Barplots & `ggplot2` Basics

Justin Post

Summarizing Categorical Variables

Categorical (Qualitative) variable - entries are a label or attribute

- Numerical summary: contingency tables
- Graphical summary: bar plots

Summarizing Categorical Variables

Categorical (Qualitative) variable - entries are a label or attribute

- Numerical summary: contingency tables
- Graphical summary: bar plots

We can easily create plots in R (one of its real strengths!)

There are three major systems for plotting in R (and others that have been ported in)

- Base R (built-in functions)
- `lattice`
- `ggplot2` (part of the tidyverse but not exactly)

We'll use `ggplot2` as it is very popular. There is a great ggplot2 reference book here!

Reading in Our Data

First, let's read in the appendicitis data from the previous lecture.

```
library(tidyverse)
library(readxl)
app_data <- read_excel("data/app_data.xlsx", sheet = 1)
app_data <- app_data |>
  mutate(BMI = as.numeric(BMI),
         US_Number = as.character(US_Number),
         SexF = factor(Sex, levels = c("female", "male"), labels = c("Female", "Male")),
         DiagnosisF = as.factor(Diagnosis),
         SeverityF = as.factor(Severity))
app_data
```

```
## # A tibble: 782 x 61
##     Age   BMI Sex    Height Weight Length_of_Stay Management   Severity
##   <dbl> <dbl> <chr>   <dbl>  <dbl>          <dbl> <chr>        <chr>
## 1  12.7  16.9 female    148   37                3 conservative uncomplicated
## 2  14.1  31.9 male      147   69.5              2 conservative uncomplicated
## 3  14.1  23.3 female    163   62                4 conservative uncomplicated
## 4  16.4  20.6 female    165   56                3 conservative uncomplicated
## 5  11.1  16.9 female    163   45                3 conservative uncomplicated
## # i 777 more rows
## # i 53 more variables: Diagnosis_Presumptive <chr>, Diagnosis <chr>,
##                        riatic_Appendicitis_Score <dbl>,
##                        ndix_Diameter <dbl>, Migratory_Pain <chr>,
##                        , Contralateral_Rebound_Tenderness <chr>,
## #   Coughing_Pain <chr>, Nausea <chr>, Loss_of_Appetite <chr>,
```

`ggplot2` Basics

`ggplot` cheat sheet is handy!

To get started:

- `ggplot(data = data_frame)` creates a plot instance

`ggplot2` Basics

`ggplot` cheat sheet is handy!

To get started:

- `ggplot(data = data_frame)` creates a plot instance
- Then we add "layers" to the plot (`geom` or `stat` layers)
    - This actually creates a visualization of the data

`ggplot2` Basics

`ggplot` cheat sheet is handy!

To get started:

- `ggplot(data = data_frame)` creates a plot instance
- Then we add "layers" to the plot (`geom` or `stat` layers)
    - This actually creates a visualization of the data
- Modify layer "mapping" args (usually with `aes()`)
    - Map variables to attributes of the plot
    - Ex: size, color, x variable, y variable

`ggplot2` Basics

`ggplot` cheat sheet is handy!

To get started:

- `ggplot(data = data_frame)` creates a plot instance
- Then we add "layers" to the plot (`geom` or `stat` layers)
    - This actually creates a visualization of the data
- Modify layer "mapping" args (usually with `aes()`)
    - Map variables to attributes of the plot
    - Ex: size, color, x variable, y variable
- Improve by adding title layers, faceting, etc.

ggplot2 Barplots

- Barplots via `ggplot() + geom_bar()`
- Across x-axis we want our categories - specify with `aes(x = ...)`

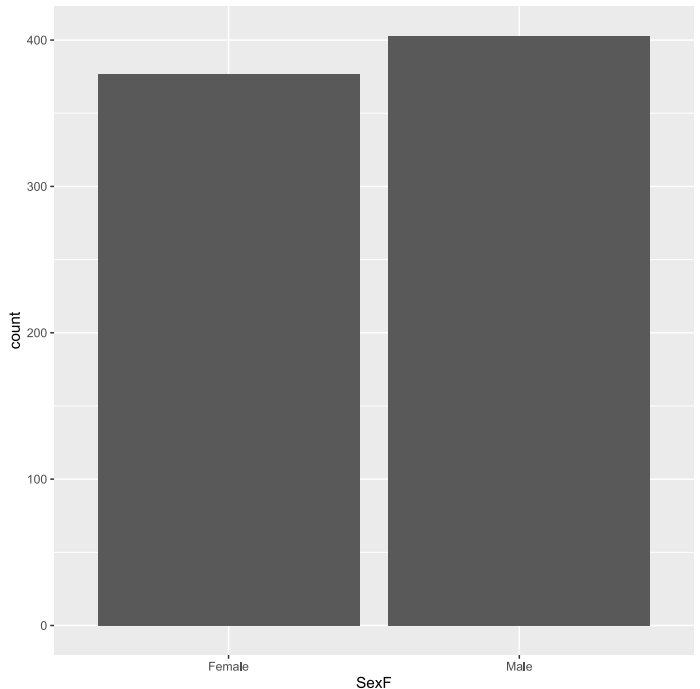
```
ggplot(data = app_data |> drop_na(SexF), aes(x = SexF))
```

NC STATE UNIVERSITY

ggplot2 Barplots

- Barplots via `ggplot() + geom_bar()`
- Must add geom (or stat) layer!

```
ggplot(data = app_data |> drop_na(SexF), aes(x = SexF)) +
  geom_bar()
```

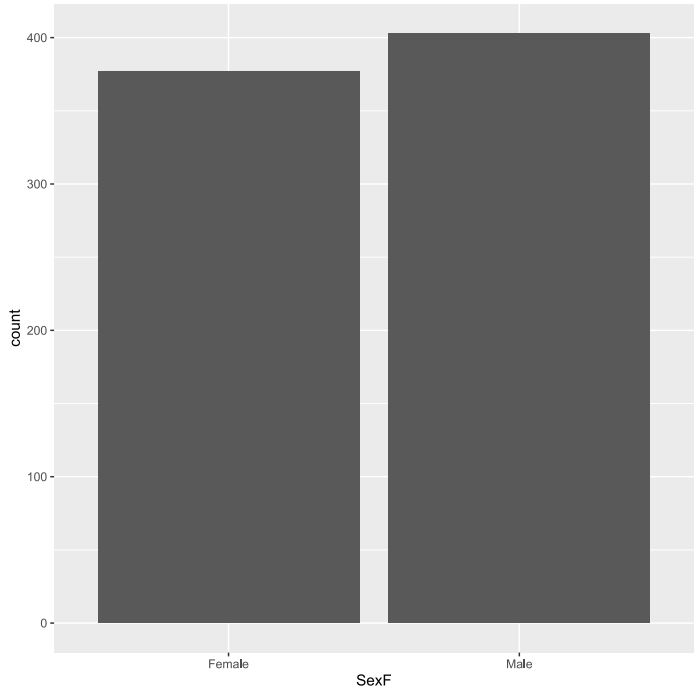


NC STATE UNIVERSITY

ggplot2 Barplots

- Generally: Save base object with **global** `aes()` assignments, then add layers

```
g <- ggplot(data = app_data |> drop_na(SexF), aes(x = SexF))
g + geom_bar()
```

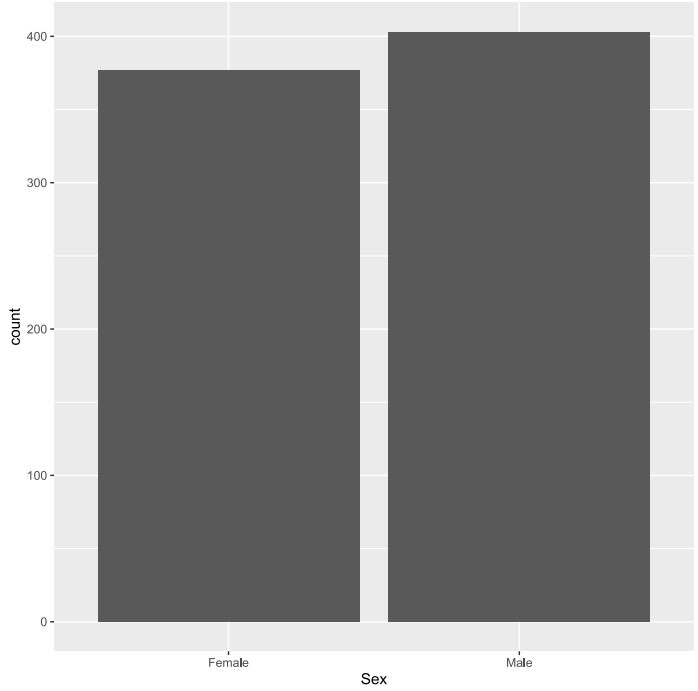


NC STATE UNIVERSITY

Better Labeling Needed

- We can easily improve the labeling on the x-axis variable (see the 2nd page of cheat sheet on the right!)

```
g <- ggplot(data = app_data |> drop_na(SexF), aes(x = SexF))
g + geom_bar() +
    labs(x = "Sex")
```



`aes()` Arguments

- `aes()` defines visual properties of objects in the plot
- Map variables in the data frame to plot elements

  `x = , y = , size = , shape = , color = , alpha = , ...`

- For a bar plot, from the cheat sheet we see

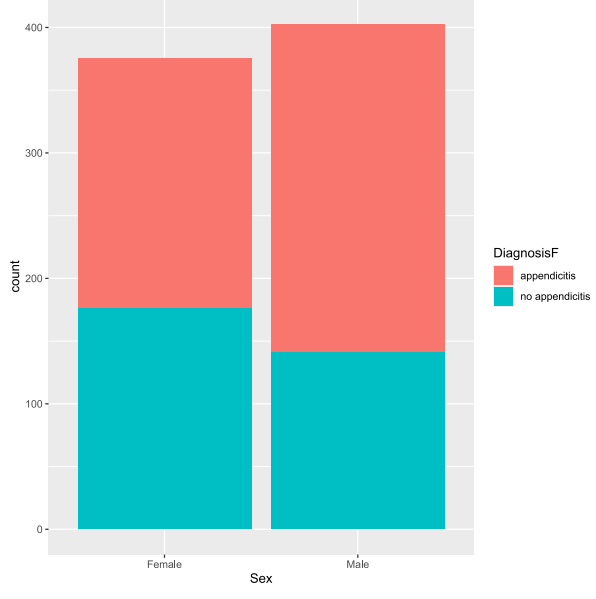
  `d + geom_bar()`

  `x, alpha, color, fill, linetype, size, weight`

`aes()` Arguments for Barplots

- **Stacked barplot** created by via `fill` aesthetic
- Automatic assignment of colors and creation of legends for `aes` elements (except group)

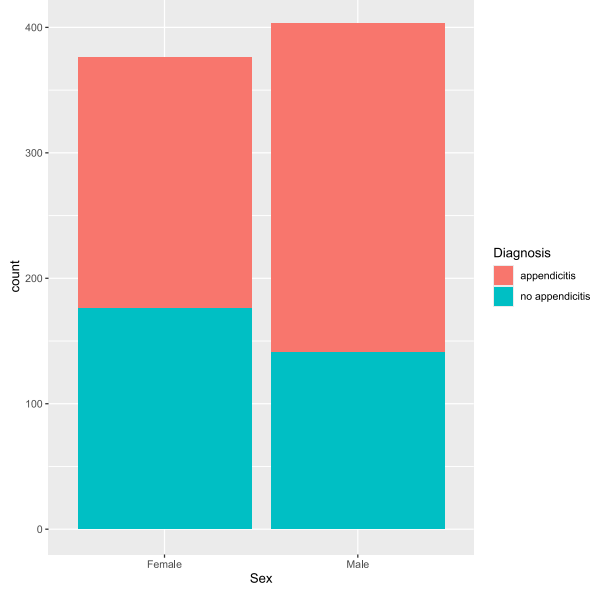
```
g <- ggplot(data = app_data |> drop_na(SexF, DiagnosisF), aes(x = SexF, fill = DiagnosisF))
g + geom_bar()+
    labs(x = "Sex")
```



Update Legend

- Can change automatically created labels/legend

```
g <- ggplot(data = app_data |> drop_na(SexF, DiagnosisF), aes(x = SexF, fill = DiagnosisF))
g + geom_bar() +
  labs(x = "Sex")+
  scale_fill_discrete("Diagnosis")
```



`ggplot2` Global vs Local Aesthetics

`data` and `aes` can be set in two ways;

- 'globally' (for all layers) via the `aes()` function in the `ggplot()` call
- 'locally' (for just that layer) via the `geom` or `stat` layer's `aes()`

ggplot2 Global vs Local Aesthetics

`data` and `aes` can be set in two ways;

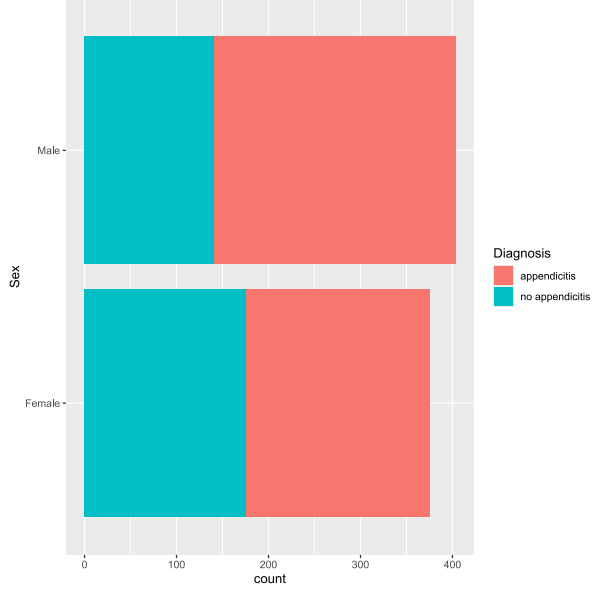
- 'globally' (for all layers) via the `aes()` function in the `ggplot()` call
- 'locally' (for just that layer) via the `geom` or `stat` layer's `aes()`

```
#global
ggplot(data = app_data |> drop_na(SexF, DiagnosisF), aes(x = SexF, fill = DiagnosisF)) +
  geom_bar()

#some local, some global
ggplot(data = app_data |> drop_na(SexF, DiagnosisF), aes(x = SexF)) +
  geom_bar(aes(fill = DiagnosisF))

#all local
ggplot(data = app_data |> drop_na(SexF, DiagnosisF)) +
  geom_bar(aes(x = SexF, fill = DiagnosisF))
```

ggplot2 Horizontal Barplots

- Easy to rotate a plot with `coord_flip()`

```
ggplot(data = app_data |> drop_na(SexF, DiagnosisF), aes(x = SexF, fill = DiagnosisF)) +
  geom_bar() +
  labs(x = "Sex")+
  scale_fill_discrete("Diagnosis")+
  coord_flip()
```



ggplot2 Stat vs Geom layers

Note: Most `geoms` have a corresponding `stat` layer that can be used

```
geom_bar(mapping = NULL, data = NULL, stat = "count",
         position = "stack", ..., width = NULL, binwidth = NULL, na.rm = FALSE,
         show.legend = NA, inherit.aes = TRUE)
```

ggplot2 Stat vs Geom layers

Note: Most `geoms` have a corresponding `stat` layer that can be used

```
geom_bar(mapping = NULL, data = NULL, stat = "count",
         position = "stack", ..., width = NULL, binwidth = NULL, na.rm = FALSE,
         show.legend = NA, inherit.aes = TRUE)
```

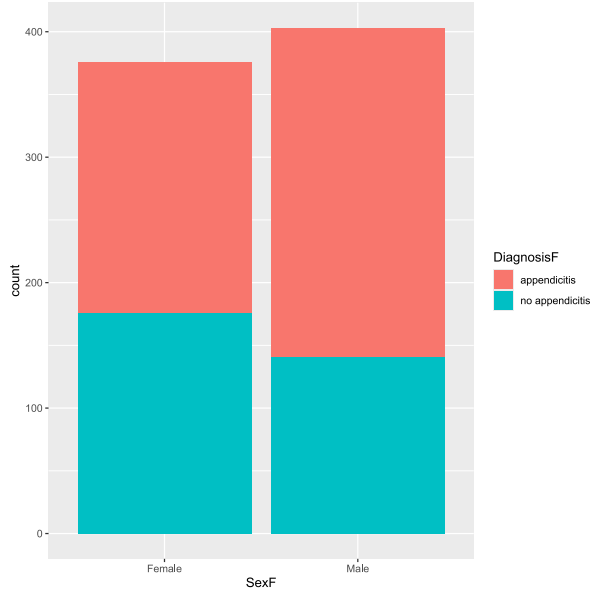
- Equivalent plots via:

```
ggplot(data = app_data |> drop_na(SexF, DiagnosisF), aes(x = SexF, fill = DiagnosisF)) + geom_bar()
ggplot(data = app_data |> drop_na(SexF, DiagnosisF), aes(x = SexF, fill = DiagnosisF)) + stat_count()
```

`ggplot2` Stat vs Geom layers

- Can modify the stat: if you have summary data, specify `y` and use `stat = identity`

```
ggplot(data = app_data |>
         drop_na(SexF, DiagnosisF) |>
         group_by(SexF, DiagnosisF) |>
         summarize(count = n()), aes(x = SexF, y = count, fill = DiagnosisF)) +
  geom_bar(stat = "identity")
```



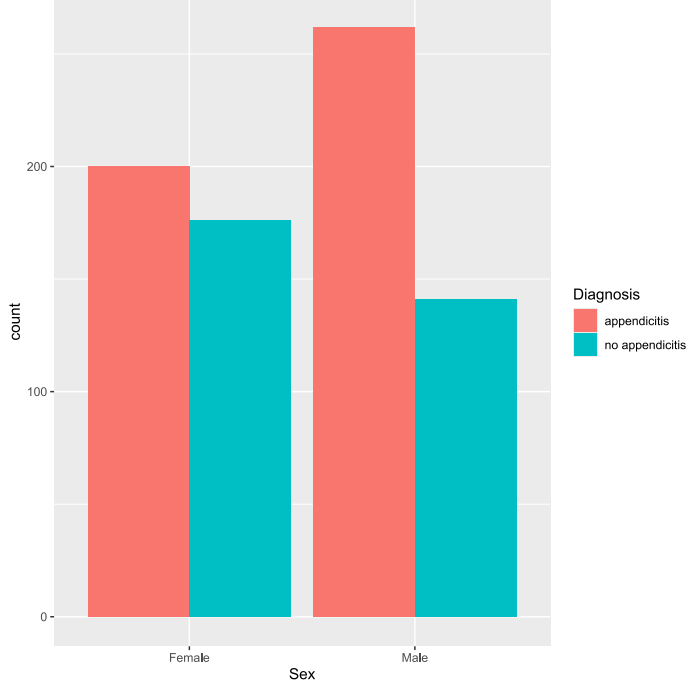
`ggplot2` Side-By-Side Barplots

- **Side-by-side barplot** created via the `position` aesthetic
  - `dodge` for side-by-side bar plot
  - `jitter` for continuous data with many points at same values
  - `fill` stacks bars and standardises each stack to have constant height
  - `stack` stacks bars on top of each other

ggplot2 Side-By-Side Barplots

- **Side-by-side barplot** created by via `position` aesthetic

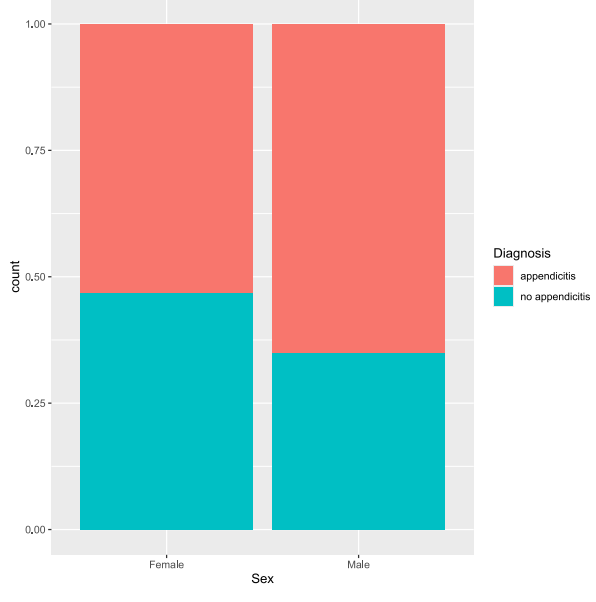
```
ggplot(data = app_data |> drop_na(SexF, DiagnosisF), aes(x = SexF, fill = DiagnosisF)) +
  geom_bar(position = "dodge") +
  labs(x = "Sex")+
  scale_fill_discrete("Diagnosis")
```



ggplot2 Filled Barplots

- `position = fill` stacks bars and standardizes each stack to have constant height (especially useful with equal group sizes)

```
ggplot(data = app_data |> drop_na(SexF, DiagnosisF), aes(x = SexF, fill = DiagnosisF)) +
  geom_bar(position = "fill") +
  labs(x = "Sex")+
  scale_fill_discrete("Diagnosis")
```



Recap!

- How to summarize categorical data?
- Numerically?
    - Tables (contingency tables)
        - Show frequency of categories
- Graphically?
    - Barplots
- ggplot (create object, add layers)
    - Data Frame
    - Geoms (Vis type)
    - Aesthetic (aes)
    - Coordinate system, stat, labels, etc.