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 · CHOOSING TOOLS

Other libraries

Altair, Bokeh, Panel, Streamlit, Dash.

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, reproducible.

DASH

Explain layout + callbacks

Inputs → function → outputs.

PROFESSIONALISM

Make defaults readable

If it only works on hover, it is fragile.

Why Interactivity Exists

WHEN STATIC BREAKS

Dense charts

Too many points, too many series, too many categories.

WHAT INTERACTIVITY DOES

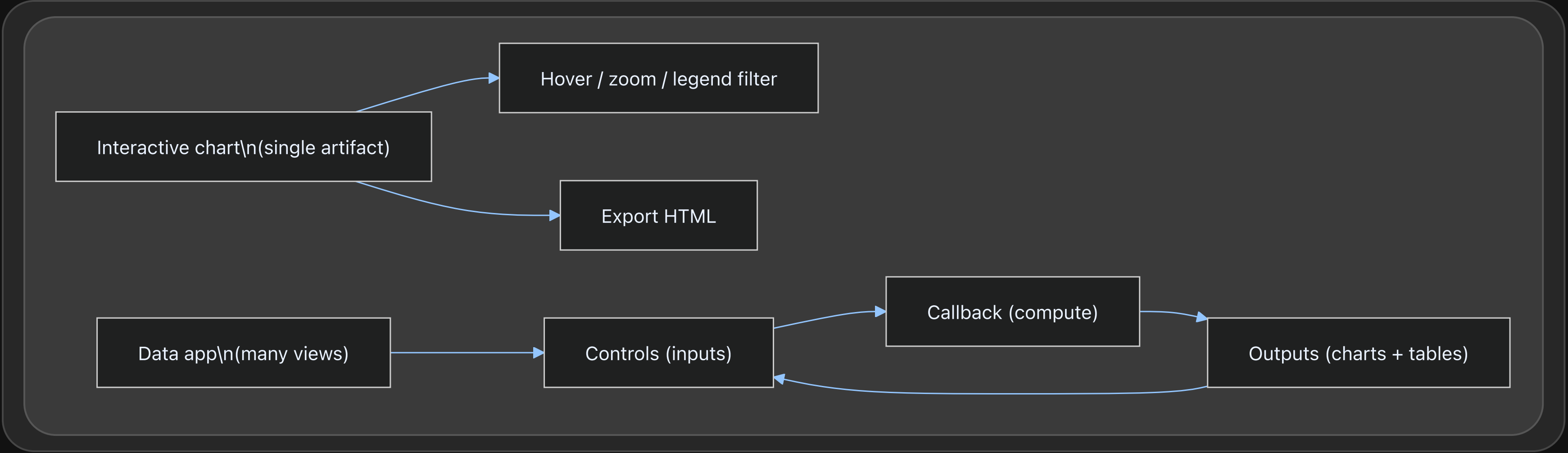
Reduces viewer work

Filter, zoom, and inspect without crowding the canvas.

Rule

Interactivity must support a task, not decorate a chart.

Interactive Chart vs Data App


Hover / zoom / legend filter
Interactive chart\n(single artifact)
Export HTML
Callback (compute)
Data app\n(many views)
Controls (inputs)
Outputs (charts + tables)

INTERACTIVE CHART

A single figure with built-in interactions.

DATA APP

A set of callbacks that update multiple views.

The 3 Non-Negotiables

01

Good defaults

Readable without hovering.

02

Fixed scales

Comparable frames and states.

03

Clear reset

No mystery states.

Professional habit

Assume your viewer will screenshot your chart while it is in a weird state.

PART 1 · INTERACTIVE CHARTS

Interactivity supports a task

Hover, zoom, selection, and linked views.

What Interactivity Is For

INSPECT

Hover for exact values

Tooltips replace cluttered labels.

FOCUS

Zoom and pan dense series

Same scale, smaller window.

COMPARE

Filter groups on demand

Legend click, dropdown, brush.

CONNECT

Linked views

One interaction updates another chart.

Pattern: Tooltips

GOAL

Precision without clutter

Do not label everything. Label on demand.

TOOLTIP SHOULD INCLUDE

- **Entity** (what point is this?)
- **Value** with formatting and units
- **Context** (time, group, filter)

Trap

If the story only exists on hover, the default view is failing.

Pattern: Zoom and Pan

USE WHEN

Time series is dense

The viewer needs a lens, not a different chart.

DESIGN RULE

Keep comparisons stable

Fixed axes, clear reset, visible current range.

Pattern: Legend Filtering

USE WHEN

Many groups

Let the viewer isolate and compare groups quickly.

DESIGN RULE

Make clicks predictable

Click toggles; double-click isolates; always offer reset.

Pattern: Selection / Brush

USE WHEN

Pick a range

Weeks 5–13, scores 80–90, or a region on a map.

DESIGN RULE

Show the selection state

Selected window, count of rows, and a clear way to clear.

Pattern: Linked Views

USE WHEN

Overview → details

Trends first, then distributions, then record lookup.

DESIGN RULE

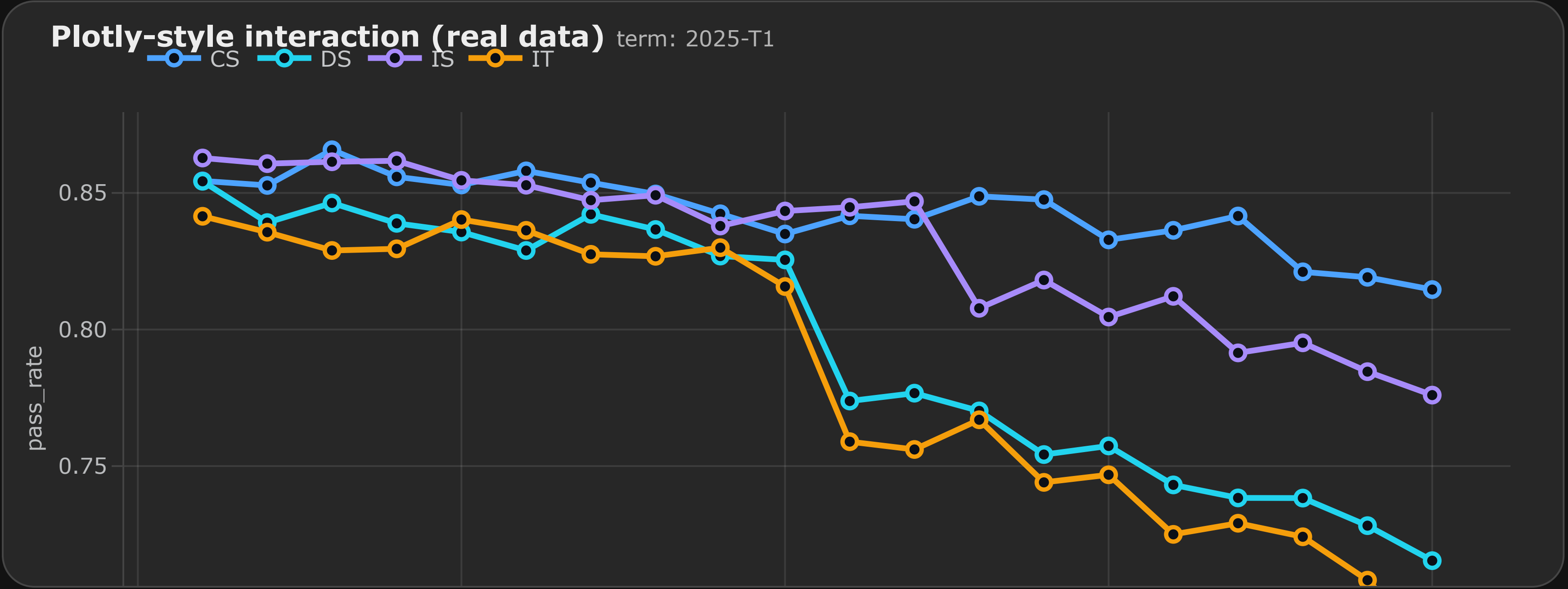
One source of truth

A single filtered dataset should drive all outputs.

Demo: Hover + Zoom + Legend Filter

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

Open interactive lab (new tab)



Plotly-style interaction (real data) term: 2025-T1

Tip: double-click to reset zoom.

Deconstruct the Demo (Why It Works)

DEFAULT VIEW

Readable line

Grid is subtle; labels are complete.

INTERACTION

Adds detail

Hover gives exact values, not more clutter.

STATE

Resettable

Zoom is reversible and predictable.

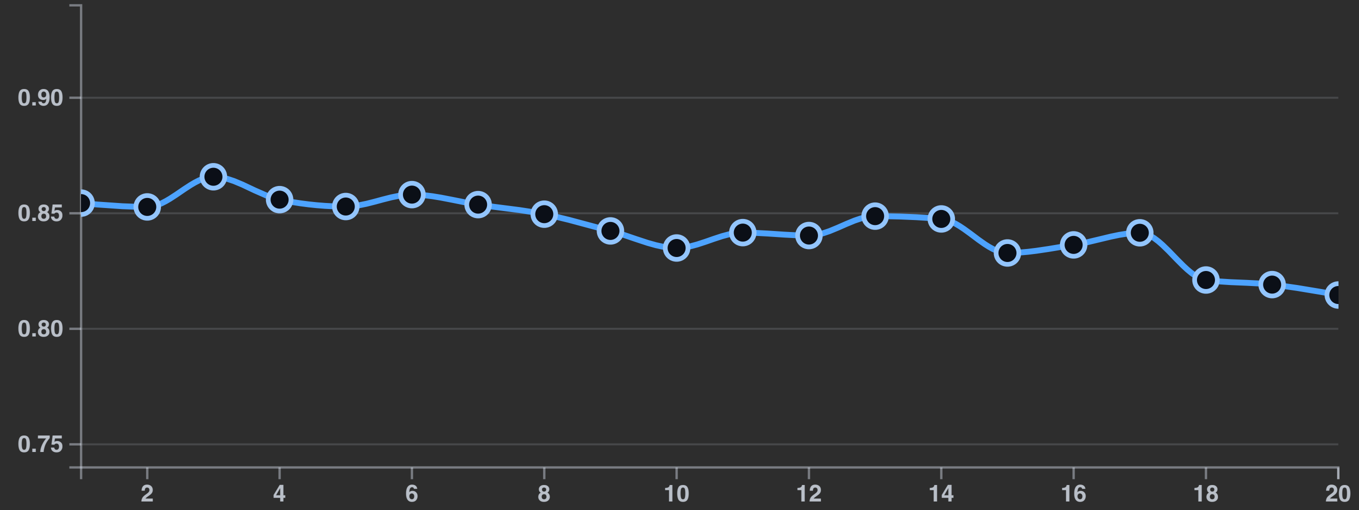
Demo: Tooltip + Zoom (Same Data, Different Engine)

This is a D3-style interaction pattern: tooltip + zoom/pan + focus line.

Open interactive lab (new tab)

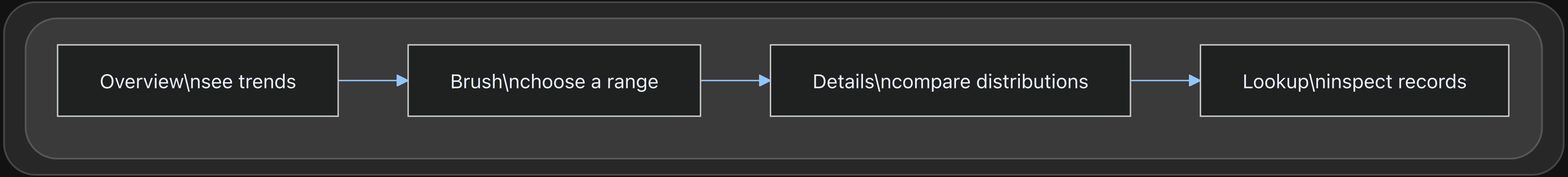
Interactive line (real data)

Scroll to zoom · drag to pan



Data: 2025-T1 · CS (aggregated)

Linked View Case Study (Reading Order)



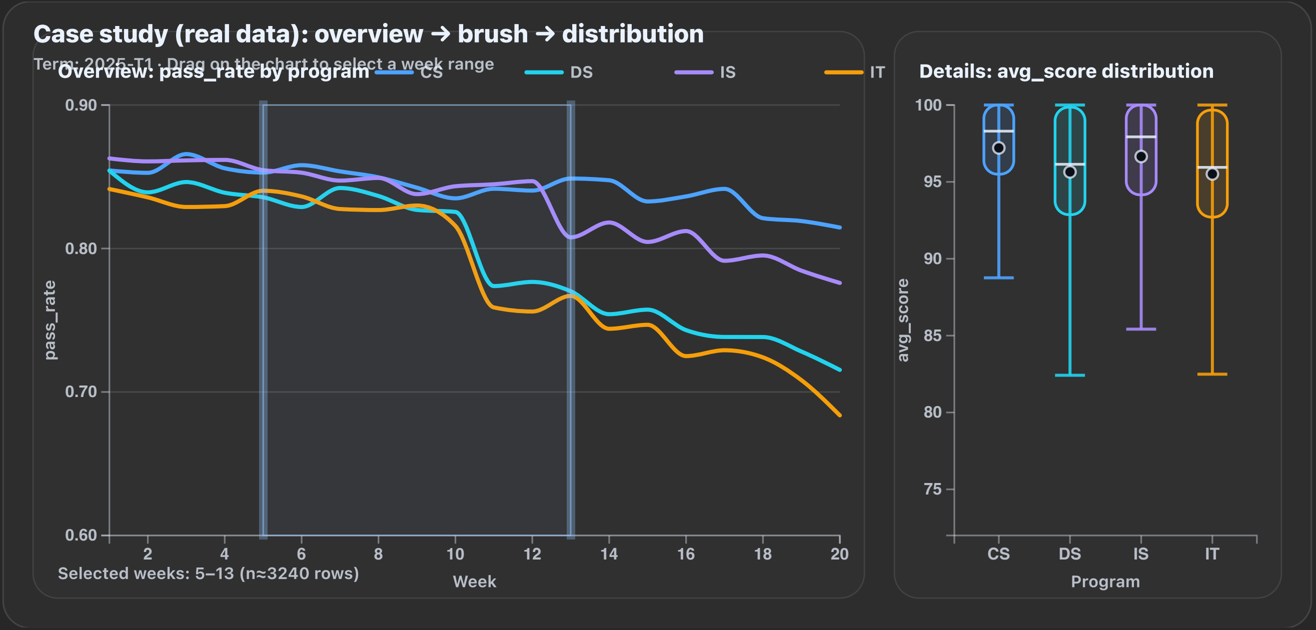
Rule

Start broad, then narrow. Do not start with tables of raw rows.

Demo: Overview → Brush → Distribution (Linked Views)

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

Open interactive lab (new tab)



Case study (real data): overview → brush → distribution

Term: 2025-T1 · Drag on the chart to select a week range

Selected weeks: 5–13 (n≈3240 rows)

Animation (When It Helps)

USE ANIMATION WHEN

- The task is to see **change over time**.
- You keep a **fixed scale** (comparable frames).
- You still provide a **static alternative** for precision.

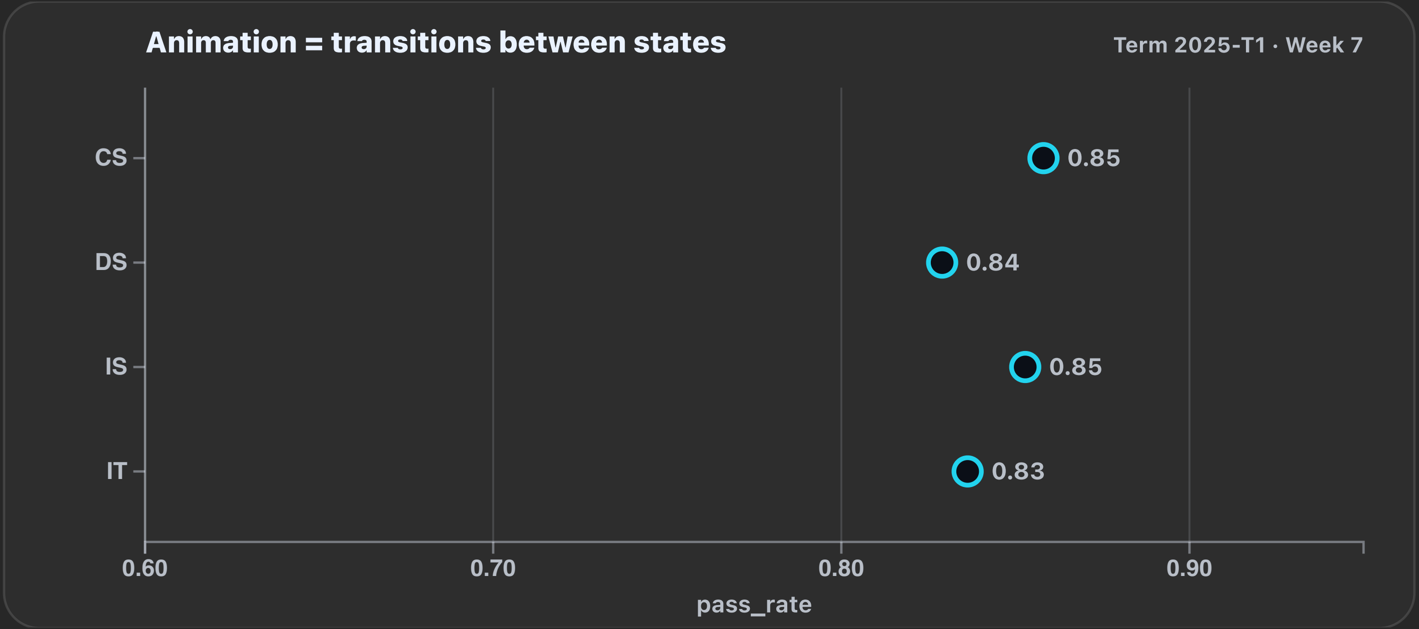
DO NOT USE WHEN

- It is only for “wow”.
- It hides the baseline.
- It changes the scale per frame.

Demo: Animation = Transitions Between States

Same data, same scale. The marks move. That is all animation should be.

Open interactive lab (new tab)



Plotly Animations (Python Concept)

```python
import plotly.express as px

# A frame per time step (week)
fig = px.scatter(
    df,
    x="pass_rate",
    y="program",
    animation_frame="week",
    animation_group="program",
    range_x=[0.60, 0.92],
)

fig.update_layout(template="plotly_dark")
fig.write_html("animated.html", include_plotlyjs="cdn")
```

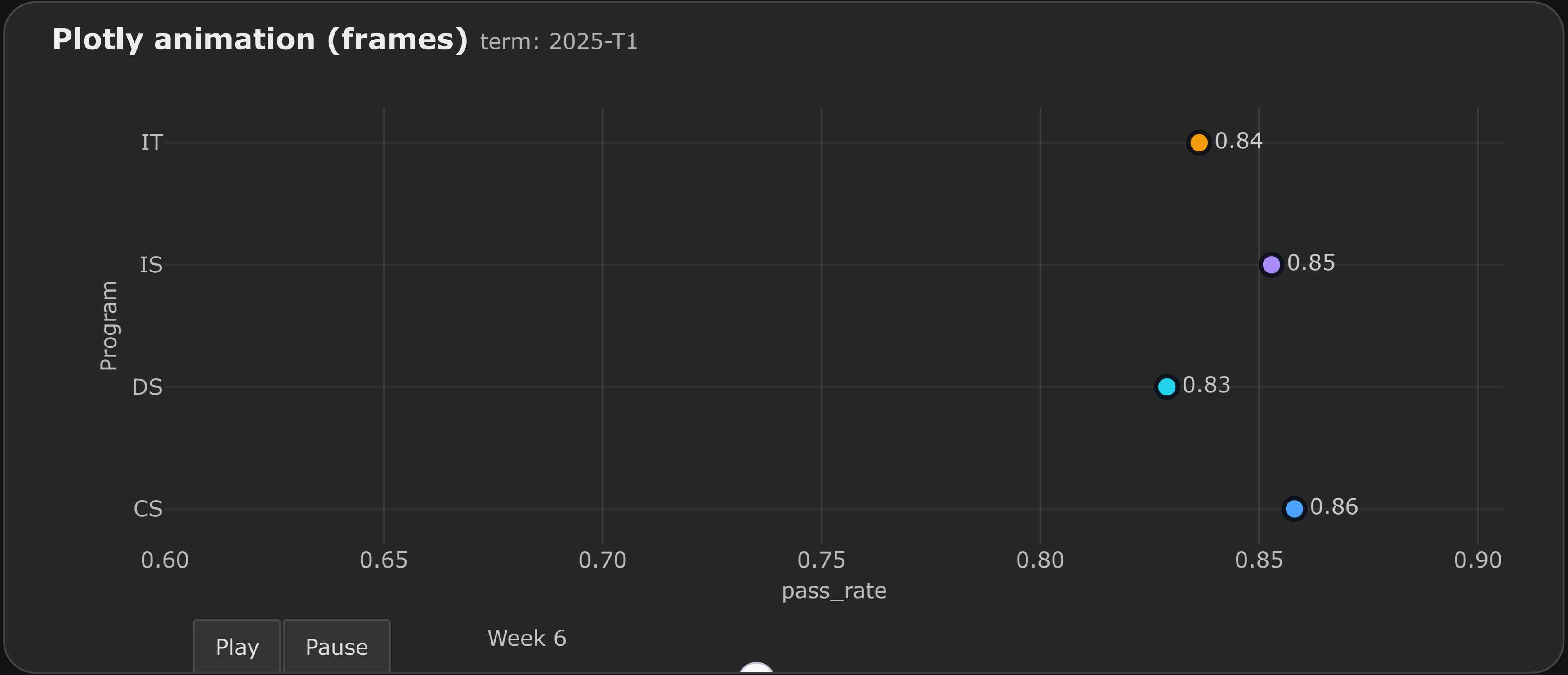
WHAT TO NOTICE

- Animation is just **frames**.
- Keep a **fixed axis range**.
- Export as a **single HTML** when sharing.

plotly.com/python/animations

Demo: Plotly Frames (In-Slide)

Use the Play button or the slider. Notice the fixed x-axis range.



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.

Static Fallback and Accessibility

FALLBACK

Default view must stand alone

Assume the viewer prints it or screenshots it.

ACCESSIBILITY

Do not encode with color only

Use labels, line styles, or direct annotations when needed.

Rule

Interactivity should reveal detail, not hide structure.

Pitfalls (Interactive Charts)

NOISE

Too many tools

Mode bars, buttons, and sliders everywhere.

COMPARABILITY

Changing scales

Every frame looks “dramatic” but is misleading.

STATE

No reset

The viewer cannot recover from a weird interaction.

DATA

Wrong grain

Most chart bugs are still data bugs.

PART 2 · PLOTLY IN PYTHON

Build figures you can ship

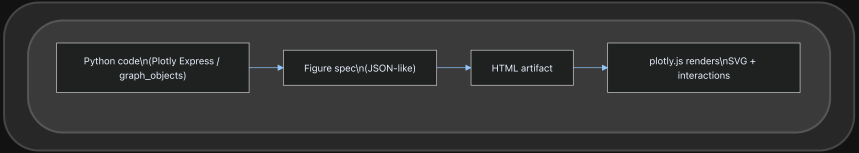
A figure is a spec. A deliverable is an artifact.

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.



Python code\n(Plotly Express / graph_objects)

Figure spec\n(JSON-like)

HTML artifact

plotly.js renders\nSVG + interactions

Plotly Express vs Graph Objects

PLOTLY EXPRESS

Fast, opinionated

Great for tidy data: `px.line`, `px.scatter`, `px.histogram`.

GRAPH_OBJECTS

Explicit control

When you need subplots, custom traces, or non-standard layouts.

Rule

Start with Express. Switch to graph_objects only when you can name the missing control.

Your Data Shape Controls Your Workload

BEST DEFAULT

Tidy / long tables

One row = one observation. Columns = fields.

COMMON FIX

Reshape before plotting

`melt`, `groupby`, and explicit sorting solve most problems.

Plotly in Python (Minimal Recipe)

```python
import plotly.express as px

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 CHECK

- **Labels** are complete (units when needed).
- **Legend** supports comparison (click behavior).
- **Default view** reads without interaction.

Styling That Signals Professional Work

MARGINS

Breathing room

Titles and axes should not collide.

TYPOGRAPHY

Intentional sizes

Large enough at 100% zoom.

GRIDS

Subtle

Data is loud. Scaffolding is quiet.

Hovertemplate (Precision Without Clutter)

```
fig.update_traces(
    hovertemplate=
        "<b>%{legendgroup}</b>" +
        "<br>week=%{x}" +
        "<br>pass_rate=%{y:.3f}" +
        "<extra></extra>"
)
```

WHY THIS MATTERS

Default tooltips are often noisy. A good tooltip reads like a label, not a log file.

Rule

Format numbers the way you would in a report.

Facets (Small Multiples)

USE WHEN

Compare distributions

Same axes, same bins, easy comparison.

DESIGN RULE

Align scales

Small multiples are only useful when the scale is shared.

Subplots and Shared Axes

PROBLEM

Too much in one panel

Multiple metrics, multiple views, or multiple tasks.

SOLUTION

Split views, share scales

Keep comparisons valid and reduce clutter.

Category Ordering Is Part of the Story

DEFAULT TRAP

Alphabetical ordering

It rarely matches the question.

PROFESSIONAL MOVE

Sort by meaning

Sort by value, change, or a domain order.

Color: Encode Meaning, Not Preference

CATEGORICAL

Different groups

Avoid rainbow. Keep contrast.

SEQUENTIAL

Low → high

Lightness carries order.

DIVERGING

Below / above

A meaningful midpoint.

Performance (When Data Gets Big)

SYMPTOMS

Lag, stutter, freezes

The browser is doing too much work per frame.

MITIGATIONS

- Aggregate or sample.
- Use WebGL traces (`scattergl`).
- Limit hover points.

Exporting From Python

HTML

`write_html`

Best for interactive delivery.

PNG / SVG

Kaleido

Static exports for reports.

SPEC

JSON

Store the figure for reproducibility.

Pitfalls and Debugging (Plotly)

DATA BUGS

Wrong unit or grain

Wrong denominators cause wrong stories.

CHART BUGS

Defaults not reviewed

Axis ranges, sorting, and hover formatting.

Debugging trick

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

When Not to Use Plotly

STATIC DELIVERABLES

You only need a PDF

Matplotlib can be simpler and lighter.

COMPLEX CUSTOM VISUALS

Highly bespoke interactions

Consider D3 or a custom front end.

plotly.com/python

PART 3 · CHOOSING TOOLS

Pick by delivery

HTML artifact vs deployed app.

Quick Heuristic

ALTAIR

Declarative exploration

Tidy data + compact specs + selections.

BOKEH

Custom tools

When interactions are the product.

PANEL

Widget composition

Quick dashboards for exploration.

STREAMLIT

Fast apps

Great defaults, less callback control.

Other Python Libraries (Quick Links)

ALTAIR

Vega-Lite grammar

altair-viz.github.io

BOKEH

Interactive plotting

docs.bokeh.org

PANEL

HoloViz ecosystem

panel.holoviz.org

STREAMLIT

Data apps fast

streamlit.io

Decision rule

Pick the tool that matches your delivery: a single HTML file, or a real app with server-side callbacks.

The Practical Split

IF YOU NEED

A single file to share

Plotly HTML, Altair HTML, Bokeh HTML.

IF YOU NEED

A real app

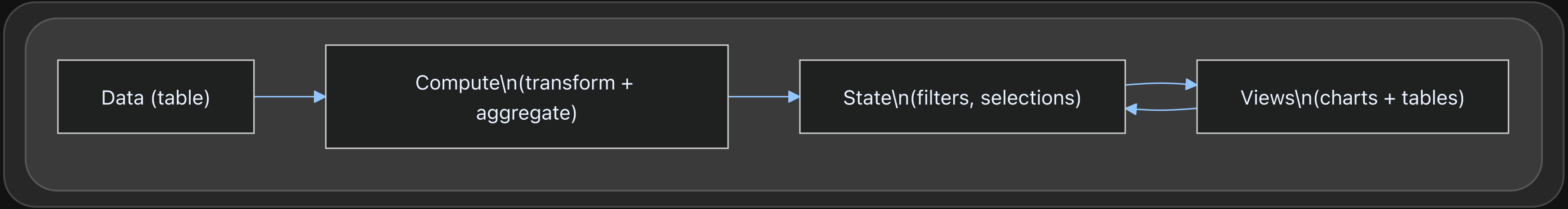
Dash, Panel, Streamlit (server-side state).

PART 4 · DATA APPS

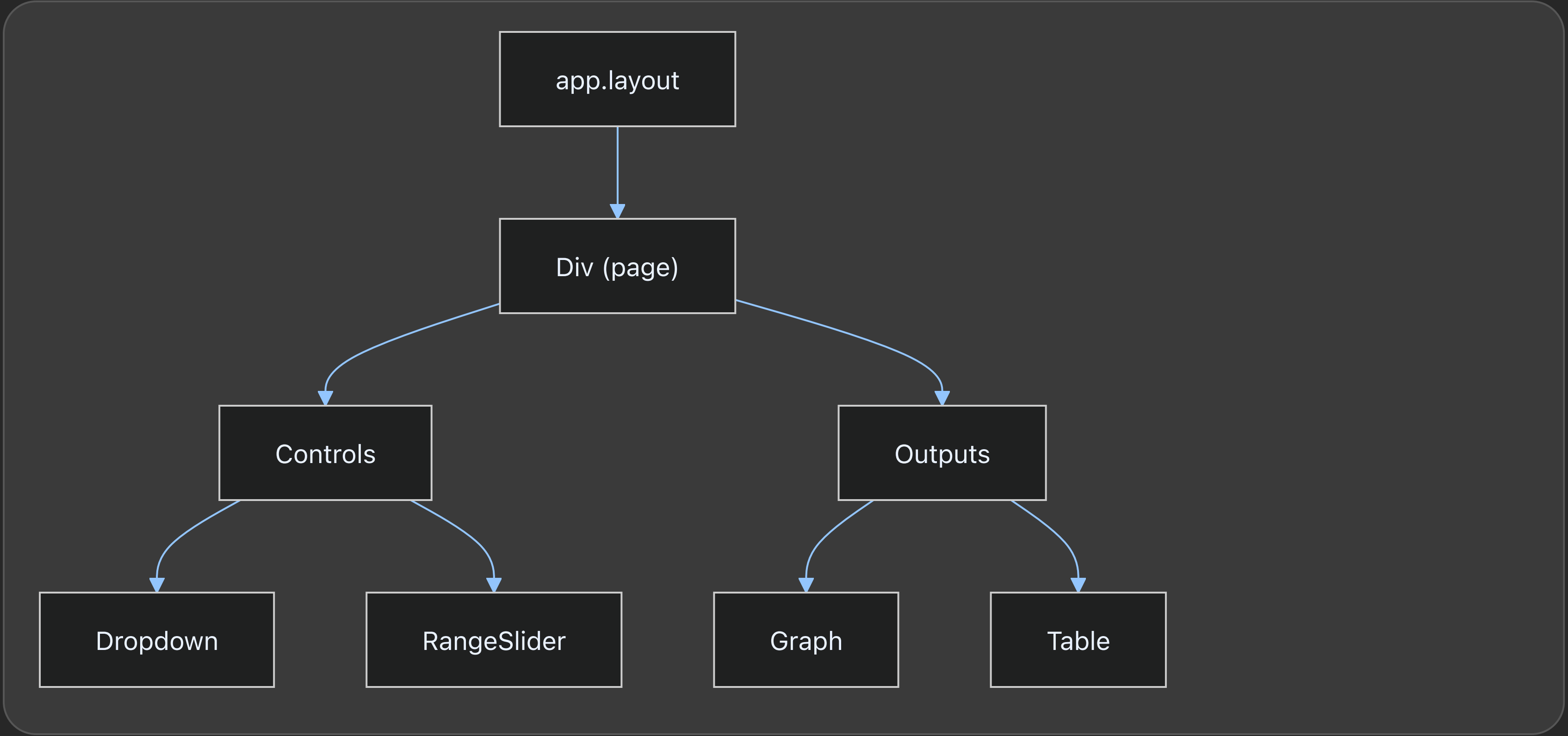
Dash fundamentals

Layout describes structure. Callbacks define behavior.

From Chart to App


Data (table)
Compute\n(transform + aggregate)
State\n(filters, selections)
Views\n(charts + tables)

Dash Layout Is a Tree



Dash Layout (Structure)

```
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 declarative

You describe the UI once. Callbacks provide the behavior.

Pro habit

Add components one at a time. Test IDs and wiring early.

Callback Mental Model

INPUTS

Controls

Dropdowns, sliders, buttons.

CALLBACK

Compute

Filter, aggregate, reshape.

OUTPUTS

Views

Charts, tables, KPI cards.

Rule

Treat callbacks like pure functions of state. That is how you debug them.

Dash Callbacks (Behavior)

```
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 <= week
    fig = px.line(view, x="week", y="pass_rate", markers
    fig.update_layout(template="plotly_dark", hovermode=
    return fig
```

DEBUGGING ORDER

- Confirm input values.
- Print the filtered row count.
- Validate units and grain.
- Then style the figure.

Debugging Checklist (Dash)

DATA

- Do you aggregate to the correct grain?
- Are types and units correct?
- Are missing values handled explicitly?

APP

- Are component IDs unique?
- Do callbacks guard empty filters?
- Is there a clear reset state?

Demo: Live Mini Dashboard (Callback Behavior)

This slide is a simulated mini app. In Dash, callbacks do the same state updates.

Open interactive lab

DASH CALLBACK MENTAL MODEL

Inputs → callback → outputs

Change an input, recompute, and redraw. In Dash, the callback does the recompute.

TERM

2025-T1

WEEK RANGE

Start: **1** End: **20**

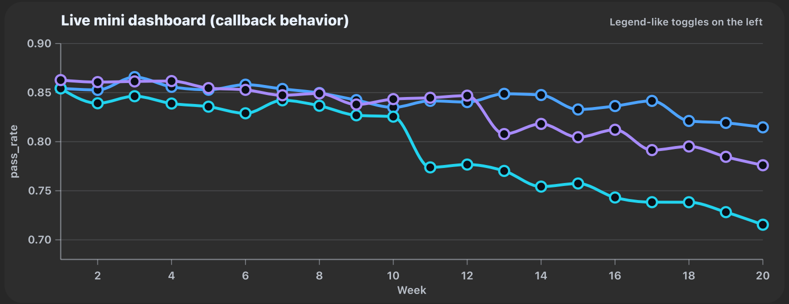
PROGRAMS

● **CS** ● **DS** ● **IS** ● **IT**

CALLBACK OUTPUTS (LIVE)

Rows: **60** · Overall pass_rate: **0.822**

204708 / 248956 students (aggregated)



Multi-Output Pattern (Common in Dash)

ONE FILTER

Many outputs

Graph updates, KPI cards update, table updates.

RULE

Compute once

Derive a filtered table once, then reuse it for all views.

Input vs State vs Store

INPUT

Triggers callback

User changes it.

STATE

Read-only context

Used inside callback, does not trigger.

STORE

Cached data

Avoid recomputing on every interaction.

Scaling Beyond One Callback

PATTERN MATCHING

Dynamic components

Generate many similar charts and wire callbacks cleanly.

CLIENTSIDE CALLBACKS

Fast UI updates

Move lightweight logic to the browser when needed.

Performance and Reliability

PERFORMANCE

- Cache expensive transforms.
- Keep callbacks small and predictable.
- Do not re-render huge figures unnecessarily.

RELIABILITY

- Guard empty selections.
- Use sensible defaults.
- Log inputs when debugging.

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