EDAV Fall 2019 PSet 2

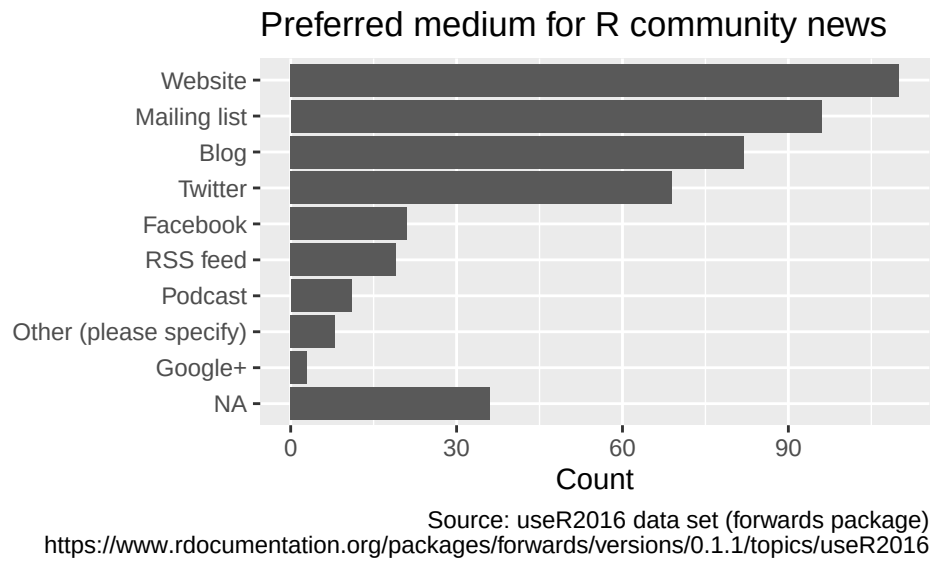
Jake Stamell, Amogh Mishra

```
# Loading packages for all questions
library(tidyverse)
library(forwards)
library(robotstxt)
library(nycflights13)
library(rvest)
library(hexbin)
library(plotly)
library(GGally)
```

1. useR2016! survey

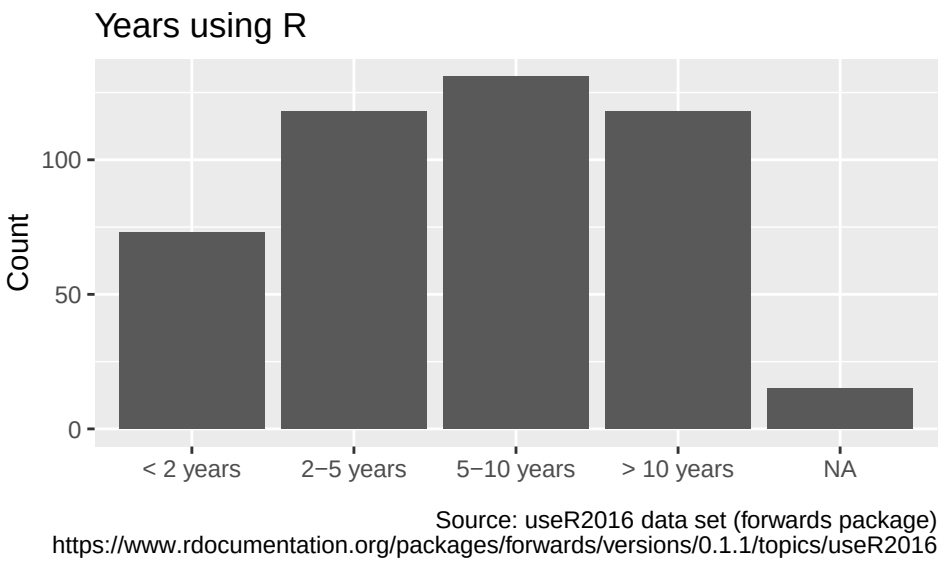
(a) Create a horizontal bar chart of the responses to Q20.

```
survey_data <- as_tibble(useR2016)
question <- "Q20"

ggplot(survey_data) +
  geom_bar(aes(x= fct_relevel(
    fct_rev(
      fct_infreq(
        fct_explicit_na(!!sym(question),"NA")))
    ,"NA"))) +
  coord_flip() +
  labs(x=element_blank(),
       y="Count",
       title="Preferred medium for R community news",
       caption="Source: useR2016 data set (forwards package)
       https://www.rdocumentation.org/packages/forwards/versions/0.1.1/topics/useR2016")
```



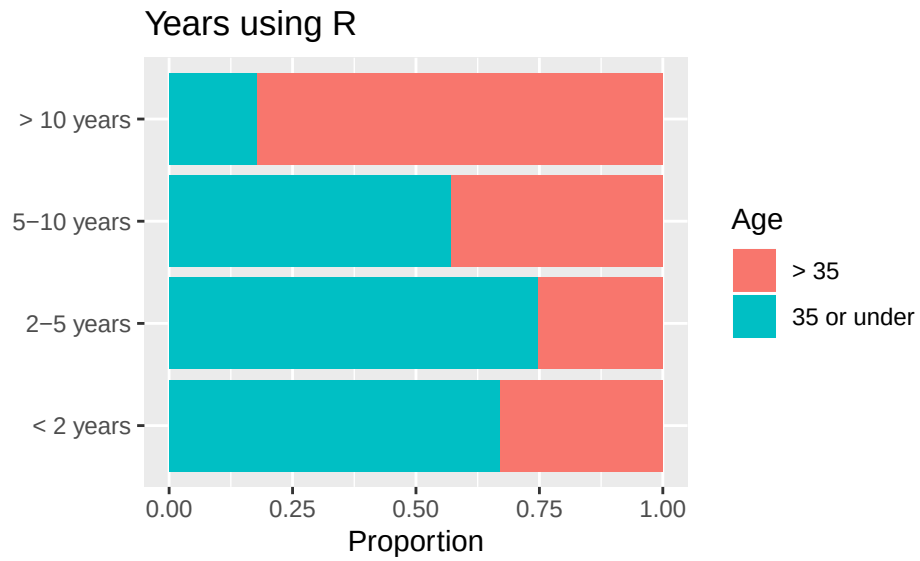
(b) Create a vertical bar chart of the responses to Q11.

```
question <- "Q11"

ggplot(survey_data) +
  geom_bar(aes(x=!!sym(question))) +
  labs(x=element_blank(),
       y="Count",
       title="Years using R",
       caption="Source: useR2016 data set (forwards package)
       https://www.rdocumentation.org/packages/forwards/versions/0.1.1/topics/useR2016")
```



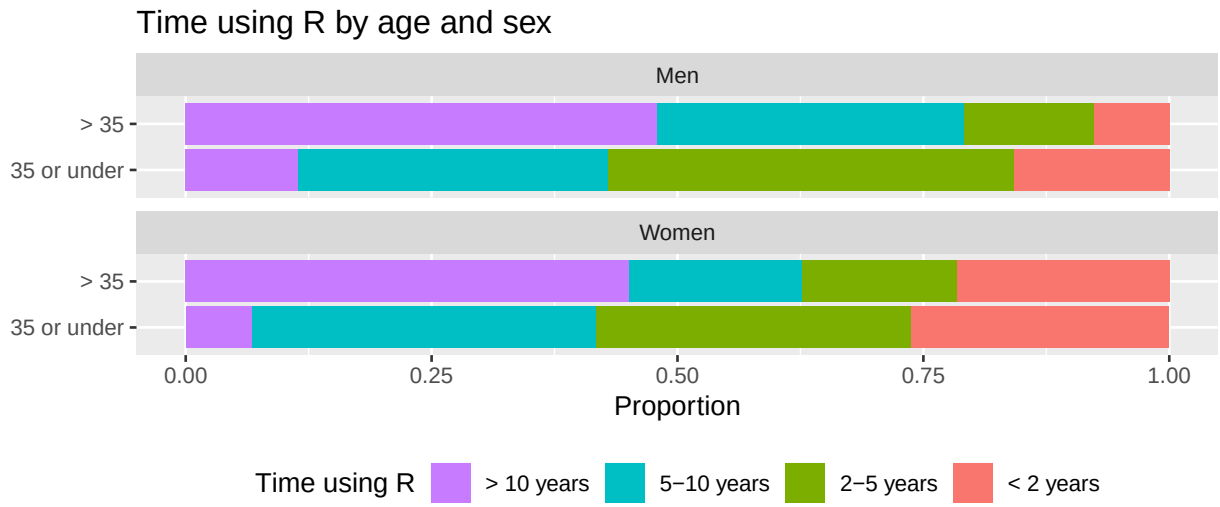
(c) Create a horizontal stacked bar chart showing the proportion of respondents for each level of Q11 who are over 35 vs. 35 or under. Use a descriptive title.

```
question <- "Q11"
fill_q <- "Q3"

ggplot(survey_data[!is.na(survey_data[,fill_q]) & !is.na(survey_data[,question]),]) +
  geom_bar(aes(x=!!sym(question), fill=!!sym(fill_q)), position="fill") +
  coord_flip() +
  labs(x=element_blank(),
       y="Proportion",
       fill="Age",
       title="Years using R")
```



(d) Create a horizontal stacked bar chart showing the proportional breakdown of Q11 for each level of Q3, faceted on Q2. Use a descriptive title.

```
question <- "Q3"
fill_q <- "Q11"
facet_q <- "Q2"

ggplot(survey_data[!is.na(survey_data[,fill_q]) & !is.na(survey_data[,question]),]) +
  geom_bar(aes(x=fct_rev(!!sym(question)), fill=!!sym(fill_q)), position="fill") +
  coord_flip() +
  facet_wrap(vars(!!sym(facet_q)),ncol=1) +
  guides(fill = guide_legend(reverse = TRUE)) +
  labs(x=element_blank(),
       y="Proportion",
       fill="Time using R",
       title="Time using R by age and sex") +
  theme(legend.position = "bottom")
```

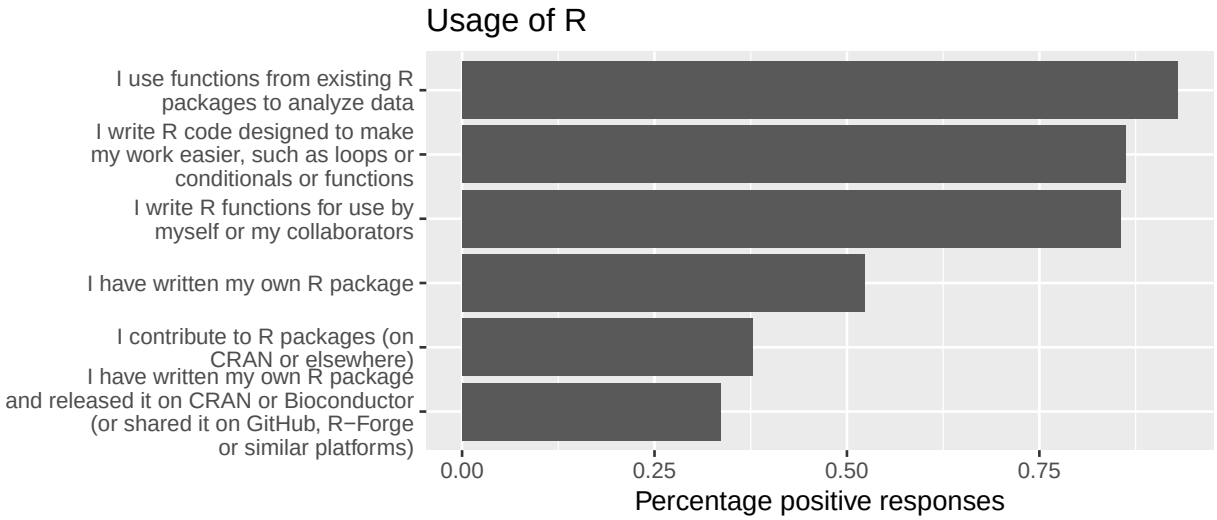


(e) For the next part, we will need to be able to add line breaks (\n) to long tick mark labels. Write a function that takes a character string and a desired approximate line length in number of characters and substitutes a line break for the first space after every multiple of the specified line length.

```
add_breaks <- function(string=as.character(), desired_length=30){
  string_out <- c()
  while(str_length(string)>desired_length){
    string_before <- str_sub(string, 1, desired_length)
    string_after <- str_sub(string, desired_length+1, str_length(string))
    space_location <- str_locate(string_after, " ")
    if(is.na(space_location[1])){break}
    string_out <- str_c(string_out, string_before,
                        str_sub(string_after, 1, space_location[1]-1),"\n")
    string <- str_sub(string_after, space_location[1]+1, str_length(string_after))
  }
  return(str_c(string_out,string))
}
```

(f) Create a horizontal bar chart that shows the percentage of positive responses for `Q13 - Q13_F`. Use your function from part (e) to add line breaks to the responses. Your graph should have one bar each for `Q13 - Q13_F`.

```
survey_data %>% select(Q13:Q13_F) %>%
  gather(key="question", value="response") %>%
  group_by(response) %>%
  summarise(num_positive = n(), prop_positive = num_positive/nrow(survey_data)) %>%
  filter(!is.na(response)) %>% rowwise() %>%
  mutate(response_adj=add_breaks(response,30)) %>%
  ungroup() %>%
  ggplot() +
  geom_bar(aes(x=reorder(response_adj,prop_positive), y=prop_positive), stat="identity") +
  coord_flip() +
  labs(x=element_blank(),
       y="Percentage positive responses",
       title="Usage of R")
```



2. Rotten Tomatoes

(a) Use the `paths_allowed()` function from **robotstxt** to make sure it's ok to scrape https://www.rottentomatoes.com/browse/box-office/. Then use **rvest** functions to find relative links to individual movies listed on this page. Finally, paste the base URL to each to create a character vector of URLs.

Display the first six lines of the vector.

```
paths_allowed("https://www.rottentomatoes.com/browse/box-office/")
```

```
## [1] TRUE
```

```
tomato_page <- read_html("https://www.rottentomatoes.com/browse/box-office/")
movie_links_html <- html_nodes(tomato_page, ".left a") %>%
  html_attr('href')
movie_links_full <- paste0("https://www.rottentomatoes.com", movie_links_html )
movie_links_full[1:6]
```

```
## [1] "https://www.rottentomatoes.com/m/downton_abbey/"
## [2] "https://www.rottentomatoes.com/m/ad_astra/"
## [3] "https://www.rottentomatoes.com/m/rambo_last_blood/"
## [4] "https://www.rottentomatoes.com/m/it_chapter_two/"
## [5] "https://www.rottentomatoes.com/m/hustlers_2019/"
## [6] "https://www.rottentomatoes.com/m/the_lion_king_2019/"
```

(b) Write a function to read the content of one page and pull out the title, tomatometer score and audience score of the film. Then iterate over the vector of all movies using `do.call() / rbind() / lapply()` or `dplyr::bind_rows() / purrr::map()` to create a three column data frame (or tibble).

Display the first six lines of your data frame.

```
read_tomatoes <- function(tomato_url){
  movie_page <- read_html(tomato_url)
  tomato_title <- html_nodes(movie_page,
                             ".mop-ratings-wrap__title--top") %>% html_text()
  tomato_meter <- html_node(movie_page,
                            "#tomato_meter_link .mop-ratings-wrap__percentage") %>%
    html_text(trim = T) %>% str_extract("[0-9]+")
  audience_meter <- html_node(movie_page,
                              ".audience-score .mop-ratings-wrap__percentage") %>%
    html_text(trim = T) %>% str_extract("[0-9]+")
  return(tibble(title=tomato_title,
                tomatometer=as.numeric(tomato_meter)/100,
                audience_score=as.numeric(audience_meter)/100))
}
  top_movies <- tibble()
for(i in movie_links_full){
  top_movies <- bind_rows(top_movies, read_tomatoes(i))
}
top_movies[1:6,]
```

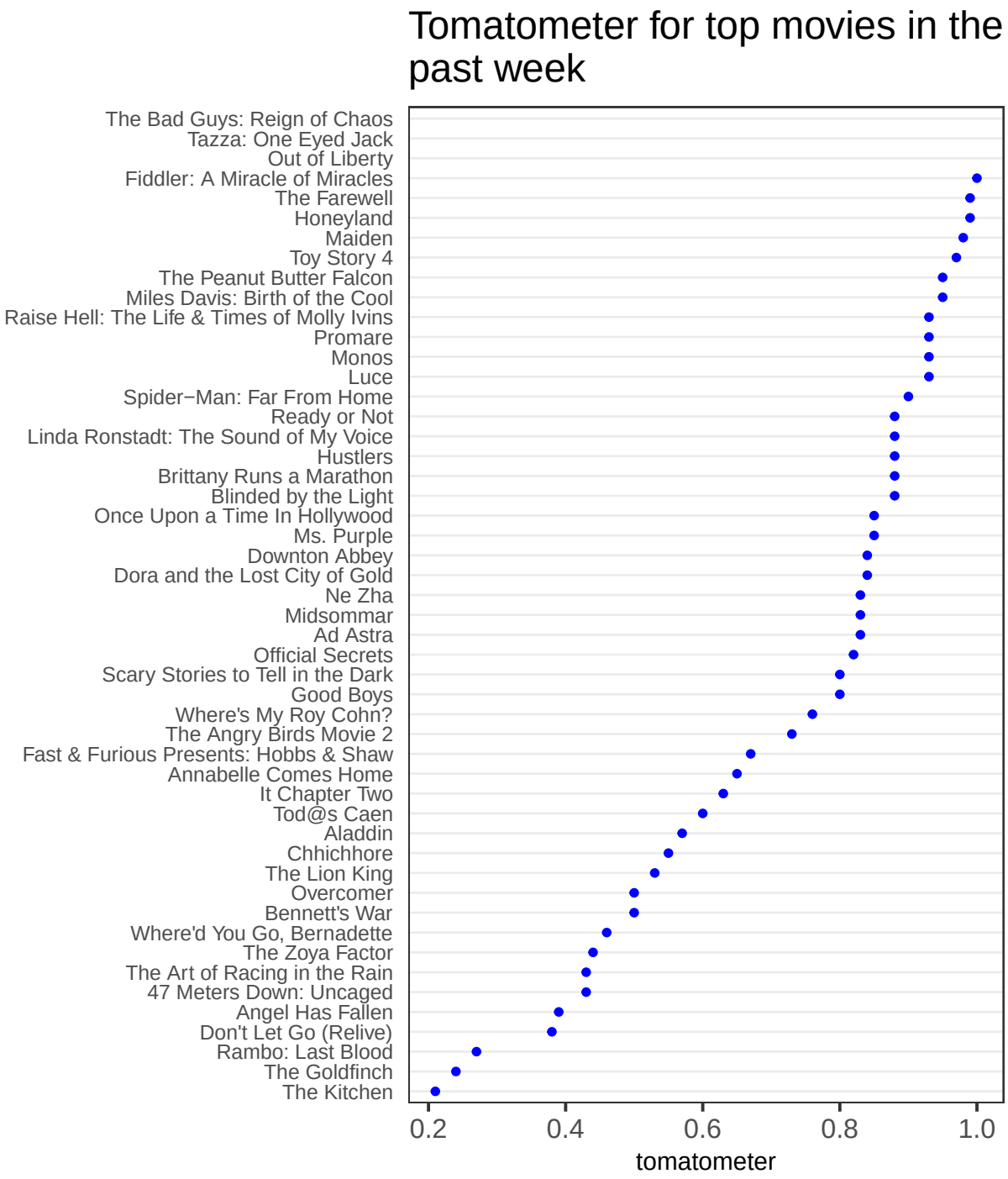
```
## # A tibble: 6 x 3
##   title             tomatometer audience_score
##   <chr>                   <dbl>          <dbl>
## 1 Downton Abbey            0.84           0.95
## 2 Ad Astra                 0.83           0.42
## 3 Rambo: Last Blood        0.27           0.83
## 4 It Chapter Two           0.63           0.78
```

```
## 5 Hustlers                0.88           0.66
## 6 The Lion King           0.53           0.88
```

```
write_csv(top_movies, "190925_top_movies.csv")
```

(c) Create a Cleveland dot plot of tomatometer scores.

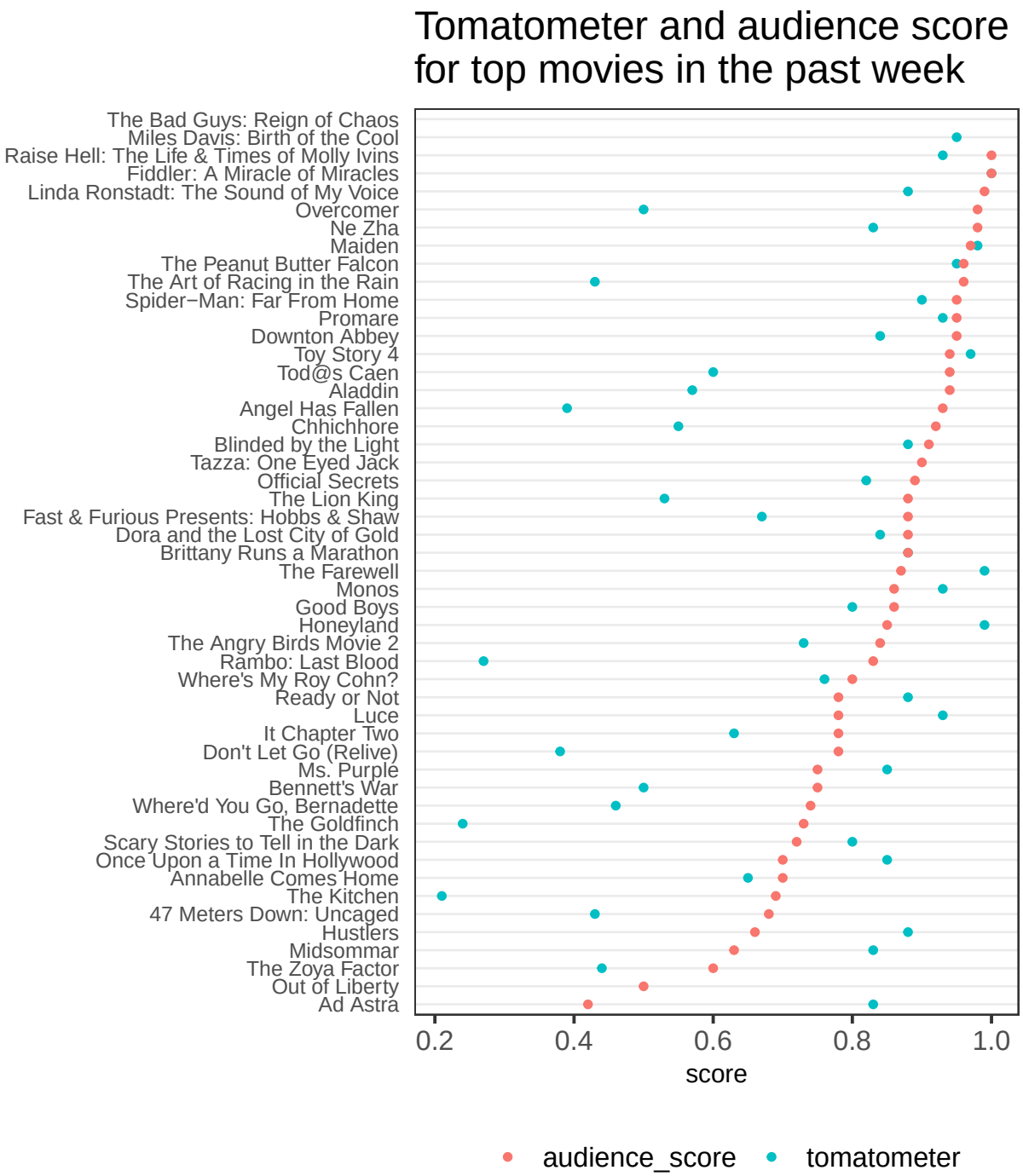
```
theme_dotplot <- theme_bw(16) +
  theme(axis.text.y = element_text(size = rel(.75)),
        axis.ticks.y = element_blank(),
        axis.title.x = element_text(size = rel(.75)),
        panel.grid.major.x = element_blank(),
        panel.grid.major.y = element_line(size = 0.5),
        panel.grid.minor.x = element_blank())

ggplot(top_movies) +
  geom_point(aes(x=tomatometer, y=reorder(title,tomatometer)), color="blue") +
  labs(y=element_blank(),
       title="Tomatometer for top movies in the\npast week") +
  theme_dotplot
```



(d) Create a Cleveland dot plot of tomatometer *and* audience scores on the same graph, one color for each. Sort by audience score.

```
top_movies %>% gather(key="score_type", value="score", -title) %>%
  ggplot() +
  geom_point(aes(x=score,
                 y=fct_reorder2(title,score_type=="audience_score",score,.desc = F),
                 color=score_type)) +
  labs(y=element_blank(),
```

```
      title="Tomatometer and audience score\nfor top movies in the past week") +
theme_dotplot +
theme(legend.position = "bottom", legend.title = element_blank())
```



Note: some movies do not have a score and are shown at the top.

(e) Run your code again for the weekend of July 5 - July 7, 2019. Use **plotly** to create a scatterplot of audience score vs. tomatometer score with the ability to hover over the point to see the film title.

```
paths_allowed("https://www.rottentomatoes.com/browse/box-office/?rank_id=11&country=us")
```

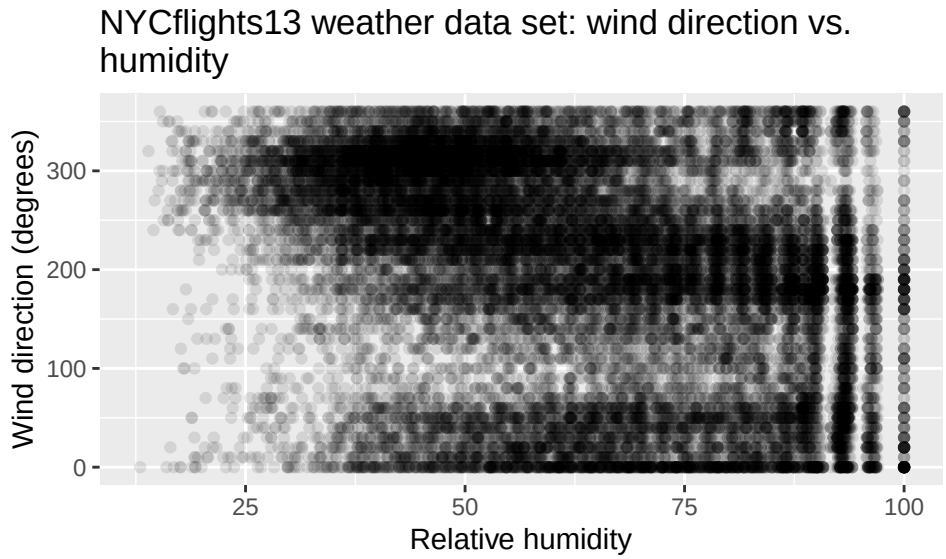
```
## [1] TRUE
```

```
tomato_page_jul <- read_html(
  "https://www.rottentomatoes.com/browse/box-office/?rank_id=11&country=us")
movie_links_html_jul <- html_nodes(tomato_page_jul, ".left a") %>%
  html_attr('href')
movie_links_full_jul <- paste0("https://www.rottentomatoes.com", movie_links_html_jul )

top_movies_jul <- tibble()
for(i in movie_links_full_jul){
  top_movies_jul <- bind_rows(top_movies_jul, read_tomatoes(i))
}

write_csv(top_movies_jul, "190929_top_movies_jul.csv")

movies_scatter <- ggplot(top_movies_jul) +
  geom_point(aes(x=tomatometer, y=audience_score, label=title)) +
  labs(title="Audience score vs. tomatometer - top movies (Jul 5-7)")

ggplotly(movies_scatter) # submitted in separate html file
```

3. Weather

(a) Points with alpha blending

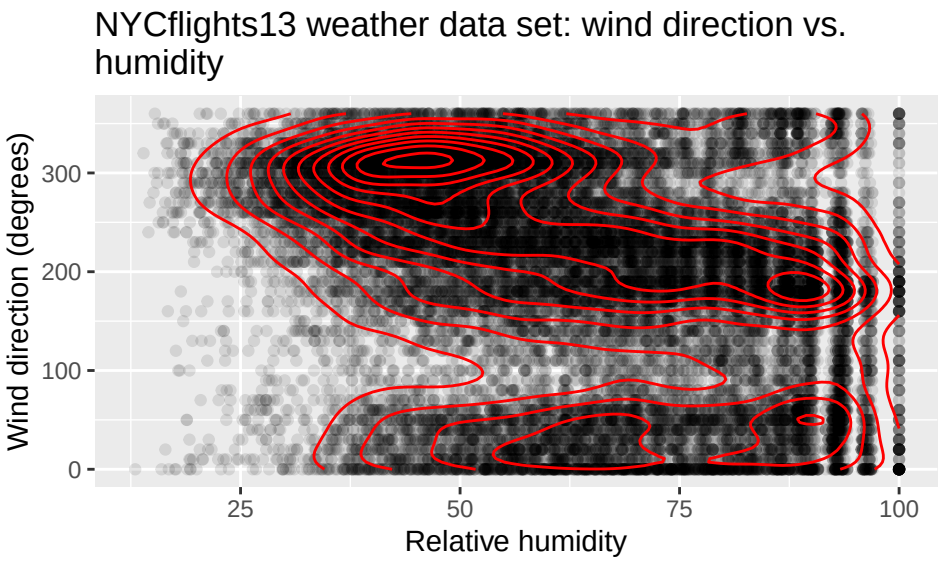
```
ggplot(weather) +
  geom_point(aes(x=humid, y=wind_dir), alpha=0.1) +
  labs(y="Wind direction (degrees)",
       x="Relative humidity",
       title="NYCflights13 weather data set: wind direction vs.\nhumidity")
```



(b) Points with alpha blending + density estimate contour lines

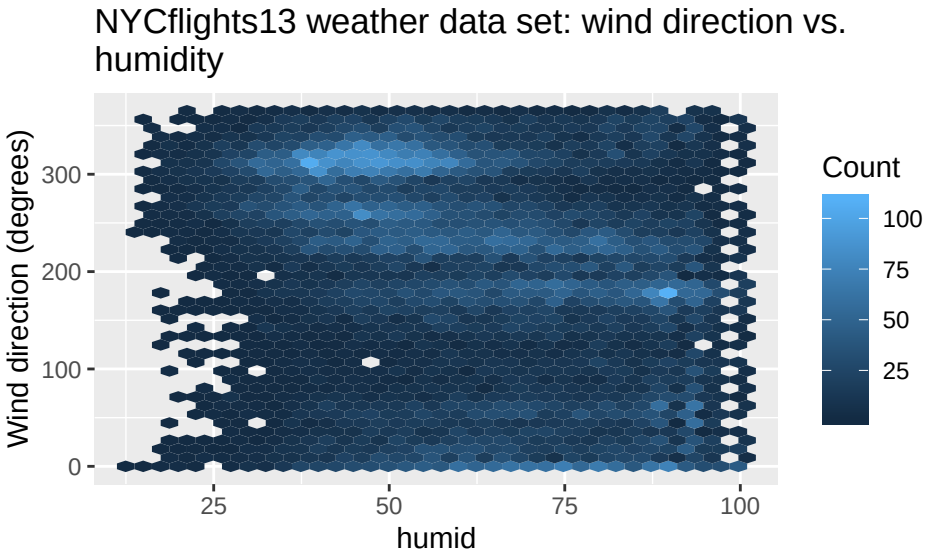
```
ggplot(weather) +
  geom_point(aes(x=humid, y=wind_dir), alpha=0.1) +
```

```
  geom_density_2d(aes(x=humid, y=wind_dir), color="red") +
  labs(y="Wind direction (degrees)",
       x="Relative humidity",
       title="NYCflights13 weather data set: wind direction vs.\nhumidity")
```



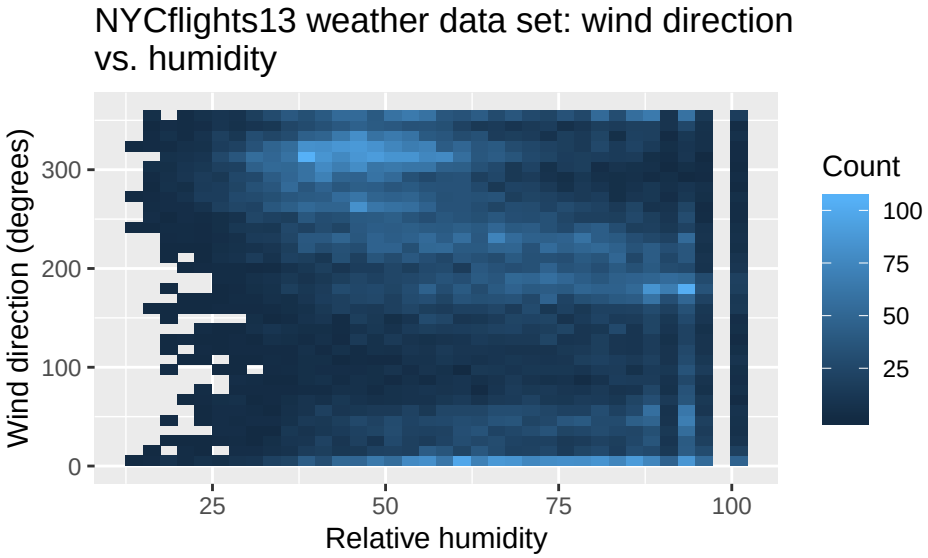
(c) Hexagonal heatmap of bin counts

```
ggplot(weather) +
  geom_hex(aes(x=humid, y=wind_dir), bins=35) +
  labs(y="Wind direction (degrees)",
       X="Relative humidity",
       fill="Count",
       title="NYCflights13 weather data set: wind direction vs.\nhumidity")
```



(d) Square heatmap of bin counts

```
ggplot(weather) +
  geom_bin2d(aes(x=humid, y=wind_dir), bins=35) +
  labs(y="Wind direction (degrees)",
       x="Relative humidity",
       fill="Count",
       title="NYCflights13 weather data set: wind direction\nvs. humidity")
```

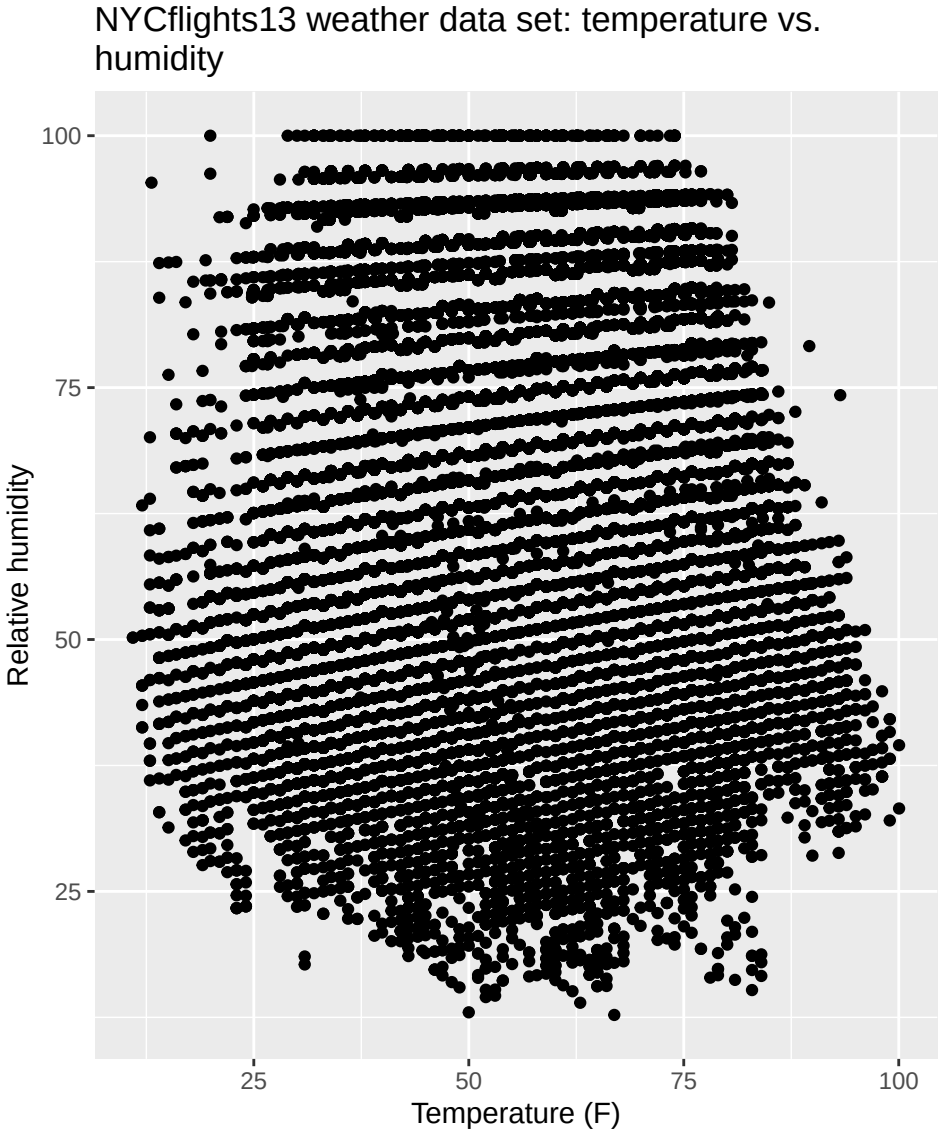


(e) Describe noteworthy features of the data, using the "Movie ratings" example on page 82 (last page of Section 5.3) as a guide.

Wind direction is reported in constant intervals (of 10 degrees). Relative humidity has a gap below 100%. Additionally, humidity is typically greater than 30. When it is less than 30, wind direction is likely to be >200 degrees. There are fewer observations in the 80-150 degree wind direction swath of the chart. There are a few areas of high density - 1) around 50% humidity and 300 degrees wind direction, 2) 90% humidity and 190 degrees, 3) 90% humidity and 50 degrees, 4) 60% humidity and 25 degrees.

(f) Draw a scatterplot of `humid` vs. `temp`. Why does the plot have diagonal lines?

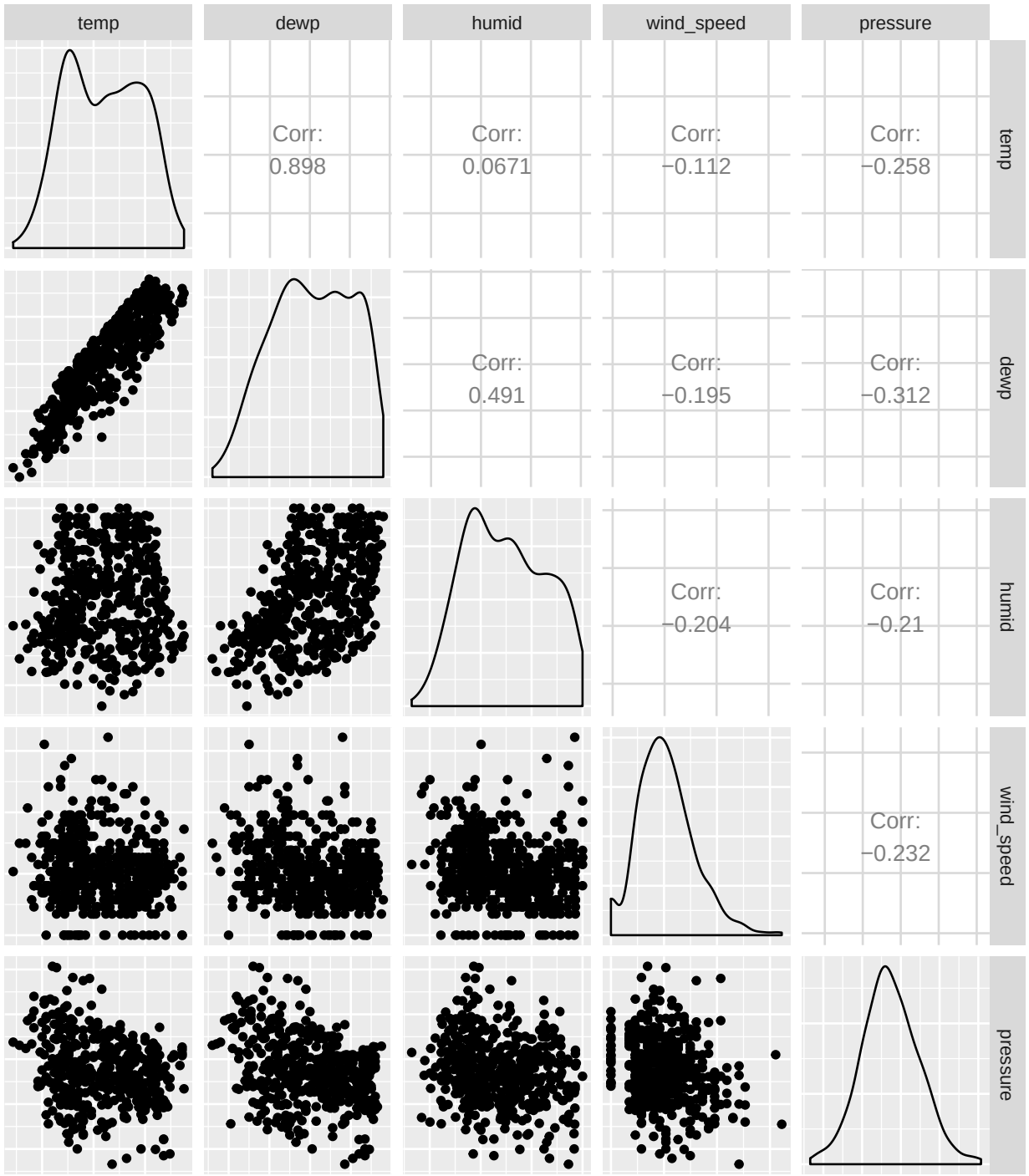
```
ggplot(weather) +
  geom_point(aes(x=temp, y=humid)) +
  labs(x="Temperature (F)",
       y="Relative humidity",
       title="NYCflights13 weather data set: temperature vs.\nhumidity")
```



Each observation in the weather data set represents a recording for a given hour (at an airport). Therefore, if we look at a set of observations for one airport for one day, we are likely to see a linear relationship between temp and humidity. This linear relationship is reflected in the data by the diagonal lines.

(g) Draw a scatterplot matrix of the continuous variables in the `weather` dataset. Which pairs of variables are strongly positively associated and which are strongly negatively associated?

```
col_sel <- c("temp", "dewp", "humid", "wind_speed", "pressure")
row_sel <- sample(nrow(weather),500)

ggpairs(weather[row_sel,col_sel], axisLabels = "none")
```



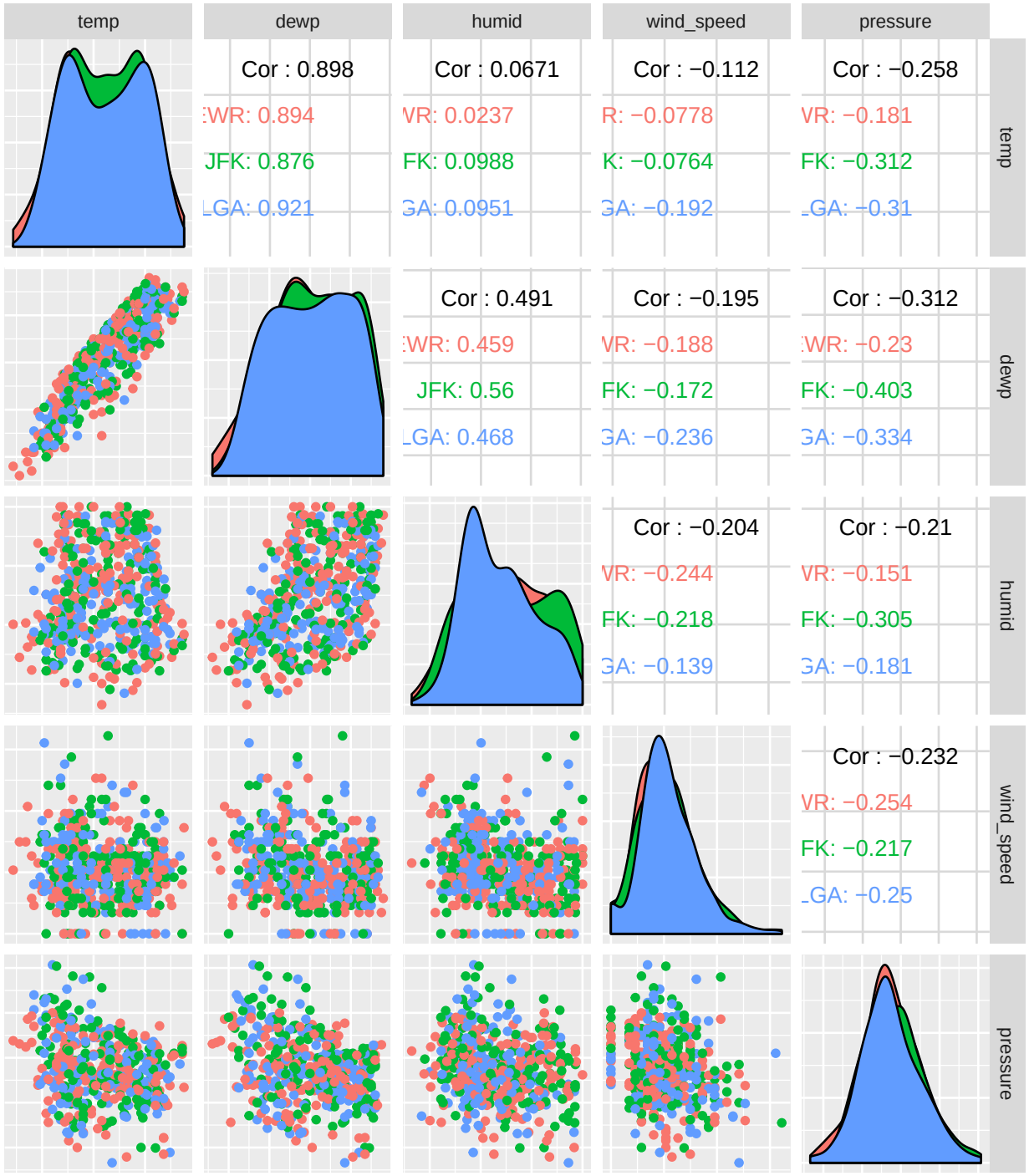
Strongly positively associated: temp/dewp, dewp/humid

Strongly negatively associated: No variables are stongly negatively associated, pressure vs. temp/dewp/humid/wind_speed and wind_speed vs. temp/dewp/humid are all slightly negatively associated.

Note: random sample of 500 rows used to make it easier to see trends. Exluded wind direction since discrete (in segments of 10 degrees), wind_gust since most values are NA, precip since >90% of values are 0, and visib since it is discrete and maxes out at 10.

(h) Color the points by `origin`. Do any new patterns emerge?

```
ggpairs(weather[row_sel,], columns = col_sel,
        axisLabels = "none", mapping=aes(color=origin))
```



No new patterns appear to emerge.