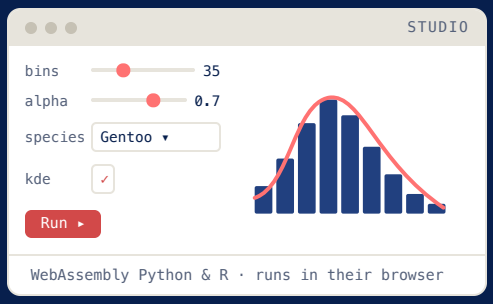
GoFigr Clean Room

Figures anyone can re-run

Clean Room turns one decorated function into an interactive app: stakeholders adjust parameters and re-run in the browser — no environment, no install, full provenance on every run.



YOUR NOTEBOOK

```
In [3]:
df.groupby('species')
  .flipper_length.describe()

In [12]:
# log scale? — nope

In [27]:
sns.histplot(df, ...)
# v3, tweak colors

In [41]:
# TODO clean up someday
```

Exploration stays exploratory — dead ends, experiments, all of it.

DISTILL →

THE CLEAN BOUNDARY

```
@reproducible(interactive=True)
def flipper_hist(
    data,  # passed in & serialized
    bins: int = SliderParam(20, min=5, max=100),
    species = DropdownParam("Adelie", choices=[…]),
    kde: bool = True,  # bool → checkbox
):
    sns.histplot(data[data.species == species],
                 x="flipper_length_mm",
                 bins=bins, kde=kde)

flipper_hist(penguins)  # captured on run
```

CROSSES THE LINE — declared params · passed data · packages

STAYS BEHIND — globals · local files · hidden state

SHARE →

THEIR BROWSER

SliderParam	
Literal / Dropdown	Adelie ▾
bool	✓
str	title…
DataFrame	bound to revision

Widgets come from the signature — no UI code, no separate app to maintain.

Every run is a revision

Stakeholders adjust parameters and re-run in the browser. Saving creates a new revision — linked to yours, parameter values included. You get notified; you rerun nothing.

stored	code · params · data · env
watermark	QR from output → revision
history	saved runs form one chain

a41f2c9 — you · first run
5b83d07 — you · bins=35
c92e4ab — you · kde=True
f21c9d3 — reviewer · species=Gentoo

Send a link, not a notebook

No install. No server. No setup instructions.

BROWSER RUNTIME
WebAssembly Python and R — packages install on demand, nothing runs server-side.

FULL STUDIO
Editable source, live variables, DataFrame previews, console output.

AI ASSISTANT
Natural-language edits with the figure, data, and packages in context.

Python `@reproducible` **R** `reproducible()`

app.gofigr.io/start

docs at gofigr.io/docs/features/clean-room

GOFIGR CLEAN ROOM

How Clean Room compares

What survives the handoff — and what it costs each side.

	RE-RUNNABLE by the recipient	WHAT-IFS no code edits	CONTEXT code · data · env	RUN HISTORY params included	RECIPIENT NEEDS to open it	AUTHOR LIFT to share it
Image in a deck also HTML export, nbviewer	—	—	—	—	○○○ nothing	○○○ zero
Notebook file `.ipynb`	if their env matches	—	code only	git, if you're diligent	●●● Python, packages, your data	●○○ attach a file
Binder	✓	—	repo contents	git commits	●●○ browser + cold start	●●○ public repo + env spec
Docker image	✓	—	baked in, opaque	image tags	●●● Docker + terminal	●●● Dockerfile + registry
Streamlit / Shiny app also Voilà, Panel, Dash	✓	✓	app only — code stays home	—	●○○ browser	●●● UI code + a server to keep alive
GoFigr Clean Room	✓	✓	✓ auto-captured	✓ linked revisions	●○○ browser	●○○ one line of code

EFFORT: ○○○ NONE ●○○ LOW ●●○ MEDIUM ●●● HIGH

Clean Room: interactive and inspectable, with minimal effort.

Python `@reproducible` **R** `reproducible()`

app.gofigr.io/start

docs at gofigr.io/docs/features/clean-room