Slides, posters, and figures

Last modified: 2026-11-13 12:24:54 (PST)

Talks, posters, and figures carry the same research as a paper, but the reader sees them for less time and cannot reread them. This chapter gives practical rules for each.

Guidelines for slide decks

When preparing slides for a lab meeting, conference, qualifying exam, or thesis defense, follow these guidelines:

- **Always include slide numbers**: Slide numbers make feedback specific. During questions or a practice run, listeners need to refer back to a slide quickly (e.g., "On slide 14, how was that variable defined?") instead of describing it or guessing where a topic occurred.
- **Complete, record, and review practice runs**: Before any significant presentation, rehearse the complete talk out loud, record the practice run (with screen recording software, a webcam, or your phone), and watch the recording. The recording shows your pacing, timing, filler words, and transitions, and whether slide text and figures stay legible on screen.
- **One main point per slide**: Limit the cognitive load of each slide. If a slide presents several findings or a complex argument, split it across consecutive slides so the audience can follow each point.
- **Prefer visual evidence**: Audiences cannot read dense paragraphs while listening to a speaker. Use clean charts, diagrams, and annotated figures instead of text-heavy bullet lists.
- **Direct attention**: Use callouts, annotations, or a sequential reveal so the audience looks at the part of the figure you are discussing.
- **Show software working**: When you present a package or a pipeline, show live output, reproducible code, or a screenshot instead of only listing its features.
- **Ensure high contrast and readability**: Use large fonts and high-contrast colors. Check that axis labels, legends, and annotations can be read from the back of a large room or on a low-resolution projector.

Presentation and poster resources

- How to tell a compelling story in scientific presentations (Kirchoff 2021)
- The science of storytelling: the David Attenborough style of scientific presentation (Ratcliff 2023)
- How to create a better research poster in less time - Mike Morrison's video introducing the #betterposter template
- Towards an "#evenbetterposter" - the MIT Biological Engineering Communication Lab's improvements to that template

Show the result in a figure

A figure often shows a pattern, an outlier or a comparison faster and more reliably than a sentence or a table can describe it. So when a result can be drawn, draw it, and use the text to point the reader to what the figure shows.

- Show the data before summarizing them. Summary statistics can hide structure that a plot of the data reveals.
- Put the key result of each section in a figure, and write the section's text around that figure.
- Make the figure show the comparison the text claims, with the groups, fitted curve or interval on the same axes.
- Say in the text what to look at in the figure (see Figures and tables).
- Use a table instead when readers need exact values to look up or compute with.

Example 0.1 (Four datasets with the same summaries)**.** Anscombe (1973) constructed four small datasets of (x, y) pairs (Wikipedia: Anscombe's quartet). All four have the same mean of x (9), the same mean of y (7.5), the same correlation (0.82 to two decimal places) and the same least-squares line, with slope 0.5. A table of these summaries cannot tell the datasets apart. A scatterplot of each can: one is roughly linear with scatter, one follows a smooth curve, one is an exact line with a single outlier, and one has all but one of its points at the same x.

Scientific figures

Include a figure wherever one helps a reader see a definition, an example, or a result. Draw a graph for a relationship between quantities, a diagram for a structure or a process, and a plot of the data for a fitted model or an estimate. Add figures to new work as you write it, and add them to existing chapters that lack them. On a website or slide deck, draw a graph as

an interactive plotly figure where feasible, and as a 3D plotly figure where a third dimension shows the idea, such as a surface over two parameters. In static formats, such as PDF, Word, and PowerPoint, still render every 3D plot and other interactive figure, as a noninteractive version of it (Ezra, 2026-11-10). A note that sends readers to the HTML version does not replace the figure.

When a definition or example graphs a model whose parameters are chosen rather than fitted to data, such as a density with a given mean and covariance, make the graph interactive in HTML output, slide decks included (Ezra, 2026-11-10). Give each parameter a slider, a numeric entry box, or both, and redraw the graph as the reader changes them. Start each control at the value the text uses, so the prose about that value describes the graph as it first appears, and in static formats draw the graph at those starting values. A graph of a model fitted to data stays fixed, because its parameters come from the data. In Quarto, build the graph with Observable JS cells, which run in both the website and the slides:

- Make a slider with a numeric box beside it with `Inputs.range`, and a numeric box alone with `Inputs.number`.
- Draw a 2D graph with Observable Plot, and a 3D surface with plotly.js.
- Put the cells in a `.content-visible when-format="html"` div.
- Draw the static version with an R chunk that runs only outside HTML output.

Give every figure alt text, a numbered caption, and a code chunk that comments what the code does and why.

Open each caption with a sentence that says what the figure or table shows, so a reader who sees only the figure or table learns its point without following a link. A cross-reference to the example or section that the figure or table illustrates may support that sentence, either after the sentence or in parentheses within the sentence, but the cross-reference never replaces the sentence. Two captions for the same figure show the difference:

- Pointer only: "The symmetry of `@exm-x`." The reader learns nothing until following the link.
- Self-contained: "The logistic function at -2 is as high as 1 minus its value at 2 (`@exm-x`)." The reader learns the point at once.

A caption does not rely on another figure's caption (Ezra, 2026-11-03): do not describe a figure as "the surface of `@fig-x`" or "the same paths as `@fig-x`". State what this figure shows, as if it were the only figure on the page. A figure that adds a view of another, such as a surface for a contour plot, gets a caption that names its own quantities and its own point.

Lead with a well-labeled figure or table instead of a long paragraph that describes a pipeline or compares groups. Use a diagram for a conceptual model, such as a directed acyclic graph (DAG), and for a process, such as a participant flow (CONSORT) diagram or a workflow schematic. Let the text interpret what the figure shows instead of listing every number the

reader can already see in it. Give the key estimates in the text and point to the figure for the rest.

A clear figure lets a reader check a result without help from the text. Before you share a figure:

- Label the x-axis and y-axis clearly. Name each quantity in plain words, and add its symbol only as an option: "Probability, π" and "Log-odds, η", not "π" or "η" alone, and not "f(x)" or "x1". A reader should not have to look up a symbol to know what an axis shows. Wrap a long y-axis title onto two lines so that it stays horizontal and readable, rather than rotating it or shortening it to a symbol.
- Use panel labels when a figure has several subplots.
- Include enough tick marks and tick labels.
- Put the ticks in the original measurement scale of each variable, however the model is parametrized and the axis is spaced. If a fit uses scaled or standardized values, place the ticks at round original values and label them with those, so a horsepower axis reads 50, 100, 150 and 200, not 0 to 1. Report numbers such as an error in the response's units too.
- Print tick labels as plain or comma-separated numbers, such as 0.001 or 10,000, not in scientific notation. A logarithmic axis is fine. Only the label format matters, so write 1, 10, 100 and 1,000, not 1e+00 through 1e+03. When the numbers are huge or tiny, rescale the units and say so in the axis title, such as "dollars (millions)".
- Use a pseudo-log axis when the data include zero or negative values that a logarithmic axis cannot show. A pseudo-log axis is linear near zero and logarithmic far from zero, so every real number has a position. The `transform_pseudo_log()` function of the R package scales maps a value x to $\operatorname{asinh}(x/(2\sigma))$ divided by the natural logarithm of its `base`, and the argument σ (`sigma`) sets the width of the linear part around zero. Pass it to a ggplot2 scale, as in `scale_y_continuous(transform = scales::transform_pseudo_log(sigma = 0.1))`. Say in the axis title or the caption that the axis is pseudo-log.
- Use font sizes that are easy to read without dominating the plot.
- Use colorblind-friendly palettes (see ColorBrewer or iwanthue). Pair color with shape or line type, so that color is never the only way to tell groups apart. This matters most with more than 3 or 4 groups, because even a colorblind-friendly palette then has pairs of colors that are close together.
- Define colors, shapes, and line types in a legend.
- Put the plot's legend (the key to colors, shapes and line types) at the bottom of the plot, unless you have a reason to put it elsewhere. A bottom legend leaves the full plot width for the data and gives every figure the same layout.
- In R, draw the static version of a figure with ggplot2, and the interactive version with plotly as described in Section . Present R results with modern packages gives the reasons.
- Draw a tree (a parse tree, decision tree, dendrogram, or any other hierarchy) as a diagram figure, not only as nested bullets. A nested list hides how the branches relate and makes

a reader count indentation to find a node's parent. Keep the list too when it adds information the diagram lacks. Draw the diagram from the tree's data and fill leaves and internal nodes differently, with the legend at the bottom. Wrap the diagram in a `#fig-` div with a caption, and set alt text that reads out the structure with `fig-alt`. The Morrison-Lab/gha check `check-tree-lists` flags a pull request that adds a definition, example, or other labeled div that mentions a tree, holds a list nested three or more levels deep, and has no figure; a `<!-- no-tree-figure: reason -->` comment inside the div exempts it.
- Minimize abbreviations in labels and legends.
- Use transparency to show overlapping data.
- Show measures of precision, such as 95% confidence intervals.

For more:

- Ten Simple Rules for Better Figures (Rougier et al. 2014)
- Fundamentals of Data Visualization by Claus Wilke - a free online book on choosing the kind of figure, the colors, and the layout for a set of data
- Tufte CSS - style HTML articles like Edward Tufte's books and handouts, with sidenotes, typography, and tight integration of graphics with text

Combine panels with Quarto layout or in code

A figure or table with several panels can be put together in two places:

- **Quarto layout**: each panel is a subfigure or subtable. A subfigure is a `#fig-` div nested in the parent figure's div, and a subtable is a `#tbl-` div nested in the parent table's div. The parent div arranges the panels with `layout-ncol`, `layout-nrow`, or `layout`.
- **Code**: one plot or table object holds every panel, made with patchwork, ggplot2 facets, plotly's `subplot()`, matplotlib's `subplots()`, or one table with grouped columns.

Use Quarto layout by default. Quarto gives each panel its own label, subcaption, and alt text, and those keep working in every output format:

- A cross-reference names the panel, as in "Figure 3b", so the text can point a reader at one panel without "left" or "right".
- Each panel's subcaption says what that panel shows, so the panels need no hand-made titles that repeat it.
- Each panel's `fig-alt` describes that panel alone, so a screen reader user hears the panels one at a time.
- Each panel keeps its own interactive widget in HTML and its own static version in PDF and Word, so a static fallback can be drawn panel by panel.

Combine the panels in code instead when they share something that Quarto layout cannot give them:

- **A shared axis**: the panels must line up on one scale, such as one y-axis across panels, so a reader can compare heights across them. Separate subfigures each get their own axis limits and plot margins, so their axes line up only by chance.
- **A shared legend**: several panels use the same colors or line types, and one legend for all of them, collected with patchwork's `guides = "collect"`, saves repeating it.
- **One plot of subsets**: the panels are the same plot drawn for each level of a variable. Use a facet, whose strip labels name each level.
- **Linked interaction**: the panels should respond together, such as a shared zoom or hover across plotly panels with `subplot(shareX = TRUE)`.

A panel combined in code has no label of its own for a cross-reference. Tag the panels in the code, in the same style as Quarto's subcaptions, with `plot_annotation(tag_levels = "a", tag_prefix = "(", tag_suffix = ")")` in patchwork, and refer to them in the text as "panel (b) of `@fig-x`". Write one alt text for the whole figure that describes each panel in turn, and give the panels in the caption by their tags. A combined interactive figure, such as one plotly widget with two 3D scenes, needs a static version that shows every panel too.

Keep panels in separate subfigures when they share nothing. Two 3D surfaces from two models, each with its own rotation, are two subfigures, not one widget with two scenes.

Subfigures with language tabsets

Put a language tabset (`group="language"`) inside each subfigure, not the subfigures inside one tabset, when each panel has R and Python versions:

````
::: {#fig-surfaces layout-ncol=2}

::: {#fig-surfaces-additive}
::: {.panel-tabset group="language"}
## R
```r
# draw the additive model's surface
```
## Python
```python
# draw the additive model's surface
```
:::
The additive model.
:::

::: {#fig-surfaces-interaction}
<!-- the same tabset for the interaction model -->
````

```
The model with an interaction term.
:::

Fitted surfaces of the two models for fuel economy over weight and model year.
:::
```

Each `r` or `python` block stands for an executed chunk, whose fence is `{r}` or `{python}` in a real page. Quarto then labels the panels (a) and (b) whichever tab the reader picks. With the subfigures inside the tabset, the Python tab's panels become (c) and (d), and the R panels' cross-references point at a tab the reader may not be on.

Subtables

The same choice holds for tables:

- Put tables that a reader compares side by side, but that have different rows or columns, in subtables of a `#tbl-` div with `layout-ncol`, each with its own subcaption.
- Put tables that share their rows in one table, with a header over each group of columns, such as a `tab_spanner()` in the R package gt, so the rows line up and each row label appears once.

Combine several figures with patchwork

The R package patchwork combines several ggplot2 figures into one figure with panel labels. The `+` operator places plots side by side, and the `/` operator stacks them. In this demo, `p1`, `p2`, and `p3` stand for three ggplot2 plots that you have already made:

```r
library(patchwork)

(p1 + p2) / p3 +
  plot_layout(guides = "collect") +
  plot_annotation(tag_levels = "a", tag_prefix = "(", tag_suffix = ")") &
  theme(legend.position = "bottom")
```

The expression puts `p1` and `p2` side by side above `p3`:

- `plot_layout(guides = "collect")` merges the plots' legends into one.
- `plot_annotation(tag_levels = "a", tag_prefix = "(", tag_suffix = ")")` labels the panels (a), (b), and (c), the style of Quarto's subcaptions.
- `&` applies the bottom legend position to every panel.

Three-dimensional graphs

A function of two variables, such as a loss function or a residual sum of squares as a function of a regression's intercept and slope, is a surface. A contour plot shows where the function takes given values and is easy to read values from. A three-dimensional surface plot shows the same function as a landscape, so valleys, ridges and minima are visible at once. Show both when the shape of the surface matters.

- Use a 3D plot for a surface, with the function's value on the vertical axis, and pair it with a contour plot of the same function.
- Pair the other way too: every contour plot or level-curve plot of a function of two variables gets a corresponding 3D surface plot of the same function (Ezra, 2026-11-03), with the same colors and marked points.
- Put a contour plot and its 3D surface plot in one figure, as two subfigures (Ezra, 2026-11-12: "why didn't you pair the 2d and 3d figures as subfigures? that should be a psw principle by now"), so one caption describes both views, and a cross-reference such as "Figure 3b" names one of them (see Combine panels with Quarto layout or in code). In HTML the 3D subfigure is the interactive plot; in PDF, Word, and PowerPoint it is the static view at a fixed angle.
- Do not use 3D for data that are not a surface, such as bars or pie charts: perspective makes lengths and areas harder to compare.
- Choose the viewing angle so that the feature you discuss, such as the minimum, is not hidden, and label all three axes.
- Give the plot a cube-shaped box, with every axis the same length on screen (Ezra, 2026-11-10). When all three axes are in the same units, such as a surface in physical space, scale the axes equally in data units instead. In plotly, set the scene's `aspectmode` to `"cube"` or to `"data"`. Do not leave it unset: plotly's default, `"auto"`, then draws each axis in proportion to the span of its data unless one axis range is more than 4 times another, so a function whose values span a wider range than its inputs draws a tall, thin column.
- An interactive plot that readers can rotate, such as one made with plotly, helps on the web, but static formats such as PDF, Word, and PowerPoint show no interactive plot, so render a static view at a fixed angle there (see Scientific figures and Make figures accessible).
- In a Python tab, draw the interactive 3D plot with plotly too, so the R and Python tabs of a figure look alike. Keep the `matplotlib` version for the static formats, where it is the static view at a fixed angle, and run it only in those outputs, the way the R tab runs its plotly code only in HTML output. In a `knitr` chunk, print the figure as

one raw HTML block under `results: asis`, because `fig.show()` needs `ipython` and several `print` calls split the chunk's output:

```
print("```{=html}\n" + fig.to_html(full_html=False, include_plotlyjs="cdn") + "\n```")
```

End each call that returns the figure, such as `add_trace` and `update_layout`, with `;` so that reticulate does not print the figure after every call, and add `plotly` to the page's `reticulate::py_require` list.

- Give a 3D plot alt text that states what the surface looks like, such as "a bowl with its lowest point at …".

> **Source**
>
> Several of the resource links in the slide deck and figure sections began in the Benjamin-Chung Lab Manual by Jade Benjamin-Chung. The section on three-dimensional graphs is original to this guide.

For further reading, see the Professional Development and Resources chapters of the UCD-SERG lab manual.

Anscombe, F. J. 1973. "Graphs in Statistical Analysis." *The American Statistician* 27 (1): 17–21. https://doi.org/10.1080/00031305.1973.10478966.

Kirchoff, Bruce. 2021. "How to Tell a Compelling Story in Scientific Presentations." *Nature* 600 (7890): S88–89. https://doi.org/10.1038/d41586-021-03603-2.

Ratcliff, William C. 2023. "The Science of Storytelling: The David Attenborough Style of Scientific Presentation." *Trends in Molecular Medicine* 29 (9): 675–77. https://doi.org/10.1016/j.molmed.2023.05.002.

Rougier, Nicolas P., Michael Droettboom, and Philip E. Bourne. 2014. "Ten Simple Rules for Better Figures." *PLOS Computational Biology* 10 (9): e1003833. https://doi.org/10.1371/journal.pcbi.1003833.