Data607 - Tidyverse Create Assignment

Amit Kapoor

3/24/2020

Assignment

Using one or more TidyVerse packages and any dataset from fivethirtyeight.com or Kaggle, the task is to create a programming sample "vignette" that demonstrates use of one or more of the capabilities of selected TidyVerse package.

Getting started

Lets load tidyverse package first. It includes readr, dplyr, tidyr, ggplot2, stringr, tibble, forcats and purr packages.

- **tidyverse**

We're going to load a dataset from fivethirtyeight.com to help us show tidyverse package at work. This data shows America's bad drivers in all the states, involved in collisions.

First step is to read the bad-drivers data from github repository. The data contains below fields:

- State
- Number of drivers involved in fatal collisions per billion miles
- Percentage Of Drivers Involved In Fatal Collisions Who Were Speeding
- Percentage Of Drivers Involved In Fatal Collisions Who Were Alcohol-Impaired
- Percentage Of Drivers Involved In Fatal Collisions Who Were Not Distracted
- Percentage Of Drivers Involved In Fatal Collisions Who Had Not Been Involved In Any Previous Accidents
- Car Insurance Premiums ($)
- Losses incurred by insurance companies for collisions per insured driver ($)

Data read using readr package

- **read_csv()** function is from **readr** package, used for reading flat file data with comma separated values.

```
# define URL for bad drivers data
theURL <- 'https://raw.githubusercontent.com/fivethirtyeight/data/master/bad-drivers/bad-drivers.csv'

# read data
bad_drivers <- read_csv(theURL)
```

```
## Parsed with column specification:
## cols(
##   State = col_character(),
##   `Number of drivers involved in fatal collisions per billion miles` = col_double(),
##   `Percentage Of Drivers Involved In Fatal Collisions Who Were Speeding` = col_double(),
##   `Percentage Of Drivers Involved In Fatal Collisions Who Were Alcohol-Impaired` = col_double(),
##   `Percentage Of Drivers Involved In Fatal Collisions Who Were Not Distracted` = col_double(),
##   `Percentage Of Drivers Involved In Fatal Collisions Who Had Not Been Involved In Any Previous Acci
##   `Car Insurance Premiums ($)` = col_double(),
##   `Losses incurred by insurance companies for collisions per insured driver ($)` = col_double()
## )
```

```
head(bad_drivers)
```

```
## # A tibble: 6 x 8
##   State `Number of driv~ `Percentage Of ~ `Percentage Of ~ `Percentage Of ~
##   <chr>            <dbl>            <dbl>            <dbl>            <dbl>
## 1 Alab~             18.8               39               30               96
## 2 Alas~             18.1               41               25               90
## 3 Ariz~             18.6               35               28               84
## 4 Arka~             22.4               18               26               94
## 5 Cali~             12                 35               28               91
## 6 Colo~             13.6               37               28               79
## # ... with 3 more variables: `Percentage Of Drivers Involved In Fatal
## #   Collisions Who Had Not Been Involved In Any Previous Accidents` <dbl>, `Car
## #   Insurance Premiums ($)` <dbl>, `Losses incurred by insurance companies for
## #   collisions per insured driver ($)` <dbl>
```

In the next, we rename columns to replace big column names with shorter names.

- **glimpse()** function is from **tibble** package, used to see every column in a data frame.

```
# rename columns
colnames(bad_drivers) <- c("STATE",
                           "DRIVERS_INVOLVED",
                           "PERC_DRIVERS_SPEED",
                           "PERC_DRIVERS_ALCHO",
                           "PERC_DRIVERS_NOT_DIST",
                           "PERC_DRIVERS_NO_ACC",
                           "INS_PREM",
                           "LOSS_INSCOMP")

glimpse(bad_drivers)
```

```
## Observations: 51
```

```
## Variables: 8
## $ STATE                 <chr> "Alabama", "Alaska", "Arizona", "Arkansas", "...
## $ DRIVERS_INVOLVED      <dbl> 18.8, 18.1, 18.6, 22.4, 12.0, 13.6, 10.8, 16....
## $ PERC_DRIVERS_SPEED    <dbl> 39, 41, 35, 18, 35, 37, 46, 38, 34, 21, 19, 5...
## $ PERC_DRIVERS_ALCHO    <dbl> 30, 25, 28, 26, 28, 28, 36, 30, 27, 29, 25, 4...
## $ PERC_DRIVERS_NOT_DIST <dbl> 96, 90, 84, 94, 91, 79, 87, 87, 100, 92, 95, ...
## $ PERC_DRIVERS_NO_ACC   <dbl> 80, 94, 96, 95, 89, 95, 82, 99, 100, 94, 93, ...
## $ INS_PREM              <dbl> 784.55, 1053.48, 899.47, 827.34, 878.41, 835....
## $ LOSS_INSCOMP          <dbl> 145.08, 133.93, 110.35, 142.39, 165.63, 139.9...
```

Data wrangling/visualization using dplyr, tidyr and ggplot2 packages As we must have noticed columns PERC_DRIVERS_SPEED, PERC_DRIVERS_ALCHO, PERC_DRIVERS_NOT_DIST, PERC_DRIVERS_NO_ACC are percentages of DRIVERS_INVOLVED. In the next step we will mutate new columns DRIVERS_SPEED, DRIVERS_ALCHO, DRIVERS_NOT_DIST, DRIVERS_NO_ACC by taking the given percentage of DRIVERS_INVOLVED column.

- **mutate()** function is from **dplyr** package, adds new variables and preserves existing ones.

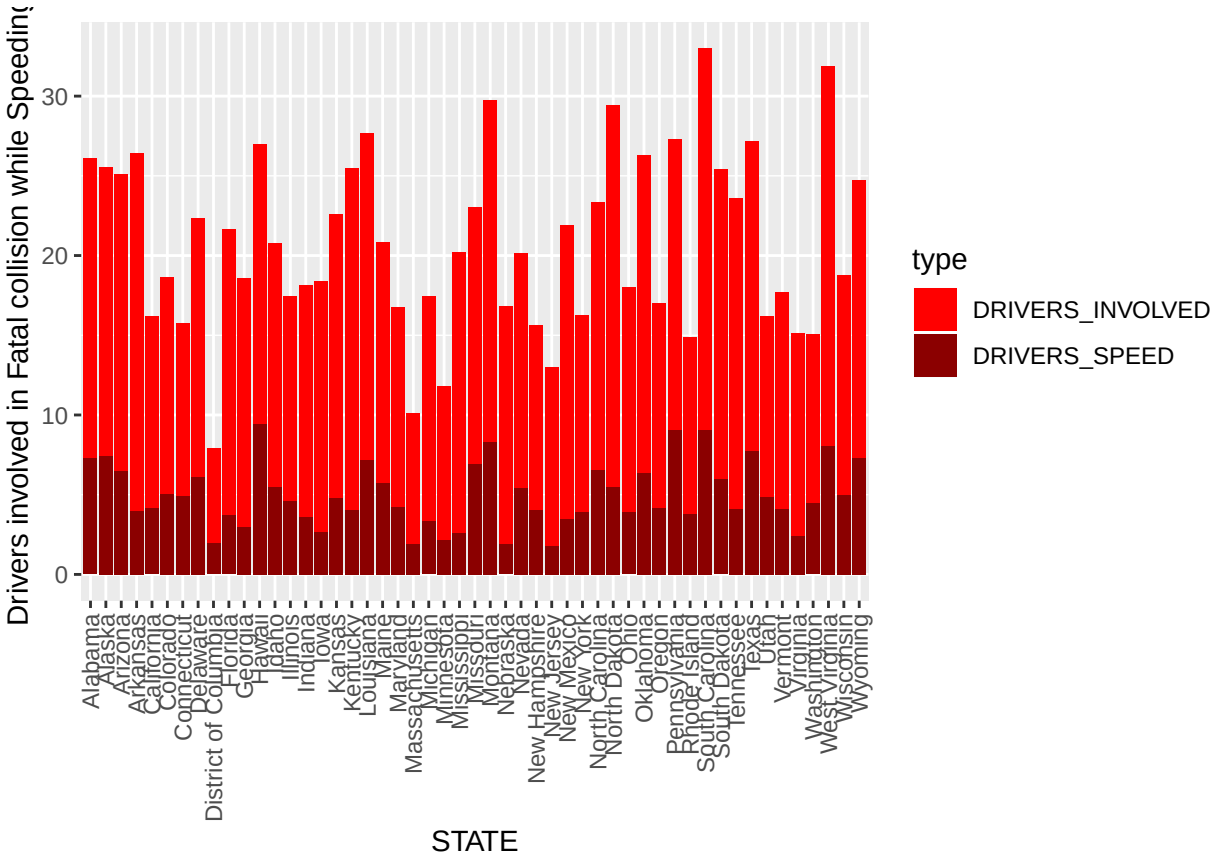
```
# create new column DRIVERS_SPEED which will be (DRIVERS_INVOLVED*PERC_DRIVERS_SPEED)/100
bad_drivers <- bad_drivers %>%
  mutate(DRIVERS_SPEED=(DRIVERS_INVOLVED*PERC_DRIVERS_SPEED)/100) %>%
  mutate(DRIVERS_ALCHO=(DRIVERS_INVOLVED*PERC_DRIVERS_ALCHO)/100) %>%
  mutate(DRIVERS_NOT_DIST=(DRIVERS_INVOLVED*PERC_DRIVERS_NOT_DIST)/100) %>%
  mutate(DRIVERS_NO_ACC=(DRIVERS_INVOLVED*PERC_DRIVERS_NO_ACC)/100)

glimpse(bad_drivers)
```

```
## Observations: 51
## Variables: 12
## $ STATE                 <chr> "Alabama", "Alaska", "Arizona", "Arkansas", "...
## $ DRIVERS_INVOLVED      <dbl> 18.8, 18.1, 18.6, 22.4, 12.0, 13.6, 10.8, 16....
## $ PERC_DRIVERS_SPEED    <dbl> 39, 41, 35, 18, 35, 37, 46, 38, 34, 21, 19, 5...
## $ PERC_DRIVERS_ALCHO    <dbl> 30, 25, 28, 26, 28, 28, 36, 30, 27, 29, 25, 4...
## $ PERC_DRIVERS_NOT_DIST <dbl> 96, 90, 84, 94, 91, 79, 87, 87, 100, 92, 95, ...
## $ PERC_DRIVERS_NO_ACC   <dbl> 80, 94, 96, 95, 89, 95, 82, 99, 100, 94, 93, ...
## $ INS_PREM              <dbl> 784.55, 1053.48, 899.47, 827.34, 878.41, 835....
## $ LOSS_INSCOMP          <dbl> 145.08, 133.93, 110.35, 142.39, 165.63, 139.9...
## $ DRIVERS_SPEED         <dbl> 7.332, 7.421, 6.510, 4.032, 4.200, 5.032, 4.9...
## $ DRIVERS_ALCHO         <dbl> 5.640, 4.525, 5.208, 5.824, 3.360, 3.808, 3.8...
## $ DRIVERS_NOT_DIST      <dbl> 18.048, 16.290, 15.624, 21.056, 10.920, 10.74...
## $ DRIVERS_NO_ACC        <dbl> 15.040, 17.014, 17.856, 21.280, 10.680, 12.92...
```

In this step we will draw a stacked bar lot using ggplot() method having states on X axis and DRIVERS_SPEED and DRIVERS_INVOLVED stacked together on Y axis. To achive this we first used select() method to get required columns. The used gather() method to make data long for DRIVERS_INVOLVED and DRIVERS_SPEED and finally used ggplot() to draw stacked bar plot.

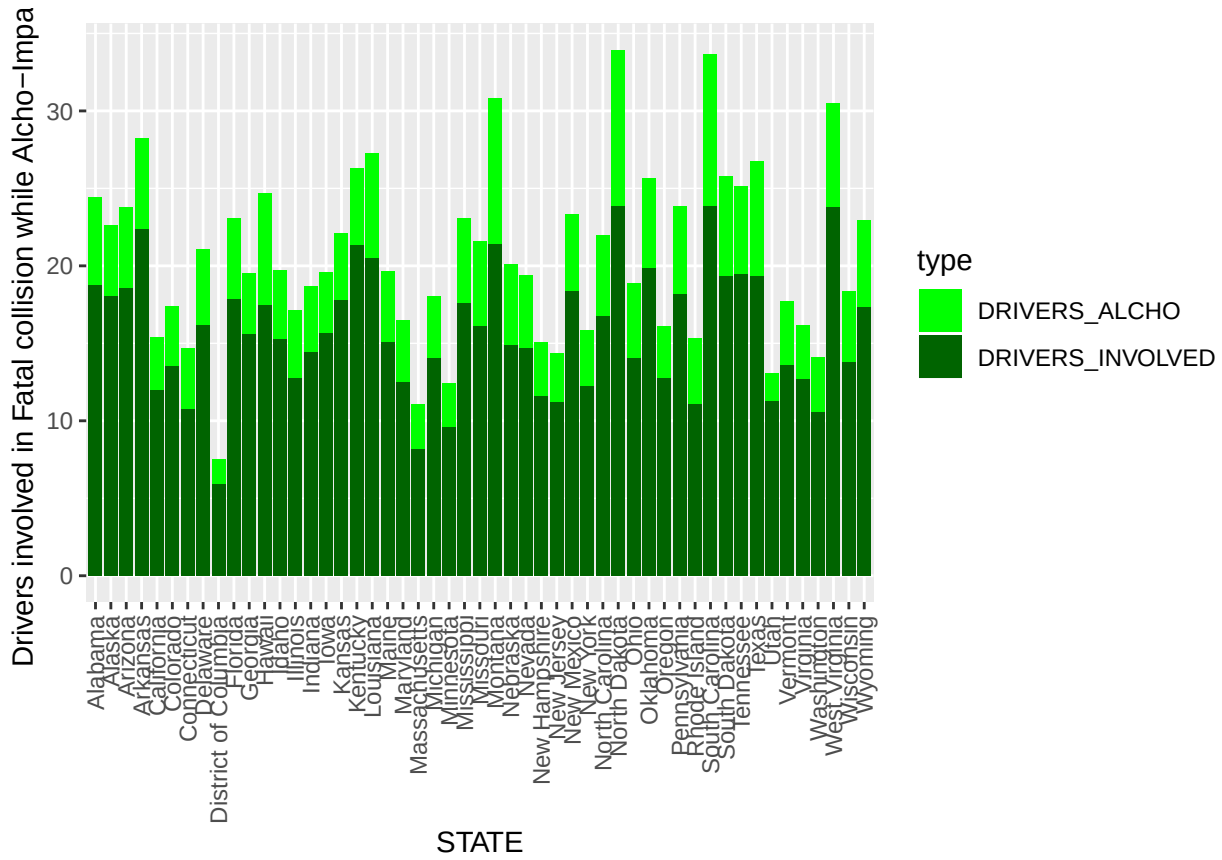
- **select()** function is from **dplyr** package that keeps only the variables we mention.
- **gather()** function is from **tidyr** package that takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed.
- **ggplot()** All **ggplot2** plots begin with a call to ggplot().

```
bad_drivers %>%
  select(STATE, DRIVERS_INVOLVED, DRIVERS_SPEED) %>%
  gather(type, value, DRIVERS_INVOLVED:DRIVERS_SPEED) %>%
  ggplot(., aes(x = STATE,y = value, fill = type)) +
  geom_bar(position = "stack", stat="identity") +
  scale_fill_manual(values = c("red", "darkred")) +
  ylab("Drivers involved in Fatal collision while Speeding") +
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5))
```



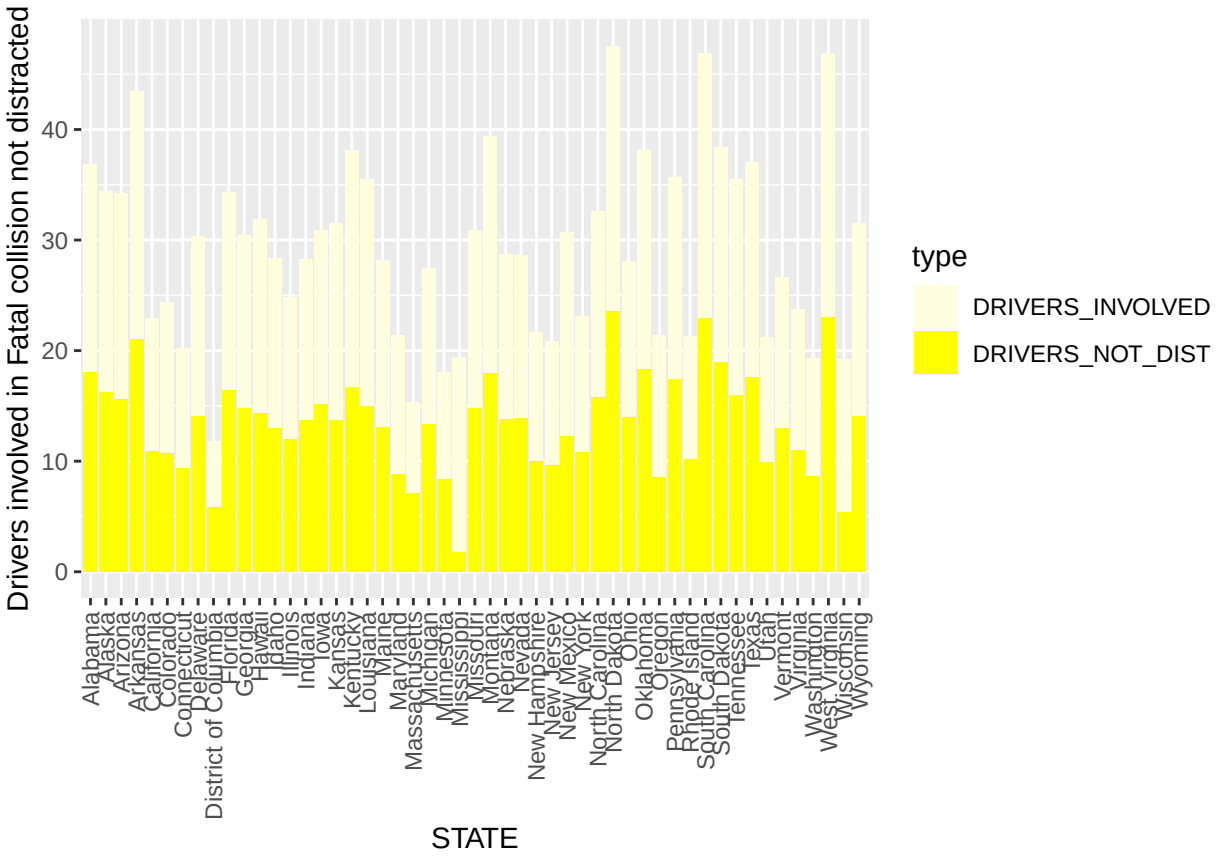
Similarly next stacked plot is having states on X axis and DRIVERS_ALCHO and DRIVERS_INVOLVED stacked together on Y axis. To achive this we first used select() method to get required columns. The used gather() method to make data long for DRIVERS_INVOLVED and DRIVERS_ALCHO and finally used ggplot() to draw stacked bar plot.

```
bad_drivers %>%
  select(STATE, DRIVERS_INVOLVED, DRIVERS_ALCHO) %>%
  gather(type, value, DRIVERS_INVOLVED:DRIVERS_ALCHO) %>%
  ggplot(., aes(x = STATE,y = value, fill = type)) +
  geom_bar(position = "stack", stat="identity") +
  scale_fill_manual(values = c("green", "darkgreen")) +
  ylab("Drivers involved in Fatal collision while Alcho-Impaired") +
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5))
```



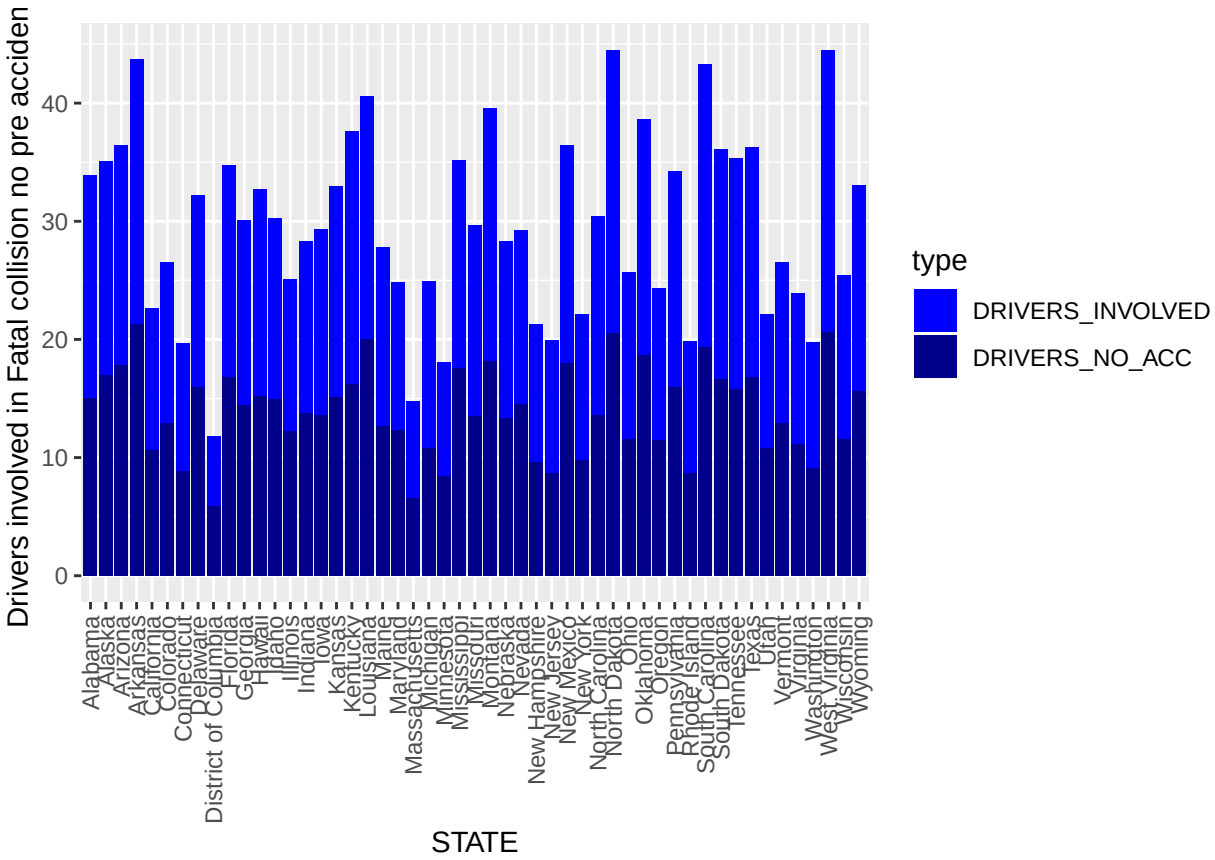
Next stacked plot is having states on X axis and DRIVERS_NOT_DIST and DRIVERS_INVOLVED stacked together on Y axis. To achive this we first used select() method to get required columns. The used gather() method to make data long for DRIVERS_INVOLVED and DRIVERS_NOT_DIST and finally used ggplot() to draw stacked bar plot.

```
bad_drivers %>%
  select(STATE, DRIVERS_INVOLVED, DRIVERS_NOT_DIST) %>%
  gather(type, value, DRIVERS_INVOLVED:DRIVERS_NOT_DIST) %>%
  ggplot(., aes(x = STATE,y = value, fill = type)) +
  geom_bar(position = "stack", stat="identity") +
  scale_fill_manual(values = c("lightyellow", "yellow")) +
  ylab("Drivers involved in Fatal collision not distracted") +
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5))
```



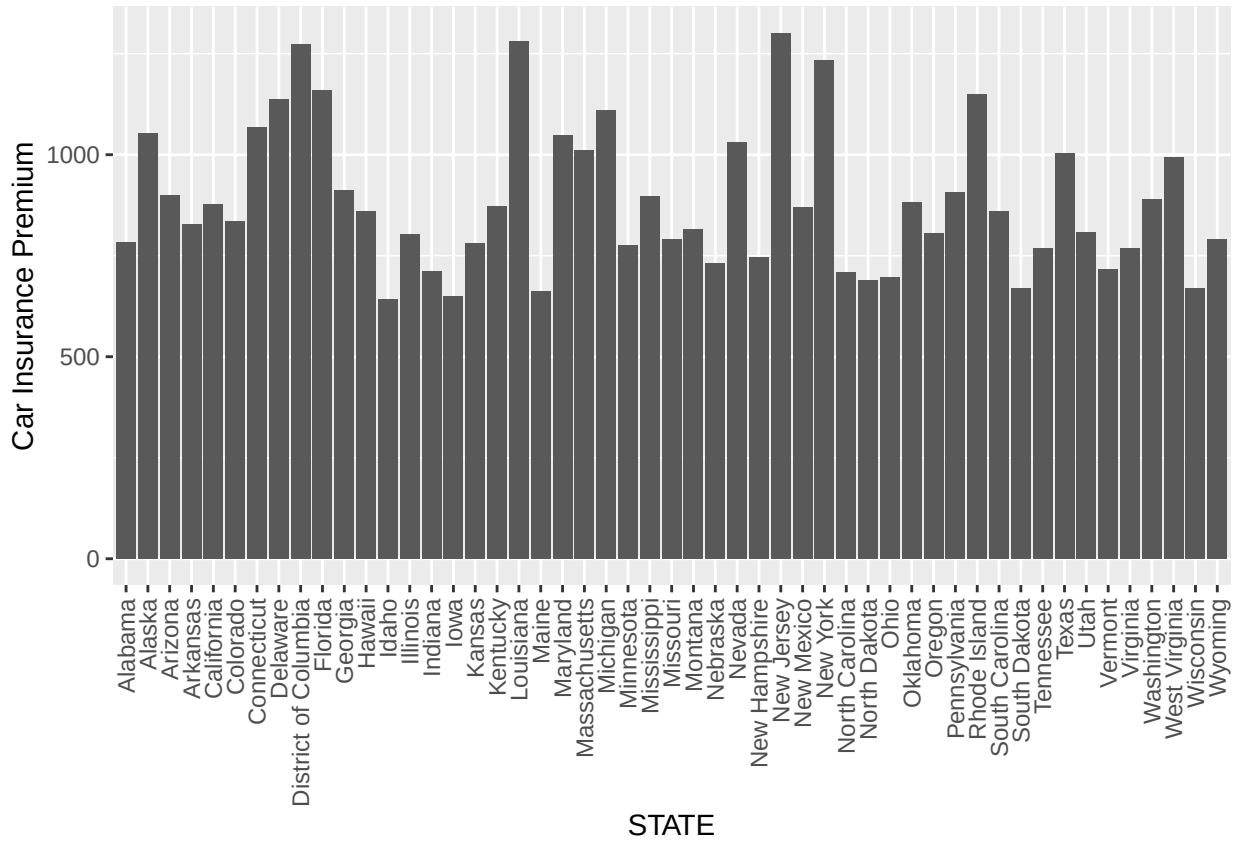
Next stacked plot is having states on X axis and DRIVERS_NO_ACC and DRIVERS_INVOLVED stacked together on Y axis. To achive this we first used select() method to get required columns. The used gather() method to make data long for DRIVERS_INVOLVED and DRIVERS_NO_ACC and finally used ggplot() to draw stacked bar plot.

```
bad_drivers %>%
  select(STATE, DRIVERS_INVOLVED, DRIVERS_NO_ACC) %>%
  gather(type, value, DRIVERS_INVOLVED:DRIVERS_NO_ACC) %>%
  ggplot(., aes(x = STATE,y = value, fill = type)) +
  geom_bar(position = "stack", stat="identity") +
  scale_fill_manual(values = c("blue", "darkblue")) +
  ylab("Drivers involved in Fatal collision no pre accident") +
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5))
```



Below plot of for STATE vs INS_PREMIUM that used ggplot() method to draw a bar plot.

```
bad_drivers %>%
  ggplot(., aes(x = STATE,y = INS_PREM)) +
  geom_bar(position = "stack", stat="identity") +
  ylab("Car Insurance Premium") +
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5))
```



Conclusion

Here we discussed various packages and their functions to explore bad drivers dataset. For complete set details refer (https://www.tidyverse.org/).

Resources:

- https://www.tidyverse.org/packages/
- https://fivethirtyeight.com/features/which-state-has-the-worst-drivers/

Gabe Extend

There are additional functions within dplyr and tidyr that allow us to transform or dive deeper into the data. The package ggplot2 also includes graphical tools that assist with statistical analysis.

We can perform mathematical operations to summarize the data using dplyr. Let's say we wanted to know the averages for all the columns, therefore giving us a descriptive statistic for all the columns.

We can use "summarise_if". This function will iterate through all the columns and determine if the column "is numeric" it will take the mean.

```
#save the data into a new variable
summmary <- bad_drivers %>% summarise_if(is.numeric, mean)
```

If we wanted to perform deeper descriptive statistics with dplyr, we can use the "summary" function.

```
summary(bad_drivers)
```

```
##     STATE           DRIVERS_INVOLVED PERC_DRIVERS_SPEED PERC_DRIVERS_ALCHO
##  Length:51          Min.   : 5.90    Min.   :13.00      Min.   :16.00
##  Class :character   1st Qu.:12.75    1st Qu.:23.00      1st Qu.:28.00
##  Mode  :character   Median :15.60    Median :34.00      Median :30.00
##                     Mean   :15.79    Mean   :31.73      Mean   :30.69
##                     3rd Qu.:18.50    3rd Qu.:38.00      3rd Qu.:33.00
##                     Max.   :23.90    Max.   :54.00      Max.   :44.00
##  PERC_DRIVERS_NOT_DIST PERC_DRIVERS_NO_ACC    INS_PREM       LOSS_INSCOMP
##  Min.   : 10.00        Min.   : 76.00      Min.   : 642.0   Min.   : 82.75
##  1st Qu.: 83.00        1st Qu.: 83.50      1st Qu.: 768.4   1st Qu.:114.64
##  Median : 88.00        Median : 88.00      Median : 859.0   Median :136.05
##  Mean   : 85.92        Mean   : 88.73      Mean   : 887.0   Mean   :134.49
##  3rd Qu.: 95.00        3rd Qu.: 95.00      3rd Qu.:1007.9   3rd Qu.:151.87
##  Max.   :100.00        Max.   :100.00      Max.   :1301.5   Max.   :194.78
##  DRIVERS_SPEED   DRIVERS_ALCHO    DRIVERS_NOT_DIST DRIVERS_NO_ACC
##  Min.   :1.792   Min.   : 1.593   Min.   : 1.76    Min.   : 5.90
##  1st Qu.:3.767   1st Qu.: 3.894   1st Qu.:10.48    1st Qu.:11.35
##  Median :4.608   Median : 4.554   Median :13.86    Median :13.78
##  Mean   :4.998   Mean   : 4.887   Mean   :13.57    Mean   :14.00
##  3rd Qu.:6.439   3rd Qu.: 5.604   3rd Qu.:16.14    3rd Qu.:16.75
##  Max.   :9.450   Max.   :10.038   Max.   :23.66    Max.   :21.28
```

The summary function gives relevant information, that allows us to look for outliers in the data.

But what if our task was to concentrate on a specific region of the United States and therefore, we need to create a subset of the data.

Lets use the filter function.

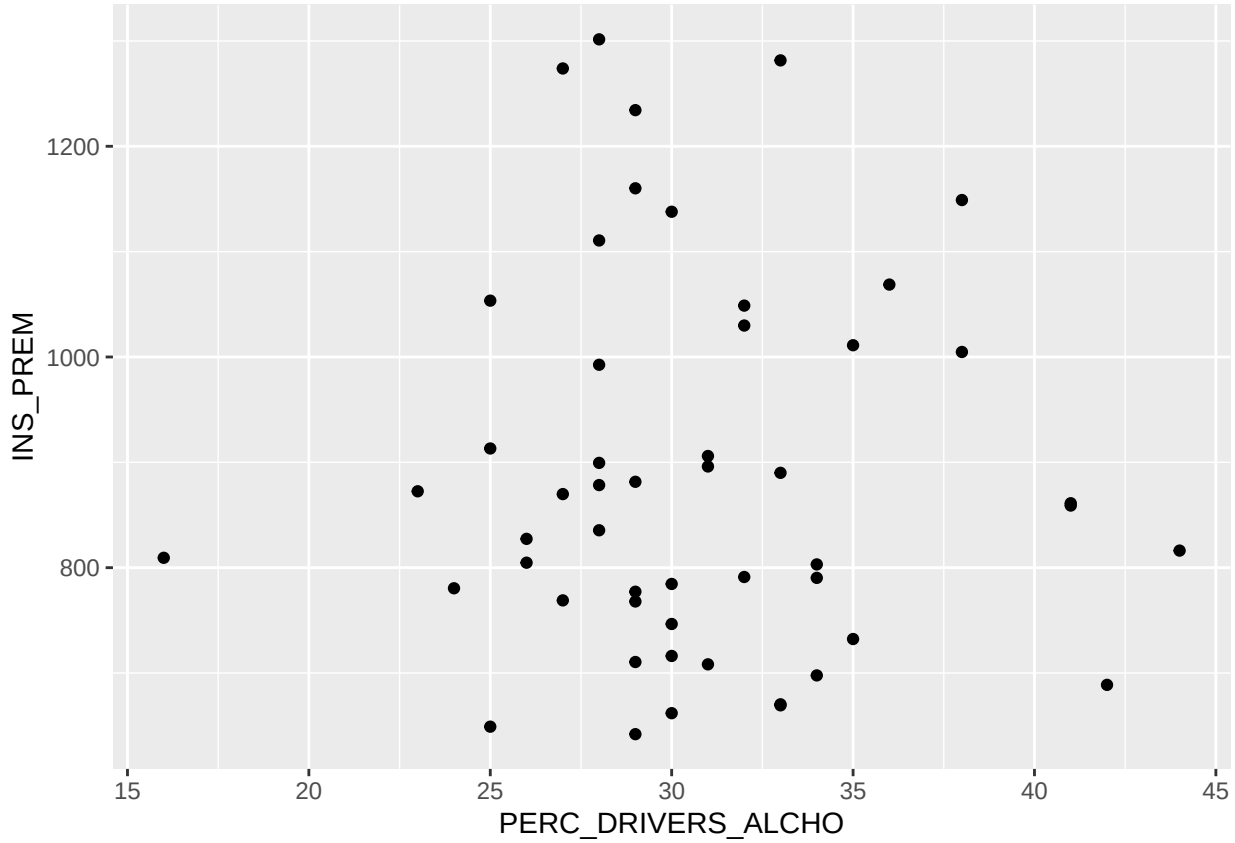
For the sake of brevity, only a couple northeastern states are included in the subset.

```
NorthEast <- bad_drivers %>% filter(STATE == "New York" | STATE == "Connecticut" | STATE == "Massachuse
```

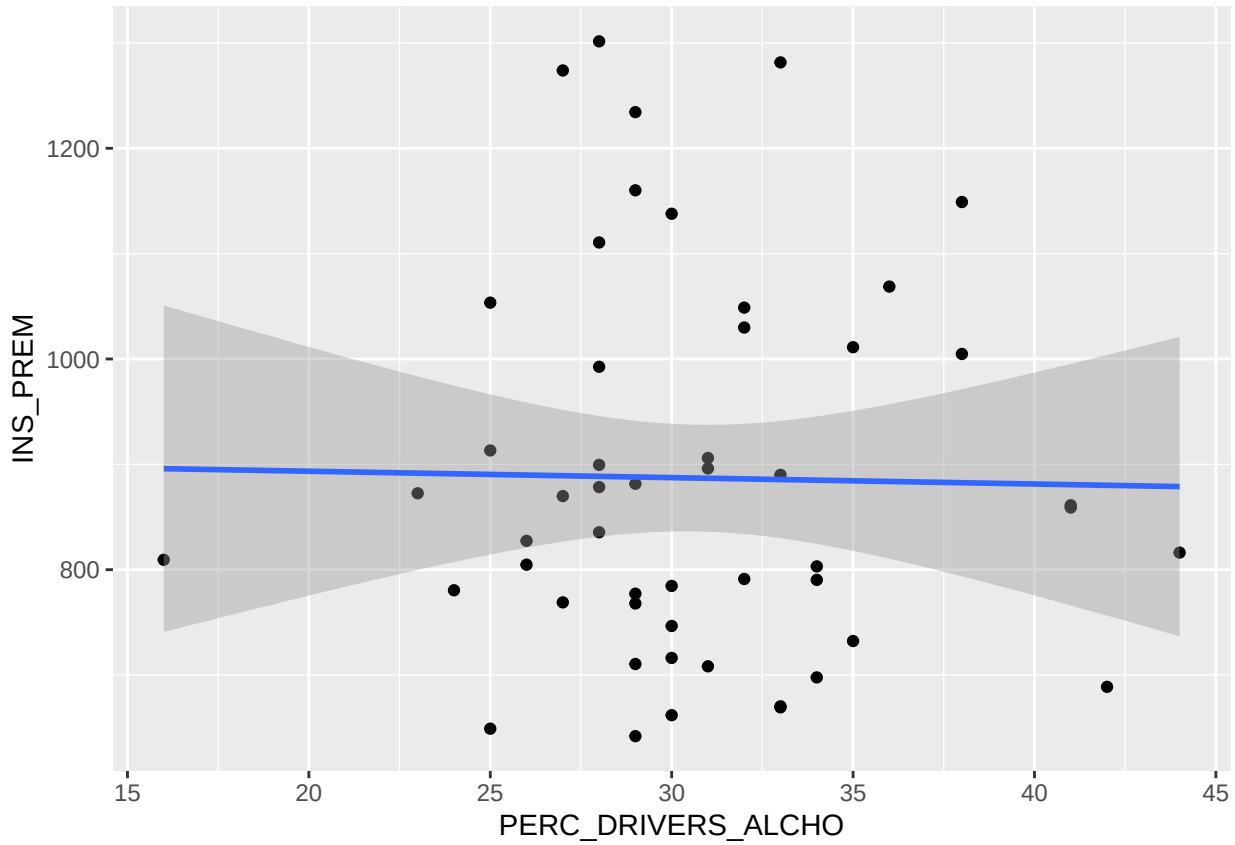
In the example above, the "|" signifies "or", so the filter function looks for and gathers all columns that is associated with the STATE variables "New York, Connecticut, Massachusetts", etc.

In statistical analysis, scatter plots are usedd to compare the relationship between 2 variables. In the bad drivers data set, lets see if there is a linear relationship between insurance premiums and percent of drivers caught drinking.

```
ggplot(bad_drivers, aes(x = PERC_DRIVERS_ALCHO, y=INS_PREM)) + geom_point()
```



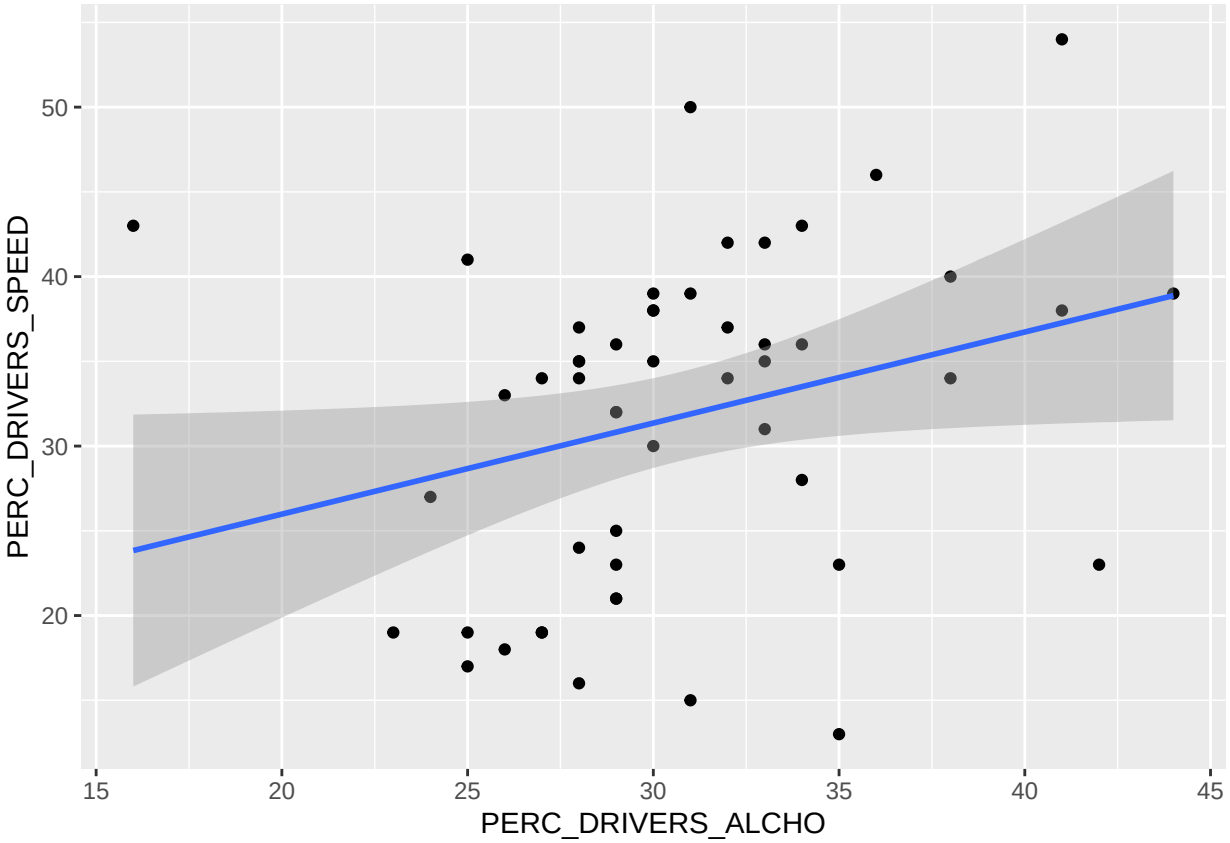

```
#Let's add a linear regression line

ggplot(bad_drivers, aes(x = PERC_DRIVERS_ALCHO, y=INS_PREM)) + geom_point() + geom_smooth(method=lm)
```



"Geom_smooth" adds a linear regression line (by default includes 95% confidence region).

Let's examine the relationship between the percentage of drives intoxicated and those speeding.

```
ggplot(bad_drivers, aes(x = PERC_DRIVERS_ALCHO, y=PERC_DRIVERS_SPEED)) + geom_point() + geom_smooth(metl
```


PERC_DRIVERS_SPEED
50
40
30
20
15
20
25
30
35
40
45
PERC_DRIVERS_ALCHO