Maternal Health
Data Innovation &
Coordination Hub

20 March 2025

JOHNS HOPKINS
UNIVERSITY

Data Visualization:
Choosing the best tools for the job

Erik Westlund, Ph.D.
Johns Hopkins Biostatistics Center
Johns Hopkins Bloomberg School of Public Health

Resources

- This presentation uses HCUP data from the AHRQ from 9 states
- Nearly all of code for the provided examples for this presentation are available on GitHub at: https://github.com/erikwestlund/mhviz-tools

Motivation

- Prior presentation focused on “the substance, statistics, and design behind good, honest graphics”
- Today, we are going to focus on more practical matters of visualization where we have some data and we need to visualize it and need to choose the “best tools for the job”

Good Visualization & Good Science

- Good visuals not only tell the truth, but effectively communicate it
- Poor methods lead to poor visualizations: “being pretty” is not enough
- Good scientific workflows that produce trustworthy data and results are a necessary, but not sufficient condition, for good data visualization
- Good workflows require picking tools fit for the job at hand

What Do We Mean By Tools?

- We can think of tools in two ways:

1. Tools as **software and workflows**: for example, "Microsoft Excel can be used to store data, from which we can create a bar plot for a report."

2. Tools as **methods of visualization**: for example, "Bar plots are a good tool for visualizing differences in a single statistic across groups."

- This presentation will focus on software and workflows.

Case Study: Microsoft Excel

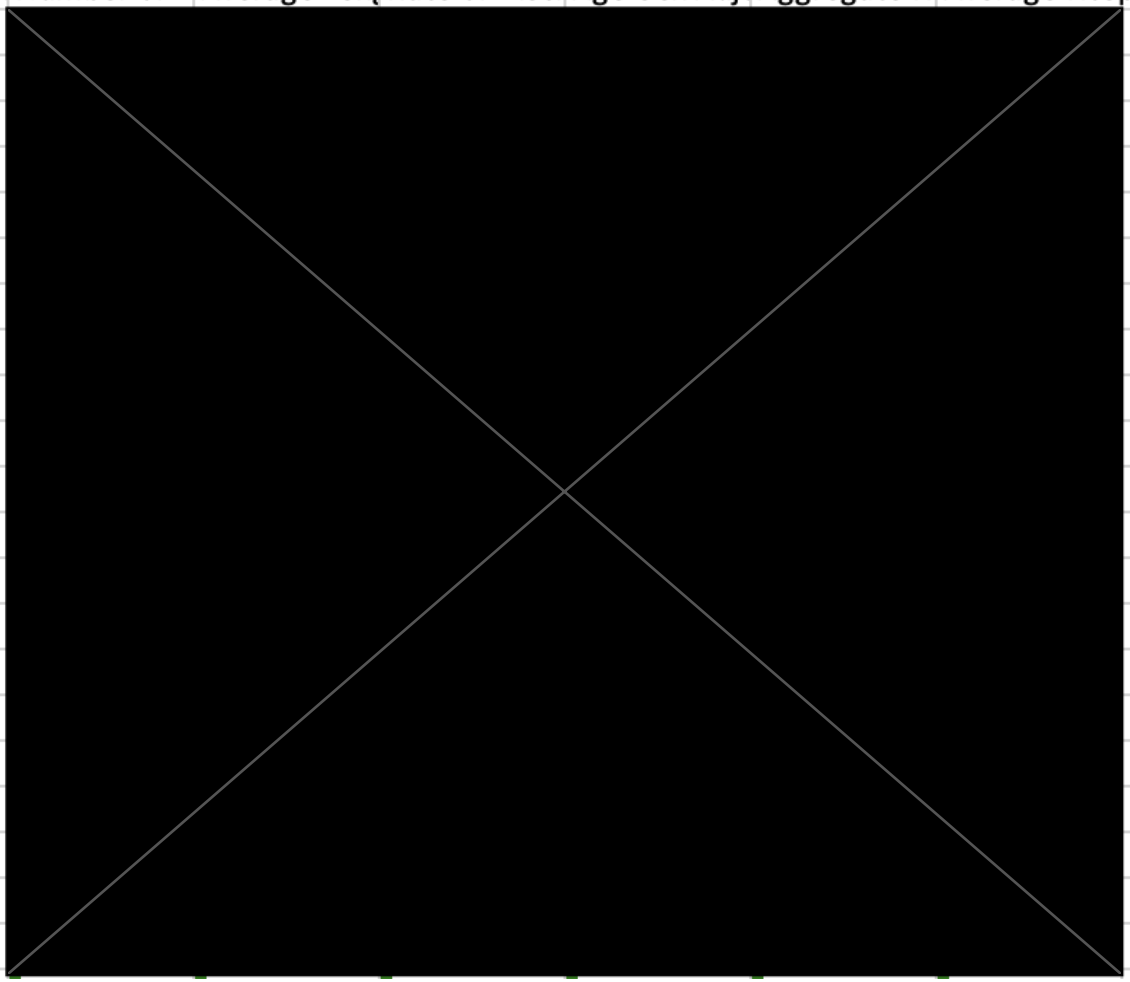
- Excel is one of the most important, versatile, and powerful pieces of software ever written
- It can combine data management, analysis, and table/figure generation into a single file that can easily be shared
- It can run code that can, in principle, do almost anything you can imagine
- For this reason, it is a staple of the modern workplace.

Utility of Excel: HCUP County Data

- HCUP stands for the "Healthcare Cost and Utilization Project"
- It's a collection of databases managed by the Agency for Healthcare Research and Quality (AHRQ)
- Data file: Inpatient Hospitalizations with a principal diagnosis of "Hypertension complicating pregnancy/birth"
- Unit of analysis: County
- Five sheets: Instructions, Title, Footnotes, Table Data, Map Data
- Comprehensive information in a single file
- Ability to extend the file (e.g., add new sheets)
- Ability to visually inspect data using sorting, etc.

The Data

	1	2	3	4	5	6	7	8	9	10	11
1	County	FIPS code	Patient Chara	Number of D	Average Leng	Rate of Disch	Age-Sex Adju	Aggregate H	Average Hospital Costs per Stay (in $)		
2	US Total		Overall								
3	State Total	55	Overall								
4	Adams	55001	Overall								
5	Ashland	55003	Overall								
6	Barron	55005	Overall								
7	Bayfield	55007	Overall								
8	Brown	55009	Overall								
9	Buffalo	55011	Overall								
10	Burnett	55013	Overall								
11	Calumet	55015	Overall								
12	Chippewa	55017	Overall								
13	Clark	55019	Overall								
14	Columbia	55021	Overall								
15	Crawford	55023	Overall								
16	Dane	55025	Overall								
17	Dodge	55027	Overall								
18	Door	55029	Overall								
19	Douglas	55031	Overall								
20	Dunn	55033	Overall								
21	Eau Claire	55035	Overall								
22	Florence	55037	Overall								



Lets Make an Excel Bar Chart

- Lets compare the average cost per hospital stay, by county, for patients with a principal diagnosis of hypertension complicating pregnancy/birth

Step 1: Clean Data

- The data file has two related problems:
  1. There are missing data coded with an asterisk (missing means: 2 or fewer hospitals, 11 or fewer discharges, or large standard error)
  2. Because there are asterisks in the data, the data type is seen as a string

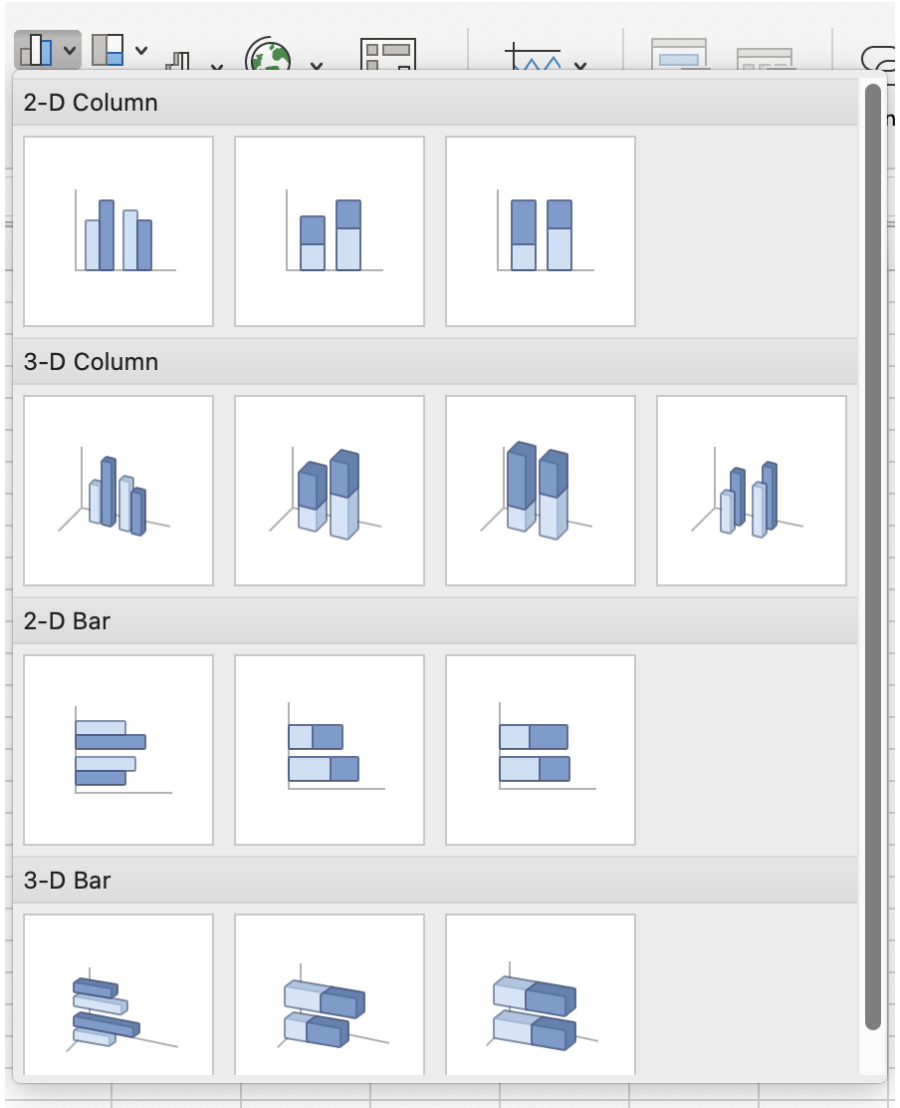
Step 1: Clean Data (cont.)

- Since we want to preserve the original data, we need to copy the data to a new sheet
- To make things easier, we'll remove columns not of interest
  - Highlight columns -> right click -> delete
- Since we cannot visualize data with asterisks, we need to remove the rows with asterisks
  - Sort -> right click -> delete -> sort back (14 columns deleted)

	1	2	3	4
1	County	Average Hospital Costs per Stay (in $)		
2	US Total			
3	State Total			
4	Ashland			
5	Barron			
6	Brown			
7	Buffalo			
8	Calumet			
9	Chippewa			
10	Clark			
11	Columbia			
12	Crawford			
13	Dane			
14	Dodge			
15	Door			
16	Douglas			
17	Dunn			
18	Eau Claire			
19	Fond du Lac			
20	Forest			
21	Grant			
22	Green			
23	Green Lake			
24	Iowa			
25	Jackson			

Step 2: Visualize data

- Highlight data of interest
- Click insert
- Click chart -> Bar
- Didn’t work



Step 3: Debug

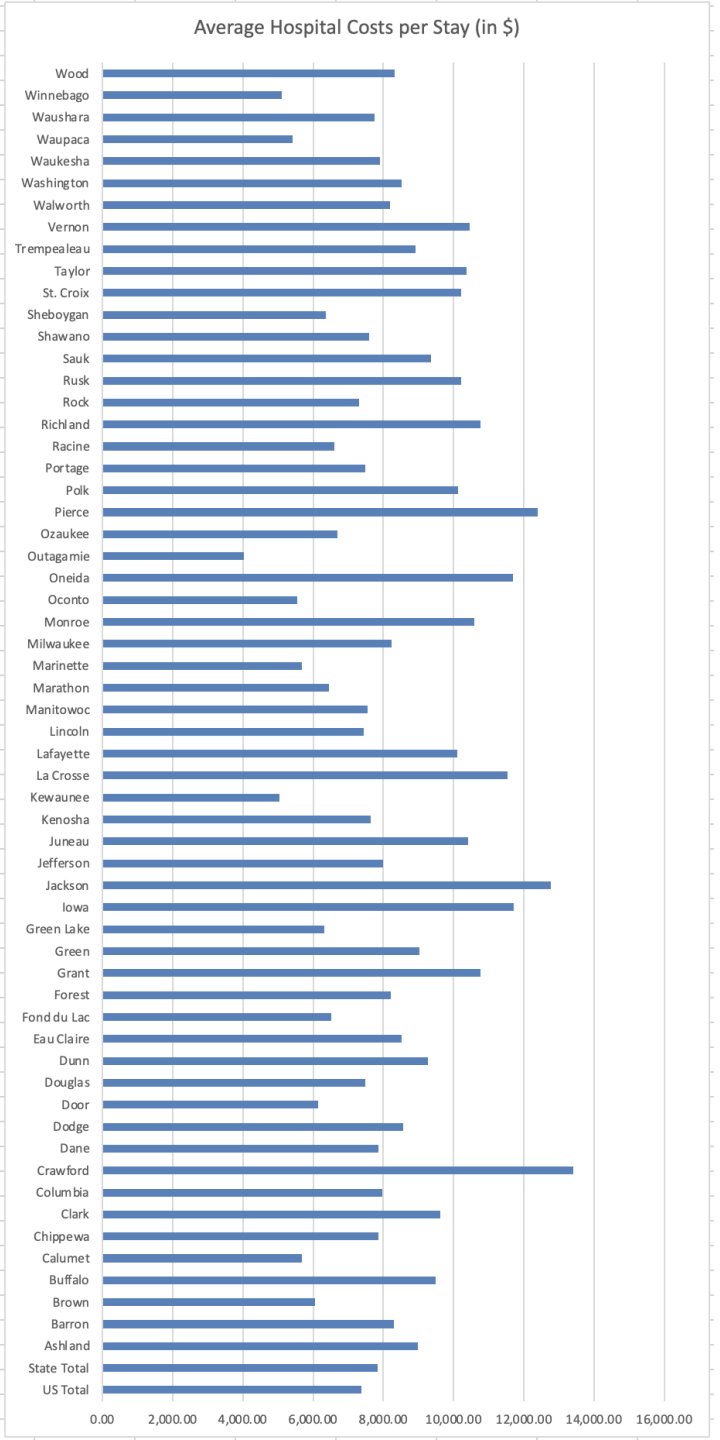
- It didn't give us a chart and we have no idea why
- Reason: it doesn't know the cost data is a number, namely a dollar
- Worse, I cannot coerce it using format.
- I had to copy the data out to a text file, and copy it back
- Then it right-aligned, hinting to me that it now knows it's a number

	1	2	3	4
1	County	Average Hospital Costs per Stay (in $)		
2	US Total			
3	State Total			
4	Ashland			
5	Barron			
6	Brown			
7	Buffalo			
8	Calumet			
9	Chippewa			
10	Clark			
11	Columbia			
12	Crawford			
13	Dane			
14	Dodge			
15	Door			
16	Douglas			
17	Dunn			

Step 4: Try again

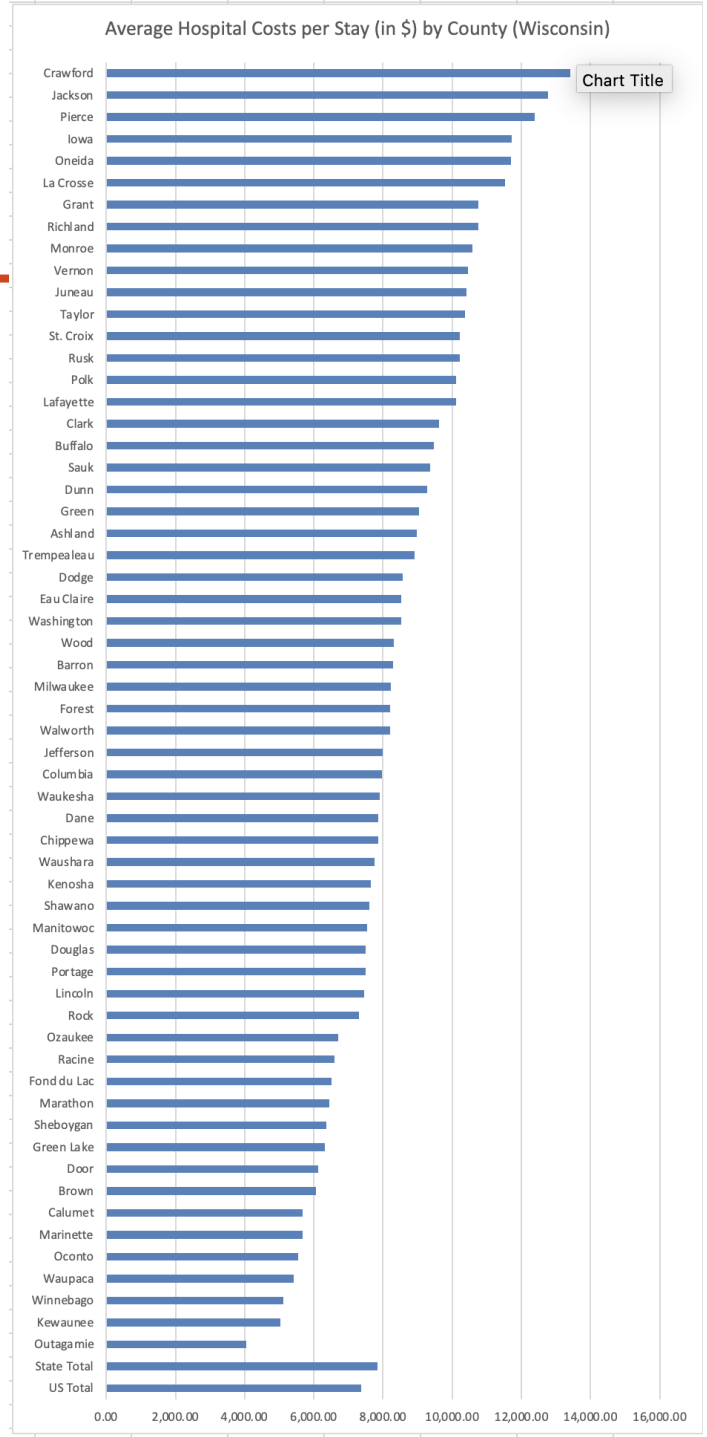
- Highlight data of interest
- Click insert
- Click chart -> Bar

- Problems:
  - It is reverse sorted
  - Title unclear
- Preference: US, State at top, rest below sorted most expensive toe least expensive



Step 5: Polish

- Sort the data, but keep US/State separate
- Double click the graph, change the title



What's Good About This?

- I have a single file that contains everything we need to know about this data file that can be easily sent around
- I am able to point and click to do things like change titles, colors, design styles, the order of data
- I can do things like separate US/State totals from the rest by hand
- Saving a figure is as easy as: Right click -> Save as picture
- Without any programming skill I can create a publication quality figure to use in a report

What's Not So Good?

- While this workflow provides the appearance of ease, it is not fact not easy:
  - It requires a lot of clicking
  - It is difficult to reproduce the steps it took exactly
- If I need to reproduce this for the other 8 states in my data, I will need to redo these steps. **This is time consuming and error prone.**

What's Not So Good? (cont.)

- If I want to make tweaks to this figure, such as changing colors, I have to do everything eight times. **This does not scale**.
- If I want to convert the visualization to a different type of visualization, I have to:
  - Make sure Excel supports it
  - Repeat my steps.

What's Not So Good? (cont.)

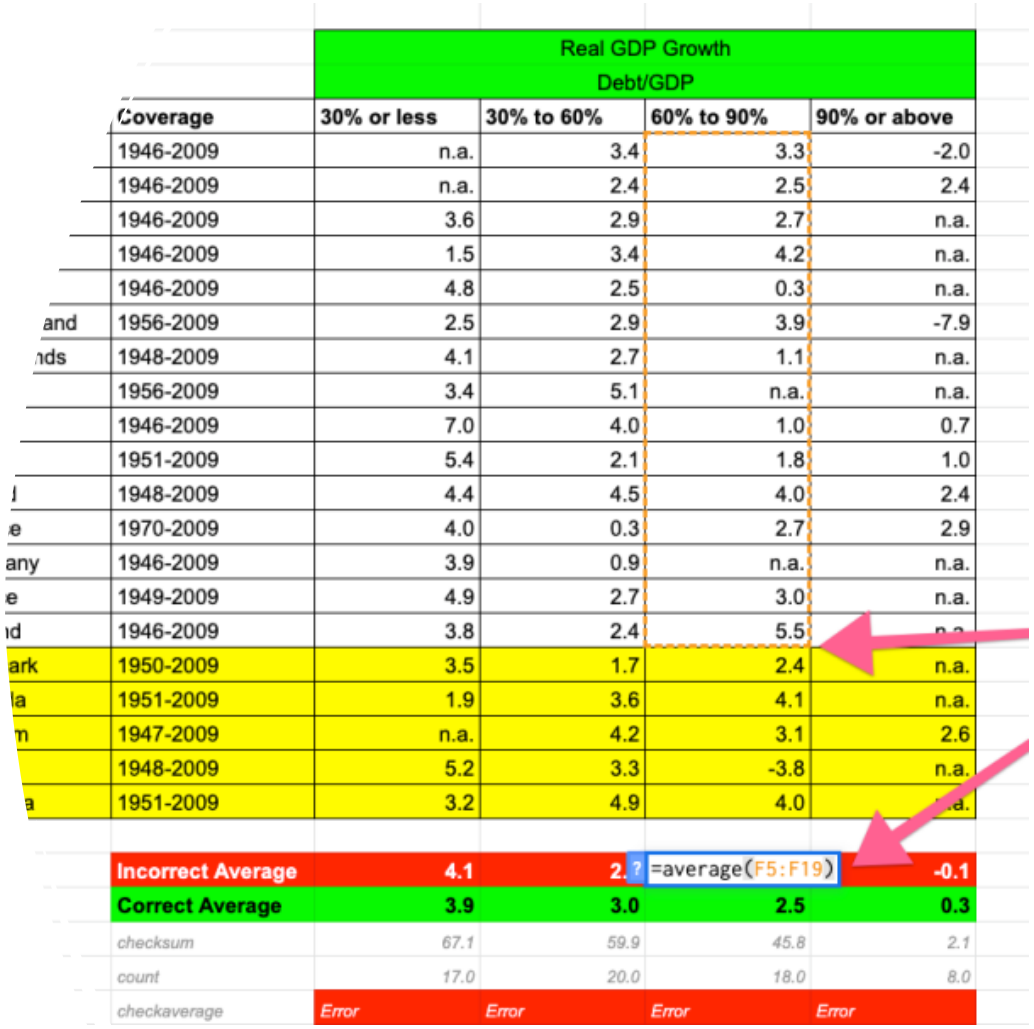
- To filter data, I needed to copy the data to a new file and manually remove rows I did not want. In so doing, there is a non-zero chance I accidentally forget to copy a row or inadvertently delete a record.
- It is difficult to change the format of figures beyond the provided values and doing so requires lots of trial and error.
- I faced data typing (numbers being saved as character strings) issues that were not apparent
- Fixing the typing required brute force: remove the data, put it back
- **There is no documentation of all the steps I took beyond the state of the current workbook.**
  - **Difficult although not impossible to audit my work.**
  - **Difficult although not impossible reproduce my workflow.**

What's Not So Good? (cont.)

- The parable of "they just walked in, doing things"

Reinhart-Rogoff Affair

- A small Excel error in a spreadsheet led two famous economists to conclude that high ratios of National Debt to GDP led national economies to contract
- Finding got cited by numerous influential people pushing austerity measures, such as Paul Ryan.
- Claim at best misleading, at worst erroneous.



	Real GDP Growth Debt/GDP			
Coverage	30% or less	30% to 60%	60% to 90%	90% or above
1946-2009	n.a.	3.4	3.3	-2.0
1946-2009	n.a.	2.4	2.5	2.4
1946-2009	3.6	2.9	2.7	n.a.
1946-2009	1.5	3.4	4.2	n.a.
1946-2009	4.8	2.5	0.3	n.a.
1956-2009	2.5	2.9	3.9	-7.9
1948-2009	4.1	2.7	1.1	n.a.
1956-2009	3.4	5.1	n.a.	n.a.
1946-2009	7.0	4.0	1.0	0.7
1951-2009	5.4	2.1	1.8	1.0
1948-2009	4.4	4.5	4.0	2.4
1970-2009	4.0	0.3	2.7	2.9
1946-2009	3.9	0.9	n.a.	n.a.
1949-2009	4.9	2.7	3.0	n.a.
1946-2009	3.8	2.4	5.5	
1950-2009	3.5	1.7	2.4	n.a.
1951-2009	1.9	3.6	4.1	n.a.
1947-2009	n.a.	4.2	3.1	2.6
1948-2009	5.2	3.3	-3.8	n.a.
1951-2009	3.2	4.9	4.0	
Incorrect Average	4.1	2.	=average(F5:F19)	-0.1
Correct Average	3.9	3.0	2.5	0.3
checksum	67.1	59.9	45.8	2.1
count	17.0	20.0	18.0	8.0
checkaverage	Error	Error	Error	Error

Image source:
https://www.linkedin.com/pulse/episode-40-spreadsheet-error-from-two-harvard-professors-al-chen/

Presentation Structure

1. Present a set of principles for good data visualization
2. Discuss each principle
3. Extend our analysis of birth complicated by hypertension using different tools:
    1. Reproducible workflow using Stata
    2. Reproducible workflow using R
    3. Advanced visualization in R using a ridgeline plot
    4. Dynamic data dashboard using R

Guiding Principles

- Tell The Truth
- Be Transparent & Reproducible

- Serve Your Audience
- Be Pragmatic & Effective
- Be Accessible
- Be Scalable

Trade-offs

- When choosing between multiple possible tools for a job we ask:

  What trade-offs should we make?

- Some things are non-negotiable, such as telling the truth
- Others are negotiable, such as software and the form/style of visualization
- This presentation and the examples will be focused around managing these trade-offs

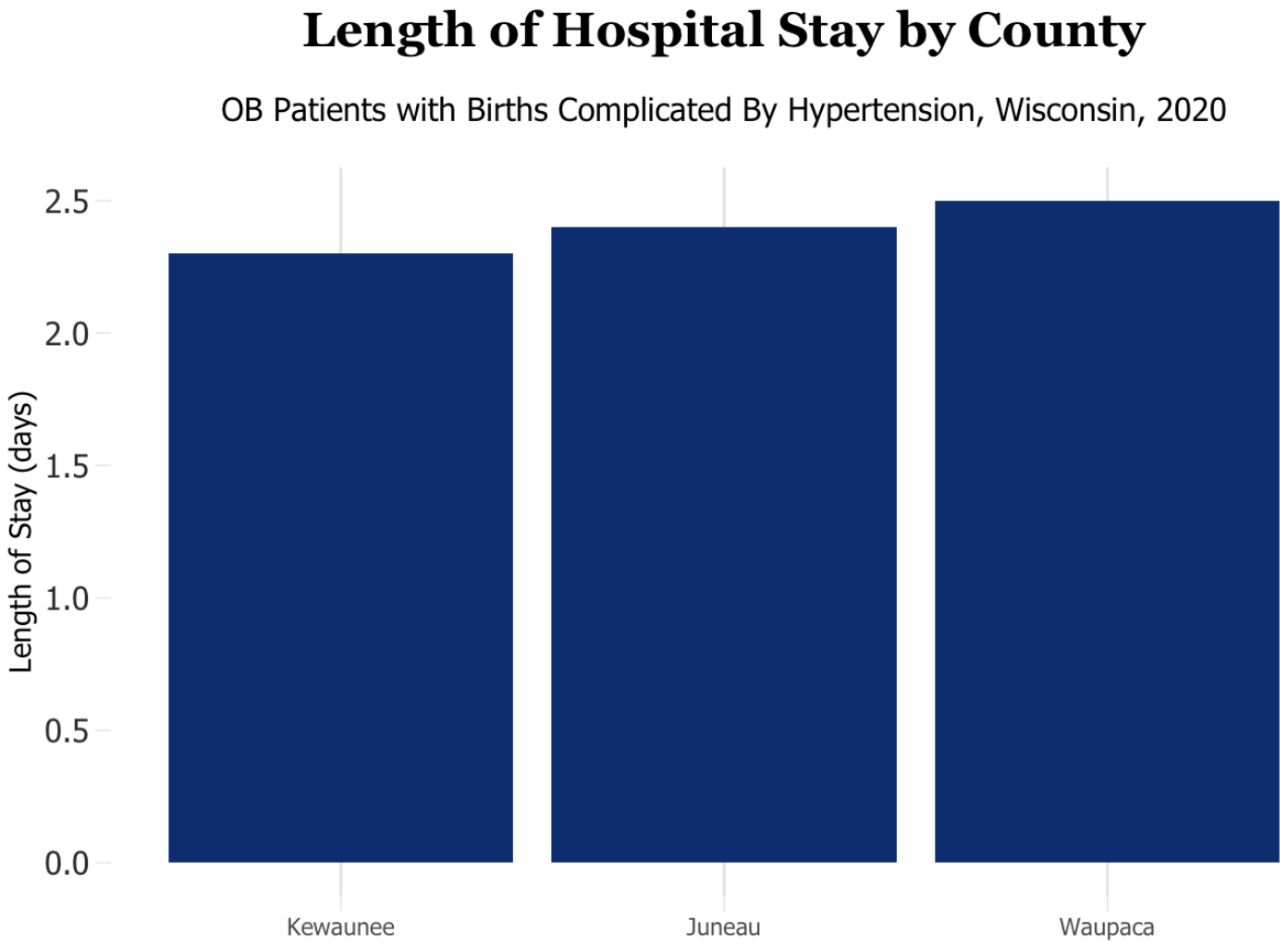
Tell The Truth

- First and foremost, above all else, tell the truth
  - Include **all** relevant data in your visualizations
  - Use sensible scales and axes

Cheating With Scales & Axes

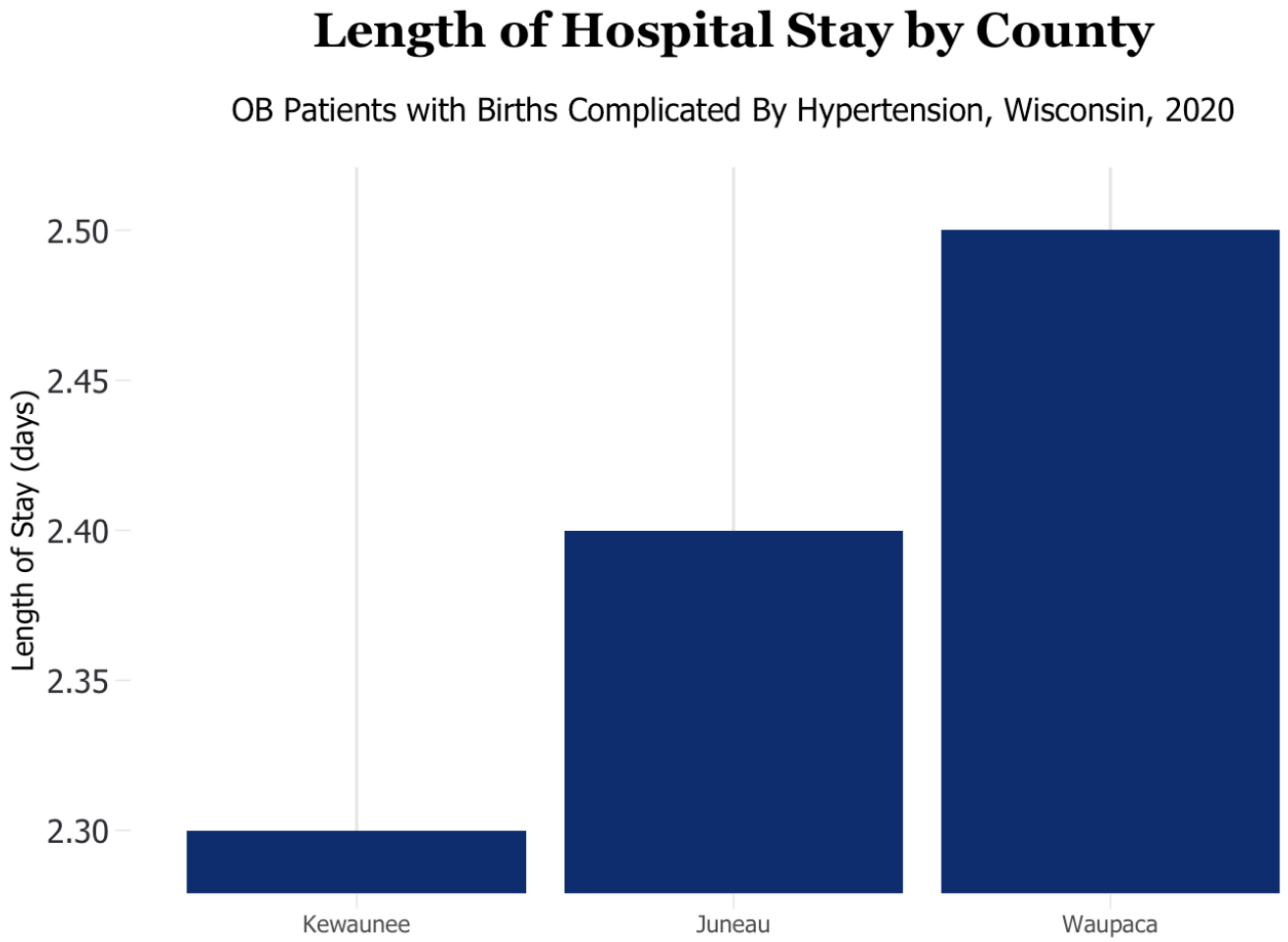
```
comparison_data <- data |>
  filter(
    county %in% c("Kewaunee", "Juneau", "Waupaca")
  )

p <- ggplot(comparison_data, aes(x = fct_reorder(county, length_of_stay), y = length_of_stay)) +
  geom_col() +
  theme_minimal() +
  labs(
    title = "Length of Hospital Stay by County",
    subtitle = "OB Patients with Births Complicated By Hypertension, Wisconsin, 2020",
    x = element_blank(),
    y = "Length of Stay (days)"
  ) +
  theme_jhu_bar()
p
```

Cheating With Scales & Axes

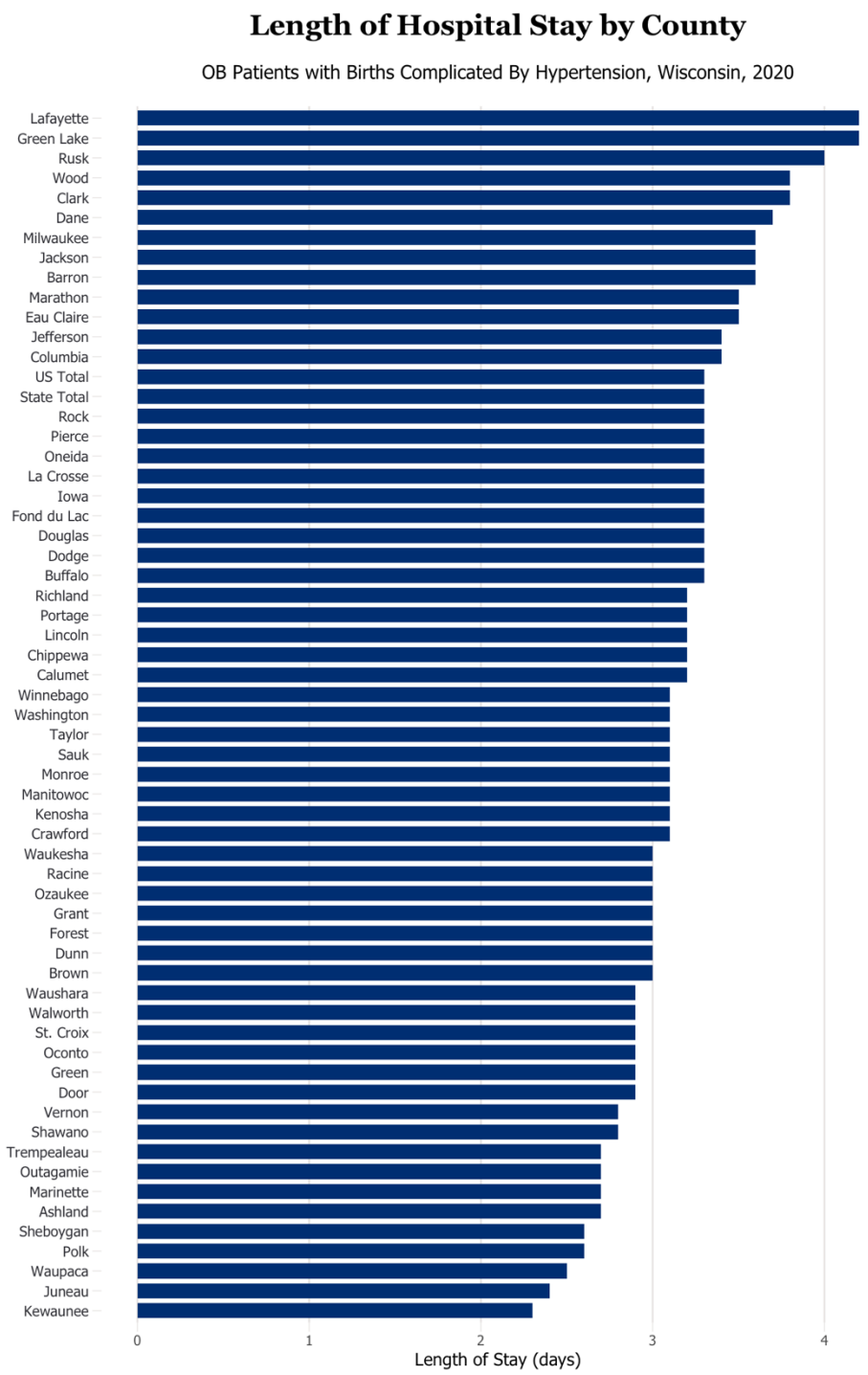


Cheating With Scales & Axes

```
c <- p +
  coord_flip() +
  coord_cartesian(ylim = c(2.29, 2.51))
```



Telling The Truth



Be Transparent & Reproducible

- When using statistical software, carefully save your code with the intention of sharing it.
- Ideally, store your code with versional control software such as Git and publish it in a public repository, such as on GitHub or the Open Science Framework (or both)
- Multiple benefits:
  - **Workflow:** You can extend your code to create new visualizations or in response to reviewer/publisher comments and keep a track record
  - **Scientific validation:** Others can use your code to verify the published results match what your code generates
  - **Future you:** Some day, you may want to reuse your work in a different context; pay it forward to your future self (or others!)

Transparency & Making Mistakes

- **Mistakes are inevitable: we should assume human fallibility**
- Sharing our work is a necessary conditions for the the error-correcting nature of science to work

Be Transparent & Reproducible (cont.)

https://github.com/erikwestlund/mhviz-tools

Erik Westlund formatting

data	telling the truth
.gitignore	interview
README.md	telling the truth
colors.R	telling the truth
mhviz-tools.Rproj	init
tell_the_truth.Rmd	formatting
tell_the_truth.html	telling the truth
theme.R	telling the truth

main mhviz-tools / tell_the_truth.Rmd ↑ Top

Code Blame Raw

```
32  ## Full Scale
33
34  Below, we visualize length of stay by county in its full scale
35
36  ```{r full_scale}
37
38  comparison_data <- data |>
39    filter(
40      county %in% c("Kewaunee", "Juneau", "Waupaca")
41    )
42
43  p <- ggplot(comparison_data, aes(x = fct_reorder(county, length_of_stay), y = length_of
44    geom_col() +
45    theme_minimal() +
46    labs(
47      title = "Length of Hospital Stay by County",
48      subtitle = "OB Patients with Births Complicated By Hypertension, Wisconsin, 2020",
49      x = element_blank(),
50      y = "Length of Stay (days)"
51    ) +
52    theme_jhu_bar()
53  p
54  ```
```

Hot Take

Software Choice & Reproducibility

- Be pragmatic, but….
- Using free and open-source software where possible contributes to an overall environment of reproducibility.
- Why? Anyone, anywhere, with a computer and an internet connection can reproduce a study where the data and code can be shared.

- Examples: R, Python

Serve Your Audience

- When deciding how to generate visualizations, consider your audience.
- Who will view your visualizations?
- For what purpose?
- How much effort do you need to put into polish for your audience?

Serve Your Audience (cont.)

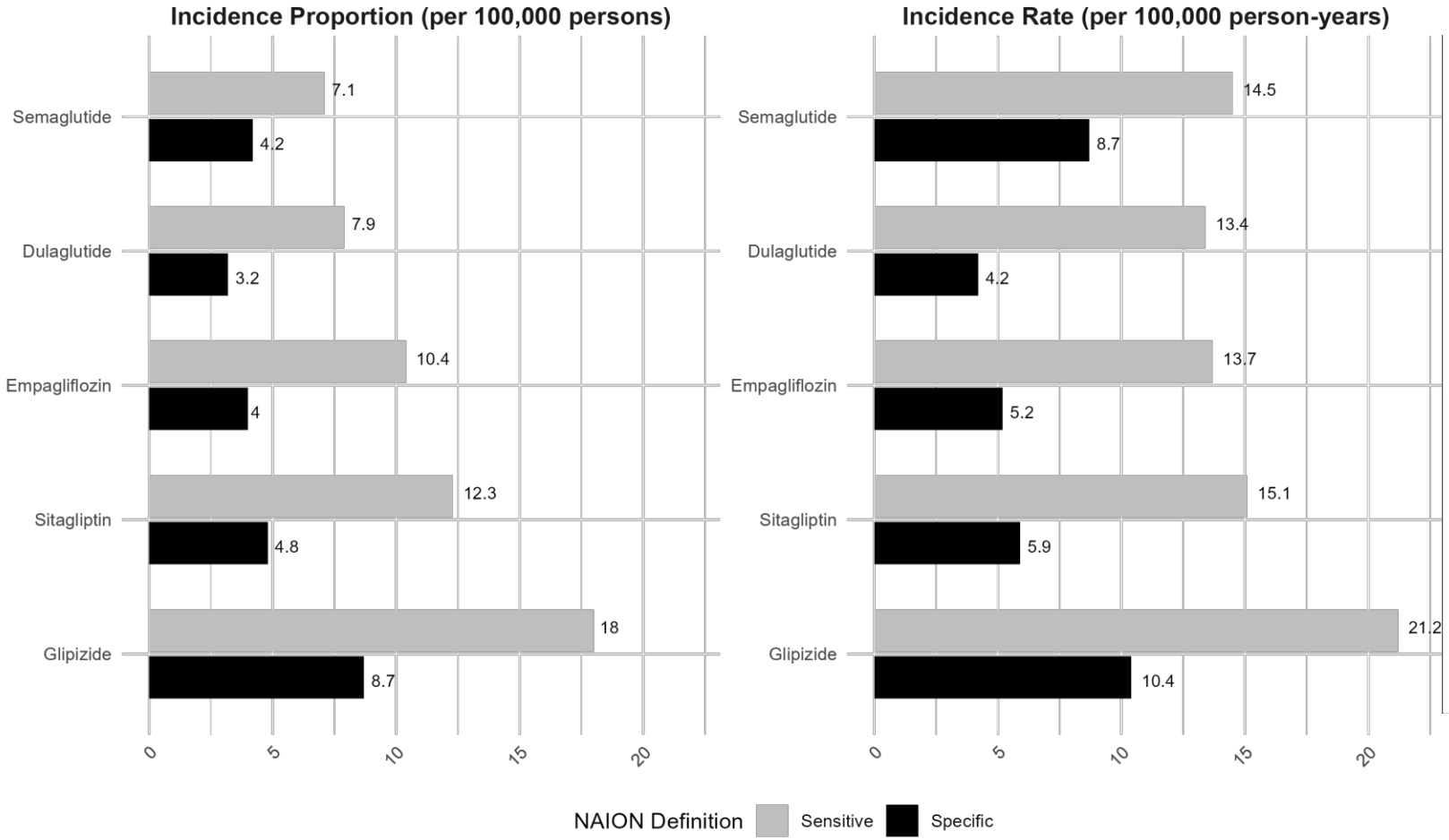
Your visualization stools should differ given the audience.

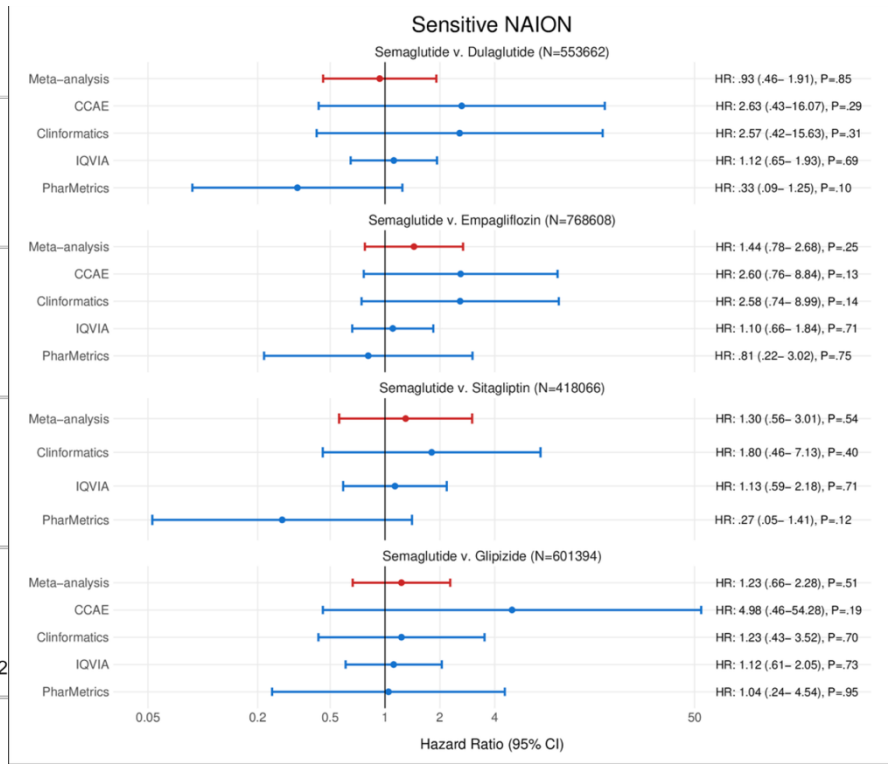
Purpose	Audience	Considerations
Exploratory data analysis	Colleagues	Provisional, but make your work scalable to the next stage of analysis High volume of visuals before filtering; **use statistical software**
Manuscript submission	Reviewers	Consider journal requirements, don't prematurely polish Make sure your workflows accommodate submission requirements (e.g., around data/code sharing)
Manuscript publication	Scientific community	Editors will often polish/recreate your figures; read submission guidelines You may not be doing the final visualization creation
Non-journal reports	Scientific community Public	Make sure your visualization can scale into publication Talk with graphics/publication team; **consider graphics software**

Case Study: Premature Polishing

- Paper: “Semaglutide and nonarteritic anterior ischemic optic neuropathy” by Cai et al.
- A colleague and I created a reproducible workflow to extract results and generate tables and figures for a manuscript submission
- A lot of time went into polishing the fine details of a set of bar and forest plot presenting estimates, ensuring the figures were exported in an accessible, publication ready format

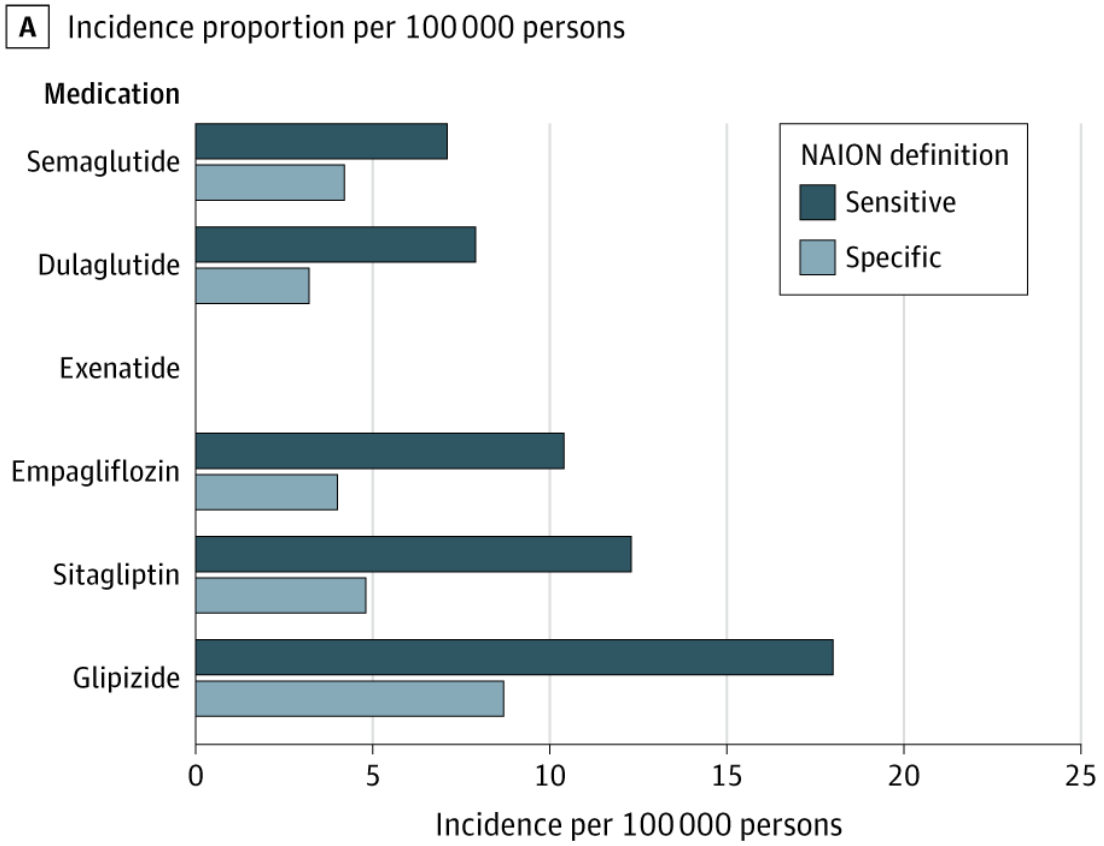
Example Figures

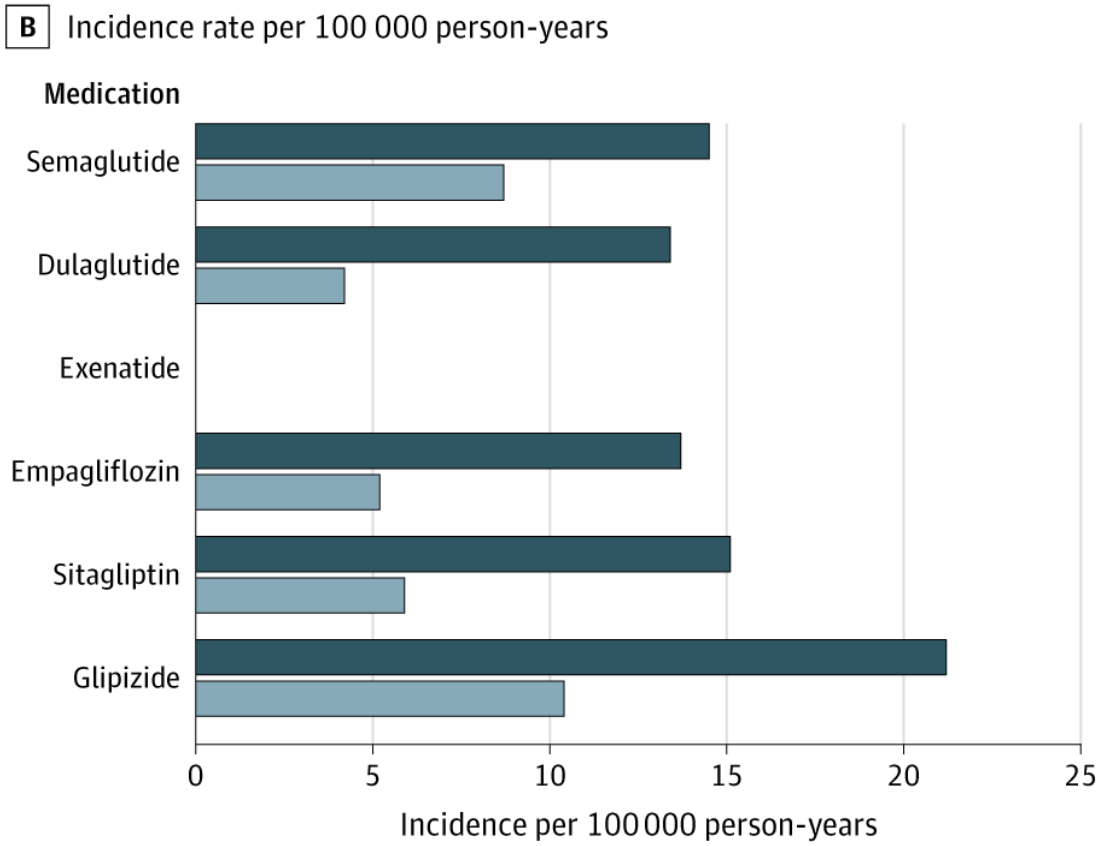




NAION Definition Sensitive Specific

Published Figures

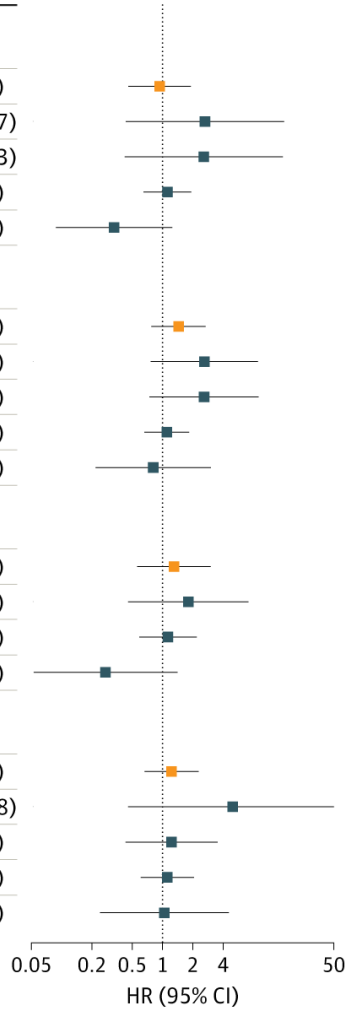




Published Figures (cont.)

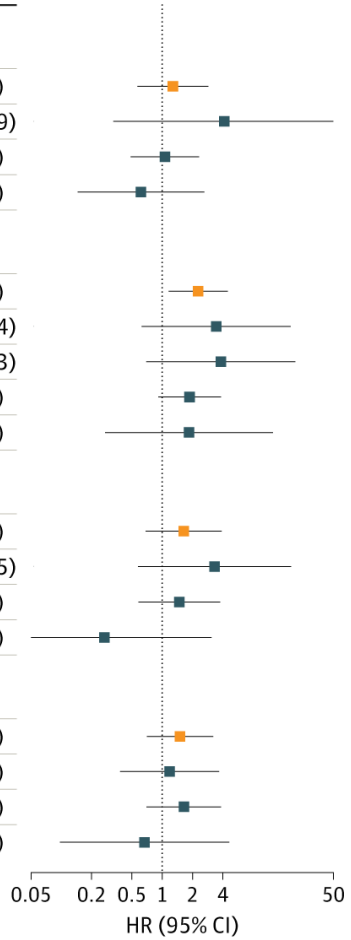
A Sensitive NAION

Study	HR (95% CI)	*P* value
Semaglutide vs dulaglutide (n = 553 662)		
Meta-analysis	0.93 (0.46-1.91)	.85
CCAE	2.63 (0.43-16.07)	.29
Clinformatics	2.57 (0.42-15.63)	.31
IQVIA	1.12 (0.65-1.93)	.69
PharMetrics	0.33 (0.09-1.25)	.10
Semaglutide vs empagliflozin (n = 768 608)		
Meta-analysis	1.44 (0.78-2.68)	.25
CCAE	2.60 (0.76-8.84)	.13
Clinformatics	2.58 (0.74-8.99)	.14
IQVIA	1.10 (0.66-1.84)	.71
PharMetrics	0.81 (0.22-3.02)	75
Semaglutide vs sitagliptin (n = 418 066)		
Meta-analysis	1.30 (0.56-3.01)	.54
Clinformatics	1.80 (0.46-7.13)	.40
IQVIA	1.13 (0.59-2.18)	.71
PharMetrics	0.27 (0.05-1.41)	.12
Semaglutide vs glipizide (n = 601 394)		
Meta-analysis	1.23 (0.66-2.28)	.51
CCAE	4.98 (0.46-54.28)	.19
Clinformatics	1.23 (0.43-3.52)	.70
IQVIA	1.12 (0.61-2.05)	.73
PharMetrics	1.04 (0.24-4.54)	.95



B Specific NAION

Study	HR (95% CI)	*P* value
Semaglutide vs dulaglutide (n = 553 982)		
Meta-analysis	1.28 (0.57-2.88)	.55
CCAE	4.13 (0.33-51.99)	.27
IQVIA	1.06 (0.49-2.33)	.88
PharMetrics	0.62 (0.14-2.61)	.51
Semaglutide vs empagliflozin (n = 769 136)		
Meta-analysis	2.27 (1.16-4.46)	.02
CCAE	3.43 (0.63-18.84)	.16
Clinformatics	3.82 (0.70-20.93)	.12
IQVIA	1.87 (0.92-3.83)	.09
PharMetrics	1.85 (.27-12.58)	.53
Semaglutide vs sitagliptin (n = 418 336)		
Meta-analysis	1.64 (0.69-3.90)	.27
Clinformatics	3.31 (0.58-19.05)	.18
IQVIA	1.48 (0.58-3.75)	.41
PharMetrics	0.27 (0.02-3.08)	.29
Semaglutide vs glipizide (n = 575 356)		
Meta-analysis	1.50 (0.70-3.21)	.29
Clinformatics	1.18 (0.38-3.67)	.77
IQVIA	1.64 (0.70-3.84)	.26
PharMetrics	0.67 (0.10-4.63)	.68



So What?

- These figures were recreated using professional graphics tools for publication.
- With a better understanding of the journal's publication process, many hours of image polishing could have been avoided.
- https://jamanetwork.com/journals/jamaophthalmology/pages/instructions-for-authors#SecFigureFileRequirementsforPublication

Be Pragmatic & Effective

- **As long as you tell the truth and are transparent, otherwise be pragmatic**
- You're a data novice, the data is simple, and you only need a simple visual? Go ahead and use Excel.
- Your team uses Stata and you can comfortably work in it and achieve all your goals? Use Stata.
- Your team uses Stata, but you need to produce a visual that R can easily achieve? Use R for that visualization and document it.
- You want to create a dynamic data visualization dashboard and you're a Stata user? Time to learn Shiny for R/Python!

Hot Take

When Open Source Isn't Worth It

- I advocate for using open source software wherever possible
- However, time limitations are a reality for all of us.

- Suppose you're a seasoned ArcGIS user and you need to create a map
- Go ahead and use the tooling you know and document your work
- It is not realistic or reasonable to expect people to spend dozens of hours to learn new software

- Much of our most important scientific advances requires money and resources not available to us; let's not be too pure.

Be Accessible

- When creating visualizations, especially for publication to the public, be aware of the needs of people with visual impairments
- Numerous tools exist to check the contrast of colors, such as https://webaim.org/resources/contrastchecker/
- In R, the **RColorBrewer** provides color schemes friendly to those with colorblindness
- Patterns can be used instead of colors to distinguish categories; this is especially useful for documents printed in black and white

Accessibility Is Effective

- In many cases, improving color contrast or using patterns to make groupings in visualizations clearer will help **all** members of your audience, not just the visually impaired.

- If a visualization has too many colors or patterns, it is likely confusing for **all,** not just the visually impaired.

Be Scalable

- There is sometimes a trade-off between short-term efficiency and scalability, especially in the long term
- For example, Microsoft Excel may be a fine tool for a single visualization and quicker than setting up a statistical environment.
- However, if in the future your project will require you to make multiple visualizations, from multiple data files, this approach is **pennywise and poundfoolish**: the time you save now may be lost many times over when you need to repeat many time-consuming manual tasks repeatedly.

Scalability Is Effective

- Code-driven, reproducible workflows are, in general,
  - More scalable
  - More reliable: to the extent that the generation of visualizations is automated, the likelihood of human error is reduced

Demonstrations

- Visualization of cost of hospital stay for OB patients with births complicated by hypertension (HCUP data) – Wisconsin only
  - Stata example
  - R example
- Extend visualization to all states we have HCUP data on
  - R example
- Create dynamic dashboard
  - R example
- All code: https://github.com/erikwestlund/mhviz-tools

Example #1: Stata Bar Plot

- The GitHub has an example of loading a data set, cleaning it, and creating the same visualization we did above in Excel
- It records the code in a log
- It outputs the image in `output/`
- The do file is located in the Github at `1_wisconsin_hypertension_complicated_birth_cost_of_stay.do`

- This workflow is shareable and reproducible.

Stata Do-file & Log

1_wisconsin_hypertension_complicated_birth_cost_of_stay.do

Open Save Print Find Show 100% Zoom Do

1_wisconsin_hypertension_complicated_birth_cost_of_stay.do

```
capture log close
log using "output/1_hcup_wisconsin_analysis.log", text replace

cd "~/code/mhviz-tools/"

import excel "data/hcup/Wisconsin/HCUPnet_Community_DX1CCSR_County_WI_hyp_2020.xlsx", ///
    sheet("Table Data") firstrow clear

* Rename columns
rename County county
rename FIPScode fips_code
rename PatientCharacteristic patient_characteristic
rename NumberofDischarges number_of_discharges
rename AverageLengthofStayindays length_of_stay
rename RateofDischargesper100000P rate_of_discharges_per_100000p
rename AgeSexAdjustedRateofDischar age_sex_adj_rate_of_discharge
rename AggregateHospitalCostsin aggregate_hospital_costs
rename AverageHospitalCostsperStay average_hospital_costs_per_stay

* Drop rows where length_of_stay is "*"
drop if length_of_stay == "*"

keep county average_hospital_costs_per_stay

* convert to numeric
destring average_hospital_costs_per_stay, replace

* generate bar chart

graph hbar (asis) average_hospital_costs_per_stay, ///
    over(county, sort(1) descending label(angle(0))) ///
    bar(1, color(navy)) ///
    ysize(10) xsize(6) /// Make the graph taller
    scale(0.8) /// Reduce overall text size
    title("Average Cost (USD) of A Hospital Visit When Giving Birth:" ///
          "OB Patients with Births Complicated By Hypertension" ///
          "(Wisconsin, 2020)", ///
    size(medium) color(black)) ///
    scheme(s1color)

graph export "output/1_hcup_wisconsin_costs.png", replace

log close
```

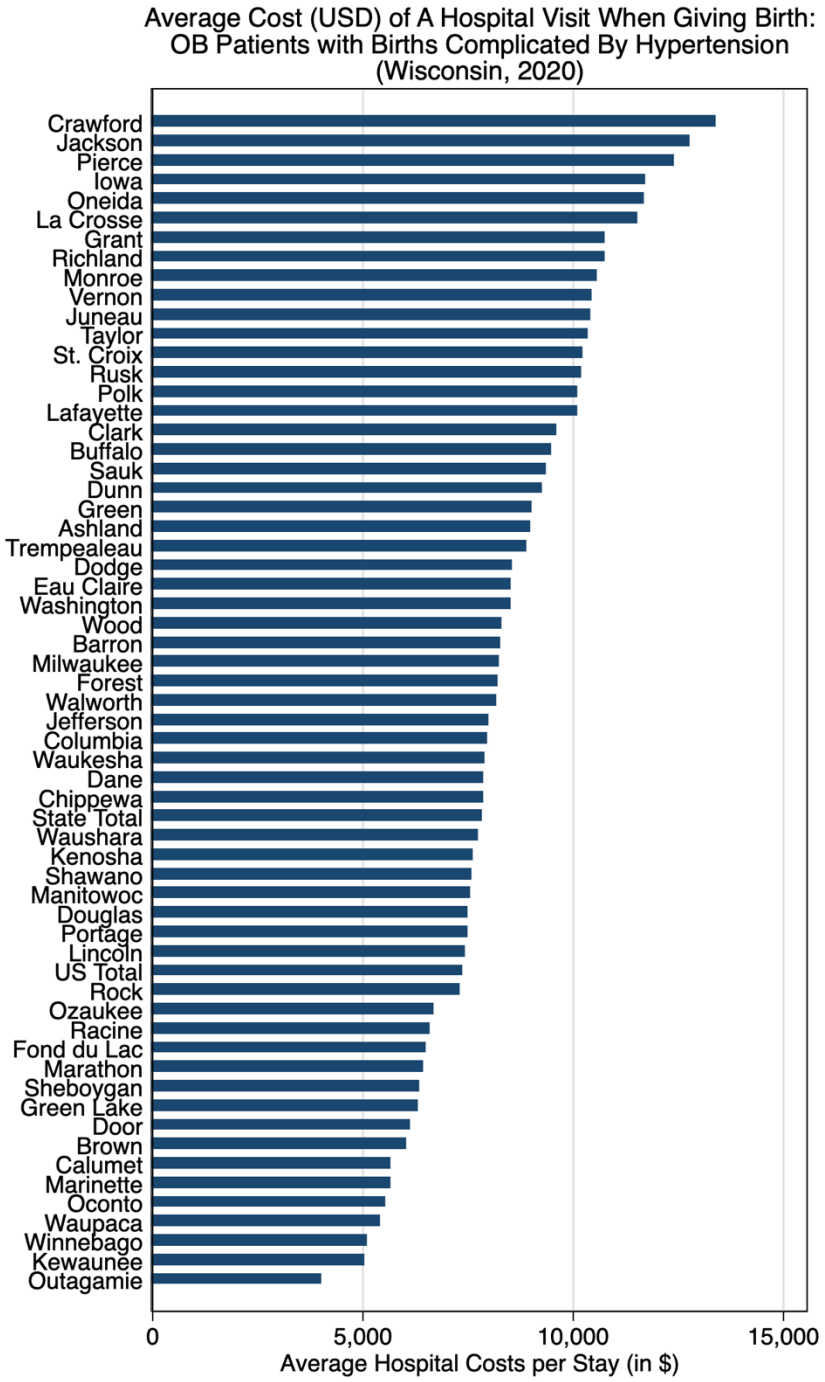
Save

65:16

1_hcup_wisconsin_anal... Reveal Now Clear Reload Share Search

```
-------------------------------------------------------------------------------
      name:  <unnamed>
       log:  /Users/erikwestlund/code/mhviz-tools/output/1_hcup_wisconsin_analysis.log
  log type:  text
 opened on:  19 Mar 2025, 17:24:33

. cd "~/code/mhviz-tools/"
/Users/erikwestlund/code/mhviz-tools

. import excel "data/hcup/Wisconsin/HCUPnet_Community_DX1CCSR_County_WI_hyp_2020.xlsx", ///
>     sheet("Table Data") firstrow clear

. * Rename columns
. rename County county

. rename FIPScode fips_code

. rename PatientCharacteristic patient_characteristic

. rename NumberofDischarges number_of_discharges

. rename AverageLengthofStayindays length_of_stay

. rename RateofDischargesper100000P rate_of_discharges_per_100000p

. rename AgeSexAdjustedRateofDischar age_sex_adj_rate_of_discharge

. rename AggregateHospitalCostsin aggregate_hospital_costs

. rename AverageHospitalCostsperStay average_hospital_costs_per_stay

. * Drop rows where length_of_stay is "*"
. drop if length_of_stay == "*"
(13 observations deleted)

. keep county average_hospital_costs_per_stay

. * convert to numeric
. destring average_hospital_costs_per_stay, replace
average_hospital_costs_per_stay: all characters numeric; replaced as int

. * generate bar chart
.
. graph hbar (asis) average_hospital_costs_per_stay, ///
>     over(county, sort(1) descending label(angle(0))) ///
>     bar(1, color(navy)) ///
>     ysize(10) xsize(6) /// Make the graph taller
>     scale(0.8) /// Reduce overall text size
>     title("Average Cost (USD) of A Hospital Visit When Giving Birth:" ///
>           "OB Patients with Births Complicated By Hypertension" ///
>                 "(Wisconsin, 2020)", ///
>     size(medium) color(black)) ///
>         scheme(s1color)

. graph export "output/1_hcup_wisconsin_costs.png", replace
(file output/1_hcup_wisconsin_costs.png written in PNG format)

. log close
      name:  <unnamed>
       log:  /Users/erikwestlund/code/mhviz-tools/output/1_hcup_wisconsin_analysis.log
  log type:  text
 closed on:  19 Mar 2025, 17:24:34
-------------------------------------------------------------------------------
```

Stata Graphics

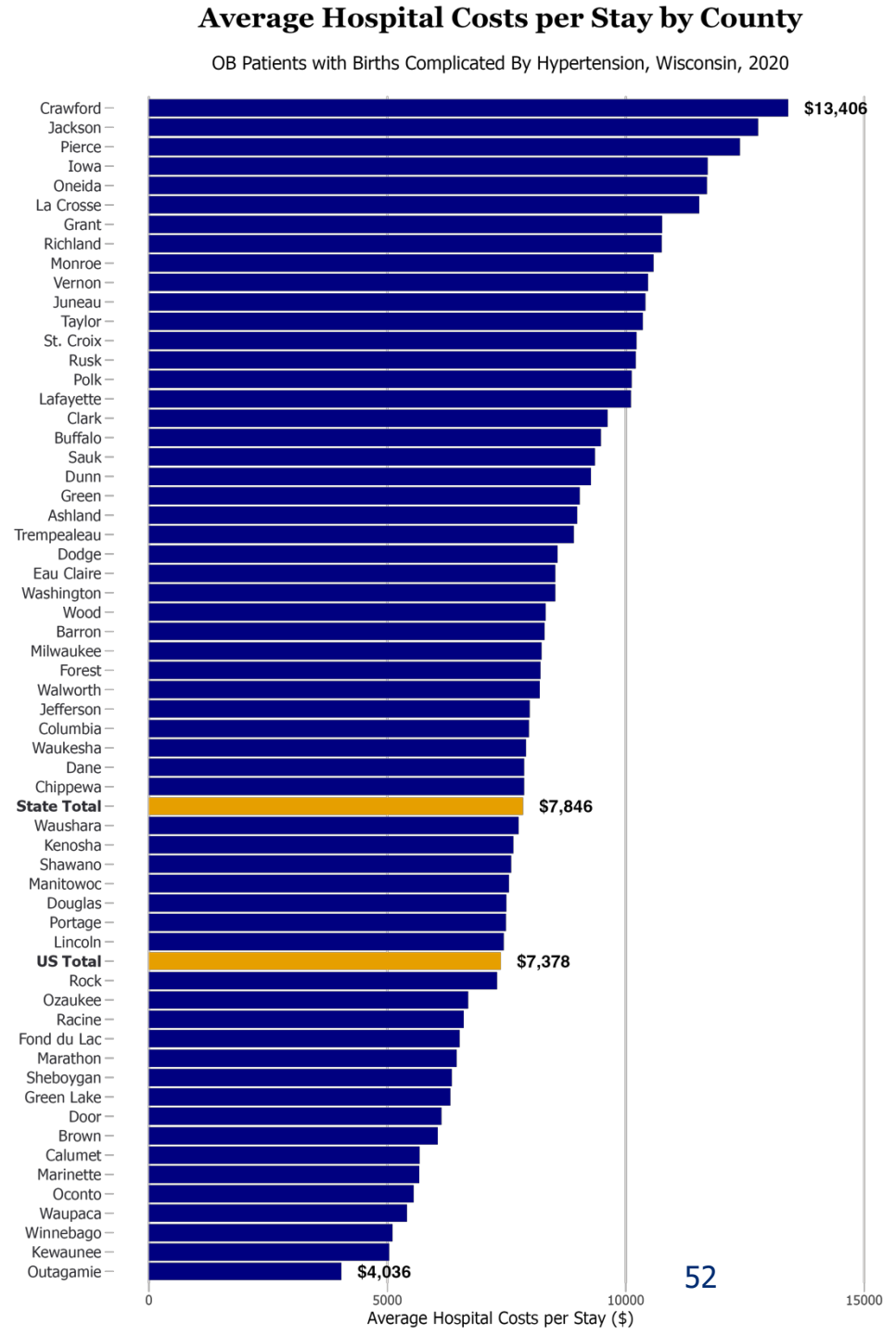


Example #2: R Bar Plot

- The GitHub has an example of an RMarkdown notebook
- Instead of a log file, the notebook allows you to run code, but also add commentary
- It can show visualizations inline, but also save them to disk.
- Like with Stata, this workflow is reproducible.\
- The Rmarkdown file in the GitHub repository is named `2_wisconsin_hypertension_complicated_birth_cost_of_stay.Rmd`

R Example

- The GitHub has an example of an RMarkdown notebook
- Instead of a log file, the notebook allows you to run code, but also add commentary
- It can show visualizations inline, but also save them to disk.
- Like with Stata, this workflow is reproducible.



Trade-off Check?

Stata	R
✅ Reproducible	✅ Reproducible
❌ Closed-source	✅ Open source
✅ Easy installation on all systems	❌ More error prone
✅ Professional support	❌ No official support channel
❌ Struggles with complex visualizations	✅ Handle complex visualizations
🔶 Jupyter notebook support in Stata 17	✅ Notebook support (Rmarkdown, Quarto, Jupyter)

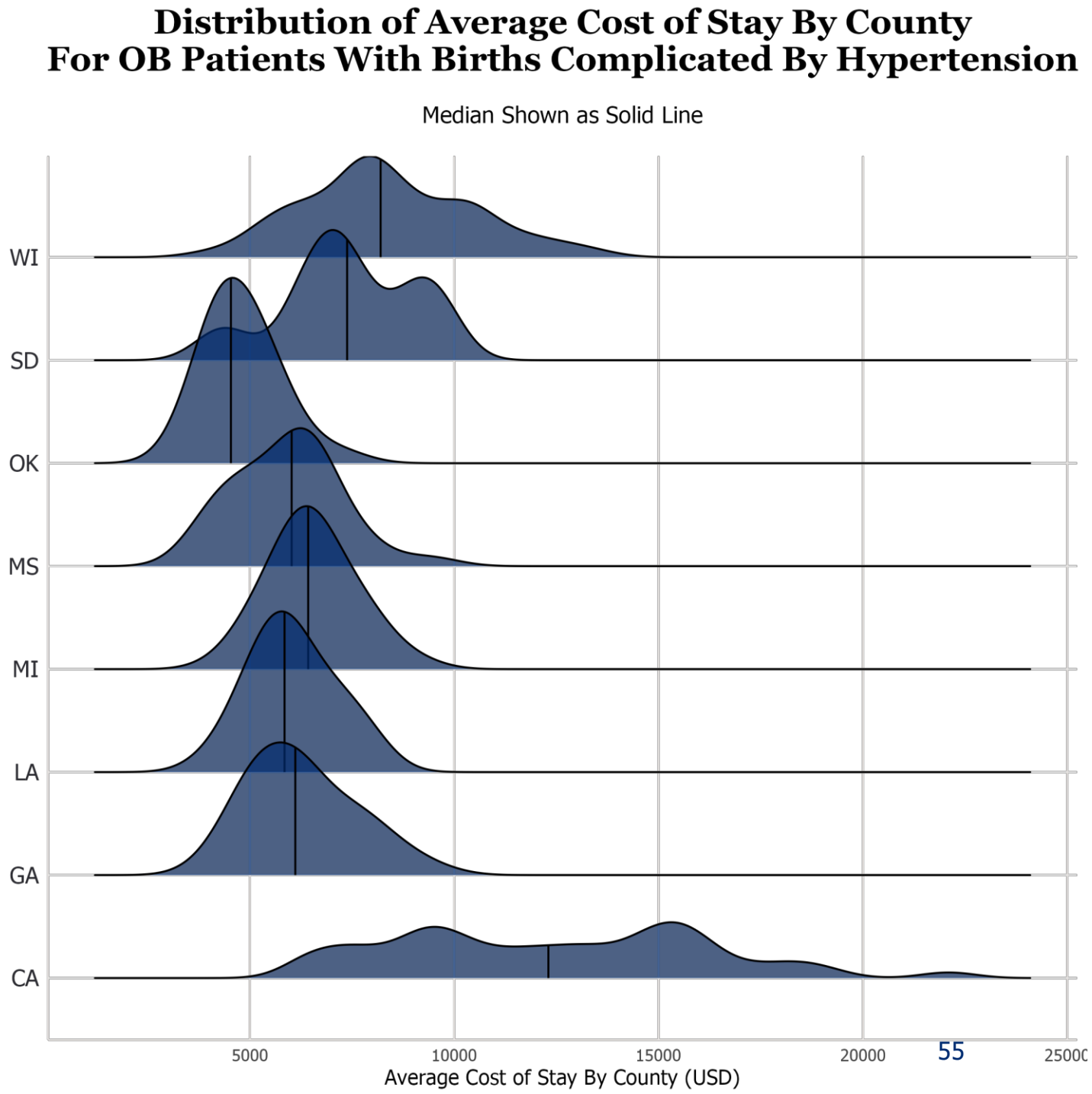
- **Conclusion:** Use what works for you

Example #3: R Ridgeline Plot

- Languages like R and Python begin to shine when working with multiple data files and doing more complex visualizations.
- Next, we'll tackle a ridgeline plot, which allow us to stack distributions on each other.
- Challenges:
  - Multiple source data files
  - Complex visualization
- File in the GitHub repository is named 3_nine_states_hypertension_complicated_birth_cost_of_stay.Rmd

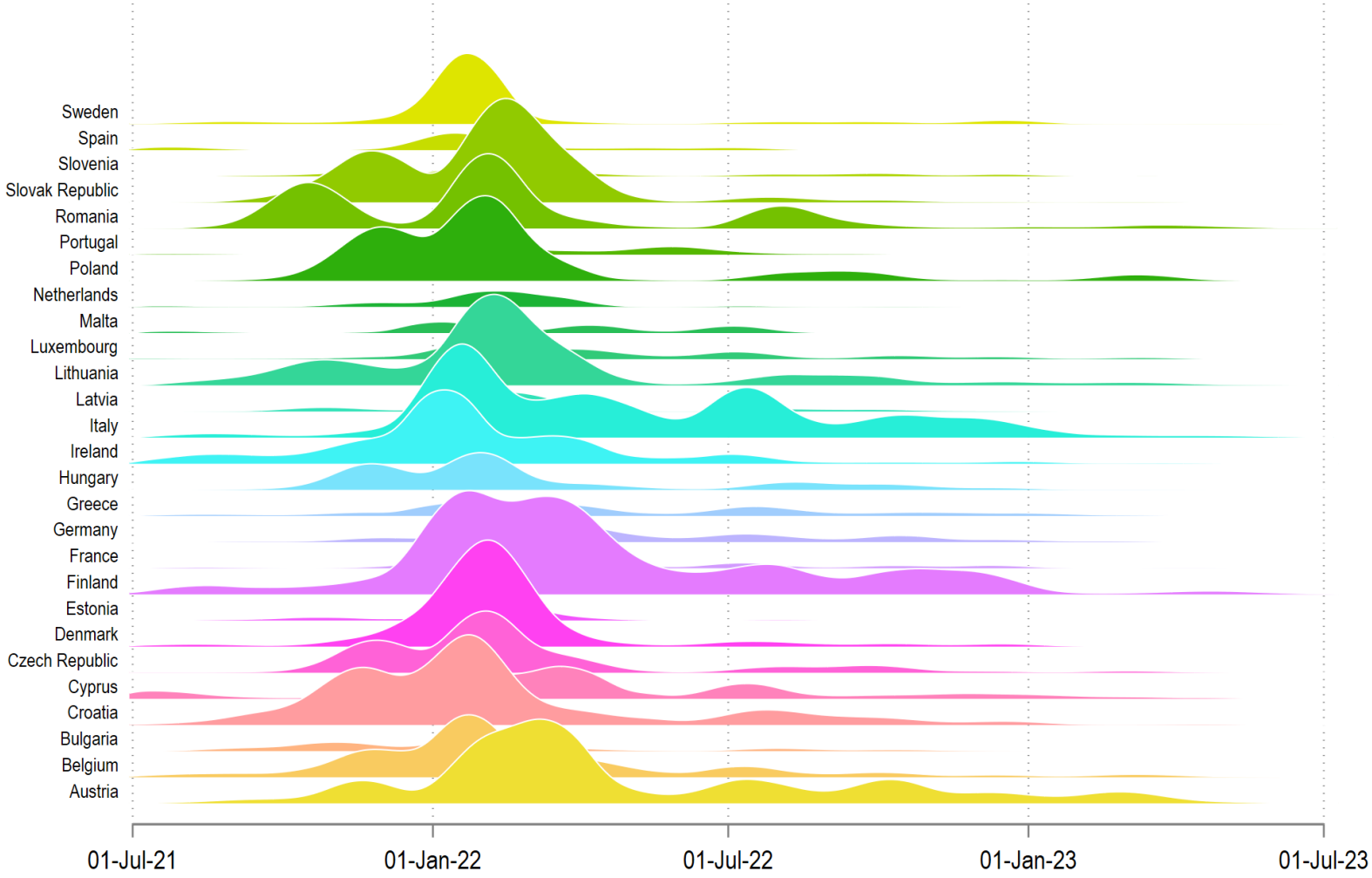
Ridgeline Plot

- R gives us the ability leverage open-source packages, such as `ggridges`, to generate complex figures without



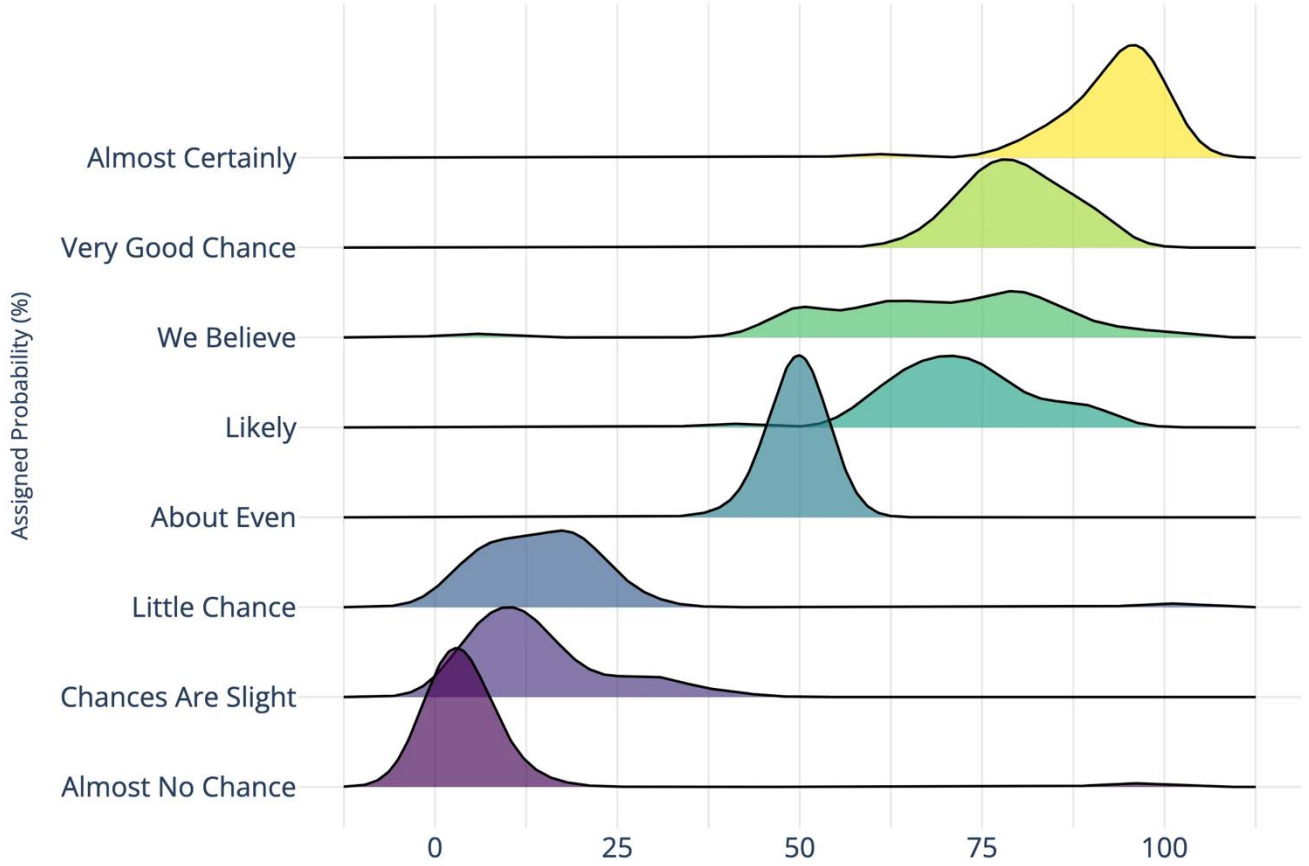
Can I make a Ridgeline plot with Stata?

- Stata has a powerful graphics engine and is extensible.
- There is a ridgeline add-on: https://github.com/asjadnaqvi/stata-ridgeline



Can I make a Ridgeline plot with Python?

- Python also has a powerful set of graphics libraries?
- For example, there is a ridgeline Python package:
- https://github.com/tpvasconcelos/ridgeplot



Can I make a Ridgeline plot with SAS?

- Yes
- See here:

https://blogs.sas.com/content/graphicallyspeaking/2020/02/24/ridgeline-plots-using-sgplot/

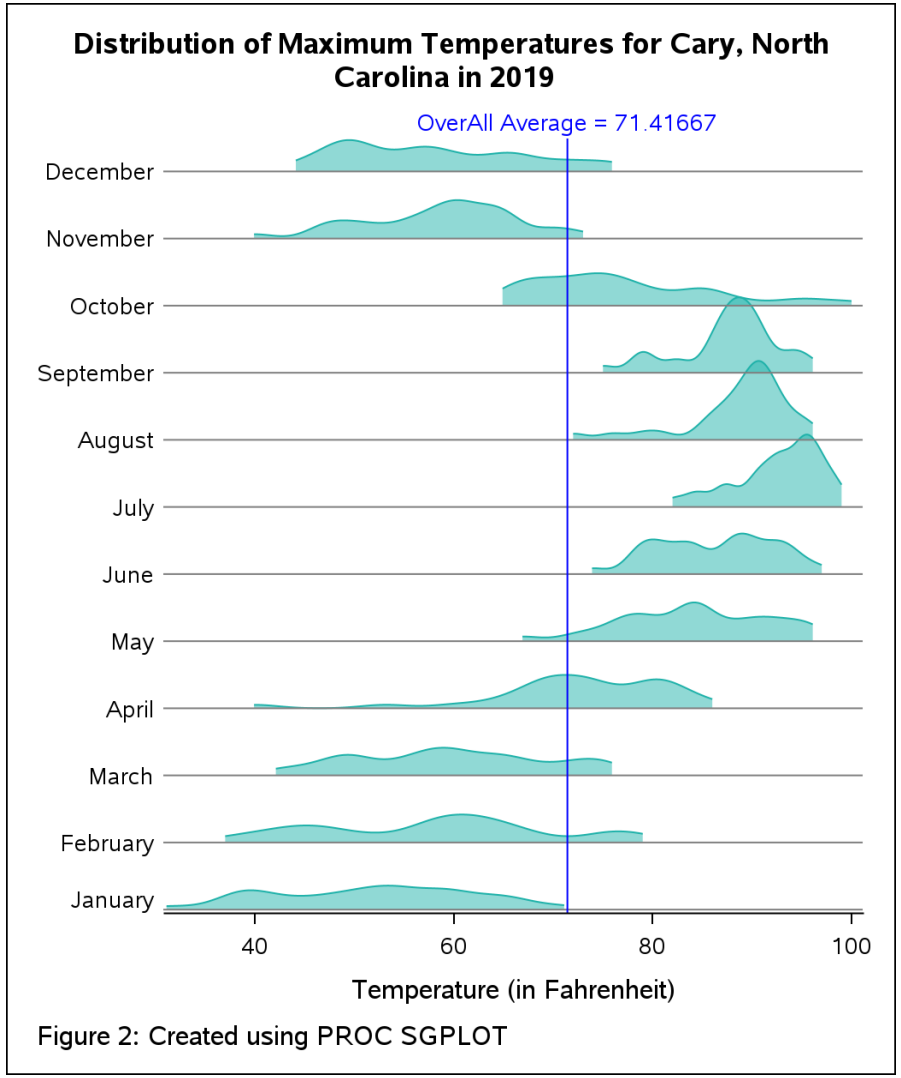


Figure 2: Created using PROC SGPLOT

Complex Visualizations

- R, Stata, SAS, and Python all have powerful, extensible ways to build complex graphics (such as ridgeline plots)
- In general, there are probably no figures you cannot in principle create in one package and not in another.
- R generally has the most robust visualization ecosystem. For example, consider our ridgeline example:
  - R Package has 418 “stars” on GitHub
  - Python package has 219 “stars”
  - Stata package has 37 “stars”

Pragmatic Strategy: Pivot

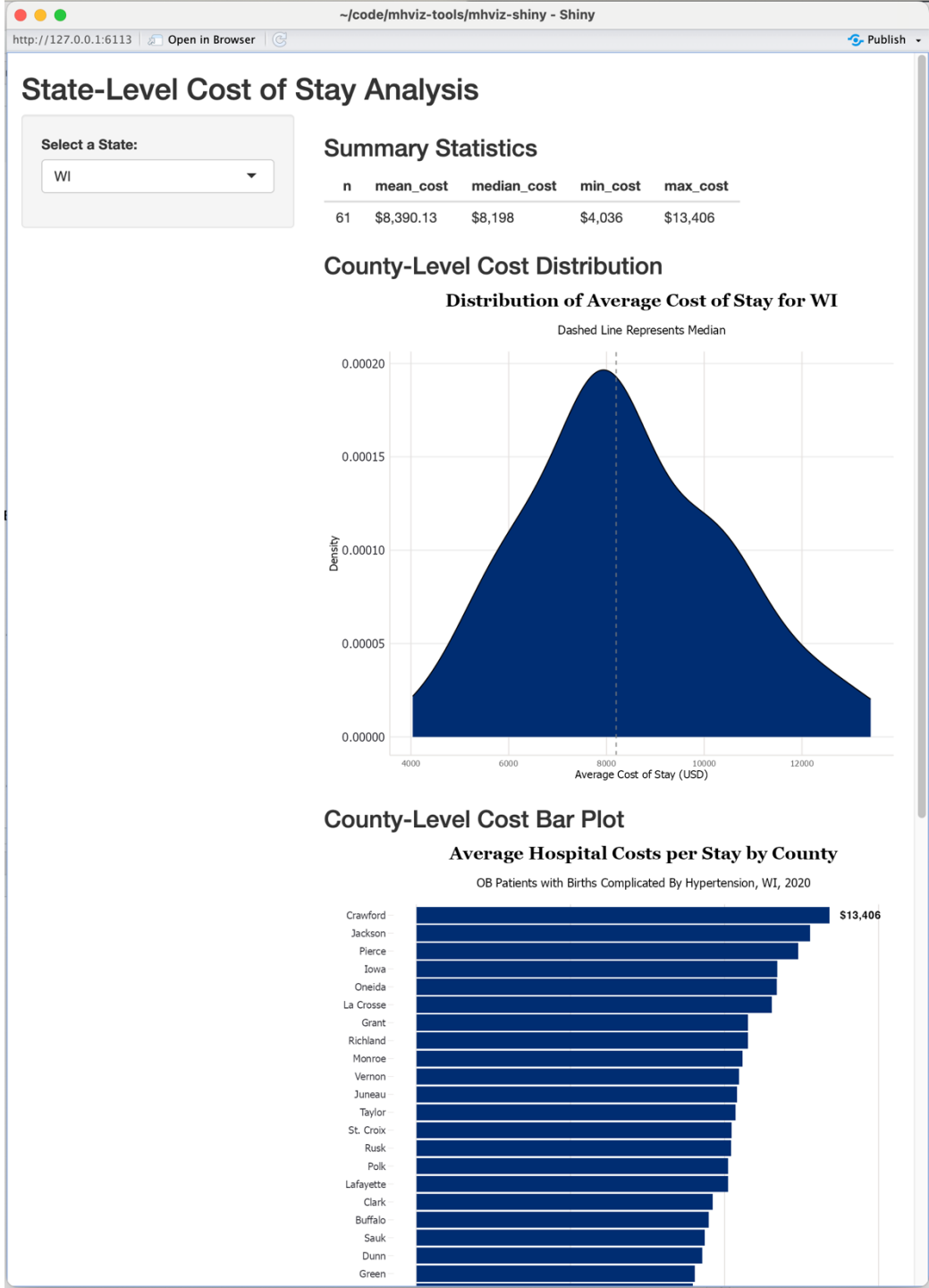
- An effective strategy that works well in projects with well-designed, reproducible workflows where each step of the data processing, analysis, and visualization is cleanly separated and documented is to reach for the tool that meets your needs.
- Necessary condition: good workflows for generating datafiles that can then be read in by other programs.
- Example workflow:
  - Clean data in R
  - Run models in Stata
  - Generate visualizations in R
- Just be transparent and keep your code organized!

Example 4: Dynamic Dashboards in R

- R and Python both have implementations of a library called “Shiny” that allows you to generate dynamic dashboards
- Users can select specific options, the app will automatically process the data, create the tables/visualizations requested, and dynamically render them.
- These are actually web applications that can be run locally on your computer or deployed to a public-facing web site.
- Let’s create a very simple Shiny app that lets users dynamically create visualizations of OB Patients with Births Complicated By Hypertension in 2020.

Shiny App

- The data the Shiny app depends on is created by the file `4_generate_data_for_shiny_app.R`
- Code for the Shiny app is in the GitHub repository at mhviz-shiny/app.R
- The simplest of these app simply comprise a function to describe the user interface (UI) and the server.
- The server returns dynamically generated data objects.



Conclusions

- Good visualizations depends on good (data) science.
- If your data workflows are solid, good visualization comes easier.
- Making trade-offs is inevitable.
- Be pragmatic and use the tools that work best for you, so long as you:
  - Tell the truth
  - Are transparent & reproducible
- Using well-designed scripts and notebooks will make your life a lot easier.
- Nearly every principle we discussed here applies to other crucial aspects of the scientific process, including scientific modeling.

Contact Me

Erik Westlund

ewestlund@jhu.edu

JHBC: https://publichealth.jhu.edu/johns-hopkins-biostatistics-center

Thank you!!