Chapter 3

Introduction

In this chapter, we discuss the data collection, listing of data properties and snapshot of collected dataset, data manipulation and pre-processing, model selection, and brief description of corresponding models, algorithms of model building and training and, generate uncertainty from predicted data, discuss tactics of CA, mechanism of slicing streamgraph and replace with CA representation, show examples of uses of CA in real world charts.

3.1 Data

Authentic data is the most important part in the data visualization research. Without having an authentic dataset research cannot be started properly and without following a smart data preparation strategy such as cleaning, validating, and consolidating raw data in the right way, research cannot succeed in the long run.

3.1.1 Data Collection

Data collection is the process of gathering, measuring, and analyzing accurate information on variables of interest, in an established systematic manner that enables one to pursue one to conduct research in right direction. Due to the global impact of pandemic, different individuals, organizations, or governments are storing data in their own way. After going through different repositories, we found that the complete and authentic data is bundled in ourworldindata.org at csv format. The following table shows the list of fields/properties of each record where many of them are not relevant to our research. For example: date, location, new_cases, total_cases are some of the useful attributes bolded in the following table.

iso_code	continent	**location**
date	**total_cases**	**new_cases**
new_cases_smoothed	**total_deaths**	**new_deaths**
new_deaths_smoothed	total_cases_per_million	new_cases_per_million

new_cases_smoothed_per_million	population_density	new_deaths_per_million
new_deaths_smoothed_per_million	stringency_index	population
new_cases_smoothed_per_million	median_age	aged_65_older
aged_70_older	gdp_per_capita	extreme_poverty
cardiovasc_death_rate	diabetes_prevalence	female_smokers
male_smokers	handwashing_facilities	hospital_beds_per_thousand
life_expectancy	human_development_index	

Table-1: COVID Data property list

3.1.2 Sample Data

iso_code	location	date	total_cases	new_cases	total_deaths	new_deaths	new_tests	total_tests	new_vaccinations	population
USA	United States	31/1/21	26249342	112152	449340	1862	943985	309077233	1545397	331002647
USA	United States	1/2/21	26384317	134975	451420	2080	1032022	310109255	1099103	331002647
USA	United States	2/2/21	26499620	115303	454846	3426	1632097	311741352	558458	331002647
USA	United States	3/2/21	26621311	121691	458728	3882	1869029	313610381	1097394	331002647
USA	United States	4/2/21	26745317	124006	462473	3745	1928145	315538526	1325456	331002647
USA	United States	5/2/21	26879739	134422	466138	3665	1829259	317367785	1615502	331002647
USA	United States	6/2/21	26983915	104176	468821	2683	1332964	318700749	2218752	331002647
USA	United States	7/2/21	27073661	89746	470257	1436	828602	319529351	2172973	331002647
USA	United States	8/2/21	27164099	90438	471877	1620	1042164	320571515	1206680	331002647
USA	United States	9/2/21	27259364	95265	474927	3050	1691635	322263150	788573	331002647
USA	United States	10/2/21	27354614	95250	478231	3304	1638478	323901628	1563780	331002647
USA	United States	11/2/21	27460378	105764	481447	3216	1414964	325316592	1620300	331002647
USA	United States	12/2/21	27560048	99670	484374	2927	1411948	326728540	2020288	331002647
USA	United States	13/2/21	27647267	87219	486570	2196	1101360	327829900	2231326	331002647
USA	United States	14/2/21	27712402	65135	487741	1171	663849	328493749	2242472	331002647
USA	United States	17/2/21	27899318	70139	492854	2397	1470698	332013338	1061463	331002647
USA	United States	18/2/21	27969229	69911	495370	2516	1414506	333427844	1455940	331002647
USA	United States	19/2/21	28048511	79282	497994	2624	1353482	334781326	1847276	331002647
USA	United States	20/2/21	28120207	71696	499829	1835	1058252	335839578	1704457	331002647
USA	United States	21/2/21	28177359	57152	501065	1236	683633	336523211	1801134	331002647
USA	United States	22/2/21	28233518	56159	502384	1319	1038714	337561925	1086840	331002647
USA	United States	23/2/21	28305788	72270	504661	2277	1643226	339205151	854609	331002647
USA	United States	24/2/21	28380537	74749	507843	3182	1687287	340892438	1432864	331002647
USA	United States	25/2/21	28458041	77504	510279	2436	1611102	342503540	1809170	331002647
USA	United States	26/2/21	28535390	77349	512357	2078	1475665	343979205	2179947	331002647
USA	United States	27/2/21	28600016	64626	513878	1521	1074248	345053453	2352116	331002647
USA	United States	28/2/21	28651438	51422	514970	1092	633047	345686500	2429823	331002647
USA	United States	1/3/21	28709536	58098	516487	1517	993543	346680043	1663984	331002647
USA	United States	2/3/21	28766634	57098	518430	1943	1611124	348291167	1731614	331002647
USA	United States	3/3/21	28833825	67191	520911	2481	1736631	350027798	1908873	331002647
USA	United States	4/3/21	28901885	68060	522833	1922	1551765	351579563	2032374	331002647
USA	United States	5/3/21	28968304	66419	524636	1803	1430138	353009701	2435246	331002647
USA	United States	6/3/21	29026558	58254	526146	1510	1066184	354075885	2904229	331002647
USA	United States	7/3/21	29067631	41073	526848	702	608815	354684700	2439427	331002647
USA	United States	8/3/21	29112548	44917	527585	737	976094	355660794	1738102	331002647
USA	United States	9/3/21	29170215	57667	529391	1806	1540048	357200842	1602746	331002647

Table-2: screenshot of sample data

In the above Table-2, it shows only a snapshot of whole dataset where there are hundreds of thousands of records for Covid data for more than 237 countries and territories. Though there are numerous fields in the data, we only needed few of them as listed in previous section. The dataset is collected as a excel file which includes daily occurances and/or counts of all properties. The total_* fields like total_cases, total_deaths, etc are cumulative and so every day that is updated with previous day's counts. Data is ordered by date and name of the country correspondingly. If there is no value in a cell for certain date for a country, that cell is kept empty, so that is needed to handle during data processing.

3.2 Machine Learning Algorithms

Although we have not done anything novel in machine learning domain, it is necessary to briefly introduce the salient algorithms that have used in our research to process the available data and generate the uncertainties of predictions since uncertainty representation is our prime concern.

3.2.1 Predictive/Forecasting Models

A time series forecasting model comprises a sequence of data points captured, using time as the input parameter. It uses the historical data to develop a numerical metric and predicts values for the next duration, for instance, data for the next few weeks using that metric.

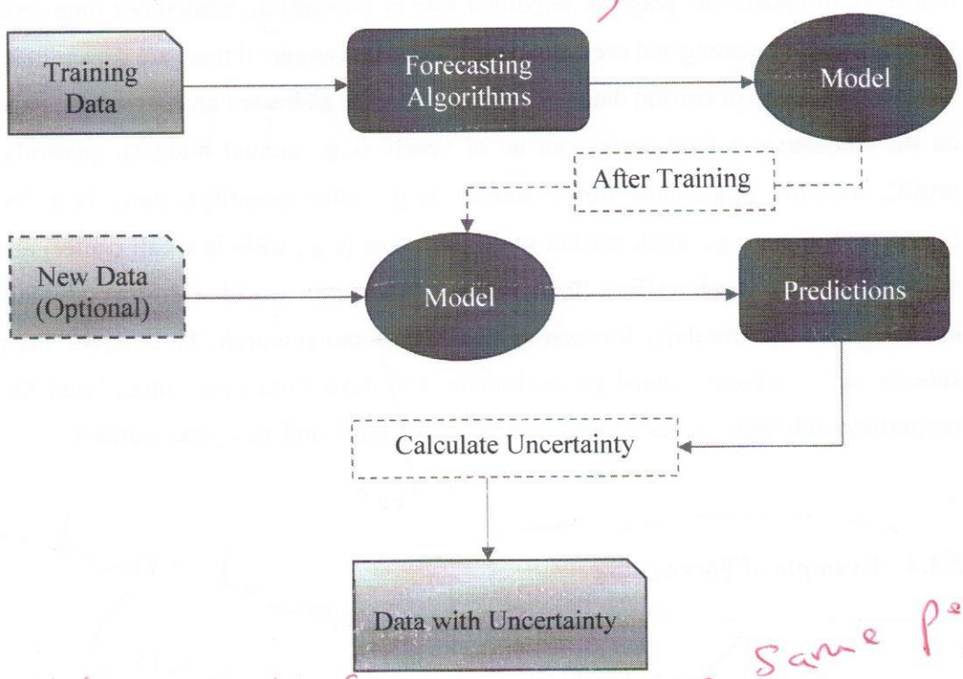


Figure-3: Predictive modeling workflow to generate uncertainty

3.2.2 Time Series Analysis vs Forecasting

Sometimes ambiguity arises between time series analysis with time series forecasting when working with temporal data. As per Song et al [1] and Beneditto el al. [4] in time series analysis, a time series is modeled to determine its components in terms of seasonal patterns, trends, and relation to external factors. In contrast, time series forecasting, Gecili et al. [6] and Brownlee [32] uses the information in a time series (perhaps with additional information) to forecast future values of that series. The COVID-19 dataset is maintained on a global basis, so it is more trustworthy and ~~with time series~~ forecasting models can be considered as suitable for our research to ~~get~~ the predicted results and hence generate our required uncertainty data to represent chromatic aberration in visualization area.

3.2.3 Concerns of Forecasting

Time series forecasting is an important area of machine learning. It is important because there are so many prediction problems that involve real life issues ~~like~~ time component. In forecasting it is very important to understand the goal of the problem and the nature of the available data. For instance, the volume of data, time horizons (short, medium, or long term), frequency of update etc. plays an important role in forecasting. Sometimes time series data requires cleaning, scaling and even transformation, for example: if there are gaps/missing data, if there are outliers or corrupt data then those need to be addressed and corrected. Depending on the frequency, a time series can be of yearly (e.g., annual budget), quarterly (e.g., profit), monthly (e.g., cash flow), weekly (e.g., sales quantity), daily (e.g., weather forecast), hourly (e.g., stock market price), minutes (e.g., calls in a call canter) and even seconds wise (e.g., web traffic). ~~Being the covid pandemic world-wide concerns for whole humanity,~~ we use the daily forecast mechanism ~~to~~ our research. To compare the results side by side we have created prediction for 200 days from every model and for ~~some~~ properties such 'new_cases', 'new_deaths', 'new_tests', and 'new_vaccinations'.

3.2.4 Example of Forecasting

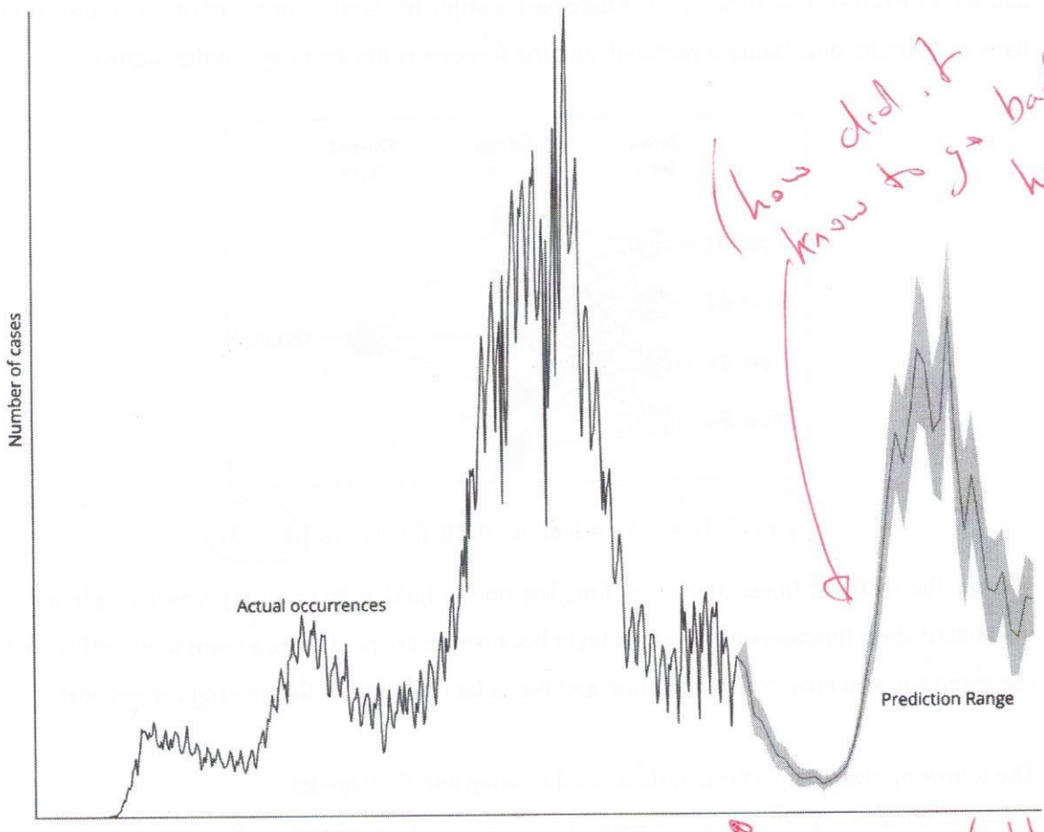


Figure-4: Example of daily covid forecasting for 200 days

The above Figure-4 shows the daily forecasting of number of new cases for United States based on previous statistics. So, in the blackish line in left shows the actual occurrences and the reddish line at right shows the predicted number of cases and greyed background surrounding the predicted line represents the ranges of model prediction, that means the model can predict a value between the lower and upper value for a certain day and that grey area represents the area of uncertainty.

3.3 MLP

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). It is a neural network connecting multiple layers in a directed graph, which means that the signal passes through the nodes only in one direction. It can be used for time series forecasting by taking multiple observations at prior time steps, called lag observations, and using them as input features and predicting one or more-time steps from those observations. The training

dataset is therefore a list of samples, where each sample has some number of observations from days prior to the time being forecasted, and the forecast is the next days in the sequence.

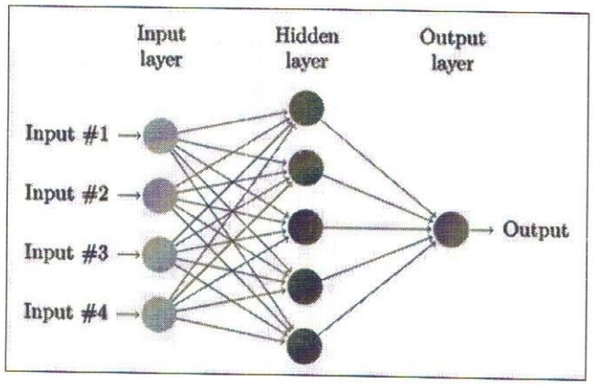


Figure-5: Basic Architecture of MLP network [ref. 33]

We use the rectified linear activation function on the hidden layer as it performs well and a linear activation function on the output layer because we are predicting a continuous value. We use mean squared error as loss function and the 'adam' optimizer for training the network.

The following steps shows the algorithm used to setup our MLP model:

---

1. Take an instance of 'Sequential' Model from Keras deep learning library.
2. Add a Dense layer to the model with stating number of inputs (24), number of nodes (500), number of epochs (100) and batch size (100), rectified linear activation function (relu).
3. Add another Dense layer with number of outputs (1), since we predict a continuous value.
4. Compile the model with mean square error (mse) loss function and 'adam' optimizer.
5. Fit the model with training data set for number of epochs (100) and batch size (100).
6. Make an ensemble of models by following the steps 1 to 5.
7. Get prediction 'yhat' for each time step (day) from all the models of the ensemble.
8. Calculate the ranges (lower bound, mean and upper bound) of each prediction.
9. Calculate uncertainty of the model for each day by using the set of yhats using the uncertainty calculating formula explained in section 3.7

Algorithm-1: MLP Model

3.4 CNN

Convolutional Neural Networks are a type of deep neural network developed for computer vision; for instance, two-dimensional image data, although they can be used for one-dimensional data such as sequences of text and time series forecasting. When operating on one-dimensional data, the CNN reads across a sequence of lag observations and learns to extract features that are relevant for making a prediction.

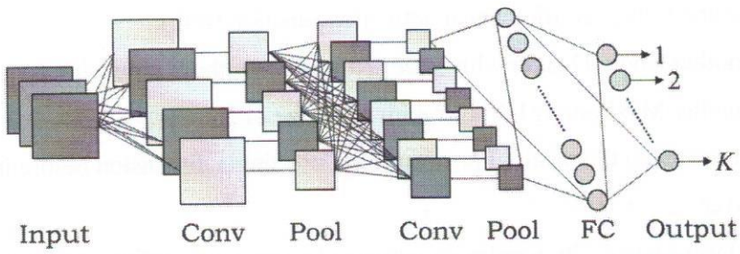


Figure-6: Basic Architecture of CNN network [ref. 34]

We define a CNN with two convolutional layers, one max-pooling layer, one flatten layer, and a dense layer from the input sequences. 1D convolution (Conv1D) layer. They have a configurable number of filters, kernel size, pool size and rectified linear activation function is used as loss function. The number of filters determines the number of parallel fields on which the weighted inputs are read and projected. A max pooling layer is used after convolutional layers to distill the weighted input features into those that are most salient, reducing the input size by 1/2. The pooled inputs are flattened to generate a long vector before being interpreted and used to make the prediction.

~~To dive into further~~ we ~~need to~~ briefly introduce some of the basic terms used in this model:

Conv1D:

A convolution layer transforms the input image in order to extract features from it. This layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs.

MaxPooling1D:

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. In other words, it downsamples the input representation by taking the maximum value over a spatial window of size (pool size). Thus, the output after

max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

The following steps shows the algorithm used to setup our CNN model:

---

1. Take an instance of 'Sequential' Model from Keras deep learning library.
2. Add a Conv1D layer to the model defining the number of filters (24), kernel size (500), input shape (100), rectified linear activation function (relu).
3. Add another Conv1D layer with same settings but without input shape.
4. Add another MaxPooling1D layer with pool size of 2.
5. Flatten (reshape) the result of previous step into single dimension before interpreted by the next layer.
6. Add a Dense layer with number of outputs (1), since we predict a continuous value.
7. Compile the model with mean square error (mse) loss function and 'adam' optimizer.
8. Fit the model with training data set for number of epochs (100) and batch size (100).
9. Create an ensemble of 6 models by following the steps 1 to 8.
10. Get prediction 'yhat' for each time step (day) from all the models of the ensemble.
11. Calculate the ranges (lower level, mean and upper level) of each prediction.
12. Calculate uncertainty of the model for each day by using the set of yhats using the uncertainty calculating formula explained in 3.7.

Algorithm-2: CNN Model

3.5 LSTM

The LSTM neural network is a member of RNN (Recurrent Neural Network) and it can be used for univariate time series forecasting. It uses an output of the network from a prior step as an input in attempt to automatically learn across sequence data. Having internal memory, LSTM allows it to accumulate internal state as it reads across the steps of a given input sequence.

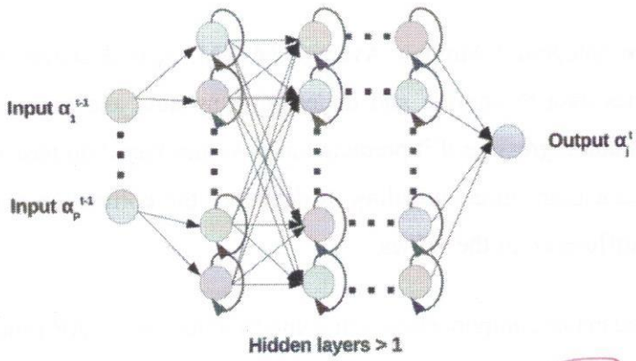


Figure-7: Basic Architecture of LSTM network (ref. 55)

For this model, we define a LSTM layer from inputs and subsequently two dense layers. Like other models, rrectified linear activation function is used in LSTM layer and in one of dense layer. A simple grid search of model hyperparameters was performed with the predefined configuration.

The following steps shows the algorithm used to setup our LSTM model:

---

1. Take an instance of 'Sequential' Model from Keras deep learning library.
2. Add an LSTM layer to the model defining the number of nodes (24), input shape (100), rectified linear activation function (relu).
3. Add a Dense layer for 24 input nodes and 'relu' activation function.
4. As we predict single value output, add a Dense output layer of 1 node.
5. Compile the model with mean square error (mse) loss function and 'adam' optimizer.
6. Fit the model with training dataset for number of epochs (100) and batch size (100).
7. Create an ensemble of 6 models by following the steps 1 to 8.
8. Get prediction 'yhat' for each time step (day) from all the models of the ensemble.
9. Calculate the ranges (lower level, mean and upper level) of each prediction.
10. Calculate uncertainty of the model for each day by using the set of yhats using the uncertainty calculating formula explained in 3.7.

Algorithm-3: LSTM Model

3.6 ARIMA

An Autoregressive Integrated Moving Average (ARIMA), is a statistical analysis model that uses time series data to analyse and understand the data and predict future trends. A statistical model is autoregressive if it predicts future values based on past values. It is a very popular technique for time series modeling. It describes the correlation between data points and considers the difference of the values.

The model has three major components which come from its name – AR (autoregressive term), I (Integrated term) and MA (moving average term). Let us briefly explain each of these components :

- AR term refers to predicting the next value using the prior values of dataset. The AR term is defined by the parameter 'p' in ARIMA.
- Integrated(I) term represents the number of times the differencing operation is performed on series to make it stationary (i.e., data values are replaced by the difference between the data values and the previous values). Test like ADF can be used to determine whether the series is stationary and help in identifying the d value. Differencing is only needed if the series is non-stationary otherwise, no differencing is needed, and in that case d=0.
- MA term is used to define number of prior/lagged forecast errors used to predict the future values. The parameter 'q' in ARIMA represents the MA term.

3.6.1 Auto ARIMA

Although ARIMA is a powerful model for forecasting time series data, the data preparation and parameter tuning processes end up being really time consuming. Before implementing ARIMA, it needs to make the series stationary, and determine the values of p and q as stated earlier. Auto ARIMA makes this complicated task simple for us as it eliminates those time-consuming tasks. Below are the steps for implementing auto ARIMA:

---

1. Load data: Collect data from the source repository and load into a data table.
2. Preprocess data: As the prerequisite of the model input is to be univariate, drop other columns from the data table and make sure all empty values replaced with NULL, so that system does not break down during runtime.
3. Build model by using open-source package named 'pmdarima'.

4. Fit the model on the univariate series of data generated in step-2 and using parameters test='adf', p=3, q=3 to get optimal value of 'd'.
5. Make predictions from the model for 200 days like other models.
6. Calculate series by using the forecasted results in previous step and with the help of panda.Series method.
7. Find the lower and upper bound of the series which will be used to calculate the uncertainties of the prediction.

Algorithm-4: ARIMA Model

3.7 Uncertainty Data Generation

Uncertainties are calculated from the ranges of predicted values for every time step (day) during the specified 200 days of forecasting period. That means we have a lower bound, mean, and upper bound of the predictions for each time step. So, the difference between upper and lower limit is the grey area of model prediction. Then find the maximum difference to set out the domain of the difference. Finally, divide each difference by maximum difference and multiply by a scaling factor to keep the maximum result in single digit. Here is given the steps to find the uncertainties using the machine learning models:

---

1. Read data from filesystem (excel file) to Data-Frame
2. Select Fields for which we need to generate uncertainty data
3. Create Machine Learning model for MLP/CNN/LSTM
4. Split data into training and test set
5. Train model with training set
6. Use model to get predicted or forecasted results
7. Find uncertainties or prediction error from model
8. Continue step 3 to 7 for each field and each model
9. Store uncertainty data as json in filesystem

Algorithm-5: calculate uncertainty (used by all models)

3.7.1 Uncertainty Data Scaling

We have shown top-level algorithm in the above section to generate uncertainty data. Since the uncertainty values are larger to accommodate in display, so it needed to scale in certain level. The following pseudo code is used to scale the uncertainty data.

```
1. country_avg_error = pred_errors_of_all_dates/number_of_days
2. max_error = find_max_error(all_country_avg_errors)
3. scaling_factor = 7
4. country_uncertainty = country_avg_error * scaling_factor / max_error;
```

Algorithm-6: data scaling

3.7.2 Snapshot of uncertainty data

Since the pandemic affected all the countries of the world and there are more than 200 countries, so we have trained the models for top 100 countries which were infected severely. Based on that setup, we have sorted the countries by obtained uncertainties in both ascending and descending orders. The following two tables shows the top 10 uncertainty attaining countries and the bottom one shows the lowest 10 uncertainty attaining countries.

3.7.3 Top 10 uncertainty countries using MLP model

Country	Actual Count	Predicted Count	Uncertainty
United States	14,851,118	15,652,300	7.00
India	15,693,425	7,409,636	4.28
Brazil	7,219,982	7,409,636	3.64
Kazakhstan	667,009	651,009	2.43
France	2,088,610	2,307,005	2.15
Peru	432,034	546,901	1.28
Germany	1,700,161	1,599,684	1.21
Spain	1,542,012	1,510,467	1.07
Turkey	3,645,288	3,389,016	1.03
Argentina	2,352,216	2,450,255	1.02

Table-3: Top uncertainty countries in the ordered list

3.7.4 Lowest 10 uncertainty countries using MLP model

next page.

Country	Actual Count	Predicted Count	Uncertainty
Qatar	36,256	36,796	0.013
Albania	62,292	65,515	0.016
Estonia	90,950	89,900	0.017
Egypt	118,376	124,175	0.019
Moldova	103,270	101,832	0.019
Australia	161,819	147,134	0.021
Algeria	86,238	82,121	0.022
Singapore	178,151	175,400	0.025
North Macedonia	57,447	57,420	0.037
South Korea	277,584	274,766	0.037

Table-4: Lowest uncertainty countries in the ordered list

From the above two tables, it is ~~clearly~~ noticeable that uncertainty is completely independent ~~on~~ the number of cases (Actual Count). For example: United States has lower number of cases than India but achieved higher uncertainty than India. Again, Kazakhstan and France exhibit ~~same~~ behavior and if we examine other countries then ~~surely~~, we ~~will get more~~.

3.7.5 Uncertainty Comparison among Models

Country	MLP	CNN	LSTM	ARIMA
United States	7.00	7.00	3.44	7.00
India	4.28	0.61	7.00	3.52
Brazil	3.64	0.51	3.24	1.27
Kazakhstan	2.43	0.42	0.35	0.17
France	2.15	0.31	0.81	0.56
Peru	1.28	0.23	0.28	0.22
Germany	1.21	0.19	0.50	0.51
Spain	1.07	0.19	0.67	0.33
Turkey	1.03	0.19	1.21	0.30
Argentina	1.02	0.14	1.08	0.25

Table-5: Lowest uncertainty countries in the ordered list

From the above comparison table of three different machine learning models, we notice that the uncertainties greatly vary for each country based on the model. There is no country which has identical uncertainty values for all three models. Though the dataset used in each of the models in similar approach, the variation appears due to their internal mechanism of the model

algorithms. Since the model superiority examination is not our goal, we are not going to discuss further about it. We use the uncertainty data whatever we obtained from model prediction and uncertainty calculation methods.

3.8 Architecture of CA Representation

As we have seen lateral chromatic aberration example in Figure-1 (Chapter 1) where all lights with different wavelengths does not focus to the same convergent point because lights having shorter wavelength refract more than the lights with longer wavelength. Inspiring from that analogy, we can consider a circle represents the predicted number of new cases for a country in a specific day. But since there is associated uncertainty of the prediction, a single circle will not be sufficient to represent bivariate (number of cases and uncertainty) distribution. That's why we need a little change on the arrangement where instead of single circle if we use three different circles of RGB color channels, apply lateral shifting of the center of the circle by the amount of uncertainty and blend them together then the resultant outcome would be a perfect representation of CA. The following Figure-8 shows such a geometric arrangement of unit radius circle.

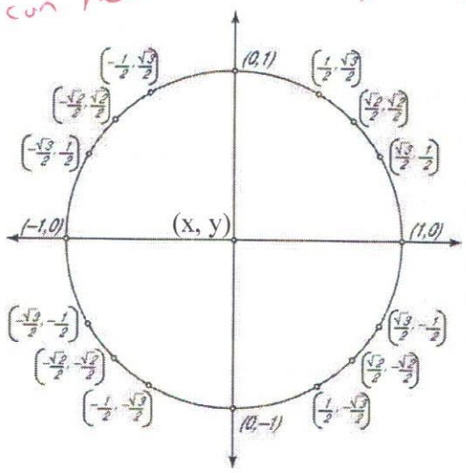


Figure-8: Underlying Geometry of CA

To draw a circle representing aberration as per above explanation if we draw 3 circles, let's call them 3 chromatic circles. Then we can conclude the technique with the following algorithm –

---

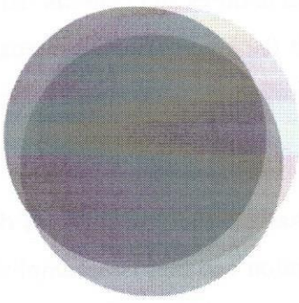
1. Let's consider the center of the target circle at (x, y).
2. Radius (radial offset) of the circle is 'r' represents uncertainty.

3. Draw first chromatic circle with color (R, 255, 255) with a shifted location of (x, y + r) where 'r' denotes red color channel.
4. Draw second chromatic circle with color (255, G, 255) with a shifted location of $(x + r * \frac{+\sqrt{3}}{2}, y + r * \frac{-1}{2})$ where 'G' denotes green color channel.
5. Draw second chromatic circle with color (255, 255, B) with a shifted location of $(x + r * \frac{-\sqrt{3}}{2}, y + r * \frac{+1}{2})$ where 'B' denotes blue color channel.
6. Apply css 'mix-blend-mode' to 'darken' to blend all three circles to get the resultant CA appearance.

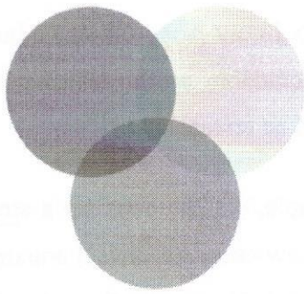
Algorithm-7: CA Construction Formula

3. 10 Examples of CA in Shapes

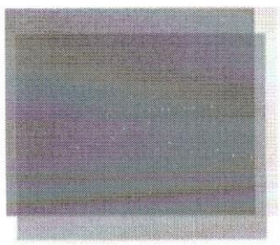
By using the above formula explained in section 3.9, a resultant aberration is presented with the uncertainty for the country India (IND) in Figure-9 below. Center dark-grey area represents the predicted number of new cases, and the bright colorful edges represent the amount of uncertainty in that prediction.



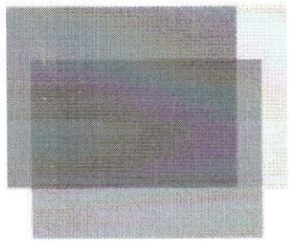
b. 10% uncertainty



a. 90% uncertainty



d. 7% uncertainty



c. 40% uncertainty

Figure 9: Example CA on Bubble and Rectangle

We have depicted the amount (in percentage) of uncertainties holds in every instance of the representation for a prediction. Same formula as explained in previous section (3.8) has been used to draw both the circular and rectangular shapes. We will show more examples in our user study representation and actual application of uncertainty visualization in different charts in next chapters.

3.10 Texture Pattern Generation

Texture is extremely common in modern web design, and it can be used in countless different ways in practical applications. Textures in web design can be very subtle, so that the visitor hardly notices, or they can be a central point of the design. In some cases, textures are used to emphasize or deemphasize certain parts of the design. Because of the versatility of textures, they can be used or generated in combination with many other design elements, such as typography, lighting, and colors. There is a subtle difference between patterns and textures. Patterns are visual element of geometric and mathematical structures that form consistent and repeated graphic shape on a surface. Visual activity across a surface is a texture when the structure forming the texture is based on irregular and random relationships over given areas. There are various kinds of textures and one of them is visual textures and patterns fall in that category. So, in our perspective, we build our textures with the help of SVG patterns where everyday predictions are presented with patterns and the collective outcome for the whole duration will be textures.

As an example, we can consider a streamgraph that emphasizes the prediction of daily of number of new cases for certain country for a specific duration and that is accomplished by filling the whole shape with flat color. But it doesn't make sense what was the uncertainty of the predictions, and to achieve this we can draw the graph with texture patterns by slicing them for smaller number of days. For example: Figure-10 show the depicts scenario explained here and underlying mechanism is explained in the later section.

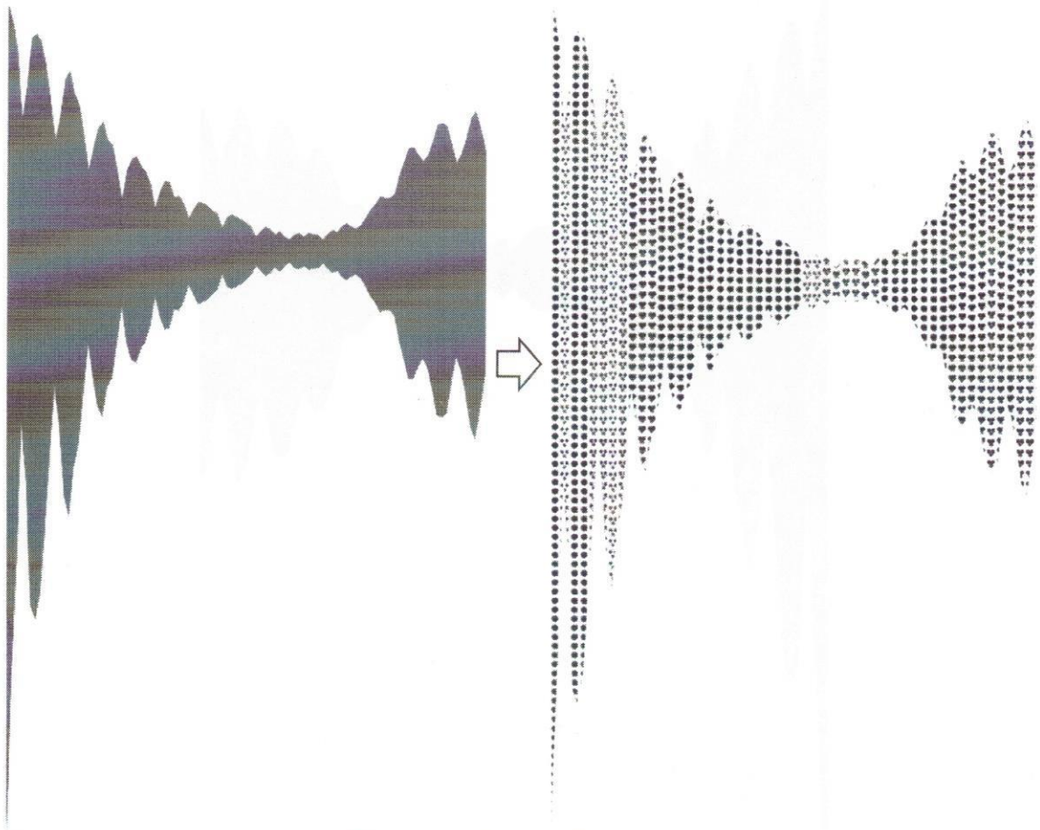


Figure 10: Streamgraph Color Filled (left), Texture Filled (right)

3.10.1 Slicing plot

In the above section, the streamgraph is shown ~~in~~ both color-filled version and texture-filled version. To better understand how the conversion is done ~~the following~~ Figure-11 gives a clear insight. We split the flow in horizontal direction and ~~made~~ a slice for every 3 days since horizontal axis represents the time in days. We have also tested by chopping the graph with other number of days like 2, 4, 5, 6, 7 and so on but 3 days gives best result among all options to pertain the shape and peaks of the curve. Because if we split it by 2 days then the width of the slice is too small to accommodate the content and if we use higher number of days then the shape of the plot undergoes ~~with~~ distortion and deteriorates the smoothness of the shapes such as peaks.

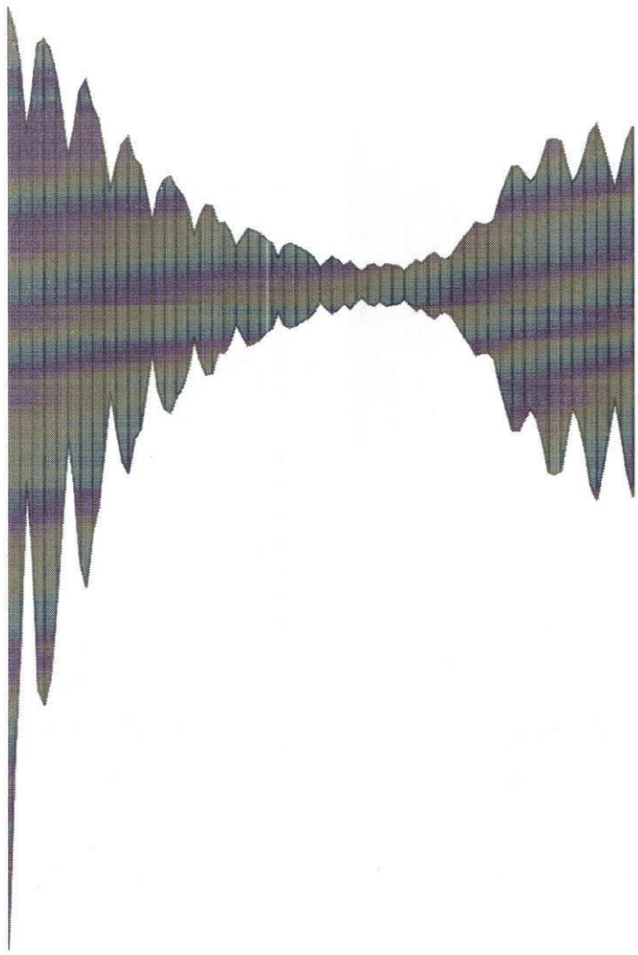


Figure-11: Sliced Streamgraph

Since each day of the duration has different value of prediction and ~~so the~~ uncertainties, we have averaged the prediction of uncertainties for every three days and presented the corresponding values for the columns. That means, every column in the representation shows different uncertainties where dark green bullets reveal lower uncertainty and light-green bullets represent higher uncertainties in Figure-11.

3.10.2 Pattern Generation Algorithm

---

1. Repeat steps 2 to 12 for all the countries
2. Define two sets of alternating colors to distinguish patterns side by side as follows:
   {0: '#ff0000', 1: '#800000', 2: '#FF00FF'} -> Reddish colors
   {0:'#008080 ', 1: '#0000FF', 2: '#000080'} -> Bluish colors
3. Repeat step 3 for three aberration points.
4. Repeat step 4 for three alternative colors.
5. Repeat steps 6 to 12 for 10 uncertainty scales [0 to 9].
6. Define a pattern by using <pattern> tag that defines a graphics object which can be redrawn at repeated x- and y- coordinate intervals to cover an area.
7. Set an 'id' of the pattern element. This 'id' is used later during filling texture. We use the following convention to define an id:
   pattern_id = 'pat-' + country + '-' + aber_indx + '-' + rgb_indx + '-' + uncertainty_scale
   where,
   country = name of the country of stream graph
   aber_indx = index of the aberration from [0, 1, 2] from step 3.
   rgb_indx = index of the color channel (0 for red, 1 for green, 2 for blue) from step 4.
   uncertainty = uncertainty in the scale of 0 to 9 (normalized to meet the range) from step 5.
8. Set width and height of the pattern.
9. Append a shape of the pattern such as 'circle', 'rect', 'ellipse' etc. In our case, we use 'circle'.
10. Set center (cx, cy) and radius (r) of the circle.
11. Set attribute 'patternUnits' to 'userSpaceOnUse' that defines the coordinate system for cx, cy, width, and height.
12. Fill the pattern by setting a fill color.

Algorithm-8: Pattern defining algorithm

3.10.3 Texture Generation Algorithm

---

1. Select the streamgraph container using d3 js.
2. Find the path of the streamgraph by from the value of 'd' property.

3. Divide the upper and lower segments of the path and save in two variables.
4. Determine the number of vertexes (coordinates) in each segment (they would be same).
5. Repeat steps 6 to 11 until all vertexes are traversed.
6. Take three vertexes from upper segment, let's call it p1.
7. Take three vertexes from lower segment, let's call it p2.
8. Build a new path string by joining p1 and p2 with standard rule of using M (moveto), L (lineto) and Z (closepath).
9. Append a new 'path' element into the container svg and set 'd' property by the path string.
10. Fill the path with the pattern id (generated by the previous algorithm) with the following syntax (value of fill attribute 'url(#pattern_id')).
11. Add blend style property 'mix-blend-mode' to 'darken'.

Algorithm-9: Texture generation algorithm