5. More tidyverse and intro to ggplot

Frank Edwards

Piping commands together

Tidyverse uses a special symbol, called a pipe `%>%` to string together commands. Cmd + shift + m will make one.

Piping commands

The pipe `%>%` tells R to use `iris` for all commands that follow

```
iris %>%
  filter(Species == "versicolor")
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1           7.0         3.2          4.7         1.4 versicolor
## 2           6.4         3.2          4.5         1.5 versicolor
## 3           6.9         3.1          4.9         1.5 versicolor
## 4           5.5         2.3          4.0         1.3 versicolor
## 5           6.5         2.8          4.6         1.5 versicolor
## 6           5.7         2.8          4.5         1.3 versicolor
## 7           6.3         3.3          4.7         1.6 versicolor
## 8           4.9         2.4          3.3         1.0 versicolor
## 9           6.6         2.9          4.6         1.3 versicolor
## 10          5.2         2.7          3.9         1.4 versicolor
## 11          5.0         2.0          3.5         1.0 versicolor
## 12          5.9         3.0          4.2         1.5 versicolor
## 13          6.0         2.2          4.0         1.0 versicolor
## 14          6.1         2.9          4.7         1.4 versicolor
## 15          5.6         2.9          3.6         1.3 versicolor
## 16          6.7         3.1          4.4         1.4 versicolor
## 17          5.6         3.0          4.5         1.5 versicolor
## 18          5.8         2.7          4.1         1.0 versicolor
## 19          6.2         2.2          4.5         1.5 versicolor
## 20          5.6         2.5          3.9         1.1 versicolor
## 21          5.9         3.2          4.8         1.8 versicolor
## 22          6.1         2.8          4.0         1.3 versicolor
## 23          6.3         2.5          4.9         1.5 versicolor
```

Piping commands

```r
iris %>%
  mutate(sepal2 = Sepal.Length + 2)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species sepal2
## 1            5.1         3.5          1.4         0.2     setosa    7.1
## 2            4.9         3.0          1.4         0.2     setosa    6.9
## 3            4.7         3.2          1.3         0.2     setosa    6.7
## 4            4.6         3.1          1.5         0.2     setosa    6.6
## 5            5.0         3.6          1.4         0.2     setosa    7.0
## 6            5.4         3.9          1.7         0.4     setosa    7.4
## 7            4.6         3.4          1.4         0.3     setosa    6.6
## 8            5.0         3.4          1.5         0.2     setosa    7.0
## 9            4.4         2.9          1.4         0.2     setosa    6.4
## 10           4.9         3.1          1.5         0.1     setosa    6.9
## 11           5.4         3.7          1.5         0.2     setosa    7.4
## 12           4.8         3.4          1.6         0.2     setosa    6.8
## 13           4.8         3.0          1.4         0.1     setosa    6.8
## 14           4.3         3.0          1.1         0.1     setosa    6.3
## 15           5.8         4.0          1.2         0.2     setosa    7.8
## 16           5.7         4.4          1.5         0.4     setosa    7.7
## 17           5.4         3.9          1.3         0.4     setosa    7.4
## 18           5.1         3.5          1.4         0.3     setosa    7.1
## 19           5.7         3.8          1.7         0.3     setosa    7.7
## 20           5.1         3.8          1.5         0.3     setosa    7.1
## 21           5.4         3.4          1.7         0.2     setosa    7.4
## 22           5.1         3.7          1.5         0.4     setosa    7.1
## 23           4.6         3.6          1.0         0.2     setosa    6.6
## 24           5.1         3.3          1.7         0.5     setosa    7.1
## 25           4.8         3.4          1.9         0.2     setosa    6.8
```

Exercises

Using `iris` and the pipe operator `%>%`

- Use select to return only the petal variables and flower species
- Use filter to return only versicolor observations
- Use arrange to sort the data.frame by Sepal.Length
- Use mutate to compute the sum of petal lengths and widths
- Identify and return a data.frame containing only observations with the smallest observed petal width

Piping many commands

```
iris %>%
  filter(Species == ”versicolor”) %>%
  select(Sepal.Length, Petal.Length, Species) %>%
  mutate(Sepal.Petal.Ratio = Sepal.Length / Petal.Length)
```

```
##    Sepal.Length Petal.Length    Species Sepal.Petal.Ratio
## 1           7.0          4.7 versicolor          1.489362
## 2           6.4          4.5 versicolor          1.422222
## 3           6.9          4.9 versicolor          1.408163
## 4           5.5          4.0 versicolor          1.375000
## 5           6.5          4.6 versicolor          1.413043
## 6           5.7          4.5 versicolor          1.266667
## 7           6.3          4.7 versicolor          1.340426
## 8           4.9          3.3 versicolor          1.484848
## 9           6.6          4.6 versicolor          1.434783
## 10          5.2          3.9 versicolor          1.333333
## 11          5.0          3.5 versicolor          1.428571
## 12          5.9          4.2 versicolor          1.404762
## 13          6.0          4.0 versicolor          1.500000
## 14          6.1          4.7 versicolor          1.297872
## 15          5.6          3.6 versicolor          1.555556
## 16          6.7          4.4 versicolor          1.522727
## 17          5.6          4.5 versicolor          1.244444
## 18          5.8          4.1 versicolor          1.414634
## 19          6.2          4.5 versicolor          1.377778
## 20          5.6          3.9 versicolor          1.435897
## 21          5.9          4.8 versicolor          1.229167
## 22          6.1          4.0 versicolor          1.525000
## 23          6.3          4.9 versicolor          1.285714
```

Summarizing data

`summarize()` uses a variety of summary functions over the data (e.g. `mean`, `min`, `max`, `sd`, etc.)

```r
iris %>%
  summarize(mean.pl = mean(Petal.Length))
```

```
##   mean.pl
## 1   3.758
```

But summarize() is more powerful with its buddy, group_by()

`group_by()` groups the data by individual groups within the data

```
iris %>%
  group_by(Species) %>%
  summarize(mean.pl = mean(Petal.Length))
```

```
## # A tibble: 3 x 2
##   Species    mean.pl
##   <fct>        <dbl>
## 1 setosa        1.46
## 2 versicolor    4.26
## 3 virginica     5.55
```

We can use it to count too

```
iris %>%
  group_by(Species) %>%
  summarize(n_obs = n())
```

```
## # A tibble: 3 x 2
##   Species    n_obs
##   <fct>      <int>
## 1 setosa        50
## 2 versicolor    50
## 3 virginica     50
```

We can use it to count too

```r
iris %>%
  filter(Sepal.Width<3) %>%
  group_by(Species) %>%
  summarize(n_obs = n())
```

```
## # A tibble: 3 x 2
##   Species    n_obs
##   <fct>      <int>
## 1 setosa         2
## 2 versicolor    34
## 3 virginica     21
```

Exercises

- Which species has the largest average sepal length?
- Which species has the smallest average petal length?
- What is the average sepal width across all species?
- Which species has the highest number of observations with petal lengths less than the average petal length across all species?

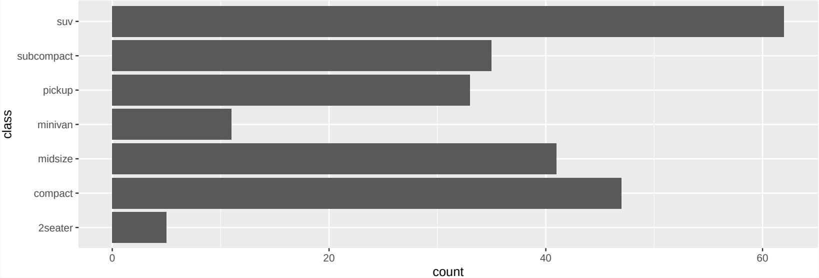
Visualizing the distribution of single variables (univariate visuals)

Visuals for categorical variables

Barplots

Show the count of rows in each value of a category

```
ggplot(mpg,
       aes(y = class)) +
  geom_bar()
```



The anatomy of a basic ggplot() call

```r
ggplot(mpg, ## object, usually a data.frame
       aes(y = class)) + ## aesthetic variables, generally x, y, color, etc
  geom_bar() ## a geom to plot the aesthetics

## Note that + in ggplot() works the same way as %>% in tidyverse:
## It strings together commands, evaluated in sequence
```

Practice

Using the `mpg` data.frame, make a barplot of `manufacturer`. Try changing the aesthetic from a `y` to an `x`. What happens?

Recall the anatomy of a ggplot call

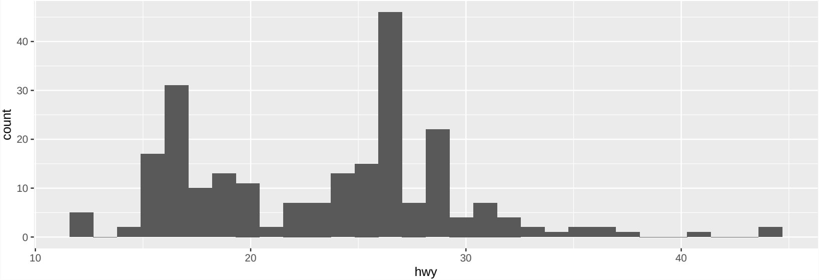
```
ggplot(DATA.FRAME.NAME,
       aes(AESTHETIC PARAMETERS)) +
  GEOM
```

Visuals for continuous variables

Histograms

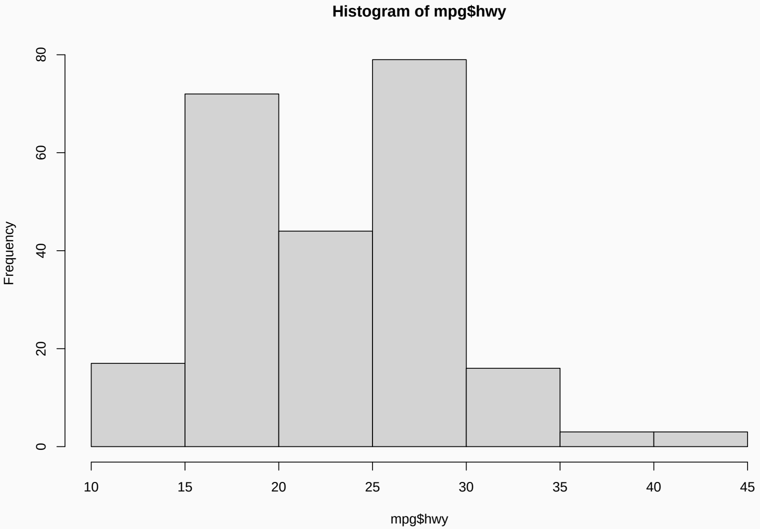
Histograms show the density of cases that fall within a given range

```
ggplot(mpg,
       aes(x = hwy)) +
  geom_histogram()
```



Histograms in base R

```
hist(mpg$hwy)
```



Practice

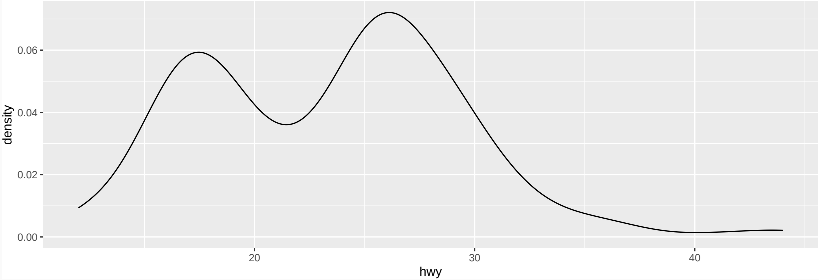
Practice

1. Make a histogram of city miles per gallon in the `mpg` dataset
2. What does this histogram tell us?

Density plots

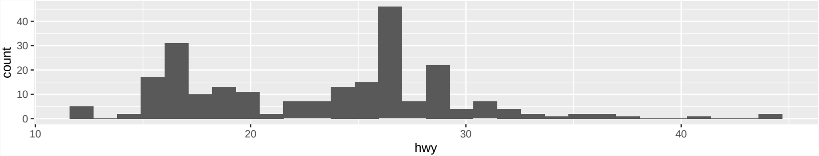
Densities are smoothed continuous histograms

```
ggplot(mpg,
       aes(x = hwy)) +
  geom_density()
```

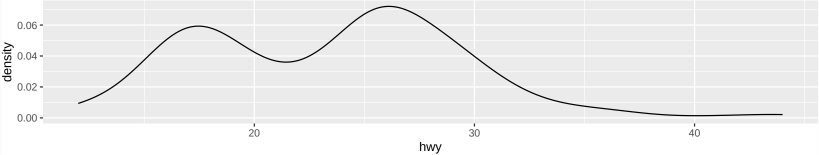


Another set of histograms/densities

```
ggplot(mpg,
       aes(x = hwy)) +
  geom_histogram()
```



```
ggplot(mpg,
       aes(x = hwy)) +
  geom_density()
```



Practice

Practice

1. Create a histogram of `displ`
2. Create a density plot of `displ`
3. Which do you think is a more effective data visual? Why?
4. Create a histogram of city mpg
5. Create a histogram of highway mpg
6. Describe differences in these distributions.
7. Using a barplot, identify which vehicle class is most common in the data
8. Using a barplot, identify which manufacturer is least common in the data