provides the information for the period of 1901-2016 by each month.
* It is to be noted that the monsoon season falls under the months of June to September, where the amount of rainfall in each state is far higher than the other months.

**3. METHODOLOGY**

For visualizing the data to obtain insights from the data, our methodology involves in the use of following tools and techniques and appraise their features and capabilities to do an exploratory data analysis.

1. **Python (Bokeh):** Python is an interpreted high-level programming language for general-purpose programming. We used Bokeh library in Python, using which we could create interactive and high-performance visualizations on large datasets. Bokeh is an interactive visualization library in Python that targets modern web browsers for presentation. Its goal is to provide elegant, concise construction of versatile graphics, and to extend this capability with high-performance interactivity over very large or streaming datasets. Easy to create interactive plots, dashboards, and data applications [2].

It gives one the power of D3 sort of visualization in a high-level programming language such as Python. We have used this tool to examine and explore the historical rainfall pattern in India. We have used Python for data pre-processing. All the irrelevant variables and observations are edited out by retaining only the actual rainfall data.

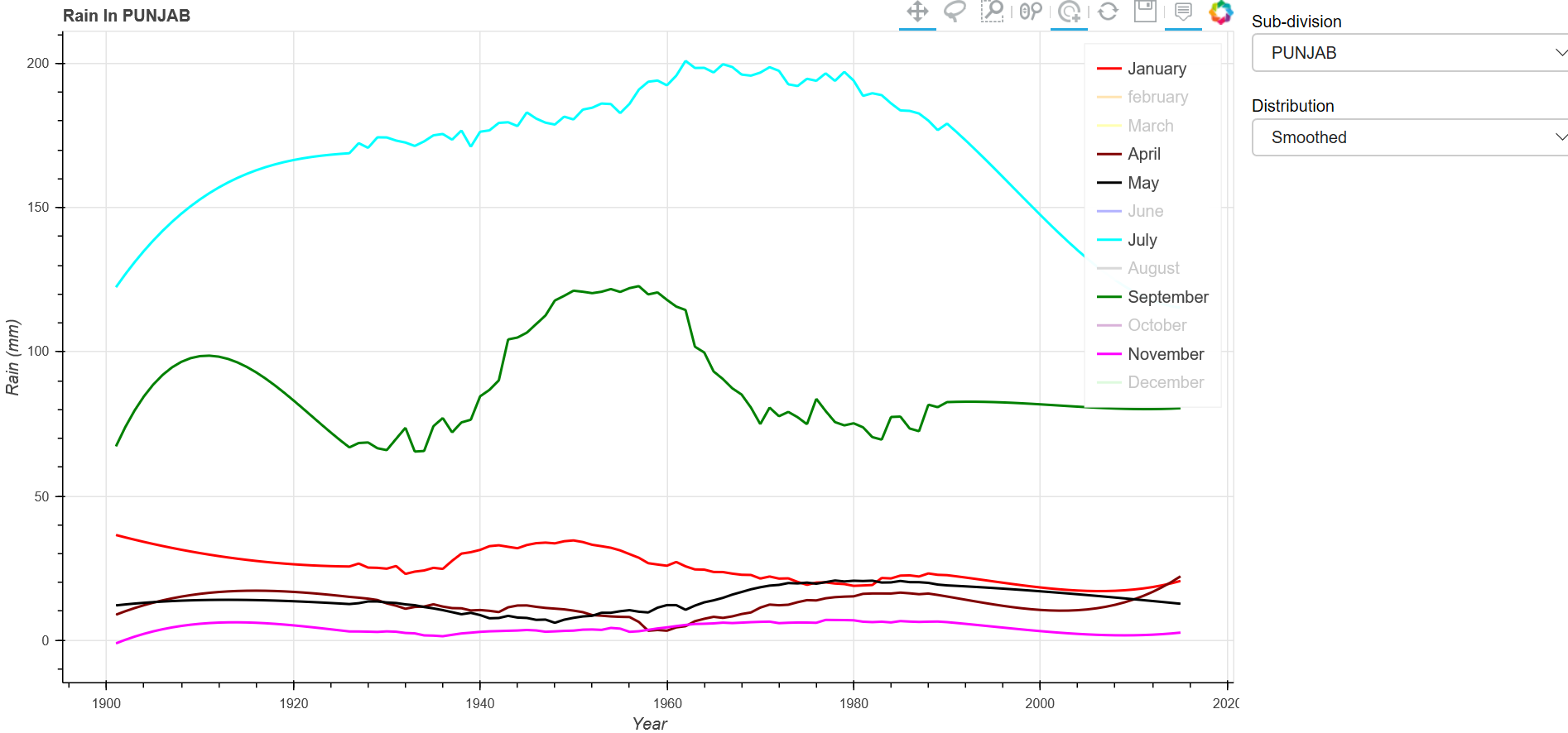
It has many interactive features which have been used in this project such as drag, tap, reset, save, hover, drop down list for state, drop down for type of plot, box and mouse wheel zoom, etc.

One great feature is the ability to save the certain section of visualization, this could be very useful in particular cases. It also has box zoom and mouse wheel feature used. Tap feature is used to provide external link to other website on click of the plotting. It is to be also noted that, other than Python, this library is also present in R and JavaScript.

Some of the interactive features that were used for the visualization of data are,

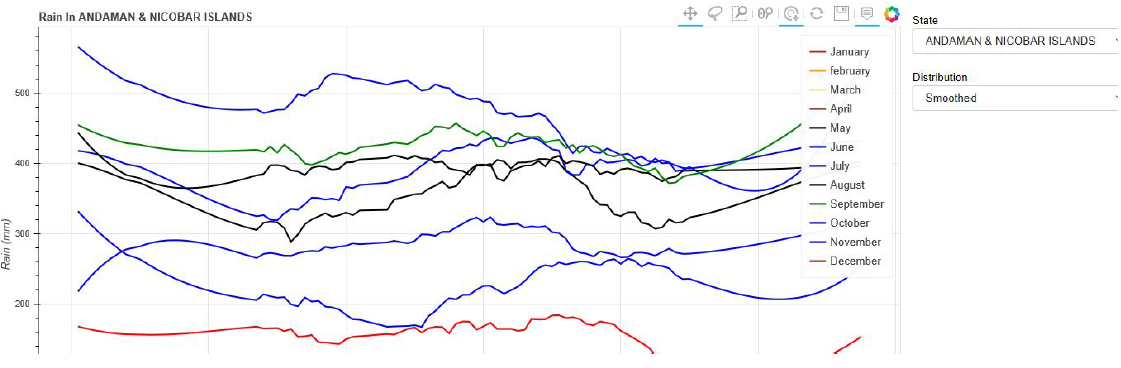
* **Drag tool:** This feature gives the ability for the user to move the graph by dragging it using the mouse. This is used when multiple tabs are open, and user needs to see the data without altering the size of browser window [4].
  + **Lasso Select tool:** Allows user to select any region in any shape without restrictions on the plot using the left-drag on mouse [4].
  + **Box zoom tool:** Allows user to select a rectangular region and zoom in the region on plot by left-dragging the mouse [4].
  + **Pan tool:** Pan tool is used for dragging the plot using left-drag on mouse. This will be useful for exploring the plot in certain focused areas [4].
* **Tap tool:** On the tap of a particular plot, an external link is opened in the browser. In this project, we have given the external link as the data source from where the data was downloaded [4].
  + **Scroll/pinch tool:** These tools are employed by pinching (on touch devices) or scrolling (on mouse devices). Only one scroll/pinch tool may be active at a time [4].
  + **Wheel zoom tool:** This tool will zoom in and out the plot based in the current mouse location [4].
* **Actions:** Actions are operations that are activated only when their button in the toolbar is tapped or clicked. They are typically modal or immediate-acting [4].
  + **Reset tool:** This is simply used to revert the changes made and reset to its default effect [4].
  + **Save tool:** Save feature is used to save a certain part of plot as a picture.
* **Inspectors:** Inspectors are passive tools that annotate or otherwise report information about the plot, based on the current cursor position. Any number of inspectors may be active at any given time. Inspectors menu in the toolbar allows users to toggle the active state of any inspector [4].
  + **Hover tool:** Hovering over the plot displays values of the respective point. We have shown only the year, but multiple attributes can be shown by customizing it.
* **Sub-division drop down list:** The dropdown is used for selecting the sub-division. The plot is shown based on the selected sub-division and further investigations can be made on the selected sub-division.
* **Distribution drop down list:** Type of plot is another important feature in here, which is used for specifying the type of plot such as discrete plot, smooth plot, etc. for visualization of the data.
* **Working mechanism**

The plot works by clicking on each option and customizing it based on the requirement. We can select the Sub-division and the type of distribution for the plot from the right pane of the window and the month are displayed in the right of the plot. Clicking on each month will add or remove the plot of the selected month. In Figure 12 we have selected PUNJAB as the sub-vision and distribution as smooth. The plot shows the data of months January, April, May, July, September and November which are selected. Similarly, we can investigate and explore the rainfall patterns of different regions between years and months.



**Insights:**

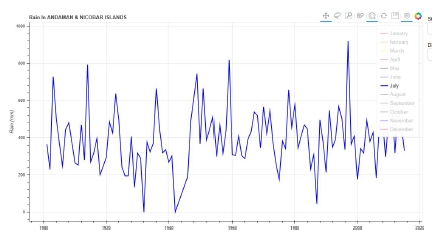
1. **Smooth viz of rainfall of Andaman and Nicobar Islands for all the months**



From the plot, we can see a maximum record of rainfall has happened in the month of July.

Above visualization shows the overall smooth value for all the 12 months for a particular state (Andaman and Nicobar Islands). Each state has its own plot, which could be changed accordingly. On the right-hand side there are two drop down list which could be used for selecting the desired state or the type of plot.

1. **Discrete viz for a particular month**



As July has the maximum record of rainfall, we take a closer look on

this month. We can see from the plot that the mean and variance has been

pretty constant throughout the month without much of a deviation.

Above visualization is the plot for just one month for a given state.

Other states data has been hidden so that each months’ rainfall history

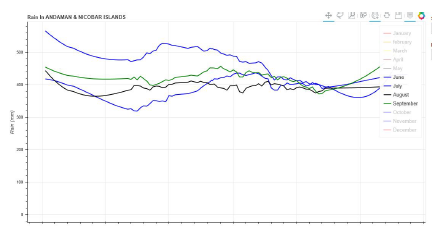
can be checked. This could be particularly useful when seasonal

rains have to be checked. Since India has four seasons, this method

is useful for understanding the monthly patterns and can be used for better

prediction of the seasonal rainfall.

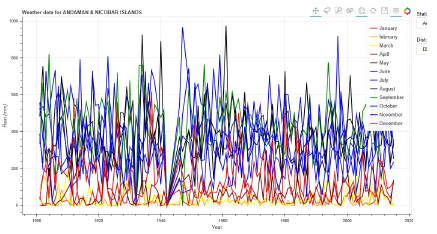
1. **Highest rainfall months**



Here we take a closer look at the monsoon season (period of June to September) where the rainfall is usually higher in the region of Andaman and Nicobar Islands. We could see that rainfall in July > September > August > June.

This plot shows the months which had the highest amount of rainfall in a given state/region. So, users can easily find out highest amount of rainfall month-wise and can be used to analyze the rain or drought patterns.

1. **All rainfall since 1901 to 2015 in discrete form**

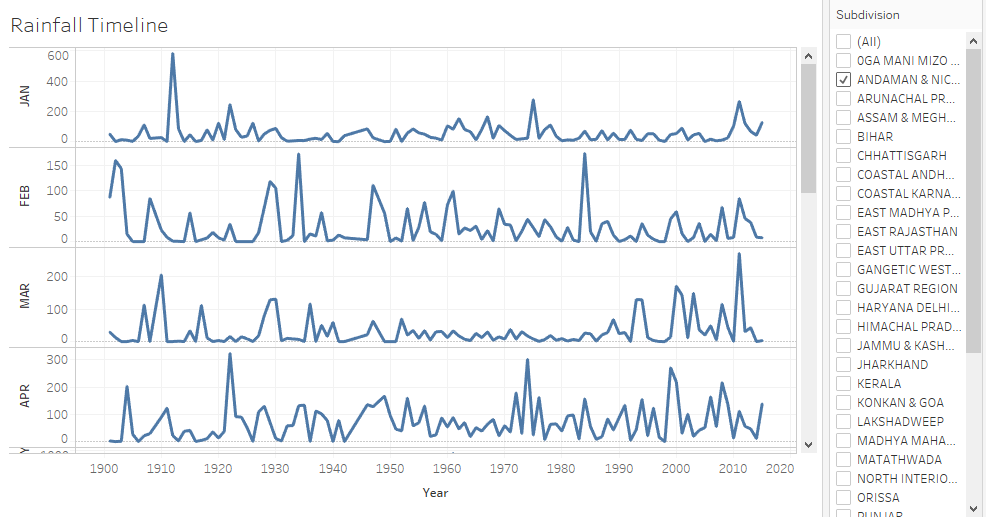


We can see the highest record of rainfall during the period of 1950‘s and 1960’s in the months of July and August, followed by August in 1930 and July in 2000. However, during 1941, we can see some dry conditions in Andaman with the least record of rainfall.

**Video demonstrating the features implemented for data visualization**

****

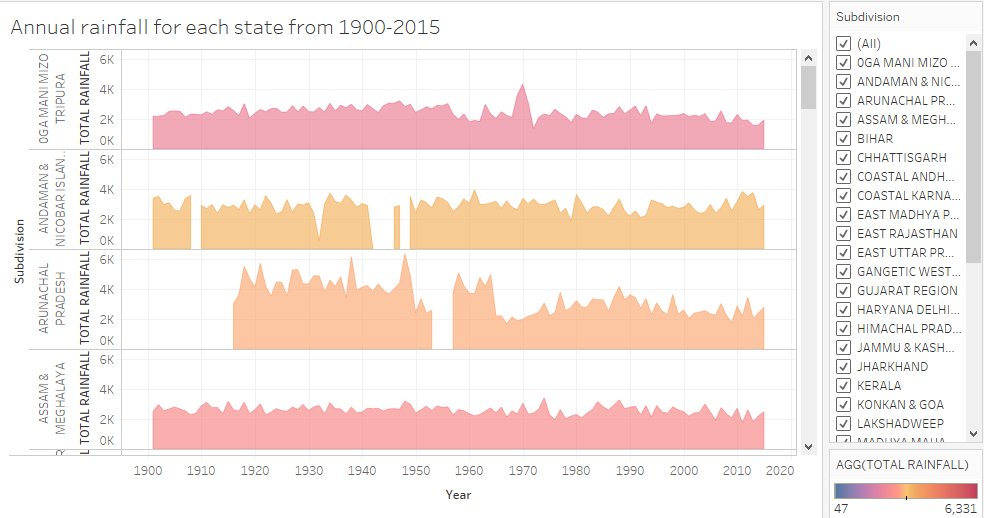
1. **Tableau:** Unlike Python, which involves coding, Tableau is an UI-based platform for data visualization. From connection through collaboration, It is the most powerful, secure, and flexible end-to-end UI-based analytics platform for data visualization [3]. With its versatility in handling the data, and simple, yet powerful techniques, Tableau still ranks on the top as it is considered one of the best reporting tools for business intelligence. With the ability to create storyboards, leveraging the power of database it is considered one of the ready-to-go tools for data analysis to provide insights for better understanding of the data.
2. **Timeline of rainfall across all the years for each month by state**



Here we visualize the record of rainfall for each month separately across all the years by specifying the appropriate region to be visualized.

1. **Annual rainfall for each state from 1900-2015**

The dataset contains the data for each month individually as a column. We needed the report for annual rainfall so, we used the calculated field to calculate the rainfall for each year from January to December and stored it in an individual measure named TOTAL RAINFALL. With this field, we plotted a graph showing the annual rainfall of each region. The color code has been selected based on the TOTAL RAINFALL measure.

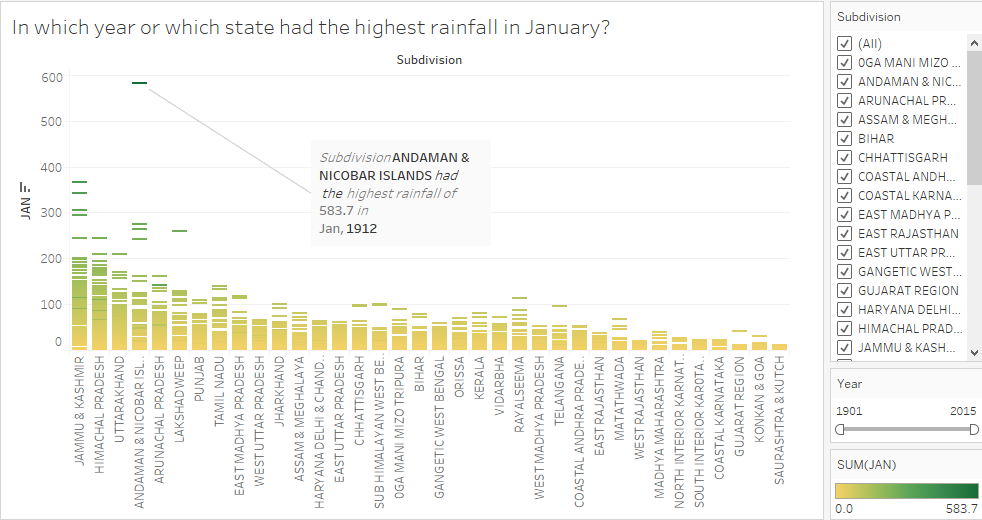
****

Here, we visualize the annual rainfall that each of the states have received during the period of 1901-2015. With the colors ranging from blue to red, (blue being the least and red being the highest)

* We can see that the Arunachal Pradesh region has the record of the highest rainfall in India during the period of 1945-50.
* While, West Rajasthan has the record for the least amount of rainfall throughout the period making it a dry region.

1. **In which year, which state had the highest rainfall**

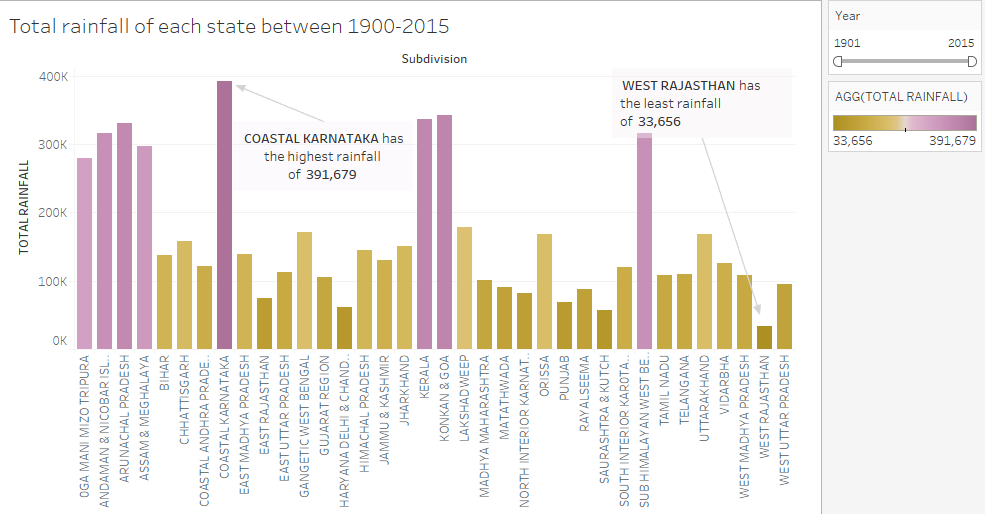
We used the monthly rainfall data to visualize the year to identify the region that got the highest rainfall in January. The highest rainfall for other months also can be visualized by selecting the specific month as the parameter. Using this we can notice the years to identify which regions got highest rainfall and compare it with other region’s rainfall pattern.

****

Through this visualization, we can see that in the year of 1912 in January, Andaman and Nicobar Islands had received the highest rainfall of 583.7mm, while Saurashtra & Kutch (Rajasthan region) had received the least rainfall.

1. **Total rainfall of each state from 1900-2015**

We used the calculated field we created to visualize the total rainfall for each sub-division between the years 1900 to 2015. From this graph we can identify the regions which got the highest and least rainfall historically. Based on the plot, we can see that Coastal Karnataka had received the highest total rainfall of 391,679, while West Rajasthan received the least rainfall of 33,656 between years 1900-2015. This also shows us how significantly rainfall patters vary across one country with respect to the region.



It is to be also noted that although Arunachal Pradesh had the record of highest annual rainfall and Andaman had the highest rainfall at a given time, the Coastal regions of Karnataka tops the chart throughout the period of 1901-2015 in receiving the maximum amount of rainfall. While, the West Rajasthan has received the least amount of rainfall during this period.

**4. KEY INSIGHTS**

* Highest rainfall (583.7mm) is recorded in Andaman and Nicobar Islands in January 1912
* Arunachal Pradesh has recorded the highest annual rainfall during the periods of 1945-50
* Coastal Karnataka has recorded the maximum rainfall throughout the period of 1901-2015
* West Rajasthan ranks at the bottom in almost all cases making it the driest state in India with the least amount of rainfall.
* Monsoon Rainfall of 764.9 mm was observed during Jun-Sep of 2015. We had observed decline of -2.15% in rainfall during Jun-Sep of 2015 over the same period of 2014.
* Monsoon Rainfall of 864.4 mm was observed during Jun-Sep of 2016. We had observed growth of 13.01% in rainfall during Jun-Sep of 2016 over the same period of previous year.
* Monsoon (Jun-Sep) rainfall shows no noticeable trend from 1901 to 2016. During this period Maximum monsoon rainfall of 1124.2 mm was observed in the year 1917 and Minimum monsoon rainfall of 697.4 mm was observed in the year 1972.

**5. CONCLUSION**

We were able to obtain some interesting insights from our rainfall data of India through data visualization with the help of tools like code-based Bokeh library and UI-based Tableau.

While this was data was chosen initially hoping to see a drastic or interesting trend of the rainfall pattern over 100 years, through our exploratory data analysis, we could see no such drastic change in the trend of rainfall. It has remained pretty constant throughout this period across all the regions with a constant mean and variance.

**6. REFERENCES**

[1] Anon, (n.d.). Open Government Data (OGD) Platform India.  
[online] Available at: <https://data.gov.in/>

[2] Anon, (n.d.). Welcome to Bokeh â˘AˇT Bokeh 0.12.15 documentation. [online] Available at: <https://bokeh.pydata.org/en/latest/>

[3] Tableau. [online] Available at: <https://www.tableau.com/>

[4] Configuring Plot Tools in Bokeh [Online] Available at: <https://bokeh.pydata.org/en/latest/docs/user_guide/tools.html#userguide-tools-pandrag>

**APPENDIX**

Python (Bokeh) code used for data visualization

#from os.path import join, dirname

import datetime

from bokeh.models import OpenURL, TapTool,HoverTool

from bokeh.layouts import gridplot

import pandas as pd

from scipy.signal import savgol\_filter

from bokeh.io import curdoc

from bokeh.layouts import row, column

from bokeh.models import ColumnDataSource, DataRange1d, Select

from bokeh.palettes import Spectral6,Greys256

from bokeh.plotting import figure

All\_record = ['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN','JUL','AUG','SEP','OCT','NOV','DEC']

def get\_input(src, name, distribution):

df = src[src.SUBDIVISION == name].copy()

del df['SUBDIVISION']

#df['YEAR'] = pd.to\_datetime(df.YEAR)

# timedelta here instead of pd.DateOffset to avoid pandas bug < 0.18 (Pandas issue #11925)

df['YEAR']=df.YEAR

#df['left'] =df.YEAR - datetime.timedelta(days=.5)

#df['right'] = df.YEAR + datetime.timedelta(days=.5)

df = df.set\_index(['YEAR'])

df.sort\_index(inplace=True)

if distribution == 'Smoothed':

window, order = 51, 3

#for key in All\_record:

# exec("%s = %d" % (codes[code][0], code))

for key in All\_record:

df[key] = savgol\_filter(df[key], window, order)

return ColumnDataSource(data=df)

def make\_plot(source, title):

plot = figure(tools="pan,wheel\_zoom,box\_zoom,reset,tap,lasso\_select,save,hover",toolbar\_location="above",plot\_width=1100, plot\_height=600)

#plot2=figure()

plot.title.text = title

legends=["January","february","March","April","May","June","July","August","September","October","November","December"]

color\_list=["Red","orange","yellow","maroon","black","blue","blue","black","green","blue","blue","Red"]

for i,j,k in zip(All\_record,legends,color\_list):

plot.line(x='YEAR',y=i,source=source,color=k, legend=str(j),line\_cap='butt',line\_width=2)

# plot.line(x='YEAR',y='JAN',source=source,color="black", legend="january",line\_cap='butt',line\_width=4)

# plot.line(x='YEAR',y='FEB',source=source,color="yellow", legend="February")

# plot.line(x='YEAR',y='MAR',source=source, legend="March",color="green")

# plot.line(x='YEAR',y='APR',source=source, legend="April",color="red")

# plot.line(x='YEAR',y='JUN',source=source, legend="june",color="orange")

# plot.line(x='YEAR',y='JUL',source=source, legend="july")

# plot.line(x='YEAR',y='AUG',source=source, legend="august")

# plot.line(x='YEAR',y='SEP',source=source, legend="september")

# plot.line(x='YEAR',y='OCT',source=source, legend="october")

# plot.line(x='YEAR',y='NOV',source=source, legend="november")

# plot.line(x='YEAR',y='DEC',source=source, legend="december")

url = "https://data.gov.in/keywords/annual-rainfall"

taptool = plot.select(type=TapTool)

taptool.callback = OpenURL(url=url)

#chup=source.\_df\_index\_name

plot.xaxis.axis\_label = "Year"

plot.yaxis.axis\_label = "Rain (mm)"

plot.legend.click\_policy="hide"

hover = plot.select(dict(type=HoverTool))

hover.tooltips = [("Year", "@YEAR"),]

hover.mode = 'mouse'

return plot

def update\_plot(attrname, old, new):

city = city\_select.value

plot.title.text = "Rain In " + cities[city]['SUBDIVISION']

src = get\_input(df, cities[city]['SUBDIVISION'], distribution\_select.value)

source.data.update(src.data)

city = 'ANDAMAN & NICOBAR ISLANDS'

distribution = 'Discrete'

type\_plot='circle'

cities = {

'ANDAMAN & NICOBAR ISLANDS': {

'SUBDIVISION': 'ANDAMAN & NICOBAR ISLANDS',

'title': ' WA', },

'ARU0CHAL PRADESH': {

'SUBDIVISION': 'ARU0CHAL PRADESH',

'title': ' MA',

},

'ASSAM & MEGHALAYA': {

'SUBDIVISION': 'ASSAM & MEGHALAYA',

'title': ' WA',

}

,

'BIHAR': {

'SUBDIVISION': 'BIHAR',

'title': ' WA',

}

,

'CHHATTISGARH': {

'SUBDIVISION': 'CHHATTISGARH',

'title': ' WA',

}

,

'COASTAL ANDHRA PRADESH': {

'SUBDIVISION': 'COASTAL ANDHRA PRADESH',

'title': 'WA',

}

,

'COASTAL KAR0TAKA': {

'SUBDIVISION': 'COASTAL KAR0TAKA',

'title': 'WA',

}

,

'EAST MADHYA PRADESH': {

'SUBDIVISION': 'EAST MADHYA PRADESH',

'title': 'WA',

}

,

'ORISSA': {

'SUBDIVISION': 'ORISSA',

'title': 'WA',

},

'PUNJAB': {

'SUBDIVISION': 'PUNJAB',

'title': 'WA',

},

'RAYALSEEMA': {

'SUBDIVISION': 'RAYALSEEMA',

'title': 'WA',

},

'UTTARAKHAND': {

'SUBDIVISION': 'UTTARAKHAND',

'title': 'WA',

}

}

city\_select = Select(value=city, title='State', options=sorted(cities.keys()))

distribution\_select = Select(value=distribution, title='Distribution', options=['Discrete', 'Smoothed'])

#plot\_type = Select(value=type\_plot, title='type of graph', options=['line', 'circle'])

df = pd.read\_csv('datafile\_Actual\_1.csv')

source = get\_input(df, cities[city]['SUBDIVISION'], distribution)

plot = make\_plot(source, "Weather data for " + cities[city]['SUBDIVISION'])

city\_select.on\_change('value', update\_plot)

distribution\_select.on\_change('value', update\_plot)

#plot\_type.on\_change('value', update\_plot)

controls = column(city\_select, distribution\_select)

curdoc().add\_root(row(plot, controls))

curdoc().title = "Rain in India"