EDAV Fall 2019 PSet 2

Chao Huang (ch3474) and Kliment Mamykin (km2770)

Read *Graphical Data Analysis with R*, Ch. 4, 5

Grading is based both on your graphs and verbal explanations. Follow all best practices as discussed in class. Data manipulation should not be hard coded. That is, your scripts should be written to work for new data.

1. useR2016! survey

[18 points]

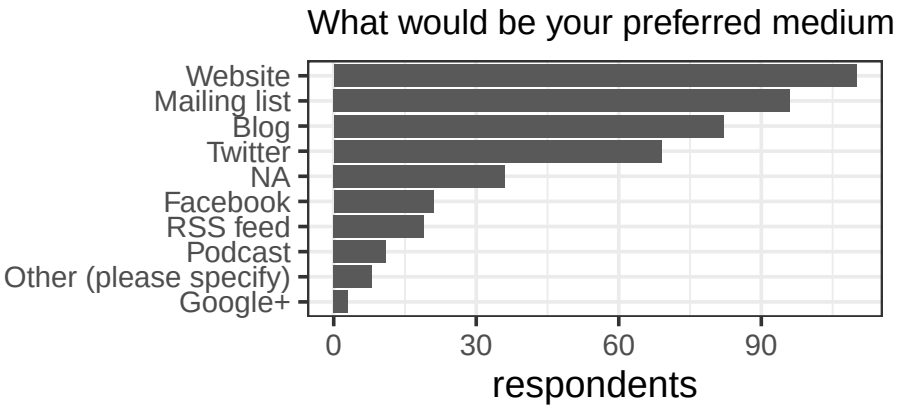
Data: `useR2016` dataset in the **forwards** package (available on CRAN)

For parts (a) and (b):

- Do not toss NAs.
- Do some research to find the wording of the questions asked as relevant and include them in the titles of your graphs.
- Include the dataset name, package name, and link to the question wording source in the graph caption.

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

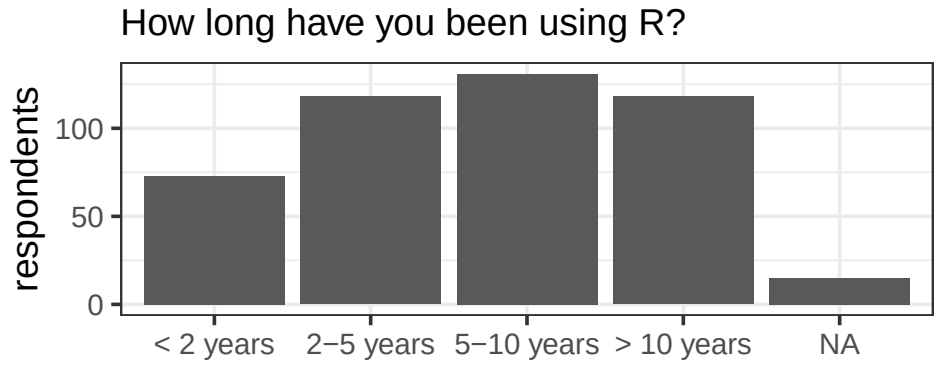
```
useR2016 %>%
  group_by(Q20) %>%
  summarise(Counts = n()) %>%
  ungroup() %>%
  mutate(Q20 = ifelse(is.na(Q20), "NA", as.character(Q20))) %>%
  ggplot(aes(fct_reorder(Q20, Counts), Counts)) +
    geom_col() +
    coord_flip() +
    theme_bw(14) +
    theme(
      plot.title = element_text(size = rel(0.9))
    ) +
    labs(title = "What would be your preferred medium for R community news?",
       caption = "Source\ndataset: userR2016
       package: forwards
       url: https://www.rdocumentation.org/packages/forwards/versions/0.1.1/topics/useR2016",
       x = "", y = "respondents")
```



Source
dataset: userR2016
package: forwards
v.rdocumentation.org/packages/forwards/versions/0.1.1/topics/useR2016

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

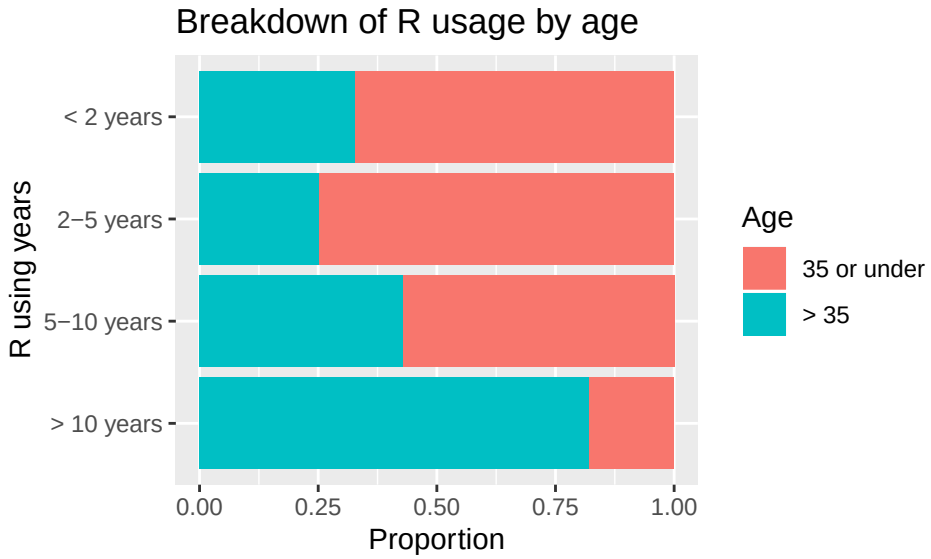
```
useR2016 %>%
  group_by(Q11) %>%
  summarise(Counts = n()) %>%
  ungroup() %>%
  ggplot(aes(fct_relevel(Q11, "> 10 years", after = 3), Counts)) +
    geom_col() +
    theme_bw(14) +
    theme(
      plot.title = element_text(size = rel(1))
    ) +
    labs(title = "How long have you been using R?",
      caption = "Source
      dataset: userR2016
      package: forwards
      url: https://www.rdocumentation.org/packages/forwards/versions/0.1.1/topics/useR2016",
       x = "", y = "respondents")
```



Source
dataset: userR2016
package: forwards
v.rdocumentation.org/packages/forwards/versions/0.1.1/topics/useR2016

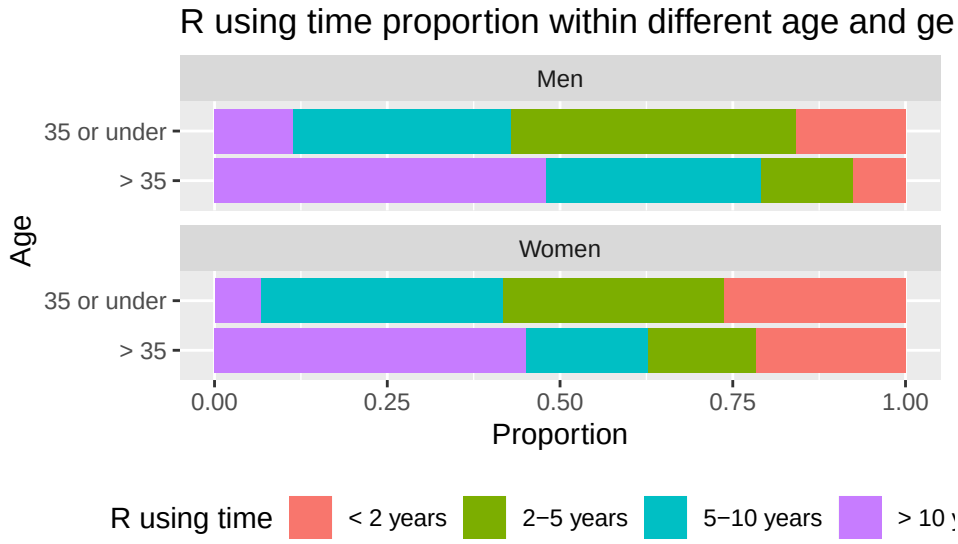
(c) Create a horizontal stacked bar chart showing the proportion of respondents for each level of Q11 who are over 35 vs. 35 or under. Use a descriptive title.

```
stack_Q11 <- useR2016 %>% select(Q3, Q11) %>%
  filter(! is.na(Q3) & ! is.na(Q11))

ggplot(stack_Q11, aes(x=fct_rev(Q11))) +
  geom_bar(aes(y=(..count..), fill = fct_rev(Q3)), position = "fill") +
  coord_flip() +
  scale_y_continuous(labels = scales::number_format(accuracy = 0.01)) +
  ylab("Proportion") +
  xlab("R using years") +
  labs(fill = "Age",
      title = "Breakdown of R usage by age")
```



(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.

```
stack_Q3 <- useR2016 %>% select(Q2, Q3, Q11) %>%
  filter(! is.na(Q11) & ! is.na(Q3)) %>%
  filter(Q2 != "Non-Binary/Unknown")

ggplot(stack_Q3, aes(x=Q3)) +
  geom_bar(aes(y=(..count..),
               fill=Q11), position = "fill") +
  facet_wrap(~ Q2, nrow = 2) +
  coord_flip() +
  xlab("Age") +
  ylab("Proportion") +
  labs(fill = "R using time",
      title = paste("R using time proportion within",
                "different age and gender groups")) +
  theme(legend.position = 'bottom',
        legend.direction = "horizontal",
        legend.box = "horizontal")
```



(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_line_breaks <- function(vec_str, len) {
  add_newline <- function(str, len) {
    space_loc <- str_locate_all(str, "[ \t]+")[[1]][,1]
    buffer <- 0
    last_loc <- 0
    for (loc in space_loc) {
      buffer <- buffer + loc - last_loc
      if (buffer > len) {
        substr(str, loc, loc) <- "\n"
        buffer <- 0
      }
    last_loc <- loc
    }
    str
  }
  unlist(map(vec_str, function(str) add_newline(str, len)))
}
```

A simple demo,

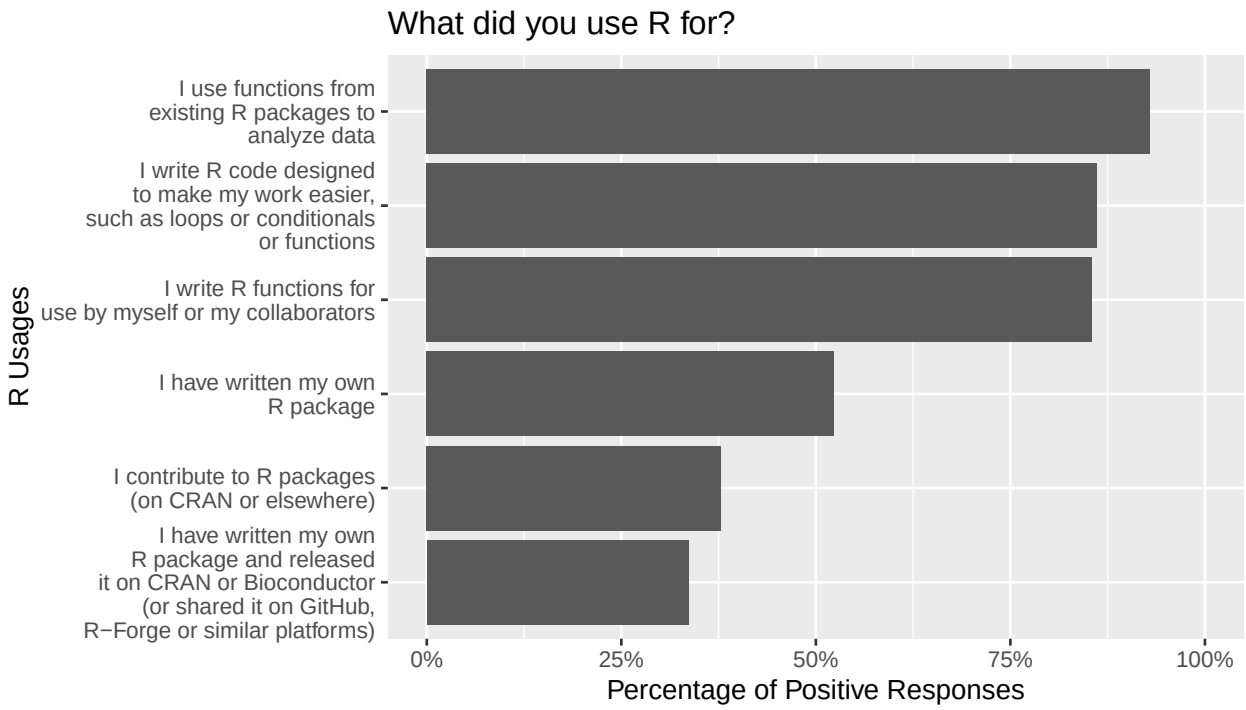
```
long_str <- paste("We hold these truths to be self-evident, that all ",
                  "men are created equal, that they are endowed by their",
                  "Creator with certain unalienable Rights, that among these ",
                  "are Life, Liberty and the pursuit of Happiness.")
cat(add_line_breaks(long_str, 50))
```

```
## We hold these truths to be self-evident, that all  men
## are created equal, that they are endowed by their Creator
## with certain unalienable Rights, that among these  are
## Life, Liberty and the pursuit of Happiness.
```

(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`.

```
base_df <- useR2016 %>% select(matches("Q13"))

pos_response <- base_df %>%
  summarise_all(.funs = function(x) levels(factor(x))) %>%
  tidyr::gather(key="question", value="posResponse", .)

countNA_Q13 <- base_df %>%
  summarise_all(.funs = function (x) sum(is.na(x))) %>%
  tidyr::gather(key="question", value="naCount", .)

pos_count <- inner_join(x = pos_response, y = countNA_Q13, "question") %>%
  mutate(percentage= 1 - naCount / nrow(base_df)) %>%
  mutate(folded=factor(add_line_breaks(posResponse, 20)))
```

```
ggplot(pos_count) +
  geom_col(aes(x=fct_reorder(folded, percentage),
               y=percentage)) +
  coord_flip() +
  xlab("R Usages") +
  ylab("Percentage of Positive Responses") +
  scale_y_continuous(labels = scales::percent, limits = c(0,1)) +
  labs(title = "What did you use R for?")
```



2. Rotten Tomatoes

[18 points]

To get the data for this problem, we'll use the **robotstxt** package to check that it's ok to scrape data from Rotten Tomatoes and then use the **rvest** package to get data from the web site.

(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(paths = "https://www.rottentomatoes.com/browse/box-office/", warn = TRUE)
```

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

```
get_all_links_by_weekend <- function (weekend="Last Weekend") {
  webpage <- read_html("https://www.rottentomatoes.com/browse/box-office/")
  select_forms <- webpage %>%
    html_nodes("select") # two forms are exactly the same
  form <- select_forms[[1]]
  all_options <- form %>% html_nodes("option")

  weekends <- map(all_options, function (opt) html_text(opt))
  rel_links <- map(all_options, function (opt) html_attr(opt, "value"))

  squish_lower_case <- function(str) {
    str %>% str_replace_all(pattern = " ", replacement = "") %>%
      tolower()
  }

  weekends <- map(weekends, squish_lower_case)
  weekend <- squish_lower_case(weekend)

  match_idx <- match(weekend, weekends)
  if(is.na(match_idx)) {
    sprintf("All available choices: %s\n", str(weekends))
    sprintf("But your choice %s is not among them!", weekend)
    stop()
  }
  chosen_weekend_link <- rel_links[[match_idx]]
  sprintf("The link to this chosen weekend: %s", chosen_weekend_link)

  home_link <- "https://www.rottentomatoes.com"
  webpage <- read_html(paste(home_link, chosen_weekend_link, sep = ""))
  elems <- webpage %>% html_nodes("td a")
  links <- purrr::map(elems, function (x)
    paste("https://www.rottentomatoes.com", html_attr(x, "href"), sep = ""))
  links <- unlist(links)
}
```

```
links <- get_all_links_by_weekend()
head(links, 6)
```

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

(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.

(Results will vary depending on when you pull the data.)

For help, see this SO post: https://stackoverflow.com/questions/36709184/build-data-frame-from-multiple-rvest-elements

```
download_contents <- function(movie_urls) {
  pager <- function(page) {
      doc <- read_html(url(page))
      data.frame(
          title = doc %>% html_node("div.score_panel > h1") %>% html_text() %>% as.character,

          tomatometer = doc %>% html_node("#tomato_meter_link") %>%
            html_text() %>% str_trim() %>% str_replace("%","") %>% as.double,

          audience_score = doc %>% html_node("div.audience-score > h2") %>%
            html_text() %>% str_trim() %>% str_replace("%","") %>% as.double,

          url = page,
          stringsAsFactors=FALSE
      )
  }
  return(do.call(rbind, lapply(movie_urls, pager)))
}
```

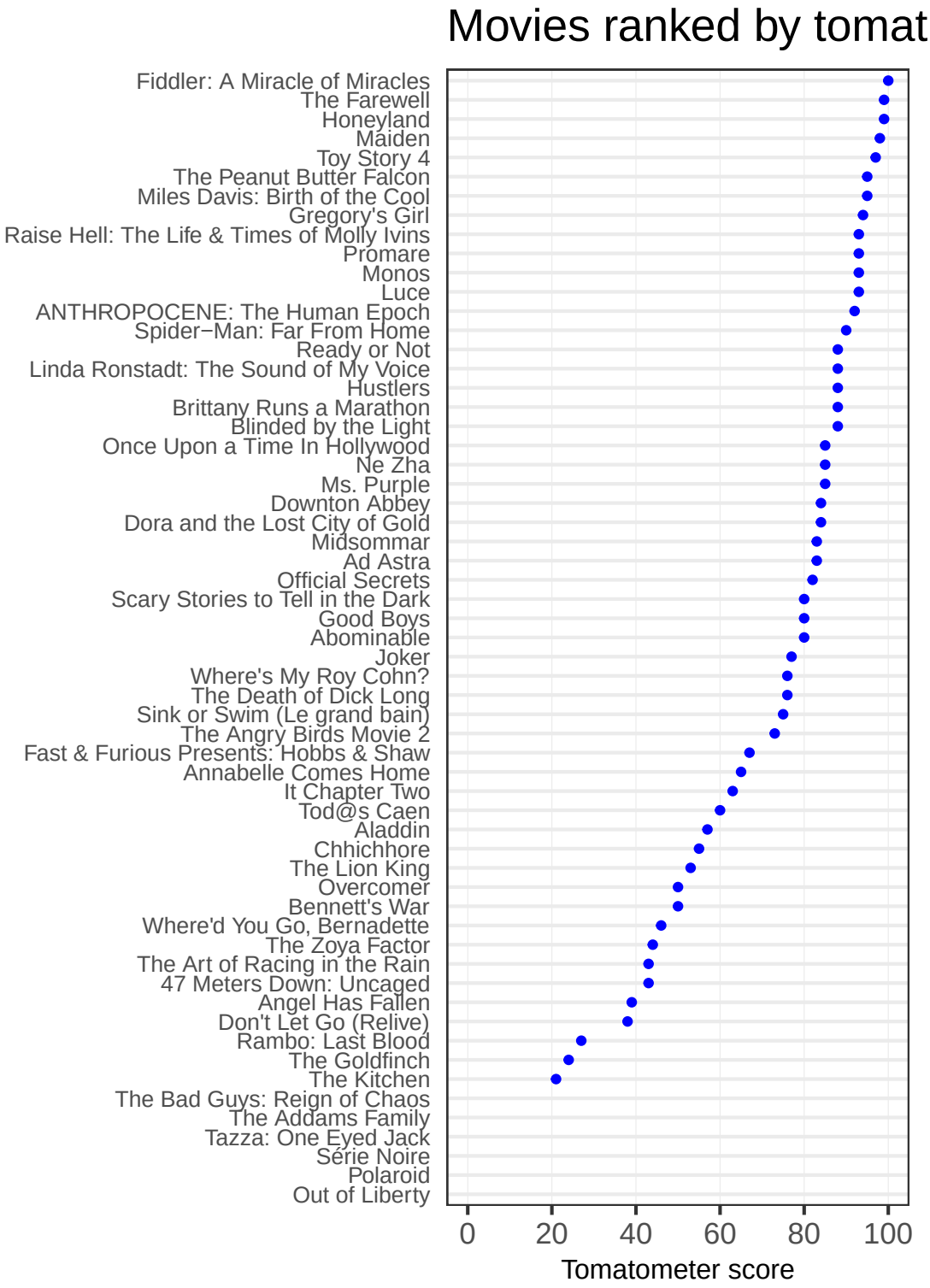
```
rottentomatoes.get_movies <- function(file_name="./rottentomatoes_movies.csv", links) {

  movies <- if(file.exists(file_name)){
    read.csv(file_name)}
  else {
    download_contents(links)}

  if (!file.exists(file_name)) {
    write.csv(movies, file_name, row.names=FALSE)
  }
  movies
}
```

```
movies <- rottentomatoes.get_movies(links = links)
head(movies, 6)
```

```
##               title tomatometer audience_score
## 1     Downton Abbey          84             95
## 2          Ad Astra          83             42
## 3 Rambo: Last Blood          27             83
## 4    It Chapter Two          63             78
## 5          Hustlers          88             66
## 6     The Lion King          53             88
##                                                  url
## 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/
```

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

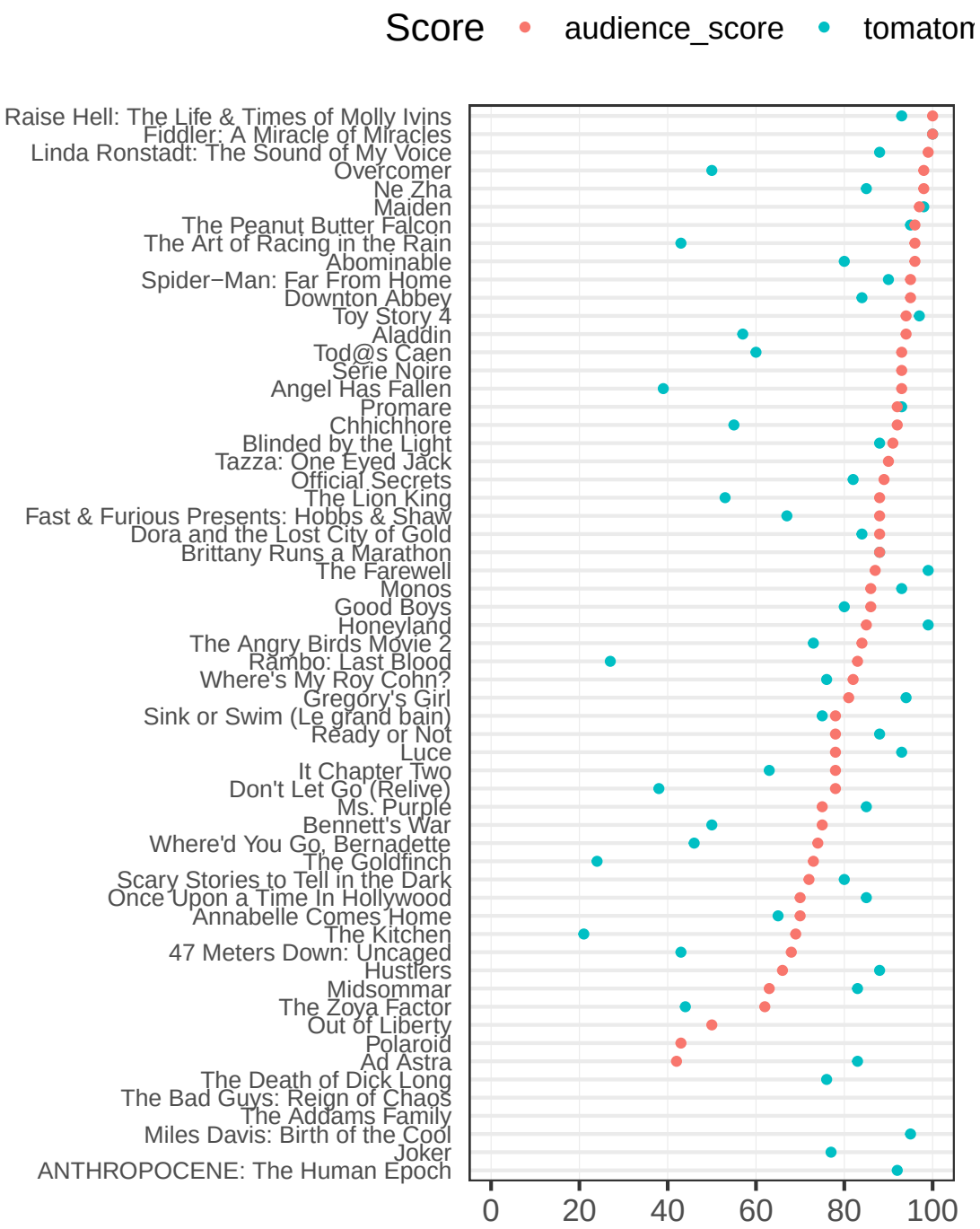
```r
# create a theme for dot plots, which can be reused
theme_dotplot <- theme_bw(16) +
    theme(
        plot.title = element_text(size = rel(1.25)),
        axis.text.y = element_text(size = rel(0.8)),
        axis.ticks.y = element_blank(),
        axis.title.x = element_text(size = rel(.75)),
        panel.grid.major.x = element_line(size = 0.2),
        panel.grid.major.y = element_line(size = 0.75),
        panel.grid.minor.x = element_blank()
    )

# create the plot
movies %>%
  # Replace NA score with -1, which causes those movies to be sorted at the bottom of the plot
  # The value of -1 is not shown since we wet the scale for x to be [0,100]
  mutate(tomatometer=ifelse(is.na(tomatometer), -1, tomatometer)) %>%
  ggplot(aes(x = tomatometer, y = fct_reorder(title, tomatometer))) +
    geom_point(color = "blue") +
    scale_x_continuous(limits = c(0, 100), breaks = seq(0, 100, 20)) +
    theme_dotplot +
    xlab("Tomatometer score") +
    ylab("") +
    ggtitle("Movies ranked by tomatometer score")
```



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

```
movies %>%
  # Replace NA score with -1, which causes those movies to be sorted at the bottom of the plot
  # The value of -1 is not shown since we wet the scale for x to be [0,100]
  mutate(tomatometer=ifelse(is.na(tomatometer), -1, tomatometer)) %>%
```

```
  mutate(audience_score=ifelse(is.na(audience_score), -1, audience_score)) %>%
  gather(key = "Score", value = "value", tomatometer, audience_score) %>%
  ggplot(aes(x = value,
             y = fct_reorder2(title, Score=='audience_score', value, .desc = FALSE),
             color = Score)) +

  geom_point() +
  scale_x_continuous(limits = c(0, 100), breaks = seq(0, 100, 20)) +
  theme_dotplot +
  theme(legend.position = "top") +
  xlab("") +
  ylab("") +
  ggtitle("Movies scores")
```



(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.

```
movies_july05_july07 <- rottentomatoes.get_movies(
  file_name = "./movies_jul5_jul7.csv",
  links = get_all_links_by_weekend("Jul 05 - Jul 07"))
```

Because the PDF file can not show the interactive plot, the answer is rendered as a separate file Q2e.html

and submitted with the assignment.

3. Weather

[14 points]

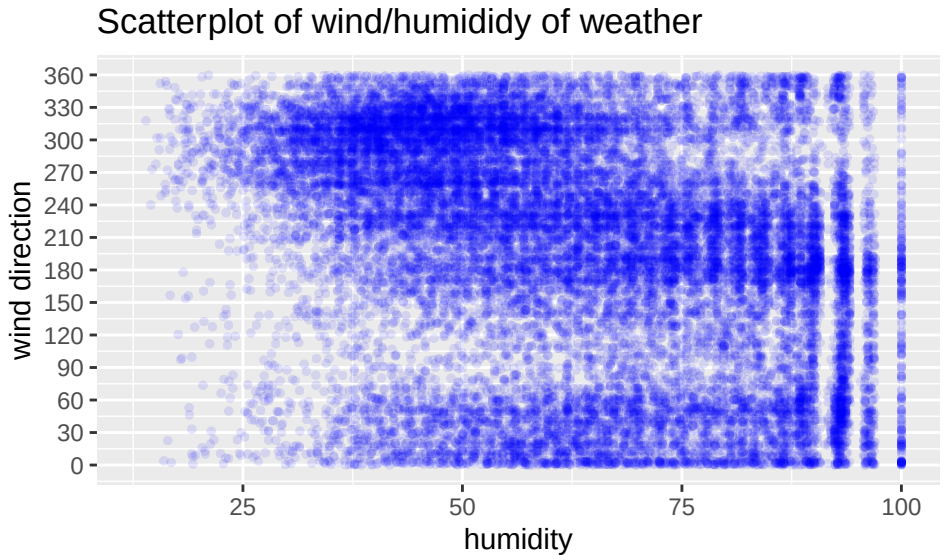
Data: `weather` dataset in **nycflights13** package (available on CRAN)

```
library(nycflights13)
```

For parts (a) - (d) draw four plots of `wind_dir` vs. `humid` as indicated. For all, adjust parameters to the levels that provide the best views of the data.

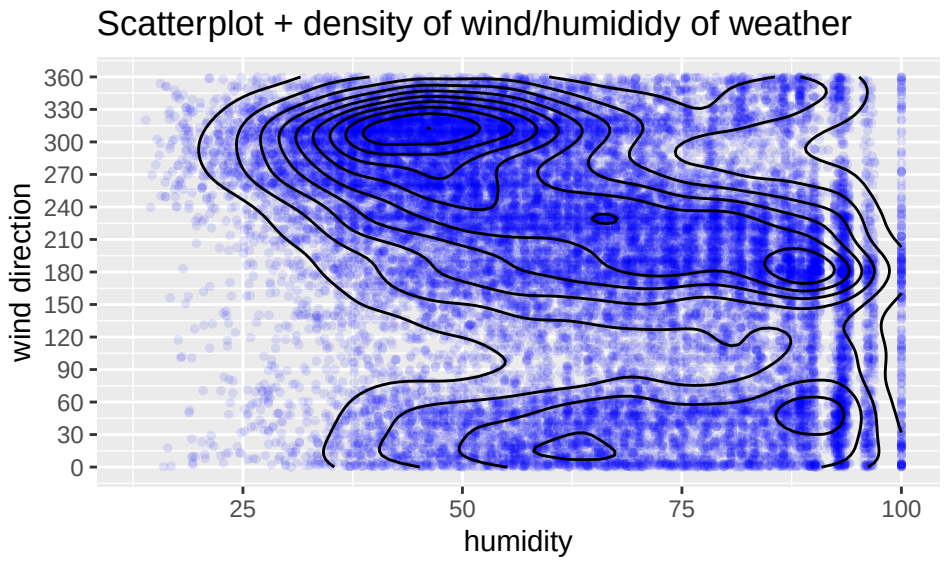
(a) Points with alpha blending

```
weather %>%
ggplot(aes(y=wind_dir, x=humid)) +
geom_point(color = "blue", size = 1.5, alpha = .1, stroke = 0, position = "jitter") +
scale_y_continuous(limits = c(0, 360), breaks = seq(0, 360, 30)) +
ylab("wind direction") +
xlab("humidity") +
ggtitle("Scatterplot of wind/humididy of weather")
```



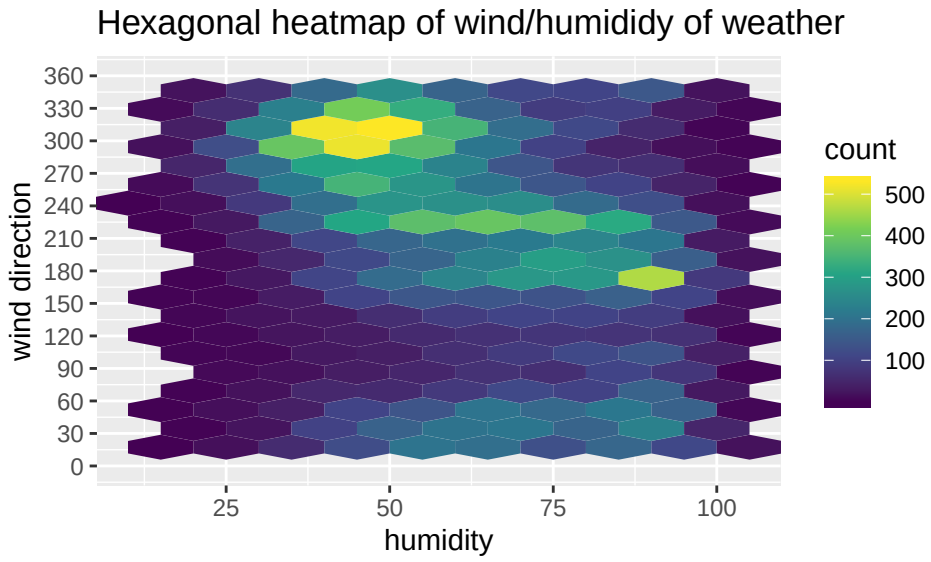
(b) Points with alpha blending + density estimate contour lines

```
weather %>%
  ggplot(aes(y=wind_dir, x=humid)) +
  geom_point(color = "blue", size = 1.5, alpha = .1, stroke = 0, position = "jitter") +
  scale_y_continuous(limits = c(0, 360), breaks = seq(0, 360, 30)) +
  geom_density2d(bins = 10, color = "black") +
  ylab("wind direction") +
  xlab("humidity") +
  ggtitle("Scatterplot + density of wind/humididy of weather")
```



(c) Hexagonal heatmap of bin counts

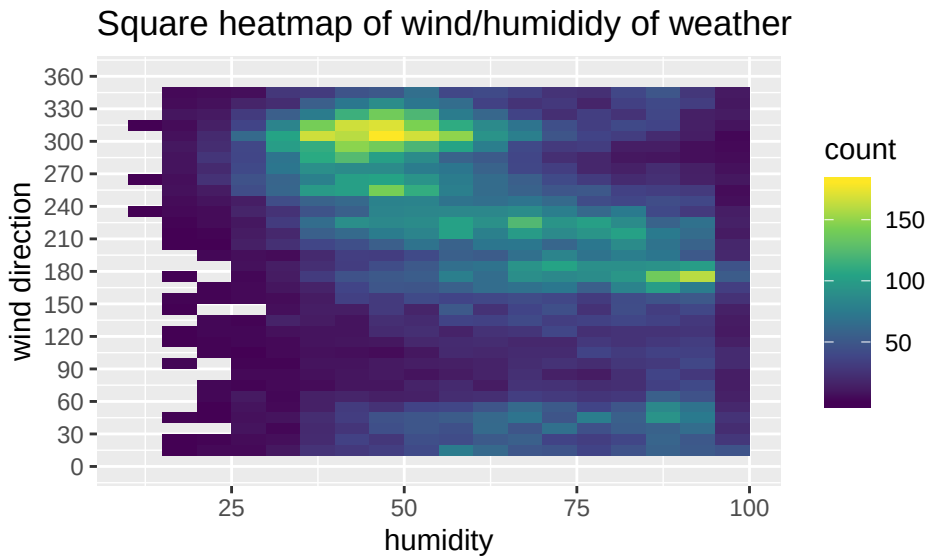
```
weather %>%
  ggplot(aes(y=wind_dir, x=humid)) +
  scale_y_continuous(limits = c(0, 360), breaks = seq(0, 360, 30)) +
  geom_hex(binwidth = c(10, 20)) +
  scale_fill_viridis_c() +
  ylab("wind direction") +
  xlab("humidity") +
  ggtitle("Hexagonal heatmap of wind/humididy of weather")
```



(d) Square heatmap of bin counts

```
weather %>%
  ggplot(aes(y=wind_dir, x=humid)) +
  scale_y_continuous(limits = c(0, 360), breaks = seq(0, 360, 30)) +
  geom_bin2d(binwidth = c(5,10)) +
  scale_fill_viridis_c() +
  ylab("wind direction") +
```

```
  xlab("humidity") +
  ggtitle("Square heatmap of wind/humididy of weather")
```

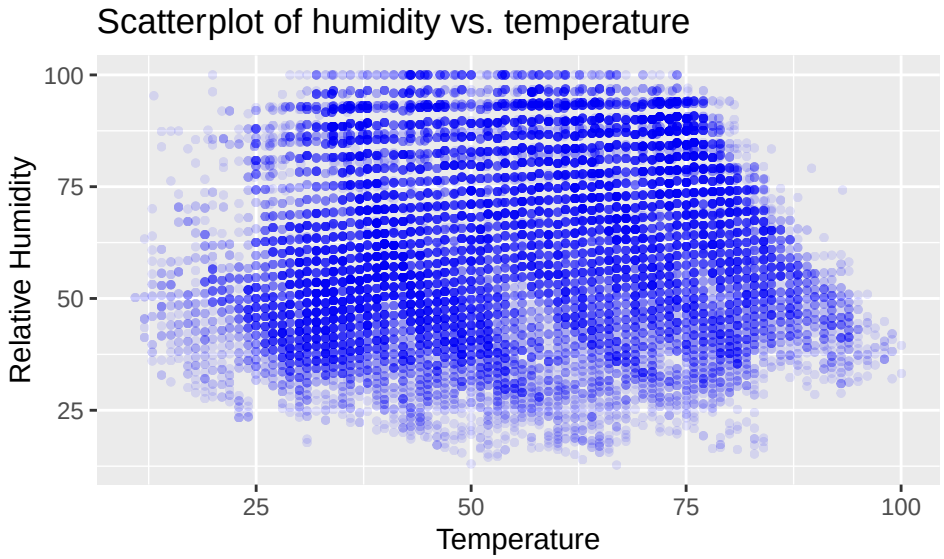


(e) Describe noteworthy features of the data, using the "Movie ratings" example on page 82 (last page of Section 5.3) as a guide.

1. The wind seldom blew from the southeast (90 to 180 degrees) side.
2. When the wind blew from the northeast side (0 to 90 degrees) or southwest side (180 to 270 degrees), the relative humidity seemed to be high. When the wind blew from the northwest side (270 to 360 degrees), the relative humidity would be low and mainly distributed around 40%.
3. The relative humidty was usually higher than 20% no matter which direction the wind blew from.
4. When the wind blew from the north and the south, the humidity would be more likely to reach 100%, meaning that it would rain.
5. There was no deterministic relationship between the wind direction and humidity, no matter where the wind blew from, the relative humidity might cover the whole range from 0 to 1.

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

```
ggplot(weather, aes(x=temp, y=humid)) +
  geom_point(color = "blue", alpha = .1, stroke = 0) +
  ylab("Relative Humidity") +
  xlab("Temperature") +
  ggtitle("Scatterplot of humidity vs. temperature")
```

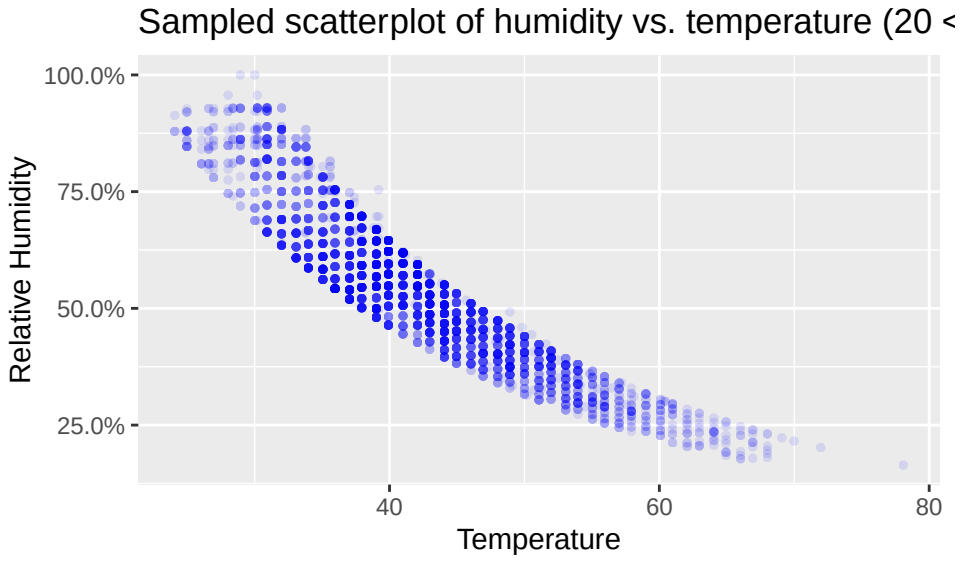


Those diagonal lines going from the left buttom corner to the right upper corner seem to be more siginificant than those going from the left upper corner to the right buttom corner. However, we don't find any reasonable explanations for them, since according in to physical laws, when the dewp is fixed, the relative humidity should have a negative correlation with the temperature. The appearance of these lines may come from the round-off error of the measurements. We will try explaning the diagonal lines going from the left upper corner to the right buttom corner in the following.

The reason is that relative humidity can be calculated using a approximate function whose inputs are temperature and dewp (dew point). The exact formula can be found in this link http://bmcnoldy.rsmas.miami.edu/Humidity.html. Given a dewp, the relative humidity has a negative correlation with temperature. So all these diagonal lines come from the left-upper side to the right-bottom side.

It's more easily to observe the diagonal lines if we sample the dataset by letting the "dewp" variable fall into a certain range, e.g., from 20 to 30.

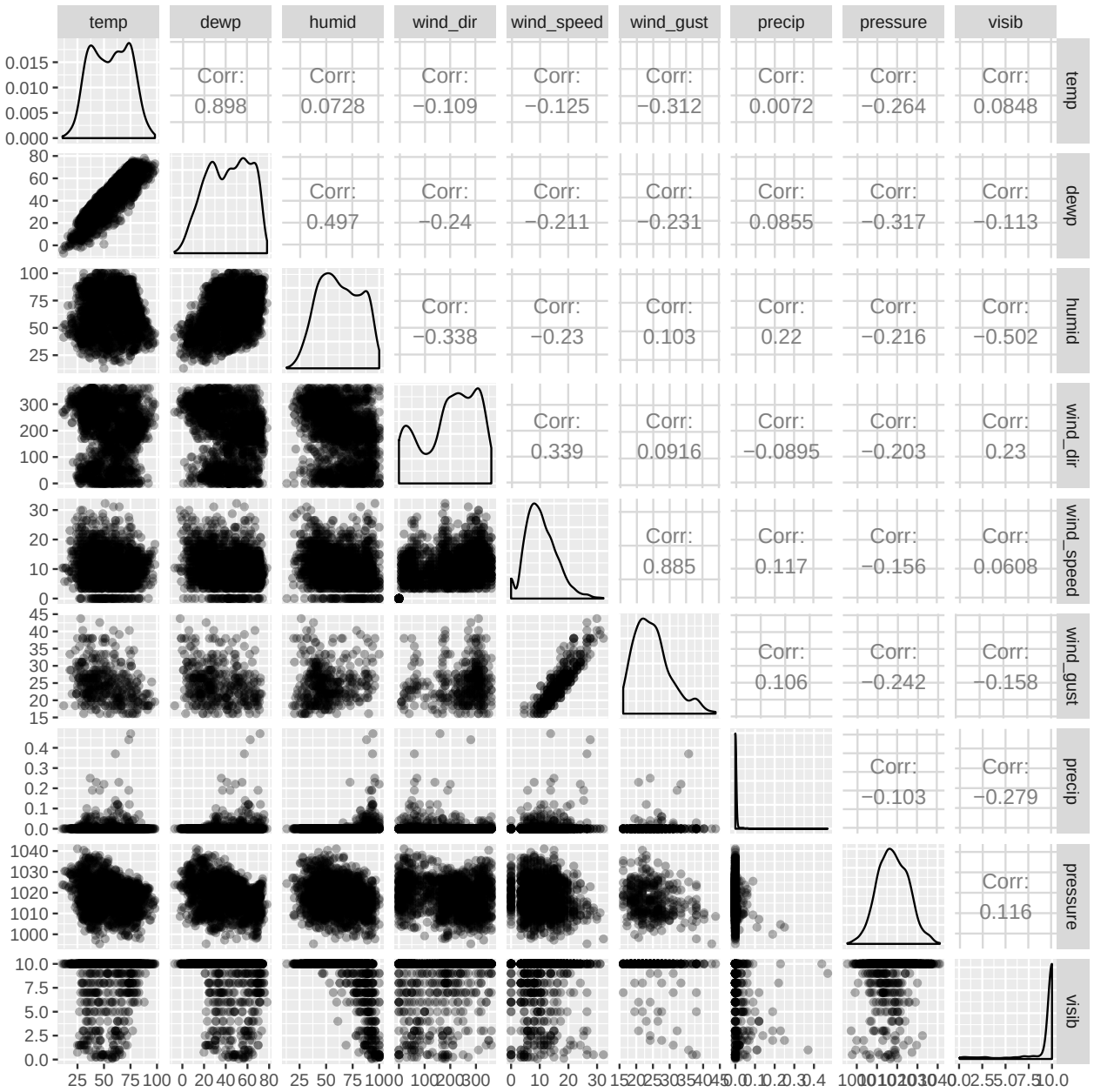
```
ggplot(weather %>% filter(dewp > 20 & dewp < 30),
       aes(x=temp, y=humid / 100)) +
  geom_point(color = "blue", alpha = .1, stroke = 0) +
  scale_y_continuous(labels=scales::percent) +
  ylab("Relative Humidity") +
  xlab("Temperature") +
  ggtitle("Sampled scatterplot of humidity vs. temperature (20 < dewp < 30)")
```



(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?

Since there is a outlier with "wind_speed" as large as 1000, we remove it so that we can observe the underlying patterns better. Also we downsample the data to contain only 2000 samples, in order to accelerate the speed of plotting and reveal most informative patterns.

```
pair_matrix <- GGally::ggpairs(weather %>% filter(wind_speed < 1000) %>% sample_n(2000),
                columns = 6:14,
                lower = list(continuous=GGally::wrap("points", alpha = 0.3)))
pair_matrix
```

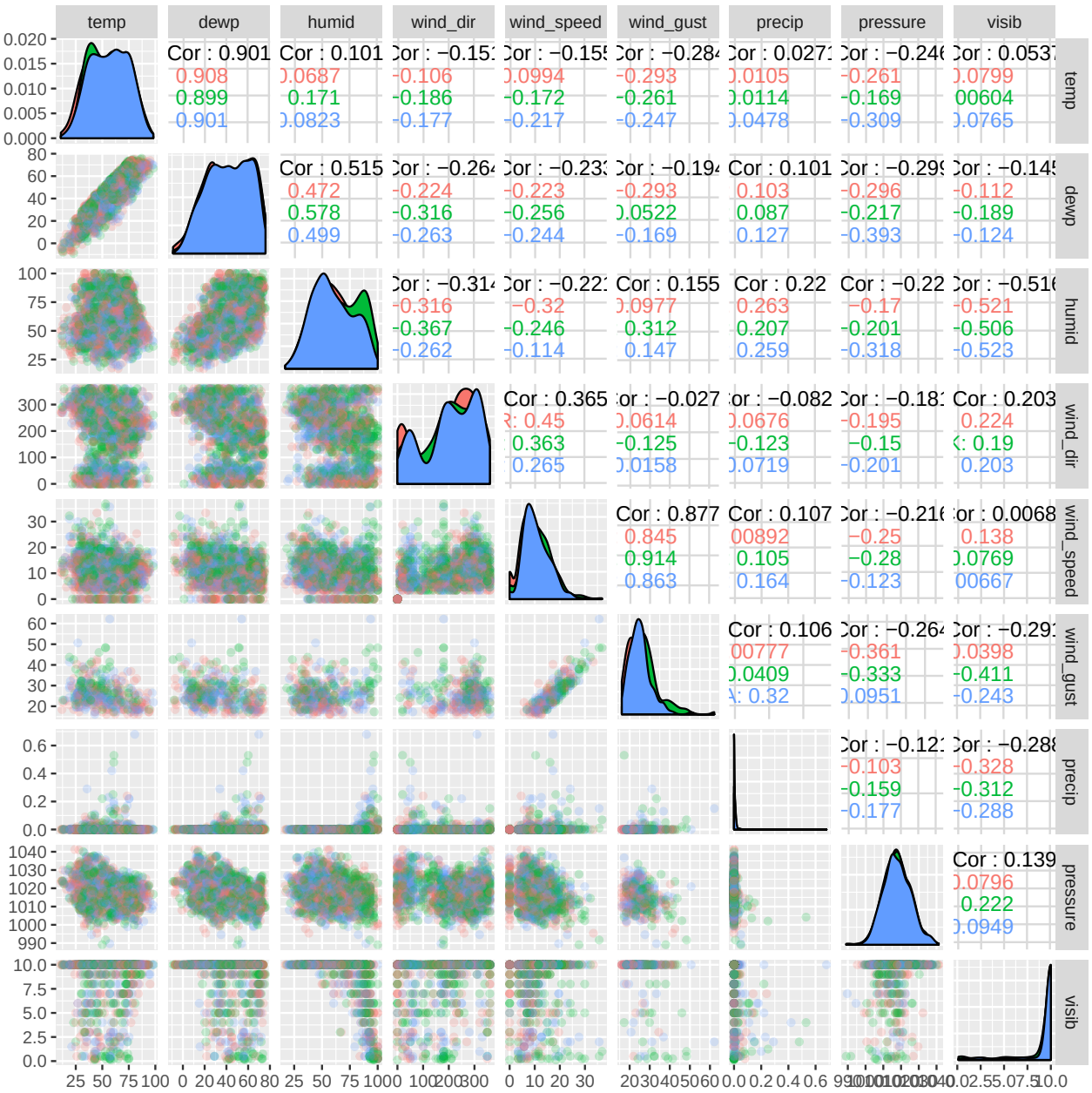


Strong positive correlation exists between `temp` and `dewp` variables, the higher temperature the higher dew point, and the lower the temperature, the lower the dew point. This is a well known physics phenomenon. There are also strong positive correlation exists between `wind_speed` and `wind_gust` variables, since strong wind gusts always bring up the wind speed.

Negative correlation can be noticed between humidity and visibility. As humidity increases, the weather becomes foggy or rainy, which decreases visibility.

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

```
GGally::ggpairs(weather %>% filter(wind_speed < 1000) %>% sample_n(2000),
                columns = 6:14,
                lower = list(continuous=GGally::wrap("points", alpha = 0.2)),
                mapping = ggplot2::aes(colour = origin))
```



JFK airport seems to be more humid than the other two. The wind directions are also different for these three airports. The main patterns of each pair of variables don't seem to vary among different pairs.