Experiments with ggplot2 barcharts

a simple data exploration exercise

Richard Robbins

March 1, 2022

Contents

1 Introduction

This document is intended to serve as a perpetual work in progress as I explore *ggplot2* and learn how to use *figures* and *figure references* in R markdown. I approach things in a very deliberate fashion, pulling apart elements one at a time. I highlight and discuss things that may be self-evident with a view to precision and being complete.

2 Global settings

2.1 YAML header block

This document uses figure labels and cross references. These features require the use of `bookdown` output format which is specified in the `YAML` header block at the beginning of the file as follows.

```
---
title: "Experiments with ggplot2 barcharts"
subtitle:  "a simple data exploration exercise"
author: "Richard Robbins"
date: "March 1, 2022"
output:
  bookdown::pdf_document2:
  toc: true
  number_sections: 2
  toc_depth: 2
---
```

2.2 Required packages and default settings

Here are the packages we need.

- tidyverse
- knitr
- patchwork

Here are few options I care about.

- `knitr` `echo = TRUE` to show code fragments
- `knitr` default figure height to 3

I set *ggplot's* default theme to `theme_minimal`.

```
require(tidyverse)
require(knitr)
require(patchwork)

knitr::opts_chunk$set(echo = TRUE, fig.height = 3)
knitr::opts_chunk$set(echo = TRUE)

theme_set(theme_minimal())
```

3 Data

3.1 Load the data

We use a data set that was created for the first lab in w203. It includes information about voting in the 2020 general election and was derived from a much larger set of reference data.

```
df <- readRDS("anes_data.rds")
```

3.2 Eliminate irrelevant columns

The data set includes a handful of columns from the original source data set that are not relevant here. Those irrelevant columns all have names that begin with `V2`. We remove them.

```
df <- df %>% select(-starts_with("V2"))
```

Here are the columns that remain.

```
colnames(df)
```

```
## [1] "case_id"                   "party"
## [3] "voter.difficulty_level"    "voted"
## [5] "voted.difficulty_level"    "voted.problem_mentioned"
## [7] "presumed"                  "presumed.reason"
## [9] "presumed.difficulty_level"
```

This exercise is only concerned with Republicans and Democrats. So, we remove information for people who are not identified as having a party affiliation.

```
df <- df %>% drop_na(party)
table(df$party)
```

```
##
##   Democrat Republican
##       3242       2808
```

3.3 Reviewing a few key columns

Let's explore `voted.difficulty_level` by party. The `voter.difficulty_level` variable is not relevant for this exercise. It is an adjusted version of `voted.difficulty_level`. We will also use the `presumed.reason` data near the end of this document.

Here are the voting difficulty levels grouped by party.

```
table(df$party, df$voted.difficulty_level)
```

```
##
##               not little moderate very extreme
##   Democrat   2751    251       92   20      14
##   Republican 2442    154       58   28      18
```

3.4 Data type matters too

In addition to exploring the values for the data we care about, we should be mindful of the data types. The `party`, `voted.difficulty_level` and `presumed.reason` variables are all `R` factors.

```
str(df$party)
```

```
##  Factor w/ 2 levels "Democrat","Republican": 2 1 2 1 2 1 1 1 2 1 ...
```

```
str(df$voted.difficulty_level)
```

```
##  Ord.factor w/ 5 levels "not"<"little"<..: NA 2 1 2 1 2 1 1 1 1 ...
```

```
str(df$presumed.reason)
```

```
##  Factor w/ 16 levels "forgot","not interested",..: 11 NA NA NA NA NA NA NA NA NA ...
```

Unlike the other two variables, `voted.difficulty_level` is an ordered factor. That means that operators like `>` and `<` are meaningful. Those operators cannot be applied to a factor that is not also an ordered factor, which makes sense.

For example, we can filter on `voted.difficulty_level` to isolate people who had more than a little difficulty. But we can't perform a similar operation on the `party` variable. Consider the following.

```
little_table <- select(df, c(party, voted.difficulty_level))
table(little_table)
```

```
##             voted.difficulty_level
## party         not little moderate very extreme
##   Democrat   2751    251       92   20      14
##   Republican 2442    154       58   28      18
```

```
table(little_table %>% filter(voted.difficulty_level > "little"))
```

```
##             voted.difficulty_level
## party        not little moderate very extreme
##   Democrat     0      0       92   20      14
##   Republican   0      0       58   28      18
```

```
filter(little_table, party < "Democrat")
```

```
## Warning in Ops.factor(party, "Democrat"): '<' not meaningful for factors
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: party <fct>, voted.difficulty_level <ord>
```

4 Usually, voting is not difficult

We want to produce visualizations to tell a story about the voting difficulty data we have. It's often hard to look at a table of numbers and draw conclusions quickly. Here, the key variable is a five point Likert scale. Bar charts work very well to visualize categorical information like this.

One thing that is apparent from the data is that most voters did not report difficulty. Let's work with that.

4.1 Default bar chart

Figure 1 shows the default bar chart for the `voted.difficulty_level` variable.

```
df %>% ggplot() +
       aes (x=voted.difficulty_level) +
       geom_bar()
```

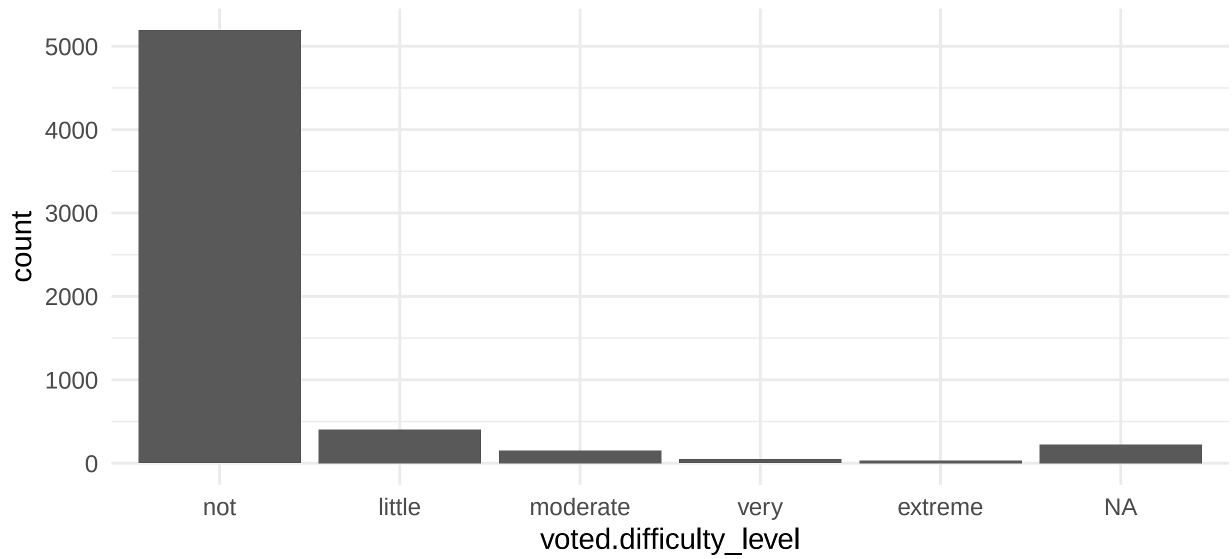

Figure 1: Default bar chart

4.2 Differentiating data between the parties

Since our goal is to compare data between Republicans and Democrats, we use `fill` to differentiate by party. See Figure 2.

```
df %>% ggplot() +
       aes (x=voted.difficulty_level, fill = party) +
       geom_bar()
```

4.3 Side by side instead of stacked

We want to compare the parties for each level of voting difficulty. That's easier to do if we use `dodge` to show the data side-by-side instead of stacked. See Figure 3.

```
df %>% ggplot() +
       aes (x=voted.difficulty_level, fill = party) +
       geom_bar(position = "dodge")
```

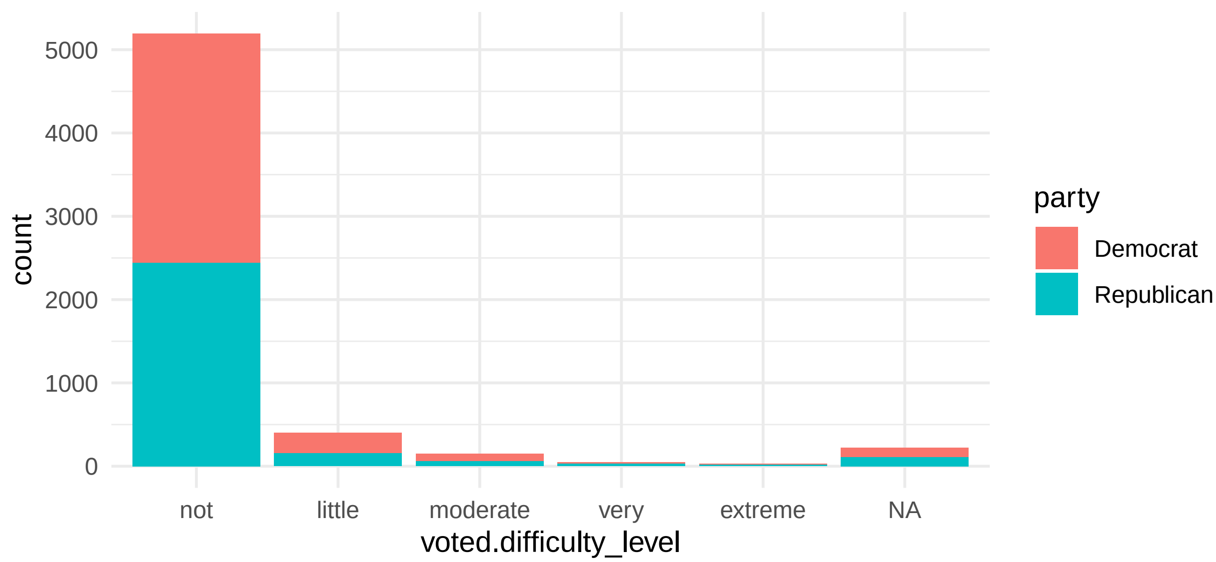

Figure 2: Differentiating between the parties

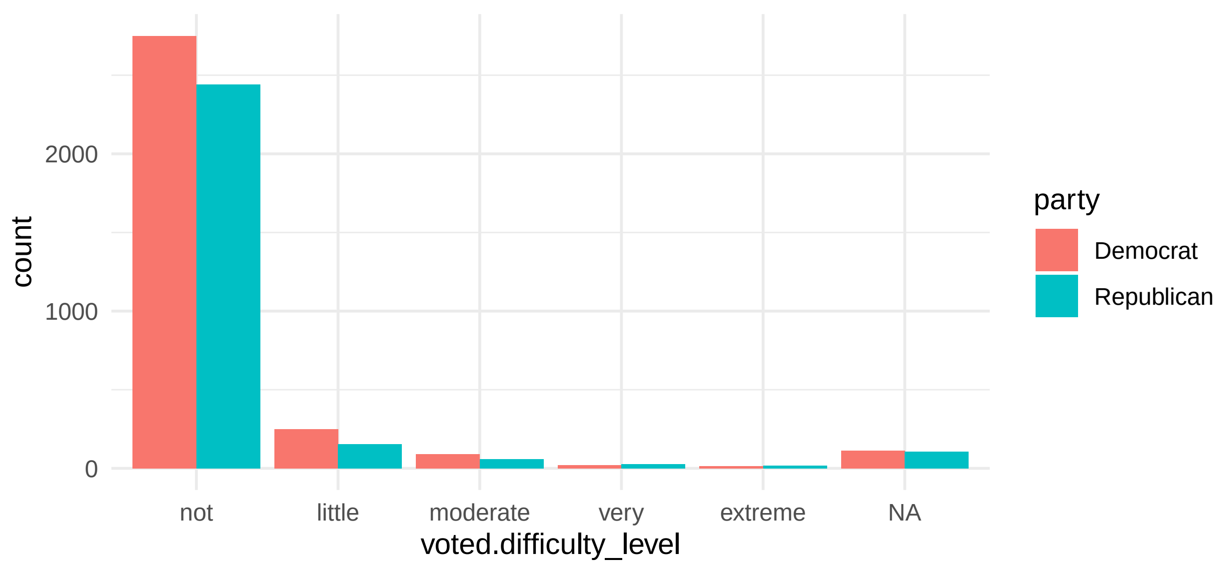

Figure 3: Using a side-by-side presentation

4.4 Eliminating NA as a category

The chart shows `NA` values as a separate category. Let's take a closer look at the data, once again with `table`, but this time we want to show `NA`.

```
table(df$party, df$voted.difficulty_level, useNA = "ifany")
```

```
##
##             not little moderate very extreme <NA>
##   Democrat   2751    251       92   20      14  114
##   Republican 2442    154       58   28      18  108
```

Sure enough, we have `NA` values in the data. There are several ways to eliminate `NA` from the x axis of the plot. We filter those values out of the data before plotting the bar chart. See Figure 4.

```
df %>% drop_na(voted.difficulty_level) %>%
  ggplot() +
  aes (x=voted.difficulty_level, fill=party) +
  geom_bar(position="dodge")
```

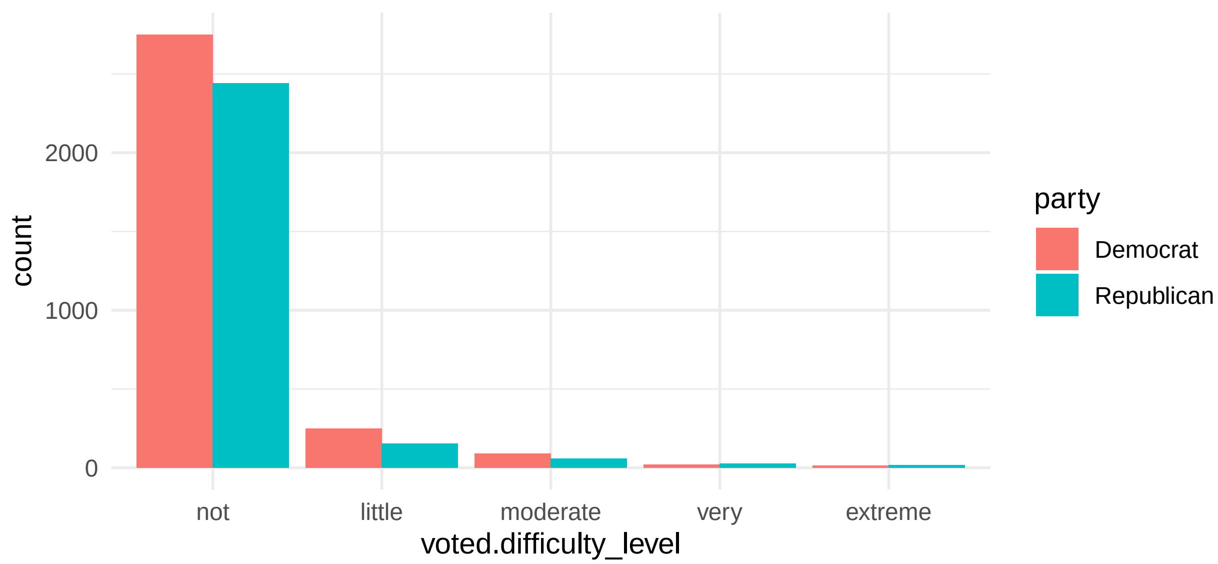

Figure 4: Eliminating NA values

4.5 Color matters

Let's choose better colors for the charts. Following the current custom of using red for Republicans and blue for Democrats seems reasonable. See Figure 5.

```
df %>% drop_na(voted.difficulty_level) %>%
  ggplot() +
  aes (x=voted.difficulty_level, fill=party) +
  geom_bar(position="dodge") +
  scale_fill_manual(values = c("blue", "red"))
```

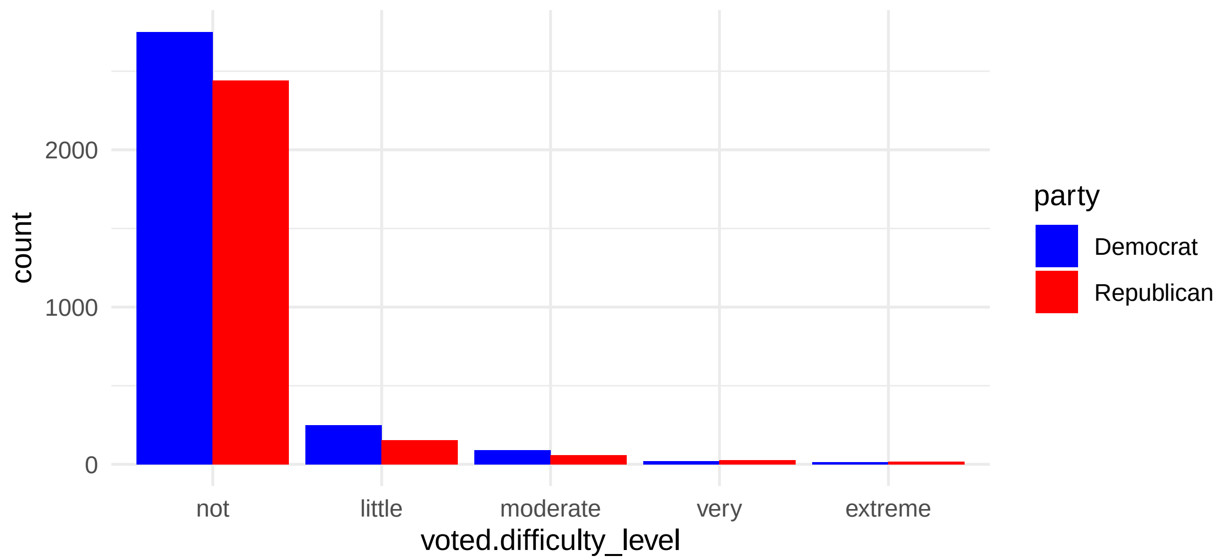

Figure 5: Changing the colors

4.6 Titles and axis labels

Let's give the chart a title and better axis labels. This chart makes it evident that as a general matter, voters did not have difficulty voting. We can get that concept into the title and use more descriptive axis labels. See Figure 6.

```
df %>% drop_na(voted.difficulty_level) %>%
  ggplot() +
  aes (x=voted.difficulty_level, fill=party) +
  geom_bar(position="dodge") +
  scale_fill_manual(values = c("blue", "red")) +
  labs (title="Voting was not difficult for most",
        y="voters",
        x="difficulty")
```

4.7 Fine tuning

It would be nice to have a tick mark on the y axis above the data to set a ceiling. We use `coord_cartesian` to do that. See Figure 7. At this point, let's save the plot as `final_chart_1`.

```
final_chart_1 <- df %>% drop_na(voted.difficulty_level) %>%
  ggplot() +
  aes (x=voted.difficulty_level, fill=party) +
  geom_bar(position="dodge") +
  scale_fill_manual(values = c("blue", "red")) +
  coord_cartesian(ylim = c(0, 3000)) +
  labs (title="Voting was not difficult for most",
        y="voters",
        x="difficulty")

final_chart_1
```

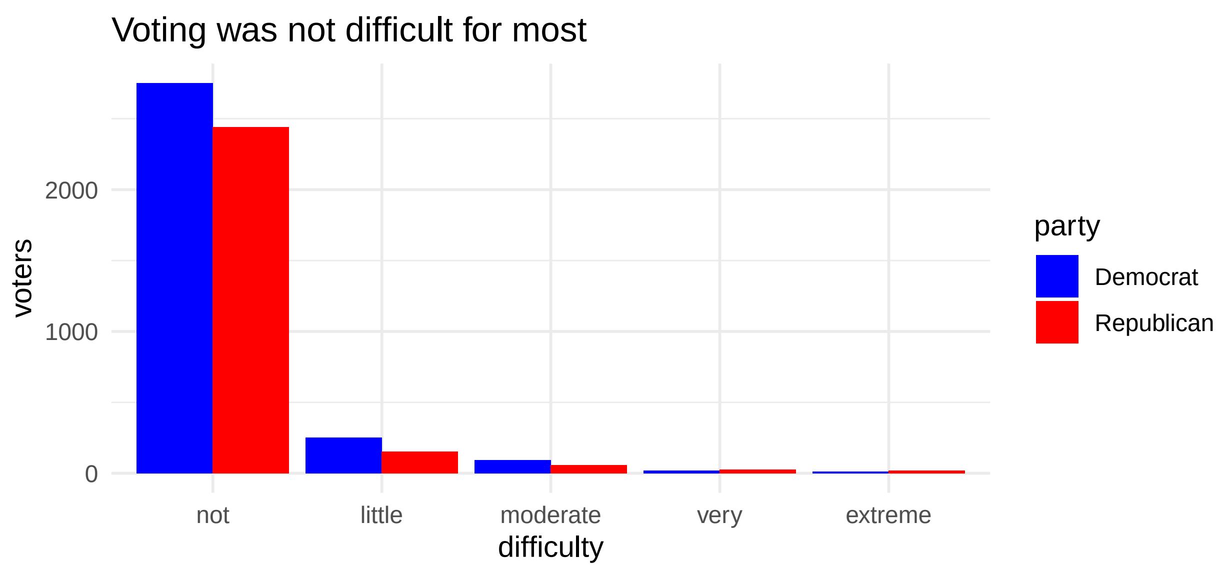

Figure 6: Title and labels

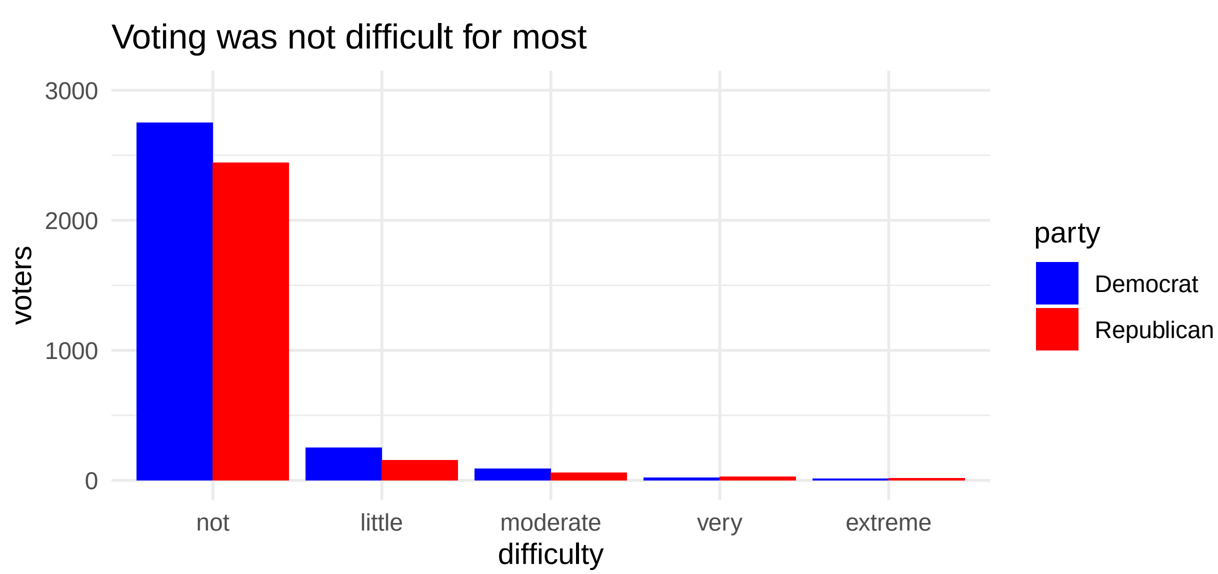

Figure 7: Fine tuning

5 But what does our data look like when voting is difficult?

The bar charts are dominated by that first bar. The rest of the data is obscured. What should we do to tease out the obscured detail?

5.1 Eliminate the dominant column to focus on what we care about

We want to show what happens when people encounter difficulty voting. We have already established that the data shows that voting is usually not difficult. So, we set aside data for people who had no difficulty in order to see what remains.

Notice that in the code below we filter for rows where `voted.difficulty_level > "not"`. That expression works because the column is an ordered factor in `R`. When the data frame was created, the column was coerced into an ordered factor with `as.ordered`.

```
df %>% drop_na(voter.difficulty_level) %>% filter(voted.difficulty_level > "not") %>%
  ggplot() +
  aes (x=voted.difficulty_level, fill=party) +
  geom_bar(position="dodge") +
  scale_fill_manual(values = c("blue", "red")) +
  labs (title="Focus on when voting is difficult",
        y="voters",
        x="difficulty")
```

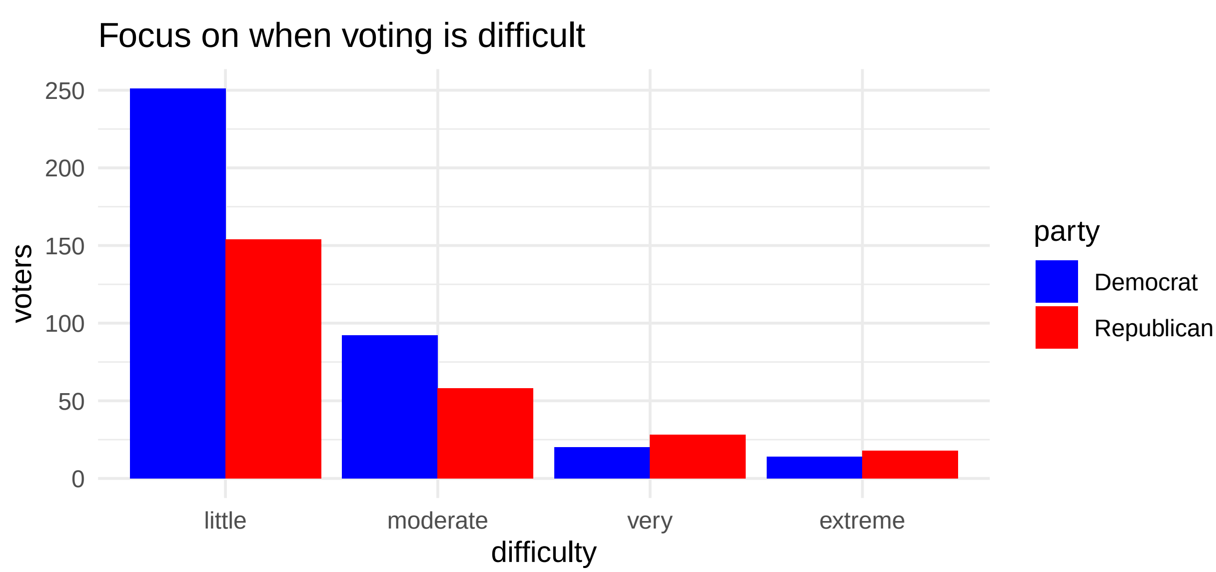

Figure 8: Only pick the categories we care about

5.2 But we don't have the same number of Republicans and Democrats

This chart is accurate, but I think it is misleading. Remember, when we looked at our data earlier we saw that there is a difference in the number of Republicans and Democrats included. To be precise, there are 434 more Democrats than Republicans. So, just looking at the raw numbers doesn't seem right.

One useful way to think about this is to compare the percentage of Republicans in our data set with a given voting difficulty level to the percentage of Democrats in the data set with that same voting difficulty level.

It is unduly difficult to get *ggplot* to generate these percentages on its own. So, rather than attempting to force *ggplot* to do the transformation from counts to the percentages we want, it's easier to calculate those percentages outside of *ggplot*, put the results in a data frame, and use a slightly different *geom* to produce the chart. The `geom_col` geom is a close relative of the `geom_bar` geom. It lets you specify the y values for the x axis data.

First we calculate the population percentages by group for each of the difficulty levels, then we use the resulting table to make a bar chart.

```
percentages <- df %>%
  drop_na(voted.difficulty_level) %>%
  group_by(party, voted.difficulty_level) %>%
  summarise(count = n()) %>%
  mutate(freq = (count/sum(count)))

(percentages)
```

```
## # A tibble: 10 x 4
## # Groups:   party [2]
##    party      voted.difficulty_level count    freq
##    <fct>      <ord>                  <int>   <dbl>
##  1 Democrat   not                     2751 0.879
##  2 Democrat   little                   251 0.0802
##  3 Democrat   moderate                  92 0.0294
##  4 Democrat   very                      20 0.00639
##  5 Democrat   extreme                   14 0.00448
##  6 Republican not                     2442 0.904
##  7 Republican little                   154 0.0570
##  8 Republican moderate                  58 0.0215
##  9 Republican very                      28 0.0104
## 10 Republican extreme                   18 0.00667
```

Now, we can use the `freq` column of the `percentages` table to produce what we really want. See Figure 9.

```
final_chart_2 <- percentages %>%
  ggplot() +
  aes (x=voted.difficulty_level, y=freq, fill=party) +
  geom_col(na.rm = TRUE, position="dodge") +
  scale_fill_manual(values = c("blue", "red")) +
  scale_x_discrete(limits = c("little", "moderate", "very", "extreme")) +
  scale_y_continuous(labels = scales::percent) +
  coord_cartesian(ylim = c(0, .08)) +
  labs (title = "Focusing on when voting is difficult",
        y = "party percentage",
        x = "difficulty")

final_chart_2
```

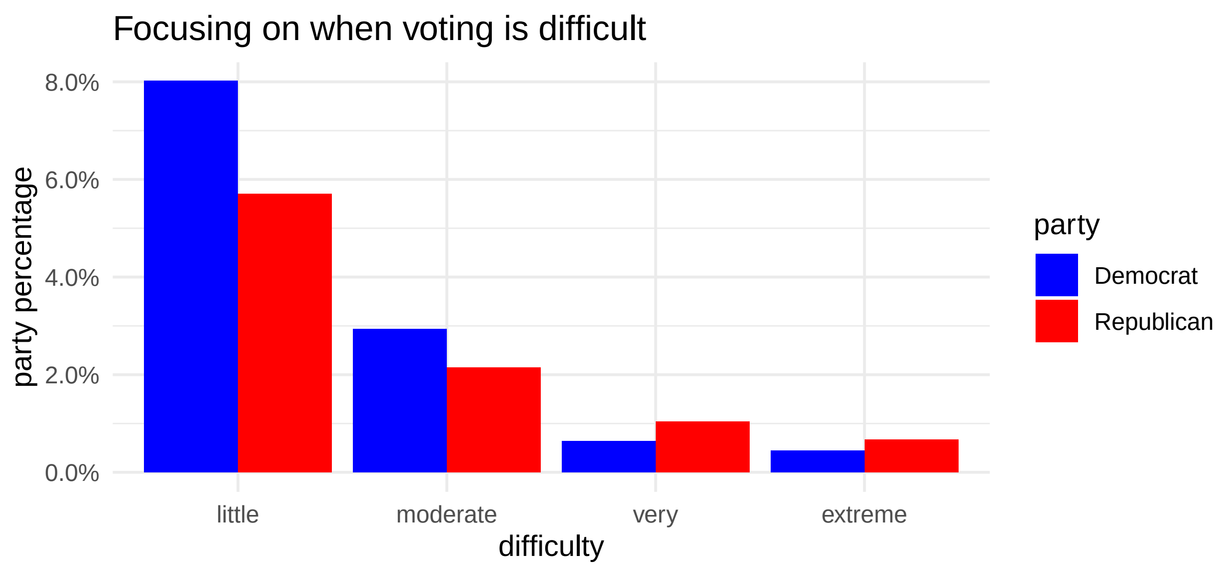

Figure 9: Difficulty levels by party and party percentage

6 Putting things together

6.1 A first attempt with patchwork

We can use `patchwork` to arrange multi-chart displays. We loaded that package when we started. Using it here is really simple, it's that `+` operator below. See Figure 10.

```
final_chart_1 + final_chart_2
```

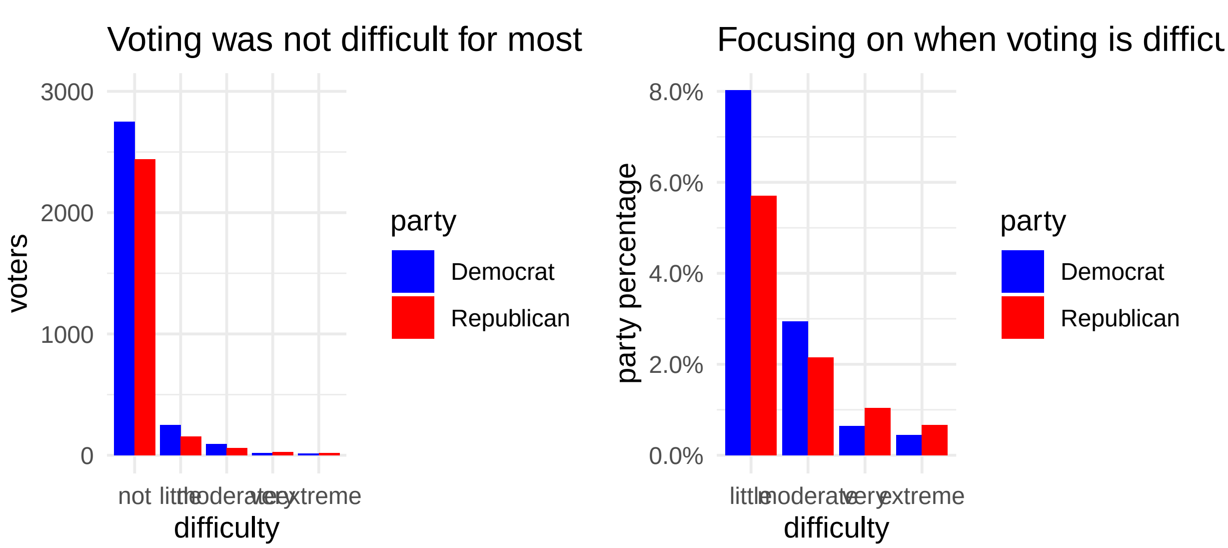

Figure 10: Side by side charts, first attempt

6.2 Let's clean it up

That last visualization has two problems. First, it's too tight across the page. Second, we really don't need the legend to appear twice. We can adjust the figure width and remove the legend from the left hand panel. See Figure 11.

The code fragment below illustrates a technique we could have used throughout. For most of this document, we have created every chart with a complete expression. This time, we derive a new plot (`final_chart_1a`) from an existing plot (`final_chart_1`) by adding a new *ggplot* element.

```
final_chart_1a <- final_chart_1 + theme(legend.position="none")
final_chart_1a + final_chart_2
```

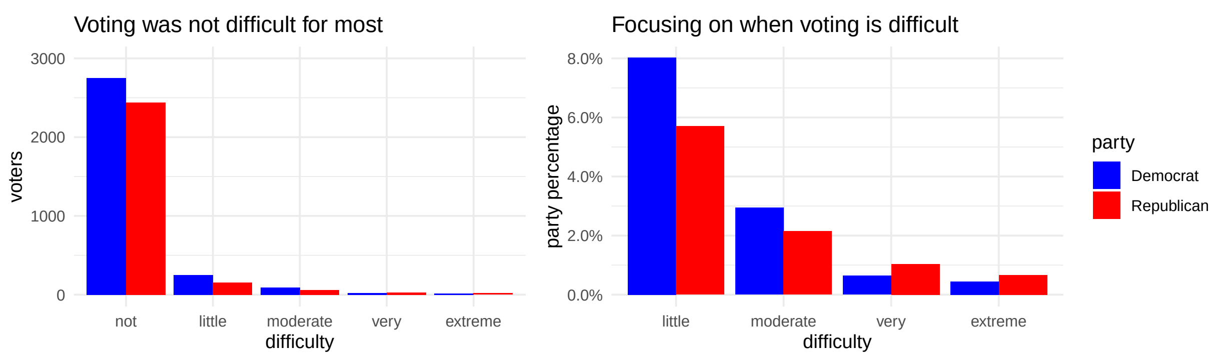

Figure 11: Putting it all together

7 What do we know about people who tried to vote but failed?

Our data includes the concept of a *presumed voter*. In this data, someone is a presumed voter if we believe they attempted to vote but faced an extreme difficulty that blocked their attempt outright. Information about these people is not captured by the `voted.difficulty_level` data (it is included in the `voter.difficulty_data` variable). Information about what kept the presumed voters from voting is captured by the `presumed_reason` variable. Here is a list of reasons people could give for not voting. Only some of the reasons are sufficient to warrant an inference that the person intended to vote.

```
levels(df$presumed.reason)
```

```
##  [1] "forgot"                            "not interested"
##  [3] "too busy"                          "candidates"
##  [5] "not registered"                    "lacked correct id"
##  [7] "out of town"                       "sick or disabled"
##  [9] "transportation"                    "bad weather"
## [11] "poll line too long"                "denied at polls"
## [13] "absentee ballot problem"           "did not know where to vote"
## [15] "lacked information about choices" "other"
```

Our goal is to create a visualization to get a better sense of this data.

7.1 A pretty good first attempt

Since this data is only captured for people who failed to vote and our data is dominated by information about people who voted, we expect a lot of `NA` values.

```
sum(is.na(df$presumed.reason))
```

```
## [1] 5874
```

Rather than filter that out before we run *ggplot*, let's show how we can filter out that data inside *ggplot*. We do this just to illustrate the technique. (I think that filtering the data earlier in the pipe before calling *ggplot* is cleaner, but you might disagree.) The key is the use of `na.rm = TRUE` in `geom_bar` coupled with the call to `scale_x_discrete` below. See Figure 12.

```
df %>%
  ggplot() +
  aes (x=presumed.reason, fill=party) +
  geom_bar(na.rm = TRUE, position="dodge") +
  scale_x_discrete(na.translate = FALSE) +
  scale_fill_manual(values = c("blue", "red")) +
  labs (title = "Voting impediments expressed by presumed voters",
        y = "voters",
        x = "reason")
```

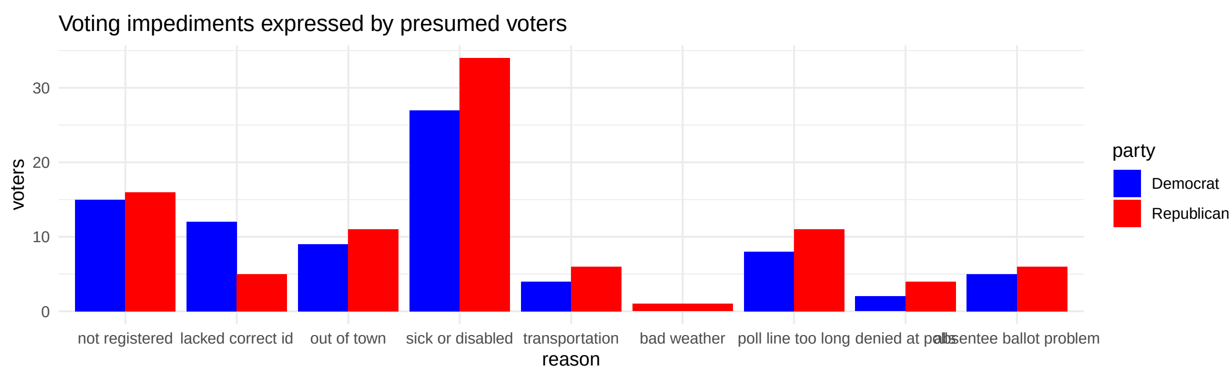

Figure 12: Why some people didn't vote

7.2 A little bit of polish

There are two small adjustments we make to Figure 12. The reason labels are too compressed, especially when rendered in a pdf, so we rotate them. Also, I do not think the x axis label adds information given the balance of chart. So, we make that label blank using `element_blank()`. Finally, when we rotate the x axis labels, that will tend to compress the figure since we are going to use more room at the bottom for those labels which now extend further down than before. To adjust for that I set the figure height to 4 to override the default of 3 that we have used throughout. The code for that adjustment appears in the header to the `R` code chunk for this chart and can be seen in the Rmd file but does not appear in the pdf. See Figure 13.

```
df %>%
  ggplot() +
  aes (x=presumed.reason, fill=party) +
  geom_bar(na.rm = TRUE, position="dodge") +
  scale_x_discrete(na.translate = FALSE) +
  scale_fill_manual(values = c("blue", "red")) +
  labs (title = "Voting impediments expressed by presumed voters",
        y = "voters",
        x = element_blank()) +
  theme(axis.text.x = element_text(angle = 90, vjust=0.5, hjust=1))
```

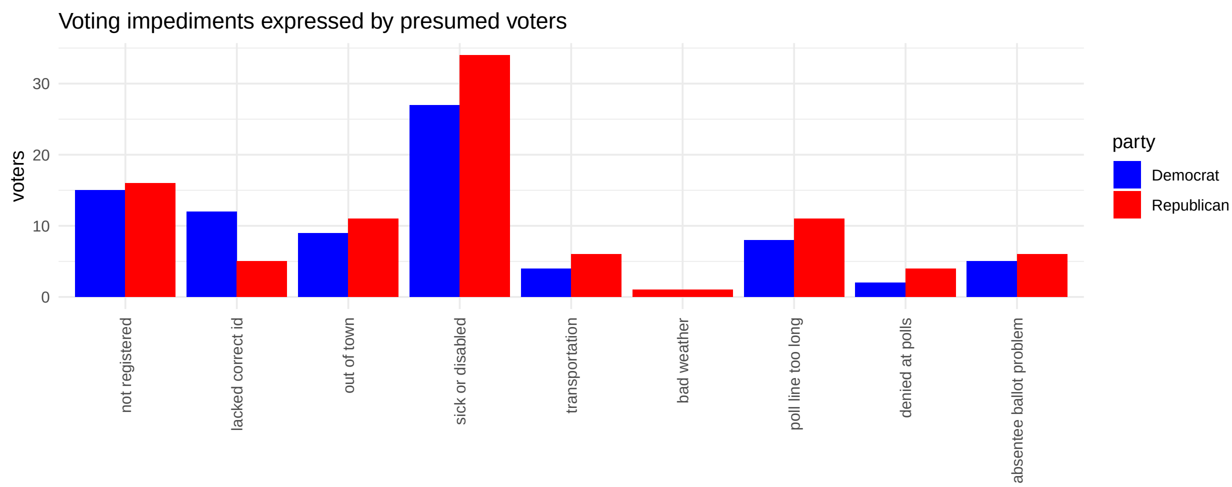

Figure 13: Why some people didn't vote (revised)

7.3 Reorder the x axis

Now lets reorder the x axis. See Figure 14.

```
library(forcats)

df %>%
  ggplot() +
  aes (x = fct_infreq(presumed.reason), fill=party) +
  geom_bar(na.rm=TRUE, position = "dodge") +
  scale_x_discrete(na.translate = FALSE) +
  scale_fill_manual(values = c("blue", "red")) +
  labs (title = "Voting impediments expressed by presumed voters",
        y = "voters",
        x = element_blank()) +
  theme(axis.text.x = element_text(angle = 90, vjust=0.5, hjust=1))
```

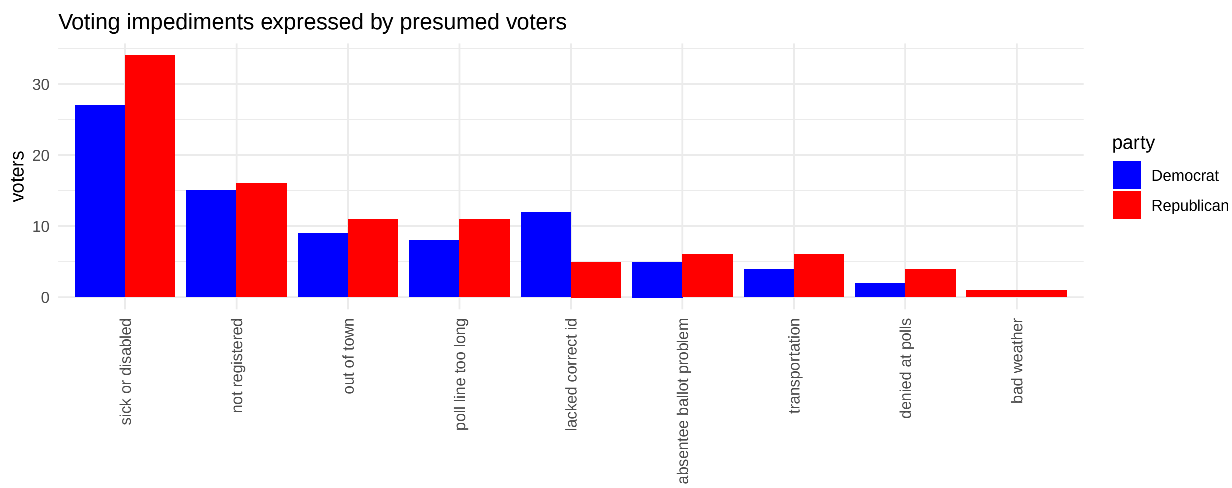

Figure 14: Reorder the X axis