Interactive Charts and Data Apps

Plotly in Python, interactive chart patterns, and Dash fundamentals (layout + callbacks).

Marc Reyes
Professional Lecturer · marc.reyes@dlsu.edu.ph
DATA101 — De La Salle University

Today's Plan

01 · INTERACTIVITY

Show not tell

Hover, zoom, selections, and linked views.

02 · PLOTLY (PYTHON)

Interactive charts

Build a figure, export HTML, ship.

03 · OTHER LIBRARIES

Altair, Bokeh, Panel

Choose based on the task and constraints.

04 · DASH

Layout + callbacks

From charts to data apps.

Learning Outcomes

DESIGN

Choose interactions intentionally

Task first, then hover/zoom/selection.

PLOTLY

Ship a single HTML artifact

Interactive, portable, and reproducible.

DASH

Explain layout + callbacks

Inputs → function → outputs.

PRACTICE

Avoid interactive chart traps

Fixed scales, clear reset, minimal clutter.

PART 1 · INTERACTIVE CHARTS

Interactivity supports a task

Not every chart should be interactive. But when it helps, it can remove work from the viewer.

Use Interactivity When

INSPECT

Hover for exact values

Tooltips replace cluttered labels.

FOCUS

Zoom and pan dense series

Same scale, smaller window.

COMPARE

Filter via legend or selection

Reduce groups without losing context.

CONNECT

Linked views (overview → detail)

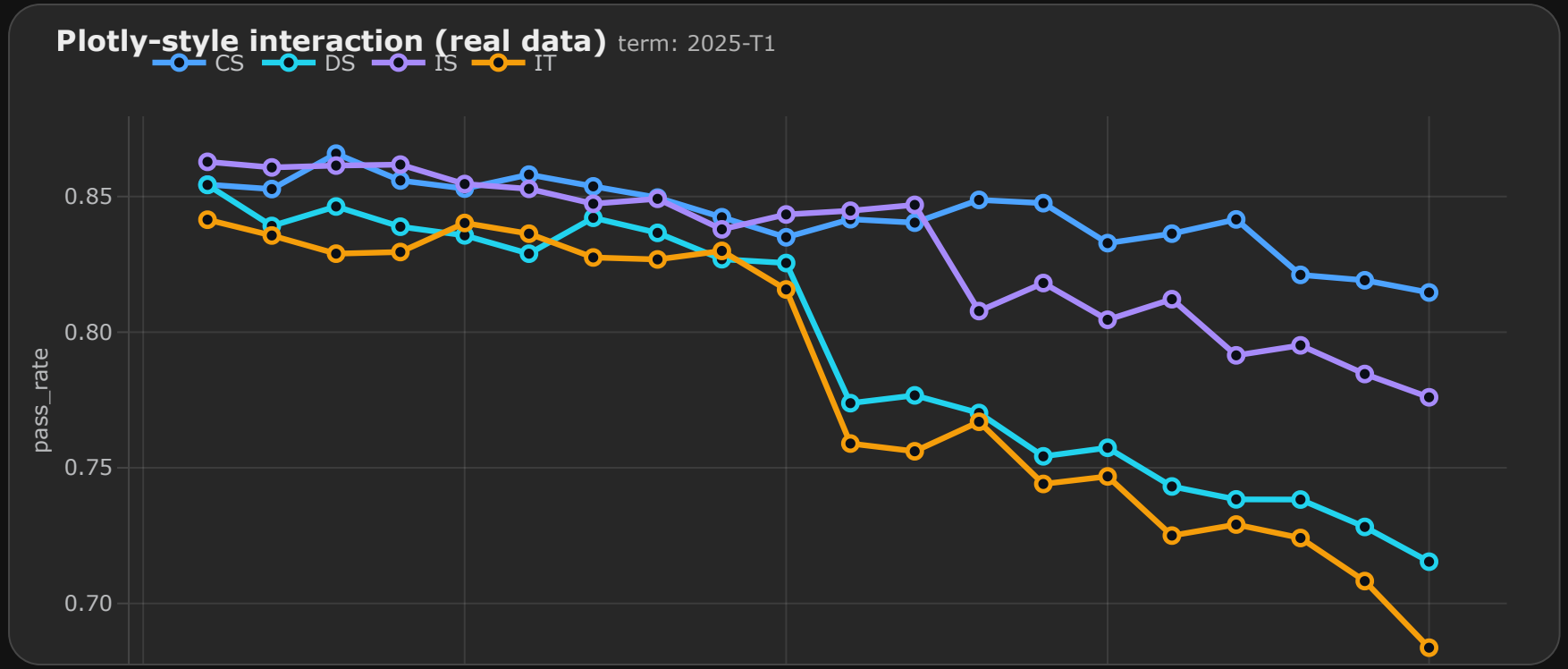
One interaction updates another chart.

Rule

Keep a fixed scale when comparing frames, and always provide a clear reset.

Demo: Hover + Zoom + Legend Filter

Try: hover a point, drag to zoom, scroll to zoom, click legend items to isolate a program.



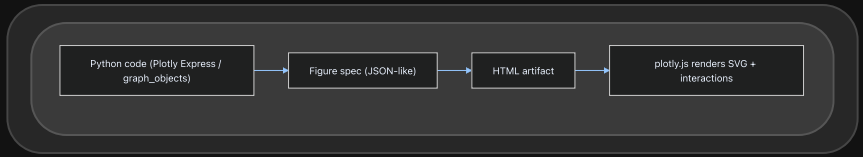
Tip: double-click to reset zoom.

Plotly’s Mental Model

KEY IDEA

A figure is a spec

In Python you build a figure object. In the browser, Plotly renders it as HTML + JavaScript.



PRACTICAL CONSEQUENCE

If you can produce a clean spec, you can export: notebook, report, or a standalone HTML file.

Plotly in Python (Minimal Recipe)

```python
import plotly.express as px

# df: tidy table (one row = one observation)
fig = px.line(
    df,
    x="week",
    y="pass_rate",
    color="program",
    markers=True,
    title="Pass rate by week",
)

fig.update_layout(
    template="plotly_dark",
    legend_title_text="Program",
    hovermode="x unified",
)

fig.write_html("pass_rate_by_week.html", include_plotly
```

WHAT TO NOTICE

- **Tidy data** makes chart creation predictable.
- **Hovermode** controls how tooltips behave.
- **Export** produces a shareable artifact.

Pro habit

Treat the HTML output like a deliverable: title, units, legend behavior, and default view all matter.

Exporting Interactive Work

HTML

Interactive

Hover, zoom, legend filtering.

Best for: web pages, LMS, dashboards.

PNG / SVG

Static

Reliable for PDF + slides.

Best for: reports, print, thumbnails.

JSON SPEC

Reusable

Store a figure, regenerate outputs.

Best for: pipelines and QA.

Rule

If the final medium is static, design the default view so it reads without interaction.

Interactivity Patterns That Scale

HOVER

Inspect without clutter

Use tooltips, not tiny labels everywhere.

LEGEND CLICK

Filter and isolate groups

Make comparisons easier on demand.

BRUSH

Select a range

Time windows, regions, numeric thresholds.

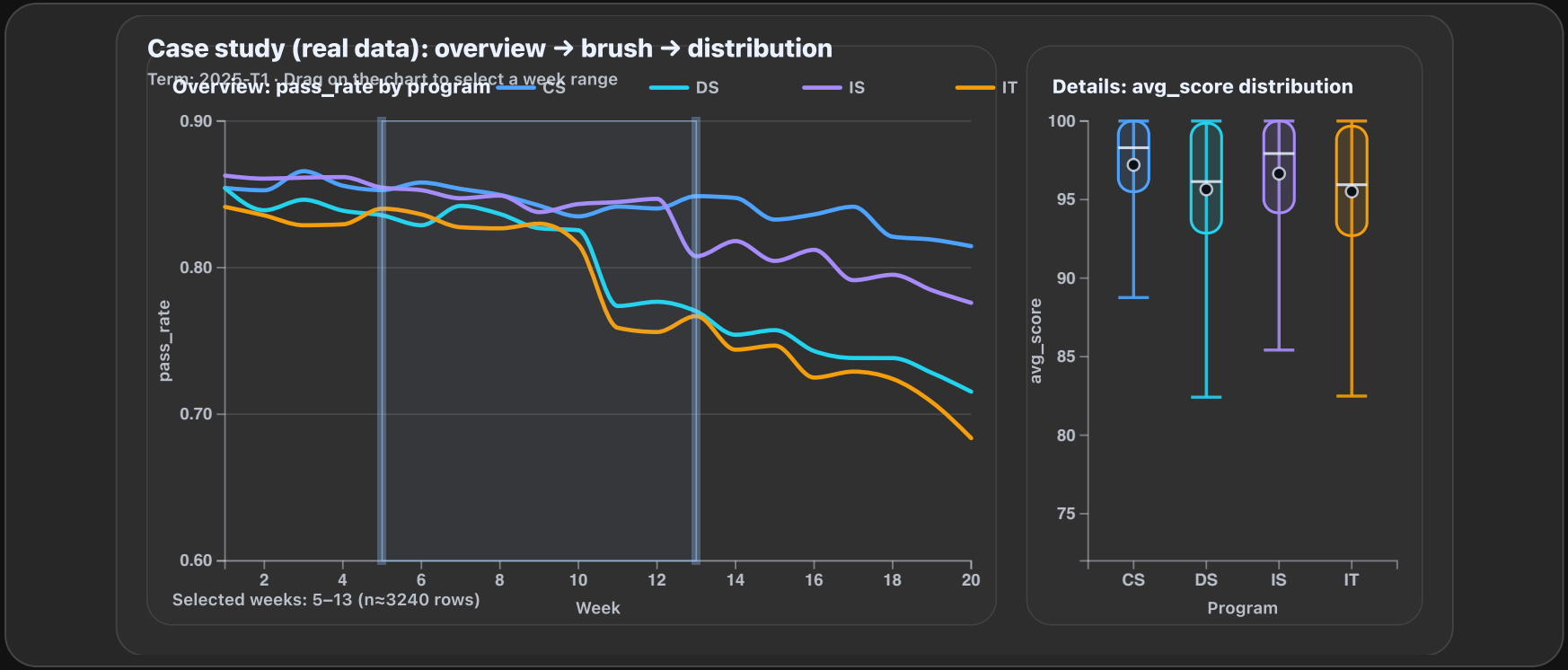
LINKED VIEWS

One input updates another chart

The core pattern behind dashboards.

Demo: Overview → Brush → Distribution (Linked Views)

Drag on the left chart to select a week range. Watch the distribution update.



Other Python Libraries (Quick Heuristic)

DECLARATIVE

Altair (Vega-Lite)

Great for rapid, tidy-data exploration and interactive selections with minimal code.

altair-viz.github.io

CUSTOM TOOLS

Bokeh

When you need non-standard interactions and fine control over tools.

docs.bokeh.org

DASHBOARDS

Panel / HoloViz

Compose widgets + plots quickly, especially for exploration.

panel.holoviz.org

FAST APPS

Streamlit

Simple data apps fast, with tradeoffs in layout and callback control.

streamlit.io

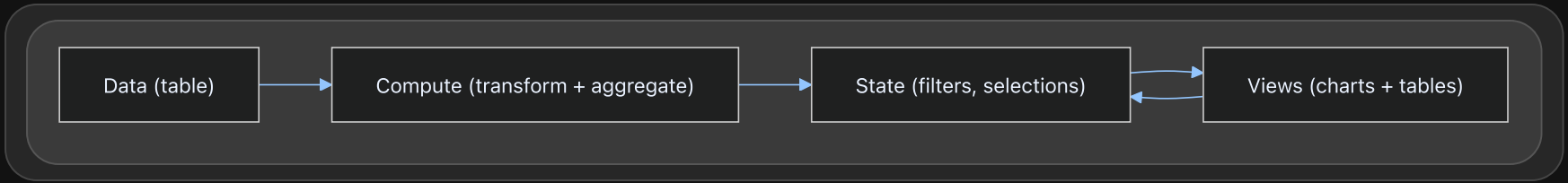
Decision rule

PART 2 · DATA APPS

Dash fundamentals

Layout describes structure. Callbacks define behavior.

From Chart to App



DASH FRAMING

Your app is a set of state transitions

A callback takes inputs and returns updated outputs.

PRO HABIT

Separate compute from render

Cache expensive transforms; keep the UI responsive.

Dash Layout (Structure)

```python
from dash import Dash, html, dcc

app = Dash(__name__)

app.layout = html.Div(
    [
        html.H1("Pass rate dashboard"),
        dcc.Dropdown(["CS", "DS", "IS", "IT"], "CS", id=
        dcc.RangeSlider(1, 20, value=[5, 13], id="week_
        dcc.Graph(id="trend"),
    ],
    className="page",
)
```

MENTAL MODEL

Layout is a tree

HTML containers + interactive controls + output components (like `dcc.Graph`).

Rule

Start with a minimal layout, then add components one at a time. Debug structure before behavior.

Dash Callbacks (Behavior)

```python
from dash import Input, Output, callback
import plotly.express as px

@callback(
    Output("trend", "figure"),
    Input("program", "value"),
    Input("week_range", "value"),
)
def update_trend(program, week_range):
    lo, hi = week_range
    view = df.query("program == @program and @lo <= weel
    fig = px.line(view, x="week", y="pass_rate", markers
    fig.update_layout(template="plotly_dark", hovermode=
    return fig
```

WHAT TO NOTICE

- **Inputs** are the controls.
- **Output** is a component property.
- The callback is just a **pure function** of state.

Debugging trick

Print the filtered table shape first. If the data is wrong, the figure will be wrong.

Live Mini Dashboard (Callback Behavior)

This slide is a simulated mini app. In Dash, the same state changes happen through callbacks.

DASH MINDSET

Inputs → callback → outputs

Change an input, recompute, and redraw. The mechanics are the same in a Vue demo and a Dash app.

TERM

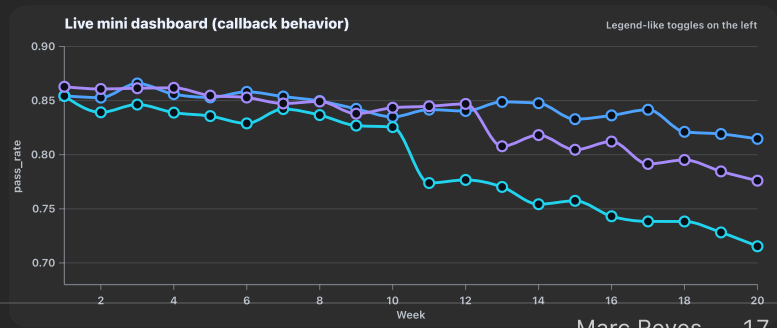
2025-T1

WEEK RANGE

Start: **1**

End: **20**

PROGRAMS



Callback Patterns You'll Use Often

SINGLE OUTPUT

One input → one figure

Start here. Keep it debuggable.

MULTI OUTPUT

One input → chart + table

Same filtered data drives multiple views.

LINKED VIEWS

Brush → update details

Core dashboard interaction pattern.

CLIENTSIDE

Fast UI updates

When server roundtrips feel slow.

Reliability and Performance

DATA

- Validate types and units before plotting.
- Decide your grain (row = what?).
- Make missingness explicit.

APP

- Cache expensive transforms.
- Keep callbacks small and predictable.
- Guard against empty filters.

Rule

Separate: compute once, render many. A dashboard is mostly data plumbing.

What I’d Ship (Professional Checklist)

TASK

What question is this answering?

Interactivity must reduce viewer work.

INTERACTION

Reset, defaults, fixed scales

No “mystery states”.

DATA

Units, grain, validation

Most bugs are data bugs.

SHIP

HTML artifact or deployed app

Pick the delivery format early.

References (Recommended)

Plotly Python docs

Interactive charts + HTML export + animations

plotly.com/python

Dash docs

Layout, callbacks, deployment patterns

dash.plotly.com

Altair docs

Declarative grammar + interactive selections

altair-viz.github.io

Bokeh docs

Custom tools + interactive plotting

docs.bokeh.org