Tutorial 1

STATS 330

Week beginning March 20, 2017

### Task 1: Create a folder

Open the *My Document* folder on the desktop. Create a folder named *330*. If you are reading this in **RStudio** you probably have done this already.

### Task 2: Load R330 package

We load the package using the library command. Note, if the package is not installed, you will get an error message.

library("R330", warn.conflicts = FALSE, quietly = TRUE)

### Task 3: Set your folder as working directory

If you haven't already set your folder from *Task 1* as your working directory then do this. Remember that you can select the folder through the Files menu in the bottom right window. You will have to write the path in here to make the **Markdown** file work! # {r} # setwd("/tmp") # Change this! #

### Task 4: Download data

Open a browser, and navigate to the course page on CANVAS. The *Rmarkdown* file and the data file can be found in the Tutorials module.

Save the data file as a text file called *vapour.txt* into the folder you created in *Task 1*.

The Petrol Vapour Data consist of 126 observations for 5 variables:

|  |  |  |
| --- | --- | --- |
| Variable | Explanation | Unit |
| t.temp | initial tank temperature | degrees F |
| p.temp | dispensed petrol temperature | degrees F |
| t.vp | initial vapour pressure in tank | psi |
| p.vp | vapour pressure of dispensed petrol | psi |
| hc | emitted dispensed hydrocarbons (response) | g |

### Task 5: Load data into *R*

If you have finished *Task 3*, then the following chunk will do the trick:

vapour.df <- read.table("vapour.txt", header = TRUE, quote = "\"")

In general **RStudio** permits to load data with the Import Dataset button in the Environment on the top right window. However, the **Markdown** file needs the command to be able to compile.

Investigate the content of the file using the summary command:

summary(vapour.df)

## t.temp p.temp t.vp p.vp   
## Min. :24.00 Min. :33.0 Min. :2.590 Min. :2.520   
## 1st Qu.:37.00 1st Qu.:50.0 1st Qu.:3.370 1st Qu.:3.430   
## Median :60.00 Median :60.0 Median :4.110 Median :4.110   
## Mean :57.32 Mean :56.5 Mean :4.307 Mean :4.253   
## 3rd Qu.:61.00 3rd Qu.:62.0 3rd Qu.:4.480 3rd Qu.:4.450   
## Max. :93.00 Max. :92.0 Max. :7.600 Max. :7.450   
## hc   
## Min. :16.00   
## 1st Qu.:25.00   
## Median :30.00   
## Mean :30.58   
## 3rd Qu.:34.00   
## Max. :58.00

### Task 6: Transformations

The temp variables are in degrees Fahrenheit. Since we are more familiar with degrees Celcius it seems reasonable to transform. The formula is xC=(xF-32)\*5/9. Creates two new variable converting the lastones and therefore creating 2 new columns

vapour.df <- transform(vapour.df, t.celcius = round((t.temp-32)\*5/9, 2))  
vapour.df <- transform(vapour.df, p.celcius = round((p.temp-32)\*5/9, 2))  
summary(vapour.df)

## t.temp p.temp t.vp p.vp   
## Min. :24.00 Min. :33.0 Min. :2.590 Min. :2.520   
## 1st Qu.:37.00 1st Qu.:50.0 1st Qu.:3.370 1st Qu.:3.430   
## Median :60.00 Median :60.0 Median :4.110 Median :4.110   
## Mean :57.32 Mean :56.5 Mean :4.307 Mean :4.253   
## 3rd Qu.:61.00 3rd Qu.:62.0 3rd Qu.:4.480 3rd Qu.:4.450   
## Max. :93.00 Max. :92.0 Max. :7.600 Max. :7.450   
## hc t.celcius p.celcius   
## Min. :16.00 Min. :-4.44 Min. : 0.56   
## 1st Qu.:25.00 1st Qu.: 2.78 1st Qu.:10.00   
## Median :30.00 Median :15.56 Median :15.56   
## Mean :30.58 Mean :14.07 Mean :13.61   
## 3rd Qu.:34.00 3rd Qu.:16.11 3rd Qu.:16.67   
## Max. :58.00 Max. :33.89 Max. :33.33

### Task 7: Create a factor

Suppose we want to divide the data into two groups: Group 1 with values of hc less than 30 and group 2 containing the remaining points.

We can create a variable group having values 1 and 2 depending on the value of hc in various ways. One way is to create a vector group having values all 2 and then change the values for those observations with hc<30 to 1. The code to this is:

group <- rep(2, nrow(vapour.df))  
group[vapour.df$hc < 30] <- 1  
print(group)

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 2 1 1 2 2 2 2 2 2  
## [36] 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1  
## [71] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 2 2 1 1 1 2 2 2  
## [106] 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1

The variable group is not a factor, but rather a numeric vector. This can be tested using the command typeof. Suppose we want to turn the vector into a factor with levels Group1 and Group2. This is achieved by:

group.factor2 <- factor(group, labels = c("Group1","Group2"))  
print(group.factor2)

## [1] Group1 Group1 Group1 Group1 Group1 Group1 Group1 Group1 Group1 Group1  
## [11] Group1 Group1 Group1 Group1 Group1 Group1 Group1 Group1 Group1 Group1  
## [21] Group2 Group2 Group2 Group2 Group1 Group2 Group2 Group1 Group1 Group2  
## [31] Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2  
## [41] Group2 Group2 Group2 Group2 Group1 Group1 Group1 Group1 Group1 Group2  
## [51] Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2  
## [61] Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group1 Group1 Group1  
## [71] Group1 Group1 Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2  
## [81] Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2  
## [91] Group1 Group1 Group1 Group1 Group1 Group1 Group1 Group2 Group2 Group1  
## [101] Group1 Group1 Group2 Group2 Group2 Group1 Group1 Group1 Group2 Group2  
## [111] Group2 Group1 Group1 Group1 Group1 Group1 Group1 Group1 Group1 Group1  
## [121] Group1 Group1 Group1 Group1 Group1  
## Levels: Group1 Group2

Another way to create a factor from a numeric is to use the command cut using the following command:

group.factor <- cut(vapour.df$hc, breaks = c(0, 30, Inf),  
 labels = c("Group1", "Group2"))

This command is particularly useful if you need to create a factor with multiple levels.

To add this variable to the data frame (although this is not really necessary) you can type

vapour.df <- data.frame(vapour.df, group.factor)

### Task 8: Generate some plots

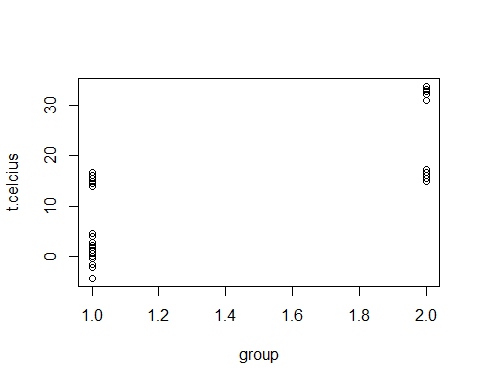
Draw the following plots:

* boxplots and violin plots of t.celcius vs. group.factor. What happens if we plot group against t.celcius?
* A pairs plot of five continuous variables.
* A conditional plot of t.celcius and t.vp conditional on p.celcius
* Using the non-trellis function coplot.
* Using the trellis function xyplot.
* A 3-dimensional scatterplot of t.celcius, t.vp and p.celcius.

#### Notes

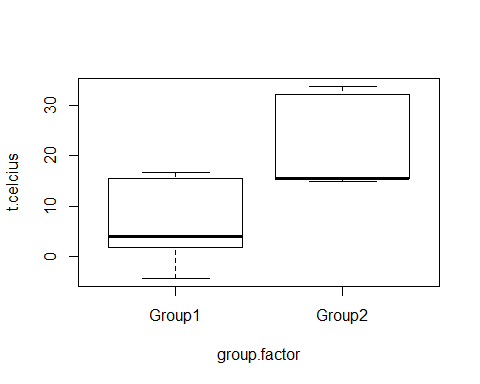
* Create a separate Chunk for each plot.
* You need to load the library scatterplot3d to the function of the same name.
* Explore the optional arguments of the following Sample Code below to change colours, change the symbols, and add axis labels.

plot(t.celcius ~ group, data = vapour.df)



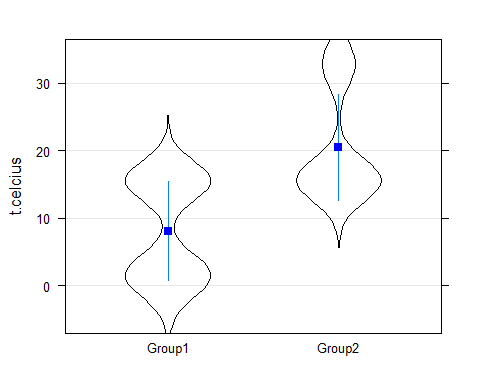
Dotplot

plot(t.celcius ~ group.factor, data = vapour.df)

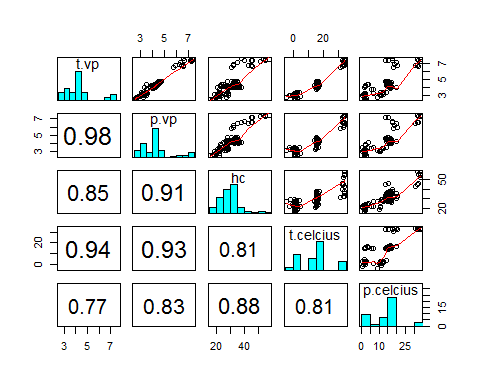


Boxplot

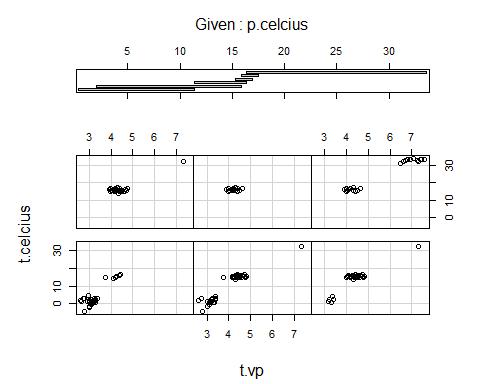
library("violinmplot")  
violinmplot(t.celcius ~ group.factor, data = vapour.df)

 The violin plot, it is similar to the box plot except it show the probability density of the data at different values. it is used to show a variable distribution or sample distribution over accross different categories

pairs20x(vapour.df[, 3:7])

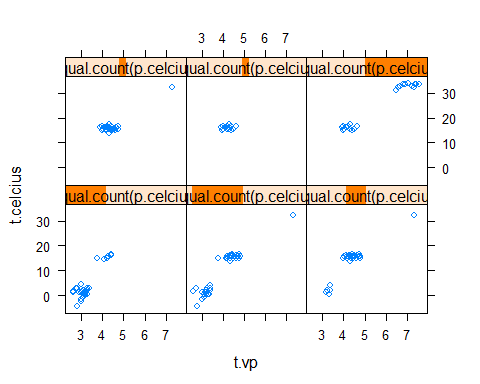
 You can change the 7 to 1 to 6 to show all the different variables in the table

coplot(t.celcius ~ t.vp | p.celcius, data = vapour.df)



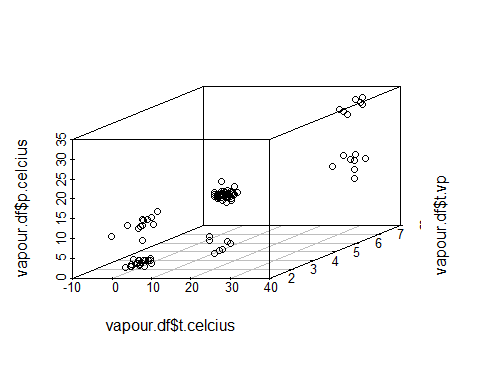
coplot

xyplot(t.celcius ~ t.vp | equal.count(p.celcius), data = vapour.df)



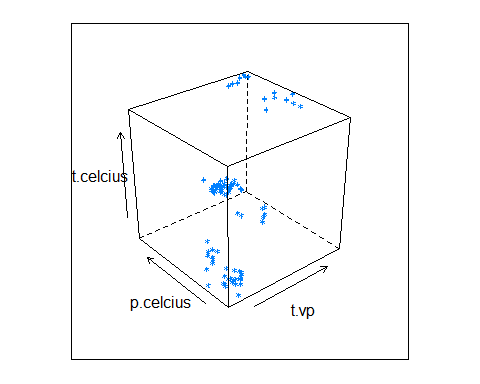
xyplot

library("scatterplot3d")  
scatterplot3d(vapour.df$t.celcius, vapour.df$t.vp,  
 vapour.df$p.celcius)



scatterplot3d

cloud(t.celcius ~ t.vp \* p.celcius, data = vapour.df)



cloud

### Task 9: Create the Tutorial protocol

Hit the Knit button to create a protocol of this session.