Now we are going to plot the labor force participation rates and the percent of workers by work situation. And in the next post, I’ll show how to write functions to automate repetitive plotting tasks using the 2016 Census education data as an example.

As always, let’s start with loading the required packages. Note **ggrepel** package, which helps to prevent overlapping of data points and text labels in our graphics.

# load packages

library(tidyverse)

library(ggrepel)

**Ordered Bar Plot: Labor Force Involvement Rates**

Why the bar plot for this data? Well, the bar plot is one of the simplest and thus easiest to interpret plots, and the data – labor force involvement rates – fits this type of plot nicely. We will plot the rates for all our regions in the same graphic, and we are going to order regions by unemployment rate.

**Creating an Ordering Vector**

The 2016 Census data for labor force involvement rates, did some preparatory work required to plot the data with **ggplot2** package, and saved the data as the ‘labor’ dataframe. There is one more step we need to complete before we can plot this data: we need to create an ordering vector with unemployment numbers and append this vector to ‘labor’.

# prepare 'labor' dataset for plotting:

# create an ordering vector to set the order of regions in the plot

labor <- labor %>%

group\_by(region) %>% # groups data by region

filter(indicator == "unemployment rate") %>%

select(-indicator) %>%

rename(unemployment = rate) %>%

left\_join(labor, by = "region") %>%

mutate(indicator = factor(indicator,

levels = c("participation rate",

"employment rate",

"unemployment rate")))

Note the left\_join call, which joins the result of manipulating the ‘labor’ dataframe back onto ‘labor’. If it seems confusing, take a look at this code, which returns the same output:

# alt. (same output):

labor\_order <- labor %>%

filter(indicator == "unemployment rate") %>%

select(-indicator) %>%

rename(unemployment = rate)

labor <- labor %>%

left\_join(labor\_order, by = "region") %>%

mutate(indicator = factor(indicator,

levels = c("participation rate",

"employment rate",

"unemployment rate")))

Also note the mutate call that manually re-assigns factor levels of the ‘indicator’ variable, so that labor force indicators are plotted in the logical order: first labor force participation rate, then employment rate, and finally the unemployment rate. Remember that **ggplot2** plots categorical variables in the order of factor levels.

**Making an Ordered Bar Plot**

# plot data

plot\_labor <-

labor %>%

ggplot(aes(x = reorder(region, unemployment),

y = rate,

fill = indicator)) +

geom\_col(width = .6, position = "dodge") +

geom\_text(aes(label = rate),

position = position\_dodge(width = .6),

show.legend = FALSE,

size = 3.5,

vjust = -.4) +

scale\_y\_continuous(name = "Percent",

breaks = seq(0, 80, by = 10)) +

scale\_x\_discrete(name = NULL) +

scale\_fill\_manual(name = "Indicator:",

values = c("participation rate" = "deepskyblue2",

"employment rate" = "olivedrab3",

"unemployment rate" = "tomato")) +

theme\_bw() +

theme(plot.title = element\_text(hjust = .5, size = 14,

face = "bold"),

plot.subtitle = element\_text(hjust = .5,

size = 13,

margin = margin(b = 15)),

panel.grid.major = element\_line(colour = "grey88"),

panel.grid.minor = element\_blank(),

axis.text = element\_text(size = 12, face = "bold"),

axis.title.y = element\_text(size = 12, face = "bold",

margin = margin(r = 8)),

legend.title = element\_text(size = 12, face = "bold"),

legend.text = element\_text(size = 12),

legend.position = "bottom",

plot.caption = element\_text(size = 11, hjust = 0,

margin = margin(t = 15))) +

labs(title = "Labor Force Indicators in Canada's Geographic Regions' Largest Cities in 2016",

subtitle = "Compared to Canada, Ordered by Unemployment Rate",

caption = "Data: Statistics Canada 2016 Census.")

Note the x = reorder(region, unemployment) inside the aes call: this is where we order the plot’s x axis by unemployment rates. Remember that we have [grouped](https://dataenthusiast.ca/?p=1122#ordering-vector) our data by region so that we could put regions on the X axis.

Note also the scale\_fill\_manual function, where we manually assign colors to the plot’s fill aesthetic (hence scale\_**fill**\_manual).

**Saving the Plot**

Now that we have made the plot, let’s create the directory where we will be saving our graphics, and save our plot to it:

# save plot to a specific folder

dir.create("output") # creates folder

ggsave("output/plot\_labor.png",

plot\_labor,

width = 11, height = 8.5, units = "in")

Finally, let’s print the plot to screen:

# print plot to screen

print(plot\_labor)

**Faceted Plot: Full Time vs Part Time Workers, by Gender**

This will be a more complex task compared to plotting labor force participation rates. Here we have the data that is broken down by work situation (full-time vs part-time), and by gender, and also by region. And ideally, we also want the total numbers for full-time and part-time workers to be presented in the same plot. This is too complex to be visualized as a simple bar plot like the one we’ve just made.

To visualize all these data in a single plot, we’ll use *faceting*: breaking down one plot into multiple sub-plots. And I suggest a donut chart – a variation on a pie chart that has a round hole in the center.

So why an I using a pie chart? Well, three reasons, really. First, we’ll only have four categories inside the chart, so it won’t be messy. Second, it is technically a donut chart, not a pie chart, and it is the empty space inside each donut where I will put the total numbers for full- and part-time workers. And third, I’d like to show how to make donut charts with **ggplot2** in case you ever need this.

**Preparing the Data for Plotting**

The 2016 Census data on the percentage of full-time and part-time workers, by gender, and saved it in the ‘work’ dataframe. Let’s now prepare the data for plotting. For that, we’ll need to add three more variables. ‘type\_gender’ will be a categorical variable that combines work type and gender – currently these are two different variables. ‘percent’ will contain percentages for each combination of work type and gender, by region. And ‘percent\_type’ will contain total percentages for full-time and part-time workers, by region.

# prepare 'work' dataset for plotting:

work <- work %>%

group\_by(region) %>%

mutate(type\_gender = str\_c(type, gender, sep = " ")) %>%

# percent of workers by region, work type, and gender

mutate(percent = round(count/sum(count)\*100, 1)) %>%

# percent of workers by work type, total

group\_by(region, type) %>%

mutate(percent\_type = sum(percent))

**Making a Faceted Donut Plot**

Now the dataset is ready for plotting, so let’s make a *faceted* plot. Since **ggplot2** doesn’t like pie-charts (of which a donut chart is a variant), there is no ‘pie’ geom, and we’ll have to get a bit hacky with the code. Pay close attention to the in-code comments.

# plot work data (as a faceted plot)

plot\_work <-

work %>%

ggplot(aes(x = "",

y = percent,

fill = type\_gender)) +

geom\_col(color = "white") + # sectors' separator color

coord\_polar(theta = "y") +

geom\_text\_repel(aes(label = percent),

# put text labels inside corresponding sectors:

position = position\_stack(vjust = .5),

# repelling force:

force = .02,

size = 4.5) +

geom\_label\_repel(data = distinct(select(work, c("region",

"type",

"percent\_type"))),

aes(x = 0, # turns pie chart into donut chart

y = percent\_type,

label = percent\_type,

fill = type),

size = 4.5,

fontface = "bold",

force = .02, # repelling force

show.legend = FALSE) +

scale\_fill\_manual(name = "Work situation",

labels = c("full time" = "all full-time",

"part time" = "all part-time"),

values = c("full time male" = "olivedrab4",

"full time female" = "olivedrab1",

"part time male" = "tan4",

"part time female" = "tan1",

"full time" = "green3",

"part time" = "orange3")) +

facet\_wrap(~ region) +

guides(fill = guide\_legend(nrow = 3)) +

theme\_void() +

theme(plot.title = element\_text(size = 14, face = "bold",

margin = margin(t = 10, b = 20),

hjust = .5),

strip.text = element\_text(size = 12, face = "bold"),

plot.caption = element\_text(size = 11, hjust = 0,

margin = margin(t = 20, b = 10)),

legend.title = element\_text(size = 12, face = "bold"),

legend.text = element\_text(size = 12),

# change size of symbols (colored squares) in legend:

legend.key.size = unit(1.1, "lines"),

legend.position = "bottom") +

labs(title = "Percentage of Workers, by Work Situation & Gender, 2016",

caption = "Note: Percentages may not add up to 100% due to values rounding.\nData source: Statistics Canada 2016 Census.")

Here is our plot:

There are a number of things in the plot’s code that I’d like to draw your attention to. First, a **ggplot2** pie chart is a *stacked* bar chart (geom\_col) made in the [polar coordinate system](https://en.wikipedia.org/wiki/Polar_coordinate_system): coord\_polar(theta = “y”). For geom\_col, position = “stack” is the default, so it is not specified in the code. Note also that geom\_col needs the X aesthetic, but a pie chart doesn’t have an X coordinate. So I used x = “” to trick geom\_col into thinking it has the X aesthetic, otherwise it would have thrown an error: “geom\_col requires the following missing aesthetics: x”.

But how do you turn a pie chart into a donut chart? To do this, I set x = 0 inside the ggrepel::geom\_label\_repel aes call. Try passing different values to x to see how it works: for example, x = 1 turns the plot into a standard pie chart, while x = -1 turns a donut into a ring.

In order to prevent the labels overlapping, I used ggrepel::geom\_text\_repel and ggrepel::geom\_label\_repel to add text labels to our plot instead of ggplot2::geom\_text and ggplot2::geom\_label. And position = position\_stack(vjust = .5) inside geom\_text\_repel puts text labels in the middle of their respective sectors of the donut plot.

The data = distinct(select(work, c(“region”, “type”, “percent\_type”)) argument to geom\_label\_repel prevents the duplication of labels containing total numbers for full-time and part-time workers.

The scale\_fill\_manual is used to manually assign colors and names to our plot’s legend items, and guides(fill = guide\_legend(nrow = 3)) changes the order of legend items.

Finally, facet\_wrap(~ region) creates a faceted plot, by region.

And just as we did with the previous plot, let’s save our plot to the ‘output’ folder and print it to screen:

# save plot to 'output' folder

ggsave("output/plot\_work.png",

plot\_work,

width = 11, height = 8.5, units = "in")

# print work plot to screen

print(plot\_work)