# Workflow Demonstration

To provide a clearer understanding of how the VV library operates in a real-world scenario, this section walks through a complete workflow of data visualization automation. The illustration starts from obtaining data from a specific data source to rendering a chart and ultimately deploying it to a Miro board and Google Drive.

## Stage 1: Data Retrieval

In this example, we consider healthcare data obtained from Google Spreadsheets, which serve as our data source (see Figure X). The spreadsheets are organized into named ranges and contain essential metrics for Cox Proportional Hazards Models (see Figure X). These metrics include hazard ratios along with their corresponding confidence intervals for various covariates. The VV library leverages the GoogleSpreadsheetDatasetBuilder class to fetch this data, which is then transformed into a format suitable for chart rendering.

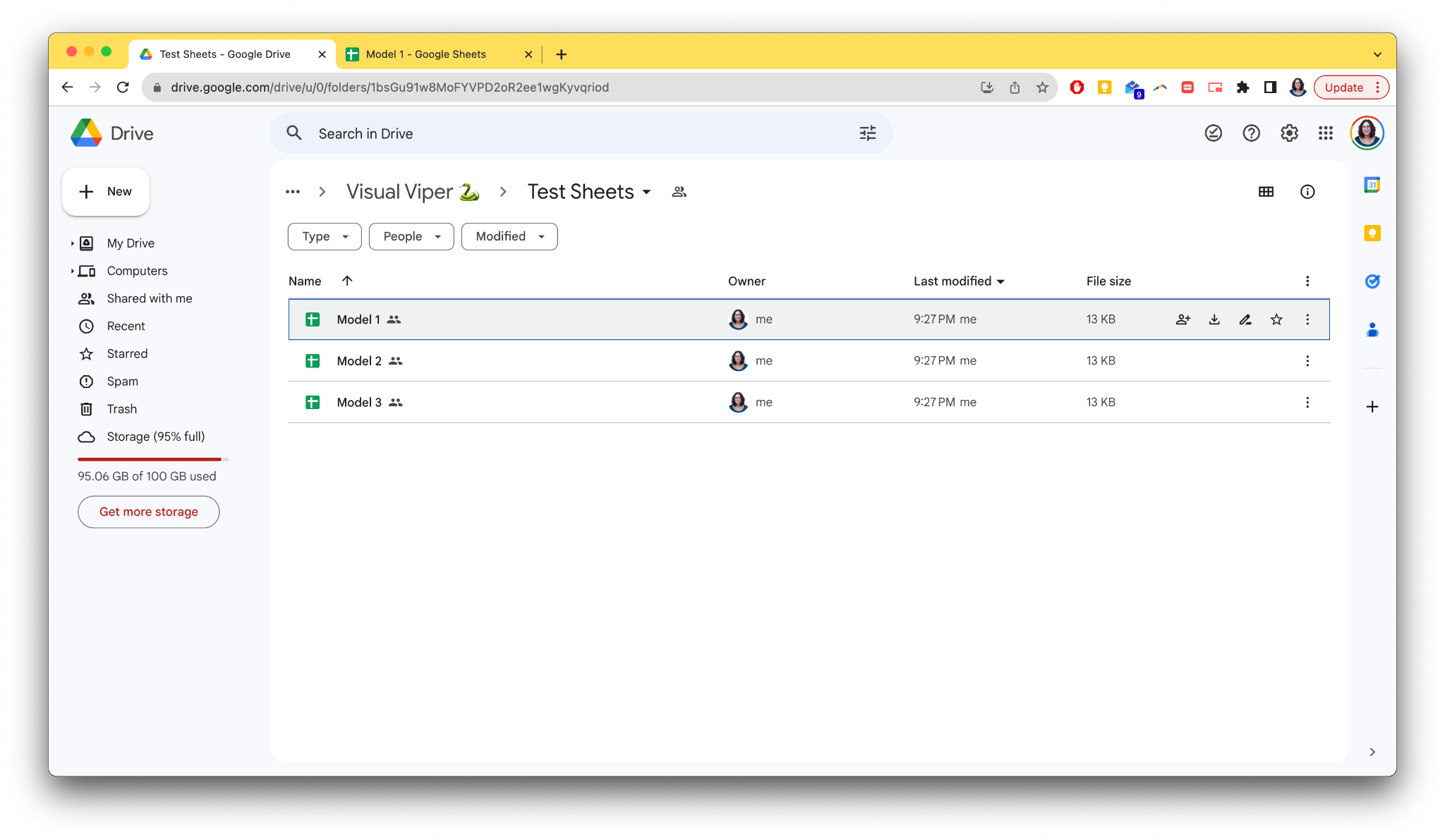


Figure X: Folder Containing Google Spreadsheets for the example.

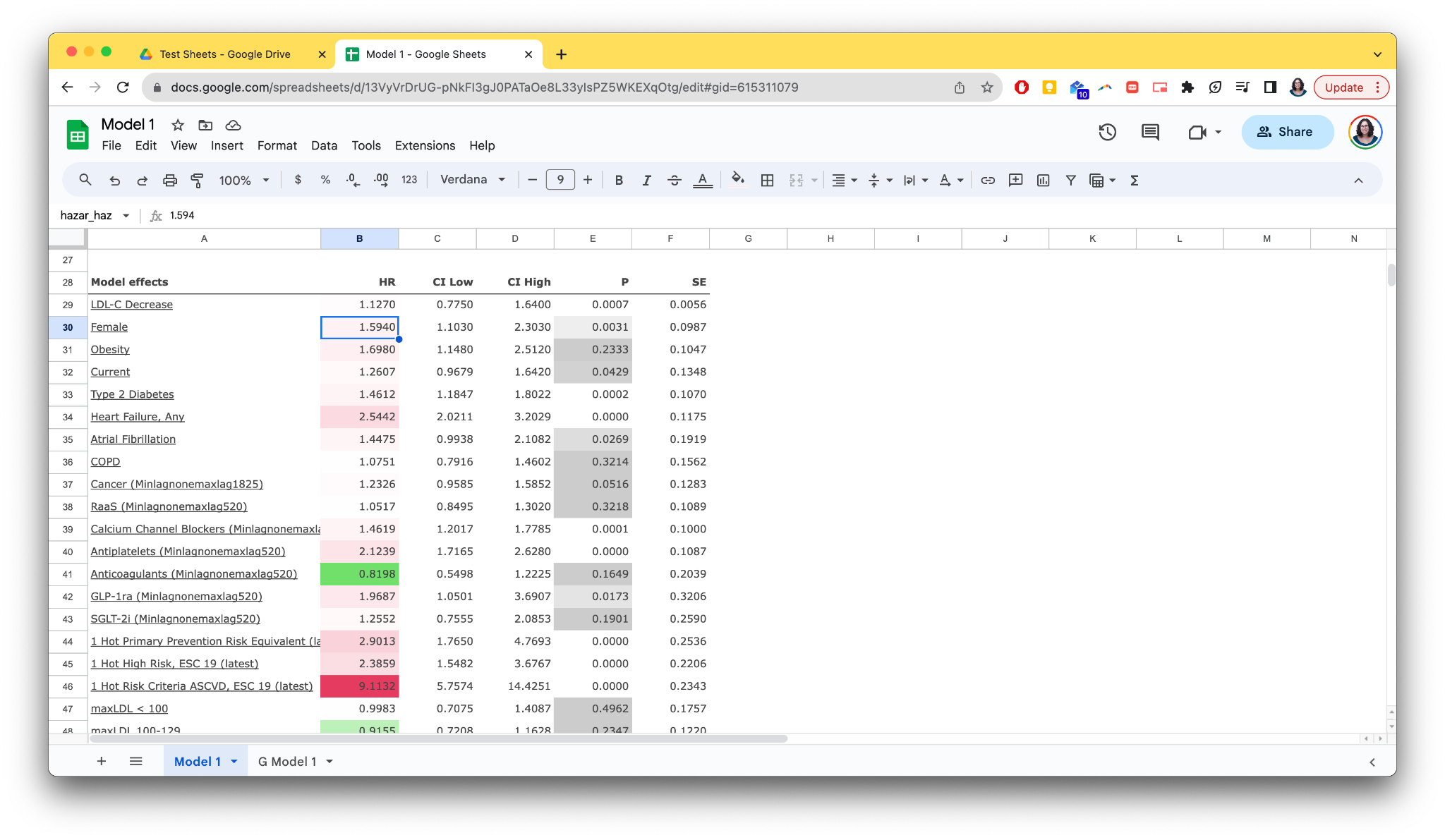


Figure X: Spreadsheet Content for Cox Proportional Hazards Model 1 of the example.

## Stage 2: Chart Configuration

Once the data is retrieved and prepared, the next step involves defining the chart specifications. For our example, we aim to visualize the metrics from the Cox Proportional Hazards Models in the form of a Forest Plot. VV library allows this by leveraging Vega-Lite, a high-level JSON syntax for generating visualizations.

To accomplish this, the ForestPlot class is employed. This class is a concrete implementation that inherits from AbstractChartNotationBuilder. It specializes in constructing Forest Plots by setting the necessary parameters, configurations, and data values. Moreover, the ForestPlotBinding class plays a vital role. This class inherits from AbstractChartNotation and is designed to hold and resolve the data points essential for a Forest Plot.

The JSON configuration for our Forest Plot, generated by the aforementioned classes, includes specific elements that are essential for visualizing the hazard ratios and their corresponding confidence intervals for the listed covariates (see Listing X). The JSON file lays out not only the type of chart to be generated but also fine-grains the aesthetic details such as titles, subtitles, and axes properties.

The configuration also takes advantage of Vega-Lite's layering capabilities. This enables us to represent multiple elements like the confidence intervals and hazard ratios within the same plot while maintaining visual coherence. Each metric, such as 'Age', 'Sex', 'Obesity', etc., is represented as a horizontal line in the Forest Plot, with markers indicating the confidence interval and a point indicating the hazard ratio. For this example we will use only three covariates.

Listing X: JSON Configuration for Forest Plot.

{

"$schema": "https://vega.github.io/schema/vega-lite/v5.json",

"data": {

"values": [

{"measure": "LDL-C decrease", "lo": 1.127, "hr":0.775, "hi": 1.64},

{"measure": "Age", "lo": 1.594, "hr": 1.103, "hi": 2.303},

{"measure": "Female", "lo": 1.698, "hr":1.148, "hi":2.512}

]

},

"title": {

"text": "Title 1",

"fontSize": 12,

"subtitle": "Subtitle 1"

},

"facet": {

"row": {

"field": "cohort",

"header": {

"labelAngle": 360,

"labelFontSize": 10.5

}

}

},

"spec": {

"encoding": {

"y": {

"field": "measure",

"type": "nominal",

"axis": {

"labelFontSize": 10

}

},

"x": {

"type": "quantitative",

"axis": {

"labelFontSize": 9

}

}

},

"layer": [

{

"mark": {

"type": "rule"

},

"encoding": {

"x": {

"field": "lo"

},

"x2": {

"field": "hi"

}

}

}

// Additional layers truncated for brevity

]

},

"config": {

"background": "#F7F7F7",

"font": "Barlow, Lato, Roboto, sans-serif"

}

}

## Stage 3: Chart Rendering

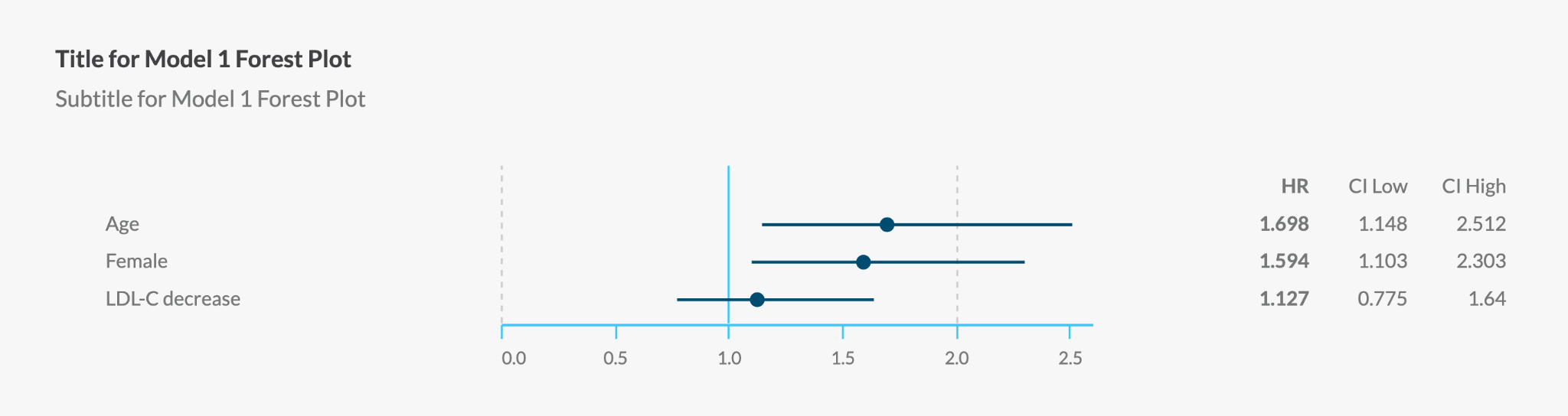
With the data properly set and the chart configuration in place, we are now ready to render the Forest Plot. To achieve this, we make use of altair-save, an external package that integrates with our architecture.

The core class responsible for this task is AltairChartRenderer, which extends the AbstractChartRenderer. This specialized class serves as a wrapper for Vega-Altair, utilizing the Altair library to perform the rendering of visualizations. In this architecture, the AltairChartRenderer takes the JSON configuration produced by ForestPlot and ForestPlotBinding classes and uses it to generate the visual representation of the Forest Plot.

In most of our workflows, the AltairChartRenderer outputs a file pointer (fp), typically an in-memory file-like object such as a StringIO object. This allows for easy manipulation and further use of the chart in the subsequent steps of deployment. However, the renderer is also flexible enough to output the chart as a saved image file, supporting various formats like SVG, for example.

In Figure X, you will find a sample of what the rendered Forest Plot looks like.

Figure X: Rendered Forest Plot for Model 1 of the example.



## Stage 4: Deployment

The final stage of the workflow involves deploying the rendered Forest Plot to a Miro board and Google Drive. To achieve this, the VV library employs the specialized classes MiroBoardDeployer and GoogleDriveDeployer.

Both classes automatically handle the upload process, ensuring that the visualizations are transferred to their designated platforms. This streamlined approach makes the visualizations readily accessible for team collaboration (see Figure X).

When deploying to a Miro board, the MiroBoardDeployer class offers additional layout capabilities. Specifically, it arranges the Forest Plots in a grid formation based on a user-defined number of columns. In our example, the Forest Plots are laid out in a two-column grid, facilitating a visually organized comparison of different plots (see Figure X).

For more extensive projects that require the deployment of a large number of Forest Plots, the MiroBoardDeployer is equally capable. It can layout tens of plots on the Miro board in an organized grid, allowing for seamless interpretation and analysis of a more extensive data set (see Figure X for a different example of Forest Plots deployed in Miro with tens of plots).

Figure X: Forest SVG files on Google Drive, uploaded by the Visual Viper agent.

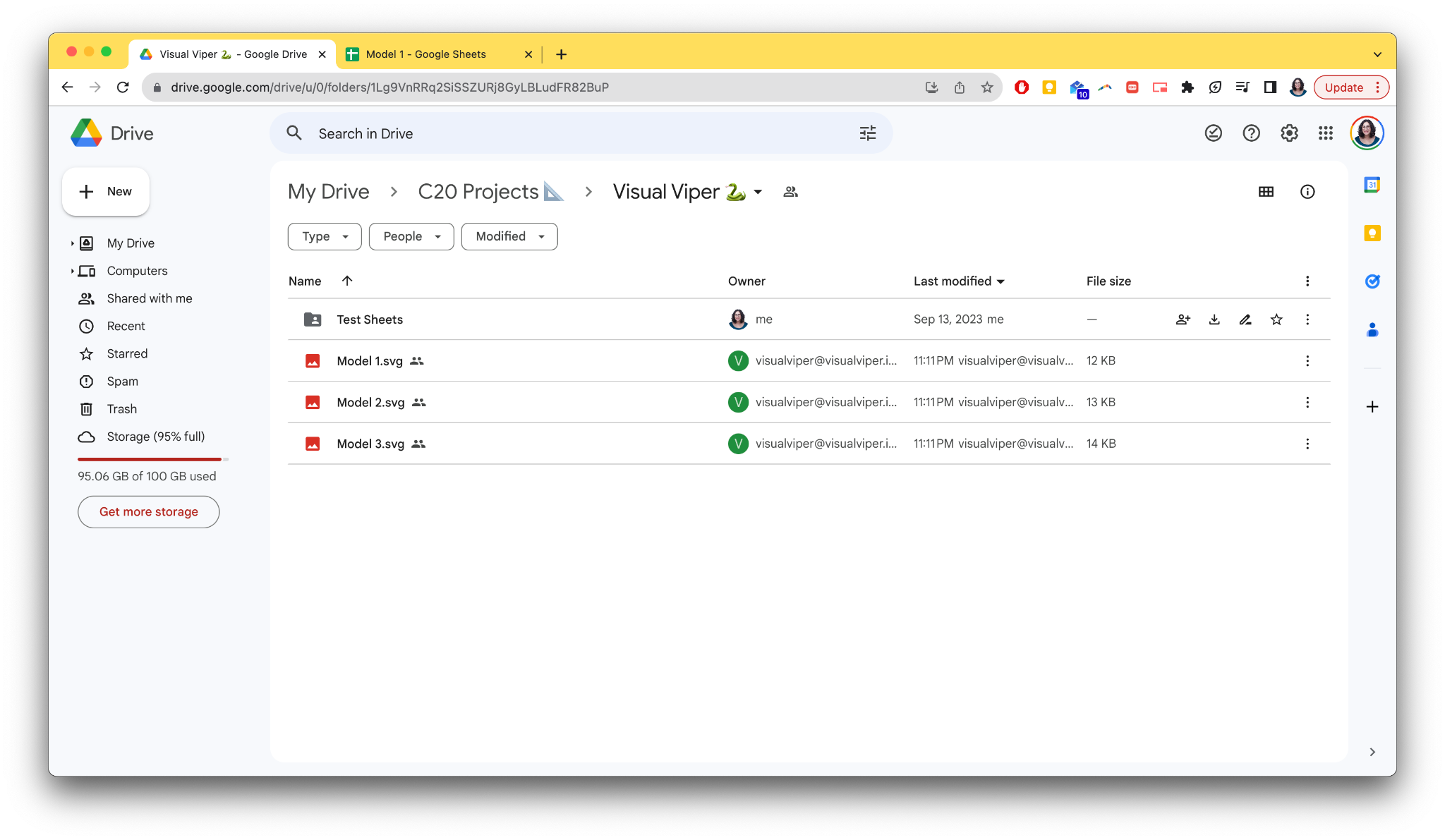


Figure X: Forest Plots for Models 1-3 of the example on Miro Board.

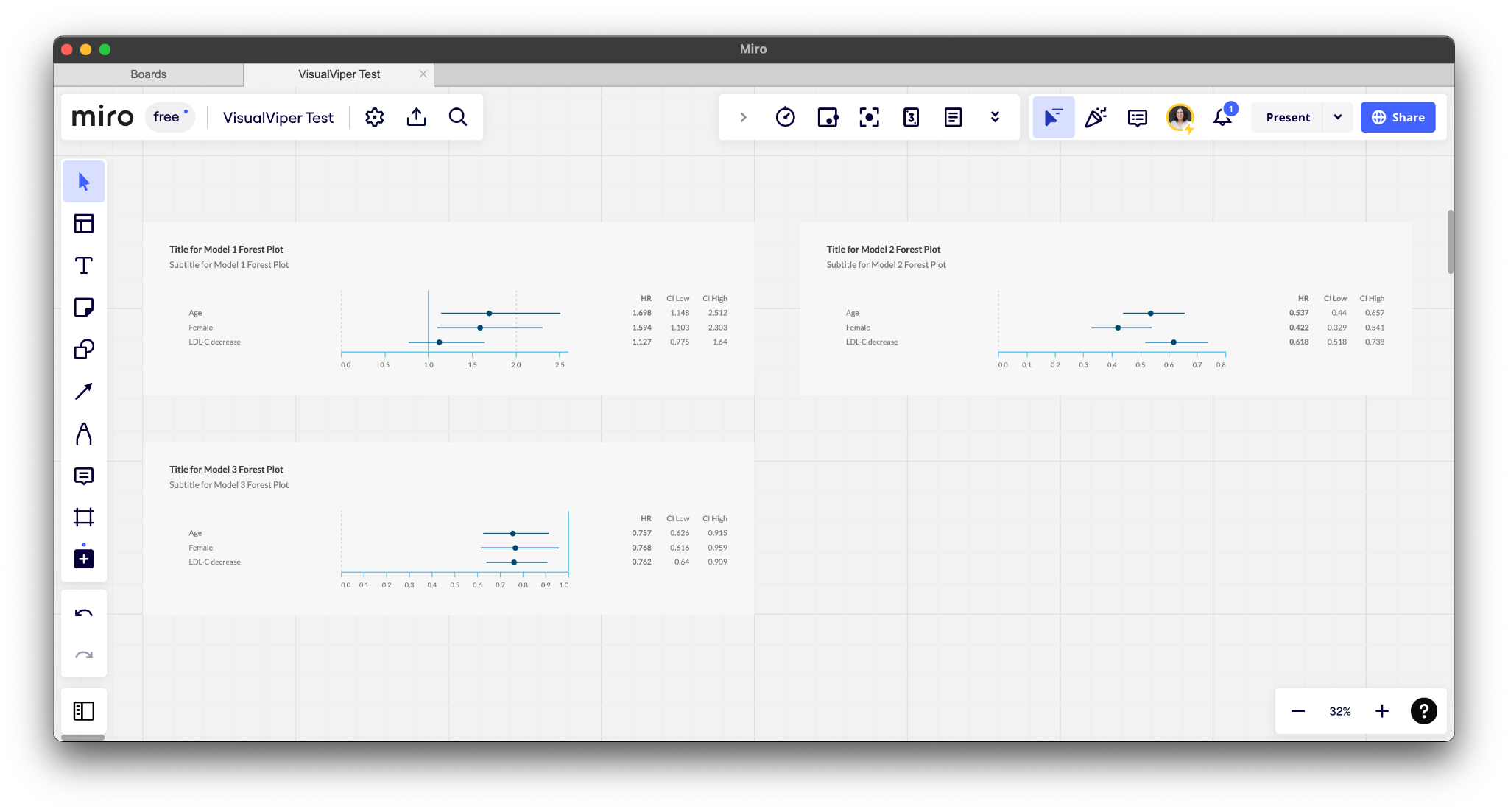


Figure X: Different example of Forest Plots deployed in Miro with tens of plots laid out in a grid.

