THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

MODULE 1 UNIT 2
Notes Video 1 Part 2 Transcript

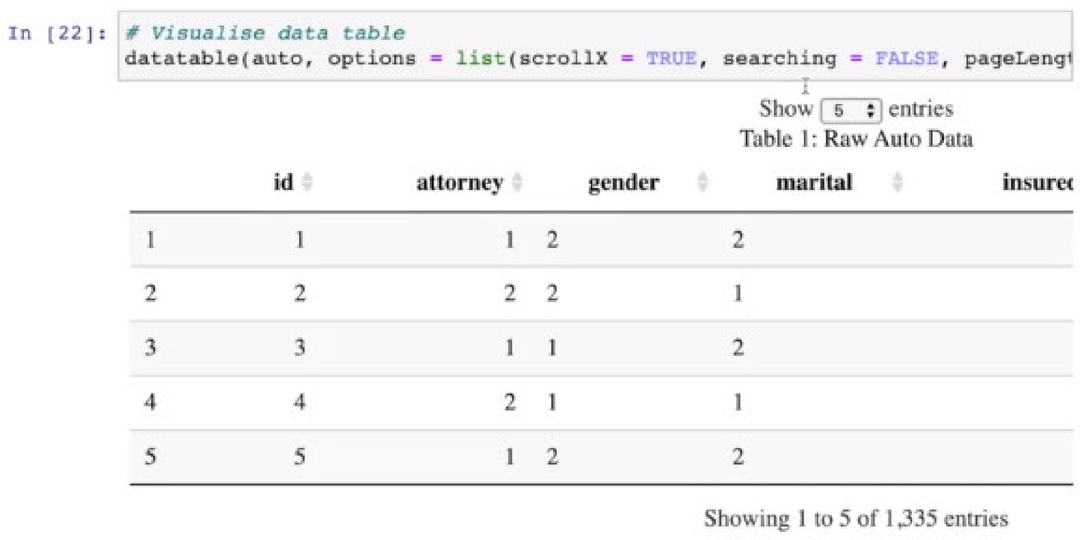
Module 1 Unit 2 Notes Video 1 Part 2 Transcript

Creating plots from data in R

KOSTAS KALOGEROPOULOS: Now that you have learned the basics of creating variables in R, how to import packages and data, and cleaning the imported data, Part 2 of this video explores the process of visualising and transforming data, as well as the process of generating plots. You will remember from Part 1 of this video that it is important to first understand the data before transforming the data. To explore the process of visualising data, scroll down to Section E of this IDE activity.

Section E: Visualising data

KALOGEROPOULOS: Once all the missing values have been removed, the data can now be visualised. In the first code cell in this section, you can see the visualisation of the data in an easy-to-use table – the data table – by making use of the "data table" function. This function segments the table into an easy-to-explore table, which can be useful when the data set is very large.



```
In [22]: # Visualise data table
         datatable(auto, options = list(scrollX = TRUE, searching = FALSE, pageLengt
```

Show 5 entries

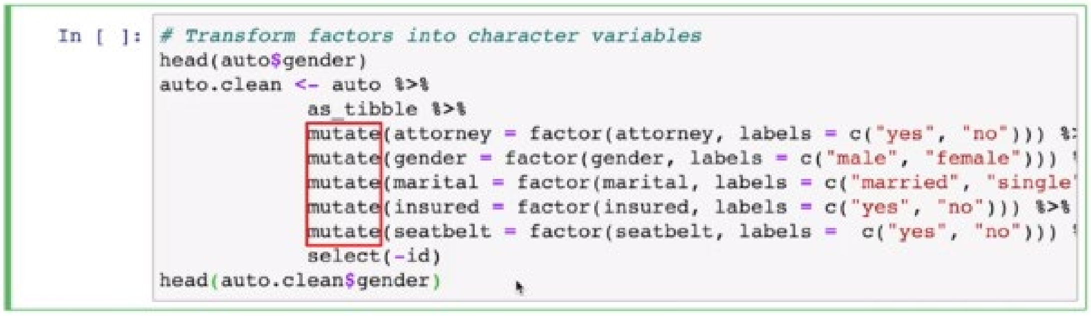
Table 1: Raw Auto Data

	id	attorney	gender	marital	insurec
1	1	1	2	2	
2	2	2	2	1	
3	3	1	1	2	
4	4	2	1	1	
5	5	1	2	2	

Showing 1 to 5 of 1,335 entries

Section F: Transforming data

KALOGEROPOULOS: In this section, the aim is to code the variables that are categorical into easy-to-understand names. For example, if you look at the gender variable, it is coded as either a 1 or 2 to distinguish between male and female individuals. However, this will become hard to track, so the next lines of code transform these factor values into text values. To do this, the dplyr package function "mutate" is used. You can see that it is possible to stack multiple operations together. At the end of this stack, the ID variable is dropped by making use of the "select" function. Instead of choosing to select ID, −ID is chosen to tell R to drop this column.



```
In [ ]: # Transform factors into character variables
        head(auto$gender)
        auto.clean <- auto %>%
                    as_tibble %>%
                    mutate(attorney = factor(attorney, labels = c("yes", "no"))) %:
                    mutate(gender = factor(gender, labels = c("male", "female"))) '
                    mutate(marital = factor(marital, labels = c("married", "single'
                    mutate(insured = factor(insured, labels = c("yes", "no"))) %>%
                    mutate(seatbelt = factor(seatbelt, labels =  c("yes", "no"))) '
                    select(-id)
        head(auto.clean$gender)
```

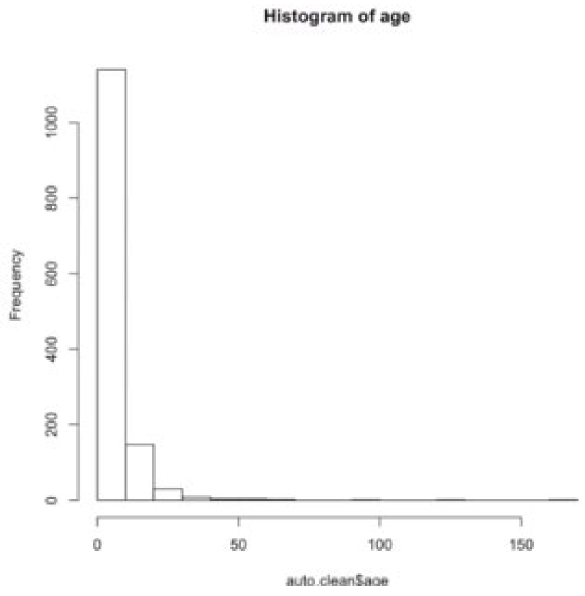
In some cases, you may want to only choose data that meets certain criteria, for example, considering only married men. The dplyr package has another useful function called "filter". The syntax is similar to "select", but instead of choosing specific columns, this function allows for the filtering of columns when a condition meets certain criteria. After the various columns have been selected, a data table is called. Note that when the "data table" function is used, each option is placed on a new line. This is to prevent having extremely long lines of code that reduces readability.

Section G: Plots

KALOGEROPOULOS: Now that the data is clean, it can be explored. A very simple but highly useful function is to generate the summary statistics of the data. For categorical variables, this provides the count of each category. For numerical variables, it shows the maximum, minimum, mean, median, and quartile data.

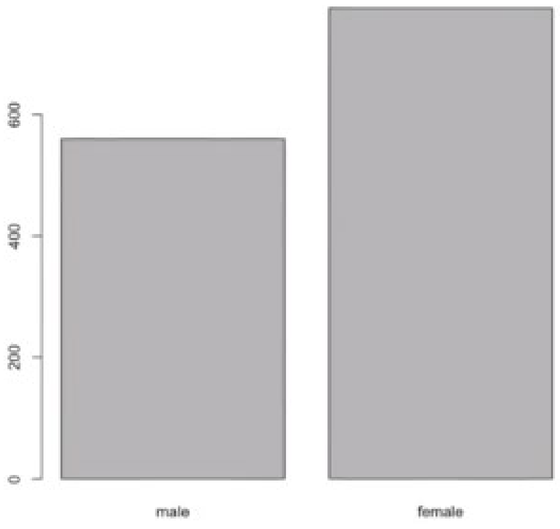
Next, the data may be graphically analysed. To look at a single continuous variable, a histogram can be useful to see how the values fall. In the second code cell of this section, a histogram of age is generated.

```
In [26]: # Create histogram
         hist(auto.clean$age,main="Histogram of age")
```



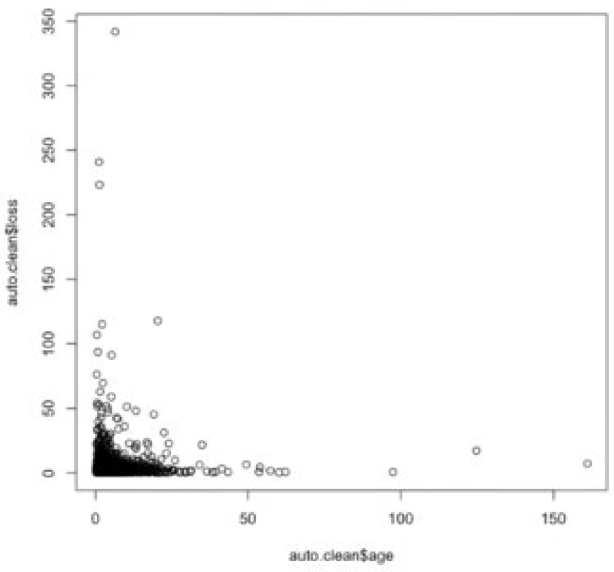
For a categorical variable, it may be necessary to see the count of each category. As you can see from the third code cell in this section, a bar chart can be used fairly easily to compare these categorical variables to one another.

```
In [27]: # Create bar chart
         plot(auto.clean$gender)
```



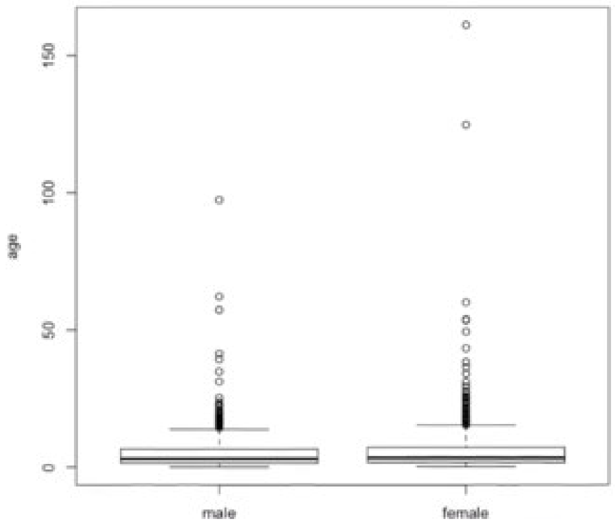
Another useful chart is a bivariate chart, which can be used to see how two variables are related to one another. For continuous variables, a scatterplot can be used. In this example, a scatterplot is generated for the variables age and loss. This is also easy to call from R. From the scatterplot, it's seen that insurance loss decreases as the individual's age increases.

```
In [28]: # Create scatterplot
         plot(auto.clean$age, auto.clean$loss)
```



Box plots can also be used to show the distribution of a continuous variable across each categorical value. In this example, a box plot is generated of gender in relation to age for both male and female individuals.

```
In [29]: # Create box plot
         boxplot(age ~ gender, data = auto.clean)
```


age
male
female