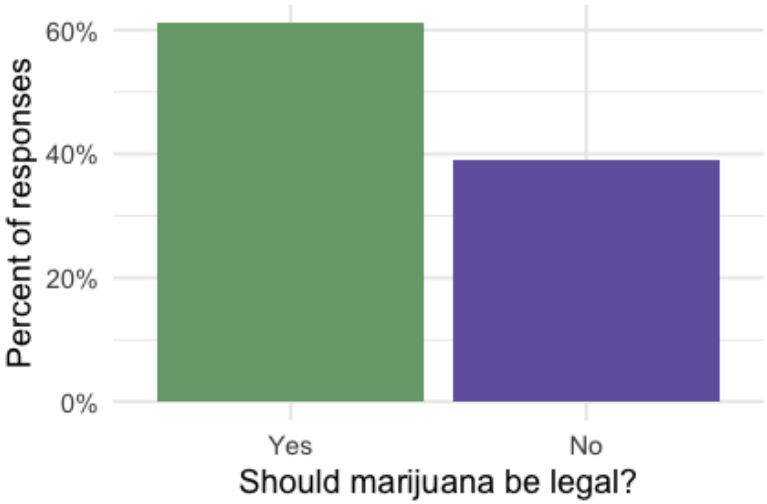
Preparing data for analysis and visualization in R

Building a basic bar graph

Jenine Harris
Brown School

Building a basic bar graph

- Follow the steps to build this graph of marijuana use:



- Notice some of the important features of the graph:
  - Both axes have titles
  - Y-axis is a percentage
  - X-axis is labeled as Yes and No
  - The background for the graph is white with a light gray grid
  - The Yes bar is green, the No bar is purple

Building a basic bar graph

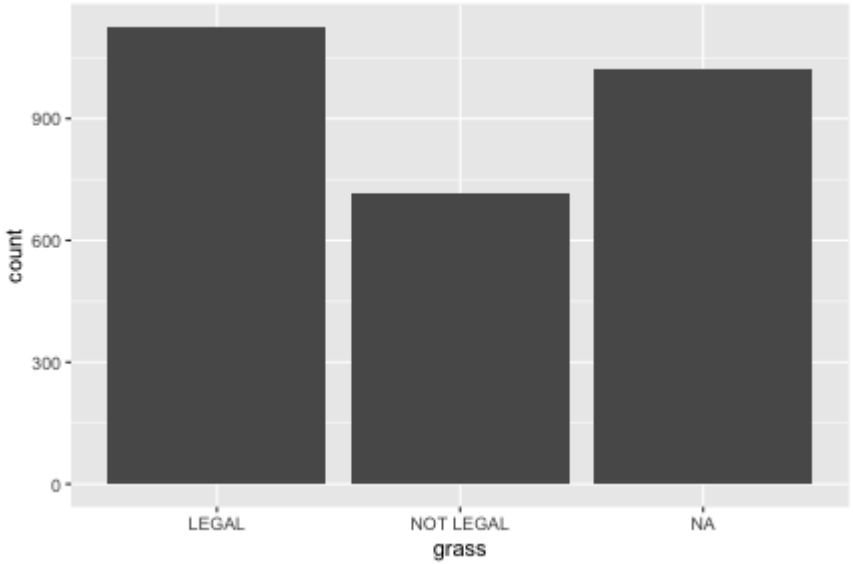
- Import and clean the data and open the tidyverse:

```
# bring in GSS 2016 data
library(package = "data.table")
gss.2016 <- fread(file = "data/legal_weed_age_GSS2016_ch1.csv")

# open the tidyverse
library(package = "tidyverse")

# clean the data
gss.2016.cleaned <- gss.2016 %>%
  mutate(age = recode(age, "89 OR OLDER" = "89")) %>%
  mutate(age = as.numeric(x = age)) %>%
  mutate(grass = as.factor(x = grass)) %>%
  mutate(grass = na_if(x = grass, y = "DK")) %>%
  mutate(grass = na_if(x = grass, y = "IAP")) %>%
  mutate(grass = droplevels(x = grass)) %>%
  mutate(age.cat = cut(x = age, breaks = c(-Inf, 29, 59, 74, Inf),
                       labels = c("< 30", "30 - 59",
                                  "60 - 74", "75+" )))
```

Building a basic bar graph

- Start with a basic plot using the `ggplot()` function from the tidyverse.

```
# make a bar graph for grass variable
legalize.bar <- gss.2016.cleaned %>%
  ggplot(aes(x = grass)) +
  geom_bar()

# show the graph
legalize.bar
```



Building a basic bar graph

```
legalize.bar <- gss.2016.cleaned %>%
  ggplot(aes(x = grass)) +
  geom_bar()
```

- `legalize.bar` is the object name of the graph
- `gss.2016.cleaned` is the clean data
- `aes()` stands for aesthetics and is where the main variables for the graph are specified
- `x =` specifies the variable for the x-axis, in this case `grass`
- `geom_bar()` is the type of geometry the graph should have, in this case `bar` for bar graph

Building a basic bar graph

- graphs in `ggplot()` are built in layers, the first layer has the **aesthetics** or `aes()`
- the second layer, for this graph, chooses the graph type with `geom_bar()`
- Note that there is no `%>%` between the layers of the `ggplot()`; this is because it is all one function

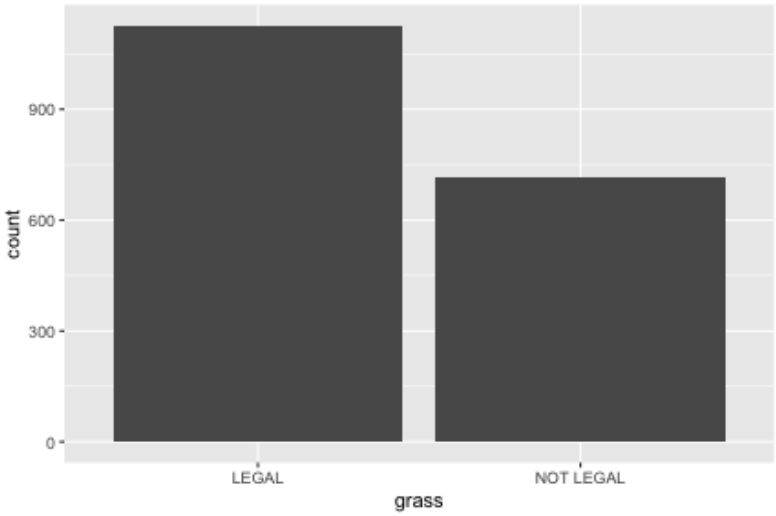
Omitting NA from a graph

- The easiest way to omit `NA` from a bar graph is adding `drop_na()` to the set of commands.
- In this case, the `NA` should be dropped from the grass variable.
- To drop the `NA` values before the graph, add `drop_na()` above `ggplot()` in the code.

```
legalize.bar <- gss.2016.cleaned %>%
  drop_na(grass) %>%
  ggplot(aes(x = grass)) +
  geom_bar()
```

Omitting NA from a graph

```
# show the graph
legalize.bar
```



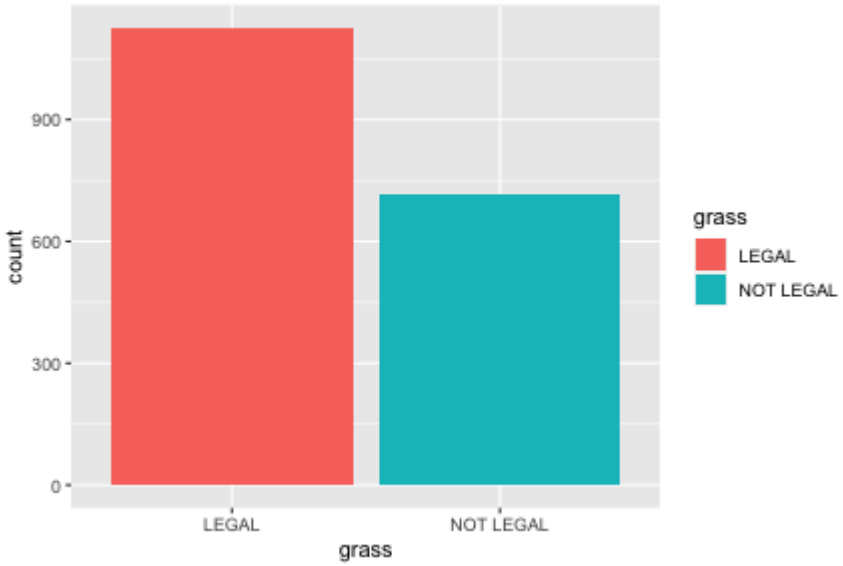
Working with color in a bar graph

- Using the aesthetic option `fill =` will fill the bars with color based on whatever variable is assigned as the argument for `fill =`
- Use the `grass` variable with `fill =` to fill the bars based on which category of `grass` each bar represents.

```
legalize.bar <- gss.2016.cleaned %>%
  drop_na(grass) %>%
  ggplot(aes(x = grass, fill = grass)) +
  geom_bar()
```

Working with color in a bar graph

```
# show the graph
legalize.bar
```



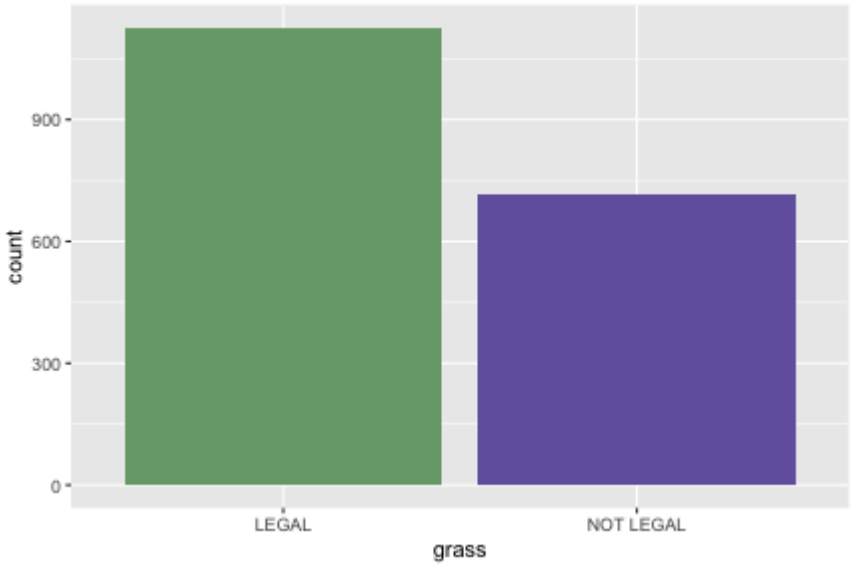
Working with color in a bar graph

- The bars have color now, but the colors are not the same as the initial graph.
- There is a legend to the right of the graph that was redundant with the x-axis.
- Adding a new layer called `scale_fill_manual()` layer allows selection of colors (as `values =`) for whatever variable is included in the `fill =` command and also has a `guide =` option to specify whether the legend appears or not.

```
legalize.bar <- gss.2016.cleaned %>%
  drop_na(grass) %>%
  ggplot(aes(x = grass, fill = grass)) +
  geom_bar() +
  scale_fill_manual(values = c('#78A678', '#7463AC'),
                    guide = FALSE)
```

Working with color in a bar graph

```
# show the graph
legalize.bar
```



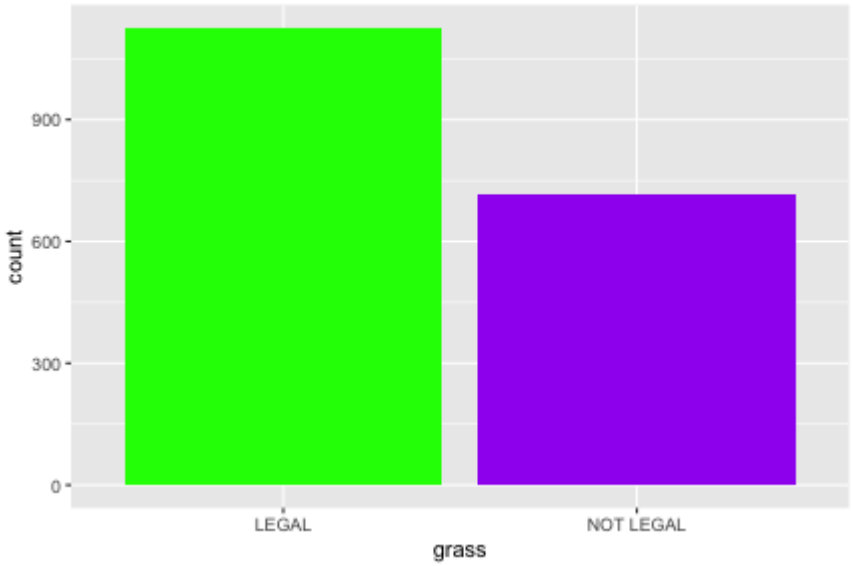
Working with color in a bar graph

- The values `78A678` and `7463AC` are **RGB** or **red-green-blue** codes that specify colors.
- Use the Color Brewer 2.0 website to find RGB codes for colors that work well for different sorts of graphs and are colorblind safe and work with printing or copying.
- The names of colors can also be used, for example, try replacing the codes with the words "green" and "purple."

```
legalize.bar <- gss.2016.cleaned %>%
  drop_na(grass) %>%
  ggplot(aes(x = grass, fill = grass)) +
  geom_bar() +
  scale_fill_manual(values = c('green', 'purple'),
                    guide = FALSE)
```

Working with color in a bar graph

```
# show the graph
legalize.bar
```



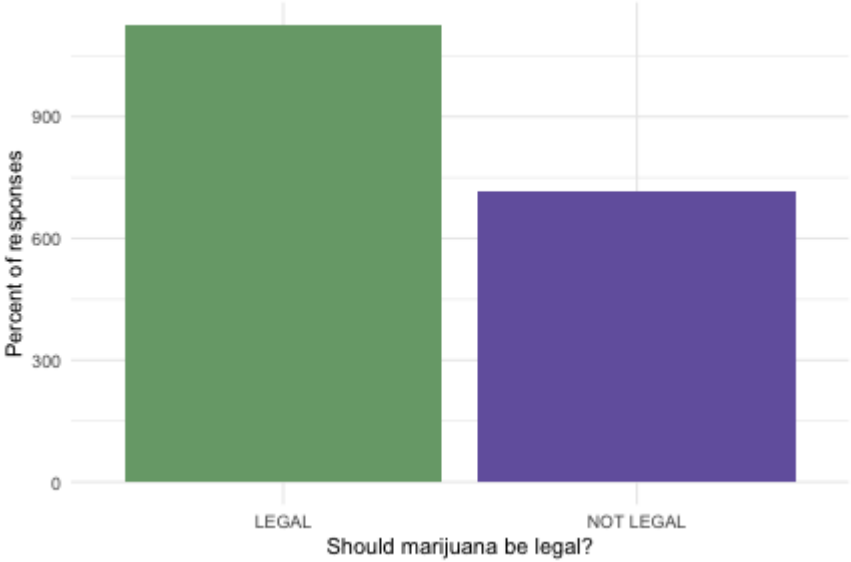
Working with color in a bar graph

- Next remove the gray background by adding another layer specifying a theme; to remove the gray background us `theme_minimal()`.
    - To see all the theme options explore https://ggplot2.tidyverse.org/reference/ggtheme.html
- Another layer can be added for the labels can be added using the `labs()` function with text entered for `x =` and `y =`.

```
legalize.bar <- gss.2016.cleaned %>%
  drop_na(grass) %>%
  ggplot(aes(x = grass, fill = grass)) +
  geom_bar() +
  scale_fill_manual(values = c('#78A678', '#7463AC'),
                    guide = FALSE) +
  theme_minimal() +
  labs(x = 'Should marijuana be legal?',
       y = 'Percent of responses')
```

Working with color in a bar graph

```
# show the graph
legalize.bar
```



- It looks great but the y-axis should be a percentage and the labels on the x-axis should be Yes and No.

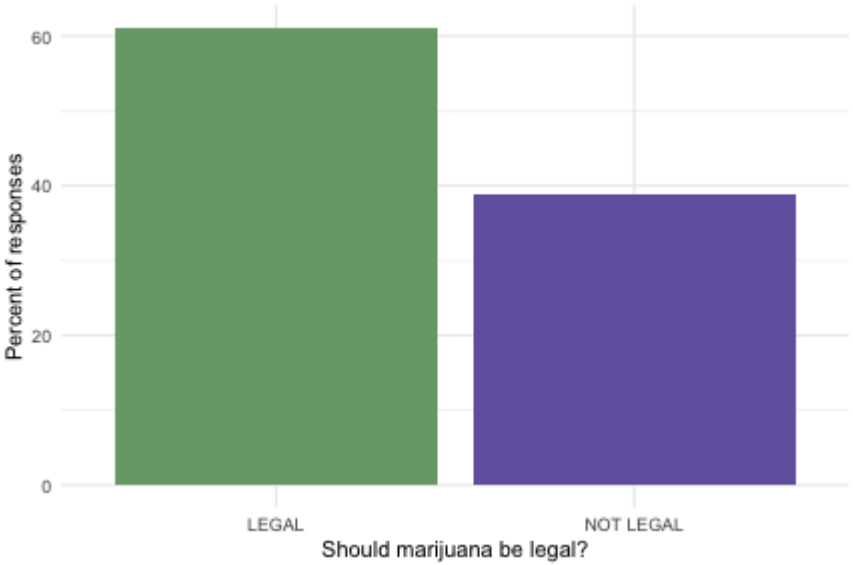
Using special variables in graphs

- One way to get the y-axis to show percentage rather than count is to use **special variables** with double periods around them.
  - Special variables are statistics computed from a data set; the *count* special variable counts the number of observations in the data set.
  - `..count..` to represent the frequency of a category and `sum(..count..)` to represent the sum of all the frequencies.
  - Multiply by 100 to get a percentage.

```
legalize.bar <- gss.2016.cleaned %>%
  drop_na(grass) %>%
  ggplot(aes(x = grass,
  y = 100*(..count..)/sum(..count..),
  fill = grass)) +
  geom_bar() +
  scale_fill_manual(values = c('#78A678', '#7463AC'),
                    guide = FALSE) +
  theme_minimal() +
  labs(x = 'Should marijuana be legal?',
       y = 'Percent of responses')
```

Using special variables in graphs

```
# show the graph
legalize.bar
```



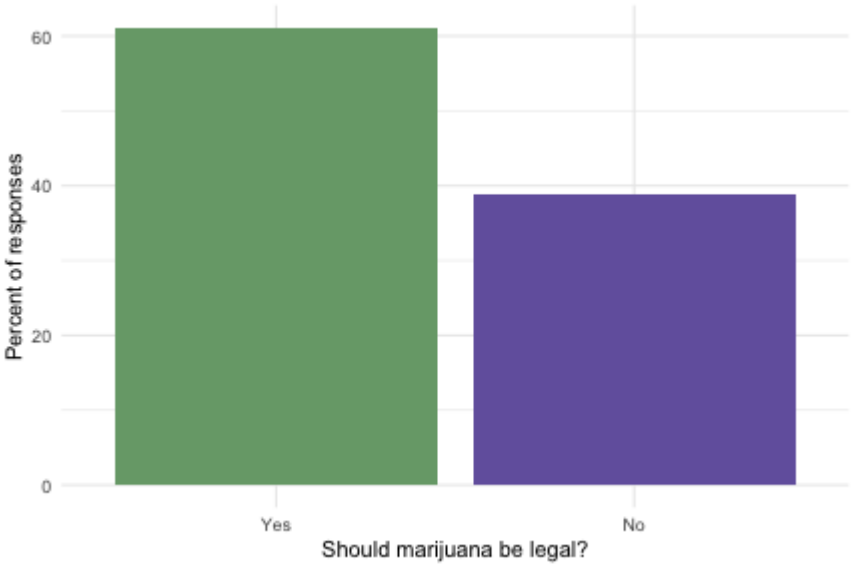
Using special variables in graphs

- The last thing to do is to recode the levels of the grass variable to be Yes and No.
- The final code needed is `mutate()` and `recode_factor()` to recode `LEGAL` and `NOT LEGAL` to `Yes` and `No`

```
legalize.bar <- gss.2016.cleaned %>%
  drop_na(grass) %>%
  mutate(grass = recode_factor(.x = grass,
                               LEGAL = 'Yes',
                               NOT LEGAL = 'No')) %>%
  ggplot(aes(x = grass,
             y = 100*(..count..)/sum(..count..),
             fill = grass)) +
  geom_bar() +
  theme_minimal() +
  scale_fill_manual(values = c('#78A678', '#7463AC'),
                    guide = FALSE) +
  labs(x = 'Should marijuana be legal?',
       y = 'Percent of responses')
```

Using special variables in graphs

```
# show the graph
legalize.bar
```



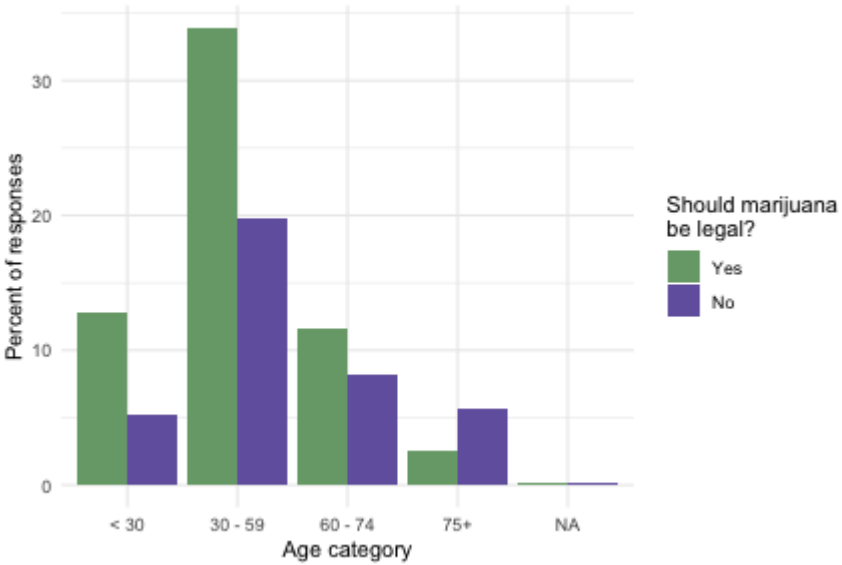
Using special variables in graphs

- One more trick can be used to add the `age.cat` variable into the graphic.
  - Change the x-axis variable in the aesthetics to be `x = age.cat`
  - Remove the `guide = FALSE` from the `scale_fill_manual()` layer
  - Change the x-axis label
  - Added `position = 'dodge'` in the `geom_bar()` layer. The `position = 'dodge'` makes the bars for Yes and No in each age category show side-by-side (Figure \@ref(fig:c1main60)).

```
legalize.bar <- gss.2016.cleaned %>%
  drop_na(grass) %>%
  mutate(grass = recode_factor(.x = grass,
                               LEGAL = 'Yes',
                               NOT LEGAL = 'No')) %>%
  ggplot(aes(x = age.cat,
             y = 100*(..count..)/sum(..count..),
             fill = grass)) +
  geom_bar(position = 'dodge') +
  theme_minimal() +
  scale_fill_manual(values = c('#78A678', '#7463AC'),
                    name = 'Should marijuana be legal?') +
  labs(x = 'Age category',
       y = 'Percent of responses')
```

Using special variables in graphs

```
# show graph
legalize.bar
```



Check your understanding

- Think about the number of missing values for `grass` and `age` in the `gss.2016.cleaned` data frame. Run the `summary()` function to confirm. If 1,824 of the 2,867 observations had values for both `grass` and `age`, 10 observations were missing `age`, and 1024 observations were missing `grass`. How many observations were missing values for both?

- Try some visual changes to the graph.

    - Change the *No* bar to the purple color *#88398a* and change the *Yes* bar to the color *gray80*.
    - Change the theme to another theme from the themes available online.

Answer (one of many possibilities)

```
# adding dodge to show bars side-by-side, remove legend
legalize.bar <- gss.2016.cleaned %>%
  drop_na(grass) %>%
  mutate(grass = recode_factor(.x = grass,
                               `LEGAL` = 'Yes',
                               `NOT LEGAL` = 'No')) %>%
  ggplot(aes(x = age.cat,
             y = 100*(..count..)/sum(..count..),
             fill = grass)) +
  geom_bar(position = 'dodge') +
  theme_classic() +
  scale_fill_manual(values = c('gray80', '#88398a'),
                    name = 'Should marijuana\nbe legal?') +
  labs(x = 'Age category',
       y = 'Percent of responses')

# show graph
legalize.bar
```

Answer (one of many possiblities)