From #EconTwitter to the White House

Real-Time Economic Data with R

Mike Konczal, Economic Security Project

August 09, 2025

Presented at the useR! 2025 Conference, Duke University

Today We'll Discuss

Over the next 15 minutes I will discuss:

- How to use R to analyze economic data right as it comes out.
- Why you should use R to do that.
- Try to convince analysts to not just use Excel, and economists to not just use Stata.
- Five examples where R really shines.

Data generally comes out promptly at 8:30 a.m., sometimes 10:00 a.m.. The 'first draft' of what has just happened is written quickly, prompt analysis is key.

My Background

Why me?

- Have provided commentary and real-time analysis on data releases for over a decade.
- Served as Special Assistant to the President and Chief Economist, National Economic Council (NEC), handling macroeconomic data releases, in 2024.
- Long-time R user and proselytizer.

A Quick Note on Political Developments

"This rationale for firing Dr. McEntarfer is without merit and undermines the credibility of federal economic statistics that are a cornerstone of intelligent economic decision-making by businesses, families, and policymakers. U.S. official statistics are the gold standard globally. When leaders of other nations have politicized economic data, it has destroyed public trust in all official statistics and in government science."

"*Statement on Commissioner McEntarfer's Removal.*" William Beach, former BLS Commissioner (2019-2023), Mercatus Center (2016-2019) and Heritage Foundation (1998-2013)

How to Get Data - White House

White House process worked on two tracks:

- Council of Economic Advisers get that early data first, as a print copy, and prepare analysis (usually in Stata).
- A handful of senior people see the data the afternoon before (POTUS, Chief of Staff, NEC Director).
- The rest of the teams prepare materials (talking points, statements) not knowing what is in it, then scramble right after.
- R is available to White House staff, but was only base R for the longest time (finally got tidyverse late 2024).

How to Get Data - Everyone Else

Can get from the relevant government website:

- API calls (library blsR).
- Flatfile downloads (own library).

Can get from other sites:

- Download from a private service (HAVER, Bloomberg).
- Download from FRED, FRED's API or certain libraries (but starting at 8:51am).

Many, from analysts to reporters, build up extensive workflows for covering data releases. So why R?

Example 1: The Grammar Deployed

BLS flat file data is tidy, and we want to examine different elements of the same items (industries, locations, etc.). This makes it perfect for the grammar of graphics.

```
ces <- getMacroTools("ces", "konczal@gmail.com")
```

```
ces %>%
  select(series_id, date, value, data_type_text, industry_name) %>%
  head(5)
```

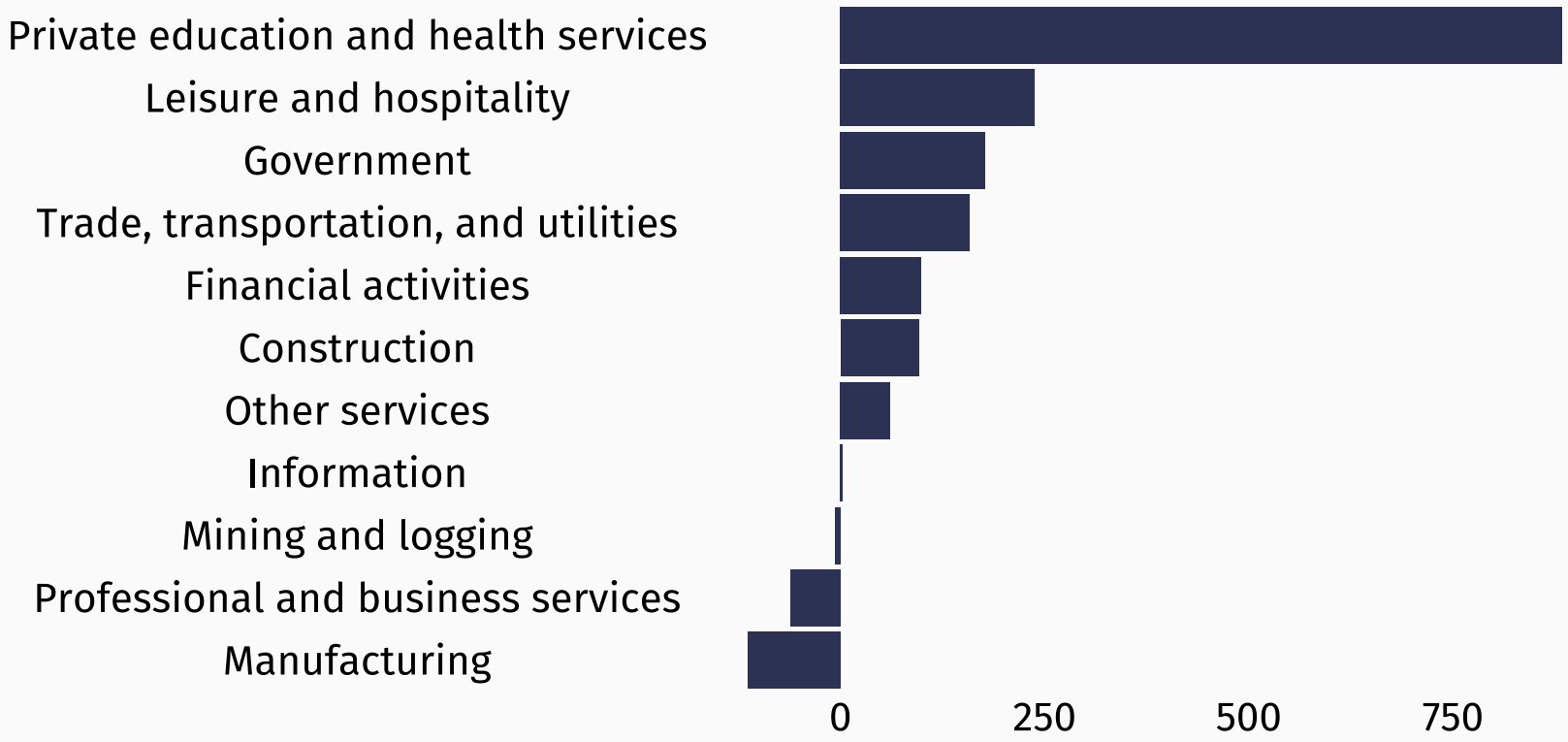
```
## # A tibble: 5 x 5
##   series_id     date        value data_type_text           industry_name
##   <chr>         <date>      <dbl> <chr>                    <chr>
## 1 CES0000000001 2023-01-01 154780 ALL EMPLOYEES, THOUSANDS Total nonfarm
## 2 CES0000000001 2023-02-01 155086 ALL EMPLOYEES, THOUSANDS Total nonfarm
## 3 CES0000000001 2023-03-01 155171 ALL EMPLOYEES, THOUSANDS Total nonfarm
## 4 CES0000000001 2023-04-01 155387 ALL EMPLOYEES, THOUSANDS Total nonfarm
## 5 CES0000000001 2023-05-01 155614 ALL EMPLOYEES, THOUSANDS Total nonfarm
```

Example 1: Tidy CES Jobs

Let's say you want to look at the number of jobs per industry over the past year.

```
ces <- getBLSFiles("ces", "konczal@gmail.com")

ces %>%
  filter(seasonal == "S",
         display_level == 2,
         # Watch this line:
         data_type_text == "ALL EMPLOYEES, THOUSANDS") %>%
  group_by(industry_name) %>%
  reframe(
    change = value[date == max(date)] - value[date == max(date) %m-% months(12)]
  ) %>%
  mutate(industry_name = fct_reorder(industry_name, change)) %>%
  ggplot(aes(industry_name, change)) +
  geom_col() +
  coord_flip()
```

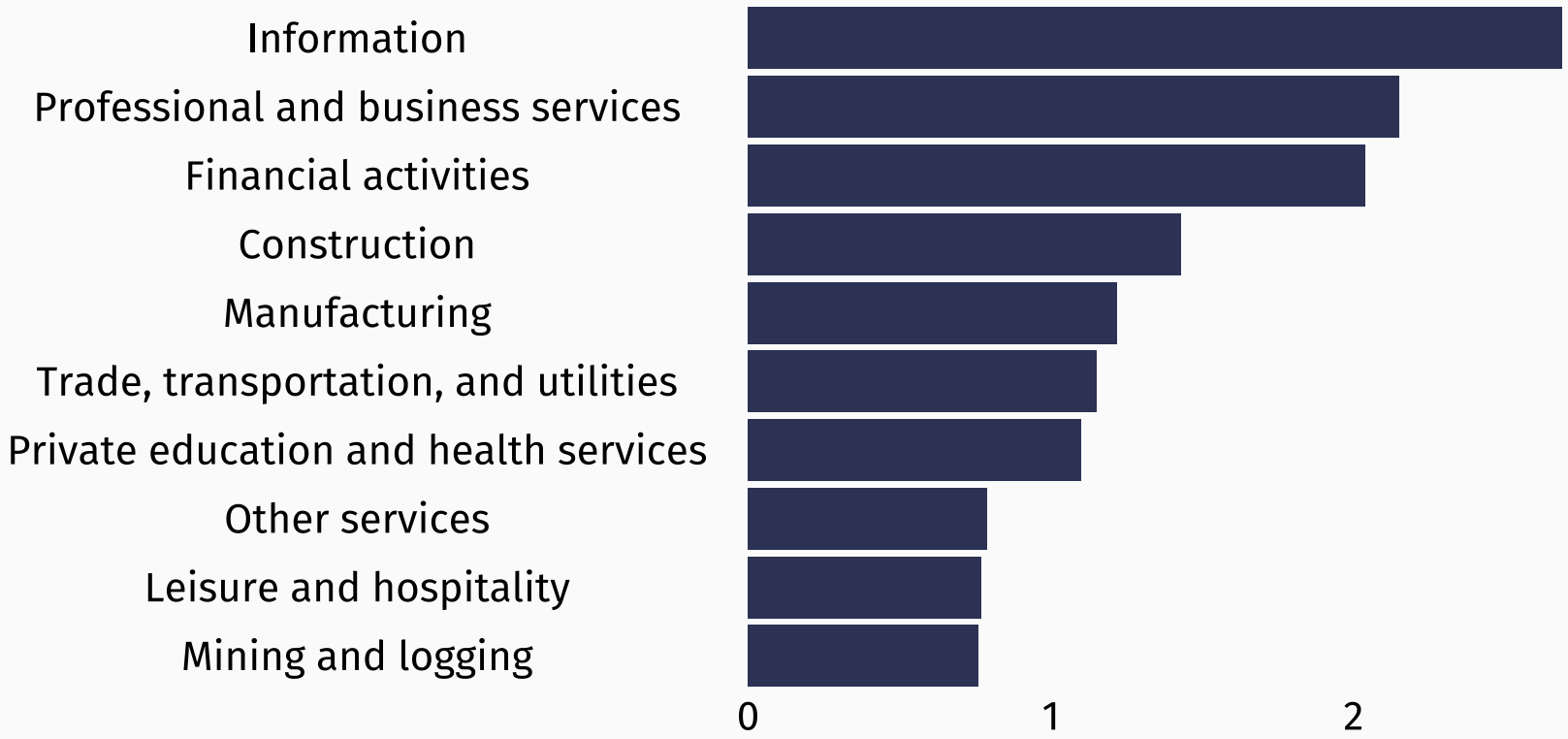
Example 1: Tidy CES Graphics



Example 1: The Grammar of Graphics Deployed

What if we want to change this to hourly earnings? We just swap thing, keeping the rest. The process reusable and auditable.

```
# Watch this line:
# data_type_text == "ALL EMPLOYEES, THOUSANDS")
data_type_text == "AVERAGE HOURLY EARNINGS OF ALL EMPLOYEES") %>%
```



Example 2: The Grammar Deployed on Inflation

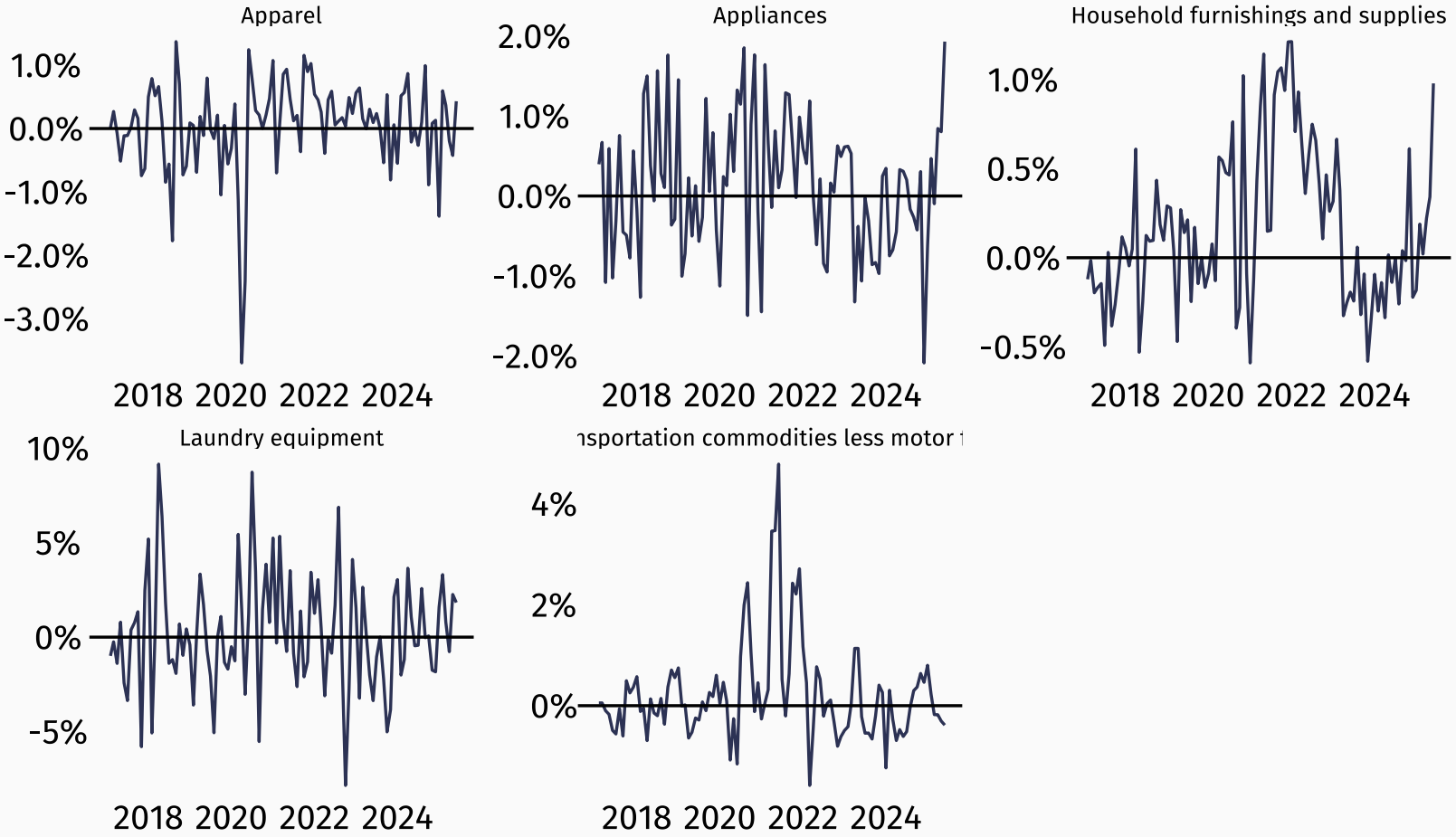
More, this tidy nature makes it easier for faceting. BLS tracks inflation price changes for hundreds of items:

```
tariff_targets <- c(
  "Appliances",
  "Transportation commodities less motor fuel",
  "Apparel",
  "Household furnishings and supplies",
  "Laundry equipment")

cpi %>% filter(item_name %in% tariff_targets) %>%
       select(date, value, item_name) %>% head(5)
```

```
## # A tibble: 5 x 3
##   date       value item_name
##   <date>     <dbl> <chr>
## 1 2017-01-01  126. Apparel
## 2 2017-02-01  126. Apparel
## 3 2017-03-01  126. Apparel
## 4 2017-04-01  126. Apparel
## 5 2017-05-01  126. Apparel
```

Example 2: Faceting Works Well Here

This kind of data is easy to facet and learn from.

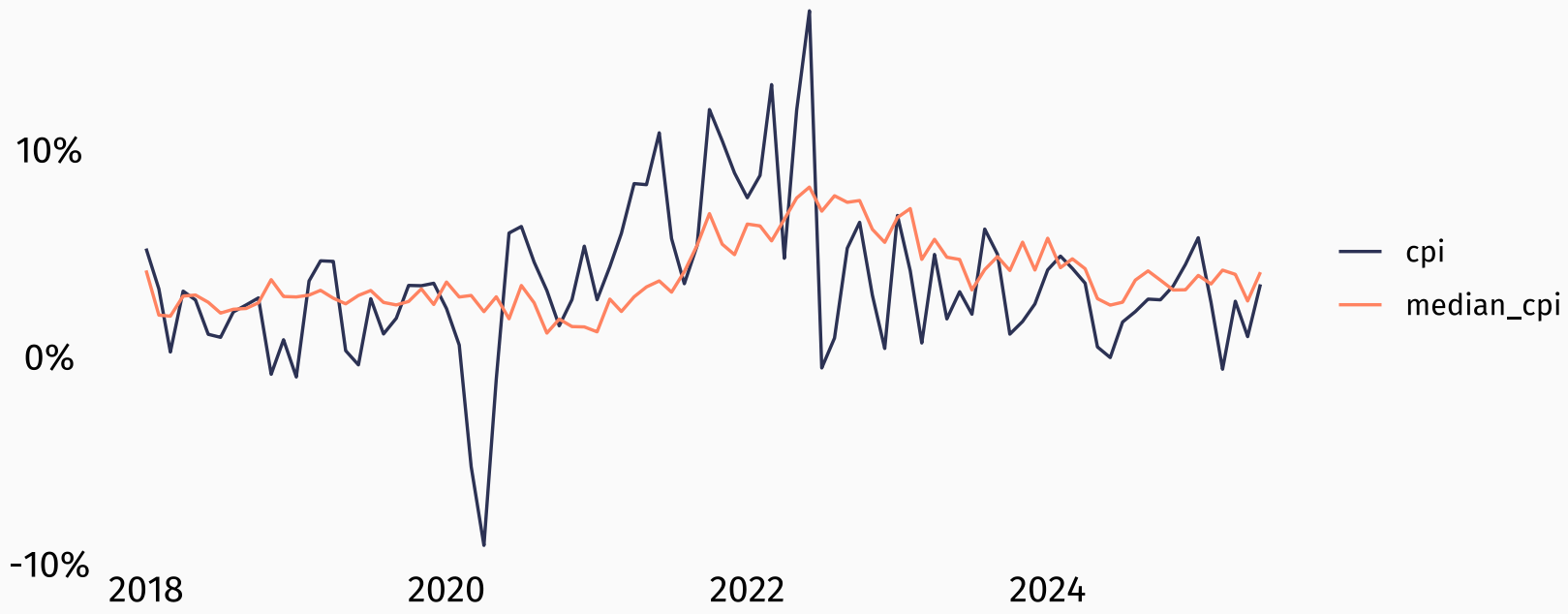


Example 3: Advanced Visualization on Inflation

BLS's CPI is built from about 94,000 price quotes each month, organized into more than 200 categories across 8 groups.

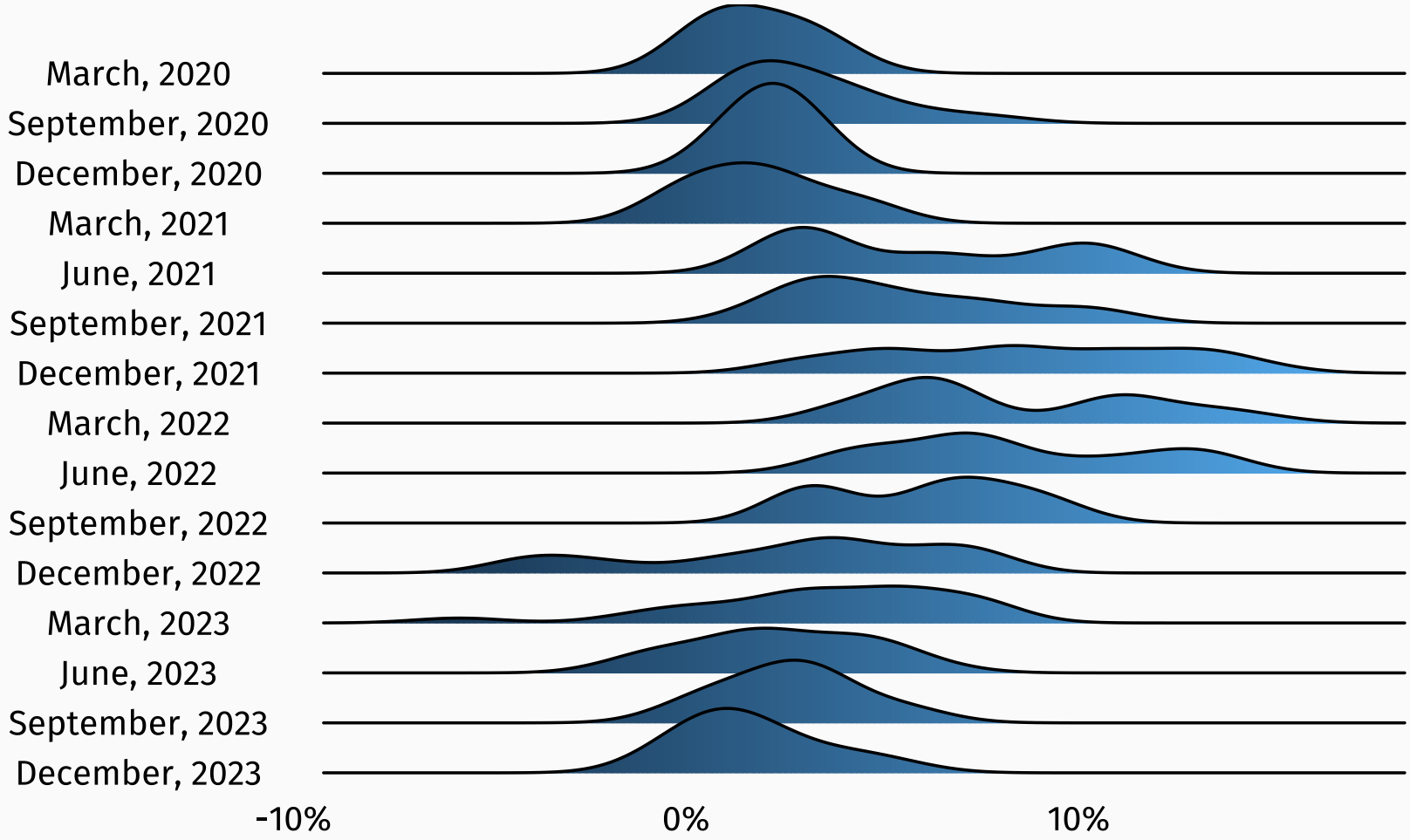
Can do distributions of inflation. The Cleveland Fed's Median CPI is the median change of 45 broad CPI components. Why not go further?

```
getFRED(cpi = "CPIAUCSL", median_cpi = "MEDCPIM158SFRBCLE")
```



Example 3: Ggridges to the Rescue!

We can use the distribution of inflation changes to create a ridgeline graphic.



Example 4: The Phillips Curve - Theory

It's easy to deploy modeling and statistics within an R development workflow.

Take the Phillips Curve, a model where inflation is a positive function of demand, proxied by a negative function of unemployment, as well as past and expected future inflation. One formulation is:

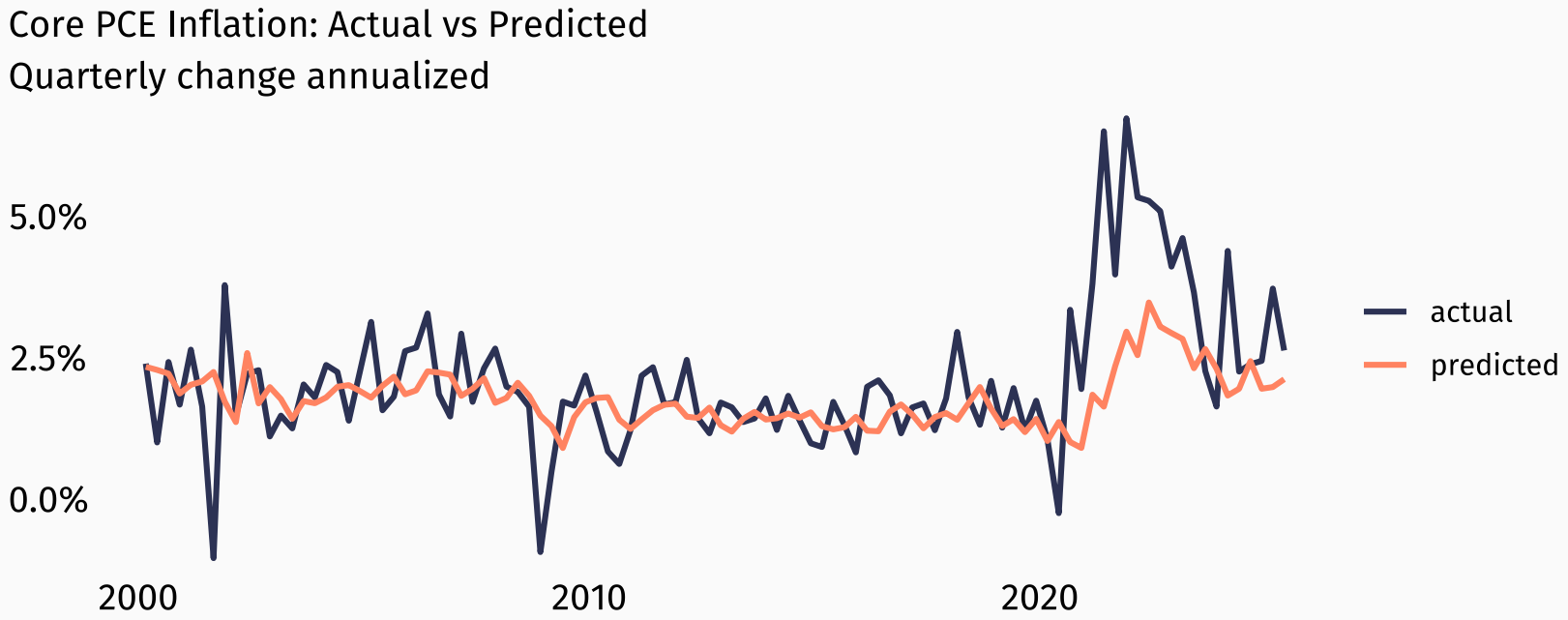
$$\pi_t = \pi_{t-1} + \pi_{t-2} + \pi_t^e + \beta(u - u^*) + \varepsilon_t$$

Example 4: The Phillips Curve - Theory is hard but automation is easy!

```r
library(broom)

phillips_curve <-
  getFRED(
    unrate = "UNRATE", core_pce = "PCEPILFE",
    exp_inf = "EXPINF10YR", keep_all = FALSE) %>%
  left_join(getFRED("nrou") %>%
    mutate(date = date %m+% months(2))) %>%
  mutate(
    core_pce = (core_pce / lag(core_pce, 3))^4 - 1,
    exp_inf = exp_inf / 100,
    unrate_gap = unrate / 100 - nrou / 100,
    lag1 = lag(core_pce, 3), lag2 = lag(core_pce, 6)) %>%
  filter(month(date) %in% c(3, 6, 9, 12)) %>%
  # do Phillips Curve regression pre-COVID and predict
  {
    fit <- lm(core_pce ~ lag1 + lag2 + exp_inf + unrate_gap,
      data = dplyr::filter(., date <= as.Date("2019-12-31"))
    )
    augment(fit, newdata = .)
  }
```

Example 4: The Phillips Curve - Easy to See

We can estimate the Phillips Curve in real time with each monthly update, and see how it is evolving.



Example 5: Other data services

R's libraries make a lot of government data easily accessible:

- **blsR** – BLS API (CES, CPS, CPI, JOLTS, more)
- **bea.R** – BEA national, industry, and regional accounts
- **fredr** – Federal Reserve Economic Data (FRED & ALFRED)
- **tidycensus** – Census ACS & Decennial Census, with geometry
- **ipumsr** – Load IPUMS ACS/CPS microdata extracts
- **eia** – U.S. Energy Information Administration data

Example 5: Other Data Sources

BLS provides employment data for nearly 400 metropolitan statistical areas (MSA). What can we link this with?

```
library(tidycensus)

se %>% select(date, value, data_type_text, area_name) %>%
  head(5)
```

```
## # A tibble: 5 x 4
##   date       value data_type_text              area_name
##   <date>     <dbl> <chr>                       <chr>
## 1 2024-01-01  33.9 All Employees, In Thousands Anniston-Oxford, AL
## 2 2024-02-01  33.9 All Employees, In Thousands Anniston-Oxford, AL
## 3 2024-03-01  33.9 All Employees, In Thousands Anniston-Oxford, AL
## 4 2024-04-01  34.3 All Employees, In Thousands Anniston-Oxford, AL
## 5 2024-05-01  34.2 All Employees, In Thousands Anniston-Oxford, AL
```

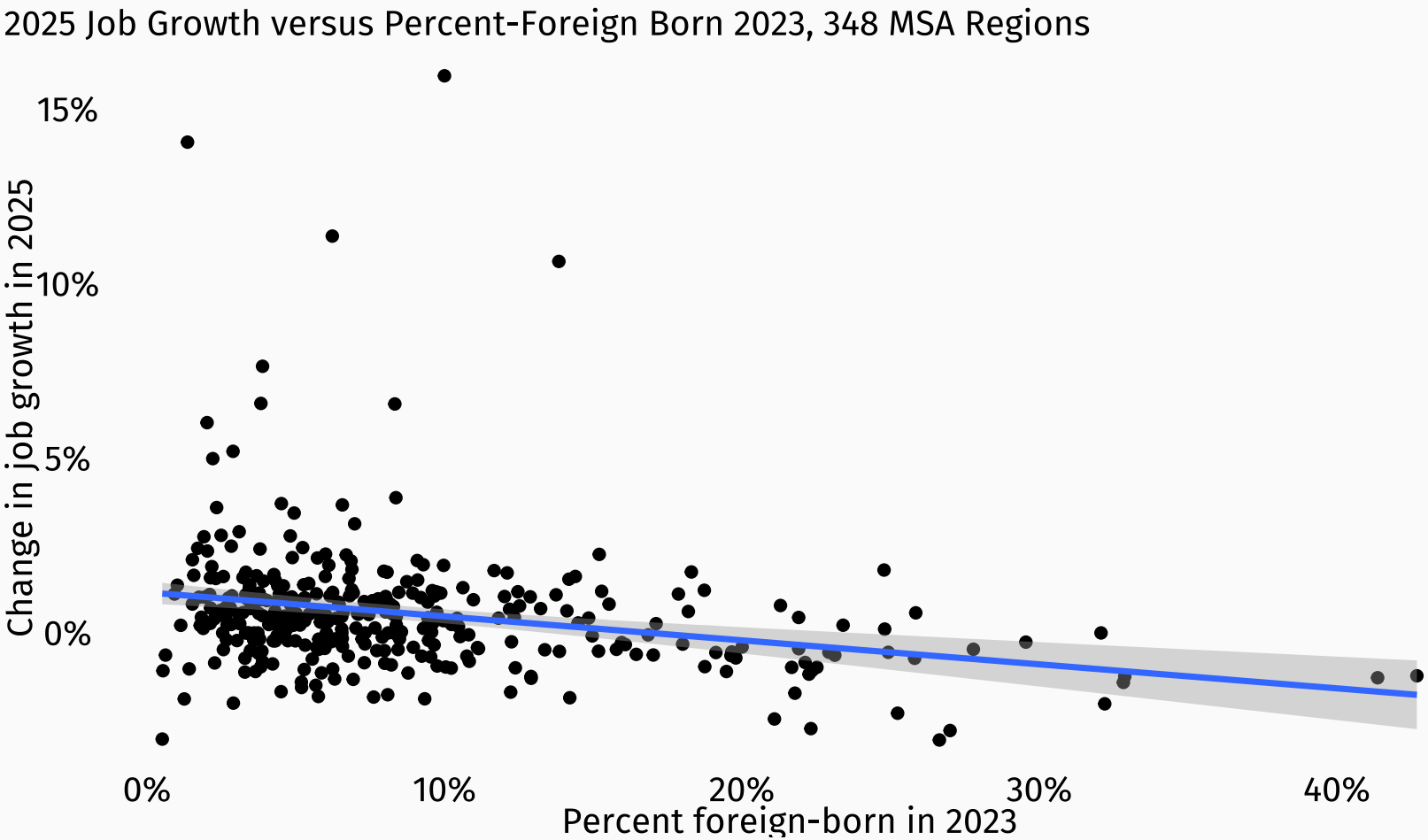
Example 5: Join New Government Data

Tidycensus makes it very easy to get MSA-level data. With a pinch of text wrangling, easy to join into BLS data.

```
library(tidycensus)

vars_b05002 <- c(total = "B05002_001", foreign_born = "B05002_013")

msa_foreign_born <- get_acs(geography = "cbsa", variables = vars_b05002,
                  year = 2023, survey = "acs1", output = "wide") %>%
  reframe(cbsa = GEOID, name = NAME, fb_share = foreign_bornE / totalE)

head(msa_foreign_born, 5)
```

```
## # A tibble: 5 x 3
##   cbsa  name                        fb_share
##   <chr> <chr>                          <dbl>
## 1 10140 Aberdeen, WA Micro Area       0.0630
## 2 10180 Abilene, TX Metro Area        0.0556
## 3 10300 Adrian, MI Micro Area         0.0178
## 4 10380 Aguadilla, PR Metro Area      0.0125
## 5 10420 Akron, OH Metro Area          0.0600
```

Example 5: Easy to Work Into Workflows



Go Forth and R!

4 Reasons to use R for real-time economic data analysis:

1. Tidy Data Workflow
2. Grammar of Graphics
3. Integrated Modeling
4. Extensive Libraries