Render an agent response as a visualization

This product or feature is subject to the "Pre-GA Offerings Terms" in the General Service Terms section of the Service Specific Terms (/terms/service-terms#1). Pre-GA products and features are available "as is" and might have limited support. For more information, see the launch stage descriptions (/products#product-launch-stages).

This page shows how to use the Python SDK to render a visualization from the chart specifications that are provided within a Conversational Analytics API (/gemini/docs/conversational-analytics-api/overview) response. The sample code (#example-render-bar-chart) extracts the chart specification (in the Vega-Lite format (https://vega.github.io/vega-lite/)) from the response's `chart` field and uses the Vega-Altair (https://altair-viz.github.io/) library to render the chart, save it as an image, and display it.

Note: This guide assumes that you're working in an environment like Colaboratory. This guide also builds on the setup in Build a data agent using the Python SDK (/gemini/docs/conversational-analytics-api/build-agent-sdk), which shows how to authenticate and initialize the required `client`, `inline_context`, and `messages` variables.

Example: Render a bar chart from an API

This example shows how to render a bar chart from a Conversational Analytics API agent response. The example sends a request with the following prompt:

```
"Create a bar graph that shows the top five states by the total number of airpor
```

The sample code defines the following helper functions:

- `render_chart_response`: Extracts the Vega-Lite configuration from the `chart` message, converts it to a format that can be used by the Vega-Altair library, renders the chart, saves it to `chart.png`, and displays it.
- `chat`: Sends a request to the Conversational Analytics API using the `inline_context` variable and the current `messages` list, processes the streaming response, and if a chart is returned, calls `render_chart_response` to display it.

To use the following sample code, replace the following:

- *sqlgen-testing*: The ID of your billing project that has the required APIs enabled (/gemini/docs/conversational-analytics-api/enable-the-api).
- ***Create a bar graph that shows the top five states by the total number of airports***: The prompt that you want to send to the Conversational Analytics API.

```
from google.cloud import geminidataanalytics
from google.protobuf.json_format import MessageToDict
import altair as alt
import proto

# Helper function for rendering chart response
def render_chart_response(resp):
  def _convert(v):
    if isinstance(v, proto.marshal.collections.maps.MapComposite):
      return {k: _convert(v) for k, v in v.items()}
    elif isinstance(v, proto.marshal.collections.RepeatedComposite):
      return [_convert(el) for el in v]
    elif isinstance(v, (int, float, str, bool)):
      return v
    else:
      return MessageToDict(v)

  vega_config = _convert(resp.result.vega_config)
  chart = alt.Chart.from_dict(vega_config)
  chart.save('chart.png')
  chart.display()

# Helper function for calling the API
def chat(q: str):
  billing_project = "sqlgen-testing"
```

```
    input_message = geminidataanalytics.Message(
        user_message=geminidataanalytics.UserMessage(text=q)
    )

    client = geminidataanalytics.DataChatServiceClient()
    request = geminidataanalytics.ChatRequest(
        inline_context=inline_context,
        parent=f"projects/{billing_project}/locations/global",
        messages=messages,
    )

    # Make the request
    stream = client.chat (https://cloud.google.com/python/docs/reference/google-cloud-geminidataar

    for reply in stream:
      if "chart" in reply.system_message:
        # ChartMessage includes `query` for generating a chart and `result` with t
        if "result" in reply.system_message.chart:
          render_chart_response(reply.system_message.chart)

  # Send the prompt to make a bar graph
  chat("Create a bar graph that shows the top five states by the total number of a
```