MCDM Techniques Using R

Prof. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology – Roorkee

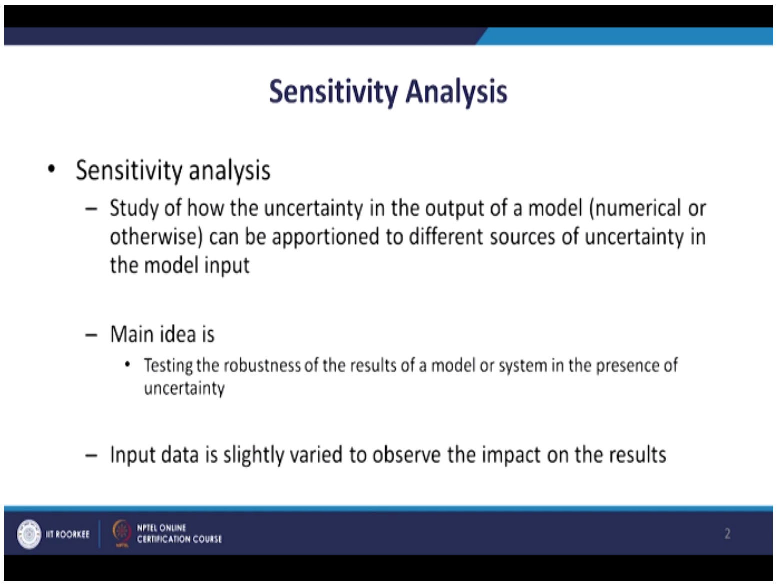
Lecture - 20
Sensitivity Analysis

Welcome to the course MCDM techniques using R. So today we are going to talk about one important aspect of MCDM techniques that is sensitivity analysis. So in this particular course we have covered a number of techniques starting from AHP and up to you know fuzzy AHP, we talked about ELECTRA, we talked about TOPSIS, VIKOR.

So you know all these techniques and the kind of the steps the kind of method that are followed there, there is always a chance that if we change the input you know input variables values if we change the input values then that is going to have an impact on the final results, the ranking that we typically produce. So how do we analyze whether you know a particular you know model is providing robust results or not.

Whether you know slight changes in the input values you know whether they are whether the model results are going to be robust with respect to that, whether the model results are not going to be you know changed just because of slight variation in the input values. So all that discussion comes under you know this topic that is sensitivity analysis. So let us start our discussion on the same. So how do we define sensitivity analysis?

(Refer Slide Time: 01:55)



Sensitivity Analysis

- Sensitivity analysis
  - Study of how the uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input
  - Main idea is
    - Testing the robustness of the results of a model or system in the presence of uncertainty
  - Input data is slightly varied to observe the impact on the results

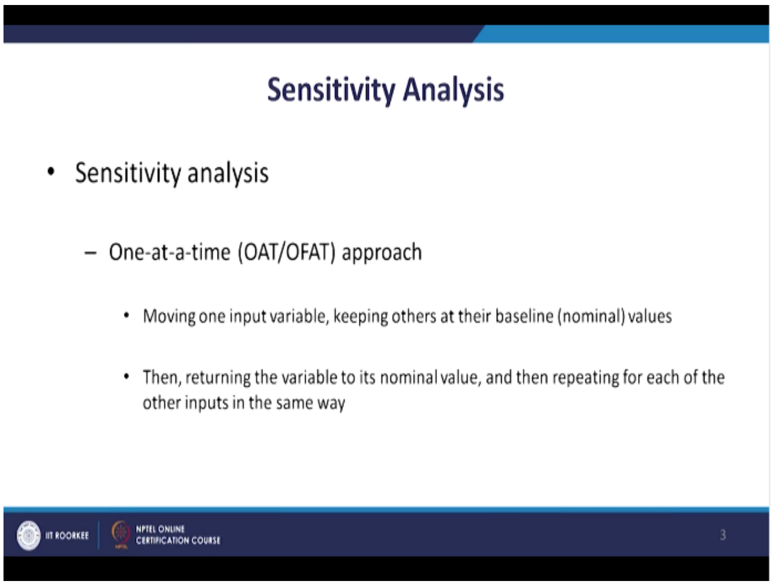
So one definition which is here that study of how the uncertainty in the output of a model whether numerical or otherwise can be apportioned to different sources of uncertainty in the model input. So this uncertainty in the output of you know a particular model that we are talking about is with respect to the uncertainty that is coming from that is originating from model input.

So with respect to the model input you know variables the model input values that changes and the uncertainty that you know can come from there. How the model results? How the output is actually impacted? So that comes under you know domain of sensitivity analysis. So as you can see, main idea being you know so there are many advantages of sensitivity analysis, many things can be done.

Few aspects of sensitivity analysis we have covered in some of the starting lectures. However, the main you know benefit or main idea being testing the robustness of the results of a model or system in the presence of uncertainty? So how robust the results of model are, so how do we check that so that comes under the domain of sensitivity analysis. So as you can see in the third point also that input data is slightly varied to observe the impact on the results.

So this is what we you know tend to do in a sensitivity analysis. Now there are a number of approaches that are available to perform sensitivity analysis. So we are going to talk about you know most popular one you know or rather commonly used one.

(Refer Slide Time: 03:45)



Sensitivity Analysis

- Sensitivity analysis
  - One-at-a-time (OAT/OFAT) approach
    - Moving one input variable, keeping others at their baseline (nominal) values
    - Then, returning the variable to its nominal value, and then repeating for each of the other inputs in the same way

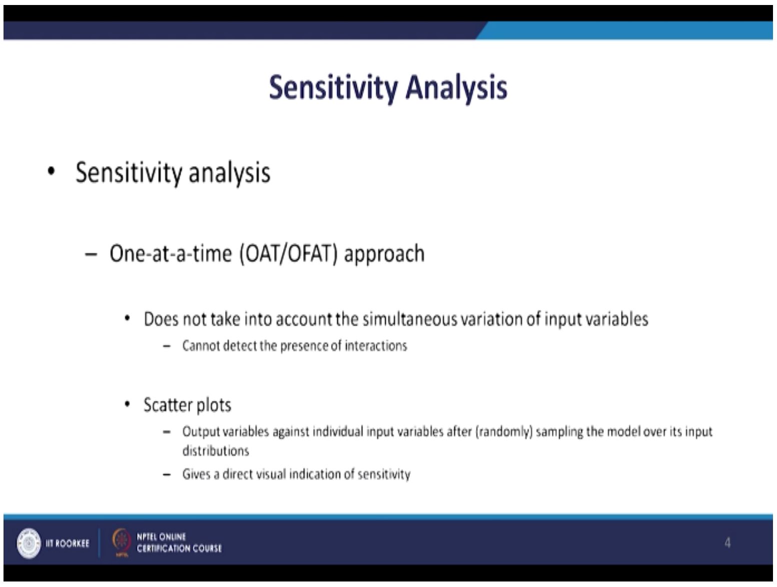
IIT ROORKEE
NPTEL ONLINE CERTIFICATION COURSE
3

This is one-at-a-time approach or OAT/OFAT approach. So here what we actually do in this approach is that we you know first step is moving one input variable keeping others at the baseline or nominal values or we can also say that keeping others constant. So we move one variable, we vary one variable and keep other constant and then you know we can bring you know we can bring back that variable to its nominal value.

And then repeat the same exercise for each of the other input variables. So this is what we do, we pick one parameter one part of variable that we would like to change and we would like to analyze the impact of you know doing certain changes, doing slight changes in a particular you know input variable on the output of the model right and then we you know bring this variable back to its you know normal value right nominal value and the baseline value.

And then we try the same thing with the other input variables. So in that fashion you know for each of those input variables you know we can always you know find out, we can always analyze how the model output is sensitive to changes in those input variables. So sensitivity of the model output with respect to the changes that we do in the input variables that can be analyzed. So let us move forward.

(Refer Slide Time: 05:27)



Sensitivity Analysis

- Sensitivity analysis
  - One-at-a-time (OAT/OFAT) approach
    - Does not take into account the simultaneous variation of input variables
      - Cannot detect the presence of interactions
    - Scatter plots
      - Output variables against individual input variables after (randomly) sampling the model over its input distributions
      - Gives a direct visual indication of sensitivity

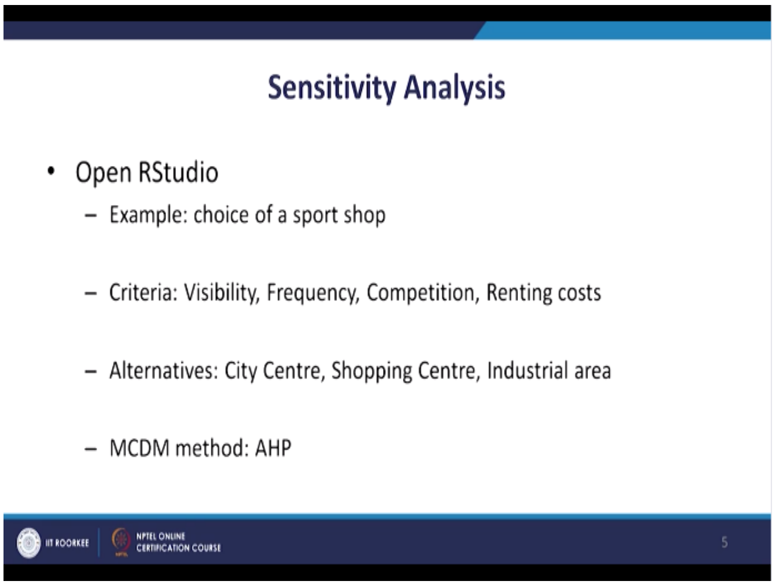
So you know few you know criticism of this approach are also you know have been done. So one of the important criticism is that does not take into account the simultaneous variation of input variables. So a particular model might be based on a number of input variables. However, the approach that we are talking about one-at-a-time approach the more commonly used approach.

It does not take all the variables in one go and then you know test the all you know test the sensitivity of the model with respect to the changes in those you know all of those input variables rather it takes the variables, so the input variables one by one and then you know we analyze the impact on the model output. So therefore what will happen is we cannot detect the presence of you know interaction.

So if some of the input variables are having certain interaction, I know that cannot be you know detected through this suppose because we are taking you know one variable at a time. Now under this approach, we typically you know use scatterplots to actually analyze the results of sensitivity analysis. So once we have done our sensitivity analysis you know modelling then we can you know use scatterplots to analyze the results.

Because I talked about the sensitivity analysis is about you know analyzing you know output of model with respect to changes in the input variables. So output variables against you know individual input variables which are of course after randomly you know sampling the model over it is you know input distribution. So you know after obtaining that, we you know create these scatter plots and what we get in this scatterplots.

And what we get in this scatterplot is a direct visual indication of sensitivity. So when the input particular variable, when a particular input variable is varied then how the model results are actually changing. So we get a visual indication of sensitivity of the model. So now what we are going to do is that we are going to apply this approach, we are going to do exercise in R studio in R environment. So the example that we are taking here is a choice of a sport shop.
(Refer Slide Time: 07:50)



Sensitivity Analysis

- Open RStudio
  - Example: choice of a sport shop
  - Criteria: Visibility, Frequency, Competition, Renting costs
  - Alternatives: City Centre, Shopping Centre, Industrial area
  - MCDM method: AHP

IIT ROORKEE
NPTEL ONLINE CERTIFICATION COURSE
5

So we have 4 criteria for this particular decision problem. So these criterias are visibility, frequency, competition and renting costs. So if someone is looking to decide you know where they should open their sports shop, then you know this could be important criteria because this is like a identification of you know geographical location where you know you would like to open your shop.

So alternatives are you know 3 alternatives are there, City Centre, shopping centre and industrial area. So all these alternatives they have their own pluses and minuses. So which are you know and these alternatives are to be analyzed are to be compared with respect to the criteria you know 4 criteria that we have. Visibility you know visibility of the shop; you know how many customers are going to able to you know visually you know locate the shop.

Frequency number of target customers visiting around that area, competition whether there are any other you know sports shop in the nearby in that same region. So renting costs, what is the renting cost if you want to open a shop, how much rent you will have to pay for that you know that locality.

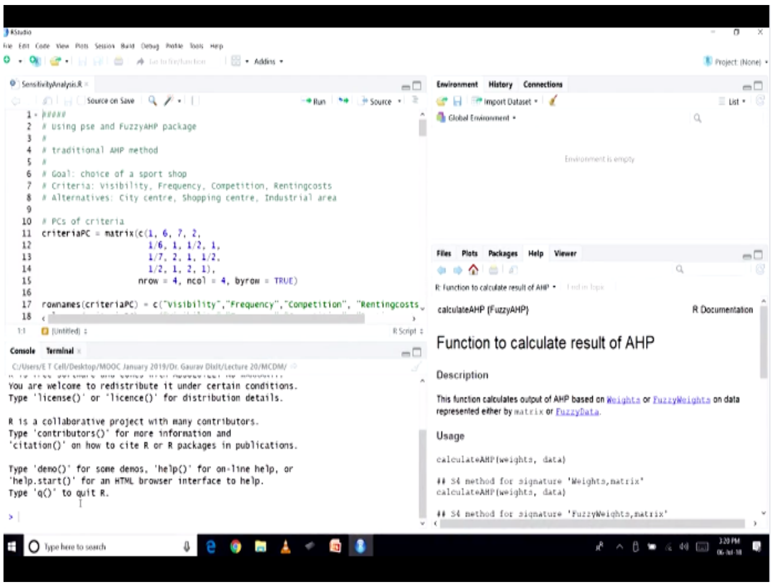
So depending on these 4 criteria, we are going to decide on which is going to be, which is the best alternative among these 3, City Centre, shopping centre, industrial area. Now City Centre is mainly you know this is more of a prime location you know, higher renting cost typically at the center of the city, so possibility of you know targeted customer and the frequency of targeted customer being you know coming to you know the City Centre is higher in comparison to the other areas.

We look at the you know shopping centre you know, it has its own characteristic slightly you know medium level of you know renting cost, competition also a medium level so this is like a medium level location. Then, the industrial area is there, so industrial area typically might you know outside the city in the suburban areas of the cities. Rental cost might be on the lower side, you know competition might be you know on the lower side but frequency and visibility.

So this is you know in that sense these 3 alternatives are going to have their own characteristics but here it is important how do we quantify. Whatever these criteria are with respect to them how we are going to quantify these things? So as we have you know discussed in our AHP, you know we have covered AHP and fuzzy AHP in this particular course.

So we are going to use AHP that traditional AHP for this you know sensitivity analysis. So let us open R studio.

(Refer Slide Time: 10:57)



```
#####
# using pse and FuzzyAHP package
#
# traditional AHP method
#
# Goal: choice of a sport shop
# Criteria: visibility, Frequency, Competition, Rentingcosts
# Alternatives: City centre, Shopping centre, Industrial area

# PCs of criteria
criteriaPC = matrix(c(1, 6, 7, 2,
                      1/6, 1, 1/2, 1,
                      1/7, 2, 1, 1/2,
                      1/2, 1, 2, 1),
                    nrow = 4, ncol = 4, byrow = TRUE)

rownames(criteriaPC) = c("Visibility","Frequency","Competition", "Rentingcosts
```

C:/Users/E T Cell/Desktop/MOOC January 2019/Dr. Gaurav Dixit/Lecture 20/MCDM/

```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

R: Function to calculate result of AHP

calculateAHP (FuzzyAHP) R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

So the package that we are going to use for this you know sensitivity analysis is fuzzy AHP package, something that we have used in the previous lecture. So this package why we are selecting this package because the other two packages that we have covered for AHP method you know the way the model function is defined, we cannot actually change the criteria weights you know if we change criteria weights you know that is not actually directly being taken as a input argument by those model functions.

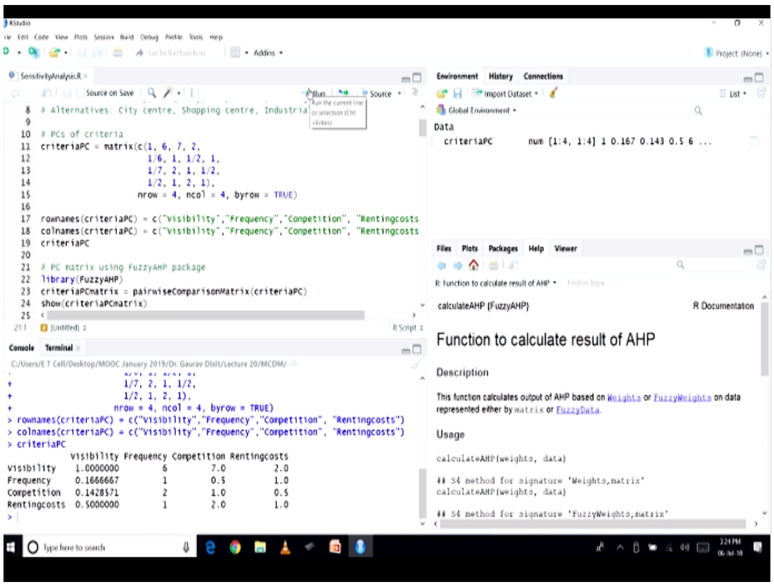
However, the function model function that is available in fuzzy AHP package that takes you know criteria weights as an input variable. So therefore you know any changes that we do in the input variable for the criteria weights that can actually be the impact of those changes can be very easily you know analyzed.

So for this reason because of the suitability of the model function that is there in fuzzy AHP package, we have selected this particular you know package for our sensitivity analysis. Then, we are using another package PSE. So this gives us the functionality that is required for us to perform sensitivity analysis given a model function. So this particular package imposes its own restriction on how the model function should be.

In terms of you know, how it is going to receive you know input variables values and you know in terms of that there are certain restrictions and so the package the function model function that we have in fuzzy AHP will have to do certain coding, will have to do, will have to write you know wrapper for the model function so that we can use the function existing functionality in the PSE package and perform our sensitivity analysis.

So as we talked about the decision problem, goal is choice of a sports shop. We have 4 criteria, visibility, frequency, competition and renting costs. We have 3 alternatives, City Centre, shopping centre and industrial area. So let us start our exercise.

(Refer Slide Time: 13:15)



```
# Alternatives: City centre, Shopping centre, Industria

# PCs of criteria
criteriaPC = matrix(c(1, 6, 7, 2,
                      1/6, 1, 1/2, 1,
                      1/7, 2, 1, 1/2,
                      1/2, 1, 2, 1),
                    nrow = 4, ncol = 4, byrow = TRUE)

rownames(criteriaPC) = c("Visibility","Frequency","Competition", "Rentingcosts
colnames(criteriaPC) = c("Visibility","Frequency","Competition", "Rentingcosts
criteriaPC

# PC matrix using FuzzyAHP package
library(FuzzyAHP)
criteriaPCmatrix = pairwiseComparisonMatrix(criteriaPC)
show(criteriaPCmatrix)
```

```
C:/Users/E T Cell/Desktop/MOOC January 2019/Dr. Gaurav Dixit/Lecture 20/MCDM/
+                 1/7, 2, 1, 1/2,
+                 1/2, 1, 2, 1),
+               nrow = 4, ncol = 4, byrow = TRUE)
> rownames(criteriaPC) = c("Visibility","Frequency","Competition", "Rentingcosts")
> colnames(criteriaPC) = c("Visibility","Frequency","Competition", "Rentingcosts")
> criteriaPC
             Visibility Frequency Competition Rentingcosts
Visibility    1.0000000         6         7.0          2.0
Frequency     0.1666667         1         0.5          1.0
Competition   0.1428571         2         1.0          0.5
Rentingcosts  0.5000000         1         2.0          1.0
>
```

```
Data
  criteriaPC      num [1:4, 1:4] 1 0.167 0.143 0.5 6 ...
```

R: Function to calculate result of AHP

calculateAHP {FuzzyAHP} R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

Usage

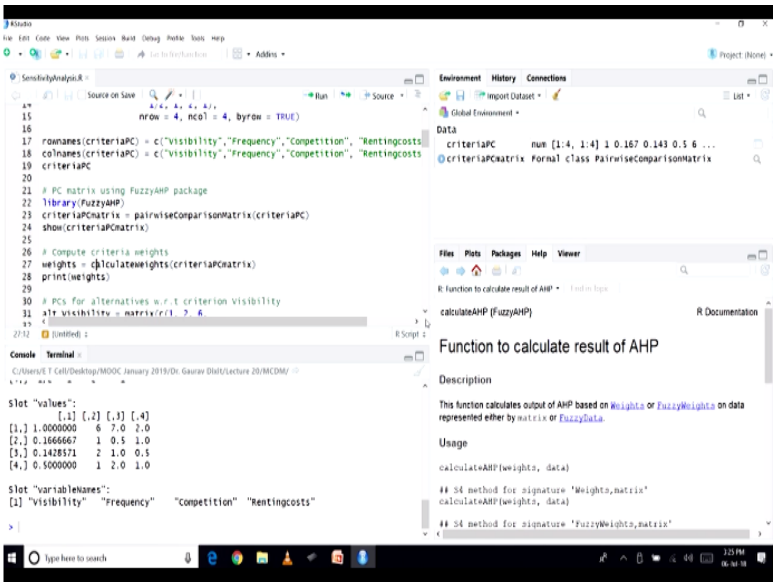
```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

So first we have pairwise comparisons of criteria. So here we have 4 you know criteria so the matrix that we are going to require for pairwise comparison is 4 x 4. So certain values as you can see are there in the first line of code here and these values are going to be used for you know or being used as pairwise comparisons of criteria. So the same function that we have been using previous lectures as well as matrix function to you know to generate the comparison matrix for criteria.

So we are using the first argument is you know using the combined function C. So all the values that we require all 16 values that we require have been combined using this function and then number of rows, number of columns 4 and 4. So let us run this code and you can see in the environment section we have generated criteria PC. This matrix is there. Now we are going to you know give appropriate names for this matrix.

So row names the names of criteria then column names name of the criteria and this is how our as you can see in the console output this is how our criteria matrix comparison matrix is going to look like. So we can see here.

(Refer Slide Time: 14:44)



```
                   nrow = 4, ncol = 4, byrow = TRUE)

rownames(criteriaPC) = c("Visibility","Frequency","Competition", "Rentingcosts
colnames(criteriaPC) = c("Visibility","Frequency","Competition", "Rentingcosts
criteriaPC

# PC matrix using FuzzyAHP package
library(FuzzyAHP)
criteriaPCmatrix = pairwiseComparisonMatrix(criteriaPC)
show(criteriaPCmatrix)

# Compute criteria weights
weights = calculateWeights(criteriaPCmatrix)
print(weights)

# PCs for alternatives w.r.t criterion Visibility
alt_visibility = matrix(c(1, 2, 6,
```

```
Slot "values":
          [,1] [,2] [,3] [,4]
[1,] 1.0000000    6  7.0  2.0
[2,] 0.1666667    1  0.5  1.0
[3,] 0.1428571    2  1.0  0.5
[4,] 0.5000000    1  2.0  1.0

Slot "variableNames":
[1] "Visibility"   "Frequency"    "Competition"  "Rentingcosts"

>
```

```
Data
  criteriaPC        num [1:4, 1:4] 1 0.167 0.143 0.5 6 ...
  criteriaPCmatrix  Formal class PairwiseComparisonMatrix
```

calculateAHP (FuzzyAHP) R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

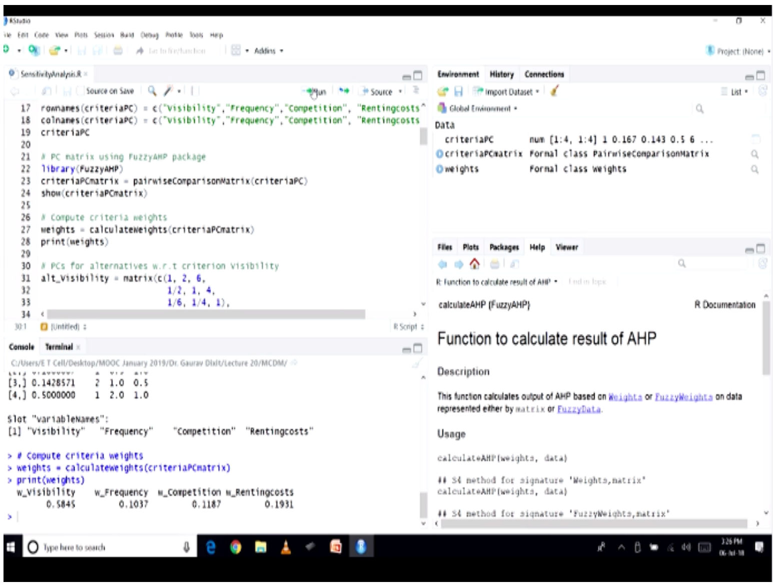
Now what we are going to use is will have to because the fuzzy package that we are using that we are going to use is actually having its own you know class of objects and therefore though we have the comparison matrix we need to convert this comparison matrix into a format that can be understood by fuzzy AHP package function. So therefore what we are going to do now is will load this library fuzzy AHP and do R environment.

So this is loaded. Now we are going to use this pairwise comparison matrix function available in this package as we noted in the previous lecture. So this is going to create us you know comparison matrix for criteria which can be used for other function in this particular fuzzy AHP package. So let us run this code. Let us print this matrix to you can see just like in the previous lecture where we used the fuzzy AHP package, we can see this is an object of class pairwise comparison matrix.

And you can see their values in the character format or displayed here in the first slot, the second slot, values in the numeric formats are displayed here as you can see the same thing we did in the previous lecture and the variable names are also given here in the last slot. So this is slightly different you know package requirements and the class of objects that are being used.

So they have created their own class of objects so therefore we need to use their functions to change you know to convert you know variables in the appropriate format.

(Refer Slide Time: 16:28)



```
rownames(criteriaPC) = c("Visibility","Frequency","Competition", "Rentingcosts"
colnames(criteriaPC) = c("Visibility","Frequency","Competition", "Rentingcosts
criteriaPC

# PC matrix using FuzzyAHP package
library(FuzzyAHP)
criteriaPCmatrix = pairwiseComparisonMatrix(criteriaPC)
show(criteriaPCmatrix)

# Compute criteria weights
weights = calculateWeights(criteriaPCmatrix)
print(weights)

# PCs for alternatives w.r.t criterion Visibility
alt_Visibility = matrix(c(1, 2, 6,
                          1/2, 1, 4,
                          1/6, 1/4, 1),
```

```
[3,] 0.1428571    2  1.0  0.5
[4,] 0.5000000    1  2.0  1.0

Slot "variableNames":
[1] "Visibility"   "Frequency"    "Competition"  "Rentingcosts"

> # Compute criteria weights
> weights = calculateWeights(criteriaPCmatrix)
> print(weights)
  w_Visibility    w_Frequency  w_Competition w_Rentingcosts
        0.5845         0.1037         0.1187         0.1931
> 
```

```
Data
  criteriaPC        num [1:4, 1:4] 1 0.167 0.143 0.5 6 ...
  criteriaPCmatrix  Formal class PairwiseComparisonMatrix
  weights           Formal class Weights
```

calculateAHP (FuzzyAHP) R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

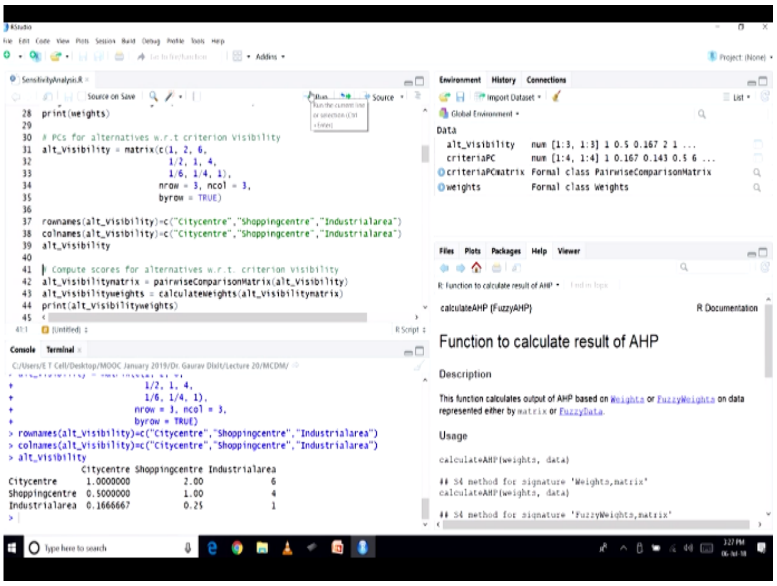
Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

Now what we are going to do is will compute criteria weights. So again we are going to use this calculate weights of function that is available in this package. So the matrix that we have just you know computed so that is going to be passed on as the argument and will get the weights for criteria. So let us run this code and print the weights.

So you can see you know criteria weights for visibility that comes out to be 0.5845, frequency 0.1037 and then competition 0.1187 and then the renting costs 0.1931, so now we have computed the criteria weights.

(Refer Slide Time: 17:09)



```
print(weights)

# PCs for alternatives w.r.t criterion Visibility
alt_Visibility = matrix(c(1, 2, 6,
                          1/2, 1, 4,
                          1/6, 1/4, 1),
                        nrow = 3, ncol = 3,
                        byrow = TRUE)

rownames(alt_Visibility)=c("Citycentre","Shoppingcentre","Industrialarea")
colnames(alt_Visibility)=c("Citycentre","Shoppingcentre","Industrialarea")
alt_Visibility

# Compute scores for alternatives w.r.t. criterion visibility
alt_Visibilitymatrix = pairwiseComparisonMatrix(alt_Visibility)
alt_Visibilityweights = calculateWeights(alt_Visibilitymatrix)
print(alt_Visibilityweights)
```

```
C:/Users/E T Cell/Desktop/MOOC January 2019/Dr. Gaurav Dixit/Lecture 20/MCDM/
+                         1/2, 1, 4,
+                         1/6, 1/4, 1),
+                       nrow = 3, ncol = 3,
+                       byrow = TRUE)
> rownames(alt_Visibility)=c("Citycentre","Shoppingcentre","Industrialarea")
> colnames(alt_Visibility)=c("Citycentre","Shoppingcentre","Industrialarea")
> alt_Visibility
               Citycentre Shoppingcentre Industrialarea
Citycentre      1.0000000           2.00              6
Shoppingcentre  0.5000000           1.00              4
Industrialarea  0.1666667           0.25              1
>
```

```
Data
  alt_Visibility    num [1:3, 1:3] 1 0.5 0.167 2 1 ...
  criteriaPC        num [1:4, 1:4] 1 0.167 0.143 0.5 6 ...
  criteriaPCmatrix  Formal class PairwiseComparisonMatrix
  weights           Formal class Weights
```

calculateAHP {FuzzyAHP} R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

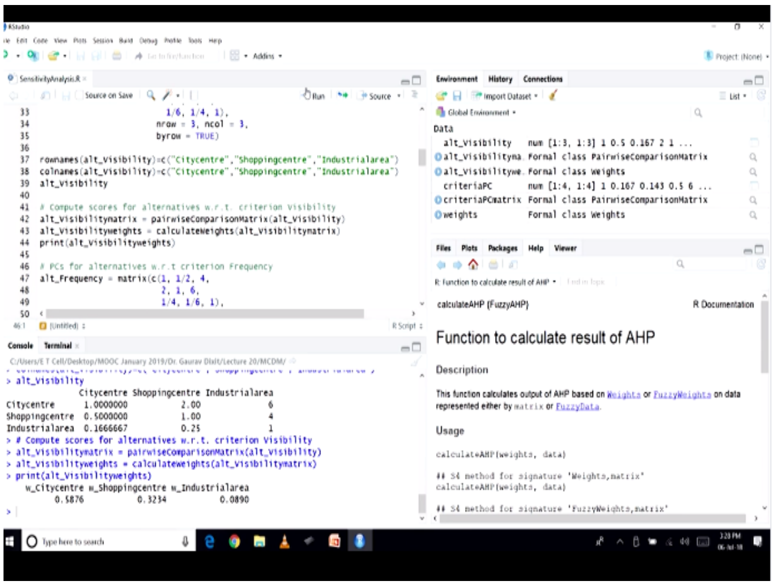
Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

Now let us move forward to pairwise comparisons of alternatives with respect to criterion visibility, similarly will do for other criterion. So here again you can see we are using matrix function and the combined function C function so that we can initialize these scores you know pairwise comparison scores for alternatives with respect to criterion visibility. So let us run this code and will get you know matrix.

Let us name these row names and column names of this matrix and our matrix is going to look like this, you can see here. City Centre, shopping centre and industrial area, row name and column names and you can see the values.

(Refer Slide Time: 18:03)



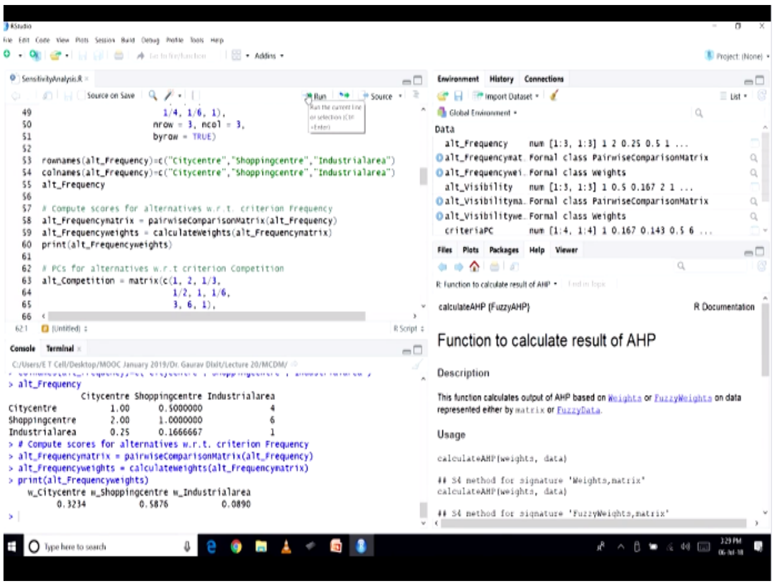
```
                    1/6, 1/4, 1),
                  nrow = 3, ncol = 3,
                  byrow = TRUE)

rownames(alt_Visibility)=c("Citycentre","Shoppingcentre","Industrialarea")
colnames(alt_Visibility)=c("Citycentre","Shoppingcentre","Industrialarea")
alt_Visibility

# Compute scores for alternatives w.r.t. criterion Visibility
alt_Visibilitymatrix = pairwiseComparisonMatrix(alt_Visibility)
alt_Visibilityweights = calculateWeights(alt_Visibilitymatrix)
print(alt_Visibilityweights)

# PCs for alternatives w.r.t criterion Frequency
alt_Frequency = matrix(c(1, 1/2, 4,
                    2, 1, 6,
                    1/4, 1/6, 1),
```

```
> alt_Visibility
               Citycentre Shoppingcentre Industrialarea
Citycentre      1.0000000           2.00              6
Shoppingcentre  0.5000000           1.00              4
Industrialarea  0.1666667           0.25              1
> # Compute scores for alternatives w.r.t. criterion Visibility
> alt_Visibilitymatrix = pairwiseComparisonMatrix(alt_Visibility)
> alt_Visibilityweights = calculateWeights(alt_Visibilitymatrix)
> print(alt_Visibilityweights)
   w_Citycentre w_Shoppingcentre w_Industrialarea
         0.5876           0.3234           0.0890
>
```

Now we are going to convert for this comparison matrix for alternatives also we are going to convert into you know class of object which can be used by the functions of fuzzy AHP package. So therefore again we are going to use the pairwise comparison matrix here and calculate weights here. So we are going to compute scores for alternatives with respect to criteria visibility now in the format which is suitable for fuzzy AHP package.

Let us run this code. Let us compute you know scores. Let us print, so now we have got these scores with respect to visibility criteria, so each alternative and their scores.

(Refer Slide Time: 18:50)



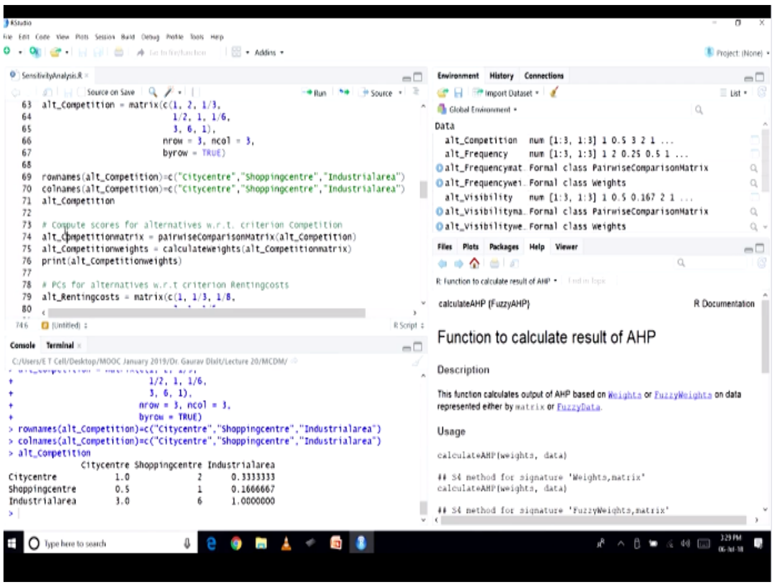
```
                    1/4, 1/6, 1),
                  nrow = 3, ncol = 3,
                  byrow = TRUE)

rownames(alt_Frequency)=c("Citycentre","Shoppingcentre","Industrialarea")
colnames(alt_Frequency)=c("Citycentre","Shoppingcentre","Industrialarea")
alt_Frequency

# Compute scores for alternatives w.r.t. criterion Frequency
alt_Frequencymatrix = pairwiseComparisonMatrix(alt_Frequency)
alt_Frequencyweights = calculateWeights(alt_Frequencymatrix)
print(alt_Frequencyweights)

# PCs for alternatives w.r.t criterion Competition
alt_Competition = matrix(c(1, 2, 1/3,
                    1/2, 1, 1/6,
                    3, 6, 1),
```

```
> alt_Frequency
               Citycentre Shoppingcentre Industrialarea
Citycentre           1.00      0.5000000              4
Shoppingcentre       2.00      1.0000000              6
Industrialarea       0.25      0.1666667              1
> # Compute scores for alternatives w.r.t. criterion Frequency
> alt_Frequencymatrix = pairwiseComparisonMatrix(alt_Frequency)
> alt_Frequencyweights = calculateWeights(alt_Frequencymatrix)
> print(alt_Frequencyweights)
   w_Citycentre w_Shoppingcentre w_Industrialarea
         0.3234           0.5876           0.0890
>
```

Now what we are going to do is the similar kind of exercise we are going to perform for you know for other criteria. So the frequency is the next one so again matrix and the command function to create the pairwise comparison matrix. Then, row names, column names, this is our comparison matrix. As you can see in the console output, now again we are going to compute the scores using these functions pairwise comparison matrix.

And then the calculate weights, now you can see these scores for these you know 3 alternatives with respect to criterion frequency. So this is done. Now let us move towards the next computation that is you know pairwise comparisons for alternatives with respect to our next criterion that is competition.

(Refer Slide Time: 19:53)

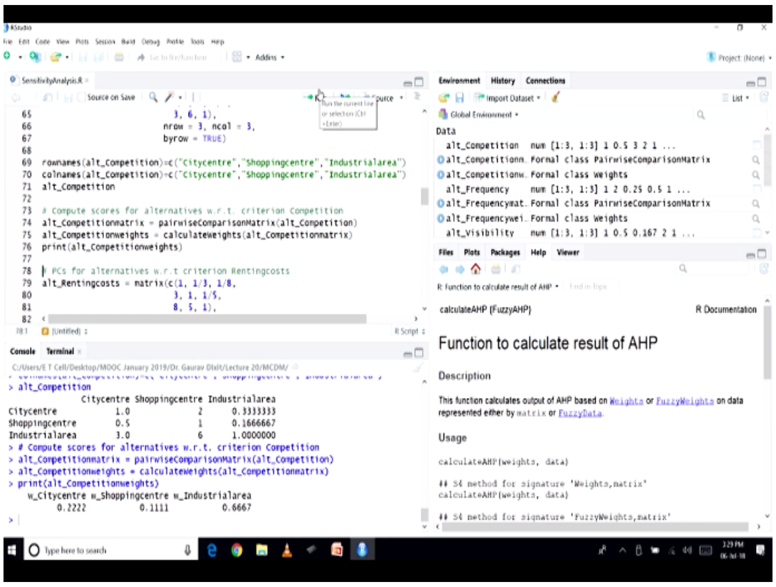


```
alt_Competition = matrix(c(1, 2, 1/3,
                           1/2, 1, 1/6,
                           3, 6, 1),
                         nrow = 3, ncol = 3,
                         byrow = TRUE)

rownames(alt_Competition)=c("Citycentre","Shoppingcentre","Industrialarea")
colnames(alt_Competition)=c("Citycentre","Shoppingcentre","Industrialarea")
alt_Competition

# Compute scores for alternatives w.r.t. criterion Competition
alt_Competitionmatrix = pairwiseComparisonMatrix(alt_Competition)
alt_Competitionweights = calculateWeights(alt_Competitionmatrix)
print(alt_Competitionweights)

# PCs for alternatives w.r.t criterion Rentingcosts
alt_Rentingcosts = matrix(c(1, 1/3, 1/8,
```

```
+                          1/2, 1, 1/6,
+                          3, 6, 1),
+                        nrow = 3, ncol = 3,
+                        byrow = TRUE)
> rownames(alt_Competition)=c("Citycentre","Shoppingcentre","Industrialarea")
> colnames(alt_Competition)=c("Citycentre","Shoppingcentre","Industrialarea")
> alt_Competition
               Citycentre Shoppingcentre Industrialarea
Citycentre            1.0              2      0.3333333
Shoppingcentre        0.5              1      0.1666667
Industrialarea        3.0              6      1.0000000
>
```

So again let us run this row names, column names and these are the values. Now we are going to calculate you know scores. So again first we need to use this function pairwise comparison matrix.

(Refer Slide Time: 20:10)

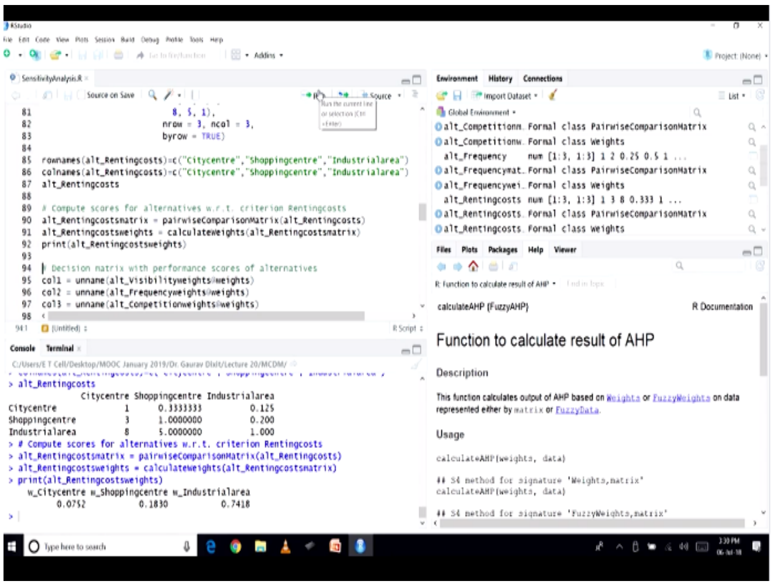


```
                           3, 6, 1),
                         nrow = 3, ncol = 3,
                         byrow = TRUE)

rownames(alt_Competition)=c("Citycentre","Shoppingcentre","Industrialarea")
colnames(alt_Competition)=c("Citycentre","Shoppingcentre","Industrialarea")
alt_Competition

# Compute scores for alternatives w.r.t. criterion Competition
alt_Competitionmatrix = pairwiseComparisonMatrix(alt_Competition)
alt_Competitionweights = calculateWeights(alt_Competitionmatrix)
print(alt_Competitionweights)

# PCs for alternatives w.r.t criterion Rentingcosts
alt_Rentingcosts = matrix(c(1, 1/3, 1/8,
                            3, 1, 1/5,
                            8, 5, 1),
```

```
> alt_Competition
               Citycentre Shoppingcentre Industrialarea
Citycentre            1.0              2      0.3333333
Shoppingcentre        0.5              1      0.1666667
Industrialarea        3.0              6      1.0000000
> # Compute scores for alternatives w.r.t. criterion Competition
> alt_Competitionmatrix = pairwiseComparisonMatrix(alt_Competition)
> alt_Competitionweights = calculateWeights(alt_Competitionmatrix)
> print(alt_Competitionweights)
  w_Citycentre w_Shoppingcentre w_Industrialarea
        0.2222           0.1111           0.6667
>
```

So that it is in the suitable format and then weights, let us see the weights, so these are the weights, so these are the scores for alternative with respect to criterion competition.

(Refer Slide Time: 20:25)



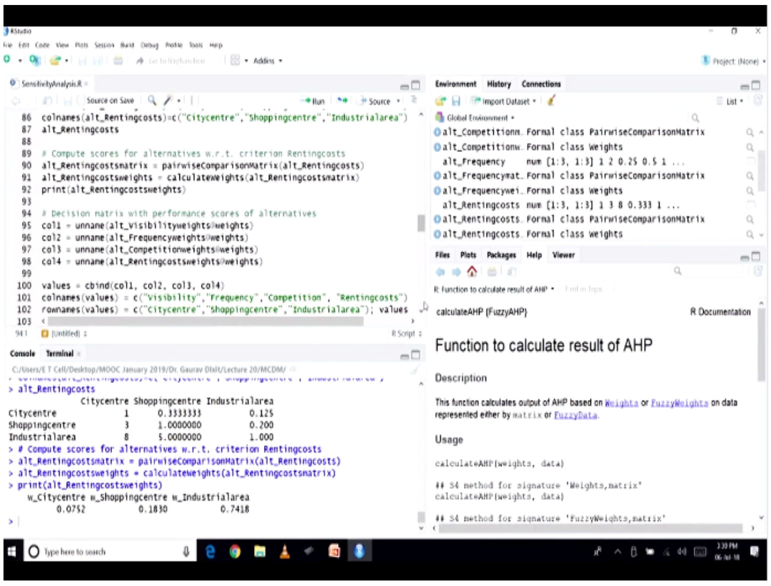
```
                     8, 5, 1),
                   nrow = 3, ncol = 3,
                   byrow = TRUE)

rownames(alt_Rentingcosts)=c("Citycentre","Shoppingcentre","Industrialarea")
colnames(alt_Rentingcosts)=c("Citycentre","Shoppingcentre","Industrialarea")
alt_Rentingcosts

# Compute scores for alternatives w.r.t. criterion Rentingcosts
alt_Rentingcostsmatrix = pairwiseComparisonMatrix(alt_Rentingcosts)
alt_Rentingcostsweights = calculateWeights(alt_Rentingcostsmatrix)
print(alt_Rentingcostsweights)

# Decision matrix with performance scores of alternatives
col1 = unname(alt_Visibilityweights@weights)
col2 = unname(alt_Frequencyweights@weights)
col3 = unname(alt_Competitionweights@weights)
```

```
> alt_Rentingcosts
               Citycentre Shoppingcentre Industrialarea
Citycentre              1      0.3333333          0.125
Shoppingcentre          3      1.0000000          0.200
Industrialarea          8      5.0000000          1.000
> # Compute scores for alternatives w.r.t. criterion Rentingcosts
> alt_Rentingcostsmatrix = pairwiseComparisonMatrix(alt_Rentingcosts)
> alt_Rentingcostsweights = calculateWeights(alt_Rentingcostsmatrix)
> print(alt_Rentingcostsweights)
   w_Citycentre w_Shoppingcentre w_Industrialarea
         0.0752           0.1830           0.7418
>
```

Function to calculate result of AHP
Description
This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.
Usage
calculateAHP(weights, data)

Now the last you know criterion that is renting cost here. So now again same set of similar set of lines here again for renting cost, row names, column names, the matrix and then pairwise comparison matrix function, then calculate weights, will get the scores and then the scores for. So these scores for alternatives with respect to renting cost criterion.

(Refer Slide Time: 20:58)



```
colnames(alt_Rentingcosts)=c("Citycentre","Shoppingcentre","Industrialarea")
alt_Rentingcosts

# Compute scores for alternatives w.r.t. criterion Rentingcosts
alt_Rentingcostsmatrix = pairwiseComparisonMatrix(alt_Rentingcosts)
alt_Rentingcostsweights = calculateWeights(alt_Rentingcostsmatrix)
print(alt_Rentingcostsweights)

# Decision matrix with performance scores of alternatives
col1 = unname(alt_Visibilityweights@weights)
col2 = unname(alt_Frequencyweights@weights)
col3 = unname(alt_Competitionweights@weights)
col4 = unname(alt_Rentingcostsweights@weights)

values = cbind(col1, col2, col3, col4)
colnames(values) = c("Visibility","Frequency","Competition", "Rentingcosts")
rownames(values) = c("Citycentre","Shoppingcentre","Industrialarea"); values
```

```
> alt_Rentingcosts
               Citycentre Shoppingcentre Industrialarea
Citycentre              1      0.3333333          0.125
Shoppingcentre          3      1.0000000          0.200
Industrialarea          8      5.0000000          1.000
> # Compute scores for alternatives w.r.t. criterion Rentingcosts
> alt_Rentingcostsmatrix = pairwiseComparisonMatrix(alt_Rentingcosts)
> alt_Rentingcostsweights = calculateWeights(alt_Rentingcostsmatrix)
> print(alt_Rentingcostsweights)
   w_Citycentre w_Shoppingcentre w_Industrialarea
         0.0752           0.1830           0.7418
>
```

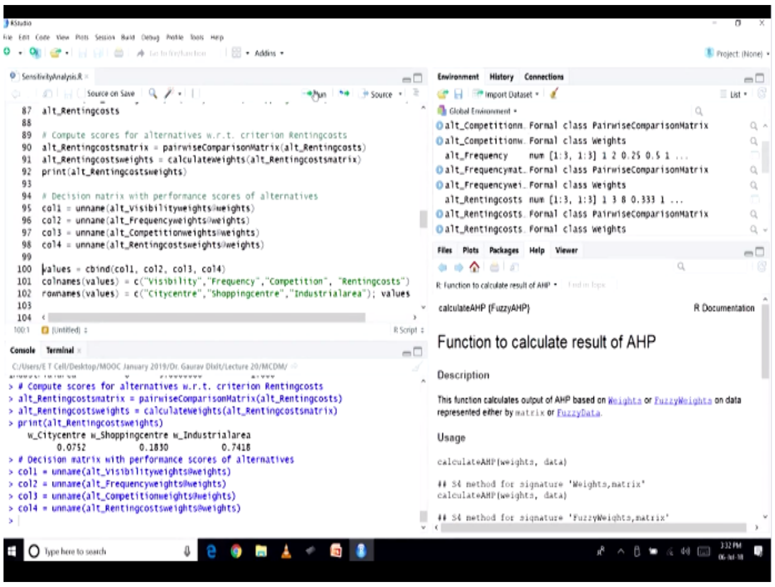
Now once we have got these scores, we can create our decision matrix, so as you know in the decision matrix what we require is on the column side will have the you know criteria, on the row side will have the alternatives and you know performance of these alternatives with respect to you know with respect to each criterion is to be there in the decision matrix. So what we are doing now is we are going to extract these scores you know that we have just computed.

So first you know visibility with respect to visibility criteria and then we can see then competition and then the renting cost here because you know the way you know these classes are defined you see we are using these scores alt visibility weights which we have already computed at the rate of weights to actually access the weights in their numeric format. So this is the notation that we need to use.

So that we are able to access the values because right now we are interested in the values here. So we are trying to extract that value in this using this particular piece of code. Now you are using a name function also. So you know right now we understand the values are not in the names that have been given there, so we are naming you know row names, column names that we have been assigning.

So we do not need that some of those you know names have been assigned by you know functions of the fuzzy AHP package as well. So we are going to remove you know those things and just extract the values here. So column 1, column 2, column 3, column 4 these are with respect to 4 criteria that we have. So let us extract to these scores for alternatives.

(Refer Slide Time: 22:45)



```
alt_Rentingcosts

# Compute scores for alternatives w.r.t. criterion Rentingcosts
alt_Rentingcostsmatrix = pairwiseComparisonMatrix(alt_Rentingcosts)
alt_Rentingcostsweights = calculateWeights(alt_Rentingcostsmatrix)
print(alt_Rentingcostsweights)

# Decision matrix with performance scores of alternatives
col1 = unname(alt_Visibilityweights@weights)
col2 = unname(alt_Frequencyweights@weights)
col3 = unname(alt_Competitionweights@weights)
col4 = unname(alt_Rentingcostsweights@weights)

values = cbind(col1, col2, col3, col4)
colnames(values) = c("Visibility","Frequency","Competition", "Rentingcosts")
rownames(values) = c("Citycentre","Shoppingcentre","Industrialarea"); values
```

```
> # Compute scores for alternatives w.r.t. criterion Rentingcosts
> alt_Rentingcostsmatrix = pairwiseComparisonMatrix(alt_Rentingcosts)
> alt_Rentingcostsweights = calculateWeights(alt_Rentingcostsmatrix)
> print(alt_Rentingcostsweights)
   w_Citycentre w_Shoppingcentre w_Industrialarea
         0.0752           0.1830           0.7418
> # Decision matrix with performance scores of alternatives
> col1 = unname(alt_Visibilityweights@weights)
> col2 = unname(alt_Frequencyweights@weights)
> col3 = unname(alt_Competitionweights@weights)
> col4 = unname(alt_Rentingcostsweights@weights)
>
```

```
alt_Competitionm… Formal class PairwiseComparisonMatrix
alt_Competitionw… Formal class Weights
alt_Frequency     num [1:3, 1:3] 1 2 0.25 0.5 1 ...
alt_Frequencymat… Formal class PairwiseComparisonMatrix
alt_Frequencywei… Formal class Weights
alt_Rentingcosts  num [1:3, 1:3] 1 3 8 0.333 1 ...
alt_Rentingcosts… Formal class PairwiseComparisonMatrix
alt_Rentingcosts… Formal class Weights
```

calculateAHP {FuzzyAHP} R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

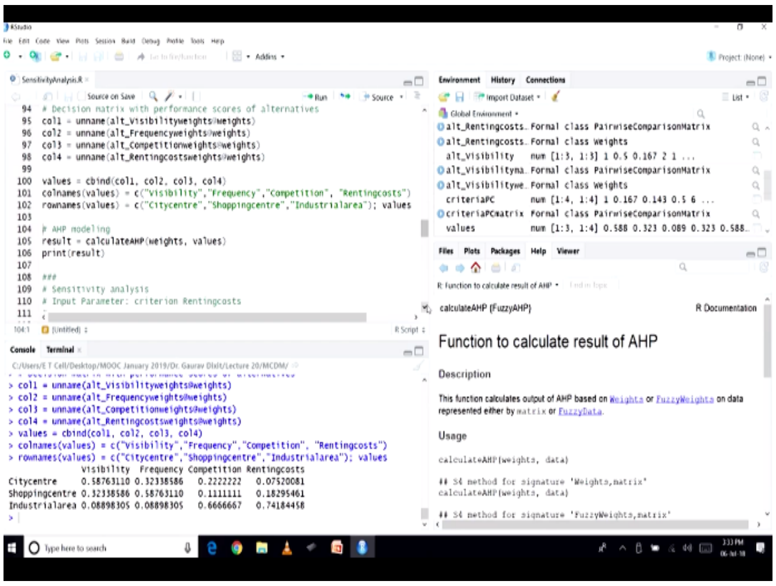
Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

So first column 1, column 2, column 3, column 4. Now once we have these you know 4 you know columns of the scores for alternatives, we are going to combine these 4 columns to generate our decision you know matrix. So for this we are going to use cbind function so cbind function is actually for columns to bind a number of columns.

So these column 1, column 2, column 3, column 4 we already have, we have already computed. Now we are going to use this cbind function to create our decision matrix.

(Refer Slide Time: 23:20)



```
# Decision matrix with performance scores of alternatives
col1 = unname(alt_Visibilityweights@weights)
col2 = unname(alt_Frequencyweights@weights)
col3 = unname(alt_Competitionweights@weights)
col4 = unname(alt_Rentingcostsweights@weights)

values = cbind(col1, col2, col3, col4)
colnames(values) = c("Visibility","Frequency","Competition", "Rentingcosts")
rownames(values) = c("Citycentre","Shoppingcentre","Industrialarea"); values

# AHP modeling
result = calculateAHP(weights, values)
print(result)

###
# Sensitivity analysis
# Input Parameter: criterion Rentingcosts
```

```
> col1 = unname(alt_Visibilityweights@weights)
> col2 = unname(alt_Frequencyweights@weights)
> col3 = unname(alt_Competitionweights@weights)
> col4 = unname(alt_Rentingcostsweights@weights)
> values = cbind(col1, col2, col3, col4)
> colnames(values) = c("Visibility","Frequency","Competition", "Rentingcosts")
> rownames(values) = c("Citycentre","Shoppingcentre","Industrialarea"); values
               Visibility  Frequency Competition Rentingcosts
Citycentre     0.58763110 0.32338586   0.2222222   0.07520081
Shoppingcentre 0.32338586 0.58763110   0.1111111   0.18295461
Industrialarea 0.08898305 0.08898305   0.6666667   0.74184458
>
```

```
alt_Rentingcosts… Formal class PairwiseComparisonMatrix
alt_Rentingcosts… Formal class Weights
alt_Visibility    num [1:3, 1:3] 1 0.5 0.167 2 1 ...
alt_Visibilityma… Formal class PairwiseComparisonMatrix
alt_Visibilitywe… Formal class Weights
criteriaPC        num [1:4, 1:4] 1 0.167 0.143 0.5 6 ...
criteriaPCmatrix  Formal class PairwiseComparisonMatrix
values            num [1:3, 1:4] 0.588 0.323 0.089 0.323 0.588…
```

calculateAHP (FuzzyAHP) R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

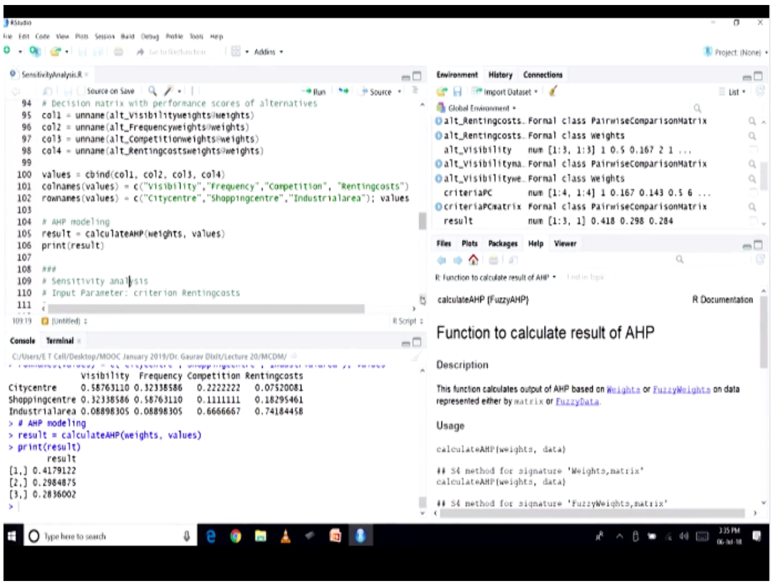
So let us run this. Now again we are going to name you know columns and rows, so again columns you know criteria names and rows alternatives name. So you can see now we have the decision matrix and the console output here. So you can see in the column side these are actually you know criteria names and on the row side what we have is you know alternatives, so performance of these alternatives with respect to each criterion.

And you can see the scores there on this matrix. Now we have computed criteria weights also and we have you know decision matrix also. So therefore we can go ahead and use the function, the model function that is calculated AHP. So in the calculated AHP, first argument is the criteria based, second argument is the values, this is decision matrix.

So when I was saying at the start of this lecture while discussing sensitivity analysis that why we have picked up you know selected this particular package because the model function calculate AHP, this is actually taking criteria weights as the input argument which is something that is not available in the other packages. So therefore it is easy for us to you know slightly vary you know these you know weights, input you know values with respect to input value with respect to criteria weights.

And we can analyze the impact on the model output, so therefore now what we are going to do is we are going to do a simple run of AHP modeling. So let us run this code.

(Refer Slide Time: 25:03)



```
# Decision matrix with performance scores of alternatives
col1 = unname(alt_Visibilityweights@weights)
col2 = unname(alt_Frequencyweights@weights)
col3 = unname(alt_Competitionweights@weights)
col4 = unname(alt_Rentingcostsweights@weights)

values = cbind(col1, col2, col3, col4)
colnames(values) = c("Visibility","Frequency","Competition", "Rentingcosts")
rownames(values) = c("Citycentre","Shoppingcentre","Industrialarea"); values

# AHP modeling
result = calculateAHP(weights, values)
print(result)

###
# Sensitivity analysis
# Input Parameter: criterion Rentingcosts
```

```
               Visibility  Frequency Competition Rentingcosts
Citycentre     0.58763110 0.32338586   0.2222222   0.07520081
Shoppingcentre 0.32338586 0.58763110   0.1111111   0.18295461
Industrialarea 0.08898305 0.08898305   0.6666667   0.74184458
> # AHP modeling
> result = calculateAHP(weights, values)
> print(result)
        result
[1,] 0.4179122
[2,] 0.2984875
[3,] 0.2836002
>
```

```
alt_Rentingcosts.. Formal class PairwiseComparisonMatrix
alt_Rentingcosts.. Formal class Weights
alt_Visibility     num [1:3, 1:3] 1 0.5 0.167 2 1 ...
alt_Visibilityma.. Formal class PairwiseComparisonMatrix
alt_Visibilitywe.. Formal class Weights
criteriaPC         num [1:4, 1:4] 1 0.167 0.143 0.5 6 ...
criteriaPCmatrix   Formal class PairwiseComparisonMatrix
result             num [1:3, 1] 0.418 0.298 0.284
```

calculateAHP {FuzzyAHP} R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

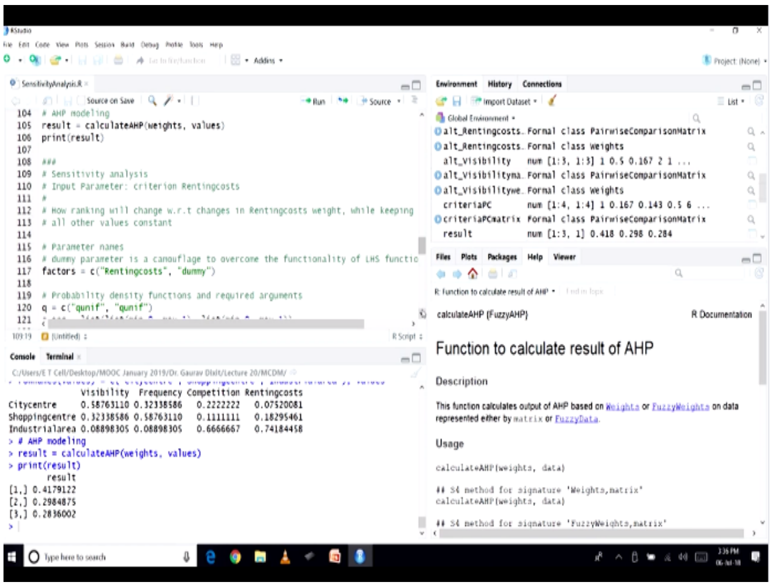
This is to tell you that given you know pairwise comparisons that we have initialized and the weights for criteria that we have computed and the decision matrix that we have computed, this is our model results. So as per this result you know the City Centre is you know 1 and then followed by you know shopping centre and following by industrial area. So you can see the values for shopping centre and industrial areas are very close.

And you know value of score for City Centre is actually on the higher side. So City Centre right now given the model results that we have City Centre is the best you know alternative. So you see when we were discussing this particular decision problem we said that probably City Centre is also going to be you know more expensive because of the prime location of the city and all that.

But despite that as per the model results and as per you know input data that was provided City Centre comes out to be the best ranked alternative there right. So now what we are going to do is we are going to perform our sensitivity analysis and you know because we are using that you know one-at-a-time approach, so in that approach we are going to pick the renting cost criterion as our input parameter.

So keeping all other values constant, we are going to change you know weight for this criterion renting cost and will analyze the impact on the model output. So our selected parameter is criterion renting cost.

(Refer Slide Time: 26:47)



```r
# AHP modeling
result = calculateAHP(weights, values)
print(result)

###
# Sensitivity analysis
# Input Parameter: criterion Rentingcosts
#
# How ranking will change w.r.t changes in Rentingcosts weight, while keeping
# all other values constant

# Parameter names
# dummy parameter is a camouflage to overcome the functionality of LHS functio
factors = c("Rentingcosts", "dummy")

# Probability density functions and required arguments
q = c("qunif", "qunif")
```

```
C:/Users/E T Cell/Desktop/MOOC January 2019/Dr. Gaurav Dixit/Lecture 20/MCDM/
               Visibility  Frequency Competition Rentingcosts
Citycentre     0.58763110 0.32338586   0.2222222   0.07520081
Shoppingcentre 0.32338586 0.58763110   0.1111111   0.18295461
Industrialarea 0.08898305 0.08898305   0.6666667   0.74184458
> # AHP modeling
> result = calculateAHP(weights, values)
> print(result)
        result
[1,] 0.4179122
[2,] 0.2984875
[3,] 0.2836002
>
```

```
alt_Rentingcosts… Formal class PairwiseComparisonMatrix
alt_Rentingcosts… Formal class Weights
alt_Visibility    num [1:3, 1:3] 1 0.5 0.167 2 1 ...
alt_Visibilityma… Formal class PairwiseComparisonMatrix
alt_Visibilitywe… Formal class Weights
criteriaPC        num [1:4, 1:4] 1 0.167 0.143 0.5 6 ...
criteriaPCmatrix  Formal class PairwiseComparisonMatrix
result            num [1:3, 1] 0.418 0.298 0.284
```

calculateAHP {FuzzyAHP} R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

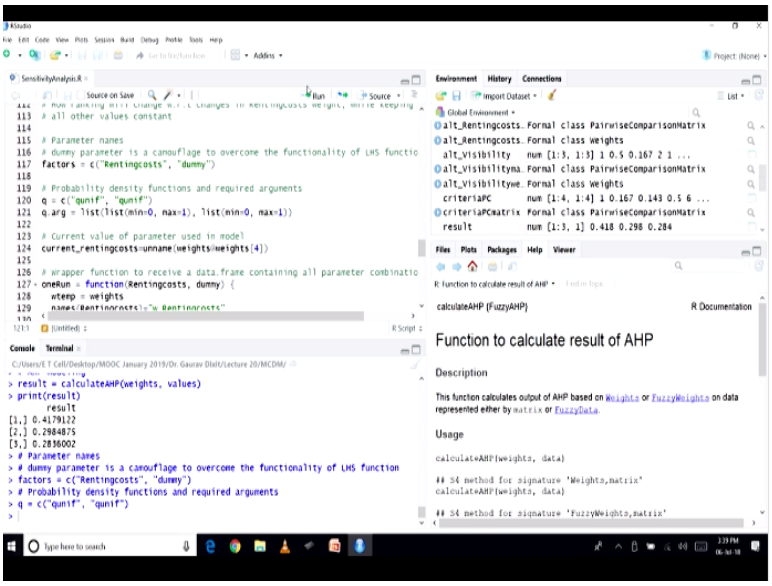
So you can see a few comments there also that how ranking will change with respect to changes in renting cost weight while keeping all other values constant. So this is what this is something that we are looking to perform. Now first thing that we need to declare here is the parameter names so which I have already mentioned that is renting cost.

However, the function in the PSE package that we are going to use is there is some problem in the function in the sense it is not you know it produces a number of errors if we are using just one parameter. So this function is actually designed for you know simultaneous changes in a number of parameters and analyzing the impact on you know model output and when we use just one parameter you know there you know you might encounter certain errors there.

So to camouflage to overcome this situation because we are using one-at-a-time approach and we are going to use just you know one variable going to change this one variable and looking at the impact on the model output. So given the approach that we want to implement we are using a dummy parameter to camouflage to overcome the functionality that we have noticed in our experience for this package PSE package and LHS function in particular.

So LHS function is actually to generate the hypercube there. So depending on the requirement, depending on the argument that we need to pass on to this function, we are going to create certain variables and initialize them. So first one is factors, this is to name the parameters. So we have just one parameter renting cost, the other one is dummy which is just to camouflage the function.

(Refer Slide Time: 28:45)



```r
# all other values constant

# Parameter names
# dummy parameter is a camouflage to overcome the functionality of LHS functio
factors = c("Rentingcosts", "dummy")

# Probability density functions and required arguments
q = c("qunif", "qunif")
q.arg = list(list(min=0, max=1), list(min=0, max=1))

# Current value of parameter used in model
current_rentingcosts=unname(weights@weights[4])

# wrapper function to receive a data.frame containing all parameter combinatio
oneRun = function(Rentingcosts, dummy) {
  wtemp = weights
```

```
> result = calculateAHP(weights, values)
> print(result)
        result
[1,] 0.4179122
[2,] 0.2984875
[3,] 0.2836002
> # Parameter names
> # dummy parameter is a camouflage to overcome the functionality of LHS function
> factors = c("Rentingcosts", "dummy")
> # Probability density functions and required arguments
> q = c("qunif", "qunif")
>
```

```
alt_Rentingcosts.. Formal class PairwiseComparisonMatrix
alt_Rentingcosts.. Formal class Weights
alt_Visibility     num [1:3, 1:3] 1 0.5 0.167 2 1 ...
alt_Visibilityma.. Formal class PairwiseComparisonMatrix
alt_Visibilitywe.. Formal class Weights
criteriaPC         num [1:4, 1:4] 1 0.167 0.143 0.5 6 ...
criteriaPCmatrix   Formal class PairwiseComparisonMatrix
result             num [1:3, 1] 0.418 0.298 0.284
```

calculateAHP (FuzzyAHP) R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

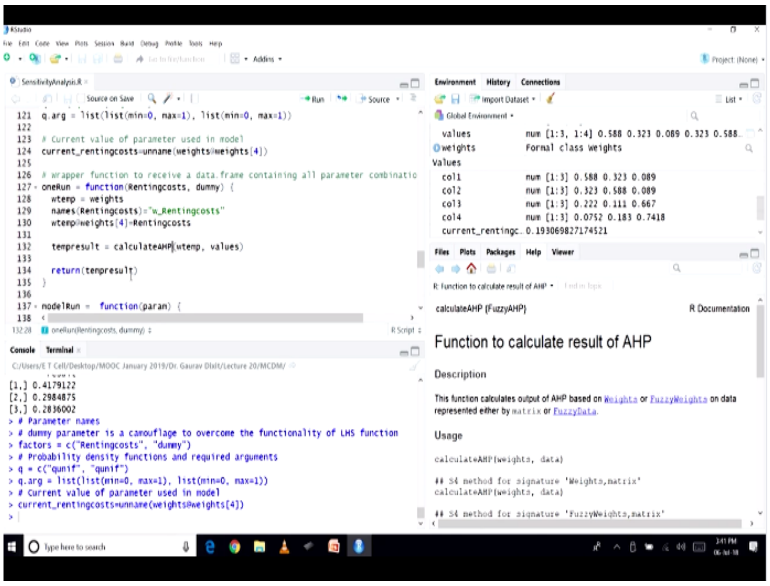
Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

Let us run this. Then, probability density function and the required arguments for those you know function so here we are using uniform distribution, so we have the quantile built-in function available in R environment, so we are using qunif for you know for our parameter. So let us initialize this variable also so that we can select our probability density function and then you know certain you know parameters that are to be used by these you know probability density function qunif.

So that is the mean value and max value, so that we have passed on a 0 and 1 because the criteria weights it has to be between 0 and 1. So that is you know range where the values should lie. So that is something that we have (()) (29:36), so let us run this code.

(Refer Slide Time: 29:37)



```r
q.arg = list(list(min=0, max=1), list(min=0, max=1))

# Current value of parameter used in model
current_rentingcosts=unname(weights@weights[4])

# wrapper function to receive a data.frame containing all parameter combinatio
oneRun = function(Rentingcosts, dummy) {
  wtemp = weights
  names(Rentingcosts)="w_Rentingcosts"
  wtemp@weights[4]=Rentingcosts

  tempresult = calculateAHP(wtemp, values)

  return(tempresult)
}

modelRun =  function(param) {
```

```
[1,] 0.4179122
[2,] 0.2984875
[3,] 0.2836002
> # Parameter names
> # dummy parameter is a camouflage to overcome the functionality of LHS function
> factors = c("Rentingcosts", "dummy")
> # Probability density functions and required arguments
> q = c("qunif", "qunif")
> q.arg = list(list(min=0, max=1), list(min=0, max=1))
> # Current value of parameter used in model
> current_rentingcosts=unname(weights@weights[4])
>
```

```
values                num [1:3, 1:4] 0.588 0.323 0.089 0.323 0.588…
weights               Formal class Weights
Values
  col1                num [1:3] 0.588 0.323 0.089
  col2                num [1:3] 0.323 0.588 0.089
  col3                num [1:3] 0.222 0.111 0.667
  col4                num [1:3] 0.0752 0.183 0.7418
  current_rentingc…   0.193069827174521
```

calculateAHP {FuzzyAHP}   R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

Now we are also recording the current value as per the model result that we have in this variable current renting cost, so given the model results that we had we are recording this, later on we will be using this value so let us record this. Now I have written wrapper function. So this wrapper function is actually designed to receive a data frame that contains all parameter combinations.

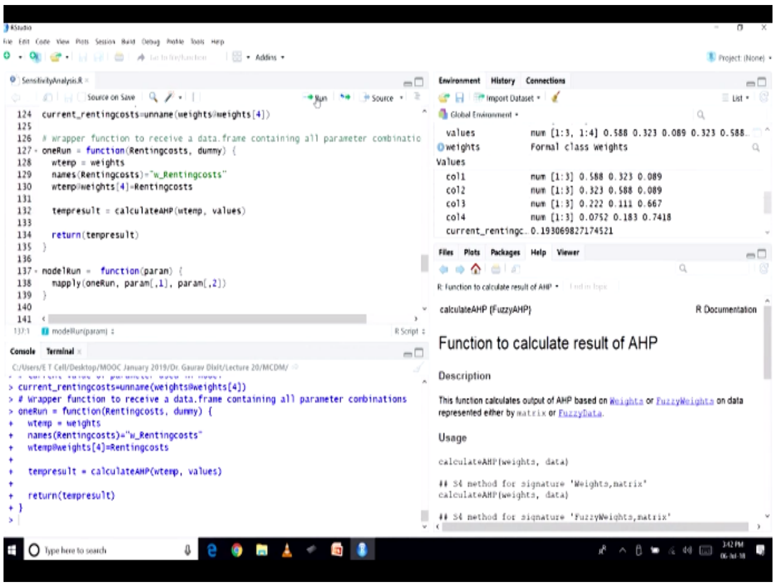
So right now we have two parameter combinations, one is renting cost, the second one is the dummy. So all you know data frame should be you know containing all these parameter combinations. However, if you look at the model function, our model function takes you know just one set of these combinations and produce the model results. So therefore model function cannot be used directly.

Therefore, we need to write a wrapper function based you know keeps on taking the values one combination each at a time and keeps on generating the model results because later we will be using you know these you know number of runs of model results and you know produce output to analyze the sensitivity. So this one run function is there, so what does is we are taking this value of renting cost which is going to be varied using the uniform distribution as we have detailed you know earlier.

You know uniform distribution between 0 to 1 the values are going to be produced as per the quantile function qunif and these values are going to be passed on to this wrapper function and will get you know number of model output. So how this function is going to behave is first you know as you know that calculate AHP here, it is going to take two arguments, one for criteria weights, another for values so values will be not changing.

Because as I said one-at-a-time approach so we are not changing other you know things there. So values are not supposed to change and the other criteria you know weights that are also we are not supposed to change. So here you can see that we are just changing one value that is you know associated with the renting cost in this line of code so that is changed and then the weights are passed on to calculate AHP function and will get the model result.

(Refer Slide Time: 32:05)



```
current_rentingcosts=unname(weights@weights[4])

# wrapper function to receive a data.frame containing all parameter combinatio
oneRun = function(Rentingcosts, dummy) {
  wtemp = weights
  names(Rentingcosts)="w_Rentingcosts"
  wtemp@weights[4]=Rentingcosts

  tempresult = calculateAHP(wtemp, values)

  return(tempresult)
}

modelRun =  function(param) {
  mapply(oneRun, param[,1], param[,2])
}
```

```
> current_rentingcosts=unname(weights@weights[4])
> # Wrapper function to receive a data.frame containing all parameter combinations
> oneRun = function(Rentingcosts, dummy) {
+   wtemp = weights
+   names(Rentingcosts)="w_Rentingcosts"
+   wtemp@weights[4]=Rentingcosts
+
+   tempresult = calculateAHP(wtemp, values)
+
+   return(tempresult)
+ }
>
```

```
values            num [1:3, 1:4] 0.588 0.323 0.089 0.323 0.588...
weights           Formal class weights
Values
  col1            num [1:3] 0.588 0.323 0.089
  col2            num [1:3] 0.323 0.588 0.089
  col3            num [1:3] 0.222 0.111 0.667
  col4            num [1:3] 0.0752 0.183 0.7418
  current_rentingc... 0.19306982717452
```

calculateAHP (FuzzyAHP) R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

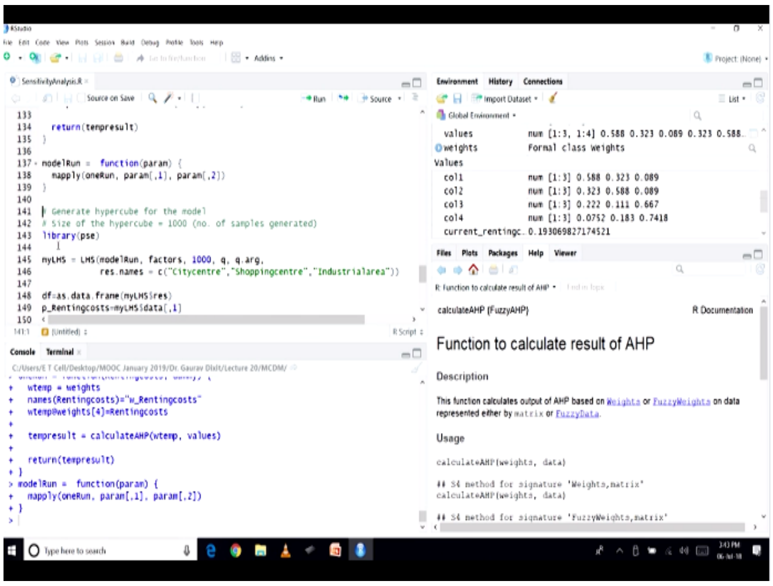
Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

Now let us run this code so that this function is defined. Now we have model run function which is actually calling this one run function by passing on you know these you know each time one set of combination, so this is the function that is actually the wrapper and it is just calling one function. So here we are using m apply and one run is being you know called upon using the values of you know second and third arguments.

Parameter 1 with respect to renting cost, parameter 2 with respect to you know dummy that will not do anything. As you can see in the one run code, we have not used (()) (32:45) dummy anywhere, it is just being used as you know as dummy as a camouflage there. So will model run so every set of combination so you know this one run will be called and the model result output will be produced.

(Refer Slide Time: 33:01)



```
  return(tempresult)
}

modelRun =  function(param) {
  mapply(oneRun, param[,1], param[,2])
}

# Generate hypercube for the model
# Size of the hypercube = 1000 (no. of samples generated)
library(pse)

myLHS = LHS(modelRun, factors, 1000, q, q.arg,
            res.names = c("Citycentre","Shoppingcentre","Industrialarea"))

df=as.data.frame(myLHS$res)
p_Rentingcosts=myLHS$data[,1]
```

```
C:/Users/E T Cell/Desktop/MOOC January 2019/Dr. Gaurav Dixit/Lecture 20/MCDM/
+   wtemp = weights
+   names(Rentingcosts)="w_Rentingcosts"
+   wtemp@weights[4]=Rentingcosts
+
+   tempresult = calculateAHP(wtemp, values)
+
+   return(tempresult)
+ }
> modelRun =  function(param) {
+   mapply(oneRun, param[,1], param[,2])
+ }
>
```

```
values             num [1:3, 1:4] 0.588 0.323 0.089 0.323 0.588...
weights            Formal class weights
Values
  col1             num [1:3] 0.588 0.323 0.089
  col2             num [1:3] 0.323 0.588 0.089
  col3             num [1:3] 0.222 0.111 0.667
  col4             num [1:3] 0.0752 0.183 0.7418
  current_rentingc.. 0.193069827174521
```

calculateAHP {FuzzyAHP} R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

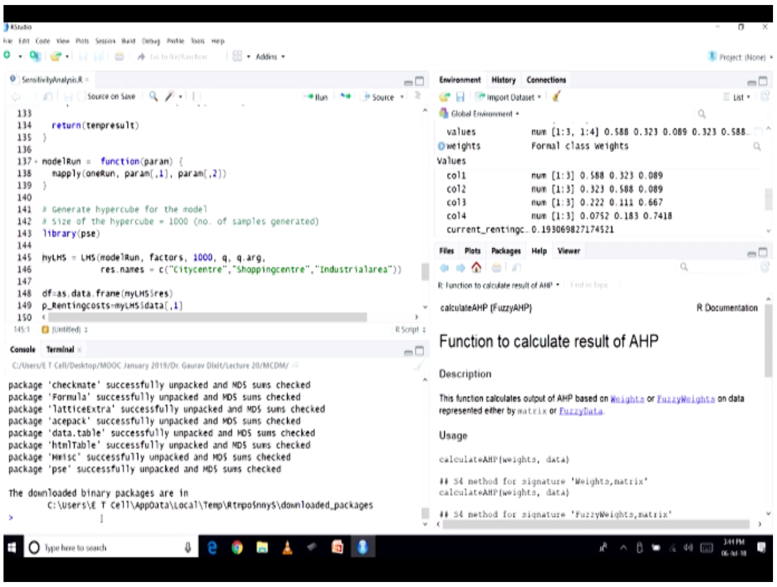
Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

So let us run this code and will get the result. I will define the function. Now these functions are defined. Now we are going to use this you know we are going to generate hypercube for the model, size of the hypercube is 1000 that means we are you know planning to you know generate 1000 samples. So actually 1000 model results are going to be produced and we will be analyzing you know the changes with respect to these 1000 sample points.

So when we said model function, this is now accepting data frame and 1000 set of combinations of arguments for one run is going to be produced and passed on using the LHS function and then the one run will execute and produce the results which are going to be recorded. So let us load this library PSE.

(Refer Slide Time: 33:54)

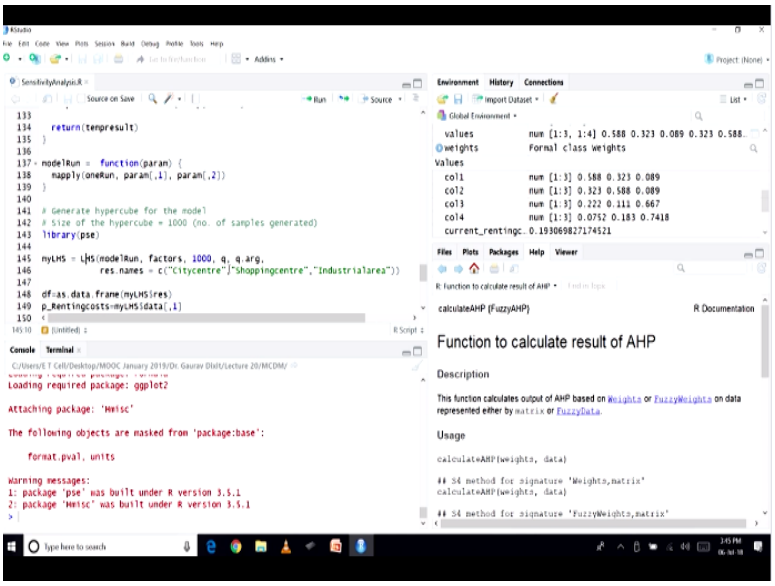


```
C:/Users/E T Cell/Desktop/MOOC January 2019/Dr. Gaurav Dixit/Lecture 20/MCDM/
package 'checkmate' successfully unpacked and MD5 sums checked
package 'Formula' successfully unpacked and MD5 sums checked
package 'latticeExtra' successfully unpacked and MD5 sums checked
package 'acepack' successfully unpacked and MD5 sums checked
package 'data.table' successfully unpacked and MD5 sums checked
package 'htmlTable' successfully unpacked and MD5 sums checked
package 'Hmisc' successfully unpacked and MD5 sums checked
package 'pse' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\E T Cell\AppData\Local\Temp\RtmpoSnnyS\downloaded_packages
>
```

So that we need so this package is right now not installed so let me first install this you know package so that we are able to use the function. So as we have been doing in other lectures we have to type install dot packages and within the double quotes we have type PSE and this package will be installed and then we will be using the LHS function that is available in this package, so first argument as you can see in the code is the model run.

So in this package LHS, we are going to call this model function which is actually a wrapper for the original model function calculate AHP. So this is going to be called and the input parameter that we have renting cost. So this input parameter is going to be varied and you know output would be recorded and will generate the hypercube. So package is now installed, so let us load the library into R environment.

(Refer Slide Time: 35:15)



```
  return(tempresult)
}

modelRun =  function(param) {
  mapply(oneRun, param[,1], param[,2])
}

# Generate hypercube for the model
# Size of the hypercube = 1000 (no. of samples generated)
library(pse)

myLHS = LHS(modelRun, factors, 1000, q, q.arg,
            res.names = c("Citycentre","Shoppingcentre","Industrialarea"))

df=as.data.frame(myLHS$res)
p_Rentingcosts=myLHS$data[,1]
```

```
C:/Users/E T Cell/Desktop/MOOC January 2019/Dr. Gaurav Dixit/Lecture 20/MCDM/
Loading required package: ggplot2

Attaching package: 'Hmisc'

The following objects are masked from 'package:base':

    format.pval, units

Warning messages:
1: package 'pse' was built under R version 3.5.1
2: package 'Hmisc' was built under R version 3.5.1
>
```

```
  values             num [1:3, 1:4] 0.588 0.323 0.089 0.323 0.588...
  weights            Formal class Weights
Values
  col1               num [1:3] 0.588 0.323 0.089
  col2               num [1:3] 0.323 0.588 0.089
  col3               num [1:3] 0.222 0.111 0.667
  col4               num [1:3] 0.0752 0.183 0.7418
  current_rentingc.. 0.193069827174521
```

calculateAHP (FuzzyAHP) R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

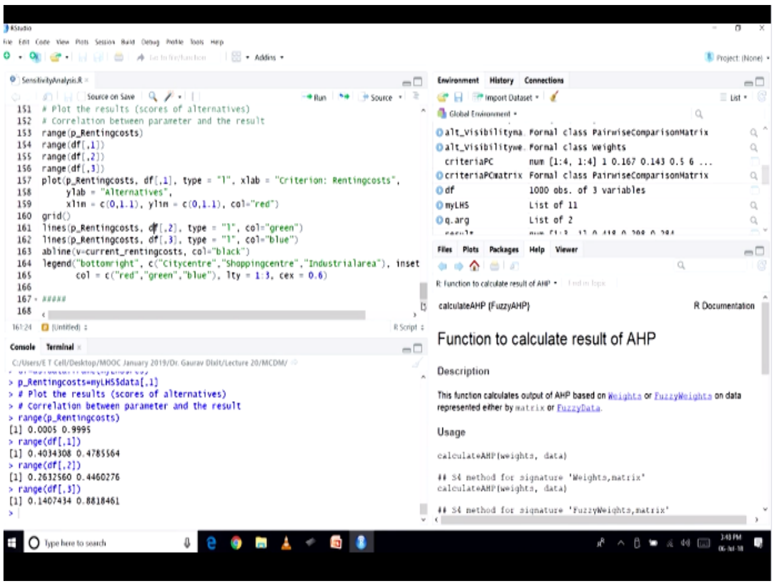
Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

So once this is done will be able to use this function. So this is done. So now let us talk about this LHS function. So model run which we have just defined, you know factors are the names of the parameters already defined, 1000 that is the number of sample points that we want to generate the size of the hypercube, q and q dot arg we have already defined and you know results names that is we have 3 alternatives.

So as you have seen in the calculate AHP model results PE got 3 scores for these 3 alternatives and City Centre was the best alternative. So we are just giving out these names so that you know it could be used by the function. So let us run this code.

(Refer Slide Time: 36:01)



```
# Plot the results (scores of alternatives)
# Correlation between parameter and the result
range(p_Rentingcosts)
range(df[,1])
range(df[,2])
range(df[,3])
plot(p_Rentingcosts, df[,1], type = "l", xlab = "Criterion: Rentingcosts",
     ylab = "Alternatives",
     xlim = c(0,1.1), ylim = c(0,1.1), col="red")
grid()
lines(p_Rentingcosts, df[,2], type = "l", col="green")
lines(p_Rentingcosts, df[,3], type = "l", col="blue")
abline(v=current_rentingcosts, col="black")
legend("bottomright", c("Citycentre","Shoppingcentre","Industrialarea"), inset
       col = c("red","green","blue"), lty = 1:3, cex = 0.6)

#####
```

```
> p_Rentingcosts=myLHS$data[,1]
> # Plot the results (scores of alternatives)
> # Correlation between parameter and the result
> range(p_Rentingcosts)
[1] 0.0005 0.9995
> range(df[,1])
[1] 0.4034308 0.4785564
> range(df[,2])
[1] 0.2632560 0.4460276
> range(df[,3])
[1] 0.1407434 0.8818461
>
```

```
alt_visibilityma  Formal class PairwiseComparisonMatrix
alt_visibilitywe  Formal class Weights
criteriaPC        num [1:4, 1:4] 1 0.167 0.143 0.5 6 ...
criteriaPCmatrix  Formal class PairwiseComparisonMatrix
df                1000 obs. of 3 variables
myLHS             List of 11
q.arg             List of 2
```

calculateAHP (FuzzyAHP) R Documentation

Function to calculate result of AHP

Description

This function calculates output of AHP based on Weights or FuzzyWeights on data represented either by matrix or FuzzyData.

Usage

```
calculateAHP(weights, data)

## S4 method for signature 'Weights,matrix'
calculateAHP(weights, data)

## S4 method for signature 'FuzzyWeights,matrix'
```

And because there are 1000 sample points just takes a bit of time. Now we are going to record the output of you know this LHS function, the model results that have been generated. So we are going to record them here in the state of m, so as dot data dot frame is the function and will record this. Then, renting cost these values data that has been generated by these function to run the model.

That also we are recording here with parameter renting cost. So these results also available you can see in the environment section as well that here p renting cost is there, p renting cost and you know the data frame that we have just you know created that is also df is there 1000 observation 3 variables because with respect to 3 alternatives that we have.

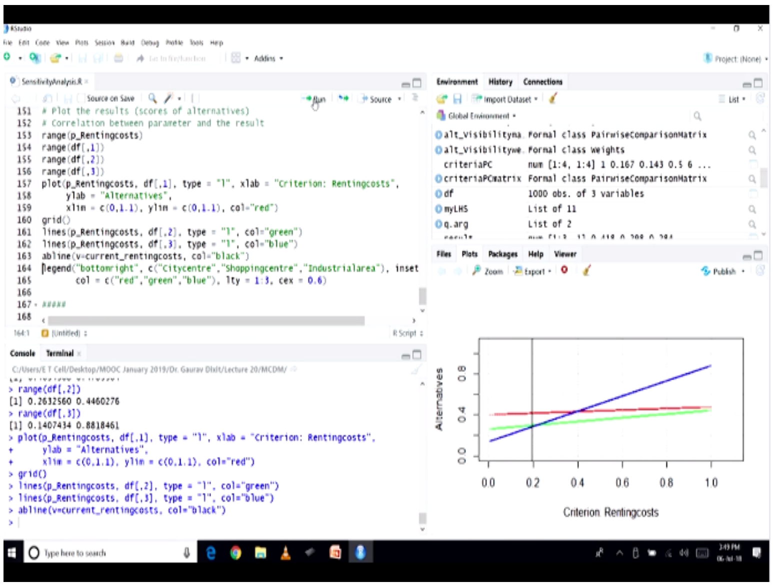
Now once we have generated the output, generated the hypercube, now what we are going to do is we are going to plot you know we are going to plot this you know correlation between parameter that is renting cost and the model result right. So these scores that we have for alternatives so they are going to be plotted with respect to the parameter. So let us look at the range of these variables so that we can use that in our plot function.

So let us look at the range. All these ranges are anyway going to be between 0 and 1. However, for certain other variables you know in other situations, it could be something different. So as a good practice you can always check at the range. So all these ranges that we can see 0, 1 so this also gives us confirmation that everything is going you know fine in terms of you know being within the range.

Now we are going to use the plot function where first argument is the parameter p renting cost. This is going to be plotted along you know x-axis. Then we have df 1 which is you know with respect to you know City Centre. So this is going to be plotted in red color and then we have lines function, grid function which will just clear the grid and lines function which will plot you know on the y-axis the second output that is with respect to shopping centre.

Then, the third one lines function. This will plot you know output with respect to industrial area. So let us run some of these codes.

(Refer Slide Time: 38:47)



```
# Plot the results (scores of alternatives)
# Correlation between parameter and the result
range(p_Rentingcosts)
range(df[,1])
range(df[,2])
range(df[,3])
plot(p_Rentingcosts, df[,1], type = "l", xlab = "Criterion: Rentingcosts",
     ylab = "Alternatives",
     xlim = c(0,1.1), ylim = c(0,1.1), col="red")
grid()
lines(p_Rentingcosts, df[,2], type = "l", col="green")
lines(p_Rentingcosts, df[,3], type = "l", col="blue")
abline(v=current_rentingcosts, col="black")
legend("bottomright", c("Citycentre","Shoppingcentre","Industrialarea"), inset
       col = c("red","green","blue"), lty = 1:3, cex = 0.6)

#####
```

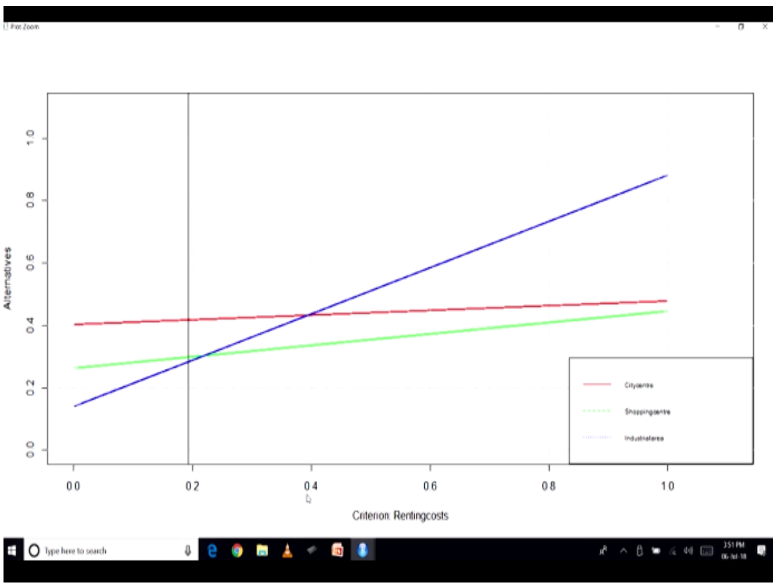
```
alt_Visibilityma  Formal class PairwiseComparisonMatrix
alt_Visibilitywe  Formal class Weights
criteriaPC        num [1:4, 1:4] 1 0.167 0.143 0.5 6 ...
criteriaPCmatrix  Formal class PairwiseComparisonMatrix
df                1000 obs. of 3 variables
myLHS             List of 11
q.arg             List of 2
```

```
> range(df[,2])
[1] 0.2632560 0.4460276
> range(df[,3])
[1] 0.1407434 0.8818461
> plot(p_Rentingcosts, df[,1], type = "l", xlab = "Criterion: Rentingcosts",
+      ylab = "Alternatives",
+      xlim = c(0,1.1), ylim = c(0,1.1), col="red")
> grid()
> lines(p_Rentingcosts, df[,2], type = "l", col="green")
> lines(p_Rentingcosts, df[,3], type = "l", col="blue")
> abline(v=current_rentingcosts, col="black")
>
```

So plot, you can see a plot has been generated here. Now let us run the so grid is nothing but it creates a certain line so that to give us a sense of where the values are lying there. So let us run another line of code so lines, this is now so first one was you know City Centre then we have shopping centre, now let us plot the industrial areas. So this is also there. Now the current value of renting cost, the model output that we had right.

So the model output with respect to you know the current renting cost value that we have, will also plot that as a vertical line here we can see. This is around 0.19 and the model output as we saw was you know City Centre was the best alternative as I described there. It had value around 0.41 right and the other values for the shopping centre and industrial area were around the range of 0.28, 0.29. So you can see here in the graphics so that being the case.

(Refer Slide Time: 39:55)



So let us run this legend also. Now we can analyze this graphics. Now let us analyze you know through this plot, so on the y-axis we have alternatives, we have 3 alternatives, City Centre, industrial area and you know shopping centre. So you can see in the legend that is mentioned. So red one is the City Centre, the green one is the shopping centre and the blue one is the industrial area.

And the black line that you are seeing around the 0.19 that is the current model results. So if you look at the current model results, City Centre came out as the best, the same thing is you know similarly it is depicted here as well. Now if you look at here if you look at the model B change the value of renting cost from 0.19 to something about 0.4 or you know more than 0.4 then you would see that industrial area would become the best alternative.

So therefore you know a big change in the renting cost would be required to change this alternative. However, once the rate of renting cost goes more than 0.4 about this point here or more than this then the industrial area is going to remain the best alternative right. So from looking at this graph, we can clearly you know observe that shopping centre is you know it is never going to be the best alternative.

So till about 0.4 value of renting cost it is the City Centre and after that it is the industrial area. Similarly, we can analyze the sensitivity of results model results with respect to other input parameters, other criteria that we have and we can find out which is the critical criteria you know in the model results are sensitive to which criteria in a relative sense. So all that can be analyzed through this kind of graphics you know that we have just computed.

So with this we conclude our discussion on sensitivity analysis and this is also the final lecture of this course MCDM techniques using R. So we have been able to cover a number of techniques starting from AHP, ELECTRA you know TOPSIS, VIKOR and fuzzy AHP, fuzzy sets and thus we have also talked about in today's lecture on you know sensitivity analysis. So we have been able to you know cover a healthy portion of MCDM techniques.

And based on the responses that we you know receive from your side on the NPTEL platform, we will definitely plan for another part of this course where we would be covering another set of you know MCDM techniques and you will also get you know flavor of R environment, how to you know do modelling, how to write your code and check results, do your analysis in R environment also. So hope you enjoyed the course. Thank you.