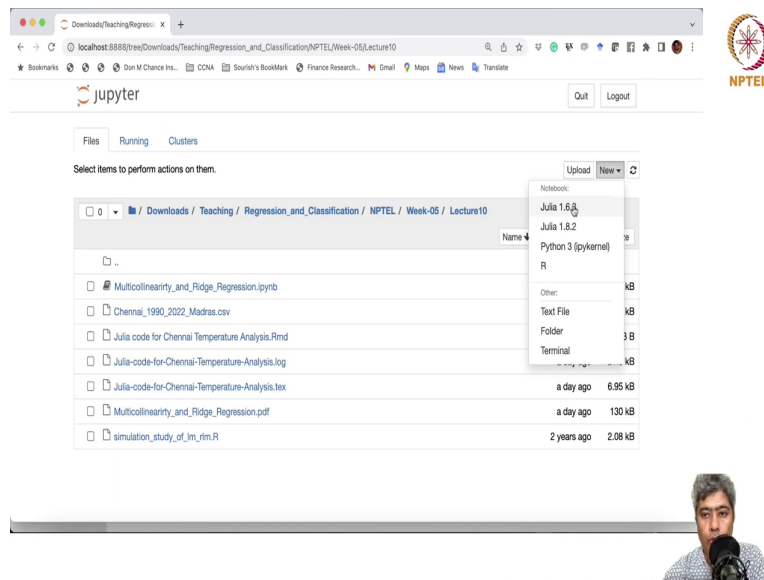


**Predictive Analytics - Regression and Classification**  
**Prof. Sourish Das**  
**Department of Mathematics**  
**Chennai Mathematical Institute**

**Lecture - 35**  
**Hands on with Julia: Implemente Chennai Temperature Analysis with Julia and CRRao**

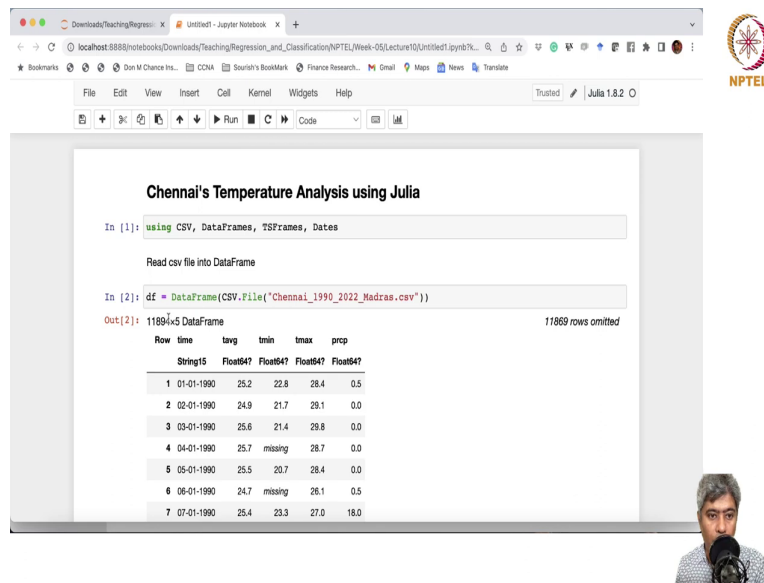
Hello all. Welcome to the Part B of Lecture 10. In this video, we are going to handle we are going to do some hands-on using Julia on Chennai's temperature data.

(Refer Slide Time: 00:33)



So, we will open Julia notebook.

(Refer Slide Time: 00:40)



Chennai's Temperature Analysis using Julia

```
In [1]: using CSV, DataFrames, TSFrames, Dates
```

Read csv file into DataFrame

```
In [2]: df = DataFrame(CSV.File("Chennai_1990_2022_Madras.csv"))
```

Out[2]: 11869x5 DataFrame 11869 rows omitted

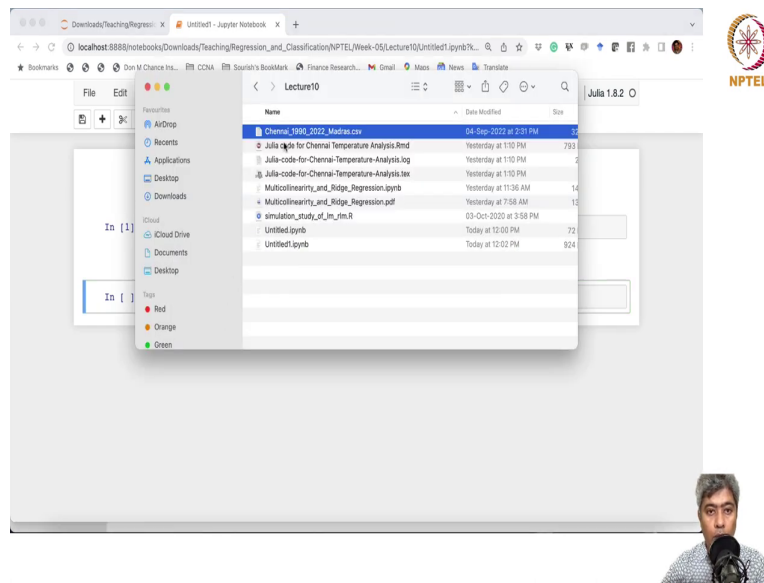
Row	time	tavg	tmin	tmax	prcp
1	01-01-1990	25.2	22.8	28.4	0.5
2	02-01-1990	24.9	21.7	29.1	0.0
3	03-01-1990	25.6	21.4	29.8	0.0
4	04-01-1990	25.7	missing	28.7	0.0
5	05-01-1990	25.5	20.7	28.4	0.0
6	06-01-1990	24.7	missing	26.1	0.5
7	07-01-1990	25.4	23.3	27.0	18.0

We want to open the Julia 1.8.2. So, first we will do some, give some headings that Chennai's Temperature Analysis using Julia. So, let me Run it.

Now, first we will do using CSV, comma DataFrames, comma, TSFrames, comma, Dates. So, these are the packages we will need CSV, we will read the CSV file and then we will put in your data frame. TS frame is another package which is built on the top of data frames. It helps us to handle time series data and then dates are essentially help us to do some data operations. So, let us Run this.

Next, we will do read CSV file into data frame. So, read CSV file into data frame. So, what we will do is a df DataFrame C from CSV dot file will do open, quote unquote.

(Refer Slide Time: 02:53)



And, then this is the file that we want to read and let us Run that. So, it has read 1108, 11894 rows and five columns. So, first column is time, but it read it as a string. So, date is dated as a string. So, we need to convert it into string. taverage is the average temperature of Chennai, tmin is the on that particular day. So, 25.2 is the average temperature of Chennai on first January, 1990.

So, tmin is the minimum temperature of Chennai on 1st January, 1990. tmax is 28.4, which is 28.4 is the max temperature on first January 1990 and precipitation 0.5 some precipitation took place on that particular day.

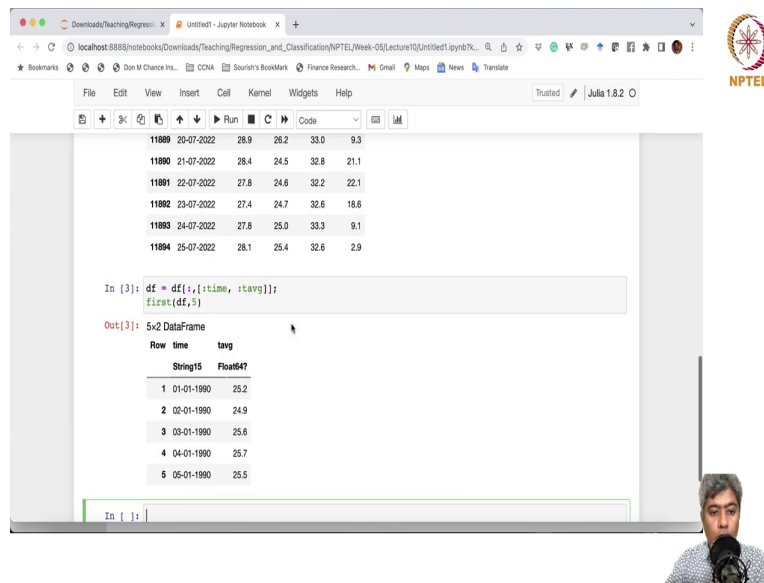
(Refer Slide Time: 04:02)

The screenshot shows a Jupyter Notebook interface with a data table. The table has four columns: a date column, and three numerical columns. The data is as follows:

3	03-01-1990	25.6	21.4	28.8	0.0
4	04-01-1990	25.7	missing	28.7	0.0
5	05-01-1990	25.5	20.7	28.4	0.0
6	06-01-1990	24.7	missing	26.1	0.5
7	07-01-1990	25.4	23.3	27.0	18.0
8	08-01-1990	25.6	22.0	28.0	0.5
9	09-01-1990	24.8	21.7	28.5	0.0
10	10-01-1990	24.7	20.7	29.0	0.0
11	11-01-1990	24.5	20.0	28.8	0.0
12	12-01-1990	23.8	18.0	29.4	0.0
13	13-01-1990	23.1	17.1	31.3	0.0
...	...	...	...	...	...
11883	14-07-2022	31.3	27.8	36.0	0.0
11884	15-07-2022	30.7	27.3	37.0	0.4
11885	16-07-2022	29.9	25.0	35.9	6.5
11886	17-07-2022	28.4	25.7	33.0	8.5
11887	18-07-2022	28.2	25.5	32.0	2.2
11888	19-07-2022	28.9	25.9	33.7	3.1

So, next what we will do?

(Refer Slide Time: 04:05)



```
In [3]: df = df[:, :time, :tavg];
first(df, 5)

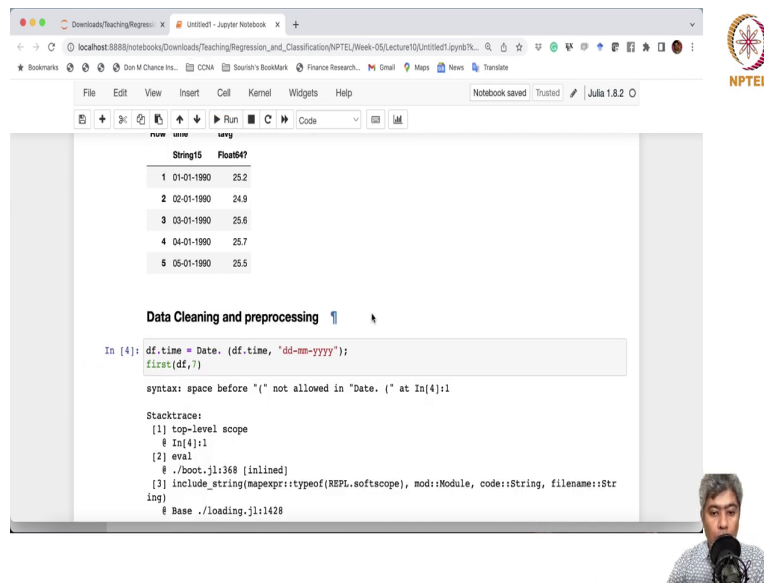
Out[3]: 5x2 DataFrame
  Row time      tavg
String15  Float64?
1  01-01-1990  25.2
2  02-01-1990  24.9
3  03-01-1990  25.6
4  04-01-1990  25.7
5  05-01-1990  25.5
```

We will we just want to work with average temperature and the date. So, we will keep these two columns and so that you know it will become we do not have to worry about the other columns. Ok.

So, what we will do, we will just write df colon and we will have time comma taverage v g and we do not want to print the entire column. So, we will just say first print the first five columns. So, now it has the printed the first five columns. Now, df has only two columns time and the taverage. Ok.

Now, we will do some data cleaning and data pre processing. So, data cleaning and pre processing. So, ok.

(Refer Slide Time: 05:07)



The screenshot shows a Jupyter Notebook interface. At the top, there's a browser address bar with the URL `localhost:8888/notebooks/Downloads/Teaching/Regression_and_Classification/NPTEL/Week-05/Lecture10/Untitled1.ipynb?...`. The notebook title is "Untitled1 - Jupyter Notebook". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The main content area displays a table with two columns: "String15" and "Float64".

String15	Float64
1 01-01-1990	25.2
2 02-01-1990	24.9
3 03-01-1990	25.6
4 04-01-1990	25.7
5 05-01-1990	25.5

Below the table, there's a section titled "Data Cleaning and preprocessing". A code cell contains the following code:

```
In [4]: df.time = Date.(df.time, "dd-mm-yyyy");  
first(df,7)
```

The output of the code cell shows a syntax error:

```
syntax: space before "(" not allowed in "Date. (" at In[4]:1
```

Below the error message, a stacktrace is displayed:

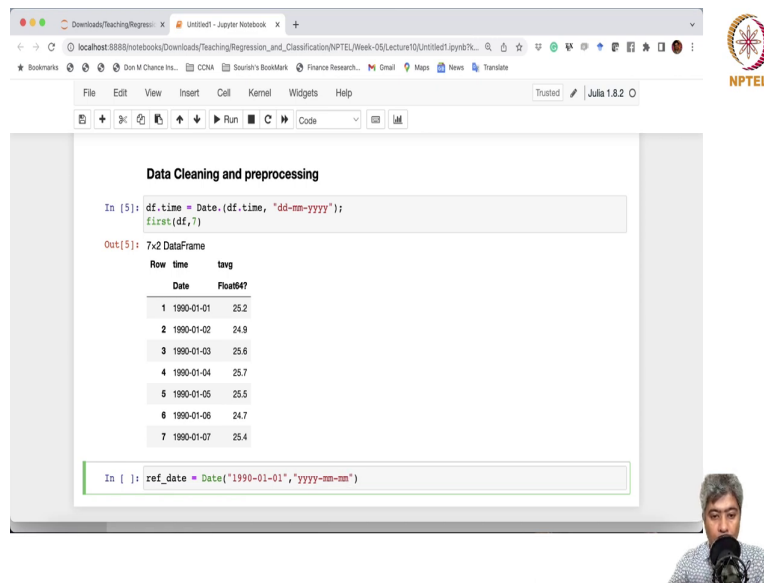
```
Stacktrace:  
[1] top-level scope  
# In[4]:1  
[2] eval  
# ./boot.jl:368 [inlined]  
[3] include_string(mapexpr::typeof(REPL.softscope), mod::Module, code::String, filename::String)  
# Base ./loading.jl:1428
```

In the bottom right corner of the notebook, there is a small video feed of a person speaking into a microphone.

So, the first thing we will do, we will take this string and we will just convert it into date column. So, `df.time`, we from `df` I am just expecting the time dot `df.time` comma. The format that we have is `dd mm y yyyy` and obviously, I do not want to print the whole thing.

If you do not give this semicolon, it will just print the whole thing. If you go up, you see, I have not put that semicolon. So, pretty much it printed a big chunk. So, I do not want to print the big chunk. If I want to just say, ok, print the first 7 columns, ok. It has something related to this maybe this will work. Yeah, ok.

(Refer Slide Time: 06:40)



The screenshot shows a Jupyter Notebook interface with the following content:

```

Data Cleaning and preprocessing

In [5]: df.time = Date.(df.time, "dd-mm-yyyy");
        first(df,7)

Out[5]: 7x2 DataFrame
         Row time      tagg
         Date      Float64?
1  1990-01-01      25.2
2  1990-01-02      24.9
3  1990-01-03      25.6
4  1990-01-04      25.7
5  1990-01-05      25.5
6  1990-01-06      24.7
7  1990-01-07      25.4

In [ ]: ref_date = Date("1990-01-01", "yyyy-mm-mm")

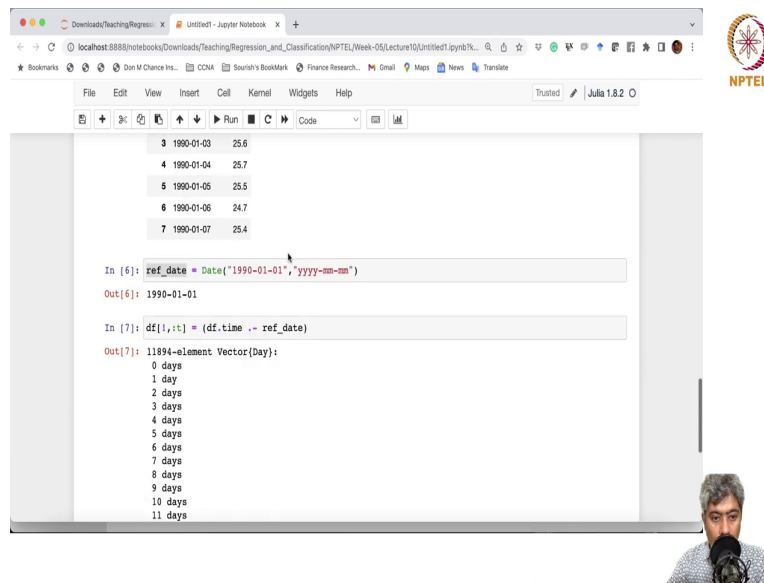
```

The output shows a DataFrame with 7 rows and 2 columns: 'time' (Date) and 'tagg' (Float64). The dates range from 1990-01-01 to 1990-01-07, and the 'tagg' values are 25.2, 24.9, 25.6, 25.7, 25.5, 24.7, and 25.4 respectively.

So, there was a unnecessary space was there. Anyway, so, now, we can see that the string is now being fixed to convert it into Date. Ok. Now, what I want is, I want to use this as a reference date and from this reference date, how many days of data we have, say, if it is zero point 5th January from 1st January, it is a five days so, I want to create the column with 1, 2, 3, 4 like that. Ok.

So, first what I will do, I will create a write a reference date, reference date, which is simpler date and 1990, 01, 01, and the format is. So, I am giving a string and then I am telling ok boss, this is the format is this. Ok.

(Refer Slide Time: 07:59)



```
3 1990-01-03 25.6
4 1990-01-04 25.7
5 1990-01-05 25.5
6 1990-01-06 24.7
7 1990-01-07 25.4

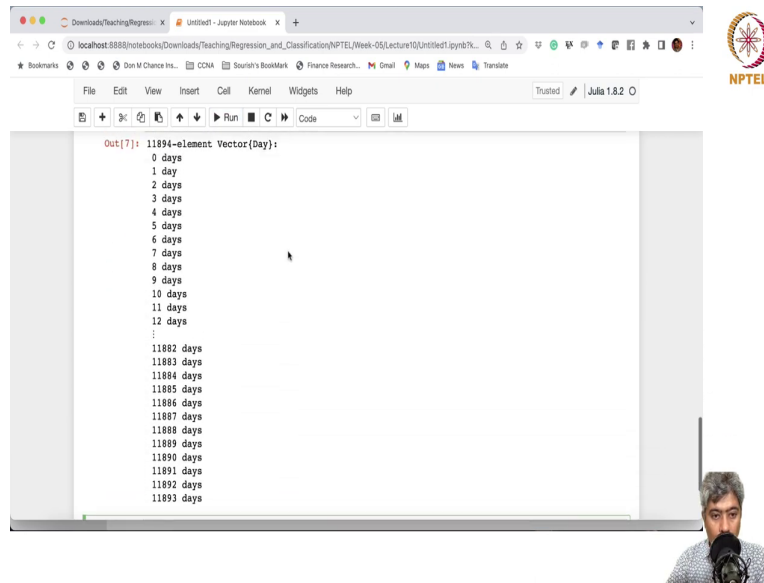
In [6]: ref_date = Date("1990-01-01", "yyyy-mm-dd")
Out[6]: 1990-01-01

In [7]: df[:, :t] = (df.time .- ref_date)
Out[7]: 11894-element Vector{Day}:
 0 days
 1 day
 2 days
 3 days
 4 days
 5 days
 6 days
 7 days
 8 days
 9 days
10 days
11 days
```

So, this is the, I have defining a variable and then I am saying, I am going to say that, create a column, df colon, exclamation mark colon t. And, in this, you create a column where you just subtract time dot minus this reference date. Put it in a ok yeah.



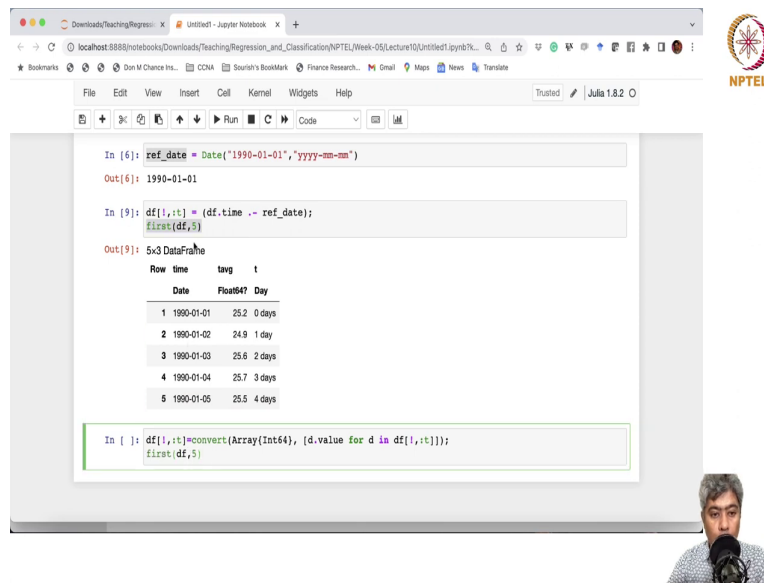
(Refer Slide Time: 08:46)



The image shows a Jupyter Notebook interface with a browser window. The browser address bar shows a local host path. The notebook's menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The status bar indicates 'Trusted' and 'Julia 1.8.2'. The main content area displays the output of a code cell, which is a 11894-element vector of days. The output is truncated, showing the first 13 elements and the last 7 elements. The first 13 elements are: 0 days, 1 day, 2 days, 3 days, 4 days, 5 days, 6 days, 7 days, 8 days, 9 days, 10 days, 11 days, and 12 days. The last 7 elements are: 11882 days, 11883 days, 11884 days, 11885 days, 11886 days, 11887 days, and 11888 days. The output is followed by a vertical ellipsis and another vertical ellipsis. A small video inset of a person speaking is visible in the bottom right corner of the notebook interface.

```
Out[7]: 11894-element Vector{Day}:  
 0 days  
 1 day  
 2 days  
 3 days  
 4 days  
 5 days  
 6 days  
 7 days  
 8 days  
 9 days  
10 days  
11 days  
12 days  
 ⋮  
11882 days  
11883 days  
11884 days  
11885 days  
11886 days  
11887 days  
11888 days  
⋮  
⋮
```

(Refer Slide Time: 08:49)



```
In [6]: ref_date = Date("1990-01-01", "yyyy-mm-dd")
Out[6]: 1990-01-01

In [9]: df[:, :t] = (df.time .- ref_date);
        first(df, 5)

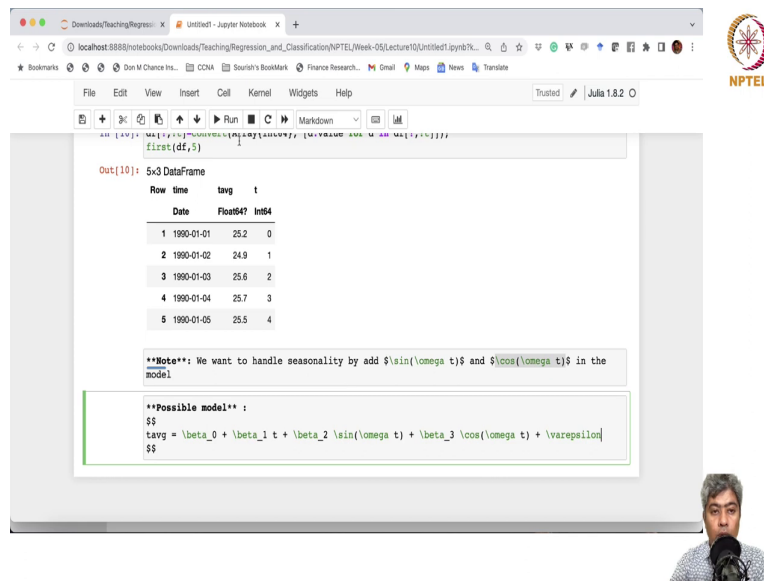
Out[9]: 5x3 DataFrame
         Row time      tavg      t
         Date      Float64 Day
  1  1990-01-01      25.2  0 days
  2  1990-01-02      24.9  1 day
  3  1990-01-03      25.6  2 days
  4  1990-01-04      25.7  3 days
  5  1990-01-05      25.5  4 days

In [ ]: df[:, :t] = convert(Array{Int64}, [d.value for d in df[:, :t]]);
        first(df, 5)
```

So, I will just put it in this and then if you just essentially print the first few columns, maybe five columns. So, you can see that it has created a new column with 0 day, 1 day, 2 day, 3 day like this. But this is a day column, I want to convert it into integer because if I end of the day, I want to supply this t as a, in a as a part of my model because I want to check if over time, over time these function, this average temperature is increasing or not. So, I want to model average as a function of t. So, that is what I want to do.

So, df colon t, sorry, no, I should have this comma exclamation mark colon t. So, I have to convert this, convert this array to integer 64, int 64, comma, d dot value. For d in df exclamation mark comma colon t. And, I want to just Run this guy.

(Refer Slide Time: 11:04)



Out[10]: 5x3 DataFrame

Row	time	tavg	t
1	1990-01-01	25.2	0
2	1990-01-02	24.9	1
3	1990-01-03	25.6	2
4	1990-01-04	25.7	3
5	1990-01-05	25.5	4

**Note**: We want to handle seasonality by add  $\sin(\omega t)$  and  $\cos(\omega t)$  in the model

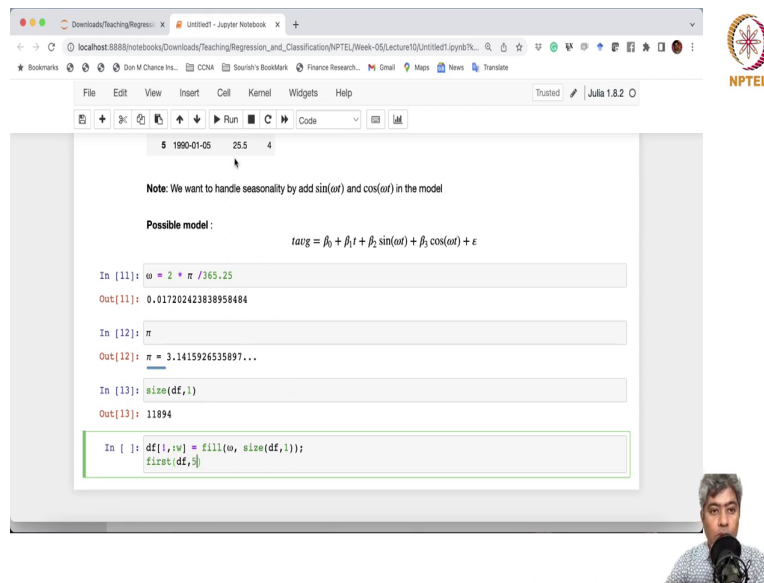
**Possible model** :

$$tavg = \beta_0 + \beta_1 t + \beta_2 \sin(\omega t) + \beta_3 \cos(\omega t) + \epsilon$$

So, now you see the initial, it was a day and now it is being converted into integer 64. So, eventually I want to model t average as a function of t. So, I cannot, and in the model, I cannot give it as a day, it cannot handle day in the model. So, I have to give it as either integer or a float. So, that is why I need this column as a integer. I have I can define it as a Float also actually. Technically, I can define it as a Float. No problem.

Next to handle seasonality, I need, so, it is a daily data. I want to handle seasonality. Let me just put a note here. So, note, we want to we want to handle seasonality. So, so, by adding  $\sin \omega t$  and  $\cos \omega t$  in the model, right.

(Refer Slide Time: 12:58)



The screenshot shows a Jupyter Notebook window with the following content:

5 1990-01-05 25.5 4

Note: We want to handle seasonality by add  $\sin(\omega t)$  and  $\cos(\omega t)$  in the model

Possible model:

$$tavg = \beta_0 + \beta_1 t + \beta_2 \sin(\omega t) + \beta_3 \cos(\omega t) + \epsilon$$

```
In [11]: ω = 2 * π / 365.25
Out[11]: 0.017202423838958484

In [12]: π
Out[12]: π = 3.1415926535897...

In [13]: size(df,1)
Out[13]: 11894

In [ ]: df[1,:v] = fill(ω, size(df,1));
        fitret = fit(df, f)
```

The interface includes a browser address bar, a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), and a toolbar with icons for file operations and execution. The NPTEL logo is visible in the top right corner.

So, we want to add  $\sin \omega t$   $\cos \omega t$  in our model. So, you can ask me what kind of model we have, we want to fit possible model could be like, a simple model could be possible model. Possible model could be easily say  $t$  average equal to  $\beta_0$  plus, so,  $t$  average is already there.  $t$  say and  $t$  we want to put, so,  $\beta_1 t$  plus  $\beta_2$ .

So, we just want to take  $\sin \omega t$  to model the seasonality, then  $\beta_3 \cos \omega t$  right plus  $\epsilon$ . So, this is the model that we want to fit. Suppose, this is the basic model that we want to fit. Now, we have  $t$  column, but we have to have a  $\omega$  column as well because  $\sin$   $\cos$  and the formula term can handle it, but we need to provide a  $\omega$ . So, let us define  $\omega$ , what will be our  $\omega$ .

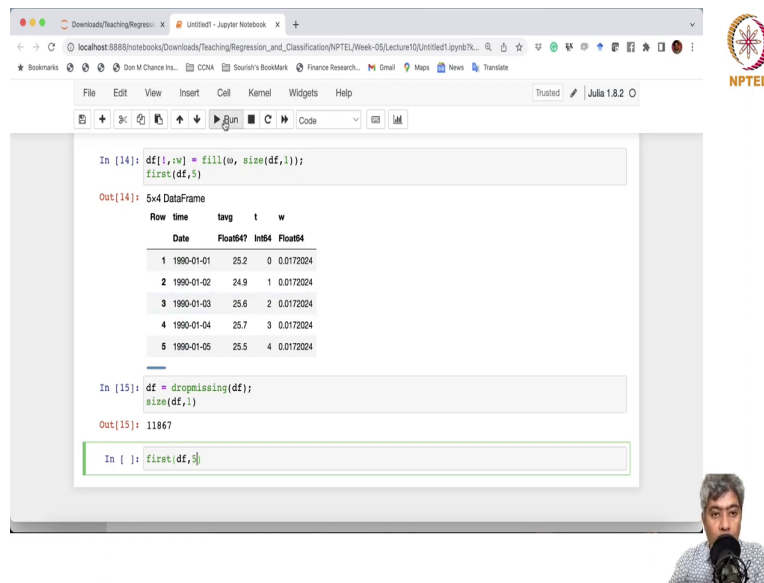
So, this is the advantage of you know, Julia where you can, because it allows you idea, allows you to be or you unicode. So, you can use unicode for defining variable. Now, so, I am saying

that  $2\pi$  by  $f$ ,  $f$  is my frequency of the day. So, this is a year when a year, I have 365.25. So, technically on we typically take year to be 365 days, but we know in order to handle the leap year, I am giving it as a 365.25. So, that you will not have to, we do not have to worry about the leap year thing. So, it will be automatically taken care of.

So, let us write this. So, this is the value of  $\omega$ . If you want to check, in fact, what is  $\pi$ ?  $\pi$  is 3.141. So, this is a all default setup, we just defined it. Now, we want to add a column of  $\omega$  df comma colon so,  $w$  stands for  $\omega$ . Fill with  $\omega$  and how many the size, whatever the size of the data frame, the first. So, let me just what is size of the data frame? Size of the data frame is 11894.

So, I am just saying that create a random, create a vector of 11894, fill it with the value of  $\omega$  and add it as a column named as  $\omega$ . So, I want to print the first column, first five rows. Ok.

(Refer Slide Time: 17:19)



```
In [14]: df[:,~v] = fill(ω, size(df,1));
         first(df,5)

Out[14]: 5x4 DataFrame
          Row time      tang      t      w
          Date      Float64 Int64 Float64
1  1990-01-01      25.2      0  0.0172024
2  1990-01-02      24.9      1  0.0172024
3  1990-01-03      25.6      2  0.0172024
4  1990-01-04      25.7      3  0.0172024
5  1990-01-05      25.5      4  0.0172024

In [15]: df = dropmissing(df);
         size(df,1)

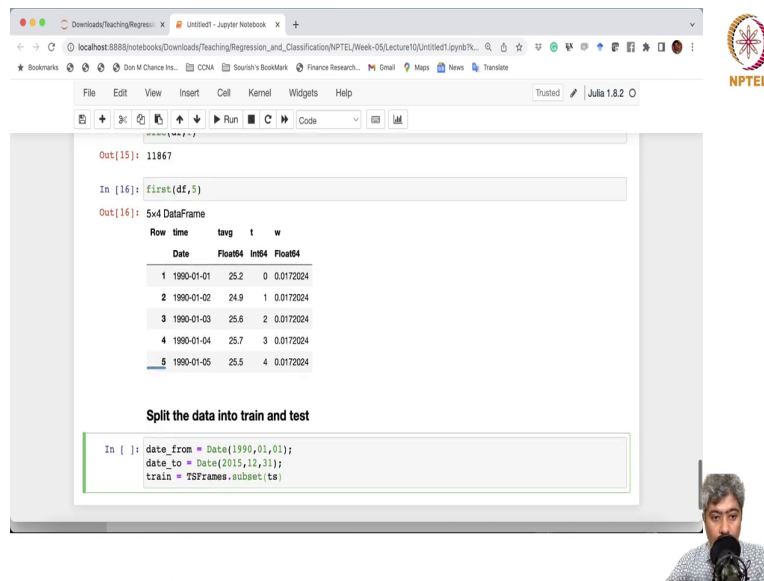
Out[15]: 11867

In [ ]: first(df,3)
```

So, now, you can see that we have a column with all values to be omega, the constant values and t values. Alright, so, I think now we are almost done. But, remember that in the t average, still we have this question mark means there are some missing values are there. So, let us drop those missing values because currently we are going to use a package called CR Rao which handles the fits, the simple linear regression, but unfortunately it cannot handles the missing value at this moment.

So, so, we are going to drop the missing df and print the size. So, you see before drop we have 11,894 rows after dropping 11,867. So, about 30 odd values were missing, but now it is better. We do not have to worry about the only so, we drop if we drop those with missing we do not have to worry.

(Refer Slide Time: 18:54)



The screenshot shows a Jupyter Notebook window with the following content:

```
Out[15]: 11867
```

```
In [16]: first(df,5)
```

```
Out[16]: 5x4 DataFrame
```

Row	time	tavg	t	w
1	1990-01-01	25.2	0	0.0172024
2	1990-01-02	24.9	1	0.0172024
3	1990-01-03	25.6	2	0.0172024
4	1990-01-04	25.7	3	0.0172024
5	1990-01-05	25.5	4	0.0172024

**Split the data into train and test**

```
In [ ]: date_from = Date{1990,01,01};  
date_to = Date{2015,12,31};  
train = TSFrames.subset(ts);
```

The NPTEL logo is visible in the top right corner of the notebook interface.

So, if we just print the first five rows and since now there is no missing. So, the question mark is gone. There is no missing now. So, we have about 11867 data points. And so, now what we have to do, we have to split the data into train and test, alright. So, I will just make a comment. Split the data into train and test. Now, the problem with this particular data is it is a time series data. We cannot randomly shuffle the rows and choose the data. So, how what to do?

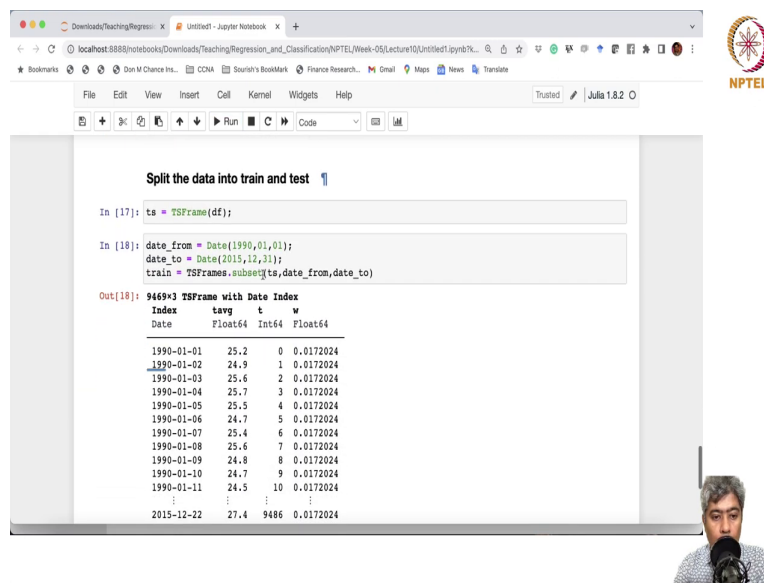
In this case, what we will do is from 1990 to 2015, we will use as a the older data we will use as a training data and the newer data we will use as a test data. So, first we will say date from, we have to define from which date. So, Date is 1990, 01, 01, to date to 2015, 01, 01. Sorry, we should take 12, 31. I do not want to print here and my train data will be. So, I am going to

use the TSFrames; TSFrames, a subset function. It will take a time series data frame and subset the data by dates and that is it pretty much.

So, ok, I think, I do not know, I have to first define the data frame as a TS and then TS frame.

So, ok, I will just do that before, in fact.

(Refer Slide Time: 21:32)




The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [17]: ts = TSFrame(df);
```

```
In [18]: date_from = Date(1990,01,01);
date_to = Date(2015,12,31);
train = TSFrames.subset(ts,date_from,date_to)
```

```
Out[18]: 9469x3 TSFrame with Date Index
```

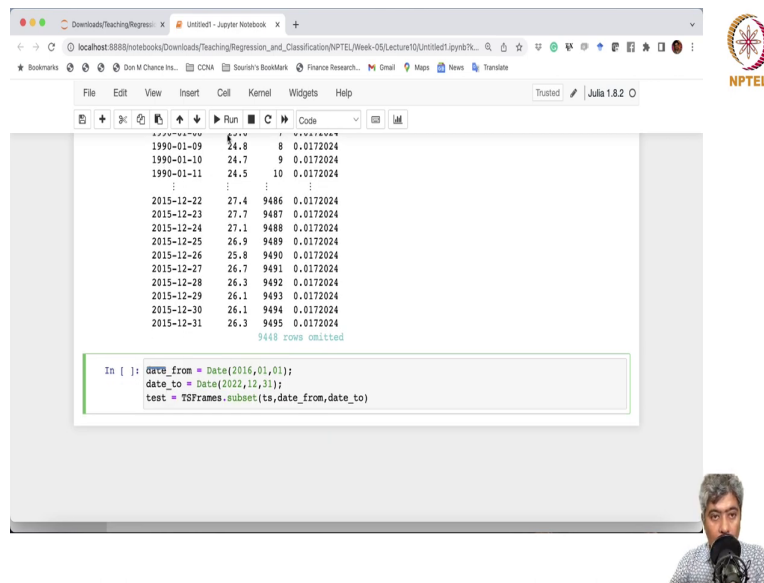
Index	tavg	t	w
Date	Float64	Int64	Float64
1990-01-01	25.2	0	0.0172024
1990-01-02	24.9	1	0.0172024
1990-01-03	25.6	2	0.0172024
1990-01-04	25.7	3	0.0172024
1990-01-05	25.5	4	0.0172024
1990-01-06	24.7	5	0.0172024
1990-01-07	25.4	6	0.0172024
1990-01-08	25.6	7	0.0172024
1990-01-09	24.8	8	0.0172024
1990-01-10	24.7	9	0.0172024
1990-01-11	24.5	10	0.0172024
⋮	⋮	⋮	⋮
2015-12-22	27.4	9486	0.0172024



So, I am just going to define it as a TS frame. I have not done that. So, it will be defined as TS frame and then I have to say that, ok, this is from date, this is to date. Ok.



(Refer Slide Time: 22:03)



The screenshot shows a Jupyter Notebook interface with a browser window at the top. The notebook contains a table of data and a code cell. The table displays a time series dataset with columns for date, value, and another value. The code cell shows the following Python code:

```
In [ ]: date_from = Date{2016,01,01};  
date_to = Date{2022,12,31};  
test = TSFrames.subset(ts,date_from,date_to)
```

The NPTEL logo is visible in the top right corner of the notebook interface.

So, you can see this is a TS frame with the index date as the index and rest of them are exactly same as it is and from starting from first January 1990 to 31st December 2015. So, 9448 rows are being included as training data.

Now, we are going to define the test data. So, test data, I am going to just copy and paste it here, but here instead of Date from will be from 2016, 01, 01 to 2022, ok, and TS and this will be test data, ok.

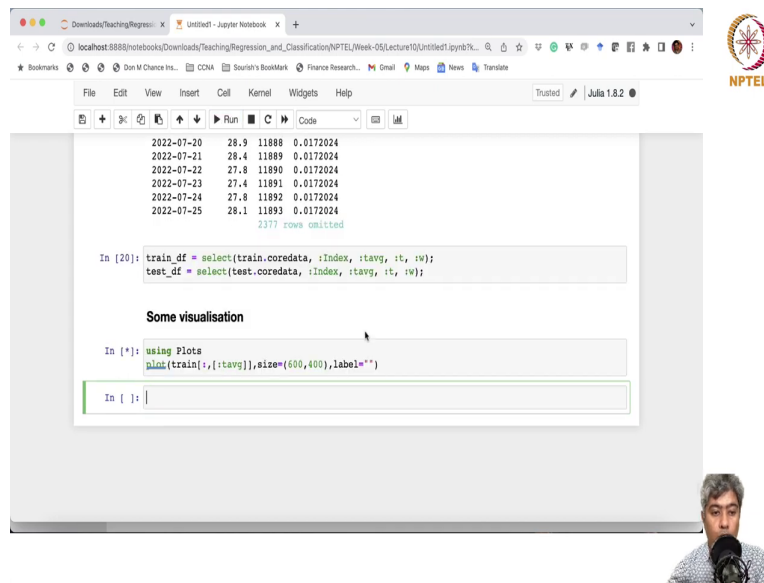
(Refer Slide Time: 22:54)

```
In [19]: date_from = Date{2016,01,01};
date_to = Date{2022,12,31};
test = TSFrames.subset(ts,date_from,date_to)

Out[19]: 239x3 TSFrame with Date Index
Index      tavg      t      w
Date      Float64  Int64  Float64
-----
2016-01-01  26.3    9496  0.0172024
2016-01-02  25.8    9497  0.0172024
2016-01-03  25.9    9498  0.0172024
2016-01-04  26.6    9499  0.0172024
2016-01-05  26.1    9500  0.0172024
2016-01-06  26.4    9501  0.0172024
2016-01-07  26.0    9502  0.0172024
2016-01-08  25.9    9503  0.0172024
2016-01-09  25.2    9504  0.0172024
2016-01-10  25.1    9505  0.0172024
2016-01-11  25.3    9506  0.0172024
⋮          ⋮          ⋮          ⋮
2022-07-16  29.9   11884  0.0172024
2022-07-17  28.4   11885  0.0172024
2022-07-18  28.2   11886  0.0172024
2022-07-19  28.9   11887  0.0172024
2022-07-20  28.9   11888  0.0172024
2022-07-21  28.4   11889  0.0172024
2022-07-22  27.8   11890  0.0172024
2022-07-23  27.4   11891  0.0172024
```

Now, if you Run this, you can see that it is starting from first January 2016.

(Refer Slide Time: 23:00)



The screenshot shows a Jupyter Notebook window with the following content:

```
2022-07-20 28.9 11888 0.0172024
2022-07-21 28.4 11889 0.0172024
2022-07-22 27.8 11890 0.0172024
2022-07-23 27.4 11891 0.0172024
2022-07-24 27.8 11892 0.0172024
2022-07-25 28.1 11893 0.0172024
2377 rows omitted
```

```
In [20]: train_df = select(train.coredata, :Index, :avg, :t, :w);
test_df = select(test.coredata, :Index, :avg, :t, :w);
```

Some visualisation

```
In [*]: using Plots
plot(train[:, :avg], size=(600, 400), label="")
```

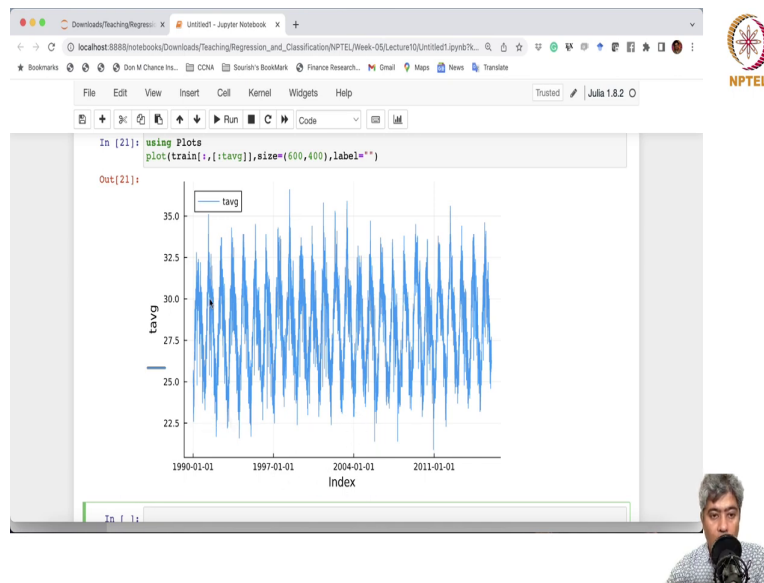
```
In [ ]: |
```

The interface includes a browser address bar, a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), and a toolbar with icons for file operations and execution. The NPTEL logo is visible in the top right corner.

It is ending at 25th July 2020, though I gave a 13 so, it is just in that range, it will just whatever data, it will pick it up and give you as a TS frame. So, though it is as a TS frame now, we will just convert it into data frame and that is it then we are ready. Convert it into train df equal to select train dot core data colon index colon avg colon t colon w and then we should do the test data as well test df, ok. I think we are done.

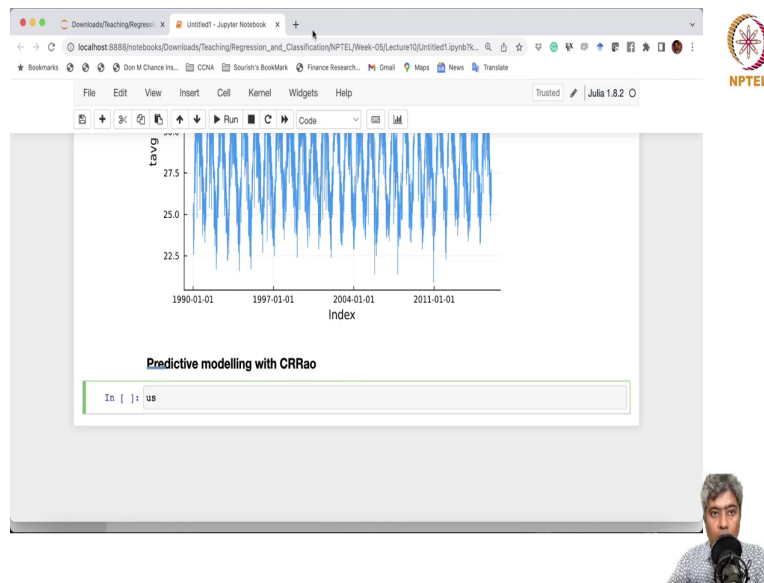
So, I think we are done. We do some visualization that will be, this is not a regular visualization, it requires lot of more thing, but I am just saying some visualization I am doing just to see the how it looks like, but not a detailed visualization. I am not going to do a so, using plots, I am just going to do a plot, simple plot of train colon comma avg size equal to 600, this is the width, this is the height, I do not want any label.

(Refer Slide Time: 25:28)



So, if you just run it, ok. So, over time it has, you can see that there is a clear sinusoidal behavior. So, let us try to model it, ok. Now, we want to start, do the predictive modeling with CR Rao package.

(Refer Slide Time: 25:53)



Now, we can ask predictive modeling with CRRao package. So, CR Rao is a package that we are developing in CMI, the CMI students and faculty are developing.

So, what we are going to do?

(Refer Slide Time: 26:22)

The screenshot displays the GitHub interface for the repository `xKDR/CR Rao.jl`. The repository is public and has 16 stars and 9 forks. The main content area shows a list of files and folders with their last update times:

File/Folder	Last Update
<code>.github</code>	3 weeks ago
<code>docs</code>	4 months ago
<code>src</code>	last month
<code>test</code>	last month
<code>.JuliaFormatter.toml</code>	8 months ago
<code>.gitignore</code>	last year
<code>LICENSE</code>	last year
<code>Project.toml</code>	last month
<code>README.md</code>	2 weeks ago

The right sidebar contains the following sections:

- About:** No description, website, or topics provided.
- Releases:** v0.1.0 (Latest) on Jan 12.
- Packages:** No packages published. Publish your first package.



(Refer Slide Time: 26:24)

The screenshot shows the GitHub repository page for CRRao. The main content area displays the README file with the following sections:

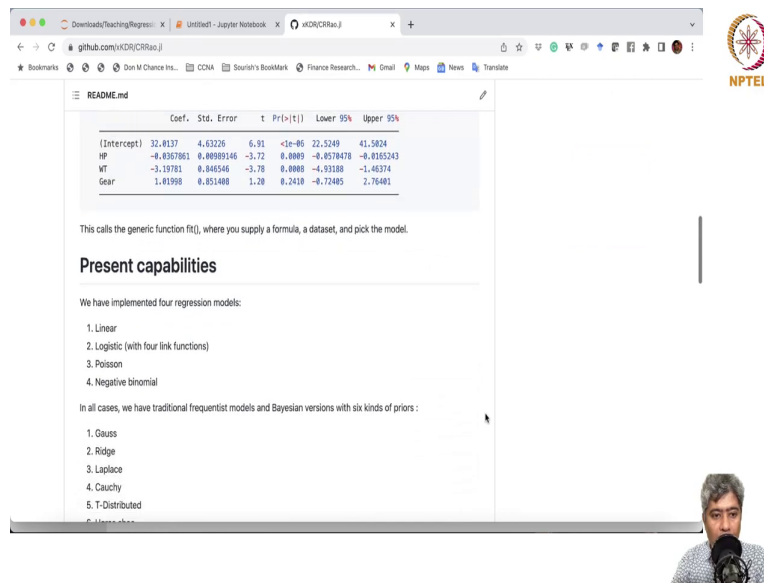
- To install:** A code block showing the installation command: `us.ing Pkg; Pkg.add(url = "https://github.com/xDR/CRRao.jl.git")`
- CRRao: A single API for diverse statistical models**
- A paragraph explaining that many statistical models can be estimated in Julia and that the CRRao package provides a simple and consistent API.
- An example of estimating linear regression:  $MPG = \beta_0 + \beta_1 HP + \beta_2 WT + \beta_3 Gear + \epsilon$
- A code block showing the Julia code for fitting a linear regression model: 

```
using CRRao, Rdatasets, StatsModels
df = dataset("datasets", "mtcars")
model = fit(@formula(MPG ~ HP + WT+Gear), df, LinearRegression())
model.fit
```

The right sidebar shows the repository's metadata, including contributors (codetalker7, sourish-cmi, ayushpatnaik98, ShouvikGhosh2048), environments (github-pages), and languages (Julia 100.0%).

And. so, I will give you the. So, here is the, it is a complete open-source development, we are doing it in GitHub. If you, there is a lot of detail that we are doing, we have given.

(Refer Slide Time: 26:34)



The screenshot shows a web browser window displaying the README file for the CRRAO package. The browser's address bar shows the URL `github.com/KDR/CRRAO`. The README content includes a table of regression coefficients and a list of capabilities.

	Coef.	Std. Error	t	Pr(> t )	Lower 95%	Upper 95%
(Intercept)	32.8137	4.63226	6.91	<1e-06	22.5249	41.5824
HP	-0.8567861	0.00999146	-3.72	0.0009	-0.8578478	-0.8165243
WT	-3.19781	0.046546	-3.78	0.0008	-4.93188	-1.46374
Gear	1.81998	0.851488	1.28	0.2410	-0.72485	2.76481

This calls the generic function `fit()`, where you supply a formula, a dataset, and pick the model.


### Present capabilities

We have implemented four regression models:

1. Linear
2. Logistic (with four link functions)
3. Poisson
4. Negative binomial

In all cases, we have traditional frequentist models and Bayesian versions with six kinds of priors:

1. Gauss
2. Ridge
3. Laplace
4. Cauchy
5. T-Distributed
6. Gamma



So, CRRAO is a package that essentially all of us in based in CMI, well Ayush and other, Ayush is in the partly Bombay, partly Delhi.



(Refer Slide Time: 26:46)

The screenshot shows a web browser window displaying the README.md file for the CR Rao package. The browser's address bar shows the URL `github.com/xKDR/CR Rao.jl`. The README content includes:

- 6. Horse shoe**  
All these models are built out of foundations in the Julia package ecosystem, such as GLM.jl and Turing.jl. Here in CR Rao.jl, we are not building additional models; we are only building the scaffolding for the consistent API to a diverse array of models.  
The traditional frequentist models estimates MLE using the GLM.jl and the Bayesian models uses the Hamiltonian Monte Carlo (HMC) methods using Turing.jl.
- Future capabilities**
  1. Gaussian Process Regression
  2. Gaussian Process Classification
  3. Gaussian Process Time Series
  4. Linear Discriminant Analysis
  5. Hierarchical Bayesian Models
- Help us build this**
  - Please use CR Rao and tell us what is not good about it.
  - We have exploited Julia capabilities to make it convenient to build additional functionality within CR Rao, and for multiple developers to build new models.
  - We want to build out CR Rao into a simple and consistent approach to the statistical modelling workflow. Please help us plan and build this.

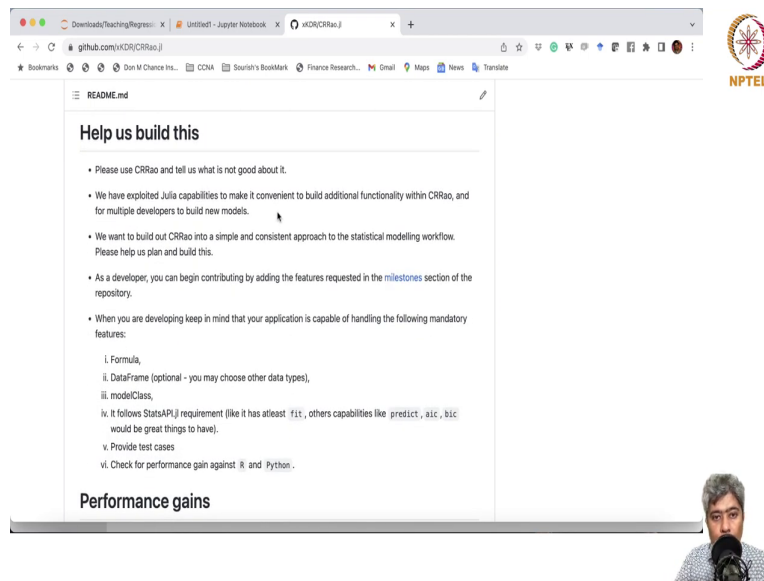
The NPTEL logo is visible in the top right corner of the browser window. A small video feed of a speaker is visible in the bottom right corner of the slide.

So, but we all are associated with CMI. And, these are the current capabilities, it can fit linear regression, logistic regression, Poisson regression, negative binomial regression. Whatever model we are going to do in this course, all these problems, all these models are now part of this package.

Now, in along with, you can implement some of the Bayesian models also, which is beyond this course, but at least you will know that, that if you want to fit a linear regression with Gauss prior or Laplace prior or Horse shoe prior, you can very easily do that. And, then automatically CR Rao will figure out whether you want to do a frequentist model or Bayesian model, automatically it will do that.

In future, we are trying to, you are trying to do, a you know, implement the mixed effect models, Gaussian process model, hierarchical Bayesian models. So, lot of, you know, development activity is happening.

(Refer Slide Time: 27:41)



The screenshot shows a web browser window displaying a GitHub README file for the CR Rao package. The browser's address bar shows the URL `github.com/vikOR/CR Rao.jl`. The README content includes a section titled "Help us build this" with a bulleted list of requests for contributors. Below this is a section titled "Performance gains" with a numbered list of mandatory features for developers. The NPTEL logo is visible in the top right corner of the browser window. A small video inset of a person speaking is located in the bottom right corner of the slide.

**Help us build this**

- Please use CR Rao and tell us what is not good about it.
- We have exploited Julia capabilities to make it convenient to build additional functionality within CR Rao, and for multiple developers to build new models.
- We want to build out CR Rao into a simple and consistent approach to the statistical modelling workflow. Please help us plan and build this.
- As a developer, you can begin contributing by adding the features requested in the [milestones](#) section of the repository.
- When you are developing keep in mind that your application is capable of handling the following mandatory features:
  - i. Formula,
  - ii. DataFrame (optional - you may choose other data types),
  - iii. modelClass,
  - iv. It follows StatsAPI.jl requirement (like it has atleast `fit`, others capabilities like `predict`, `aic`, `bic` would be great things to have).
  - v. Provide test cases
  - vi. Check for performance gain against R and Python.

**Performance gains**

So, you can please help us building this package.

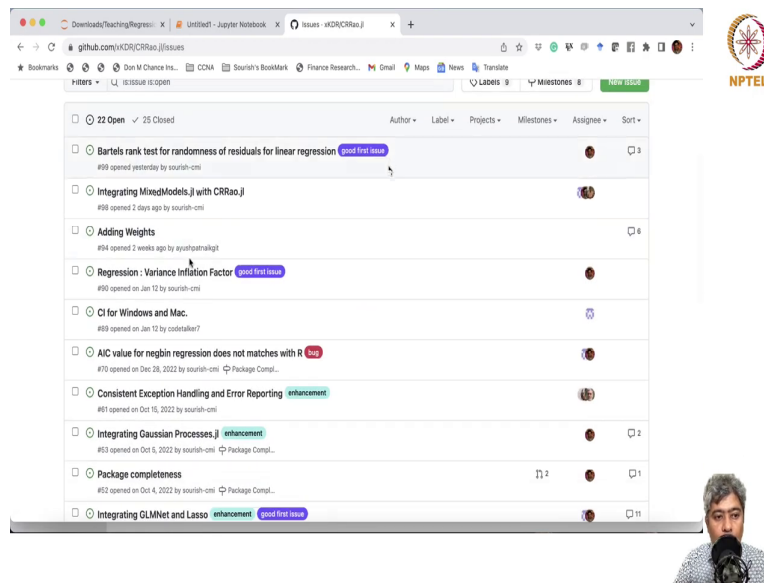
(Refer Slide Time: 28:05)

The screenshot displays the GitHub interface for the repository `xKDR/CR Rao.jl`. At the top, there is a navigation bar with options like 'Code', 'Issues (22)', 'Pull requests (14)', 'Discussions', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. Below this, a message reads: 'Label issues and pull requests for new contributors. Now, GitHub will help potential first-time contributors discover issues labeled with good first issue.' A filter bar shows 'is:issue is:open' and a 'New issue' button. The main content is a table of issues:

<input type="checkbox"/>	<input type="radio"/>	22 Open	25 Closed	Author	Label	Projects	Milestones	Assignee	Sort
<input type="checkbox"/>	<input type="radio"/>	<b>Bartels rank test for randomness of residuals for linear regression</b>	<b>good first issue</b>	#99 opened yesterday by sourish-cmi					3
<input type="checkbox"/>	<input type="radio"/>	<b>Integrating MixedModels.jl with CR Rao.jl</b>		#98 opened 2 days ago by sourish-cmi					
<input type="checkbox"/>	<input type="radio"/>	<b>Adding Weights</b>		#94 opened 2 weeks ago by ayushpathakgit					6
<input type="checkbox"/>	<input type="radio"/>	<b>Regression : Variance Inflation Factor</b>	<b>good first issue</b>	#90 opened on Jan 12 by sourish-cmi					
<input type="checkbox"/>	<input type="radio"/>	<b>CI for Windows and Mac.</b>		#82 opened on Jan 05 by sourish-cmi					

In the bottom right corner, there is a small video inset showing a person with a beard and glasses speaking into a microphone.

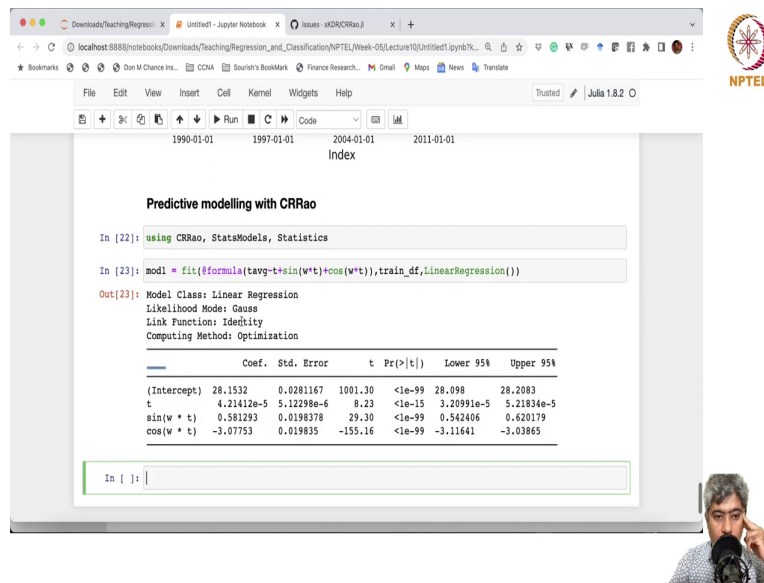
(Refer Slide Time: 28:07)



You can just, easiest thing will be like, you know, if you go to the Issues you know, there are some good first issues I have mentioned, you can try to implement those issues. Then, or you can raise the issue, if you can use this and you can raise issues to, you know, report bug. So, that will be this, how that is how you can help us with this package.

If you like this package, then please hit the start button. It is like that will help us encourage us to do more about this package. So, to develop more about this package, if you do not like, then some feature of the model package, please write in the issues that, ok, I do not like this package or issues or this feature, can you make this feature available or something like that. And, definitely, we will try to do that. If you can help developing, that will be even awesome.

(Refer Slide Time: 29:06)



1990-01-01 1997-01-01 2004-01-01 2011-01-01  
Index

### Predictive modelling with CRRao


```
In [22]: using CRRao, StatsModels, Statistics
```

```
In [23]: mod1 = fit(@formula(tavg-t+sin(w*t)+cos(w*t)),train_df,LinearRegression())
```

Out[23]: Model Class: Linear Regression  
Likelihood Mode: Gauss  
Link Function: Identity  
Computing Method: Optimization

	Coef.	Std. Error	t	Pr(> t )	Lower 95%	Upper 95%
(Intercept)	28.1532	0.0281167	1001.30	<1e-99	28.098	28.2083
t	4.21412e-5	5.12298e-6	8.23	<1e-15	3.20991e-5	5.21834e-5
sin(w * t)	0.581293	0.0198378	29.30	<1e-99	0.542406	0.620179
cos(w * t)	-3.07753	0.019835	-155.16	<1e-99	-3.11641	-3.03865

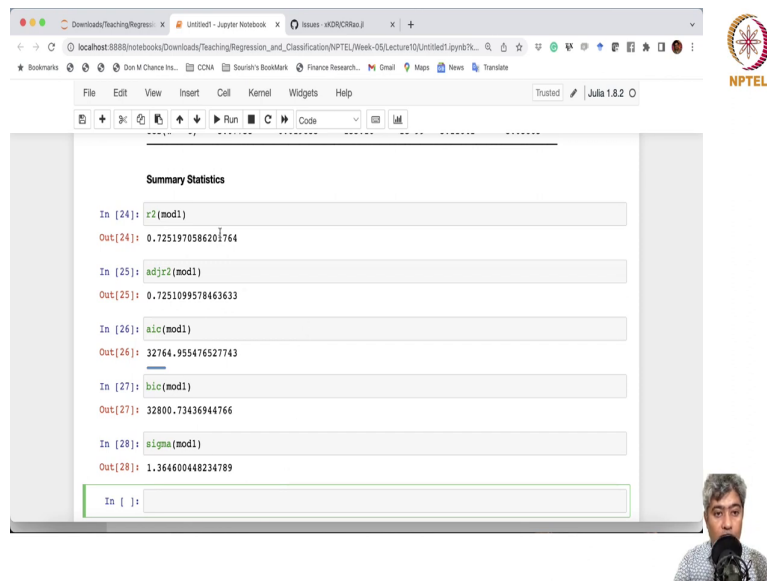
In [ ]:



So, I am going to call using CRRao and some more things, stats models. So, I am going to, from stats model, I am going to use formula and statistics, ok. So, let me call this, ok. Now, so, the first model perhaps, we can try this particular model, we have discussed here, maybe. Ok. So, why not?

So, the first model is model 1, fit, at the rate formula,  $tavg$   $t$  plus  $\sin w$  times  $t$  plus  $\cos t$  times  $w$  times  $t$ . This is the formula, then train; we have to provide the dataset name, train data frame and then, we have to say, which model class. In this case, it is linear regression. So, if you want to fit a logistic regression, then you have to say logistic regression. So, like this only. Ok. So, let me Run and here is the fitting of the model. Ok.

(Refer Slide Time: 31:20)



```
Summary Statistics

In [24]: r2(mod1)
Out[24]: 0.7251970586201764

In [25]: adjr2(mod1)
Out[25]: 0.7251099578463633

In [26]: aic(mod1)
Out[26]: 32764.955476527743

In [27]: bic(mod1)
Out[27]: 32800.73436944766

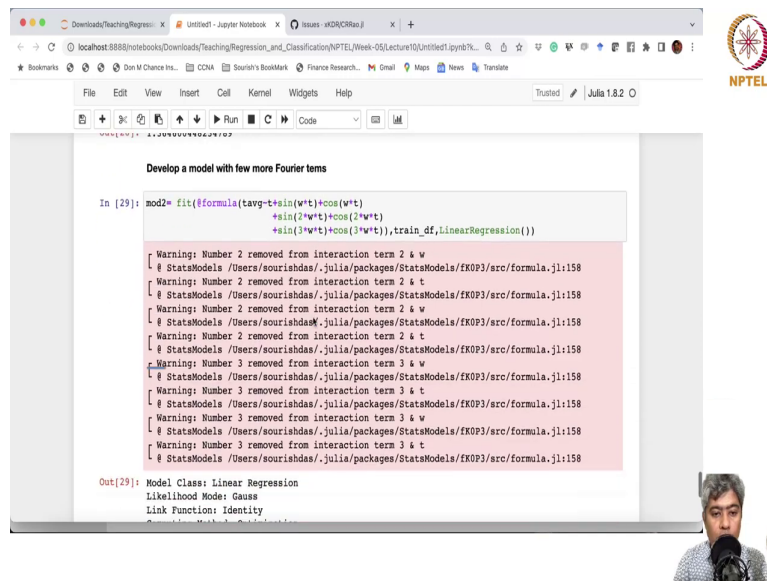
In [28]: sigma(mod1)
Out[28]: 1.364600448234789

In [ ]:
```

So, some summary statistics, we can pull. Summary Statistics, maybe  $r^2$  of `mod1` is the  $r$  square, and then adjusted  $r^2$ , I think, `adjr2(mod1)` will give me adjusted  $r$  square. So, I can then `aic` will give me the `aic` of the model, then `bic` will give me the `bic` of the model. Then, if I call `sigma`, that will give me the standard deviation or residual standard error of the model. But these are all in sample summary statistics. Ok.

Now, if we try, we will do out sample statistics, but we want to bring another model, maybe a little bit more complex model, few more Fourier terms.

(Refer Slide Time: 32:42)



Develop a model with few more Fourier terms

```
In [29]: mod2= fit(#formula(tavg~t*asin(w*t)+cos(w*t)
                +sin(2*w*t)+cos(2*w*t)
                +sin(3*w*t)+cos(3*w*t)),train_df,LinearRegression())
```

[ Warning: Number 2 removed from interaction term 2 & w  
# StatsModels /Users/sourishdas/.julia/packages/StatsModels/EXOP3/src/formula.jl:158  
Warning: Number 2 removed from interaction term 2 & t  
# StatsModels /Users/sourishdas/.julia/packages/StatsModels/EXOP3/src/formula.jl:158  
Warning: Number 2 removed from interaction term 2 & w  
# StatsModels /Users/sourishdas/.julia/packages/StatsModels/EXOP3/src/formula.jl:158  
Warning: Number 2 removed from interaction term 2 & t  
# StatsModels /Users/sourishdas/.julia/packages/StatsModels/EXOP3/src/formula.jl:158  
Warning: Number 3 removed from interaction term 3 & w  
# StatsModels /Users/sourishdas/.julia/packages/StatsModels/EXOP3/src/formula.jl:158  
Warning: Number 3 removed from interaction term 3 & t  
# StatsModels /Users/sourishdas/.julia/packages/StatsModels/EXOP3/src/formula.jl:158  
Warning: Number 3 removed from interaction term 3 & w  
# StatsModels /Users/sourishdas/.julia/packages/StatsModels/EXOP3/src/formula.jl:158  
Warning: Number 3 removed from interaction term 3 & t  
# StatsModels /Users/sourishdas/.julia/packages/StatsModels/EXOP3/src/formula.jl:158

```
Out[29]: Model Class: Linear Regression
Likelihood Mode: Gauss
Link Function: Identity
```

Develop a model with few more Fourier terms, develop a model. Develop a model with few more Fourier terms. Ok. And then, this is the second model. So, and what I am going to do, I am just going to copy this, and just add this here and say,  $\sin 2 \omega t \cos 2 \omega t$ . Maybe one more Fourier,  $3 \omega t \cos 3 \omega t$ . Ok.  $3 \omega t \cos 3 \omega t$  and, else, everything is same.

(Refer Slide Time: 33:53)

Likelihood Mode: Gauss  
Link Function: Identity  
Computing Method: Optimization

	Coef.	Std. Error	t	Pr(> t )	Lower 95%	Upper 95%
(Intercept)	28.1622	0.0255043	1104.21	<1e-99	28.1122	28.2122
t	4.03369e-5	4.64717e-6	8.68	<1e-17	3.12274e-5	4.94463e-5
sin(w * t)	0.581518	0.0179916	32.32	<1e-99	0.546251	0.616786
cos(w * t)	-3.07823	0.0179891	-171.12	<1e-99	-3.11349	-3.04297
sin(2 * w * t)	-0.503292	0.0179906	-27.98	<1e-99	-0.538558	-0.468027
cos(2 * w * t)	-0.5997	0.0179852	-33.34	<1e-99	-0.634955	-0.564446
sin(3 * w * t)	0.189473	0.0179928	10.53	<1e-25	0.154204	0.224743
cos(3 * w * t)	0.113289	0.0179811	6.30	<1e-09	0.0780419	0.148536

```
In [30]: [adjr2(mod1), adjr2(mod2)]  
Out[30]: 2-element Vector{Float64}:  
 0.7251099578463633  
 0.7738933046307549
```

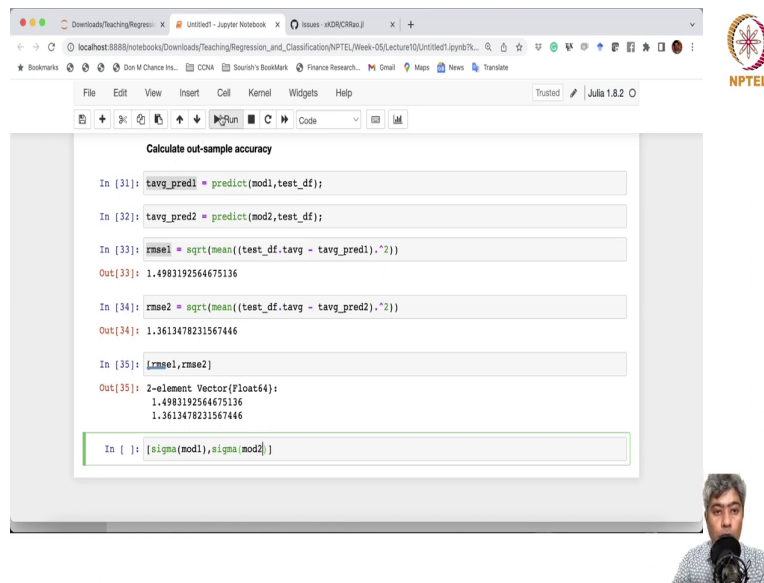
**\*\*Calculate out-sample accuracy\*\***

So, let us Run. It gives some warning, but I think all of them are significant. The time do have effect and all these Fourier do have effect. And what we can do? We can just compare the, maybe adjusted r square of model 1 and adjusted r square of model 2 as a vector, let us see.

So, first model has a 72 percent accuracy; when we add second and third, these two Fourier terms, it jump up to 77 percent. But these are both of them are in sample accuracy. So, we will calculate the out of the sample accuracy. So, let me just calculate out sample accuracy.



(Refer Slide Time: 35:01)



```
Calculate out-sample accuracy

In [31]: tavg_pred1 = predict(mod1, test_df);
In [32]: tavg_pred2 = predict(mod2, test_df);
In [33]: rmse1 = sqrt(mean((test_df.tavg - tavg_pred1).^2))
Out[33]: 1.4983192564675136

In [34]: rmse2 = sqrt(mean((test_df.tavg - tavg_pred2).^2))
Out[34]: 1.3613478231567446

In [35]: [rmse1, rmse2]
Out[35]: 2-element Vector{Float64}:
 1.4983192564675136
 1.3613478231567446

In [ ]: [sigma(mod1), sigma(mod2)]
```

tavg predict 1 equals to. So, what I will do? I will just predict mod1, and I have to give test data frame test data frame ok. And, maybe I will just calculate the second one with second model. So, this gives me the predicted value of averaging the test data frame. Now, what I am going to do, I am going to calculate the root mean square error, root mean square error.

So, rmse1 equal to square root mean of first from test, oops, sorry about that, test data frame, you calculate t average just for it, extract t average, ok minus, you just take the first prediction from the first model and yeah, square. So, this is a broadcasting operator, and then it should work. And, then let me just Run it. So, you have 1.49 and then, this is the second one, rmse of the second one. So, this is 1.36.

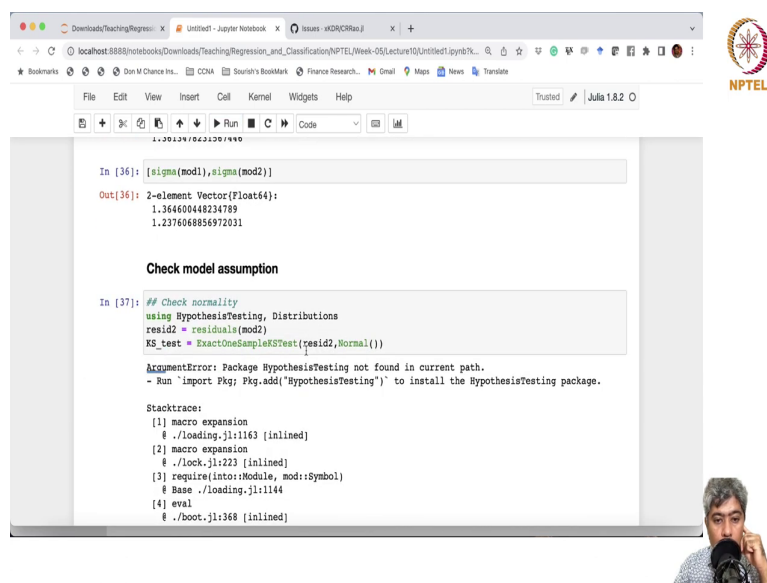
So, now what we can do? We can just compare the rmse1 and rmse2, which is in sample, sorry, out of the sample rmse1 and out of the sample rmse2, the second one is clearly much

better. And, then also we can calculate the sigma, which is in sort of in sample rmse mod1 and sigma mod2.

So, clearly in both cases rmse of course, out of the sample rmse slightly more than the in sample rmse, but this is definitely better. I mean, in second model is definitely better in this case. Our model accuracy is doing reasonably ok, about 77 – 8 percent accuracy we are getting; even if in the out sample probably we are getting little, maybe around 70 percent accuracy.

Now, still we have to do some model checking, model assumption checking. So, check the assumptions and at least if it is doing normality on going well or not.

(Refer Slide Time: 37:38)



```
In [36]: [sigma(mod1),sigma(mod2)]
Out[36]: 2-element Vector{Float64}:
 1.364600448234789
 1.2376068856972031

Check model assumption

In [37]: ## Check normality
using HypothesisTesting, Distributions
resid2 = residuals(mod2)
KS_test = ExactOneSampleSTest(resid2,Normal())

ArgumentError: Package HypothesisTesting not found in current path.
- Run `import Pkg; Pkg.add("HypothesisTesting")` to install the HypothesisTesting package.

Stacktrace:
 [1] macro expansion
   @ ./loading.jl:1163 [inlined]
 [2] macro expansion
   @ ./lock.jl:223 [inlined]
 [3] require(intro::Module, mod::Symbol)
   @ Base ./loading.jl:1144
 [4] eval
   @ ./boot.jl:368 [inlined]
```

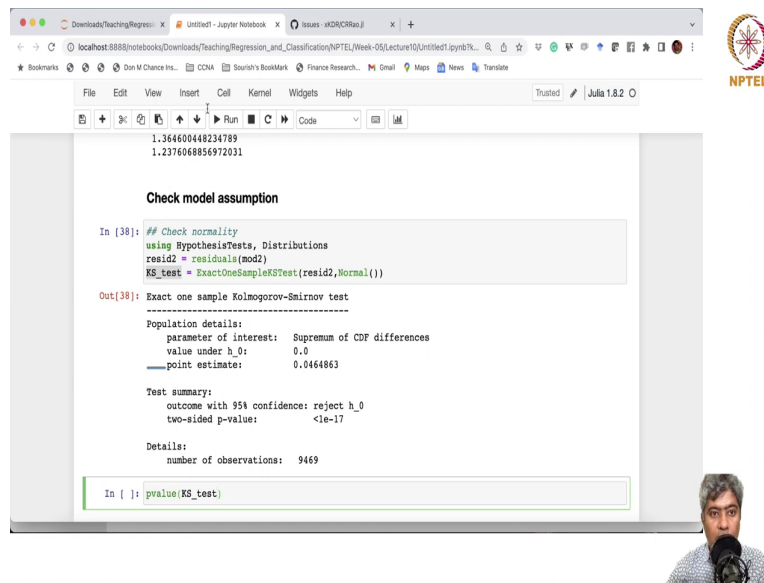


Check model assumptions. At least first thing we have to check this model assumption oh for the second model, we just we do not need to check model because we are going not going to use the first model in any way. Second model is significantly better than the first model. So, we will just do the check the model assumption of the first model.

So, let us check first normality, check normality. So, what we will do? We will just run a Kolmogorov – Smirnov of test for normality using hypothesis testing is a package and distribution distributions, ok. So, KS test is ExactOneSampleKSTest and we have to give residuals ok. So, what we will do, we have to first. So, resid2 from the resid2 will help me imagine ok this is from the model 2 resid residuals from mod2.

Sorry, mod2 and this needs to go to a Kolmogorov – Smirnov of test with normal distribution the test will be against normal distribution. So, let me just Run this ok. It is saying that I do not have the hypothesis testing. So, let me just pause the video if I just run this piece of code, it is saying that it will import the hypothesis testing. So, let me just pause the video and I will come back once I will install and I will come back, ok.

(Refer Slide Time: 40:59)



```
1.364600448234789
1.2376068856972031

Check model assumption

In [38]: ## Check normality
using HypothesisTests, Distributions
resid2 = residuals(mod2)
KS_test = ExactOneSampleKSTest(resid2, Normal())

Out[38]: Exact one sample Kolmogorov-Smirnov test
-----
Population details:
 parameter of interest: Supremum of CDF differences
 value under h_0:      0.0
 point estimate:       0.0464863

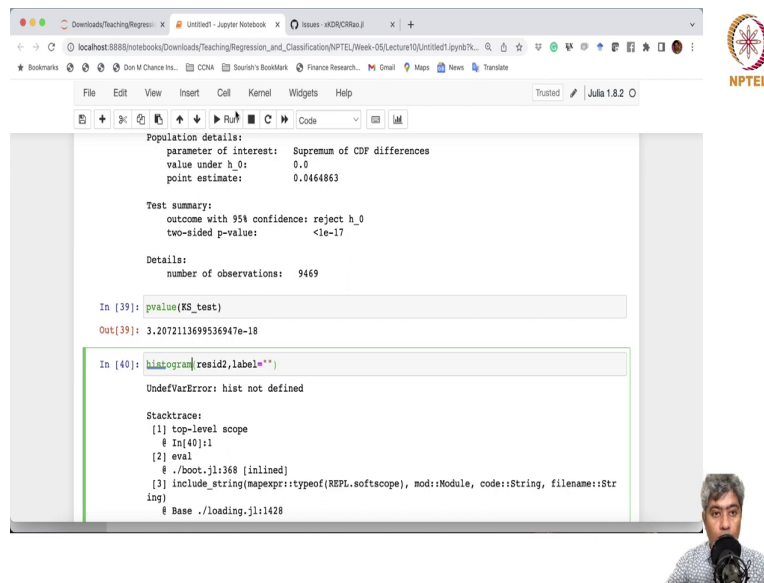
Test summary:
 outcome with 95% confidence: reject h_0
 two-sided p-value:      <1e-17

Details:
 number of observations: 9469

In [ ]: pvalue(KS_test)
```

Ok, I realized that the spelling there was a spelling mistake it should not be hypothesis testing this should be hypothesis tests. So, that is what the issue, I think. So, let me just run it yeah and its can and for two sided p test, the p value is really small. In fact, if you just want to see what is the pvalue if you just provide the test itself it will extract the pvalue and print it for you. So, it is like really small p value. So, it cannot really test the model.

(Refer Slide Time: 41:32)



The screenshot shows a Jupyter Notebook interface with the following content:

```
Population details:
  parameter of interest:  Supremum of CDF differences
  value under h_0:        0.0
  point estimate:         0.0464863



Test summary:
  outcome with 95% confidence: reject h_0
  two-sided p-value:      <1e-17

Details:
  number of observations:  9469

In [39]: pvalue(KS_test)
Out[39]: 3.2072113699536947e-18

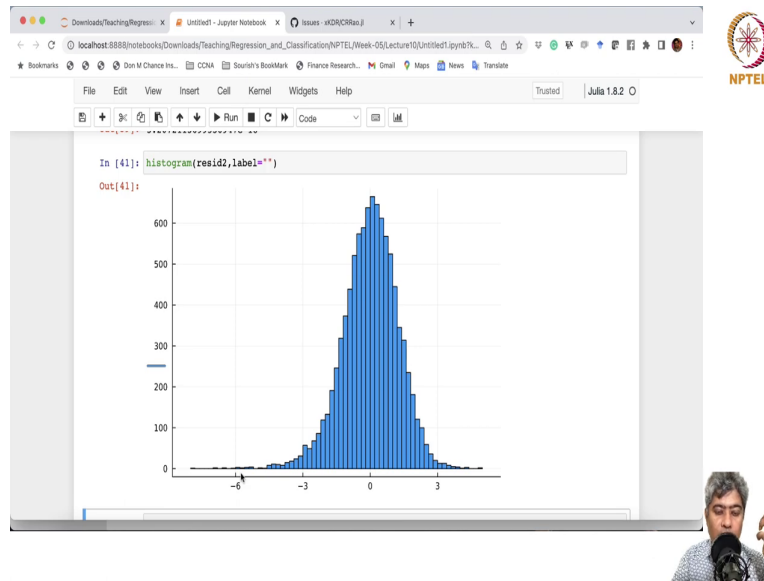
In [40]: histogram(resid2, label="")
Uncaught Error: hist not defined

Stacktrace:
[1] top-level scope
     @ In[40]:1
[2] eval
     @ ./boot.jl:368 [inlined]
[3] include_string(mapexpr::typeof(REPL.softscope), mod::Module, code::String, filename::String)
     @ Base ./loading.jl:1428
```



So, what we will do we will just also check how the actually this residual behaves. So, is it really bad? Let me just do residual resid 2 with no level maybe. Histogram I have to sorry I have to just write histogram.

(Refer Slide Time: 42:18)



So, ok this is the histogram of the residual, it is not bad. It is almost like a bell-shaped only thing it has a bit of a long negative tail. So, that means, that it has a essentially it means the way if you see it looks like it is underestimated, it is doing some underestimation probably.

So, like an underestimation happens when probably if a model is saying that ok it will be 30 degree Celsius and turns out to be it is 32 or 33 degree Celsius. So, naturally there will be a negative of the negative residuals and that residuals we are seeing here. There quite a way almost 6 degree difference in the predictions and that has also happened.

So, let me just stop here. So, that means, normality is an issue model is models predictive accuracy is reasonably ok almost 70 percent, 78 percent. So, we cannot if we want to go and say that ok this is beyond reasonable doubt that you know. The problem main problem that we will face is here and I will tell you what the problem is saying.

So, based on these analysis when we are running this model based on these analysis the coefficient of t over time is positive, it is increasing and p value is really small confidence interval does not include 0.

Now, the problem now we are facing is, so, based on this Chennai's temperature is increasing. Now, somebody can come and say that well your underlying assumption is that your residual follow normal distribution your normality assumption does not hold good ok clearly normality assumption does not hold good. Kolmogorov's – Smirnov test reject that assumption.

Therefore, you are in where therefore, your these inference that temperature on an average temperature of Chennai is going up over the last 20 years 20 almost 30 years is not correct. It is not correct because your underlying assumption is wrong and what is the probability that it will be correct? The way to solve this issue is very simple we know that we have to run bootstrap regression. Residual bootstrap regression does not or even paired bootstrap regression does not rely on assumption of normality.

In fact, it assumed that any residual follow any probability distribution with some mean and variance and looks like this is fine. The mean zero is assumption is fine and some variance is also fine. So, we will we can implement this model using bootstrap regression. So, I am leaving this to you as we have already studied bootstrap regression, I am leaving it to you. Try to write the bootstrap regression and implement this.

I will share this Jupyter notebook in the NPTELs forum and in the NPTEL's platform in Swayam platform. Please go there, download this notebook and also the data will be available there I already shared the data with you. Please go ahead, download this notebook and implement the bootstrap regression using CRRao package ok.

So, CR Rao package has not implemented the bootstrap regression yet. But I think you can do it very simply, quickly by yourself we have done it in R you have seen basically you have to translate that code into bootstrap regression.

So, with that I will stop here. Thank you so much. Take care, bye.