Report for MA615 Assignment-EDA for Strawberries

Chenxun Li

10/18/2020

1.Background

This is an assignment to help us know how data cleaning, data organization and EDA are generally happens. I use the strawberries for this assignment.

2.Data Cleaning

(1)Read Data

At first, we read the data.

These data were collected from the USDA database selector: https://quickstats.nass.usda.gov

The data were stored online and then downloaded as a CSV file.

```
original_berries <- read_csv("~/Desktop/615/class17/berries(3).csv", col_names = TRUE)
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   Year = col_double(),
##   `Week Ending` = col_logical(),
##   `Ag District` = col_logical(),
##   `Ag District Code` = col_logical(),
##   County = col_logical(),
##   `County ANSI` = col_logical(),
##   `Zip Code` = col_logical(),
##   Region = col_logical(),
##   Watershed = col_logical(),
##   `CV (%)` = col_logical()
## )

## See spec(...) for full column specifications.
```

(2)Data Preparing

We can find that the data selected from the NASS database often has columns without any data or with a single repeated Values, and the berries data had only 8 out of 21 columns containing meaningful data.

So, we need to remove some columns without any data.

```
## look at number of unique values in each column
original_berries %>% summarize_all(n_distinct) -> aa

## make a list of the columns with only one unique value
bb <- which(aa[1,]==1)
```

```
## list the 1-unique value column names
cn <- colnames(original_berries)[bb]

## remove the 1-unique columns from the data set
original_berries  <-  original_berries[,-all_of(bb)]
aa <-  aa[,-all_of(bb)]

## State name and the State ANSI code are (sort of) redundant
## Just keep the name
original_berries <-  original_berries[,-4]
aa <- aa[,-4]
```

(3)Choose commodity

In this assignment, I choose Strawberries as the commodity.

And, I find that some values are undisclosed and none, so I discard them.

```
#choose STRAWBERRIES
original_STRAWBERRIES <- filter(original_berries, Commodity=="STRAWBERRIES")

#discard useless values
original_STRAWBERRIES <- filter(original_STRAWBERRIES, Value != "(D)")
original_STRAWBERRIES <- filter(original_STRAWBERRIES, Value != "(NA)")
```

(4)Data Processing- ‘Data Item’

I find there are much information in the column ‘Data Item’ and I only need the unit associated with the statistical category, so I use regular expression to gain the ‘unit_desc’.

```
#read the ’Data Item’
dt_item <- original_STRAWBERRIES$‘Data Item‘

#replace the ’-’ with ’,’ to prepare for spliting
dt_item_with_comma <- gsub(" - ",",",dt_item)

#extract ’MEASURED IN’
original_STRAWBERRIES$unit_desc <- str_extract_all(dt_item_with_comma,"MEASURED.*[^./AVG]|ACRES.*")
```

Now, we can see the categories of measurement.

```
unique(original_STRAWBERRIES$unit_desc)
```

```
## [[1]]
## [1] "MEASURED IN $ / CWT"
##
## [[2]]
## [1] "ACRES HARVESTED"
##
## [[3]]
## [1] "ACRES PLANTED"
##
## [[4]]
## [1] "MEASURED IN $"
##
## [[5]]
## [1] "MEASURED IN CWT"
##
## [[6]]
## [1] "MEASURED IN CWT / ACRE"
##
## [[7]]
## [1] "MEASURED IN LB"
##
## [[8]]
## [1] "MEASURED IN LB / ACRE / APPLICATION, "
##
## [[9]]
## [1] "MEASURED IN LB / ACRE / YEAR, "
##
## [[10]]
## [1] "MEASURED IN NUMBER, "
##
## [[11]]
## [1] "MEASURED IN PCT OF AREA BEARING, "
##
## [[12]]
## [1] "MEASURED IN $ / TON"
##
## [[13]]
## [1] "MEASURED IN TONS"
```

(5)Data Processing- ‘Domain’

I find that ‘Domain’ includes both characteristic of operations that produce a particular commodity and some details, so I separate it into two parts.

```
original_STRAWBERRIES <- separate(data=original_STRAWBERRIES, col =6,
                             into=c("Domain", "Domain.Detail"), sep=",")
```

```
## Warning: Expected 2 pieces. Missing pieces filled with ‘NA‘ in 579 rows [1, 2,
## 3, 4, 5, 6, 7, 8, 9, 83, 84, 85, 86, 156, 157, 158, 159, 229, 230, 231, ...].
```

After that, I find some pieces are filled with ‘NA’, that is the domain=TOTAL, which means there are no further breakouts and the domain=FERTILIZER. So we supplement them.

```
for(i in 1:length(original_STRAWBERRIES)){
  if(is.na(original_STRAWBERRIES$Domain.Detail[i]) == T){
    original_STRAWBERRIES$Domain.Detail[i] = original_STRAWBERRIES$Domain[i]
  }
}
```

(6)Delete duplicates

Now, we have the data set with 10 columns, but there also some variables are repetitive and useless.

i.’category’

I have choosn strawberries, so this column is useless.

ii.’Data Item’

I have gained the information that I am interested in from it, so I discard this column.

iii.’Domain Category’

I have separated the Domain into two parts, and one of them can reveal some information in Domain Category. Besides, I am not care about some other information in 'Domain Category' such as the price of chemicals.

So, I delete these variables.

```
original_STRAWBERRIES <- original_STRAWBERRIES[,-8]
original_STRAWBERRIES <- original_STRAWBERRIES[,-5]
original_STRAWBERRIES <- original_STRAWBERRIES[,-4]
```

(7)Change types of variables

I would like to change 'Value' from char to numeric for computing in EDA. However, there are some comma symbols in it(these data will turn to NA when use as.numeric), so I try to delete the comma symbols.

```
#delete the comma symbols in Values
for(i in 1:length(original_STRAWBERRIES$Value)){
  original_STRAWBERRIES$Value[i] <- gsub(pattern = ",", replacement = "",
                                original_STRAWBERRIES$Value[i])
}

#change the type
original_STRAWBERRIES$Year <- as.integer(original_STRAWBERRIES$Year)
original_STRAWBERRIES$Value <- as.numeric(original_STRAWBERRIES$Value)
```

```
## Warning: NAs introduced by coercion
```

The NAs are value=0, and I will process them in EDA.

(8)Finish cleaning

```
#get the new data set
STRAWBERRIES <- original_STRAWBERRIES
```

3.EDA-mean value grouped by unit in years&states

(1)Group by

At first, we need to select the information are interested in. I would like to research value in different years in different states. However, the unit is not uniform, so we group by the unit.

```
strawberry_unit_desc <- STRAWBERRIES %>% group_by(unit_desc) %>%
  summarize(
    state=State,
    year=Year,
    numbers=n(),
    value=Value
  )
```

```
## ‘summarise()‘ regrouping output by ’unit_desc’ (override with ‘.groups‘ argument)
```

(2)Plot for differnet units

Now, we can EDA for the new data set.

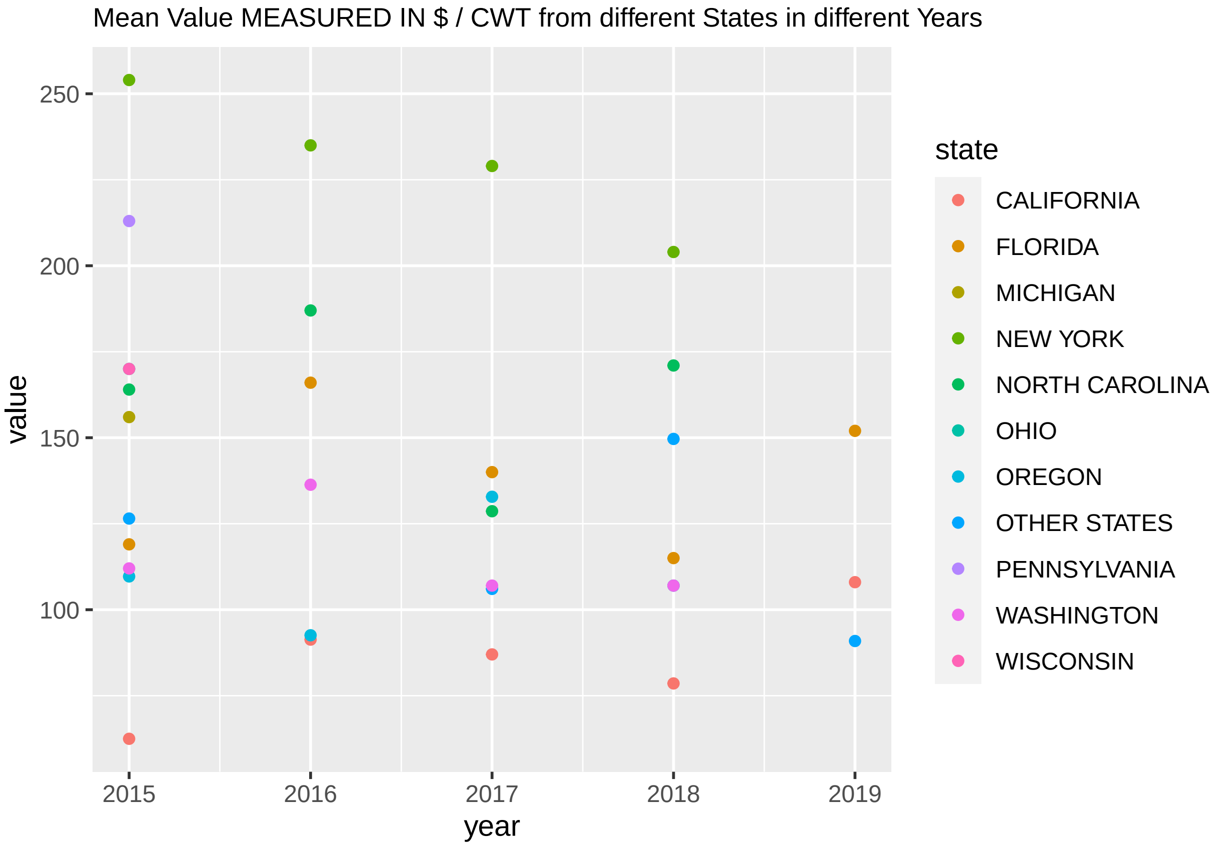
```
#create a data frame of different units
dt <- unique(strawberry_unit_desc$unit_desc)
dt <- do.call(rbind,dt)
```

```
## create the loop to perform 13 plots
plot_list <- list()
for(i in 1:length(dt)){
  #create flag variables
  strawberry_unit_desc_for_loop <- data.frame(strawberry_unit_desc)
  #extract specific unit from data set
  strawberry_unit_desc_specific <- filter(strawberry_unit_desc_for_loop, unit_desc==dt[i])
  #replace the value=0 to NA
  strawberry_unit_desc_specific$value[strawberry_unit_desc_specific$value==0] <- NA
  #I would like to research values from different States in different Years
  #so I choose group by year and state.
  strawberry_unit_desc_specific_new <- group_by(strawberry_unit_desc_specific, year,state)
  strawberry_specific <- summarize(strawberry_unit_desc_specific_new, value= mean(value, na.rm=T))
  #plot the mean value from different States in different Years
  plot_list[[i]] <- ggplot(strawberry_specific, aes(x=year, y=value))+geom_point(aes(color=state))+
    ggtitle(paste("Mean Value",dt[i],"from different States in different Years"))+
    theme(plot.title = element_text( size = 10))+
    scale_x_continuous(breaks=c(2015,2016,2017,2018,2019))
}
```

```
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
```

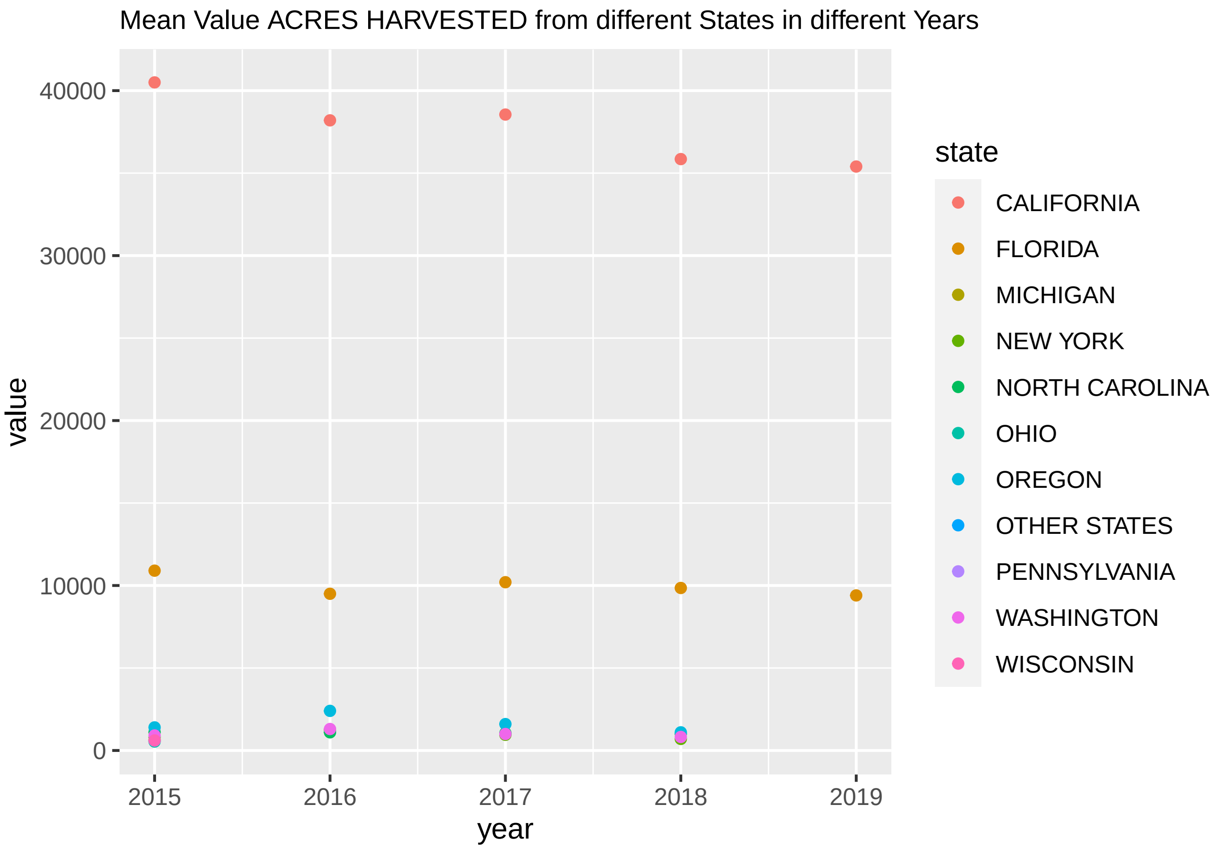
```
plot_list[[1]]
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



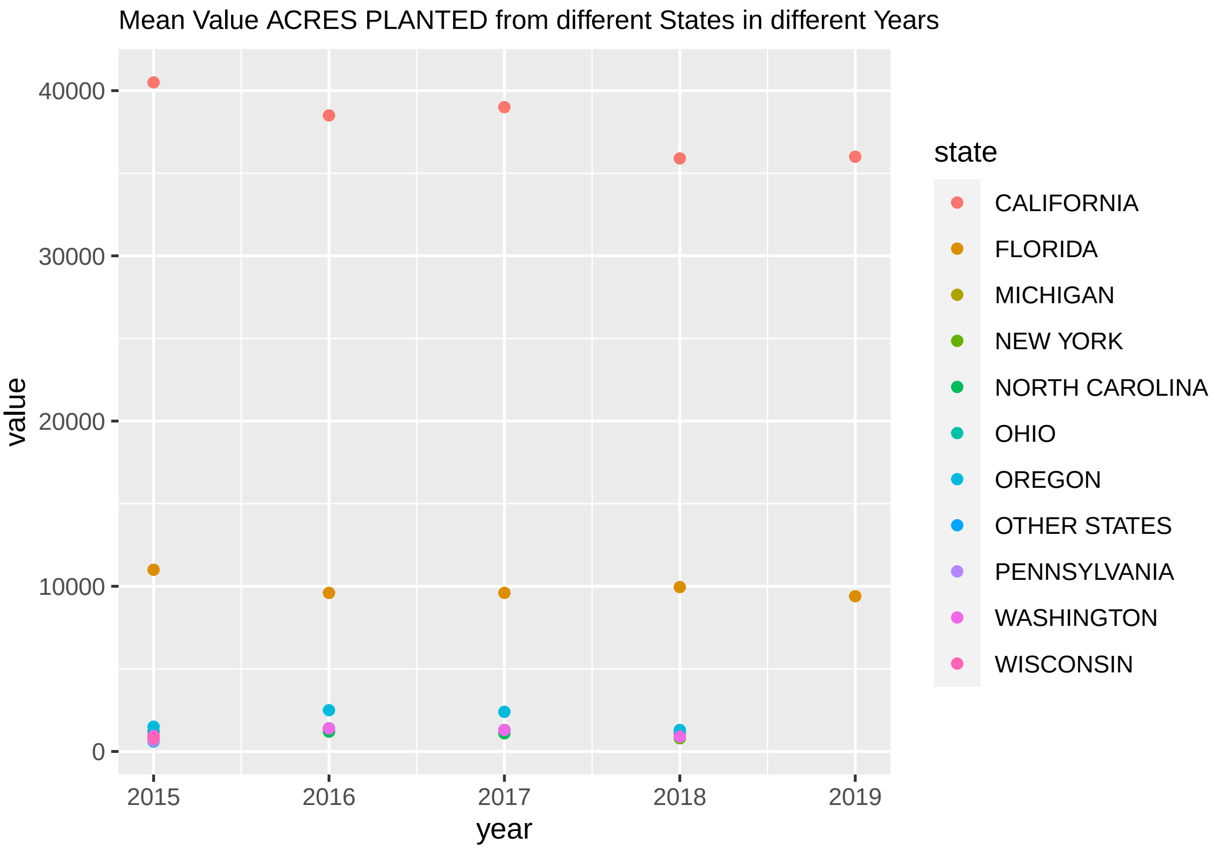
```
plot_list[[2]]
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



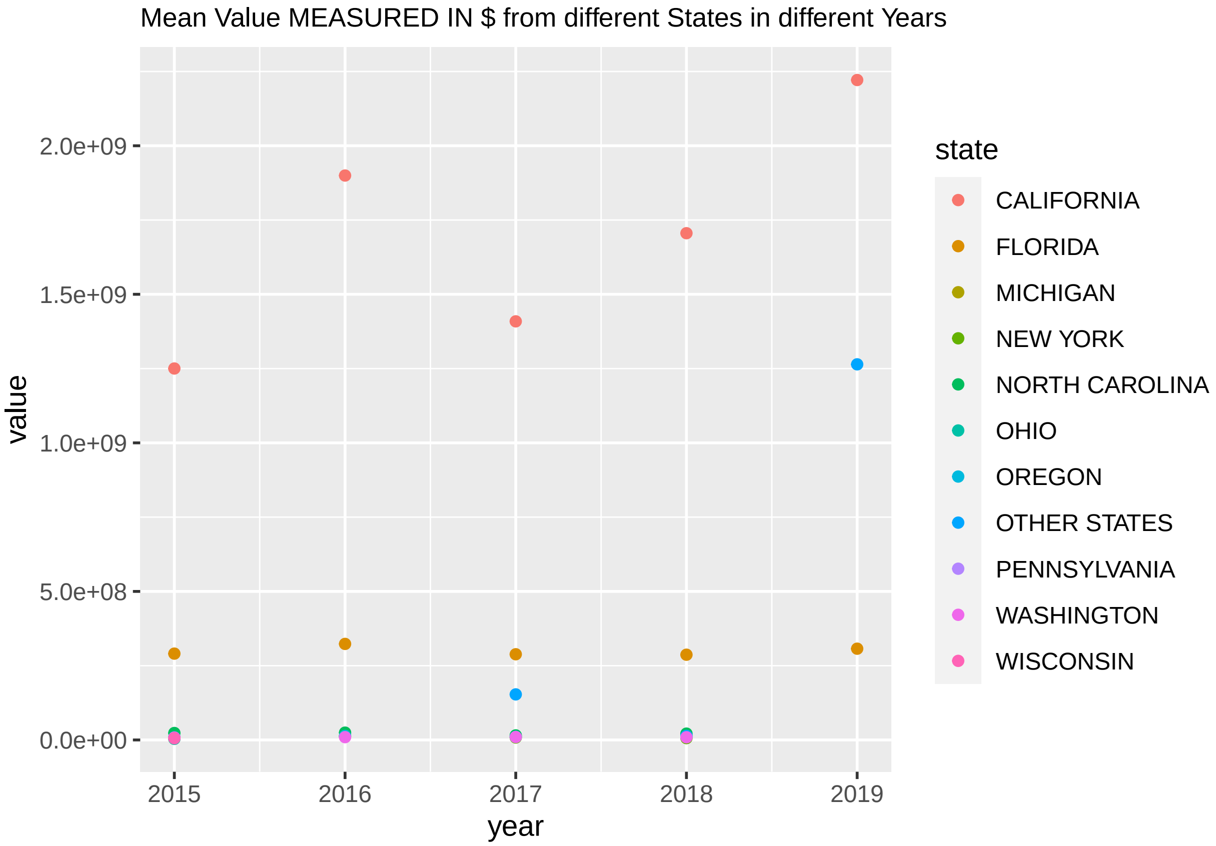
```
plot_list[[3]]
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



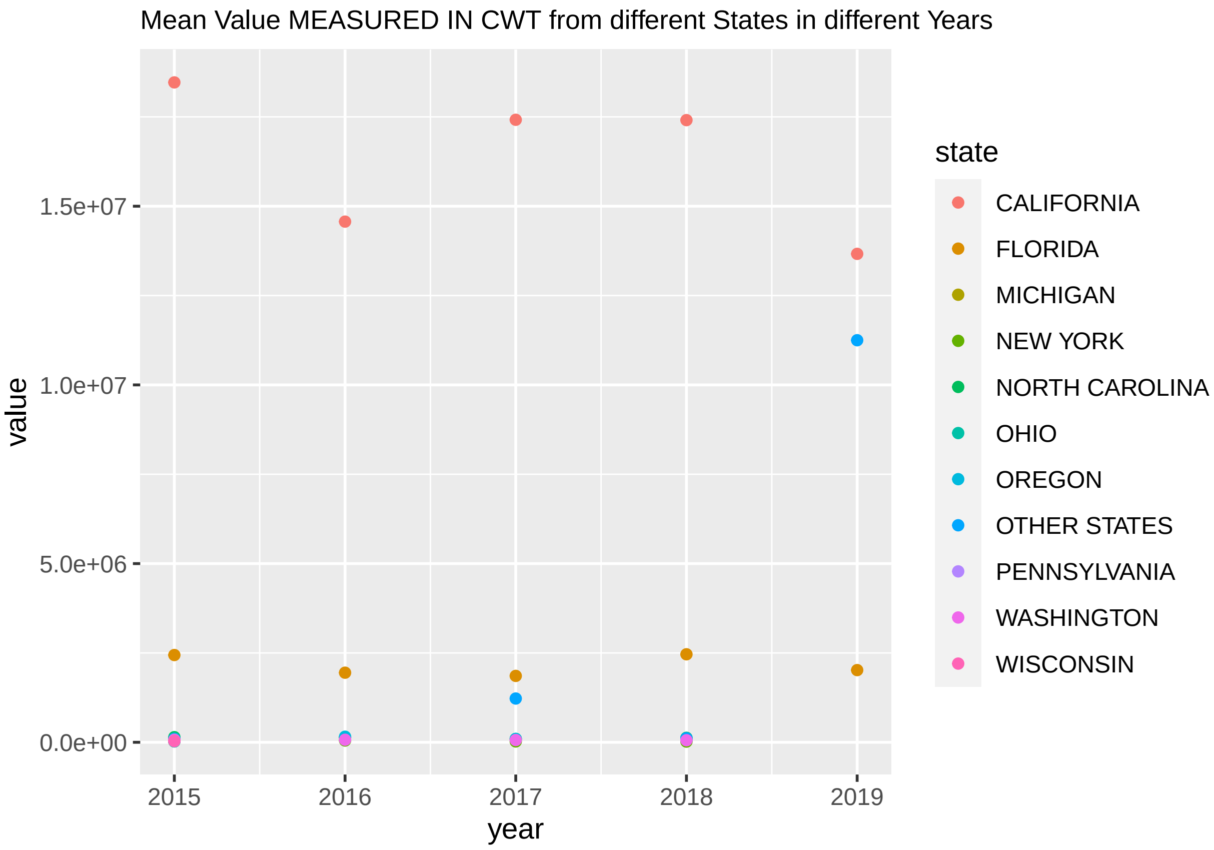
```
plot_list[[4]]
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



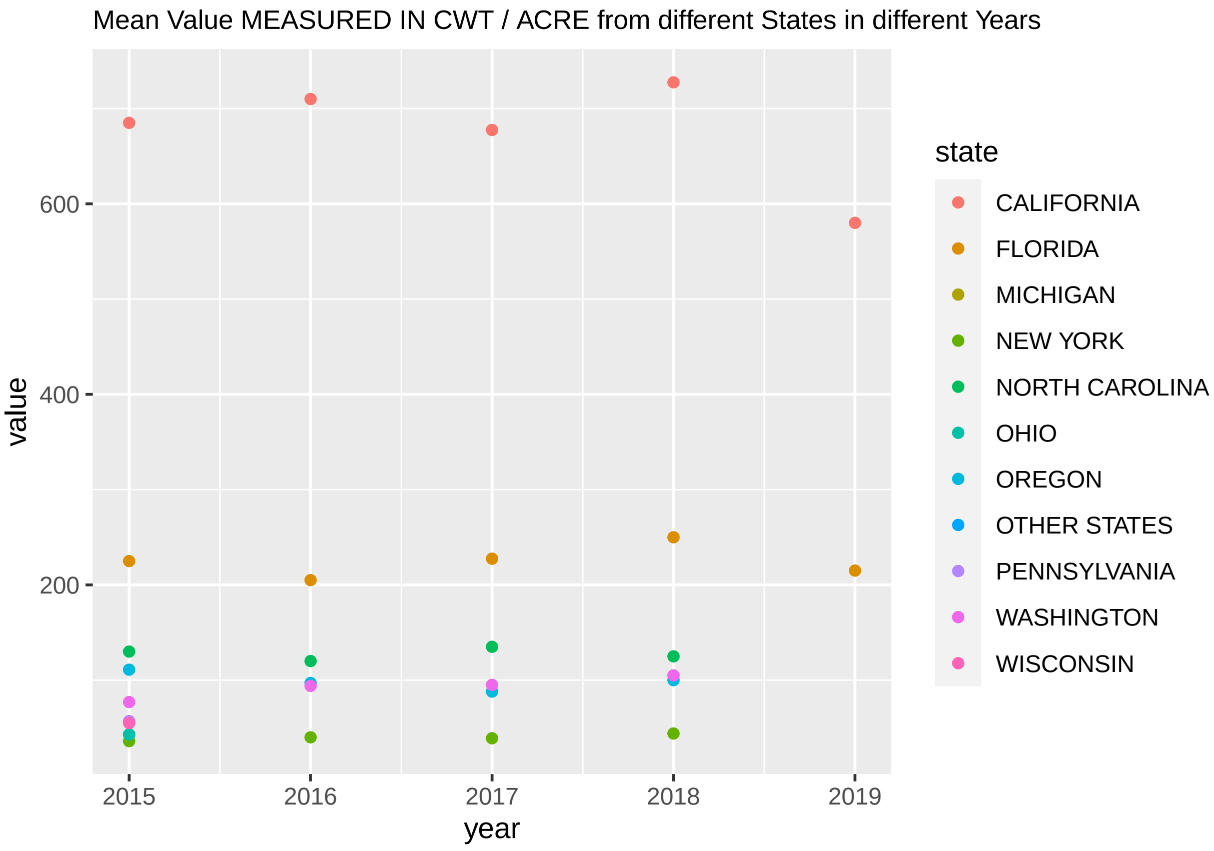
```
plot_list[[5]]
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

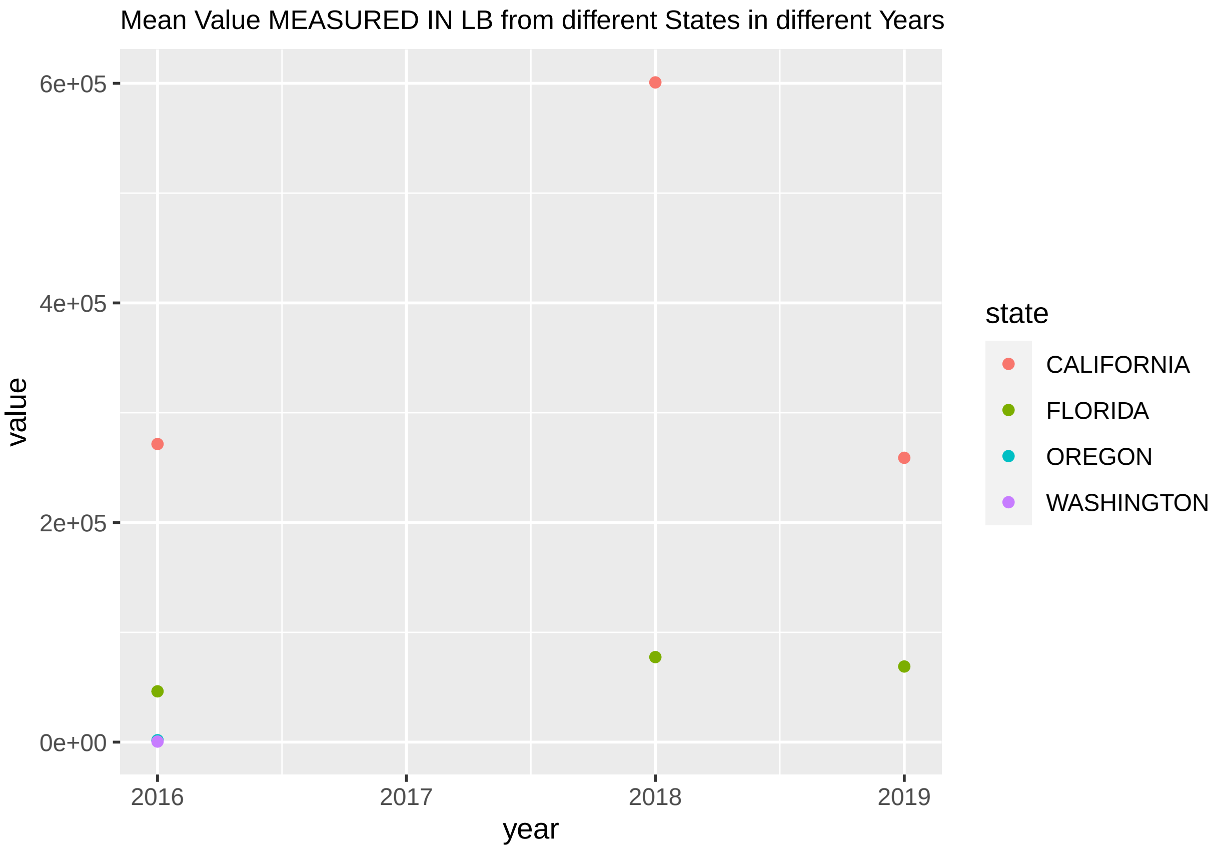


```
plot_list[[6]]
```

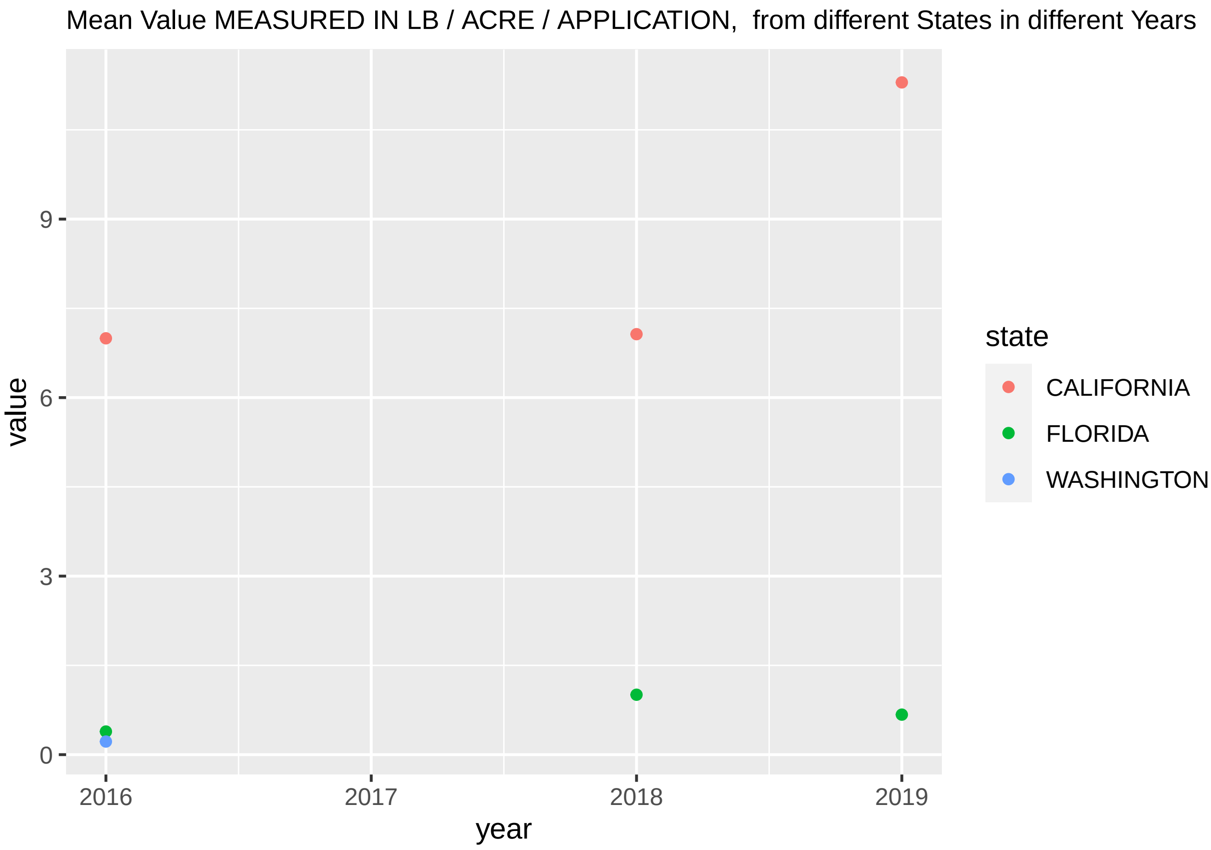
```
## Warning: Removed 2 rows containing missing values (geom_point).
```


Mean Value MEASURED IN CWT / ACRE from different States in different Years
value
600
400
200
2015
2016
2017
2018
2019
year
state
CALIFORNIA
FLORIDA
MICHIGAN
NEW YORK
NORTH CAROLINA
OHIO
OREGON
OTHER STATES
PENNSYLVANIA
WASHINGTON
WISCONSIN

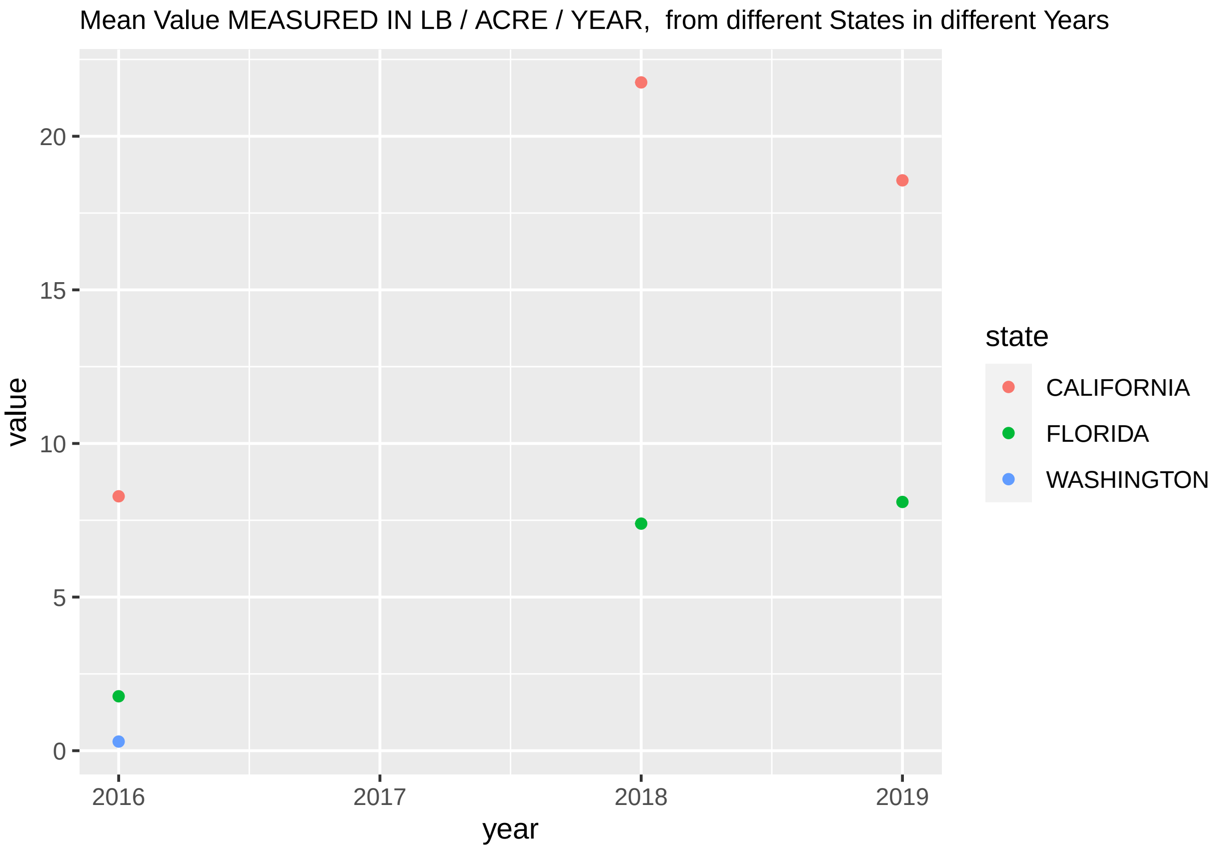
```
plot_list[[7]]
```


Mean Value MEASURED IN LB from different States in different Years
6e+05
4e+05
2e+05
0e+00
value
2016
2017
2018
2019
year
state
CALIFORNIA
FLORIDA
OREGON
WASHINGTON

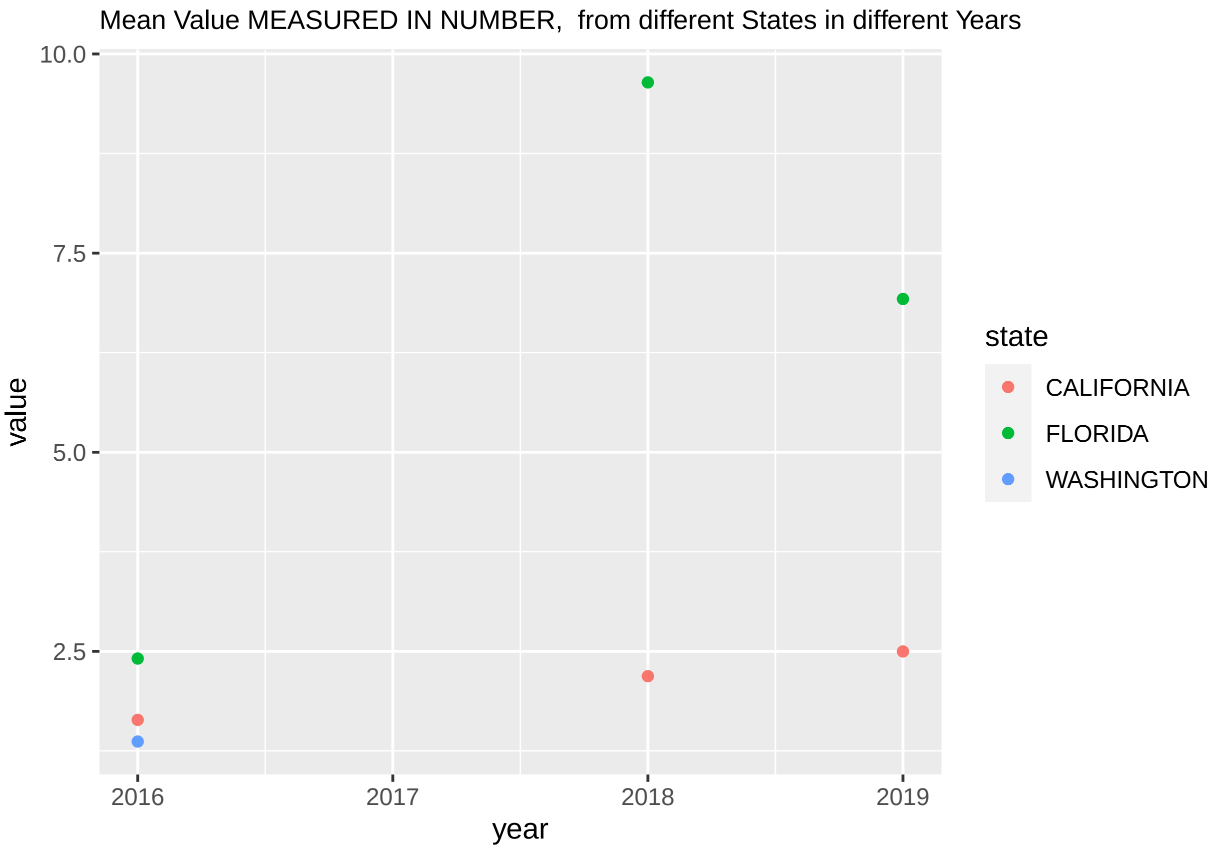
```
plot_list[[8]]
```


Mean Value MEASURED IN LB / ACRE / APPLICATION, from different States in different Years
value
0
3
6
9
year
2016
2017
2018
2019
state
CALIFORNIA
FLORIDA
WASHINGTON

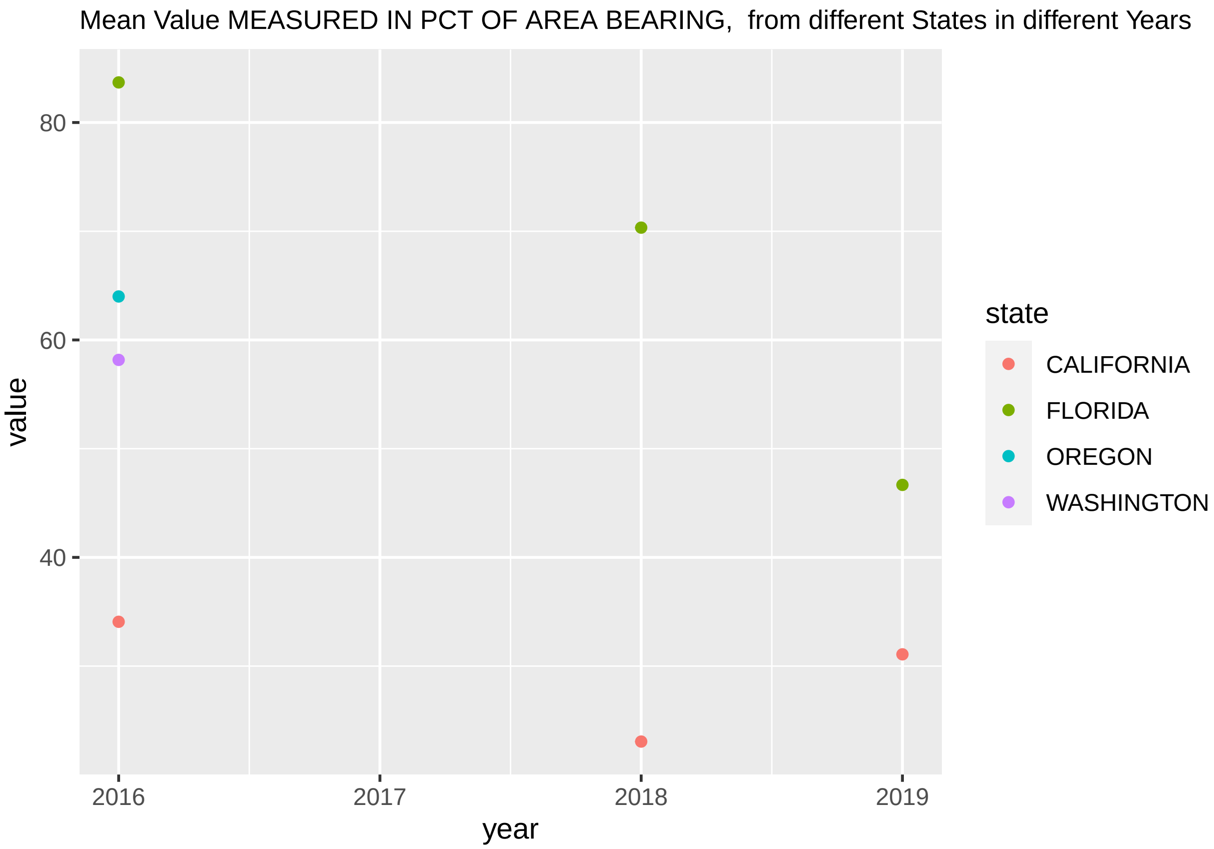
```
plot_list[[9]]
```


Mean Value MEASURED IN LB / ACRE / YEAR, from different States in different Years
value
20
15
10
5
0
2016
2017
2018
2019
year
state
CALIFORNIA
FLORIDA
WASHINGTON

```
plot_list[[10]]
```

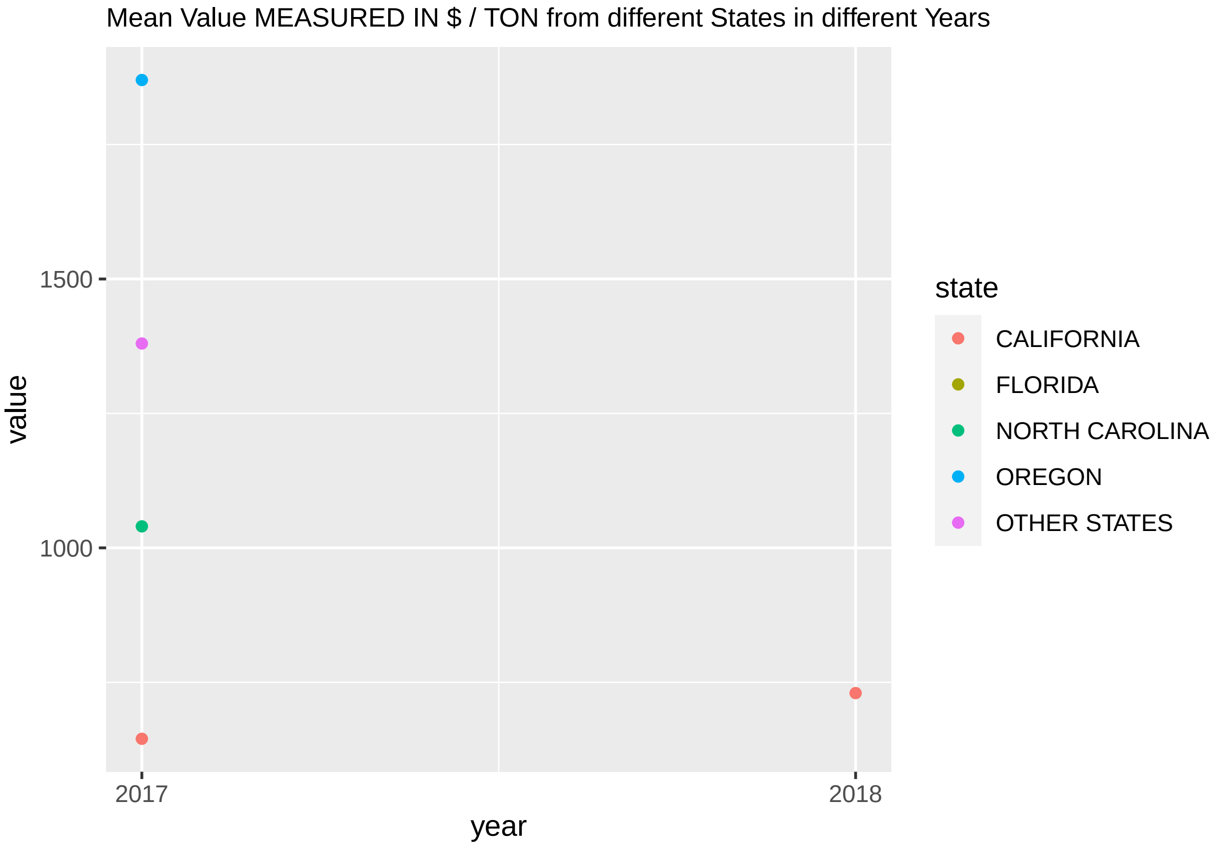

Mean Value MEASURED IN NUMBER, from different States in different Years
10.0
7.5
5.0
2.5
value
2016
2017
2018
2019
year
state
● CALIFORNIA
● FLORIDA
● WASHINGTON

```
plot_list[[11]]
```



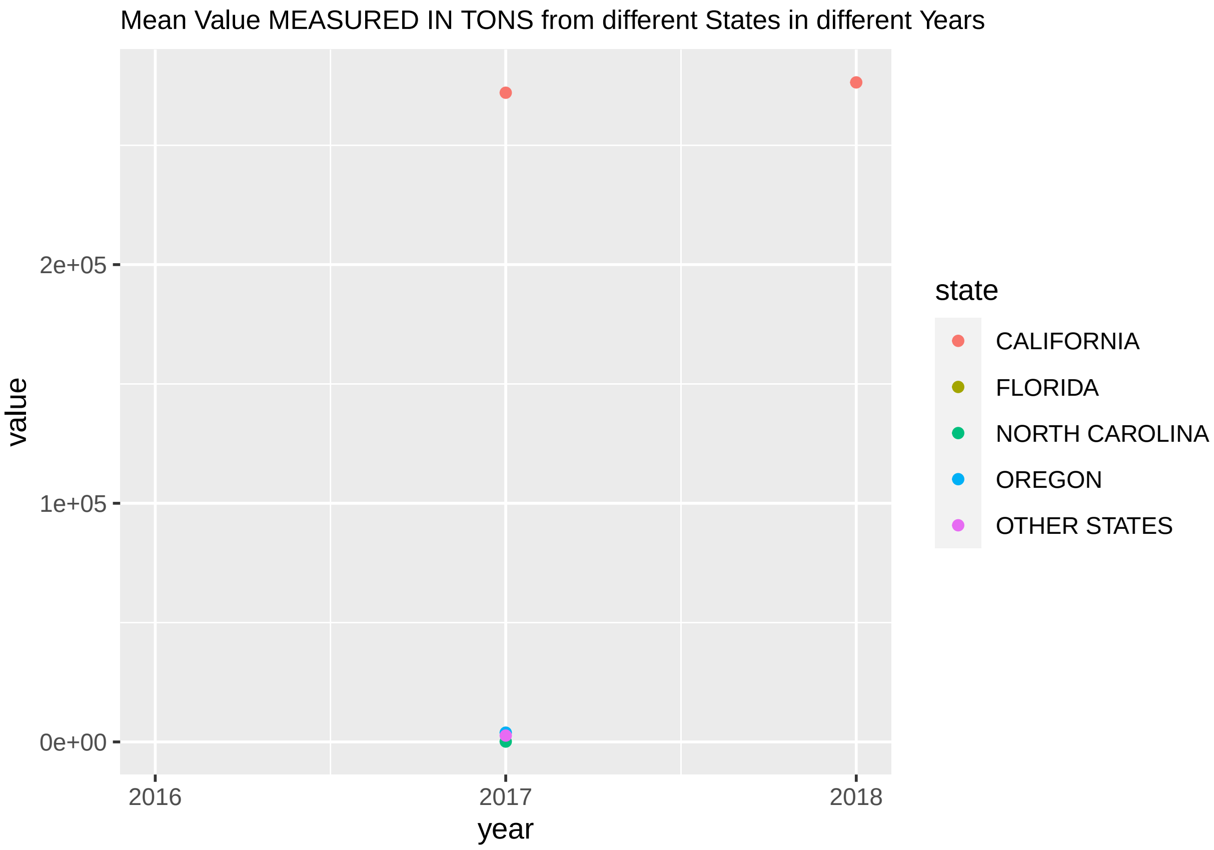
```
plot_list[[12]]
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



```
plot_list[[13]]
```

```
## Warning: Removed 3 rows containing missing values (geom_point).
```



(3)Result analysis

From the first picture, we can find the mean value in New York state is much higher than the other states, and I think probably because of its developed economics. And from the other pictures, we can find mean value in California is also higher than the other states.

4.EDA-mean value grouped by unit in years&domains

Now, I would like to research the relationship in years and domains grouped by unit.

(1)Group by

At first, I also group by the unit, but I choose domain as variable.

```
strawberry_unit_desc_2 <- STRAWBERRIES %>% group_by(unit_desc) %>%
  summarize(
    domain=Domain,
    year=Year,
    numbers=n(),
    value=Value
  )
```

```
## ‘summarise()‘ regrouping output by ’unit_desc’ (override with ‘.groups‘ argument)
```

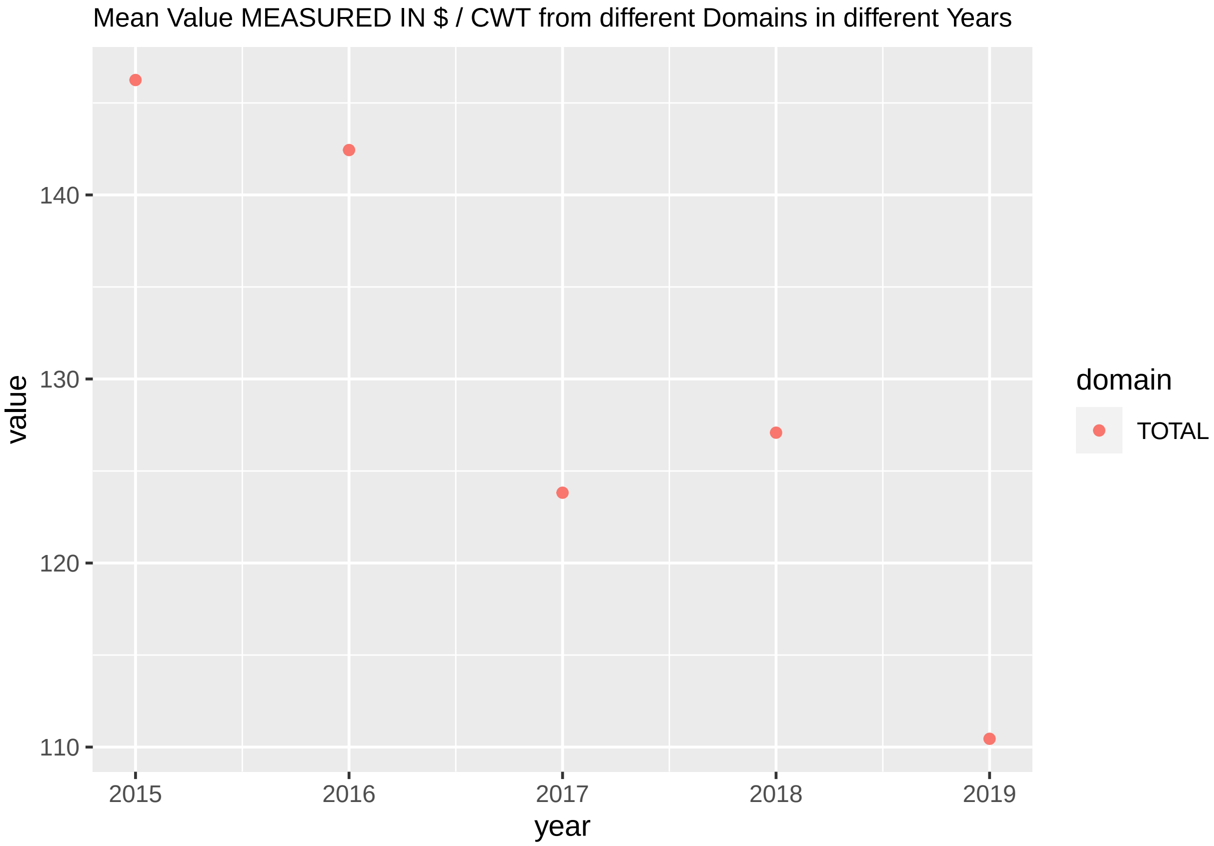
(2)Plot for differnet units

```
dt <- unique(strawberry_unit_desc$unit_desc)
dt <- do.call(rbind,dt)
```

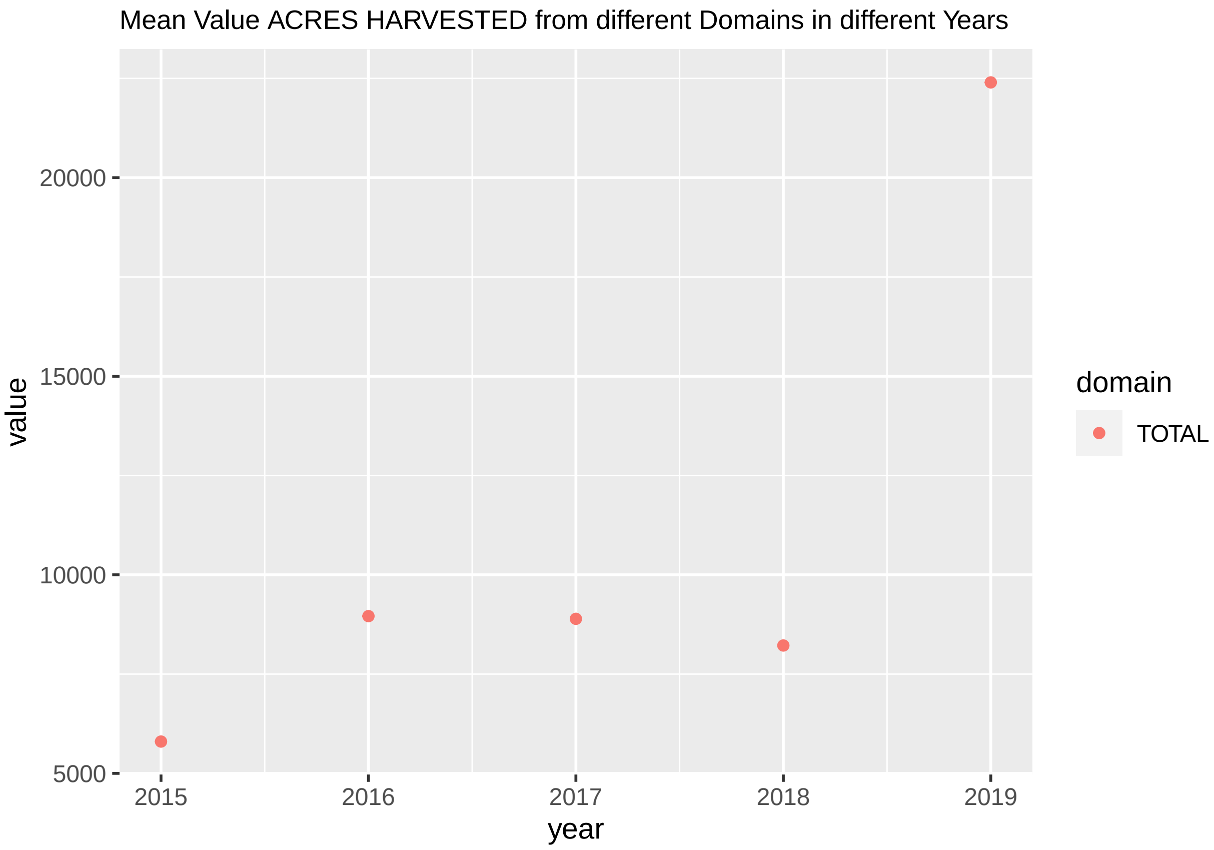
```
## create the loop to perform 13 plots
plot_list2 <- list()
for(i in 1:length(dt)){
  #create flag variables
  strawberry_unit_desc_for_loop <- data.frame(strawberry_unit_desc_2)
  #extract specific unit from data set
  strawberry_unit_desc_specific <- filter(strawberry_unit_desc_for_loop, unit_desc==dt[i])
  #replace the value=0 to NA
  strawberry_unit_desc_specific$value[strawberry_unit_desc_specific$value==0] <- NA
  #I would like to research values from different Domains in different Years
  #so I choose group by year and domain.
  strawberry_unit_desc_specific_new <- group_by(strawberry_unit_desc_specific, year,domain)
  strawberry_specific <- summarize(strawberry_unit_desc_specific_new, value= mean(value, na.rm=T))
  #plot the mean value from different Domains in different Years
  plot_list2[[i]] <- ggplot(strawberry_specific, aes(x=year, y=value))+geom_point(aes(color=domain))+
    ggtitle(paste("Mean Value",dt[i],"from different Domains in different Years"))+
    theme(plot.title = element_text( size = 10))+
    scale_x_continuous(breaks=c(2015,2016,2017,2018,2019))
}
```

```
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
## ‘summarise()‘ regrouping output by ’year’ (override with ‘.groups‘ argument)
```

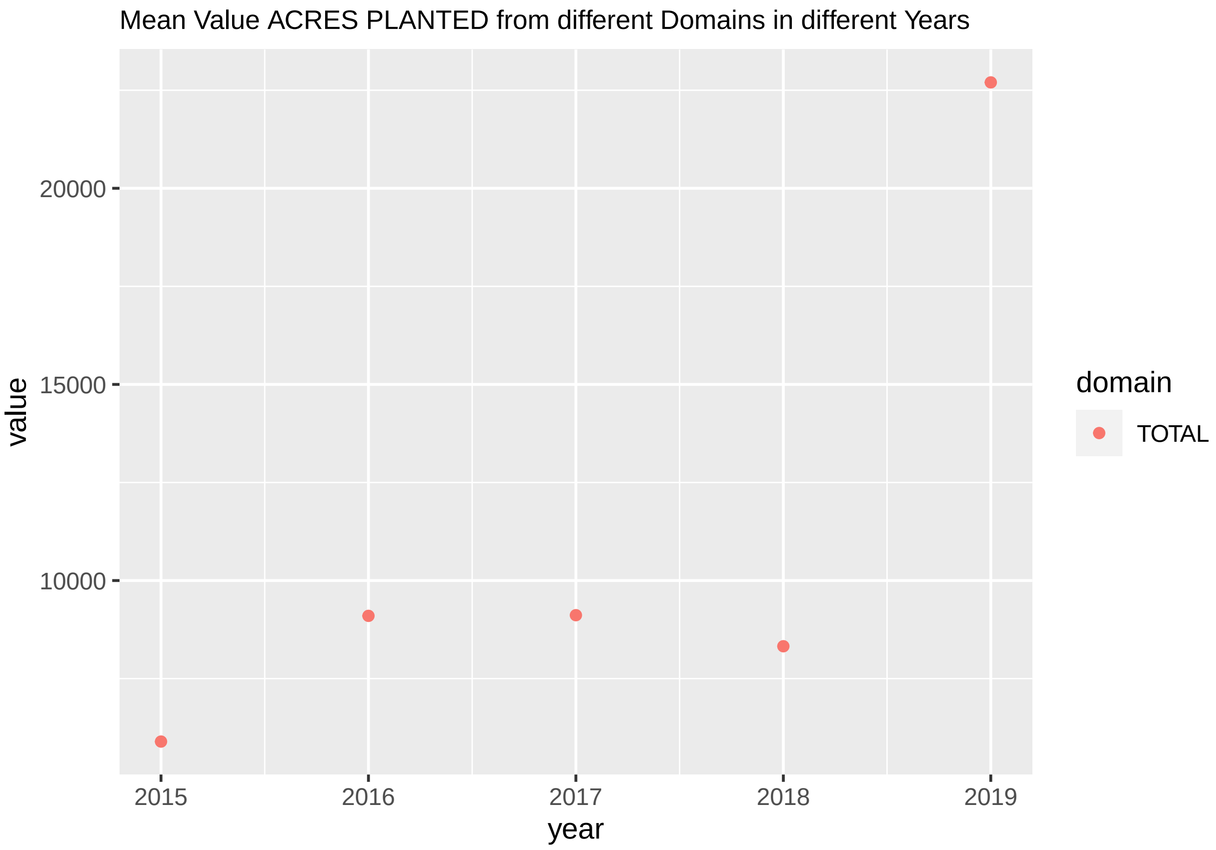
```
plot_list2[[1]]
```


Mean Value MEASURED IN $ / CWT from different Domains in different Years
value
140
130
120
110
2015
2016
2017
2018
2019
year
domain
TOTAL

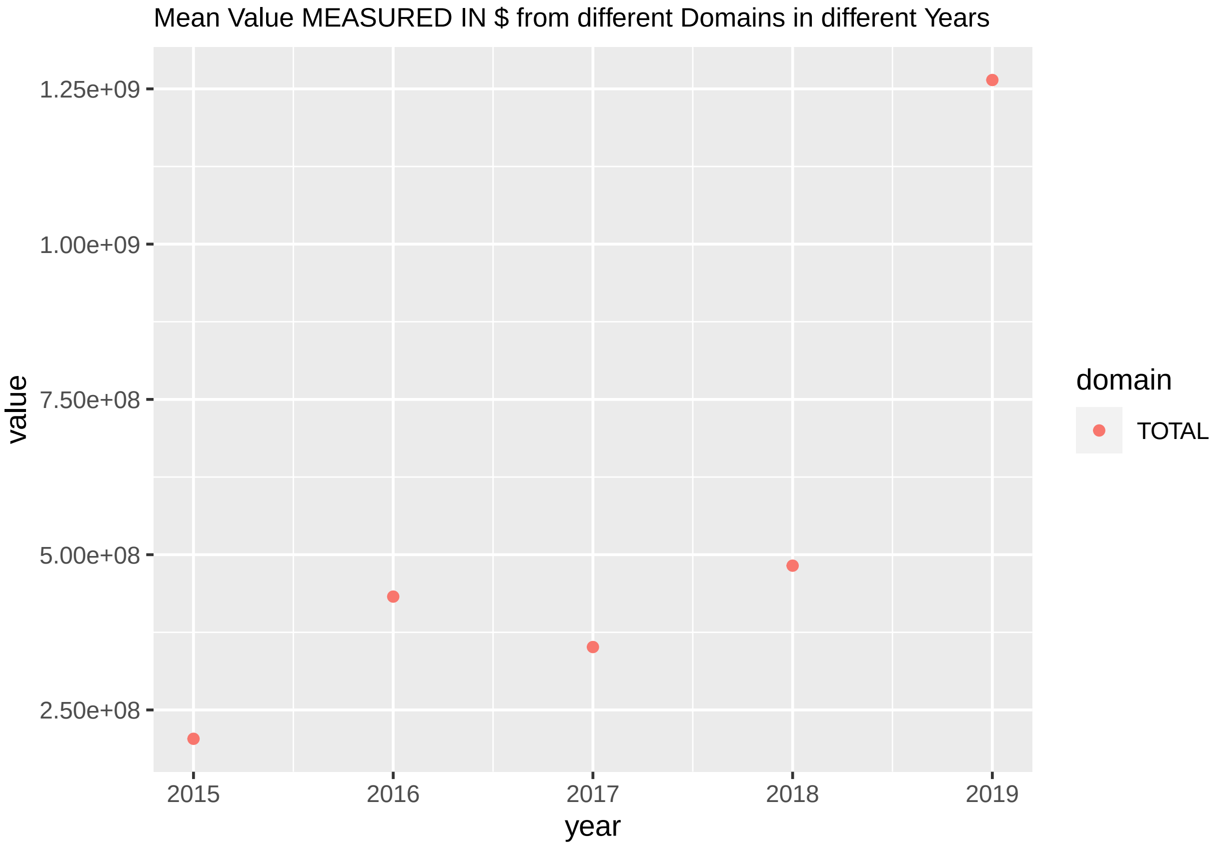
```
plot_list2[[2]]
```


Mean Value ACRES HARVESTED from different Domains in different Years
value
20000
15000
10000
5000
2015
2016
2017
2018
2019
year
domain
● TOTAL

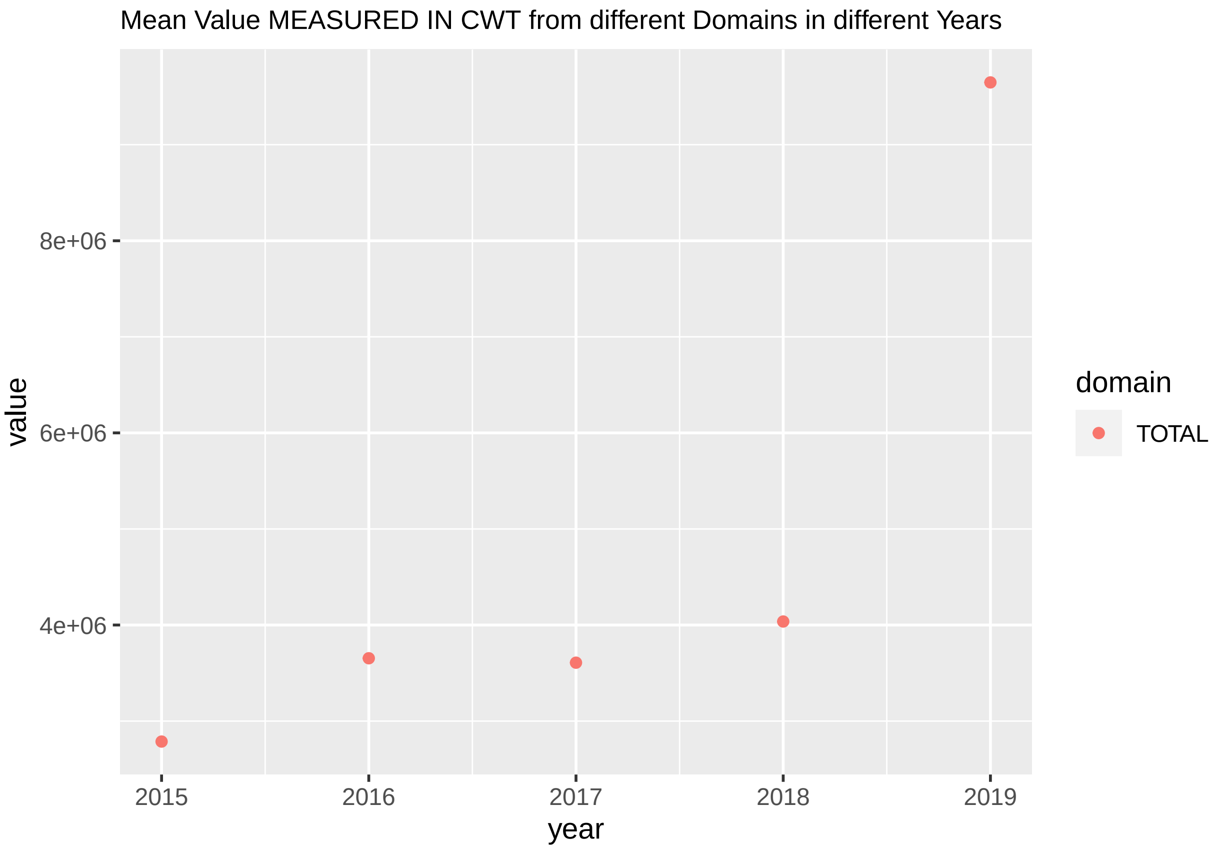
```
plot_list2[[3]]
```


Mean Value ACRES PLANTED from different Domains in different Years
value
20000
15000
10000
2015
2016
2017
2018
2019
year
domain
● TOTAL

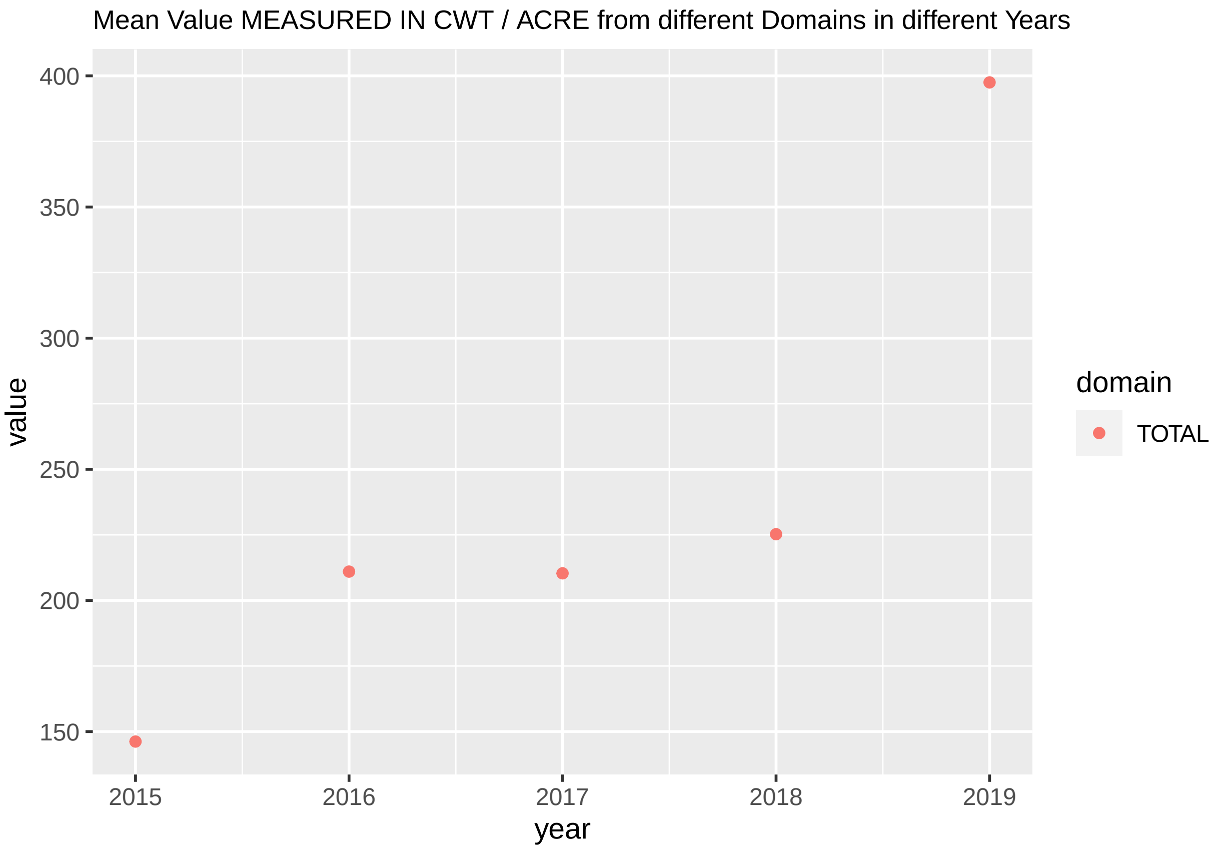
```
plot_list2[[4]]
```


Mean Value MEASURED IN $ from different Domains in different Years
1.25e+09
1.00e+09
7.50e+08
5.00e+08
2.50e+08
value
2015
2016
2017
2018
2019
year
domain
● TOTAL

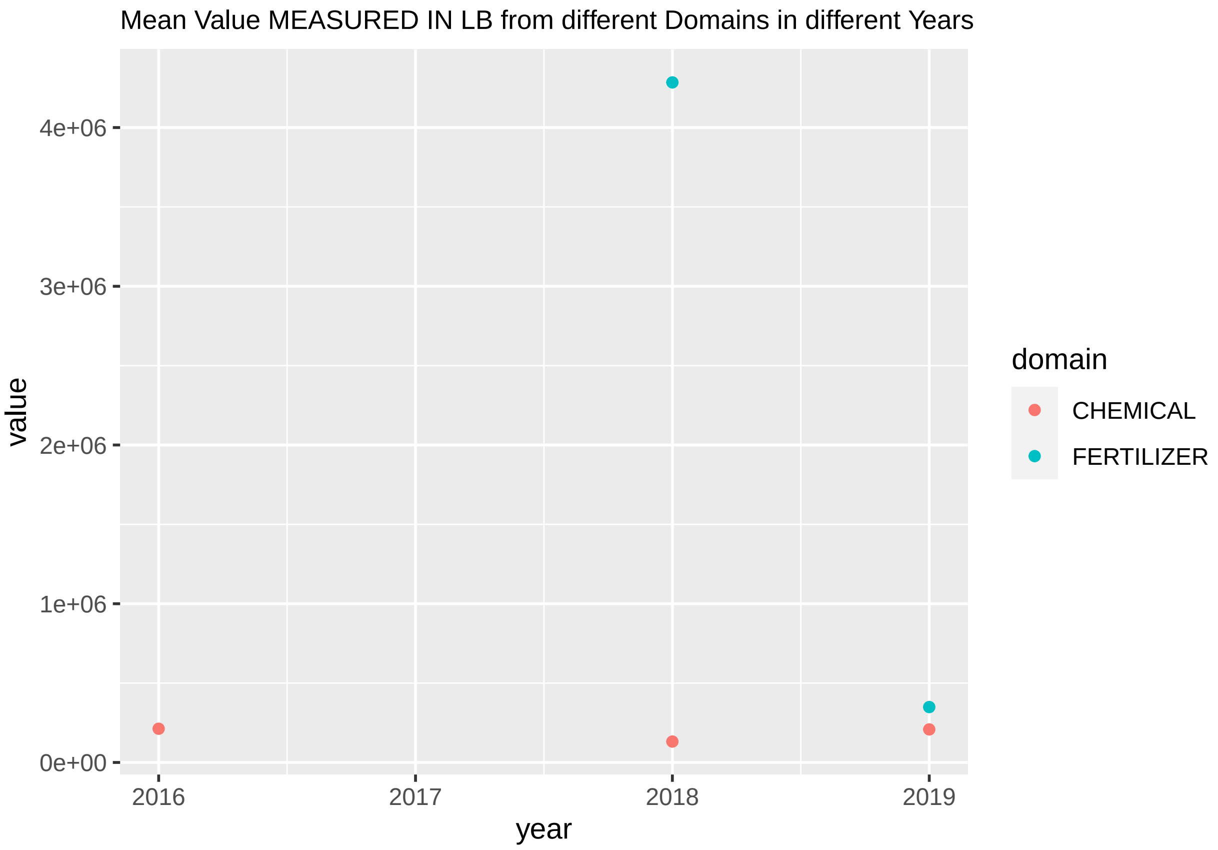
```
plot_list2[[5]]
```


Mean Value MEASURED IN CWT from different Domains in different Years
value
8e+06
6e+06
4e+06
2015
2016
2017
2018
2019
year
domain
● TOTAL

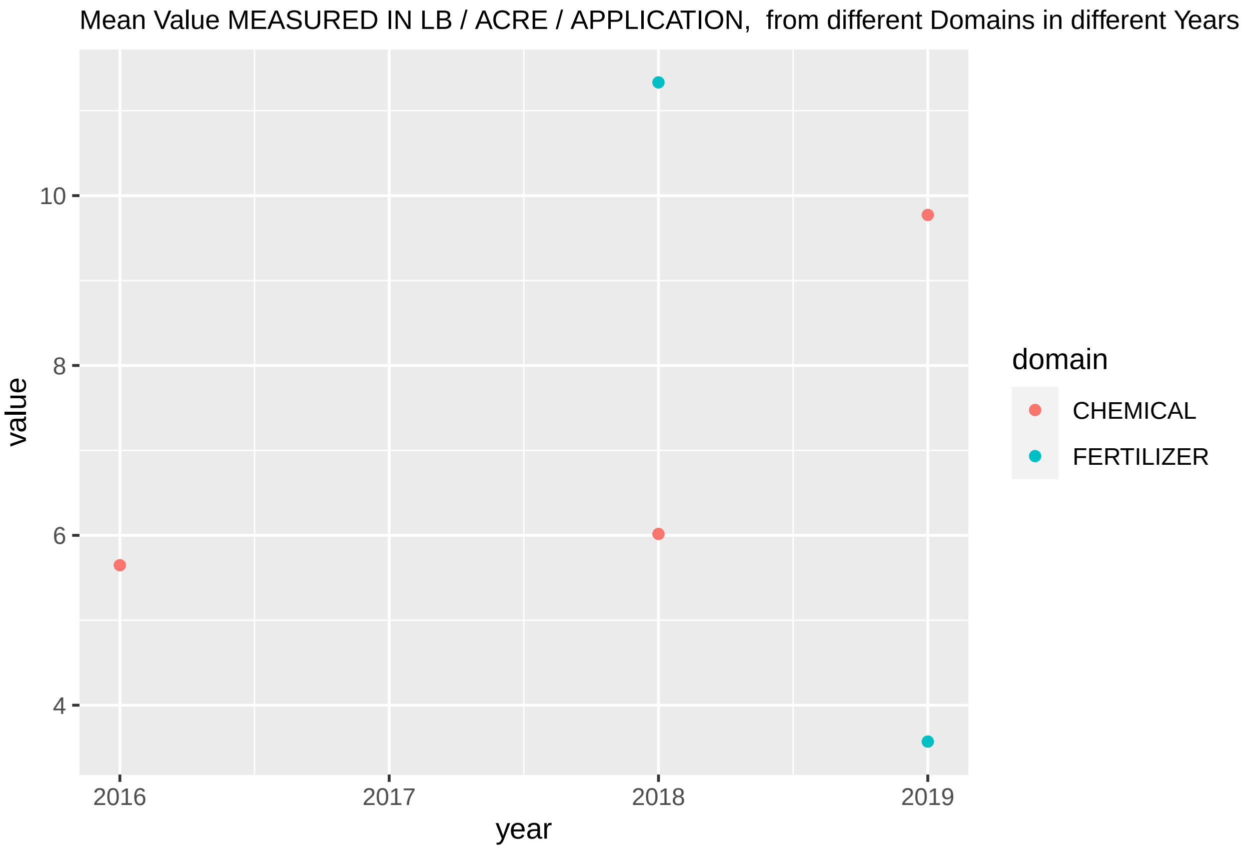
```
plot_list2[[6]]
```


Mean Value MEASURED IN CWT / ACRE from different Domains in different Years
400
350
300
250
200
150
value
2015
2016
2017
2018
2019
year
domain
● TOTAL

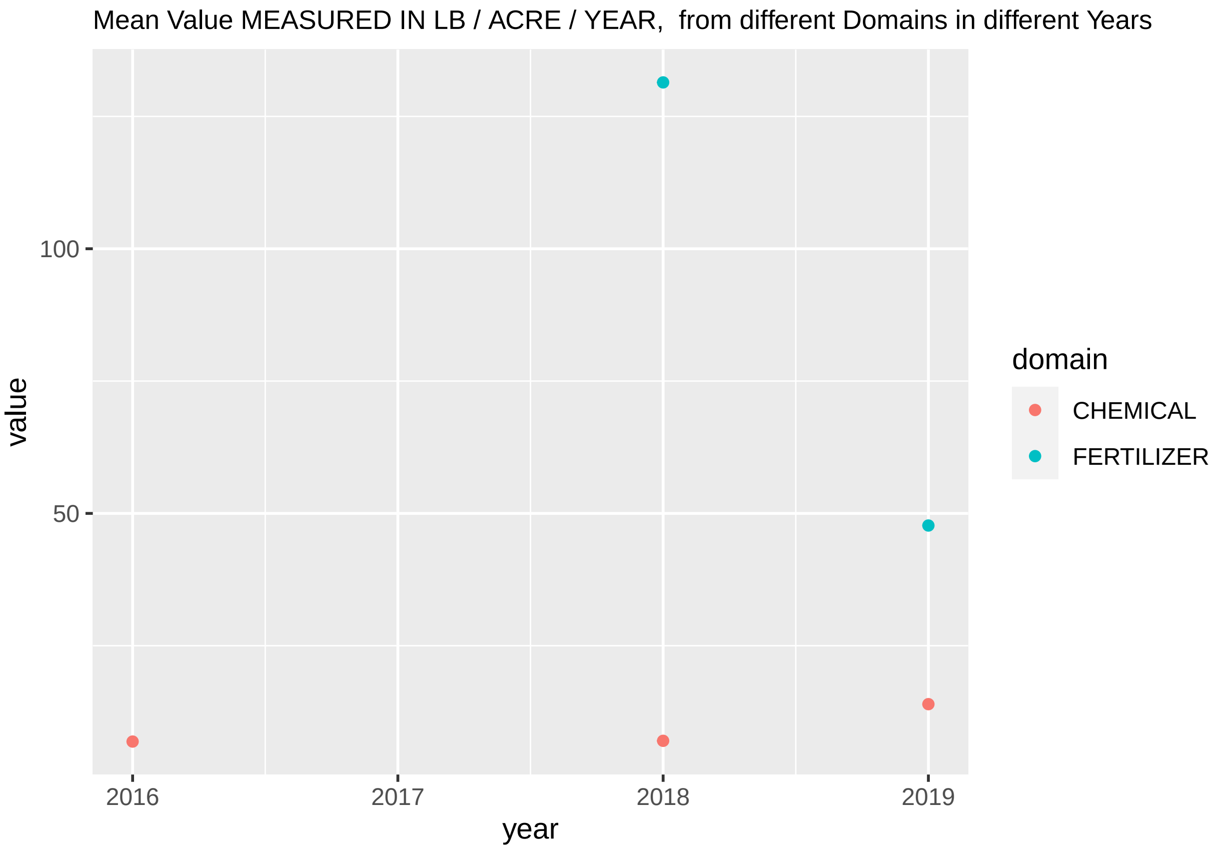
```
plot_list2[[7]]
```


Mean Value MEASURED IN LB from different Domains in different Years
value
4e+06
3e+06
2e+06
1e+06
0e+00
2016
2017
2018
2019
year
domain
● CHEMICAL
● FERTILIZER

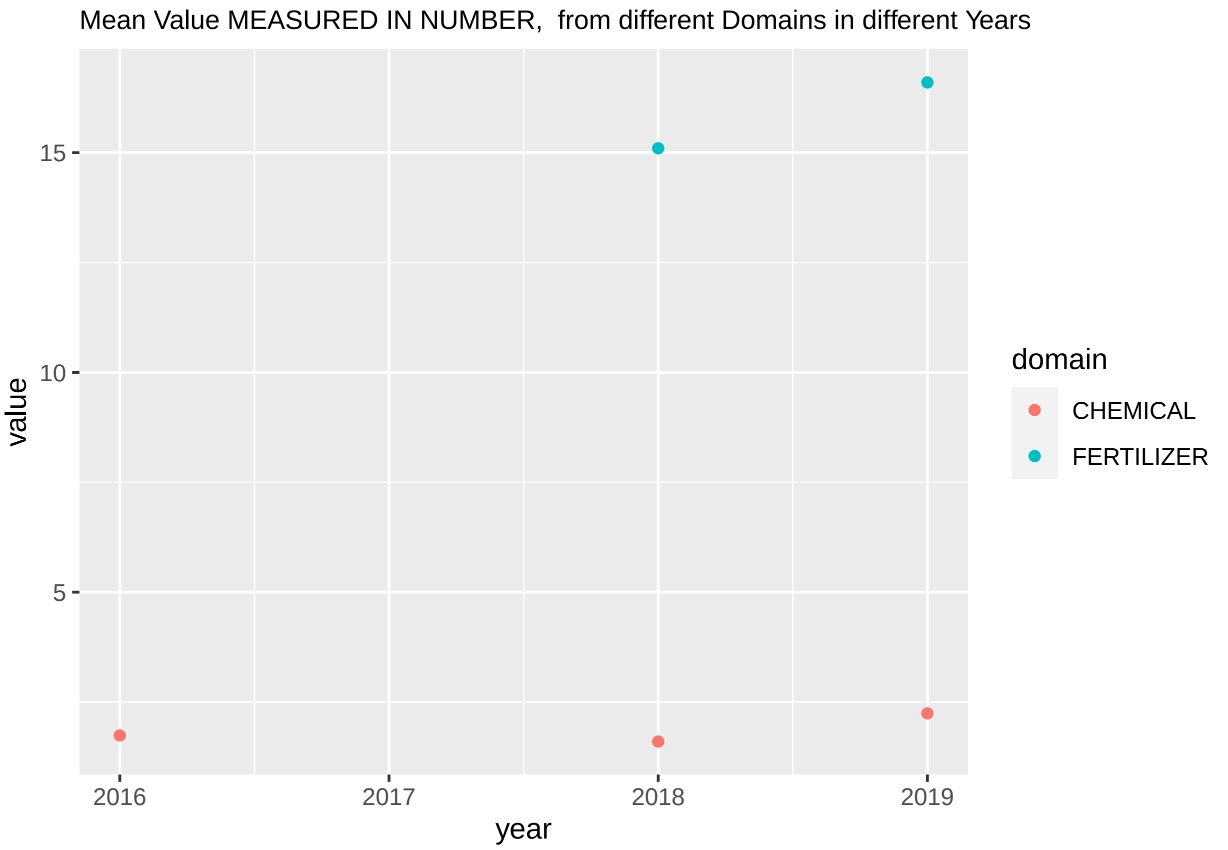
```
plot_list2[[8]]
```


Mean Value MEASURED IN LB / ACRE / APPLICATION, from different Domains in different Years
value
10
8
6
4
2016
2017
2018
2019
year
domain
CHEMICAL
FERTILIZER

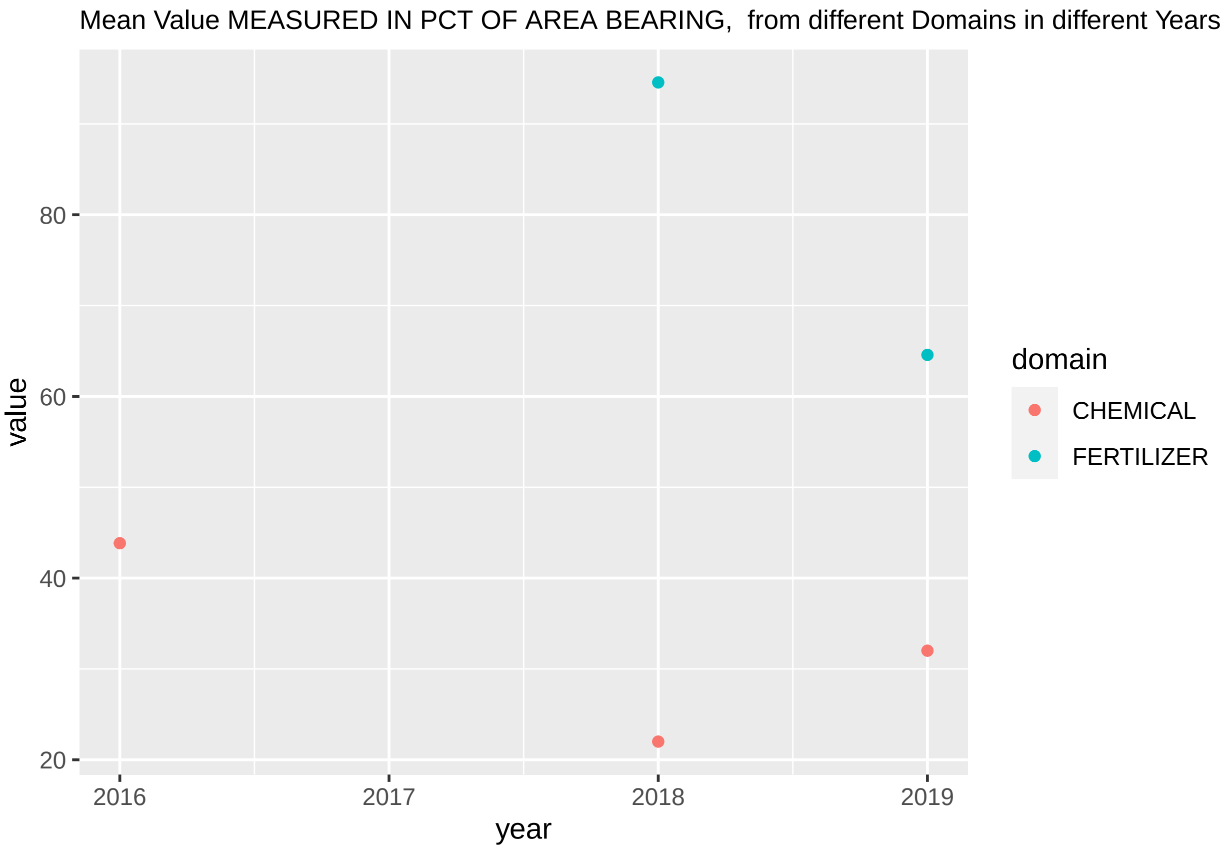
```
plot_list2[[9]]
```



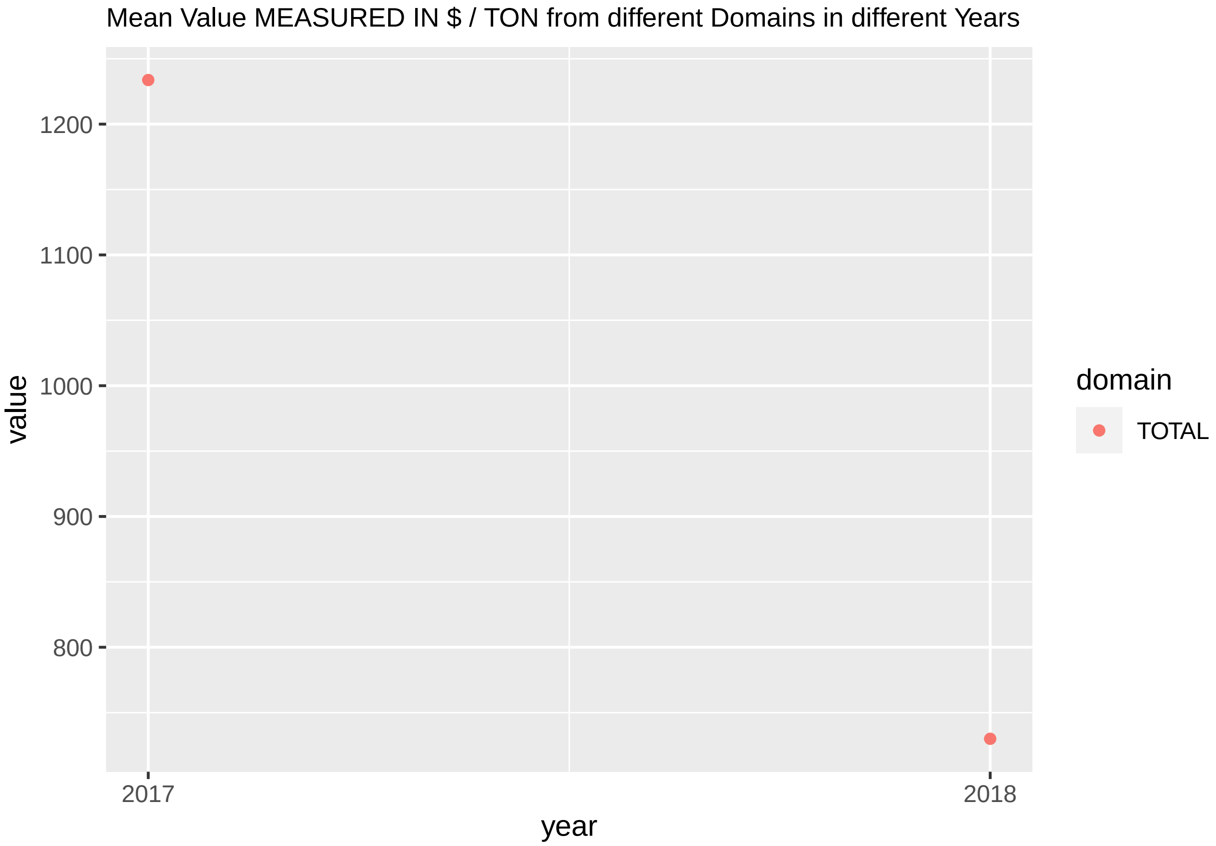
```
plot_list2[[10]]
```



```
plot_list2[[11]]
```

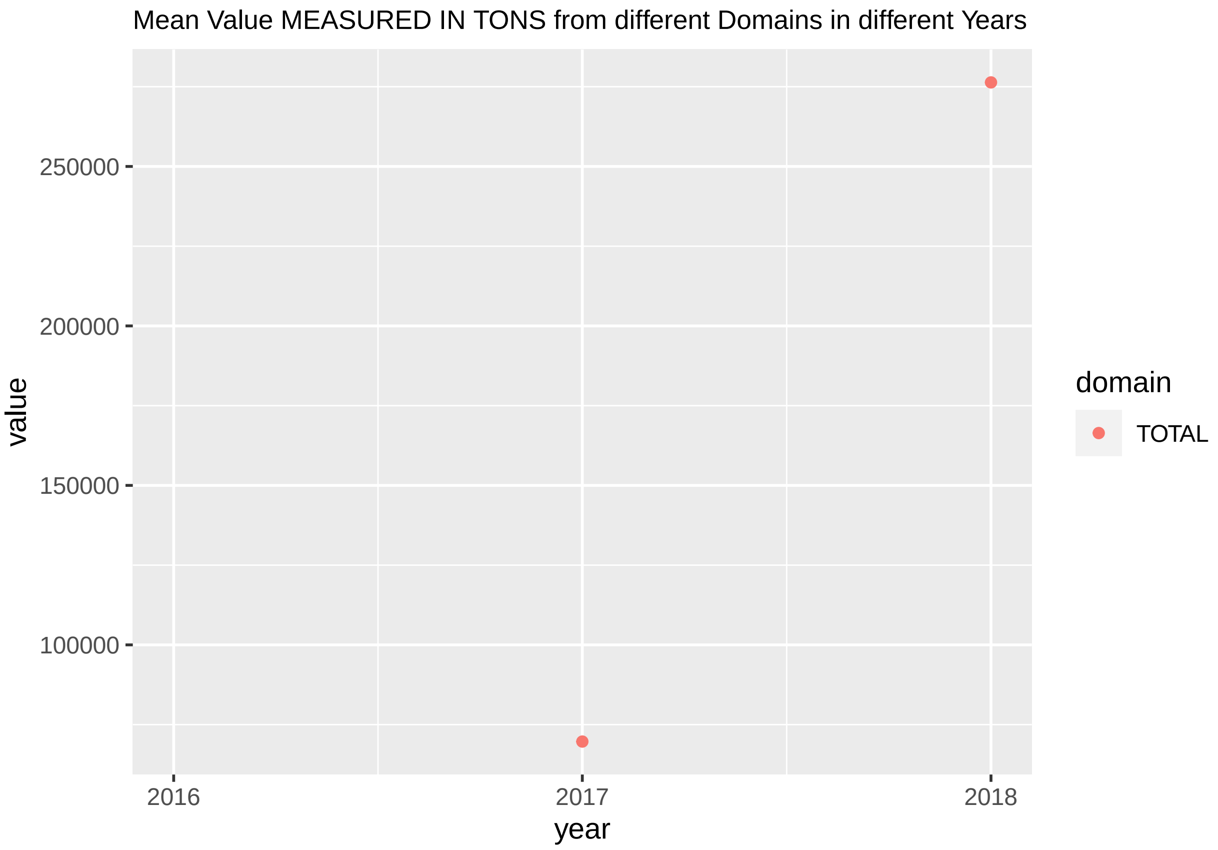


```
plot_list2[[12]]
```



```
plot_list2[[13]]
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



(3)Results analysis

At first, I find that total is used the most, that means in most situations, producing strawberries need chemicals and fertilizers. Besides, I find that the value in 2019 is biggest in most situations.

5.Conclutions

Through this assignment, I completed data cleaning and organization and EDA, detail are as below.

(1)I have learned how to use package 'stringr' to process characters and strings such as deleting empty columns, separating strings into parts (two ways: using regular expression or loop) and changing types of variables.

(2)I have learned how to use package 'srvyr' to group by and summarize for plotting at last.

6.Reference

(1)R packages Documentation:tidyverse, magrittr, plyr, stringr, srvyr (2)Hadley Wickham & Garrett Grolemund(2017). R for data science