intro_ggplot2

October 19, 2020

Introduction to ggplot2

1 1. What is *ggplot2*?

- Hugely popular R package for visualization
- Authored by Hadley Wickham (of *dplyr* and *tidyverse* fame)
- Implements the "grammar-of-graphics" design philosophy (hence "gg")
- Easily produces beautiful and informative visualizations

2 2. Plotting Person-Level Characteristics in Arrests

- The Pvd arrests data are at the *violation*-level
- We want person-level data on the invdividuals arrested
- This "*level-of-analysis*" or "*level-of-granularity*" problem is ubiquitous

3 3. Generating Person-Level Data

- We are aggregating "up" from the violation level
- Will use the `group_by()` and `summarise()` idiom

```
In [2]: # Load necessary packages
        library(stringr)
        library(dplyr)
        library(ggplot2)

        arrests_df <- read.csv("./data/pvd_arrests_2020-10-03.csv")
```

3.1 3.1 Computing Number of Officers (correctly)

- First, need to determine if `arresting_officers` is in *full-name*-format or *first-initial*-format

```
In [3]: is_uppercase <- function(chr) {
            res <- chr %in% LETTERS
            return(res)
        }
```

```
has_full_names <- function(names_str) {
    char1 <- substr(names_str, 1, 1)
    char2 <- substr(names_str, 2, 2)

    res <- !(is_uppercase(char1) && is_uppercase(char2))
    return(res)
}
```

3.1.1 3.1.1 Couting the Names

- Want to correct count names regardless of format
- Update our `count_names()` function

```
In [5]: count_names <- function(names_str) {
            names_str_trm <- str_trim(names_str)     # remove whitespace

            if (has_full_names(names_str_trm)) {
                split_char <- "/ "
            } else {
                split_char <- ", "
            }

            name_vec <- unlist(str_split(names_str_trm, split_char))
            k <- length(name_vec)

            return(k)
        }
```

3.1.2 3.1.2 Counting Officers (correctly)

- Note the sequence of function calls:
  - `count_officers()` => `count_names()` => `has_full_names()` => `is_uppercase()`

```
In [6]: count_officers <- function(col) {

            n <- length(col)    # get the length of our input column
            cnts <- rep(0, n)   # allocate vector of zeros to populate with counts

            for (i in 1:n) {
                cnts[i] <- count_names(col[i])
            }
            return(cnts)
        }
```

```
In [7]: arrests_df$officer_cnt <- count_officers(arrests_df$arresting_officers)
```

3.2 3.2 Add Violent Offense Flag

```
In [8]: # Write function to flag alleged violent crimes from the
        # description of of the statute violation

        is_violent_offense <- function(v) {

            violent_terms <- c("domestic-asslt", "assault", "battery", "murder")
            n_obs <- length(v)
            is_violent <- rep(FALSE, n_obs)

            # iterate over all statute descriptions
            for (i in 1:n_obs) {

                # iterate over the 4 terms associated with violence
                for (term in violent_terms) {
                    if (!is.na(v[i]) && str_detect(tolower(v[i]), term)) {

                        is_violent[i] <- TRUE
                    }
                }
            }
            return(is_violent)
        }
```

3.2.1 3.2.1 Test our Function (always!!)

```
In [9]: vio_vec <- c("DISORDERLY CONDUCT",
                     "RESISTING LEGAL OR ILLEGAL ARREST",
                     "DOMESTIC-SIMPLE ASSAULT/BATTERY",
                     "SIMPLE ASSAULT OR BATTERY")

        is_violent_offense(vio_vec)     # Should be: FALSE, FALSE, TRUE, TRUE
```

1. FALSE 2. FALSE 3. TRUE 4. TRUE

3.2.2 3.2.2 Create `violent` Column in `arrests_df`

```
In [10]: arrests_df$violent <- is_violent_offense(arrests_df$statute_desc)
```

```
In [12]: head(arrests_df)
```

A data.frame: 6 Œ 20

	arrest_date	year	month	gender	race	ethnicity	year_of
	<chr>	<int>	<int>	<chr>	<chr>	<chr>	<int>
1	2019-08-24T02:23:00.0	2019	8	Male	White	NonHispanic	1981
2	2019-08-24T02:02:00.0	2019	8				1994
3	2019-08-24T02:02:00.0	2019	8	Female	Black	NonHispanic	1984
4	2019-08-24T02:02:00.0	2019	8	Female	Black	NonHispanic	1984
5	2019-08-24T02:02:00.0	2019	8	Female	Black	Unknown	2001
6	2019-08-24T02:02:00.0	2019	8	Female	Black	Unknown	2001

3.3 3.3 Aggregating to *Person-Level* DataFrame

- Use the `group_by()` and `summarise()` pattern from *dplyr* functions

```
In [14]: person_df <- arrests_df %>%
             group_by(arrestee_id) %>%
             summarise(
                 total_charges = n(),
                 num_uniq_arrests = length(unique(case_number)),
                 prop_violent = mean(violent),
                 mean_officer_cnt = mean(officer_cnt),
                 age = age[1],
                 gender = gender[1]
             )
```

```
`summarise()` ungrouping output (override with `.groups` argument)
```

```
In [15]: head(person_df)
```

A tibble: 6 Œ 7

arrestee_id <chr>	total_charges <int>	num_uniq_arrests <int>	prop_violent <dbl>	mean_office <dbl>
pvd10005240635544439514	2	1	0	1.000000
pvd10007039892056892673	1	1	0	1.000000
pvd10015003399035869819	6	2	0	4.000000
pvd10015761183771579680	1	1	0	2.000000
pvd10016651127192901464	1	1	1	2.000000
pvd10028326204653807523	3	3	0	1.666667

4 4. Intro to *ggplot2*

- Operates on `data.frame` objects
- Map variables to aesthetics, and then display using "geom" (i.e., "geometric object")
- Geom layers can be stack over one another to add information

```
In [16]: ggplot(person_df, aes(x = age))     # does nothing...
```

4.1 4.1 Plotting Histogram of age

```
In [17]: ggplot(person_df, aes(x = age)) +
             geom_histogram()   # kinda boring...
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

4.1.1 4.1.1 Adding `colour` and `fill` to `geom_histogram()`

```
In [19]: ggplot(person_df, aes(x = age)) +
             geom_histogram(fill = "skyblue", colour = "lightblue", bins = 30)
```

4.2 4.2 Density Plot of age

```
In [21]: ggplot(person_df, aes(x = age)) +
             geom_density(fill = "skyblue", colour = "lightblue")
```

4.2.1 4.2.1 Adjusting alpha

```
In [22]: ggplot(person_df, aes(x = age)) +
             geom_density(fill = "skyblue", colour = "lightblue", alpha = 0.5)
```

4.2.2 4.2.2 Adding `gender` Variable Aesthetic

```
In [23]: ggplot(person_df, aes(x = age, y = stat(count), fill = gender, colour = gender)) +
             geom_density(alpha = 0.4) +
             xlab("Age of Person Arrested") +
             ylab("Count")
```

4.3 4.3 Scatter Plot of age and total_charges

```
In [24]: ggplot(person_df, aes(x = age, y = total_charges)) +
             geom_point()
```

4.3.1 4.3.1 Adjusting `colour` and `alpha`

```
In [25]: ggplot(person_df, aes(x = age, y = total_charges)) +
             geom_point(colour = "skyblue", alpha = 0.6)
```

4.3.2 4.3.2 Using `geom_jitter` for Scatterplots

```
In [30]: ggplot(person_df, aes(x = age, y = total_charges)) +
             geom_jitter(colour = "purple", alpha = 0.4)
```

4.4 4.4 Plotting `num_uniq_arrests` and `total_charges` with a `stat_smooth()` Layer

```
In [31]: ggplot(person_df, aes(x = num_uniq_arrests, y = total_charges)) +
             geom_jitter(colour = "violet", alpha = 0.4) +
             stat_smooth(method = "lm", formula = y ~ x)
```

4.5 4.5 Adding Third Variable to aes()

```
In [33]: ggplot(person_df, aes(x = num_uniq_arrests, y = total_charges, colour = prop_violent)
             geom_jitter(alpha = 0.5) +
             xlim(1, 6) +
             ylim(1, 15)

Warning message:
Removed 2585 rows containing missing values (geom_point).
```