assign4.R

harry

2020-10-09

Problem 1(50 pts)

```
#install.packages("nycflights13")
library(nycflights13)
library("tidyverse")
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(maps)
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
library(dplyr)
library(ggplot2)

#data(package = "nycflights13")
flights=as.data.frame(flights)
weather = as.data.frame(weather)
airports = as.data.frame(airports)
```

a.(10/9 pts)

```
flights_a <-(filter(flights, dest == 'TPA'))
flights_a<- filter(flights_a, time_hour >= '2013-11-01 12:00:00' &
                     time_hour <= '2013-11-01 24:00:00')
flights_a <- subset(flights_a,select =
                      c("tailnum","year","month","day","hour","origin"))
weather_a <-subset(weather,select =
                     c("year","month","day","hour","origin","humid"))
answer_a <- left_join(flights_a, weather_a, by =
                        c("year","month","day","hour","origin"), suffix =
                        c("_flights", "_weather"))
head(answer_a)
```

```
##   tailnum year month day hour origin humid
## 1  N567UA 2013    11   1   15    EWR 52.80
## 2  N779JB 2013    11   1   15    EWR 52.80
## 3  N561JB 2013    11   1   16    LGA 50.60
## 4  N974DL 2013    11   1   18    JFK 74.75
## 5  N319NB 2013    11   1   19    LGA 60.51
## 6  N76265 2013    11   1   19    EWR 72.53
```

b.(10/9 pts)

```
#an anti join will return all rows from x where there are not matching values
#in y, keeping only the columns from x,thus, the first will return all rows from
#flights that have no matching values in airports, while the second returns all
#rows from airports that have no matching values in flights.
```

c.(10/9 pts)

```
answer_c <- inner_join(x=flights,y=airports,by = c("origin" = "faa"),
                       all.x=FALSE, all.y=FALSE)
answer_c <- inner_join(x=answer_c,y=airports,by = c("dest" = "faa"),
                       all.x=FALSE, all.y=FALSE)
answer_c <-subset(answer_c,select = c("name.x",
                                      "lat.x","lon.x","name.y","lat.y","lon.y"))
head(answer_c)
```

```
##                name.x    lat.x     lon.x                          name.y
## 1 Newark Liberty Intl 40.69250 -74.16867    George Bush Intercontinental
## 2          La Guardia 40.77725 -73.87261    George Bush Intercontinental
## 3 John F Kennedy Intl 40.63975 -73.77893                     Miami Intl
## 4          La Guardia 40.77725 -73.87261 Hartsfield Jackson Atlanta Intl
## 5 Newark Liberty Intl 40.69250 -74.16867              Chicago Ohare Intl
## 6 Newark Liberty Intl 40.69250 -74.16867  Fort Lauderdale Hollywood Intl
##      lat.y     lon.y
## 1 29.98443 -95.34144
## 2 29.98443 -95.34144
## 3 25.79325 -80.29056
## 4 33.63672 -84.42807
## 5 41.97860 -87.90484
## 6 26.07258 -80.15275
```

d.(10/9 pts)

```
testflights <- na.omit(flights)
answer_d <- testflights %>% group_by(origin,dest) %>%
  summarise(count = n_distinct(c(origin,dest)))
```

```
## `summarise()` regrouping output by 'origin' (override with `.groups` argument)
```

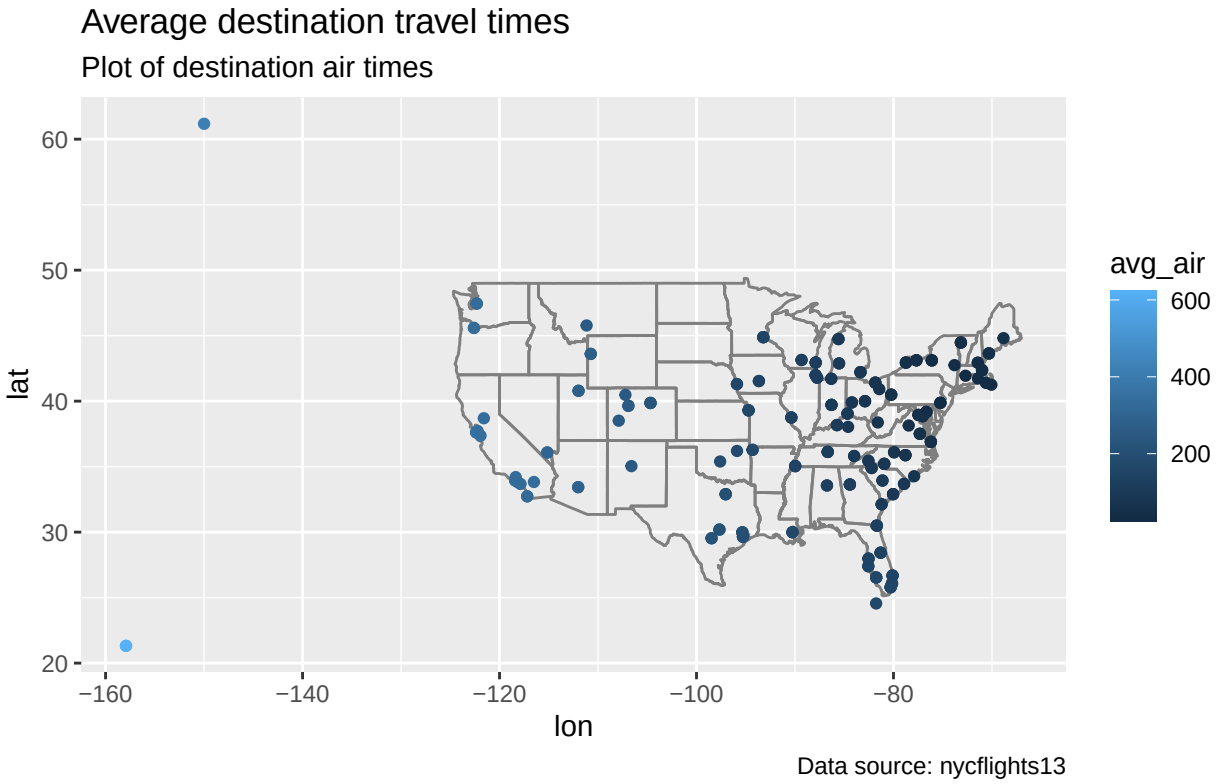
```
nrow(answer_d)
```

```
## [1] 223
```

```
#I'm unsure of how we ended up with the extra unique combos but we did,
```

e.(10/9 pts)

```
answer_e <- testflights %>%
  group_by(origin,dest) %>%
  summarise_at(vars(air_time),
               list(avg_air = mean))

airports_e <- airports%>% right_join(answer_e,c("faa"="dest"))
airports_e <- na.omit(airports_e)

p <-ggplot(airports_e,aes(lon,lat))+
  borders("state")+
  geom_point(aes(colour = avg_air))+
  coord_quickmap()+
  labs(title = "Average destination travel times",
       subtitle = "Plot of destination air times",
       caption = "Data source: nycflights13")
print(p)
```



Problem 2 (30 pts)

```
#?register_google
library(ggmap)
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

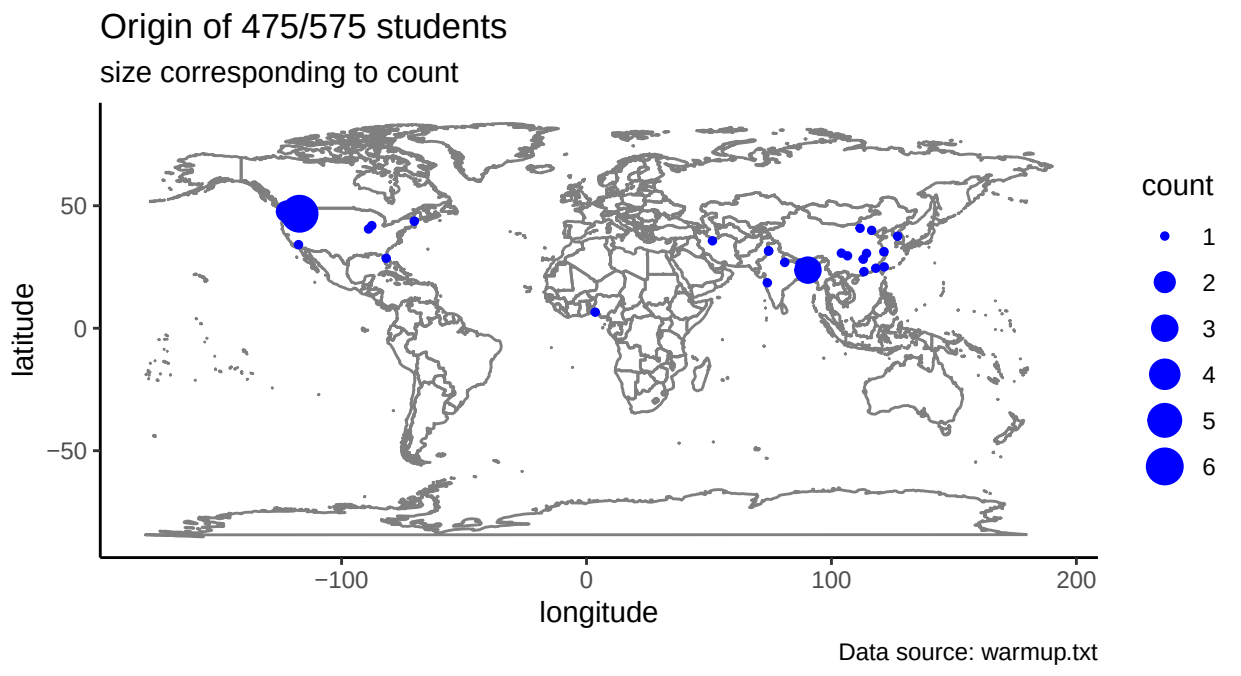
```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
warmupdat <- read.csv("warmupcsv.csv",header = FALSE,quote = "\"" )
addrs <- warmupdat[1]
#geocodes <- geocode(as.character(warmupdat$addr))
#warmupdat <- data.frame(warmupdat[,1:2],geocodes)


#this would've been how I got the lat and long for my data, however, geocoding
#would've required signing up to either Google or Bings api which required
#credit card info and convoluted trials/billing plans.

#instead I used geocode.localfocus.nl and pasted the unique contents of my
#csv instead.
#and will load the resulting csv.

warmupdat_complete <- distinct(read.csv("warmuplatlon.csv",
                                        header = FALSE,quote = "\""))
```

```
warmupdat<- left_join(addrs, warmupdat_complete, by = c("V1" = "V1"))

warm_group <- warmupdat %>%
  group_by(V1) %>%
  summarize(count = n())
```

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

```
warmupdat_complete <- left_join(warmupdat_complete,
                                warm_group, by = c("V1" = "V1"))
warmupdat_complete$count=as.numeric(warmupdat_complete$count)
colnames(warmupdat_complete) <- c("location", "latitude", "longitude","count")
q <-ggplot(warmupdat_complete,aes(longitude,latitude))+
  borders("world")+
  theme_classic() +
  geom_point(aes(size = count),color = "blue")+
  coord_quickmap()+
  labs(title = "Origin of 475/575 students",
       subtitle = "size corresponding to count",
       caption = "Data source: warmup.txt")
print(q)
```



Problem 3 (20 pts)

```r
#for this problem I followed an excellent tutorial online, I chose to use
#the entire bee movie script as my example text.
library("tm")
```

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
library("wordcloud")
```

```
## Loading required package: RColorBrewer
```

```r
library("RColorBrewer")

text <- readLines("beemovie.txt")
```

```
## Warning in readLines("beemovie.txt"): incomplete final line found on
## 'beemovie.txt'
```

```r
# Load the data
docs <- Corpus(VectorSource(text))

# Convert the text to lower case, removing whitespace, stopwords, punctuation and
#setting it all to lowercase
docs <- tm_map(docs, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(docs, content_transformer(tolower)):
## transformation drops documents
```

```r
docs <- tm_map(docs, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(docs, removeNumbers): transformation drops
## documents
```

```r
docs <- tm_map(docs, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, stopwords("english")):
## transformation drops documents
```

```r
docs <- tm_map(docs, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(docs, removePunctuation): transformation drops
## documents
```

```
docs <- tm_map(docs, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(docs, stripWhitespace): transformation drops
## documents
```

```
dtm <- TermDocumentMatrix(docs)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)

#"WordCloud of the entire Bee Movie script"
wordcloud(words = d$word, freq = d$freq, min.freq = 8,
          max.words=250, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(12, "Paired"))
```