Today, in honor of last week’s jobs report from the [Bureau of Labor Statistics](https://www.bls.gov/) (BLS), we will visualize jobs data with ggplot2 and then, more extensively with highcharter. Our aim is to explore highcharter and its similarity with ggplot and to create some nice interactive visualizations. In the process, we will cover how to import BLS data from FRED and then wrangle it for visualization. We won’t do any modeling or statistical analysis today, though it wouldn’t be hard to extend this script into a forecasting exercise. One nice thing about today’s code flow is that it can be refreshed and updated on each BLS release date.

Let’s get to it!

We will source our data from [FRED](https://fred.stlouisfed.org/) and will use the tq\_get() function from tidyquant which enables us to import many data series at once in tidy, tibble format. We want to get total employment numbers, ADP estimates, and the sector-by-sector numbers that make up total employment. Let’s start by creating a tibble to hold the FRED codes and more intuitive names for each data series.

library(tidyverse)

library(tidyquant)

codes\_names\_tbl <- tribble(

~ symbol, ~ better\_names,

"NPPTTL", "ADP Estimate",

"PAYEMS", "Nonfarm Employment",

"USCONS", "Construction",

"USTRADE", "Retail/Trade",

"USPBS", "Prof/Bus Serv",

"MANEMP", "Manufact",

"USFIRE", "Financial",

"USMINE", "Mining",

"USEHS", "Health Care",

"USWTRADE", "Wholesale Trade",

"USTPU", "Transportation",

"USINFO", "Info Sys",

"USLAH", "Leisure",

"USGOVT", "Gov",

"USSERV", "Other Services"

)

Now we pass the symbol column to tq\_get().

fred\_empl\_data <-

tq\_get(codes\_names\_tbl$symbol,

get = "economic.data",

from = "2007-01-01")

We have our data but look at the symbol column.

fred\_empl\_data %>%

group\_by(symbol) %>%

slice(1)

# A tibble: 15 x 3

# Groups: symbol [15]

symbol date price

1 MANEMP 2007-01-01 14008

2 NPPTTL 2007-01-01 115437.

3 PAYEMS 2007-01-01 137497

4 USCONS 2007-01-01 7725

5 USEHS 2007-01-01 18415

6 USFIRE 2007-01-01 8389

7 USGOVT 2007-01-01 22095

8 USINFO 2007-01-01 3029

9 USLAH 2007-01-01 13338

10 USMINE 2007-01-01 706

11 USPBS 2007-01-01 17834

12 USSERV 2007-01-01 5467

13 USTPU 2007-01-01 26491

14 USTRADE 2007-01-01 15443.

15 USWTRADE 2007-01-01 5969.

The symbols are the FRED codes, which are unrecognizable unless you have memorized how those codes map to more intuitive names. Let’s replace them with the better\_names column of codes\_names\_tbl. We will do this with a left\_join(). (This explains why I labeled our original column as symbol – it makes the left\_join() easier.) Special thanks to [Jenny Bryan](https://twitter.com/JennyBryan) for pointing out this [code flow](http://stat545.com/bit008_lookup.html)!

fred\_empl\_data %>%

left\_join(codes\_names\_tbl,

by = "symbol" ) %>%

select(better\_names, everything(), -symbol) %>%

group\_by(better\_names) %>%

slice(1)

# A tibble: 15 x 3

# Groups: better\_names [15]

better\_names date price

1 ADP Estimate 2007-01-01 115437.

2 Construction 2007-01-01 7725

3 Financial 2007-01-01 8389

4 Gov 2007-01-01 22095

5 Health Care 2007-01-01 18415

6 Info Sys 2007-01-01 3029

7 Leisure 2007-01-01 13338

8 Manufact 2007-01-01 14008

9 Mining 2007-01-01 706

10 Nonfarm Employment 2007-01-01 137497

11 Other Services 2007-01-01 5467

12 Prof/Bus Serv 2007-01-01 17834

13 Retail/Trade 2007-01-01 15443.

14 Transportation 2007-01-01 26491

15 Wholesale Trade 2007-01-01 5969.

That looks much better, but we now have a column called price, that holds the monthly employment observations, and a column called better\_names, that holds the more intuitive group names. Let’s change those column names to employees and sector.

fred\_empl\_data <-

fred\_empl\_data %>%

left\_join(codes\_names\_tbl,

by = "symbol" ) %>%

select(better\_names, everything(), -symbol) %>%

rename(employees = price, sector = better\_names)

head(fred\_empl\_data)

# A tibble: 6 x 3

sector date employees

1 ADP Estimate 2007-01-01 115437.

2 ADP Estimate 2007-02-01 115527.

3 ADP Estimate 2007-03-01 115647

4 ADP Estimate 2007-04-01 115754.

5 ADP Estimate 2007-05-01 115809.

6 ADP Estimate 2007-06-01 115831.

fred\_empl\_data has the names and organization we want, but it still has the raw number of employees per month. We want to visualize the month-to-month *change* in jobs numbers, which means we need to perform a calculation on our data and store it in a new column. We use mutate() to create the new column and calculate monthly change with value - lag(value, 1). We are not doing any annualizing or seasonality work here – it’s a simple substraction. For yearly change, it would be value - lag(value, 12).

empl\_monthly\_change <-

fred\_empl\_data %>%

group\_by(sector) %>%

mutate(monthly\_change = employees - lag(employees, 1)) %>%

na.omit()

Our final data object empl\_monthly\_change is tidy, has intuitive names in the group column, and has the monthly change that we wish to visualize. Let’s build some charts.

We will start at the top and use ggplot to visualize how total non-farm employment (Sorry farmers. Your jobs don’t count, I guess) has changed since 2007. We want an end-user to quickly glance at the chart and find the months with positive jobs growth and negative jobs growth. That means we want months with positive jobs growth to be one color, and those with negative jobs growth to be another color. There is more than one way to accomplish this, but I like to create new columns and then add geoms based on those columns. (Check out [this post](http://lenkiefer.com/2018/03/11/charting-jobs-friday-with-r/) by Freddie Mac’s Len Kiefer for another way to accomplish this by nesting ifelse statements in ggplot's aesthetics. In fact, if you like data visualization, check out all the stuff that Len writes.)

Let’s walk through how to create columns for shading by positive or negative jobs growth. First, we are looking at total employment here, so we call filter(sector == "Nonfarm Employment") to get only total employment.

Next, we create two new columns with mutate(). The first is called col\_pos and is formed by if\_else(monthly\_change > 0, monthly\_change,...). That logic is creating a column that holds the value of monthly change if monthly change is positive, else it holds NA. We then create another column called col\_neg using the same logic.

empl\_monthly\_change %>%

filter(sector == "Nonfarm Employment") %>%

mutate(col\_pos =

if\_else(monthly\_change > 0,

monthly\_change, as.numeric(NA)),

col\_neg =

if\_else(monthly\_change < 0,

monthly\_change, as.numeric(NA))) %>%

dplyr::select(sector, date, col\_pos, col\_neg) %>%

head()

# A tibble: 6 x 4

# Groups: sector [1]

sector date col\_pos col\_neg

1 Nonfarm Employment 2007-02-01 85 NA

2 Nonfarm Employment 2007-03-01 214 NA

3 Nonfarm Employment 2007-04-01 59 NA

4 Nonfarm Employment 2007-05-01 153 NA

5 Nonfarm Employment 2007-06-01 77 NA

6 Nonfarm Employment 2007-07-01 NA -30

Have a qucik look at the col\_pos and col\_neg columns and make sure they look right. col\_pos should have only positive and NA values, col\_neg shoud have only negative and NA values.

Now we can visualize our monthly changes with ggplot, adding a separate geom for those new columns.

empl\_monthly\_change %>%

filter(sector == "Nonfarm Employment") %>%

mutate(col\_pos =

if\_else(monthly\_change > 0,

monthly\_change, as.numeric(NA)),

col\_neg =

if\_else(monthly\_change < 0,

monthly\_change, as.numeric(NA))) %>%

ggplot(aes(x = date)) +

geom\_col(aes(y = col\_neg),

alpha = .85,

fill = "pink",

color = "pink") +

geom\_col(aes(y = col\_pos),

alpha = .85,

fill = "lightgreen",

color = "lightgreen") +

ylab("Monthly Change (thousands)") +

labs(title = "Monthly Private Employment Change",

subtitle = "total empl, since 2008",

caption = "inspired by @lenkiefer") +

scale\_x\_date(breaks = scales::pretty\_breaks(n = 10)) +

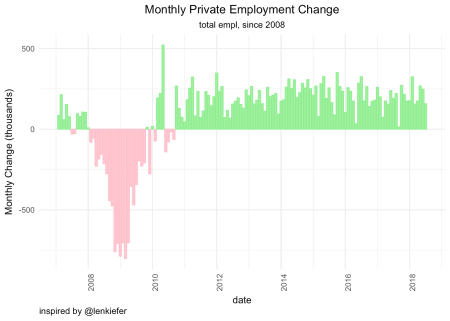
theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1),

plot.title = element\_text(hjust = 0.5),

plot.subtitle = element\_text(hjust = 0.5),

plot.caption = element\_text(hjust=0))



That plot is nice, but it’s static! Hover on it and you’ll see what I mean.

Let’s head to highcharter and create an interactive chart that responds when we hover on it. By way of brief background, highcharter is an R hook into the fantastic [highcharts](https://rviews.rstudio.com/2018/08/09/highcharting-jobs-friday/www.highcharts.com) JavaScript library. It’s free for personal use but a license is required for commercial use.

One nice feature of highcharter is that we can use very similar aesthetic logic to what we used for ggplot. It’s not identical, but it’s similar and let’s us work with tidy data.

Before we get to the highcharter logic, we will add one column to our tibble to hold the color scheme for our positive and negative monthly changes. Notice how this is different from the ggplot flow above where we create one column to hold our positive changes for coloring and one column to hold our negative changes for coloring.

I want to color positive changes light blue and negative changes pink, and put the [rgb](https://www.w3schools.com/colors/colors_picker.asp) codes for those colors directly in the new column. The rgb code for light blue is “#6495ed” and for pink is “#ffe6ea”. Thus we use ifelse to create a column called color\_of\_bars that holds “#6495ed” (light blue) when monthly\_change is postive and “#ffe6ea” (pink) when it’s negative.

total\_employ\_hc <-

empl\_monthly\_change %>%

filter(sector == "Nonfarm Employment") %>%

mutate(color\_of\_bars = ifelse(monthly\_change > 0, "#6495ed", "#ffe6ea"))

head(total\_employ\_hc)

# A tibble: 6 x 5

# Groups: sector [1]

sector date employees monthly\_change color\_of\_bars

1 Nonfarm Employment 2007-02-01 137582 85 #6495ed

2 Nonfarm Employment 2007-03-01 137796 214 #6495ed

3 Nonfarm Employment 2007-04-01 137855 59 #6495ed

4 Nonfarm Employment 2007-05-01 138008 153 #6495ed

5 Nonfarm Employment 2007-06-01 138085 77 #6495ed

6 Nonfarm Employment 2007-07-01 138055 -30 #ffe6ea

Now we are ready to start the highcharter flow.

We start by calling hchart to pass in our data object. Note the similarity to ggplot where we started with ggplot.

Now, intead of waiting for a call to geom\_col, we set type = "column" to let hchart know that we are building a column chart. Next, we use hcaes(x = date, y = monthly\_change, color = color\_of\_bars) to specify our aesthetics. Notice how we can control the colors of the bars from values in the color\_of\_bars column.

We also supply a name = "monthly change" because we want monthly change to appear when a user hovers on the chart. That wasn’t a consideration with ggplot.

library(highcharter)

hchart(total\_employ\_hc,

type = "column",

pointWidth = 5,

hcaes(x = date,

y = monthly\_change,

color = color\_of\_bars),

name = "monthly change") %>%

hc\_title(text = "Monthly Employment Change") %>%

hc\_xAxis(type = "datetime") %>%

hc\_yAxis(title = list(text = "monthly change (thousands)")) %>%

hc\_exporting(enabled = TRUE)

Let’s stay in the highcharter world and visualize how each sector changed in the most recent month, which is July of 2018.

First, we isolate the most recent month by filtering on the last date. We also don’t want the ADP Estimate and filter that out as well.

empl\_monthly\_change %>%

filter(date == (last(date))) %>%

filter(sector != "ADP Estimate")

# A tibble: 14 x 4

# Groups: sector [14]

sector date employees monthly\_change

1 Nonfarm Employment 2018-07-01 149128 157

2 Construction 2018-07-01 7242 19

3 Retail/Trade 2018-07-01 15944 7.1

4 Prof/Bus Serv 2018-07-01 21019 51

5 Manufact 2018-07-01 12751 37

6 Financial 2018-07-01 8568 -5

7 Mining 2018-07-01 735 -4

8 Health Care 2018-07-01 23662 22

9 Wholesale Trade 2018-07-01 5982. 12.3

10 Transportation 2018-07-01 27801 15

11 Info Sys 2018-07-01 2772 0

12 Leisure 2018-07-01 16371 40

13 Gov 2018-07-01 22334 -13

14 Other Services 2018-07-01 5873 -5

That filtered flow has the data we want, but we have two more tasks. First, we want to arrange this data so that it goes from smallest to largest. If we did not do this, our chart would still “work”, but the column heights would not progress from lowest to highest.

Second, we need to create another column to hold colors for negative and positive values, with the same ifelse() logic as we used before.

emp\_by\_sector\_recent\_month <-

empl\_monthly\_change %>%

filter(date == (last(date))) %>%

filter(sector != "ADP Estimate") %>%

arrange(monthly\_change) %>%

mutate(color\_of\_bars = if\_else(monthly\_change > 0, "#6495ed", "#ffe6ea"))

Now we pass that object to hchart, set type = "column", and choose our hcaes values. We want to label the x-axis with the different sectors and do that with hc\_xAxis(categories = emp\_by\_sector\_recent\_month$sector).

last\_month <- lubridate::month(last(empl\_monthly\_change$date),

label = TRUE,

abbr = FALSE)

hchart(emp\_by\_sector\_recent\_month,

type = "column",

pointWidth = 20,

hcaes(x = sector,

y = monthly\_change,

color = color\_of\_bars),

showInLegend = FALSE) %>%

hc\_title(text = paste(last\_month, "Employment Change", sep = " ")) %>%

hc\_xAxis(categories = emp\_by\_sector\_recent\_month$sector) %>%

hc\_yAxis(title = list(text = "Monthly Change (thousands)"))

Finally, let’s compare the ADP Estimates to the actual Nonfarm payroll numbers since 2017. We start with filtering again.

adp\_bls\_hc <-

empl\_monthly\_change %>%

filter(sector == "ADP Estimate" | sector == "Nonfarm Employment") %>%

filter(date >= "2017-01-01")

We create a column to hold different colors, but our logic is not whether a reading is positive or negative. We want to color the ADP and BLS reports differently.

adp\_bls\_hc <-

adp\_bls\_hc %>%

mutate(color\_of\_bars =

ifelse(sector == "ADP Estimate", "#ffb3b3", "#4d94ff"))

head(adp\_bls\_hc)

# A tibble: 6 x 5

# Groups: sector [1]

sector date employees monthly\_change color\_of\_bars

1 ADP Estimate 2017-01-01 123253. 245. #ffb3b3

2 ADP Estimate 2017-02-01 123533. 280. #ffb3b3

3 ADP Estimate 2017-03-01 123655 122. #ffb3b3

4 ADP Estimate 2017-04-01 123810. 155. #ffb3b3

5 ADP Estimate 2017-05-01 124012. 202. #ffb3b3

6 ADP Estimate 2017-06-01 124166. 154. #ffb3b3

tail(adp\_bls\_hc)

# A tibble: 6 x 5

# Groups: sector [1]

sector date employees monthly\_change color\_of\_bars

1 Nonfarm Employment 2018-02-01 148125 324 #4d94ff

2 Nonfarm Employment 2018-03-01 148280 155 #4d94ff

3 Nonfarm Employment 2018-04-01 148455 175 #4d94ff

4 Nonfarm Employment 2018-05-01 148723 268 #4d94ff

5 Nonfarm Employment 2018-06-01 148971 248 #4d94ff

6 Nonfarm Employment 2018-07-01 149128 157 #4d94ff

And now we pass that object to our familiar hchart flow.

hchart(adp\_bls\_hc,

type = 'column',

hcaes(y = monthly\_change,

x = date,

group = sector,

color = color\_of\_bars),

showInLegend = FALSE

) %>%

hc\_title(text = "ADP v. BLS") %>%

hc\_xAxis(type = "datetime") %>%

hc\_yAxis(title = list(text = "monthly change (thousands)")) %>%

hc\_add\_theme(hc\_theme\_flat()) %>%

hc\_exporting(enabled = TRUE)

That’s all for today. Try revisiting this script on September 7th, when the next BLS jobs data is released, and see if any new visualizations or code flows come to mind.