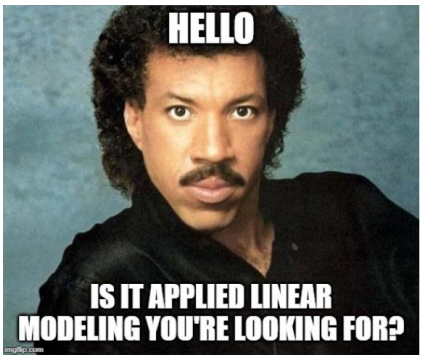
# Week 1

## Welcome to Applied Linear Modeling!

Jenine Harris

August 27, 2019




## All the things for today

- The requisite icebreaker activity
- Get-to-know-the-syllabus discussion
- What is Applied Linear Modeling anyway?
- Intro to R, RStudio, R Markdown
- Data management workshop


## Icebreaker: An ode to your classmate

- Find one or two people to talk to…try someone you don't already know
- Spend 10 minutes learning about each other (take notes!)
  - *Be sure to ask them if they prefer to be named or not named in the ode*
- Spend 5 minutes to use what you learned to write an ode to one of the people you talked with
  - *If you talked with more than one person, make sure everyone in the group has an ode!*
- Share the ode with the person it is about
- Make revisions if needed and tape to the front board

# Ode structure

**(from poet Danielle Pafunda)**

**Ode to_____**

*one word describing the subject*
*one word describing the subject*
*fact about the subject*
*wild card line* (imagine your subject speaking or acting or speak to your subject)

***Example:***

**Ode to Nancy**

*mom*
*brilliant*
*autism researcher*
*how far did you run today?*

# Ode instructions

- Find one or two people to talk to…try someone you don't already know
- Spend 10 minutes learning about each other (take notes!)
- Spend 5 minutes to use what you learned to write an ode to one of the people you talked with
- Share the ode with the person it is about
- Make revisions if needed and tape to the front board
- Ode template:

  ***Ode to*_____**

  *one word describing the subject*
  *one word describing the subject*
  *fact about the subject*
  *wild card line (imagine your subject speaking or acting or speak to your subject)*

# A typical week in ALM

- Discussion of exercises from prior week
- New topic workshop

# Get-to-know-the-syllabus discussion activity

- Most weeks we will start the day by going over the exercises from the prior week together
  - *Participating in this is the majority of your participation/professionalism points*
- Let's practice the *process* using the syllabus
  - *Work on your own or with classmate(s) to complete the questions about the syllabus*
  - *Pick a number from the* ***to-do*** *jar and write the answer on the board*
    - If you want to work with someone, pick two numbers from the **to-do** jar and work together to write both answers on the board
  - *Write your name(s) on the number(s) you picked and drop in the* ***done*** *jar*

# New topic workshop

- Most weeks materials for the workshop will be saved on GitHub by Monday at noon
  - *https://github.com/jenineharris/applied-linear-modeling*
- Recommended strategy for organizing weekly files:
  - *Make an overall class folder on your laptop called alm-2019 or something similar*
  - *Within this folder, keep a folder for each week of class (e.g., week-1-aug27)*
    - Download all the files for the week into the weekly folder before class
- Go ahead and set up your folders and download today's files and save them **both in the same folder**
  - *week-1-workshop.Rmd*
  - *legal_weed_age_GSS2016_ch1.csv*

# What is Applied Linear Modeling anyway?

- Applied linear models are approaches to understanding relationships among variables
  - *Explaining or predicting something*
  - *Exploratory or confirmatory*
- Two kinds of applied linear models
  - *General linear models*
  - *Generalized linear models*

# General linear models

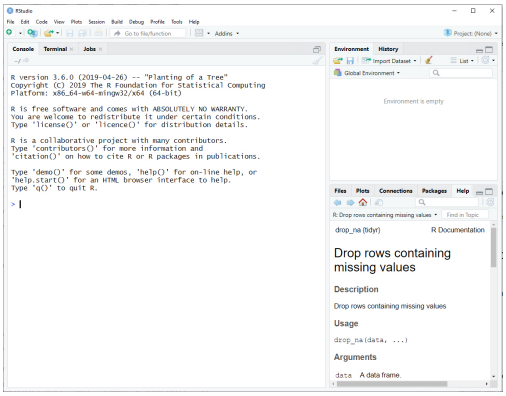
- General linear models have an assumption of normality, which requires a continuous outcome variable
- General linear models we will cover
  - *t-test (in review only)*
  - *ANOVA*
  - *linear regression*

# Generalized linear models

- Generalized linear models do not assume normality, so the outcome variable can be categorical
  - *These models transform the outcome variable to use principles from general linear models*
- Generalized linear models we will cover
  - *binary logistic regression*

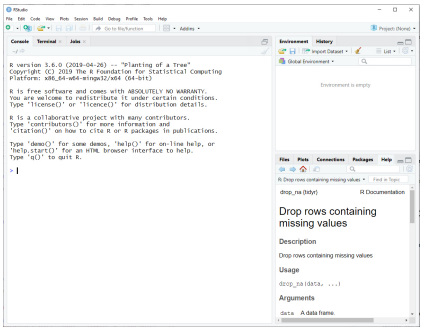
# Intro to R, RStudio, RMarkdown

- R Studio is an IDE or Interactive Development Environment
- Open RStudio to see this screen

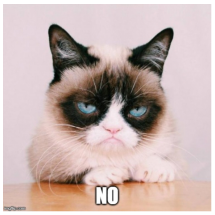



# RStudio is a pane

- The RStudio window has 3 **panes** as a default
  - *Top right pane has the Environment and History tabs*
  - *Bottom right pane has the Help and Plots tabs (and others)*
  - *Left pane is the Console where you can write code and see output*



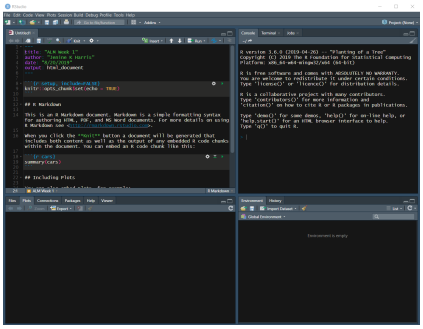

# Customizing your RStudio window


# Options for changing your work space

- Usually you will have a fourth pane in the top left that is a code editor
  - *File -> Open File -> [week-1-workshop.Rmd]*
- Customize how you see your R code
  - *Tools -> Global Options… -> Appearance*
- Customize the layout of the panes
  - *Tools -> Global Options… -> Pane Layout*

## Modified work space




## Data cleaning and management workshop (with a little graphing!)

- Importing data into R
- Checking and correcting data types
- Making sure missing values are handled correctly
- Recoding variables
- Graphing

## The goal!

We will work to import and clean a data set and then graph the data so that we can answer the research question:

**What is the level of support for marijuana legalization in the US? How might it change in the next 20 years?**

## The pot policy problem

- Marijuana use remains illegal under federal law
- By 2017, 29 states and the District of Columbia had legalized marijuana at the state level for medical or recreational use or both
- With new ballot measures at the state level being introduced and passed by voters on a regular basis, there appears to be momentum for a nationwide shift toward legalization

# Support for legalization

- The 2016 General Social Survey (GSS) included a question about legalization:

  *Do you think the use of marijuana should be made legal or not?*

- It also included a question about age, which might be useful in determining how support might change as people age
- Use the GSS Data Explorer at https://gssdataexplorer.norc.org/variables/vfilter to learn about the variables
- Let's import and clean up the data to try and answer the question

# Importing a data set into R

- Import a data file from the GSS that contains the grass variable and the age variable:

```
# read the GSS 2016 data
gss.2016 <- read.csv(file = "legal_weed_age_GSS2016_ch1.csv")
```

# Examine the contents of the data file

```
# examine the contents of the file
summary(object = gss.2016)
```

```
##       ï..grass         age       
##  DK       : 110   57     :  70  
##  IAP      : 911   58     :  67  
##  LEGAL    :1126   52     :  65  
##  NOT LEGAL: 717   53     :  60  
##  NA's     :   3   27     :  58  
##                   (Other):2537  
##                   NA's   :  10
```

What do you think is going on here?

# Another way to import csv data

- We can try using a different function to open the file
- The `fread()` function in the `data.table` package is useful for opening csv (comma separated values) files
- To use a function from the `data.table` package, install the package first
- Once the package is installed, open it using the `library()` function

```
# open the data.table package
library(package = "data.table")
```

# Using fread to import data

- Use the `fread()` function from `data.table` to open the file again:

```
# open the GSS data with fread
gss.2016 <- fread(input = "legal_weed_age_GSS2016_ch1.csv")
```

# Try the summary function again to see if it looks better

```
# examine the contents of the file
summary(object = gss.2016)
```

```
##     grass              age
##  Length:2867        Length:2867
##  Class :character   Class :character
##  Mode  :character   Mode  :character
```

# Checking and correcting the data types of variables

- R has many different data types, the three most useful for this course are:
  - *Numeric: continuous variables (sometimes called "double" in R)*
  - *Factor: categorical variables*
  - *Character/string: text variables*
- Many R functions can only use data that are the right type, so checking and correcting data types is an important data cleaning step

# Check data types using the class function

- The class function is useful for determining what data type a variable is
- Use the class function with the two variables in the `gss.2016` data set

```
# what data type is grass variable
class(x = gss.2016$grass)
```

```
## [1] "character"
```

```
# what data type is age variable
class(x = gss.2016$age)
```

```
## [1] "character"
```

# Fixing the data types

- Based on the codebook, `grass` looks categorical so it should be a **factor** in R
- Based on the codebook, `age` looks closer to continuous so it should be **numeric** in R
- We will be using the `tidyverse` package for data cleaning and management, so install it through the Tools menu and open it with the library function

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

# Our first data cleaning task!

- `mutate()` is used to make changes to variables
- `as.factor()` changes the data type to factor
- `as.numeric()` changes the data type to numeric

```
# change the data types for grass and age
gss.2016.clean <- gss.2016 %>%
    mutate(grass = as.factor(grass)) %>%
    mutate(age = as.numeric(age))
```

# Checking our work

```
# check the types
class(x = gss.2016.clean$grass)
```

```
## [1] "factor"
```

```
class(x = gss.2016.clean$age)
```

```
## [1] "numeric"
```

```
# use summary to check the new data
summary(object = gss.2016.clean)
```

```
##        grass           age
##  DK        : 110   Min.   :18.00
##  IAP       : 911   1st Qu.:34.00
##  LEGAL     :1126   Median :49.00
##  NOT LEGAL: 717   Mean   :48.85
##  NA's      :   3   3rd Qu.:62.00
##                    Max.   :88.00
##                    NA's   :32
```

# Coding missing values

- R recognizes NA as a missing value
- The grass variable has NA but also DK for **Don't Know** and IAP for **Inapplicable**
- Don't know and Inapplicable might be considered missing depending on the research question you are asking
- Use the data management code you started and add to it!
- Use `na_if()` function

```
# recode don't know and inapplicable to NA
gss.2016.clean <- gss.2016 %>%
    mutate(grass = as.factor(grass)) %>%
    mutate(age = as.numeric(age)) %>%
    mutate(grass = na_if(x = grass, y = "DK")) %>%
    mutate(grass = na_if(x = grass, y = "IAP"))
```

## Check our work

```
# check for NA
summary(object = gss.2016.clean)
```

```
##        grass           age        
##  DK       :   0   Min.   :18.00  
##  IAP      :   0   1st Qu.:34.00  
##  LEGAL    :1126   Median :49.00  
##  NOT LEGAL: 717   Mean   :48.85  
##  NA's     :1024   3rd Qu.:62.00  
##                   Max.   :88.00  
##                   NA's   :32
```

## Get rid of unused categories of a factor

- There are no observations where grass is DK or IAP, so these categories can be dropped
- Use `droplevels()` to drop unused levels
- Add to your existing data cleaning code

```
# recode don't know and inapplicable to NA
gss.2016.clean <- gss.2016 %>%
    mutate(grass = as.factor(grass)) %>%
    mutate(age = as.numeric(age)) %>%
    mutate(grass = na_if(x = grass, y = "DK")) %>%
    mutate(grass = na_if(x = grass, y = "IAP")) %>%
    mutate(grass = droplevels(x = grass))
```

## Check our work

```
# check for dropped levels
summary(object = gss.2016.clean)
```

```
##        grass           age        
##  LEGAL    :1126   Min.   :18.00  
##  NOT LEGAL: 717   1st Qu.:34.00  
##  NA's     :1024   Median :49.00  
##                   Mean   :48.85  
##                   3rd Qu.:62.00  
##                   Max.   :88.00  
##                   NA's   :32
```

## Let's recode age to be categorical!

- Often continuous variables are recoded to categorical
- This is sometimes necessary to use certain types of statistical models
- There are good things and bad things about categorizing a continuous variable
- Bring your data cleaning code and let's add an `age.cat` variable with 5 categories of age

```
# recode don't know and inapplicable to NA
gss.2016.clean <- gss.2016 %>%
    mutate(grass = as.factor(grass)) %>%
    mutate(age = as.numeric(age)) %>%
    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(18, 29, 59, 74, 88),
                         labels = c("18 - 29", "30 - 59", "60 - 74", "75+")))
```

# Let's rename the legalization categories

- The two categories of `grass` are coded `LEGAL` and `NOT LEGAL`
- Let's change to `Yes` and `No`

```
# recode don't know and inapplicable to NA
gss.2016.clean <- gss.2016 %>%
    mutate(grass = as.factor(grass)) %>%
    mutate(age = as.numeric(age)) %>%
    mutate(grass = na_if(x = grass, y = "DK")) %>%
    mutate(grass = na_if(x = grass, y = "IAP")) %>%
    mutate(grass = droplevels(x = grass)) %>%
    mutate(grass = recode_factor(.x = grass,
                         `LEGAL` = "Yes",
                         `NOT LEGAL` = "No")) %>%
    mutate(age.cat = cut(x = age,
                         breaks = c(18, 29, 59, 74, 88),
                         labels = c("18 - 29", "30 - 59", "60 - 74", "75+")))
```

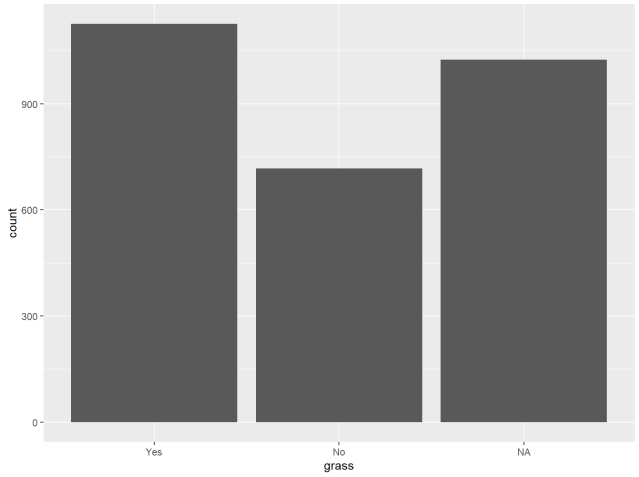
# Check our work

```
# check for age.cat variable
summary(object = gss.2016.clean)
```

```
##   grass          age           age.cat
##  Yes :1126   Min.   :18.00   18 - 29: 474
##  No  : 717   1st Qu.:34.00   30 - 59:1517
##  NA's:1024   Median :49.00   60 - 74: 598
##              Mean   :48.85   75+    : 239
##              3rd Qu.:62.00   NA's   :  39
##              Max.   :88.00
##              NA's   :32
```

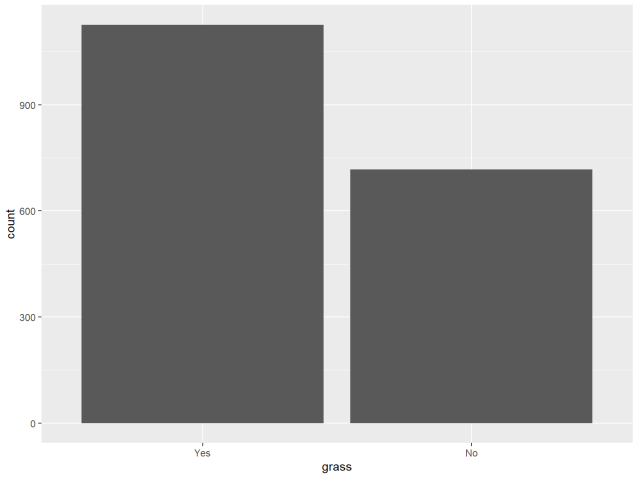
# Let's make a graph!

```
# graph support for legalization
gss.2016.clean %>%
    ggplot(aes(x = grass)) +
    geom_bar()
```



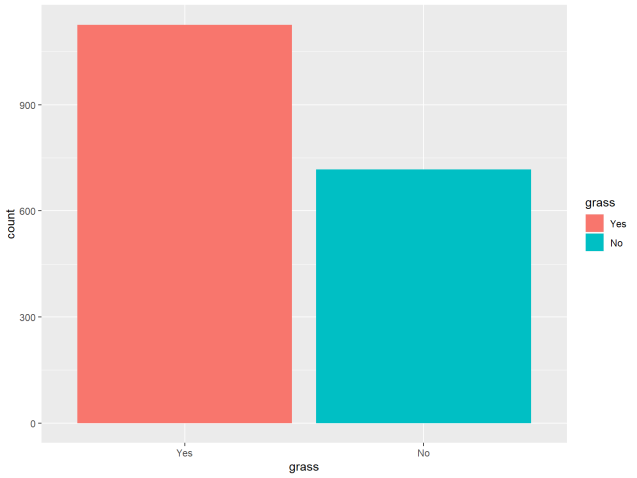

# Let's get rid of the NA in the graph

```
# graph support for legalization
gss.2016.clean %>%
    drop_na(grass) %>%
    ggplot(aes(x = grass)) +
    geom_bar()
```



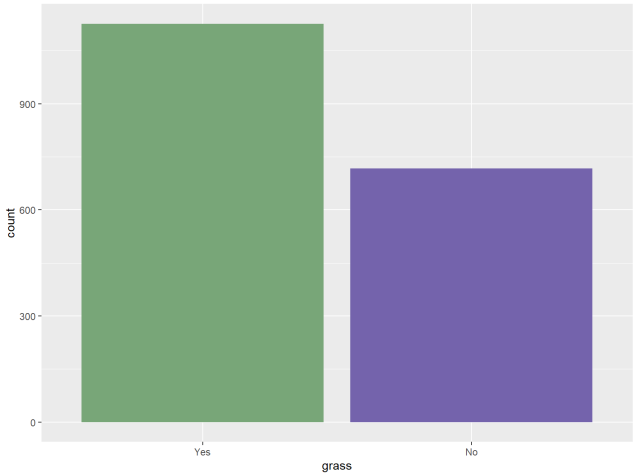
## Let's add some color

```
# graph support for legalization
gss.2016.clean %>%
    drop_na(grass) %>%
    ggplot(aes(x = grass, fill = grass)) +
    geom_bar()
```



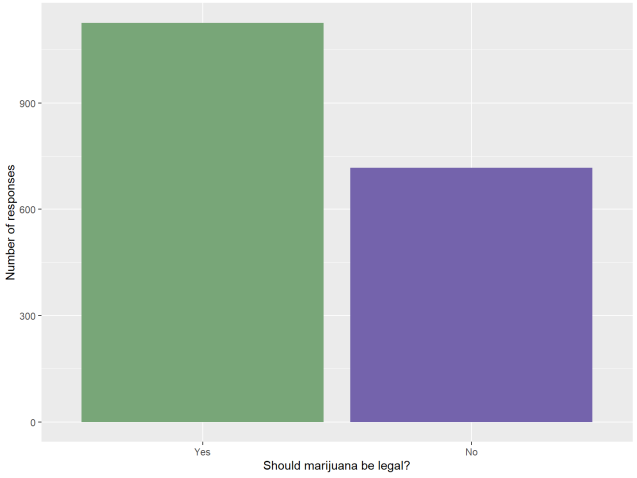

## How about green instead? …and we do not need the legend

```
# graph support for legalization
gss.2016.clean %>%
    drop_na(grass) %>%
    ggplot(aes(x = grass, fill = grass)) +
    geom_bar() +
    scale_fill_manual(values = c("#78A678", "#7463AC"),
                      guide = FALSE)
```



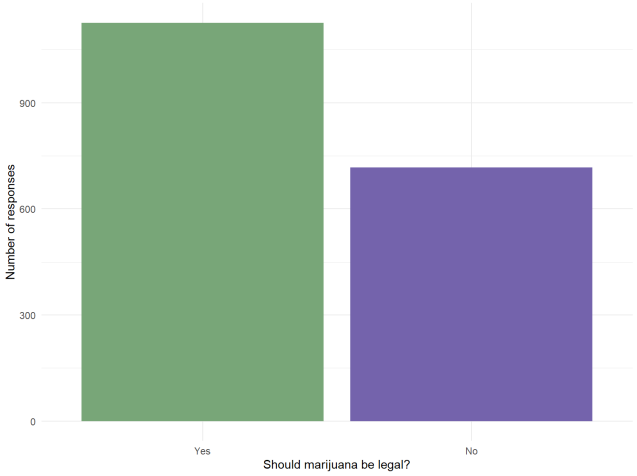

## Let's add better labels to the axes

```
# graph support for legalization
gss.2016.clean %>%
    drop_na(grass) %>%
    ggplot(aes(x = grass, fill = grass)) +
    geom_bar() +
    scale_fill_manual(values = c("#78A678", "#7463AC"),
                      guide = FALSE) +
    labs(x = "Should marijuana be legal?",
         y = "Number of responses")
```



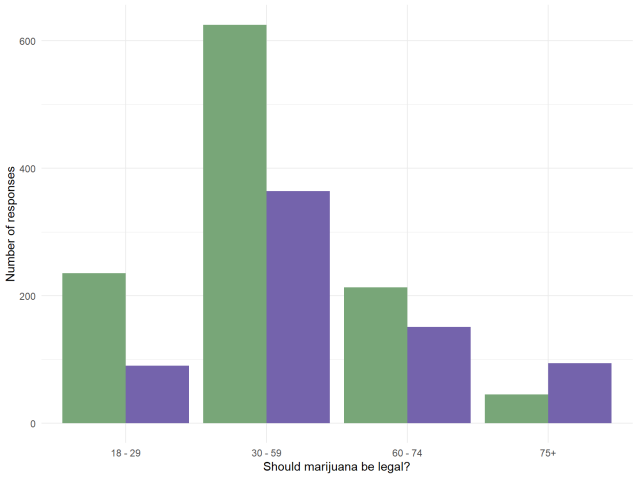

## Get rid of the gray background

```
# graph support for legalization
gss.2016.clean %>%
    drop_na(grass) %>%
    ggplot(aes(x = grass, fill = grass)) +
    geom_bar() +
    scale_fill_manual(values = c("#78A678", "#7463AC"),
                      guide = FALSE) +
    labs(x = "Should marijuana be legal?",
         y = "Number of responses") +
    theme_minimal()
```



## Add age to the graph

```
# graph support for legalization
gss.2016.clean %>%
    drop_na(grass) %>%
    drop_na(age.cat) %>%
    ggplot(aes(x = age.cat, fill = grass)) +
    geom_bar(position = "dodge") +
    scale_fill_manual(values = c("#78A678", "#7463AC"),
                      guide = FALSE) +
    labs(x = "Should marijuana be legal?",
         y = "Number of responses") +
    theme_minimal()
```




## Can we answer our research question now?

**What is the level of support for marijuana legalization in the US? How might it change in the next 20 years?**

## Hey that was A LOT!

- If you are feeling a little lost in the ggplot…everyone does! This tweet is from the creator of ggplot:


- The great thing about ggplot is that it is extremely flexible; you can do almost anything
- This makes it complex to use, though

## The End