Teach Once, Use Often

Building AI Skills for Data Analysis

Tyson Barrett · posit::conf(2026)

Environmental impact

Stochastic output

Skill atrophy

Verbose co

e offloading

Shallow learning

"AI is like a box of chocolates. You never know what you're gonna get."

Forrest Gump

CRAFTSMAN
NICKEL PLATED

Delegate

Trust AI to do the work entirely (or errors are easy to catch).

Collaborate

AI does the typing or planning. But you own the decisions.

Own

You own the judgment, the decisions, the typing, the planning, the output.

Delegate

Trust AI to do the work entirely (or errors are easy to catch).

Slide drafts, Quarto/Jupyter/Rmarkdown scaffolding, color palettes, code documentation, data labeling.

Collaborate

AI does the typing or planning. But you own the decisions.

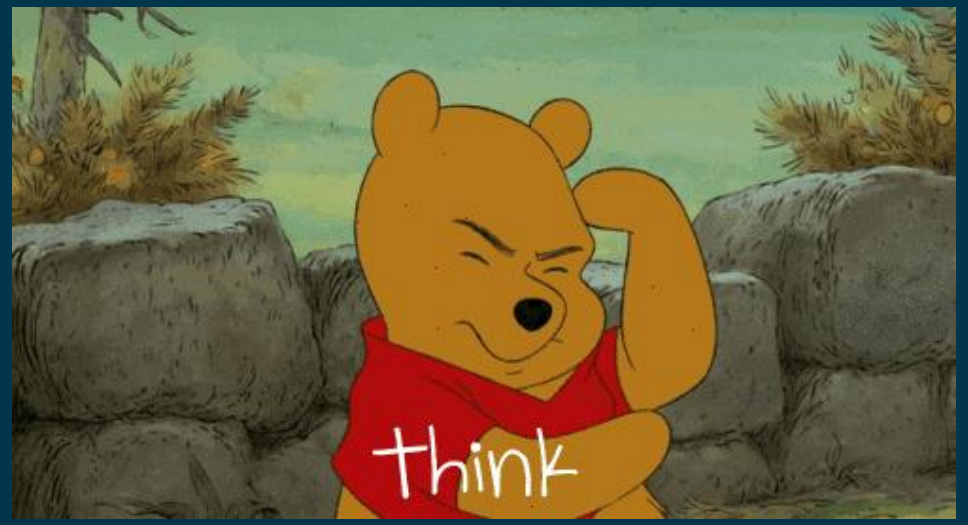
Drafting code, planning the analysis, exploratory data analysis, brainstorming solutions to data issues.

Own

You own the judgment, the decisions, the typing, the planning, the output.

The model specs, the estimand, the interpretation, improving code performance, reviewing code/output.

Pause and



1 CLAUDE.md (or Agents.md)

```
# Response Style

- Be concise. Skip preamble and restating question.
- Prefer code and tables over prose.

# Code Style

- Write clear code over clever code.
- Make small, reviewable changes. Explain what changed and why briefly.
- Default to R for data cleaning and wrangling and statistical models; default to python for machine learning. For all others, ask me.
- Use `ggplot2` and `plotnine` for data graphics
- Do not write comments that restate the code. Comment intent only.
- Manage python environments with `uv` (`uv add`, `uv run`). Do not use bare `pip install`.
- Descriptive and summary tables (`furniture::table1()`, `gtsummary`)
- Regression, multilevel, and Bayesian models (`lme4`, `brms`)
- Tables for rendering (`great_tables`); summaries via `tableone`
- For structural equation modeling and measurement models, use `lavaan`, `psych`, `semTools` in R
- Never commit data, credentials, or `.env` files.
- Don't push, rebase, or force anything without asking.
- In R, use `<-` for assignment. Two-space indentation. `snake_case` names.

# Data Analysis

- When doing exploratory data analysis, provide graphics over tables.
- When cleaning the data, do not change the data without asking me
- State your assumptions about all joins and filters
- Folder structure should look something like this:

```
data/raw/ # never modified
data/derived/
R/ or src/ # functions
analysis/ # numbered scripts or Quarto docs
output/ # tables, figures
```

Make Errors Visible

- Never drop NAs
- Create checks after any major data step
- Do not change the data to pass a check without asking.
- Run code to verify it works before telling me it does. If you couldn't run it, say so.
- Don't fabricate package functions, arguments, or citations. If unsure, check the docs or tell me you're unsure.
- Don't overwrite raw data.
- Don't delete files or outputs without asking.
- Don't claim results are significant, meaningful, or causal. Report the numbers; I interpret them.

Statistical Modeling

- I own every modeling decision.
- You can make recommendations but when coding, leave a #TODO where the models should go.
- If you think my specification is wrong, say so and explain why but do not make changes unless I ask for it.

Teaching

- For teaching materials, default to python using `pandas`, `statsmodels`, `bambi`, and `plotnine`
- Use Quarto for course materials.
```

2 SKILL.md

````
---
name: quarto-analysis
description: Builds reproducible Quarto (.qmd) documents for data analysis in R or Python. Use this skill whenever the user wants to create, restructure, or render a Quarto or R Markdown analysis, an analysis report, a notebook-style write-up of results, or says things like "write this up in Quarto", "make an analysis doc", "turn this script into a report", even if they don't say "Quarto" explicitly. Use when the user wants an HTML report, a PDF report, slides for a presentation, or a Word document presenting data.
---

# Quarto Data Analysis Documents

## Workflow
1. **Clarify scope.** Confirm the research question, data source, language (R or Python), and output format (HTML by default). If a project already exists, match its language and conventions.
2. **Start from the template.** Copy `assets/analysis-template-r.qmd` or `assets/analysis-template-py.qmd`. Don't write the YAML from scratch.
3. **Fill sections in order**: setup → data import → data checks → cleaning → descriptives → analysis (as specified) → diagnostics → session info.
4. **Choose chunk options deliberately.** Read `references/chunk-options.md` whenever you add figures, tables, cached chunks, or parameters.
5. **Report statistics correctly.** Read `references/stats-reporting.md` before writing any results prose, tables of estimates, or inline numbers.
6. **Render and verify** (see "Render checks"). A document that hasn't rendered isn't done.

## Document structure
Use these top-level sections. Skip ones that don't apply; don't invent new top-level ones
without reason.

```
# Purpose question, estimand/target in plain language, data source
# Setup packages, options, seed, paths (chunk hidden in output)
# Data import, dimensions, codebook/variable glossary
# Data checks types, missingness, duplicates, ranges, key assumptions
# Cleaning every exclusion with counts before/after
# Descriptives Table 1, distributions, key bivariate plots
# Analysis models exactly as specified by the analyst
# Diagnostics assumption checks, convergence, sensitivity (as requested)
# Summary results with uncertainty; limitations
# Reproducibility session info, package versions, render date
```

## Core rules (and why)

- **Be brief in prose.** Do not explain things in long paragraphs. Keep any commentary short.
- **Label every chunk.** Labels make errors traceable, enable caching, and are required for cross-references. Use `fig-` and `tbl-` prefixes for figures and tables.
- **Use `#|` chunk options, not `{r, echo=FALSE}`.** Hash-pipe syntax works in both knitr and Jupyter engines and keeps options readable.
- **Put shared defaults in YAML `execute:`**, override per chunk only when needed.
- **Never hard-code numbers in prose.** Use inline code (`` `r x` `` or `` `{python} x` ``) so text and results can't drift apart.
- **Keep heavy logic out of the .qmd.** Source functions from `R/` or `src/`. The document should read as a narrative with short chunks.
- **Raw data is read-only.** Read from `data/raw/`, write derived data to `data/derived/`.
- **Set a seed once in setup** and state it in the text, for anything stochastic.
- **Show exclusions as a flow.** Report n at each cleaning step; readers need to know who the results apply to.
- **Don't suppress warnings globally for model chunks.** Convergence and singular-fit warnings are results. Suppress package-load messages only.
- **Missing data is reported, never silently dropped.** State the n used in every table and model.

## Language defaults

**R (knitr engine):** `tidyverse` for wrangling, `furniture::table1()` or `gtsummary` for descriptive tables, `ggplot2` for figures, `broom`/`broom.mixed` or `parameters` for tidying estimates, `gt` or `great_tables`-style output for display. Use `renv` if the project has a lockfile.


**Python (jupyter engine):** `polars` or `pandas`, `tableone` for descriptives, `great_tables` for display, `statsmodels`/`pingouin` for classical stats, `plotnine` for figures (unless built in function is easy and clear). Environments via `uv`; render with
`uv run quarto render`.


## Render checks
After rendering, confirm:
- [ ] Render completes from a clean session (`quarto render analysis.qmd`) with no errors
- [ ] No unexpected warnings in the output (expected ones are explained in text)
- [ ] Every figure and table has a caption
- [ ] Every number in prose comes from inline code
- [ ] n is stated for each table and model
- [ ] Session info section is present
- [ ] If `cache: true` was used anywhere, re-render with `--cache-refresh` once before sharing

Report any check that fails instead of declaring the document finished.

## Files in this skill

- `references/chunk-options.md` — chunk and YAML options with when to use each
- `references/stats-reporting.md` — how to present estimates, uncertainty, diagnostics
- `assets/analysis-template-r.qmd` — starting template (R/knitr)
- `assets/analysis-template-py.qmd` — starting template (Python/Jupyter)
````

1 CLAUDE.md

```
# Response Style

- Be concise. Skip preamble and restating question.
- Prefer code and tables over prose.

# Code Style

- Write clear code over clever code.
- Make small, reviewable changes. Explain
- Default to R for data cleaning and wrangling
me.
- Use `ggplot2` and `plotnine` for data graphics
- Do not write comments that restate the code. Comment in
- Manage python environments with `uv` (`uv add`, `uv run
- Descriptive and summary tables (`furniture::table1()`,
- Regression, multilevel, and Bayesian models (`lme4`, `brms`)
- Tables for rendering (`great_tables`); summaries via `tableone`
- For structural equation modeling and measurement models, use `lavaan`, `psych`, `semTools` in R
- Never commit data, credentials, or `.env` files.
- Don't push, rebase, or force anything without asking.
- In R, use `<-` for assignment. Two-space indentation. `snake_case` names.

# Data Analysis

- When doing exploratory data analysis, provide graphics over tables.
- When cleaning the data, do not change the data without asking me
- State your assumptions about all joins and filters
- Folder structure should look something like this:

```
data/raw/ # never modified
data/derived/
R/ or src/ # functions
analysis/ # numbered scripts or Quarto docs
output/ # tables, figures
```

Make Errors Visible

- Never drop NAs
- Create checks after any major data step
- Do not change the data to pass a check without asking.
- Run code to verify it works before telling me it does. If you couldn't run it, say so.
- Don't fabricate package functions, arguments, or citations. If unsure, check the docs or tell me you're unsure.
- Don't overwrite raw data.
- Don't delete files or outputs without asking.
- Don't claim results are significant, meaningful, or causal. Report the numbers; I interpret them.

Statistical Modeling

- I own every modeling decision.
- You can make recommendations but when coding, leave a #TODO where the models should go.
- If you think my specification is wrong, say so and explain why but do not make changes unless I ask for it.

Teaching

- For teaching materials, default to python using `pandas`, `statsmodels`, `bambi`, and `plotnine`
- Use Quarto for course materials.
```

Response Style

- Be concise. Skip preamble and restating question.
- Prefer code and tables over prose.

2 SKILL.md

1 CLAUDE.md

Response Style

- Be concise. Skip preamble and restating question.
- Prefer code and tables over prose.

 # Code Style

- Write clear code over clever code.
- Make small, reviewable changes. Explain what changed and why briefly.
- Default to R for data cleaning and wrangling and statistical models; default to python for machine learning. For all others, ask me.
- Use `ggplot2` and `plotnine` for data graphics
- Do not write comments that restate the code. Comment intent only.
- Manage python environments with `uv` (`uv add`, `uv run`). Do not use bare `pip install`.
- Descriptive and summary tables (`furniture::table1()`, `gtsummary`)
- Regression, multilevel, and Bayesian models (`lme4`, `brms`)
- Tables for rendering (`great_tables`); summaries via `tableone`
- For structural equation modeling and measurement models, use `lavaan`, `psych`, `se
- Never commit data, credentials, or `.env` files.
- Don't push, rebase, or force anything without asking.
- In R, use `<-` for assignment. Two-space indentation. `snake_case` names.

Data Analysis

- When doing exploratory data analysis, provide graphics over tab
- When cleaning the data, do not change the data without asking m
- State your assumptions about all joins and filters
- Folder structure should look something like this:

```
data/raw/ # never modified
data/derived/
R/ or src/ # functions
analysis/ # numbered scripts or Quarto docs
output/ # tables, figures
```

Make Errors Visible

- Never drop NAs
- Create checks after any major data step
- Do not change the data to pass a check without asking.
- Run code to verify it works before telling me it does. If you couldn't run it, say
- Don't fabricate package functions, arguments, or citations. If unsure, check the do
- Don't overwrite raw data.
- Don't delete files or outputs without asking.
- Don't claim results are significant, meaningful, or causal. Report the numbers; I i

Statistical Modeling

- I own every modeling decision.
- You can make recommendations but when coding, leave a #TODO where the models should go.
- If you think my specification is wrong, say so and explain why but do not make changes unless I ask for it.

Teaching

- For teaching materials, default to python using `pandas`, `statsmodels`, `bambi`, and `plotnine`
- Use Quarto for course materials.

 # Code Style

- Use `ggplot2` and `plotnine` for data graphics
- Do not write comments that restate the code. Comment intent only.
- Use functions from packages rather than writing new complex functions.
- Descriptive and summary tables (`furniture::table1()`, `gtsummary`)

2 SKILL.md

1 CLAUDE.md

2 SKILL.md

```
# Response Style

- Be concise. Skip preamble and restating question.
- Prefer code and tables over prose.

# Code Style

- Write clear code over clever code.
- Make small, reviewable changes. Explain what changed and why briefly.
- Default to R for data cleaning and wrangling and statistical models; default to python for machine learning. For all others, ask
me.
- Use `ggplot2` and `plotnine` for data graphics
- Do not write comments that restate the code. Comment intent only.
- Manage python environments with `uv` (`uv add`, `uv run`). Do not use bare `pip install`.
- Descriptive and summary tables (`furniture::table1()`, `gtsummary`)
- Regression, multilevel, and Bayesian models (`lme4`, `brms`)
- Tables for rendering (`great_tables`); summaries via `tableone`
- For structural equation modeling and measurement models, use `lavaan`, `psych`,
- Never commit data, credentials, or `.env` files.
- Don't push, rebase, or force anything without asking.
- In R, use `<-` for assignment. Two-space indentation. `snake_case` names.

# Data Analysis

- When doing exploratory data analysis, provide graphics over tables.
- When cleaning the data, do not change the data without asking me
- State your assumptions about all joins and filters
- Folder structure should look something like this:

```
data/raw/ # never modified
data/derived/
R/ or src/ # functions
analysis/ # numbered scripts or Quarto docs
output/ # tables, figures
```

Make Errors Visible

- Never drop NAs
- Create checks after any major data step
- Do not change the data to pass a check without asking.
- Run code to verify it works before telling me it does. If you couldn't run it, say
- Don't fabricate package functions, arguments, or citations. If unsure, check the
- Don't overwrite raw data.
- Don't delete files or outputs without asking.
- Don't claim results are significant, meaningful, or causal. Report the numbers; I

Statistical Modeling

- I own every modeling decision.
- You can make recommendations but when coding, leave a #TODO where the models should go.
- If you think my specification is wrong, say so and explain why but do not make changes unless I ask for it.

Teaching

- For teaching materials, default to python using `pandas`, `statsmodels`, `bambi`, and `plotnine`
- Use Quarto for course materials.
```

```
# Make Errors Visible

- Never drop NAs
- Create checks after any major data step
- Do not change the data to pass a check without asking.
- Don’t overwrite raw data.
- Don’t delete files or outputs without asking.
- When updating a file, create a backup so no other changes get lost.
```

1 CLAUDE.md

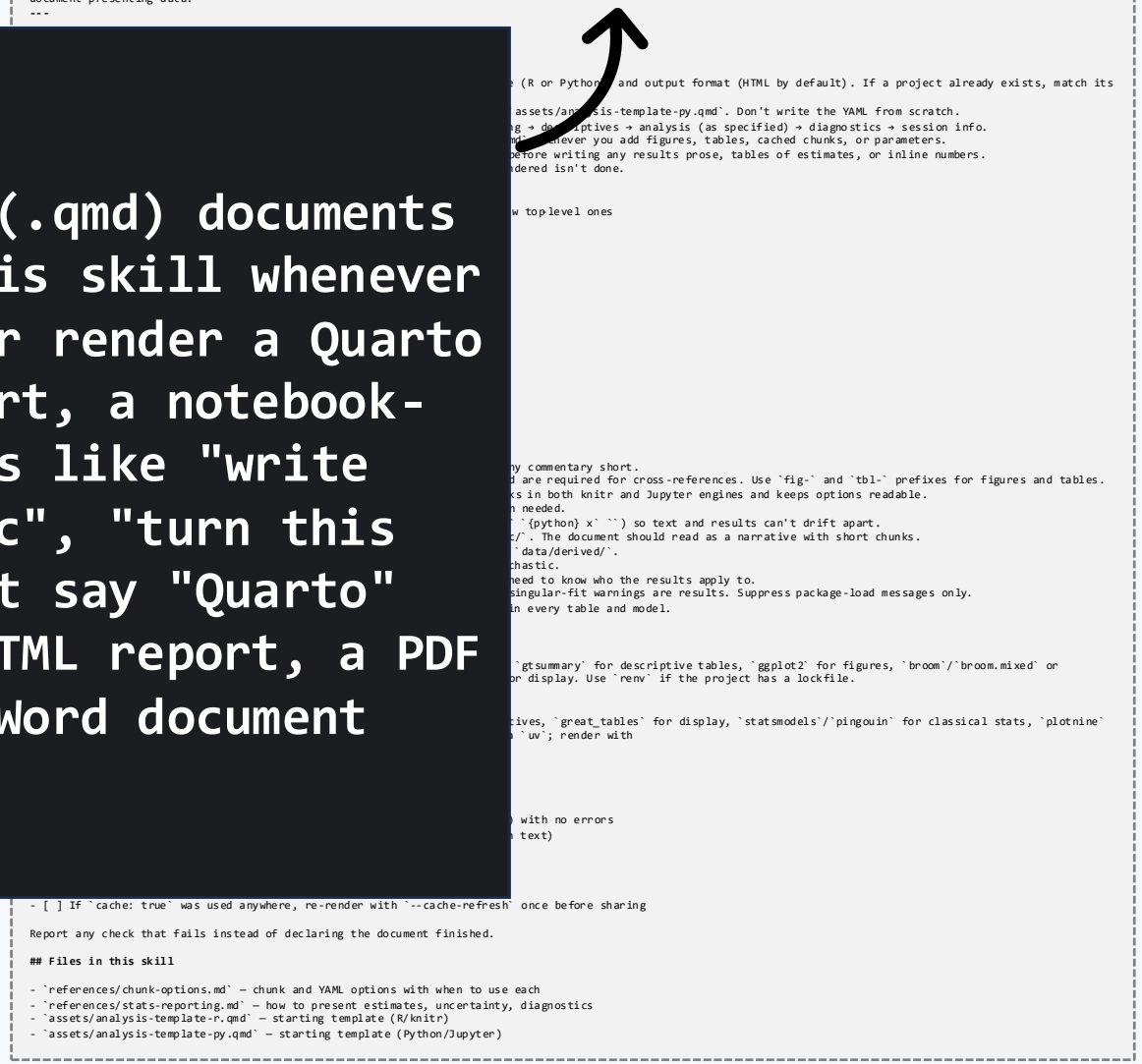
2 SKILL.md

```
# Response Style

- Be concise. Skip preamble and restating question.
- Prefer code and tables over prose.

# Code Style

- Write clear code over clever code.
- Make small, reviewable changes. Explain what changed and why briefly.
- Default to R for data cleaning and wrangling and statistical models; default to python for machine learning. For all others, ask
me.
- Use `ggplot2` and `plotnine` for data graphics
- Do not write comments that restate the code. Comment intent only.
- Manage python environments with `uv` (`uv add`, `uv run`). Do not use bare `pip install`.
- Descriptive and summary tables (`furniture::table1()`, `gtsummary`)
- Regression, multilevel, and Bayesian models (`lme4`, `brms`)
- Tables for rendering (`great_tables`); summaries via `tableone`
- For structural equation modeling and measurement models, use `lavaan`, `psych`, `semTools` in R
- Never commit data, credentials, or `.env` files.
- Don't push, rebase, or force anything without asking.
- In R, use `<-` for assignment. Two-space indentation. `snake_case` names.

# Data Analysis

- When doing exploratory data analysis, provide graphics over tables.
- When cleaning the data, do not change the data without asking me
- State your assumptions about all joins and filters
- Folder structure should look something like this:

```
data/raw/ # never modified
data/derived/
R/ or src/ # functions
analysis/ # numbered scripts or Quarto docs
output/ # tables, figures
```

Make Errors Visible

- Never drop NAs
- Create checks after any major data step
- Do not change the data to pass a check without asking.
- Run code to verify it works before telling me it does. If you couldn't run it, say so.
- Don't fabricate package functions, arguments, or citations. If unsure, check the
- Don't overwrite raw data.
- Don't delete files or outputs without asking.
- Don't claim results are significant, meaningful, or causal. Report the numbers; I

Statistical Modeling

- I own every modeling decision.
- You can make recommendations but when coding, leave a #TODO where the models should go.
- If you think my specification is wrong, say so and explain why but do not make changes unless I ask for it.

Teaching

- For teaching materials, default to python using `pandas`, `statsmodels`, `bambi`, and `plotnine`
- Use Quarto for course materials.
```

```
# Statistical Modeling

- I own every modeling decision.
- You can make recommendations but when coding, leave a #TODO
where the models should go.
- If you think my specification is wrong, say so and explain
why but do not make changes unless I ask for it.
```

1 CLAUDE.md

2 SKILL.md



```
---
name: quarto-analysis
description: Builds reproducible Quarto (.qmd) documents for data analysis in R or Python. Use this skill whenever the user wants to create, restructure, or render a Quarto or R Markdown analysis, an analysis report, a notebook-style write-up of results, or says things like "write this up in Quarto", "make an analysis doc", "turn this script into a report", even if they don't say "Quarto" explicitly. Use when the user wants an HTML report, a PDF report, slides for a presentation, or a Word document presenting data.
---
```

1 CLAUDE.md

2 SKILL.md

```
## Workflow

1. **Clarify scope.** Confirm the research question, data source, language (R or Python), and output format (HTML by default). If a project already exists, match its language and conventions.
2. **Start from the template.** Copy `assets/analysis-template-r.qmd` or `assets/analysis-template-py.qmd`. Don't write the YAML from scratch.
3. **Fill sections in order**: setup → data import → data checks → cleaning → descriptives → analysis (as specified) → diagnostics → session info.
...
```

1 CLAUDE.md

2 SKILL.md

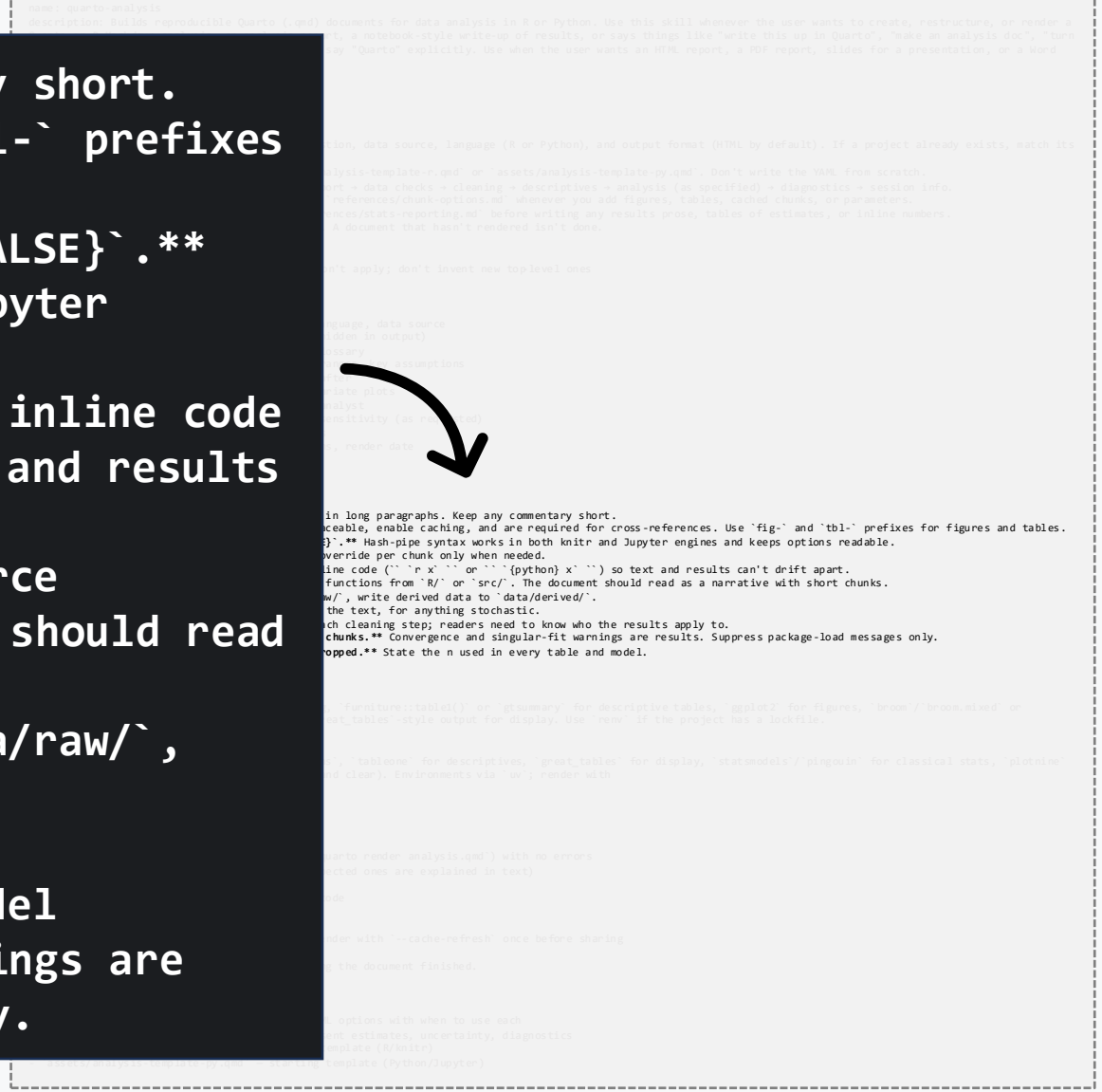
Workflow

1. **Clarify scope.** Confirm the research question, data source, language (R or Python), and output format (HTML by default). If a project already exists, match its language and conventions.
2. **Start from the template.** Copy `assets/analysis-template-r.qmd` or `assets/analysis-template-py.qmd`. Don't write the YAML from scratch.
3. **Fill sections in order**: setup → data import → data checks → cleaning → descriptives → analysis (as specified) → diagnostics → session info.
...

1 CLAUDE.md

- **Be brief in prose.** Keep any commentary short.
- **Label every chunk.** Use `fig-` and `tbl-` prefixes for figures and tables.
- **Use `#|` chunk options, not `{r, echo=FALSE}`.** Hash-pipe syntax works in both knitr and Jupyter engines and keeps options readable.
- **Never hard-code numbers in prose.** Use inline code (`` `r x` `` or `` `{python} x` ``) so text and results can't drift apart.
- **Keep heavy logic out of the .qmd.** Source functions from `R/` or `src/`. The document should read as a narrative with short chunks.
- **Raw data is read-only.** Read from `data/raw/`, write derived data to `data/derived/`.
- **Report n at each cleaning step**.
- **Don't suppress warnings globally for model chunks.** Convergence and singular-fit warnings are results. Suppress package-load messages only.

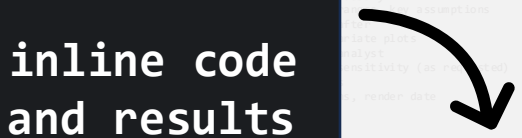
2 SKILL.md



1 CLAUDE.md

- **Be brief in prose.** Keep any commentary short.
- **Label every chunk.** Use `fig-` and `tbl-` prefixes for figures and tables.
- **Use `#|` chunk options, not `{r, echo=FALSE}`.** Hash-pipe syntax works in both knitr and Jupyter engines and keeps options readable.
- **Never hard-code numbers in prose.** Use inline code (`` `r x` `` or `` `{python} x` ``) so text and results can't drift apart.
- **Keep heavy logic out of the .qmd.** Source functions from `R/` or `src/`. The document should read as a narrative with short chunks.
- **Raw data is read-only.** Read from `data/raw/`, write derived data to `data/derived/`.
- **Report n at each cleaning step**.
- **Don't suppress warnings globally for model chunks.** Convergence and singular-fit warnings are results. Suppress package-load messages only.

2 SKILL.md



1 CLAUDE.md

```
# Response Style

- Be concise. Skip preamble and restating question.
- Prefer code and tables over prose.

# Code Style

- Write clear code over clever code.
- Make small, reviewable changes. Explain what changed and why briefly.
- Default to R for data cleaning and wrangling and statistical models; default to python for machine learning. For all others, ask me.
- Use `ggplot2` and `plotnine` for data graphics
- Do not write comments that restate the code. Comment intent only.
- Manage python environments with `uv` (`uv add`, `uv run`). Do not use bare `pip install`.
- Descriptive and summary tables (`furniture::table1()`, `gtsummary`)
- Regression, multilevel, and Bayesian models (`lme4`, `brms`)
- Tables for rendering (`great_tables`); summaries via `tableone`
- For structural equation modeling and measurement models, use `lavaan`, `psych`, `semTools` in R
- Never commit data, credentials, or `.env` files.
- Don't push, rebase, or force anything without asking.
- In R, use `<-` for assignment. Two-space indentation. `snake_case` names.

# Data Analysis

- When doing exploratory data analysis, provide graphics over tables
- When cleaning the data, do not change the data without asking me
- State your assumptions about all joins and filters
- Folder structure should look something like this:

```
data/raw/ # never modified
data/derived/
R/ or src/ # functions
analysis/ # numbered scripts or Quarto docs
output/ # tables, figures
```

Make Errors Visible

- Never drop NAs
- Create checks after any major data step
- Do not change the data to pass a check without asking me. If you couldn't run it, say so.
- Run code to verify it works before telling me it does.
- Don't fabricate package functions, arguments, or citations. If unsure, check the docs or tell me you're unsure.
- Don't overwrite raw data.
- Don't delete files or outputs without asking.
- Don't claim results are significant, meaningful, or causal. Report the numbers; I interpret them.

Statistical Modeling

- I own every modeling decision.
- You can make recommendations but when coding, leave a #TODO where the models should go.
- If you think my specification is wrong, say so and explain why but do not make changes unless I ask for it.

Teaching

- For teaching materials, default to python using `pandas`, `statsmodels`, `bambi`, and `plotnine`
- Use Quarto for course materials.
```

DELEGATE

DELEGATE

COLLABORATE

DELEGATE

OWN

COLLABORATE

2 SKILL.md

```
---
name: quarto-analysis
description: Builds reproducible Quarto (.qmd) documents for data analysis in R or Python. Use this skill whenever the user wants to create, restructure, or render a Quarto or R Markdown analysis, an analysis report, a notebook-style write-up of results, or says things like "write this up in Quarto", "make an analysis doc", "turn this script into a report", even if they don't say "Quarto" explicitly. Use when the user wants an HTML report, a PDF report, slides for a presentation, or a Word document presenting data.
---

# Quarto Data Analysis Documents

## Workflow
1. **Clarify scope.** Confirm the research question, data source, language (R or Python), and output format (HTML by default). If a project already exists, match its language and conventions.
2. **Start from the template.** Copy `assets/analysis-template-r.qmd` or `assets/analysis-template-py.qmd`. Don't write the YAML from scra
3. **Fill sections in order**: setup → data import → data checks → cleaning → descriptives → analysis (as specified) → diagnostics → sessi
4. **Choose chunk options deliberately.** Read `references/chunk-options.md` whenever you add figures, tables, cached chunks, or parameters.
5. **Report statistics correctly.** Read `references/stats-reporting.md` before writing any results prose, tables of estimates, or inline numbers.
6. **Render and verify** (see "Render checks"). A document that hasn't rendered isn't done.

## Document structure
Use these top-level sections. Skip ones that don't apply; don't invent new top-level ones without reason.

```
# Purpose question, estimand/target in plain language, data source
# Setup packages, options, seed, paths (chunk hidden in output)
# Data import, dimensions, codebook/variable glossary
# Data checks types, missingness, duplicates, ranges, key assumptions
# Cleaning every exclusion with counts before/after
# Descriptives Table 1, distributions, key bivariate plots
# Analysis models exactly as specified by the analyst
# Diagnostics assumption checks, convergence, sensitivity (as requested)
# Summary results with uncertainty; limitations
# Reproducibility session info, package versions, render date
```

Core rules (and why)

- **Be brief in prose.** Do not explain things in long paragraphs. Keep any commentary short.
- **Label every chunk.** Labels make errors traceable, enable caching, and are required for cross-references. Use `fig-` and `tbl-` prefixes for figures and tables.
- **Use `#|` chunk options, not `{r, echo=FALSE}`.** Hash-pipe syntax works in both knitr and Jupyter engines and keeps options readable.
- **Put shared defaults in YAML `execute:`**, override per chunk only when needed.
- **Never hard-code numbers in prose.** Use inline code (`` `r x` `` or `` `{python} x` ``) so text and results can't drift apart.
- **Keep heavy logic out of the .qmd.** Source functions from `R/` or `src/`. The document should read as a narrative with short chunks.
- **Raw data is read-only.** Read from `data/raw/`, write derived data to `data/derived/`.
- **Set a seed once in setup** and state it in the text, for anything stochastic.
- **Show exclusions as a flow.** Report n at each cleaning step; readers need to know who the results apply to.
- **Don't suppress warnings globally for model chunks.** Convergence and singular-fit warnings are results. Suppress package-load messages only.
- **Missing data is reported, never silently dropped.** State the n used in every table and model.

Language defaults

R (knitr engine): `tidyverse` for wrangling, `furniture::table1()` or `gtsummary` for descriptive tables, `ggplot2` for figures, `broom`/`broom.mixed` or `parameters` for tidying estimates, `gt` or `great_tables`-style output for display. Use `renv` if the project has a lockfile.

Python (jupyter engine): `polars` or `pandas`, `tableone` for descriptives, `great_tables` for display, `statsmodels`/`pingouin` for classical stats, `plotnine` for figures (unless built in function is easy and clear). Environments via `uv`; render with `uv run quarto render`.

Render checks
After rendering, confirm:
- [ ] Render completes from a clean session (`quarto render analysis.qmd`) with no errors
- [ ] No unexpected warnings in the output (expected ones are explained in text)
- [ ] Every figure and table has a caption
- [ ] Every number in prose comes from inline code
- [ ] n is stated for each table and model
- [ ] Session info section is present
- [ ] If `cache: true` was used anywhere, re-render with `--cache-refresh` once before sharing

Report any check that fails instead of declaring the document finished.

Files in this skill

- `references/chunk-options.md` — chunk and YAML options with when to use each
- `references/stats-reporting.md` — how to present estimates, uncertainty, diagnostics
- `assets/analysis-template-r.qmd` — starting template (R/knitr)
- `assets/analysis-template-py.qmd` — starting template (Python/Jupyter)
```

COLLABORATE

DELEGATE

OWN

COLLABORATE

DELEGATE

assistant.posit.co

A note about Posit Assistant

Thanks!

tysonbarrett.com/assets/posit26/posit2026.pdf


tyson.barrett@usu.edu


github.com/tysonstanley


tysonbarrett.com

Photography by the artists at Unsplash

RESOURCES

Templates and Examples

opensource.posit.co/software/skills

github.com/nimrodfisher/data-analytics-skills

opensource.posit.co/software/skills

Posit's skill library

github.com/posit-dev/skills

opensource.posit.co/software/skills

Posit AI

posit.co/blog/introducing-ai-in-rstudio