EDAV Fall 2019 PSet 2

Chao Huang (ch3474)

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)

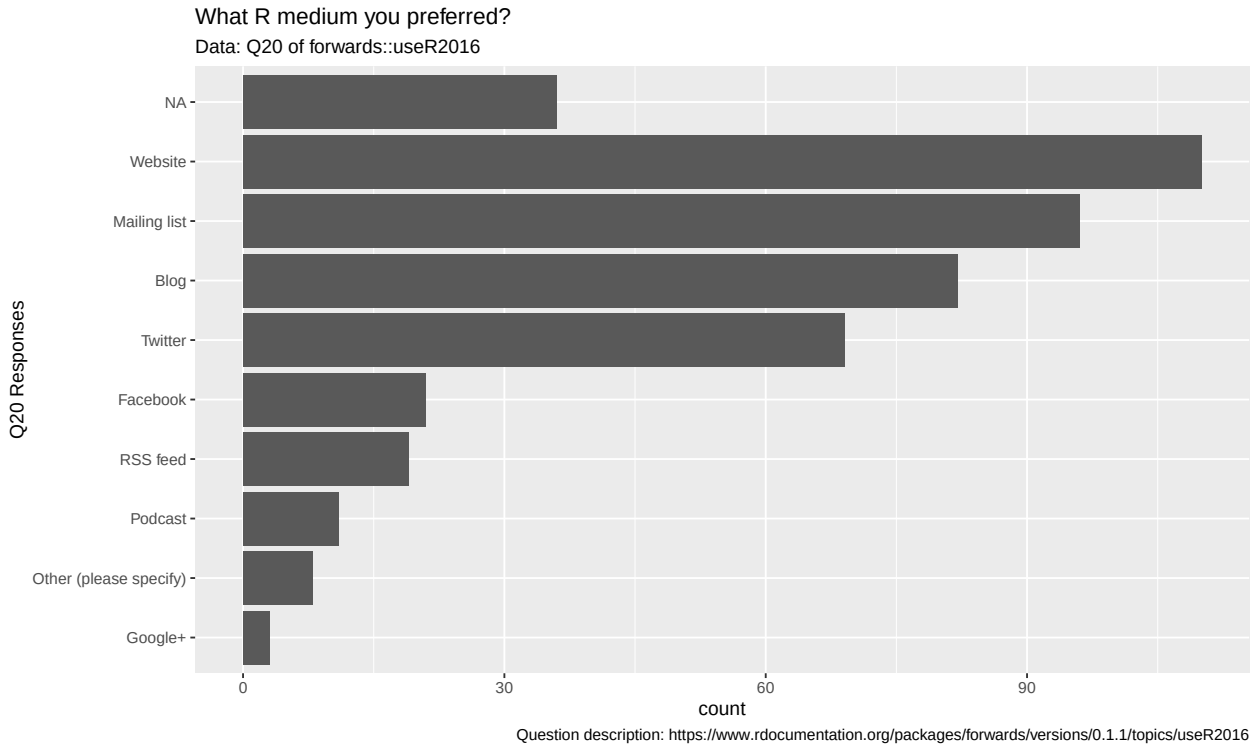
For parts (a) and (b):

- 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.

```
# load dataset
data("useR2016", package = "forwards")
data <- useR2016
```

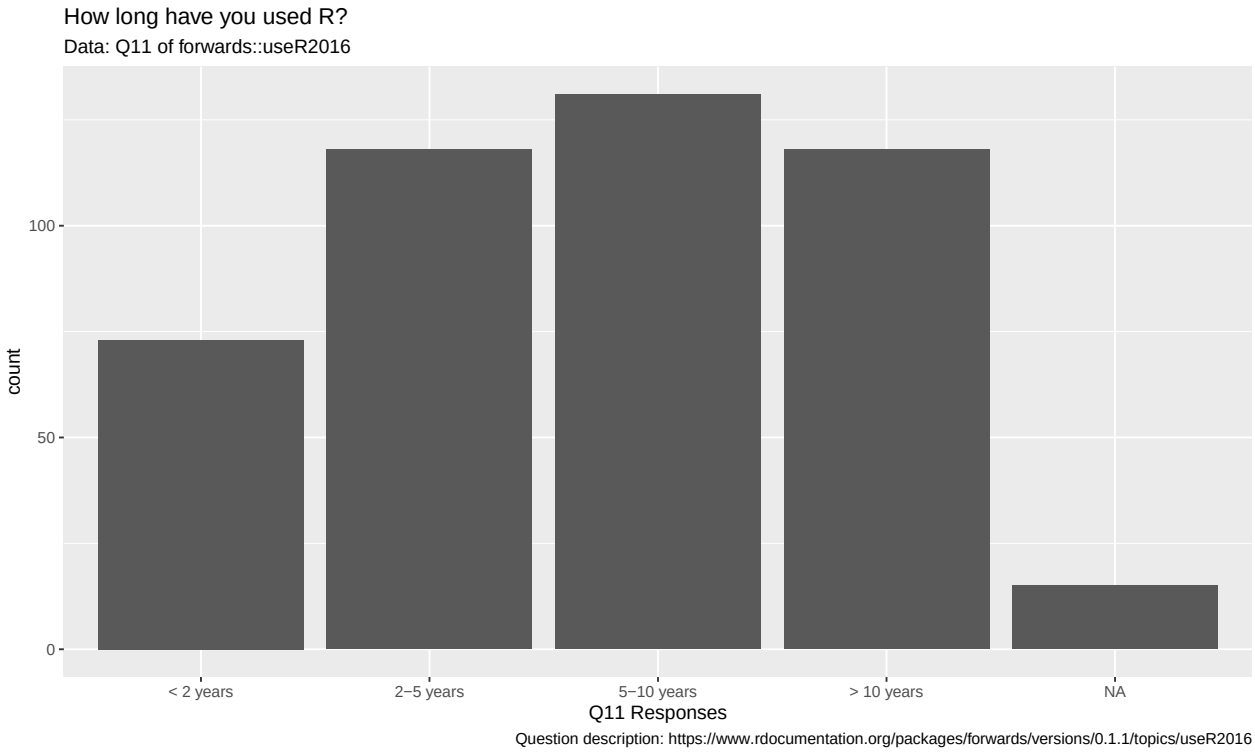
(a) Create a horizontal bar chart of the responses to Q20.

```
ggplot(data, aes(fct_rev(fct_infreq(Q20, ordered = TRUE)))) +
  geom_histogram(stat = "count") +
  coord_flip() +
  xlab("Q20 Responses") +
  labs(
    title = "What R medium you preferred?",
    subtitle = "Data: Q20 of forwards::useR2016",
    caption = "Question description: https://www.rdocumentation.org/packages/forwards/versions/0.1.1/to
```



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

```
ggplot(data, aes(Q11)) +
  geom_histogram(stat = "count") +
  xlab("Q11 Responses") +
  labs(
    title = "How long have you used R?",
    subtitle = "Data: Q11 of forwards::useR2016",
    caption = "Question description: https://www.rdocumentation.org/packages/forwards/versions/0.1.1/to
```

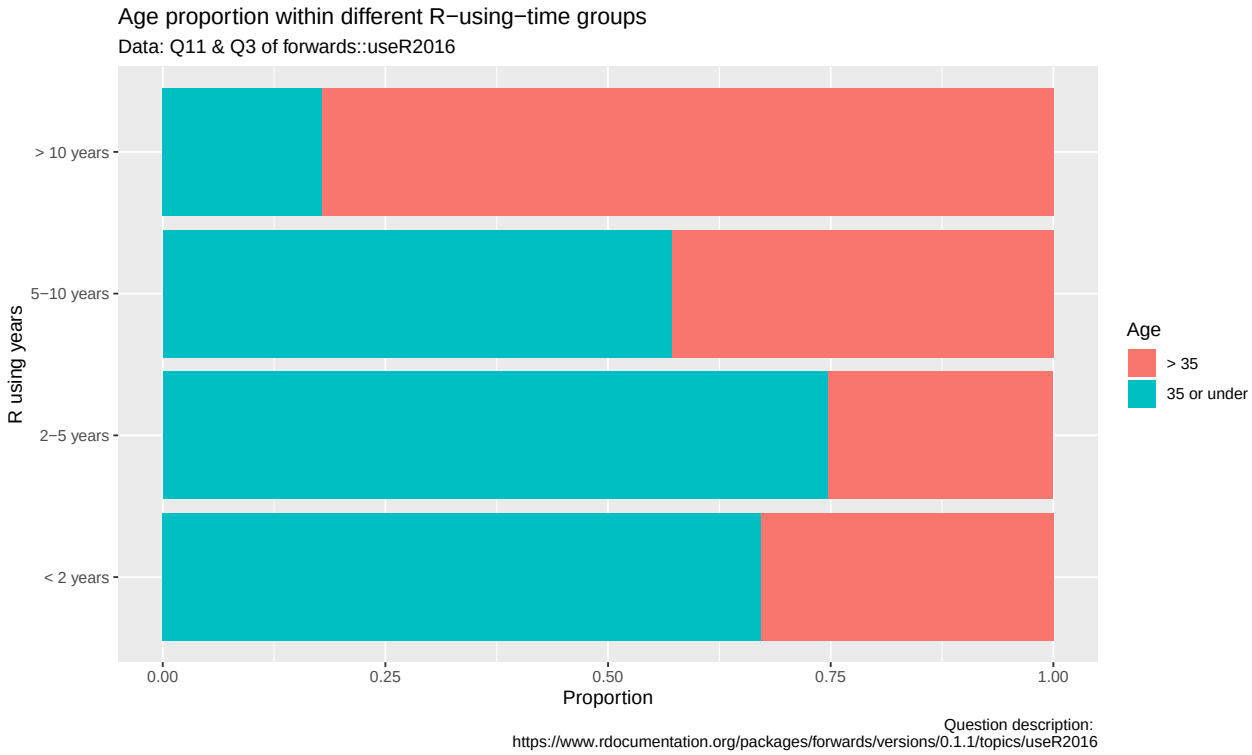


(c) Create a horizontal stacked bar chart showing the proportion of respondents for each level of Q11 who are over 35 vs. 35 or under. Use a descriptive title.

In order to show the correct stack bar, all NA are removed to guarantee that there are only two types of ages.

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

ggplot(stack_Q11, aes(x=Q11)) +
  geom_bar(aes(y=(..count..), fill = 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 = "Age proportion within different R-using-time groups",
      subtitle = "Data: Q11 & Q3 of forwards::useR2016",
      caption = "Question description: \n https://www.rdocumentation.org/packages/forwards/versions/0.1
```

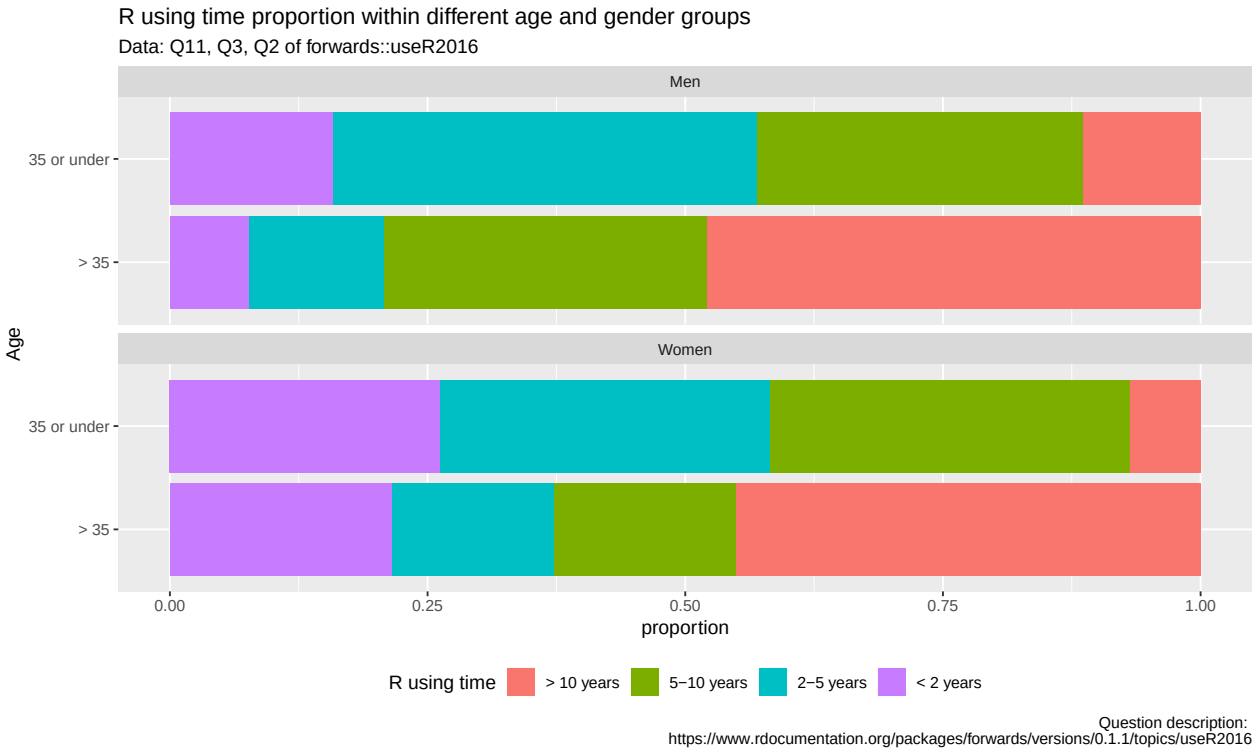


(d) Create a horizontal stacked bar chart showing the proportional breakdown of Q11 for each level of Q3, faceted on Q2. Use a descriptive title.

Still all NA are removed, which means that only answers with explicit age, using time and gender are used in this plot.

```
stack_Q3 <- data %>% 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=fct_rev(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"),
      subtitle = "Data: Q11, Q3, Q2 of forwards::useR2016",
      caption = "Question description: \n https://www.rdocumentation.org/packages/forwards/versions/0.1
  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.

```
library(stringr)
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 <- "We hold these truths to be self-evident, that all men are created equal, that they are ende
add_line_breaks(long_str, 50)
```

```
## [1] "We hold these truths to be self-evident, that all men\nare created equal, that they are endowed
```

(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`.

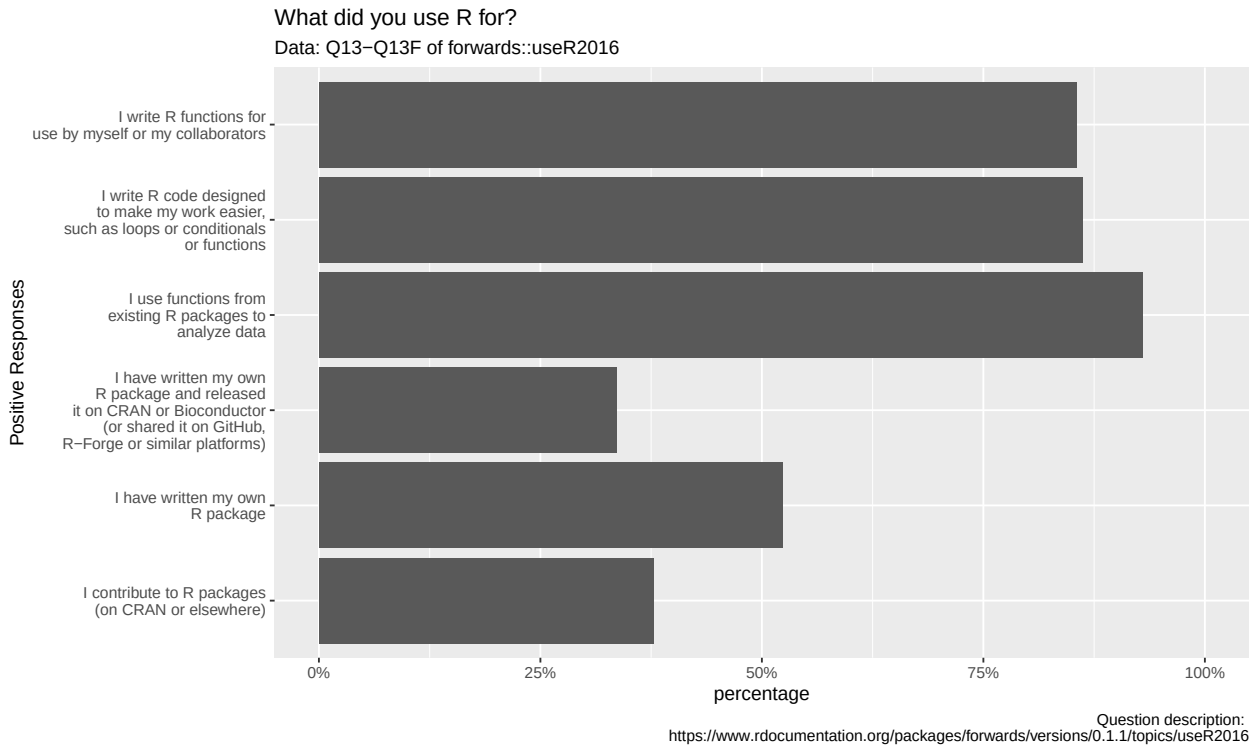
```
library(purrr)
base_df <- data %>% select(matches("Q13"))

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))
```

```
ggplot(pos_count) +
  geom_col(aes(x=add_line_breaks(posResponse, 20), y=percentage)) +
  coord_flip() +
  xlab("Positive Responses") +
  scale_y_continuous(labels = scales::percent, limits = c(0,1)) +
  labs(title = "What did you use R for?",
       subtitle = "Data: Q13-Q13F of forwards::useR2016",
       caption = "Question description: \n https://www.rdocumentation.org/packages/forwards/versions/0.1
```



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.

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

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

```
library(rvest)
```

```
get_all_links_by_weekend <- function (weekend="Last Weekend") {
  webpage <- read_html(base_url)
  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, 3)
```

```
## [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/"
```

(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

Write your data to file so you don't need to scrape the site each time you need to access it.

```
read_content <- function(link) {
  page <- read_html(link)
  scores <- page %>%
    html_nodes(".mop-ratings-wrap__percentage") %>%
    html_text() %>%
    stringr::str_extract("\\d+")
  scores <- c(scores, rep(NaN, 2-length(scores)))# fill NA if scores not found
  name <- page %>%
    html_node("h1") %>%
    html_text() %>%
    stringr::str_trim()

  contents <- c(name, scores)
  names(contents) <- c("name", "tomato_score", "audience_score")
  contents
}
```

```
load_scores <- function(file_name="./ratings.csv") {

  download_content <- function() {
    scores <- map(links, read_content)
    rt_df <- data.frame(
      name=map_chr(scores, function(x) x["name"]),
      tomato_score=unlist(map(scores,
                  function(x) as.numeric(x["tomato_score"])/100)),
      audience_score=unlist(map(scores,
                    function(x) as.numeric(x["audience_score"])/100))
    )
    rt_df
  }

  scores <- if(file.exists(file_name)){
    print("Loading scores from existing file...")
    read.csv(file_name) %>%
      select("name", matches("*score"))
    }
  else {
    print("Loading scores by downloading contents online...")
    download_content()
    }

  if (!file.exists(file_name)) {
    write.csv(scores, file_name)
  }
  scores
```

```
}
```

```
scores <- load_scores()
```

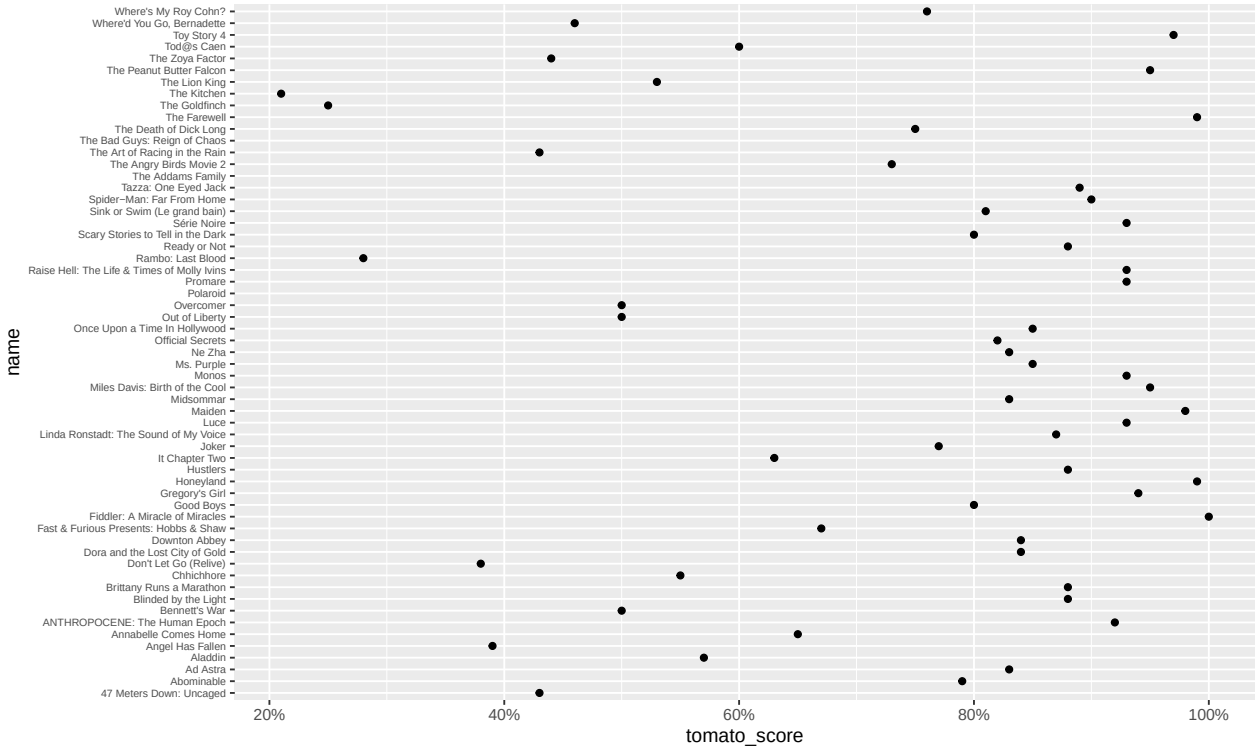
```
## [1] "Loading scores from existing file..."
```

```
head(scores, 6)
```

```
##                name tomato_score audience_score
## 1     Downton Abbey         0.84           0.95
## 2           Ad Astra         0.83           0.43
## 3 Rambo: Last Blood         0.28           0.83
## 4    It Chapter Two         0.63           0.79
## 5           Hustlers         0.88           0.66
## 6     The Lion King         0.53           0.88
```

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

```
ggplot(scores,
       aes(x=name,
           y=tomato_score)) +
  geom_point() +
  scale_y_continuous(labels = scales::percent_format(1)) +
  xlab("name") +
  coord_flip() +
  theme(axis.text.y = element_text(size = 6))
```

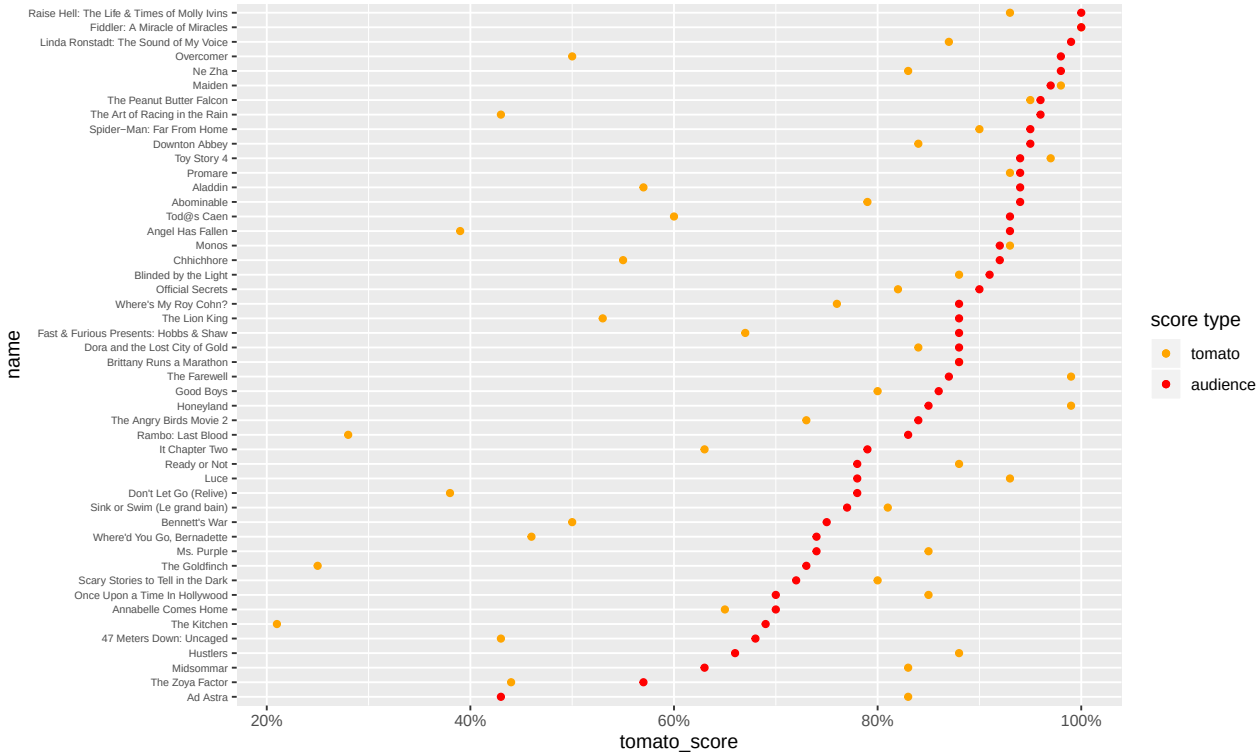


(d) Create a Cleveland dot plot of tomatometer *and* audience scores on the same graph, one color for each. Sort by audience score.

All movies with either no tomatometer or no audience score are removed to prove that each line has exactly two dots.

```
ggplot(scores %>%
         filter(!is.na(tomato_score) & !is.na(audience_score)),
       aes(x=fct_reorder(name, audience_score))) +
  geom_point(aes(y=tomato_score, color="tomato")) +
  geom_point(aes(y=audience_score, color="audience")) +
  scale_y_continuous(labels = scales::percent_format(1)) +
  scale_colour_manual("score type",
                      breaks = c("tomato", "audience"),
                      values = c("red", "orange")) +

  xlab("name") +
  coord_flip() +
  theme(
    axis.text.y = element_text(size = 6)
  )
```



(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.

```
links <- get_all_links_by_weekend("Jul 05 - Jul 07")
scores <- load_scores("./ratings_jul5_jul7.csv")
```

```
## [1] "Loading scores from existing file..."
```

```
interactive_plot <- plotly::plot_ly(
  data = scores,
  y = ~audience_score, x = ~tomato_score,
  text = ~name, hoverinfo = "text")
interactive_plot
```

```
# using online session
api_key="DiEMvM80IDIc6kfHHuiU"
Sys.setenv("plotly_username"="iamironmanx")
Sys.setenv("plotly_api_key"=api_key)
options(browser = 'false')
url <- plotly::api_create(interactive_plot,
                          filename = "scores_scatter_plot")
url$web_url
```

```
## [1] "https://plot.ly/~iamironmanx/1/"
```

3. Weather

[14 points]

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

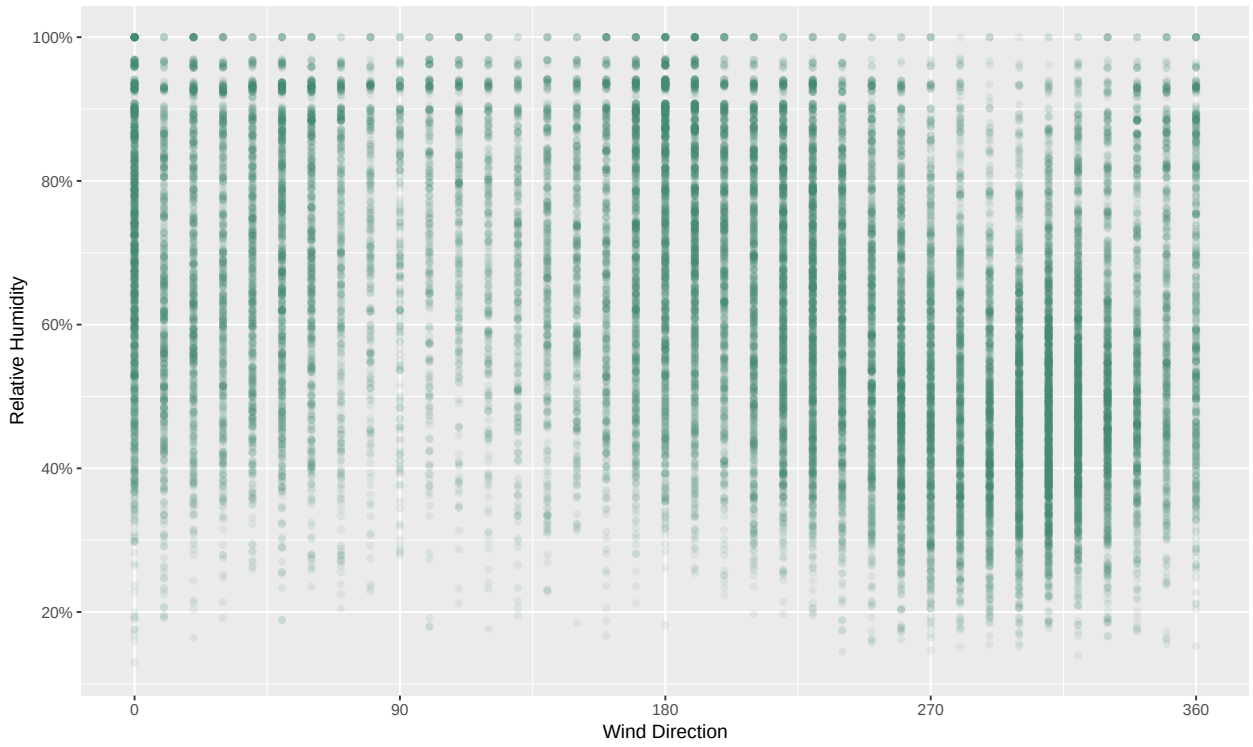
For parts (a) - (d) draw four plots of `wind_dir` vs. `humid` as indicated. For all, adjust parameters to the levels that provide the best views of the data.

```
data("weather", package="nycflights13")
data <- weather
```

(a) Points with alpha blending

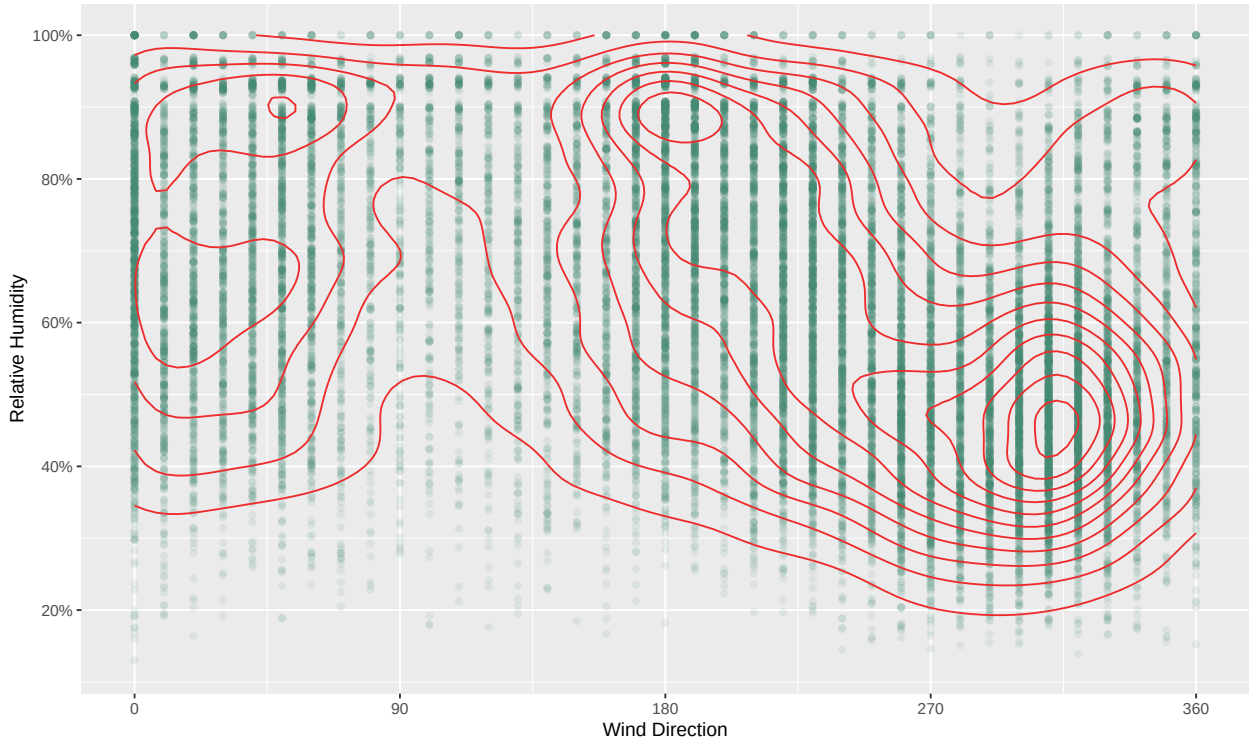
```
base <- ggplot(data, aes(y=humid/100, x=wind_dir))

alpha_blend <- base +
  geom_point(alpha=0.1, color="aquamarine4") +
  scale_y_continuous(labels = scales::percent_format(1),
                     breaks = seq.int(0,1,0.2)) +
  scale_x_continuous(breaks = seq.int(0,360,90)) +
  ylab("Relative Humidity") +
  xlab("Wind Direction")
alpha_blend
```



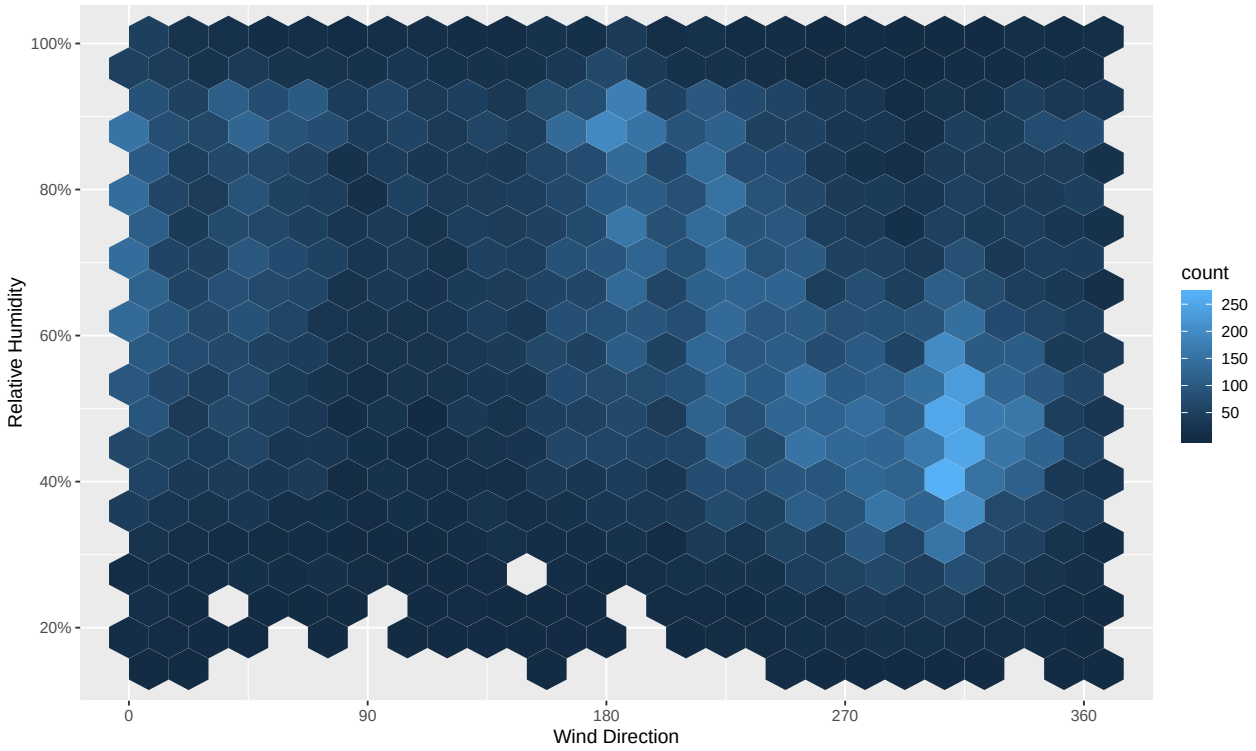
(b) Points with alpha blending + density estimate contour lines

```
alpha_blend + geom_density2d(color="firebrick2")
```



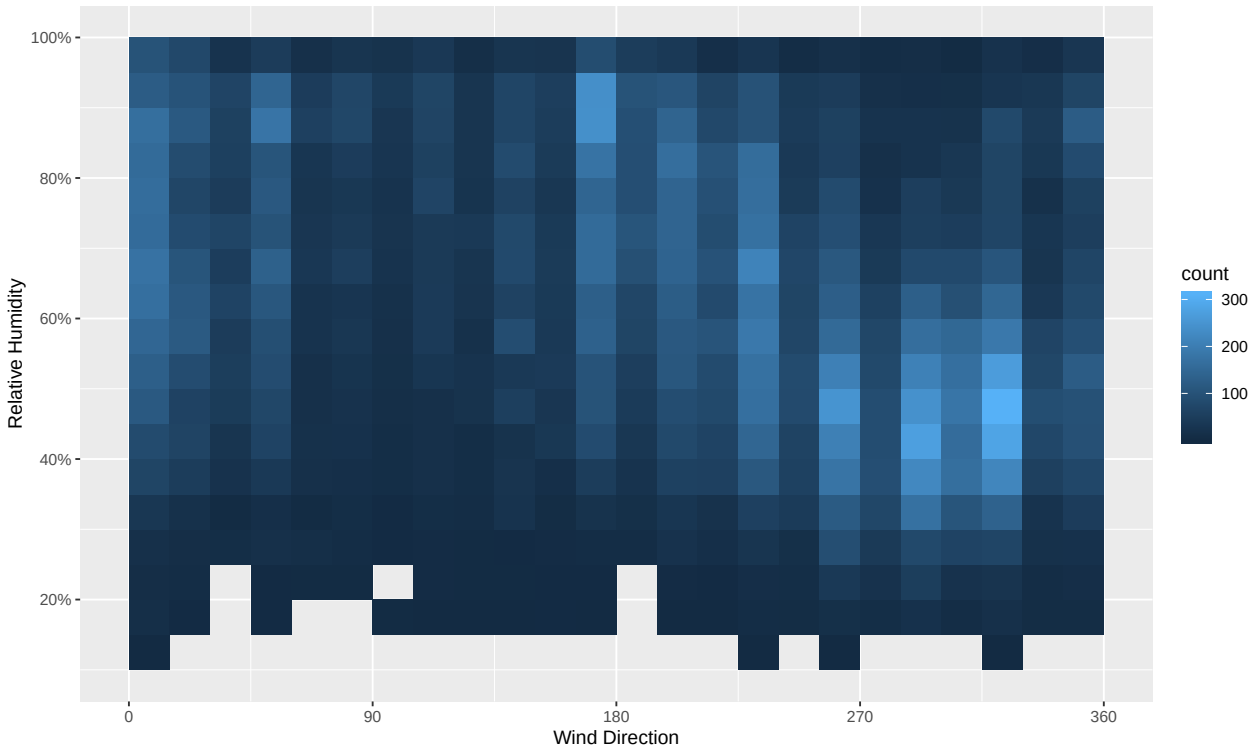
(c) Hexagonal heatmap of bin counts

```
base + geom_hex(binwidth=c(15, 0.05)) +
  scale_y_continuous(labels = scales::percent_format(1),
                     breaks = seq.int(0,1,0.2)) +
  scale_x_continuous(breaks = seq.int(0,360,90)) +
  ylab("Relative Humidity") +
  xlab("Wind Direction")
```



(d) Square heatmap of bin counts

```
base + geom_bin2d(binwidth=c(15, 0.05)) +
  scale_y_continuous(labels = scales::percent_format(1),
                     breaks = seq.int(0,1,0.2)) +
  scale_x_continuous(breaks = seq.int(0,360,90)) +
  ylab("Relative Humidity") +
  xlab("Wind Direction")
```



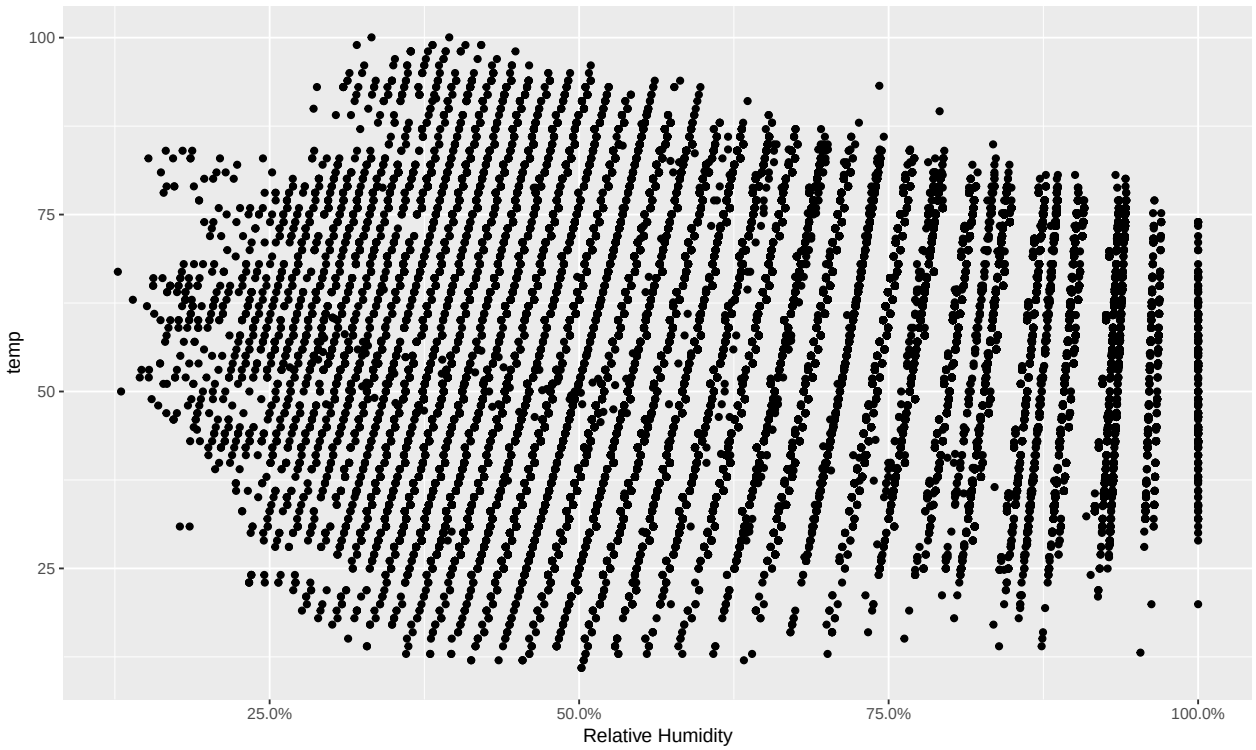
(e) Describe noteworthy features of the data, using the "Movie ratings" example on page 82 (last page of Section 5.3) as a guide.

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?

It's more easily to observe the trend if "temp" is put on the y-axis, so I filpped the coordinates, however, in convensions, when one says "a" vs. "b", "a" is usually the dependent variable and should lie on the y-axis.

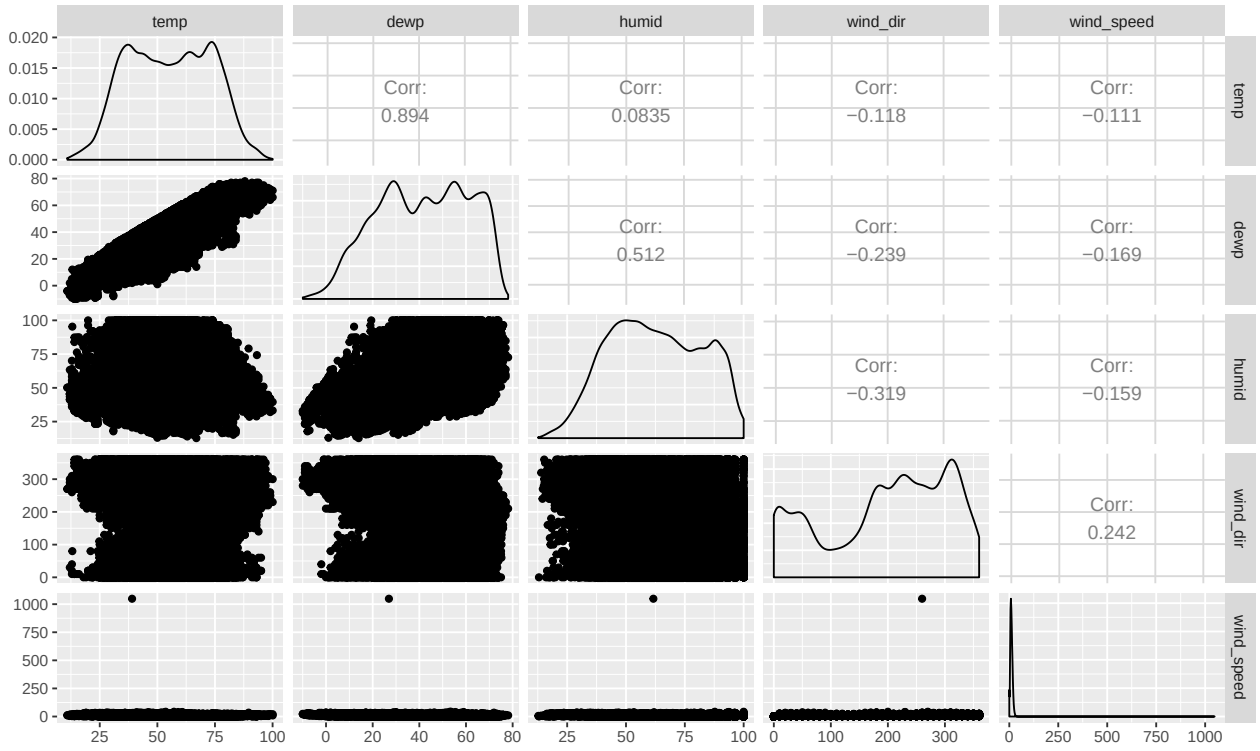
```
ggplot(data, aes(x=temp, y=humid / 100)) +
  geom_point() +
  scale_y_continuous(labels=scales::percent) +
  ylab("Relative Humidity") +
  coord_flip()
```



The reason is that relative humidity is **RELATIVE** to the temperature of the air. It is expressed as the amount of water vapor in the air as a percentage of the total amount that could be held at its current temperature. So it's actually a function of the current temperature and the absolute humidity (or the amount of water in the air) is exactly a parameter of this function. Each diagonal line represent a different parameterization of absolute humidity. Since the sensitivity of measuring absolute humidity can not be unlimited small, there are gaps between diagonal lines due to round-off errors.

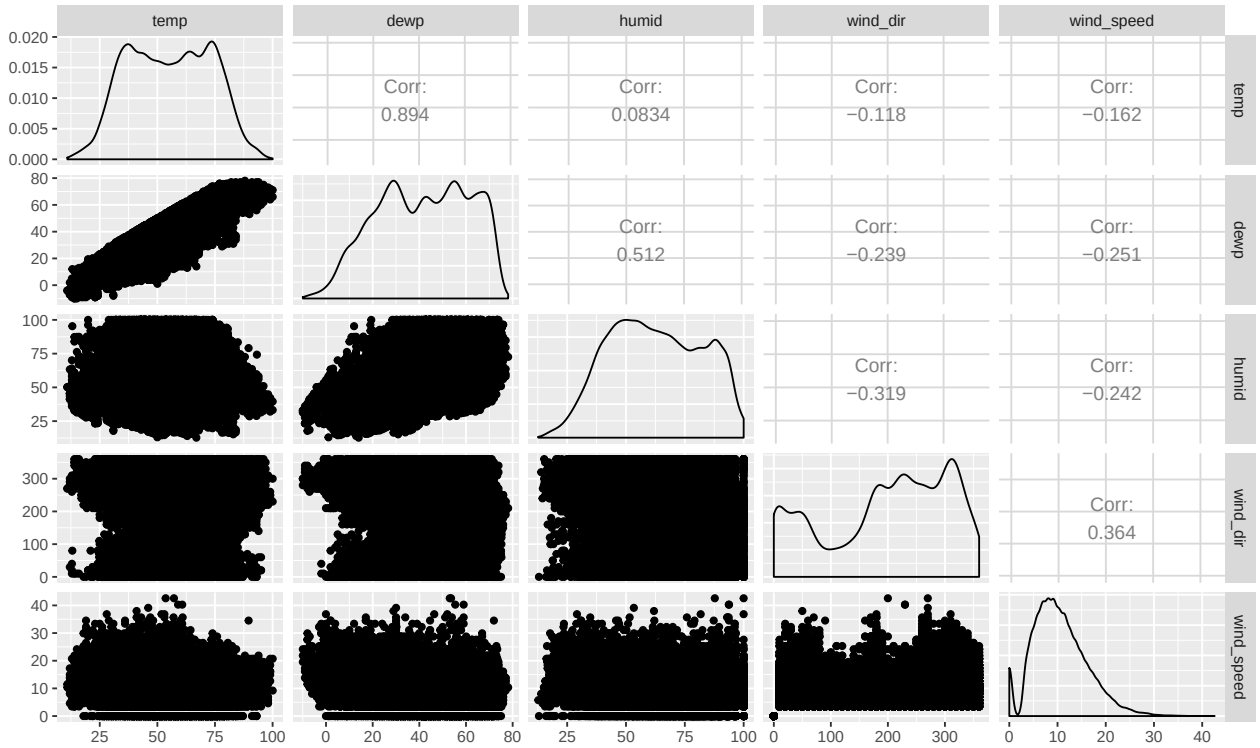
(g) Draw a scatterplot matrix of the continuous variables in the `weather` dataset. Which pairs of variables are strongly positively associated and which are strongly negatively associated?

```
GGally::ggpairs(data, columns=6:10,
                upper = list(continuous = "cor"),
                lower = list(continuous = "points"))
```



There seems to be an outlier in the "wind_speed" with wind speed higher than 1000 m/s. Now we remove it to make the matrix more intuitive and find the correct correlation values.

```
pair_matrix <- GGally::ggpairs(data %>% filter(wind_speed < 1000),
                columns = 6:10)
pair_matrix
```

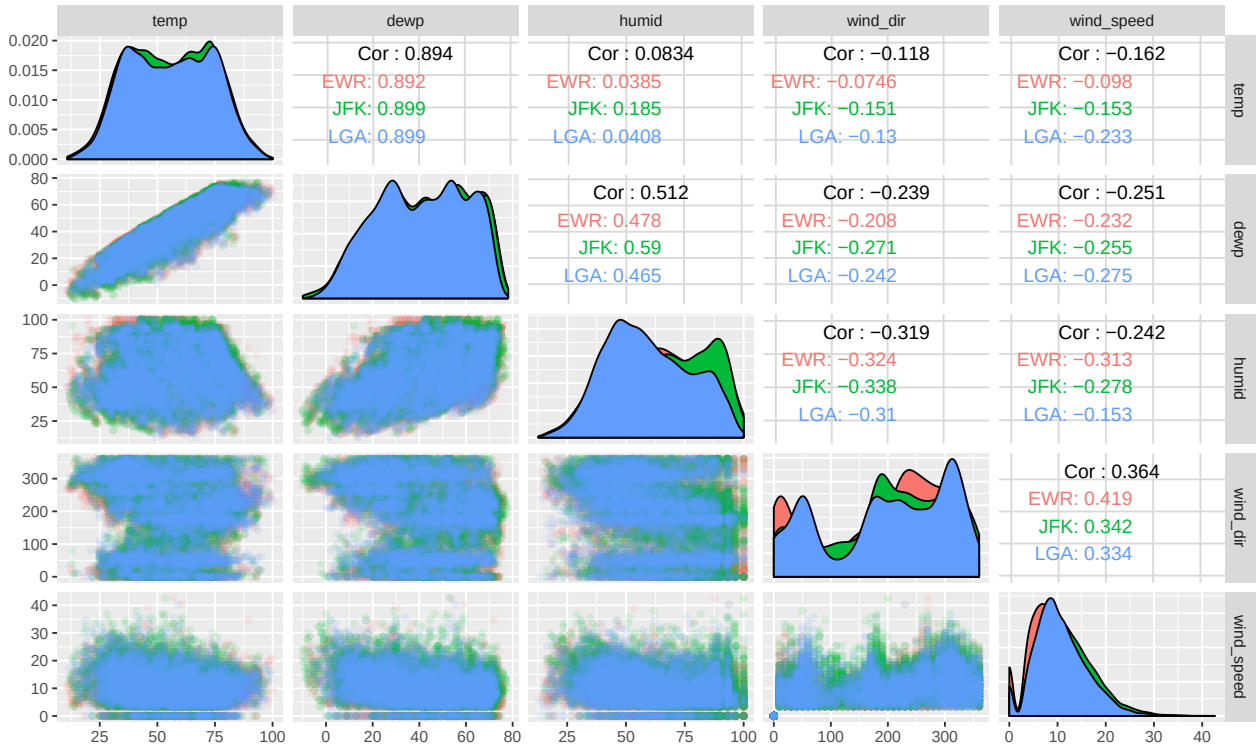


Two pairs are strongly positively associated, namely, “temp vs.”dewp" (corr=0.894) and “dewp vs.”humid" (corr=0.512).

Three pairs are strongly negatively associated, namely, “wind_dir vs.”humid" (corr=-0.319), “wind_speed vs.”dewp" (corr=-0.251) and “wind_dir vs.”dewp" (corr=-0.239).

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

```
GGally::ggpairs(data %>% filter(wind_speed < 1000),
                columns = 6:10,
                lower = list(continuous=GGally::wrap("points", alpha = 0.1)),
                mapping = ggplot2::aes(colour = origin))
```



JFK airport seems to be more humid than the other two and has higher wind speed. The main patterns of each pair of variables don’t seem to vary among different pairs.