Practice on dplyr with bridges data

import the data

```
library(ggplot2)
library(choroplethr)

## Loading required package: acs

## Loading required package: stringr

## Loading required package: plyr

## Loading required package: XML

##
## Attaching package: 'acs'

## The following object is masked from 'package:base':
##
##     apply

library(dplyr)

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:acs':
##
##     combine

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(readr)
library(data.table)

## --------------------------------------------------------------------
-----
```

```
## data.table + dplyr code now lives in dtplyr.
## Please library(dtplyr)!

## -------------------------------------------------------------------
-----

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

dest = "https://www.fhwa.dot.gov/bridge/nbi/2016/delimited/AK16.txt"
tmp = fread(dest)
classes = sapply(tmp, class)
states= read_csv("http://pages.stat.wisc.edu/~karlrohe/classes/data/sta
teAbv.txt")

## Parsed with column specification:
## cols(
##   Alberta = col_character(),
##   AB = col_character()
## )

states=states[-(1:12),]
states[51,] = c("WashDC", "DC")
states[52,] = c("Puerto Rico", "PR")
dest= rep("", 52)
for(i in 1:52) dest[i]=paste("https://www.fhwa.dot.gov/bridge/nbi/2016/
delimited/", states[i,2],"16.txt", sep = "")
x16 = ldply(dest, fread, colClasses = classes)

## Warning in FUN(X[[i]], ...): Some columns have been read as type
## 'integer64' but package bit64 isn't loaded. Those columns will displ
ay as
## strange looking floating point data. There is no need to reload the
data.
## Just require(bit64) to obtain the integer64 print method and print t
he data
## again.

## Warning in FUN(X[[i]], ...): Some columns have been read as type
## 'integer64' but package bit64 isn't loaded. Those columns will displ
ay as
## strange looking floating point data. There is no need to reload the
data.
## Just require(bit64) to obtain the integer64 print method and print t
he data
## again.
## Warning in FUN(X[[i]], ...): Some columns have been read as type
## 'integer64' but package bit64 isn't loaded. Those columns will displ
ay as
## strange looking floating point data. There is no need to reload the
data.
## Just require(bit64) to obtain the integer64 print method and print t
he data
## again.

## Warning in FUN(X[[i]], ...): Bumped column 97 to type character on d
ata
## row 1357, field contains 'CAN'. Coercing previously read values in t
his
## column from logical, integer or numeric back to character which may
not
## be lossless; e.g., if '00' and '000' occurred before they will now b
e just
## '0', and there may be inconsistencies with treatment of ',,' and ',N
A,' too
## (if they occurred in this column before the bump). If this matters p
lease
## rerun and set 'colClasses' to 'character' for this column. Please no
te that
## column type detection uses a sample of 1,000 rows (100 rows at 10 po
ints)
## so hopefully this message should be very rare. If reporting to datat
able-
## help, please rerun and include the output from verbose=TRUE.

## Warning in FUN(X[[i]], ...): Bumped column 97 to type character on d
ata
## row 973, field contains '38-'. Coercing previously read values in th
is
## column from logical, integer or numeric back to character which may
not
## be lossless; e.g., if '00' and '000' occurred before they will now b
e just
## '0', and there may be inconsistencies with treatment of ',,' and ',N
A,' too
## (if they occurred in this column before the bump). If this matters p
lease
## rerun and set 'colClasses' to 'character' for this column. Please no
te that
## column type detection uses a sample of 1,000 rows (100 rows at 10 po
ints)
## so hopefully this message should be very rare. If reporting to datat
able-
## help, please rerun and include the output from verbose=TRUE.
## Warning in FUN(X[[i]], ...): Bumped column 97 to type character on d
ata
## row 16239, field contains 'CAN'. Coercing previously read values in
this
## column from logical, integer or numeric back to character which may
not
## be lossless; e.g., if '00' and '000' occurred before they will now b
e just
## '0', and there may be inconsistencies with treatment of ',,' and ',N
A,' too
## (if they occurred in this column before the bump). If this matters p
lease
## rerun and set 'colClasses' to 'character' for this column. Please no
te that
## column type detection uses a sample of 1,000 rows (100 rows at 10 po
ints)
## so hopefully this message should be very rare. If reporting to datat
able-
## help, please rerun and include the output from verbose=TRUE.

## Warning in FUN(X[[i]], ...): Bumped column 97 to type character on d
ata
## row 2635, field contains 'CAN'. Coercing previously read values in t
his
## column from logical, integer or numeric back to character which may
not
## be lossless; e.g., if '00' and '000' occurred before they will now b
e just
## '0', and there may be inconsistencies with treatment of ',,' and ',N
A,' too
## (if they occurred in this column before the bump). If this matters p
lease
## rerun and set 'colClasses' to 'character' for this column. Please no
te that
## column type detection uses a sample of 1,000 rows (100 rows at 10 po
ints)
## so hopefully this message should be very rare. If reporting to datat
able-
## help, please rerun and include the output from verbose=TRUE.

x16=tbl_df(x16)
#here are the variables of potential interests
keep = c("STATE_CODE_001", "STRUCTURE_NUMBER_008", "COUNTY_CODE_003", "
LAT_016", "LONG_017", "TOLL_020", "ADT_029", "YEAR_ADT_030", "YEAR_BUIL
T_027", "TOTAL_IMP_COST_096")
x = select(x16, one_of(keep))
```

In this part I just copy the code post on Github.

data wrangling

In the lecture, I was deeply impressed by the "map" of Wisconsin only using the data of its bridges. I think it is a very cool idea to make a map of United States justing using the latitude and longitude of bridges in the country. Because it can reflect how economy develops in one states. So I dig into the variables of latitude and longitude to make an accurate map.

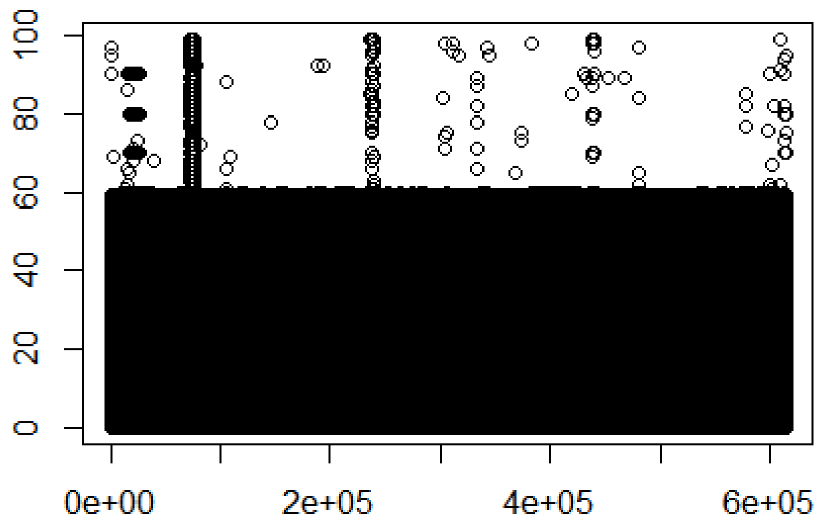
The first thing is that we should notice LAT_016 has 8 digits and LONG_017 has 9 digits for the longitude of US is mainly between W70 to W130. This is very important and we should care about it.

The second thing to do is analyzing the structure of two variable, LAT_016 and LONG_017. In Karl's R code, he set all the 3 to 8 digits divided by 600000. But I think the first two digits inply degree while the 3 to 4 digits inply minutes and the 6 to 8 digits inply seconds. Because it is more accurate. So I do some summary to figure it out.

```
summary(as.numeric(substr(x$LAT_016,5,6))>=60)

##    Mode   FALSE    TRUE    NA's
## logical  610054    2382    1951

plot(as.numeric(substr(x$LAT_016,5,6)),xlab="",ylab="")
```



From the plot and summary we can see that 99% of the 5 to 6 digits of data are under 60. The porobality of this event is nearly 0 if all the 3 to 8 digits of data are minutes. Because if it were ture, 5 to 6 digits of data are just the decimals part of minutes and they should have almostly uniform distribution from 0 to 99. Then it is impossiable to have 99% numbers under 60 with such a large amount.

After figuring out these two points we can transform the data into a more accurate form.

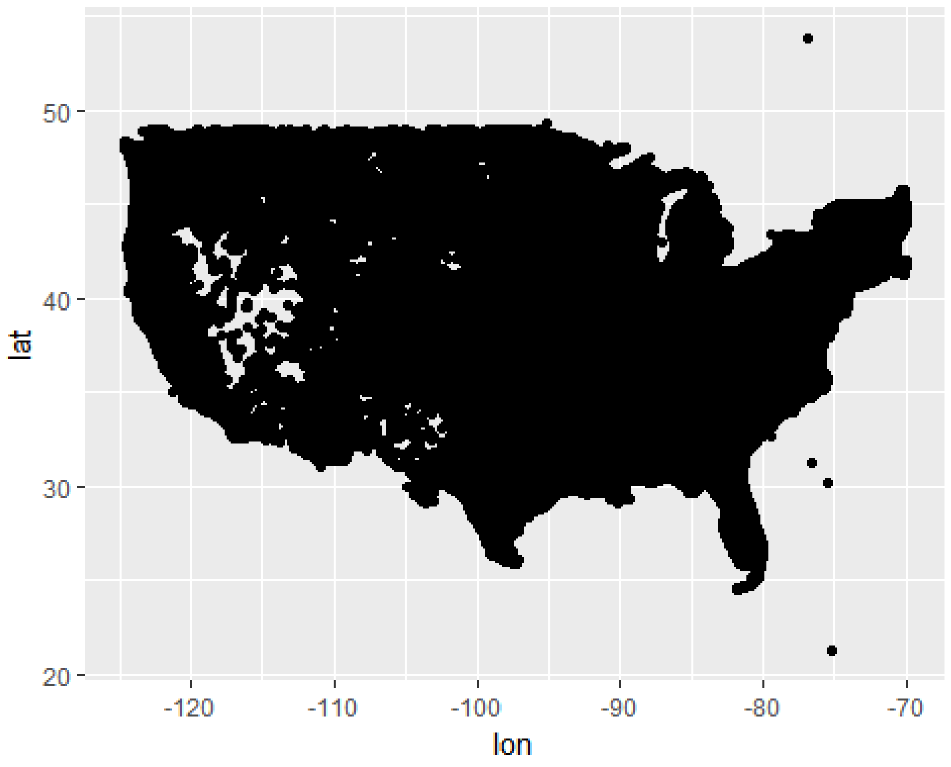
```r
#I do the transformation separately because whenever I combine longitud
e transformation of 8 digits and 9 digits together, R get a bug. I do n
ot know how to solve it. So I do it separately.
min2dec.lat = function(x){
  as.numeric(substr(x,1,2)) + as.numeric(substr(x,3,4))/60 + as.numeric
(substr(x,5,8))/360000 %>% return
}
min2dec.lon8 = function(x){
    as.numeric(substr(x,1,2)) + as.numeric(substr(x,3,4))/60 + as.numer
ic(substr(x,5,8))/360000 %>% return
}
min2dec.lon9 = function(x){
    as.numeric(substr(x,1,3)) + as.numeric(substr(x,4,5))/60 + as.numer
ic(substr(x,6,9))/360000 %>% return
}
#get rid of NA
x = filter(x,is.na(x$LAT_016)==0&is.na(x$LONG_017)==0)
#do latitude transformation and do a filter. Because USA is mainly betw
een 20N and 60N except Hawaii and Alaska.
x1 = mutate(x,lat = min2dec.lat(LAT_016))
x1 = filter(x1,lat<60&lat>20)
#do longitude transformation. I think these codes below are quite stupi
d.
n=nchar(x1$LONG_017)
x2=x1
x3=x1
x2 = mutate(x2[n==8,],lon = min2dec.lon8(LONG_017))

## Warning in min2dec.lon8(c(87341340, 87340890, 84583800, 87581200,
## 87225400, : 强制改变过程中产生了 NA

x3 = mutate(x3[n==9,],lon = min2dec.lon9(LONG_017))
x1$lon=rep(0,dim(x1)[1])
x1[n==8,"lon"]=x2$lon
x1[n==9,"lon"]=x3$lon
#do filter according to geographic facts about USA.
x1 = filter(x1,lon<130&lon>70)
#set the longitude negative sign so the final plot will have the same d
irection as the real map.
x1$lon=-x1$lon
```

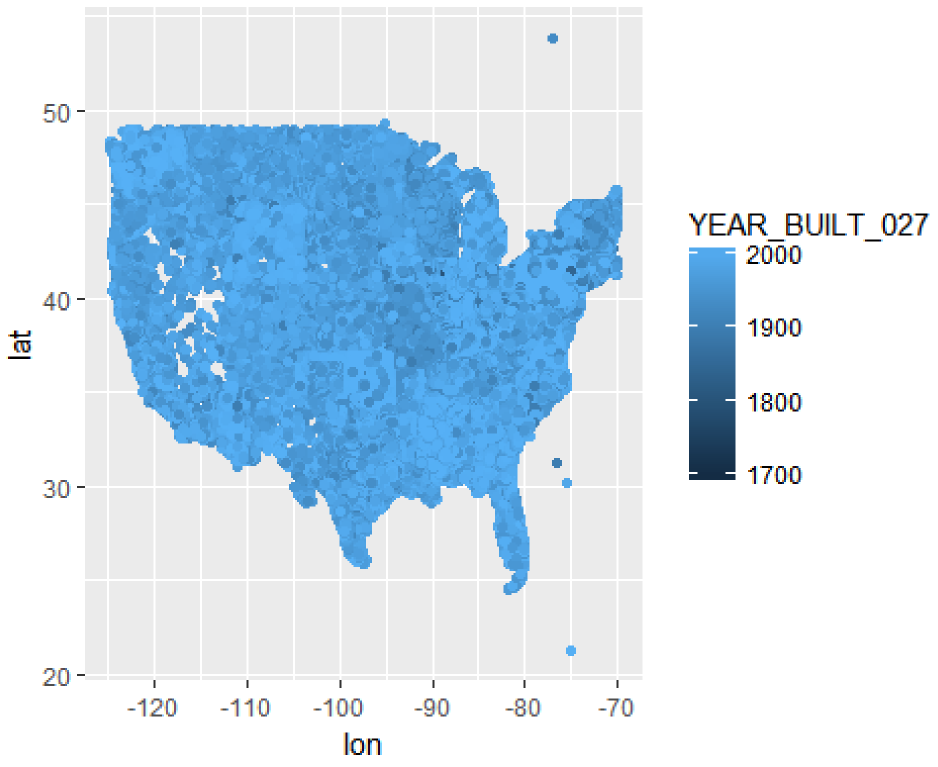
Plot the data and get interesting facts.

```r
ggplot(data = x1) +geom_point(mapping = aes(y = lat, x = lon))
```

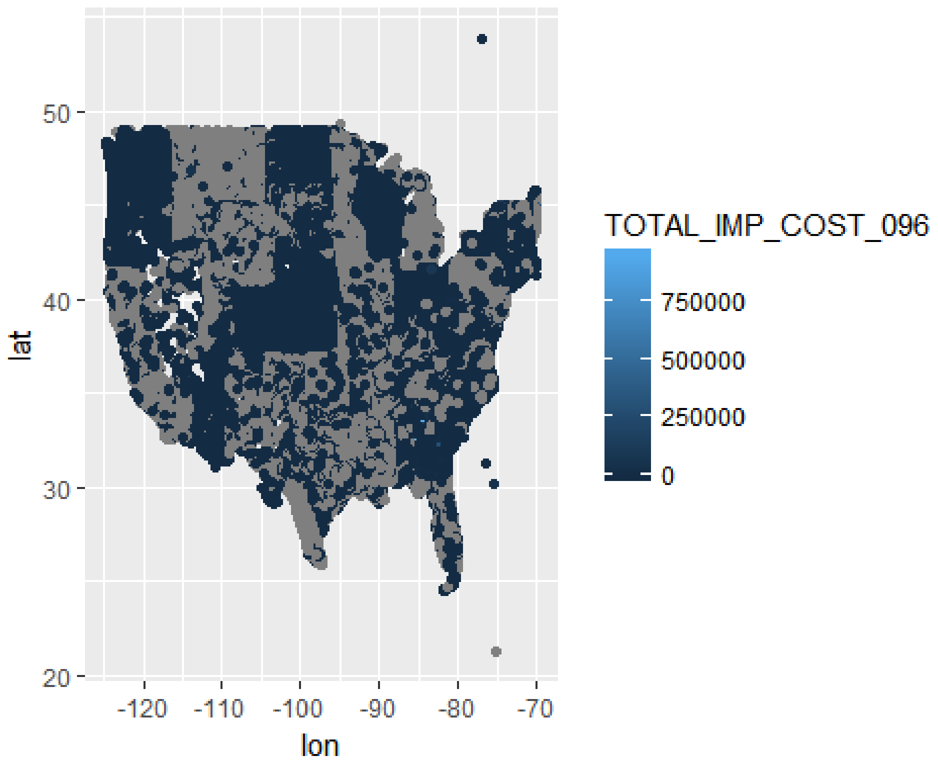


We can see that this plot is almost identical to the real map. From the plot we can see that the blanks, which imply lack of bridges and potential economic downturn, are mainly around the mid-west states, such like Nevada, Utah, Arizona and New Mexcico. And there are also two blanks around Texas and Iowa.

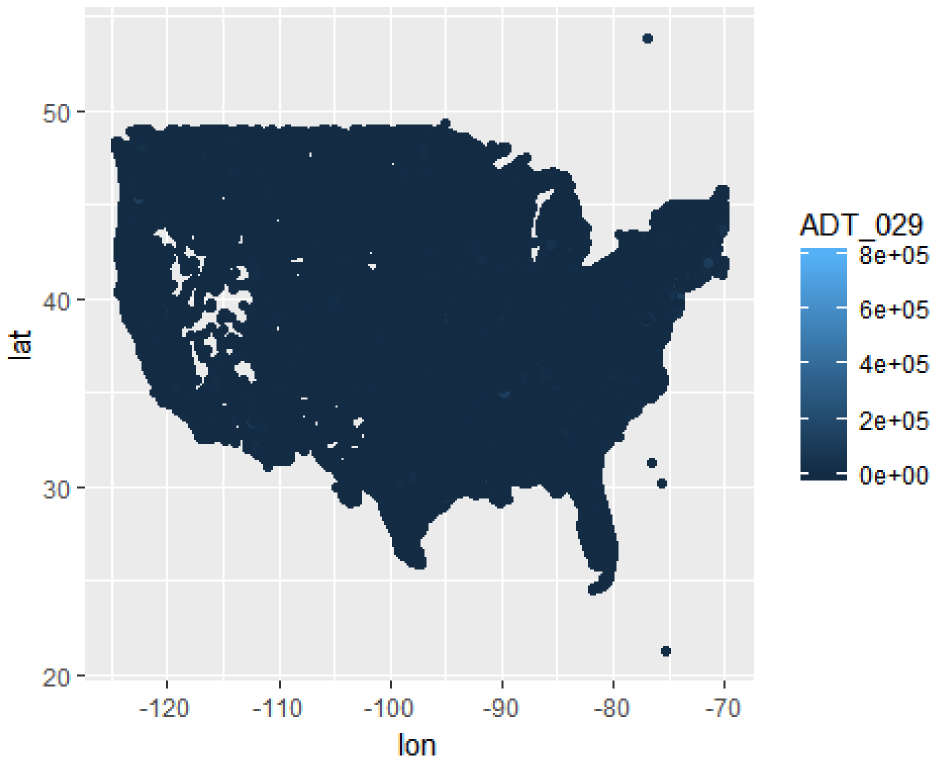
```
ggplot(data = x1) +geom_point(mapping = aes(y = lat, x = lon,col =YEAR_
BUILT_027))
```



```
ggplot(data = x1) +geom_point(mapping = aes(y = lat, x = lon,col =TOTAL
_IMP_COST_096))
```



```r
ggplot(data = x1) +geom_point(mapping = aes(y = lat, x = lon,col =ADT_0
29))
```



We can do further analysis on the relationship between the accident rate and the year built or total improvement project cost.