CS 251 Statistical Computing

HOP 4: R for statistical project

4/05/2020 Developed by Aya Khalil

10/18/2019 Reviewed by Shanshan Yu

School of Technology and Computing @City University of Seattle (CityU)

**Before You Start**

* If you already finished this module through any CityU Technology Institute (TI) courses,  
  just skim this module and skip it.
* Version numbers may not match with the guide. But that should be fine.  
  If given the option to choose between stable release (long-term support) or most recent, please choose the stable release.
* This guide targets Windows OS users. So, MacOS users may have different commands to input in the shell/terminal.
* We cannot explain every step. **This cookbook always needs your own creative judgement.**
* **For your working directory, use your course number.** The hands-on tutorial may use a different course number as an example.

**Learning Outcomes**

* Descriptive Statistics
* Normal Distribution

**Resource**

* Hui, E. G. M. (2019). [*Learn R for applied statistics: With data visualization, regressions, and statistics*](https://login.proxy.cityu.edu/sso/skillport?context=144516). Apress.
* Percentiles: <https://www.statology.org/how-to-easily-calculate-percentiles-in-r-with-examples/>
* Normal Distribution: <https://www.tutorialspoint.com/r/r_normal_distribution.htm>
* Normal Distribution: <https://data-flair.training/blogs/normal-distribution-in-r/>
* Normal Distribution: <https://www.statology.org/dnorm-pnorm-rnorm-qnorm-in-r/>

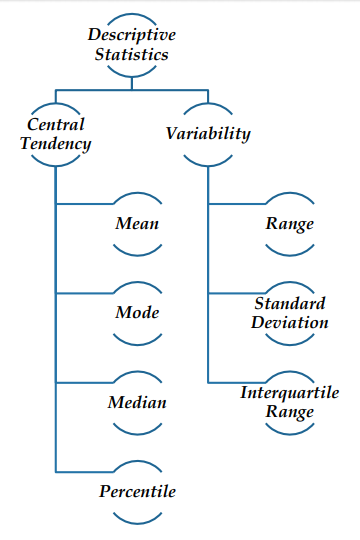
**Section1: Descriptive Statistics**

Descriptive statistics is a set of math used to summarize data. Descriptive statistics can be distribution, central tendency, and dispersion of data. The distribution can be a normal

distribution or binomial distribution. The central tendency can be mean, median, and mode. The dispersion or spreadness can be the range, interquartile range, variance, and

standard deviation.

descriptive statistics, central tendency measurements, dispersion measurements, and distributions will be explained. we will look into how R programming can be used to calculate all these values, and how to test and see whether data is normally distributed.



Mean, median, and mode are the most common measures for central tendency. Central tendency is a measure that best summarizes the data and is a measure that is related to the center of the data set.

**Descriptive Statistics: Central Tendency - Mean**:

Also known as Average

- Affected by extreme values

Example: 10, 11, 14, 9, 6

Mean = (10+11+14+9+6)/5 = 50/5 = 10

**Setup Working Environment for Module4**

1. Open VS Code.

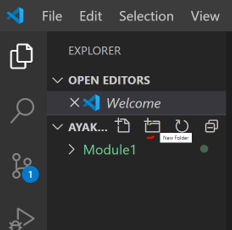
* Open Desktop/CS251/hop04-YourUserName ( File > Open Folder)

1. Then, create the “**Module4**” directory in the VSCode.

>>>mkdir Module4

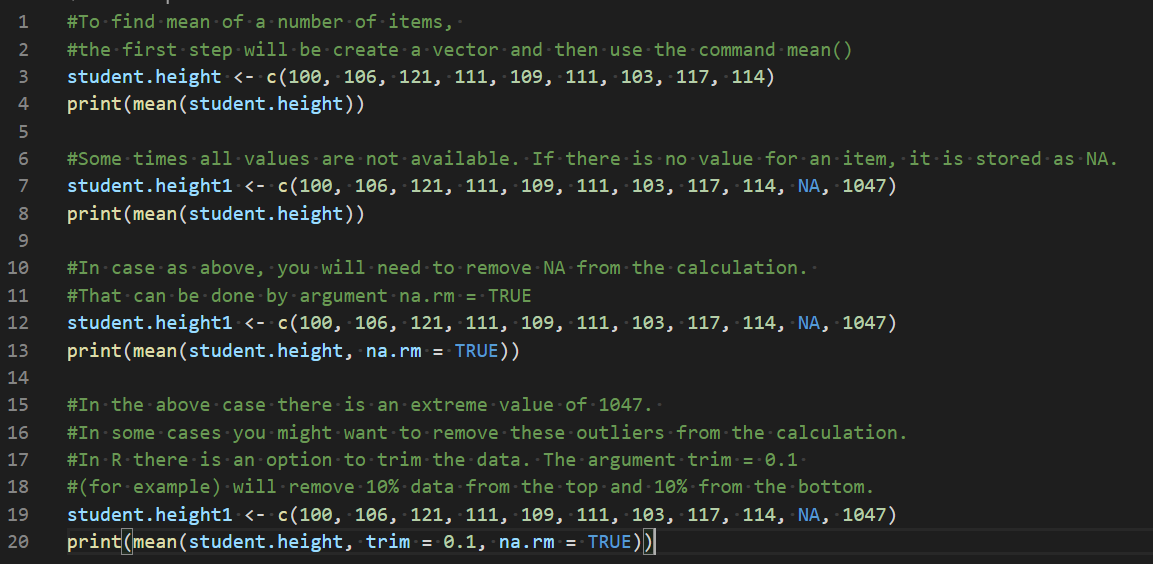
OR

Click on New Folder button that behind your name, and name the folder Module4



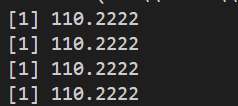
-In module4 project folder, create new file DescrpStatMean.R

- Type the following code in DescrpStatMean.R file



-Run your code:

- Select Run Source button

Output: 

**R comes with several built-in data sets, which are generally used as demo data for playing with R functions.**

In the R console, type data(). You will see the R data sets that we can use

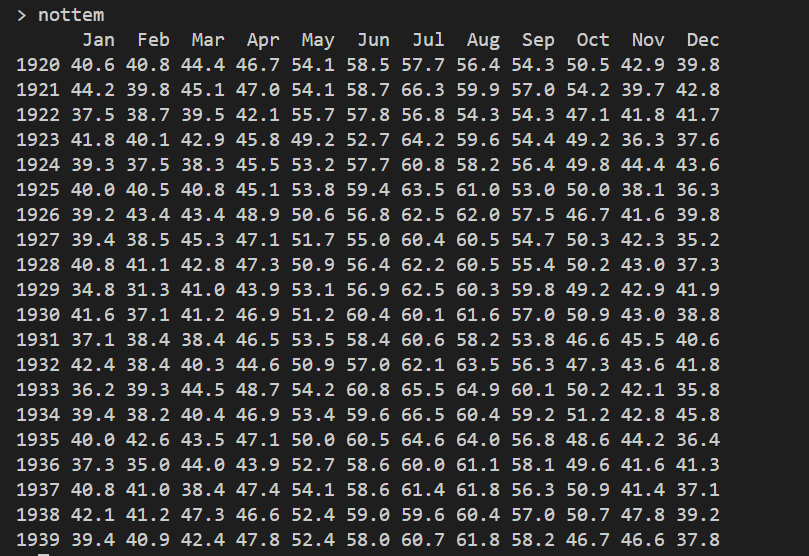


Scroll down to see the different datasets that we can use in R.

For this module, we are going to use nottem dataset

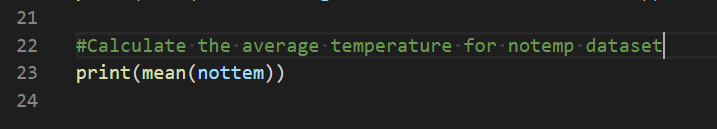
nottem is the Average Monthly Temperatures at Nottingham, 1920-1939

Go ahead and type nottem in the R console to see the dataset and you will see the following



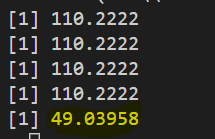
Let’s calculate the mean of the temperature for that data Frame

- Add the following code to DescrpStatMean.R file



Save your code, file>save or ctrl+s

-Run your code: Select Run Source button

Output: 

This means that 49.03958 was the average temperature at Nottingham, 1920-1939

**Descriptive Statistics: Central Tendency - Median and Mode**

**Median:**

Middle value when put in ascending or descending order.

Example: 10, 11, 14, 9, 6

In ascending order 6,9,10,11,14

Median = 10

Example: 10, 11, 14, 9, 6, 11

In order 6,9,10,11, 11,14

Median = 10.5

**Mode:**

Most occurring item

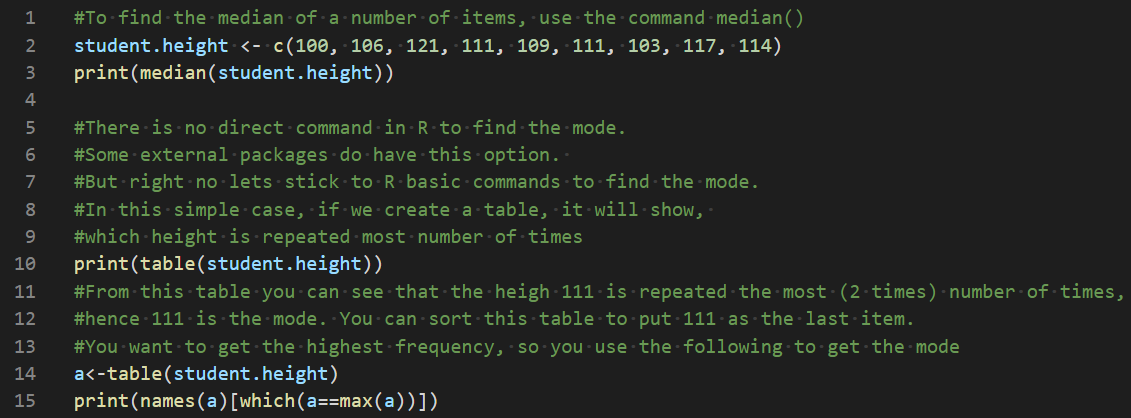
Example: 10, 11, 14, 9, 6, 10

Mode = 10

**You should be in:**

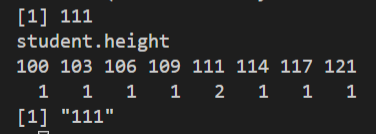
* Desktop/CS251/hop04-YourUserName
* In **Module4** project folder, create new file modMedian.R

- Type the following code in modMedian.R file



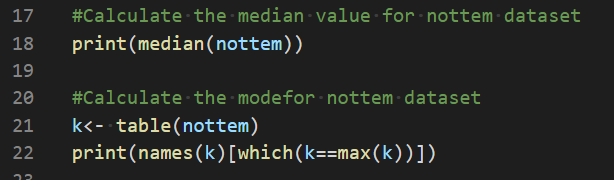
Save your code, file>save or ctrl+s

-Run your code: Select Run Source button

-Output: 

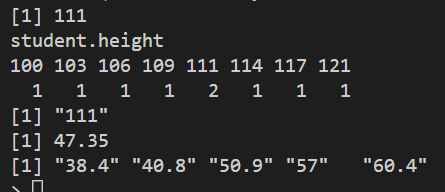
Let’s try to calculate the median of nottem dataset

-Add the following to update modMedian.R file



Save your code, file>save or ctrl+s

-Run your code: Select Run Source button

Output: 

**Descriptive Statistics: Central Tendency – Percentile**

The nth percentile of a dataset is the value that cuts off the first n percent of the data values when all of the values are sorted from least to greatest.

For example, the 90th percentile of a dataset is the value that cuts of the bottom 90% of the data values from the top 10% of data values.

**Please watch the following video to understand the concept of percentile.**

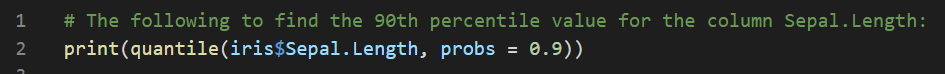
StatQuest: Quantiles and Percentiles, Clearly Explained!!!. Retrieved from: <https://www.youtube.com/watch?v=IFKQLDmRK0Y>

We will use the built-in data frame iris

**You should be in:**

* Desktop/CS251/hop04-YourUserName
* **In Module4** project folder, create new file Percen.R

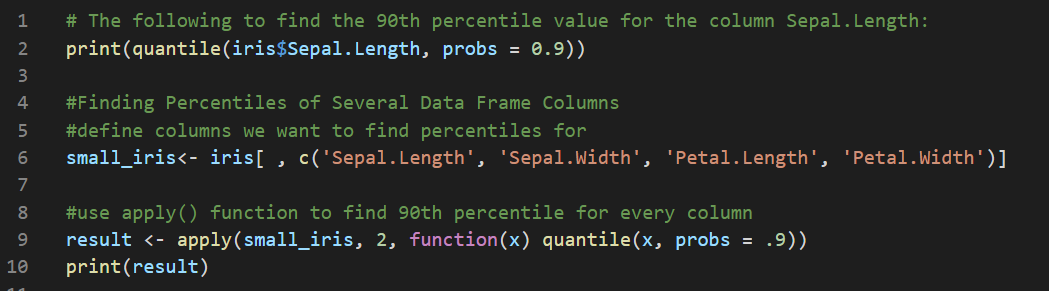
Add the following to Percen.R



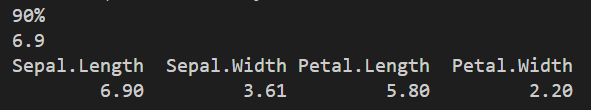
Run your code.

Output: 

We can Find the Percentiles of several Data Frame columns

* Add the following to update Percen.R

Run your code.

Output: 

**Measurement of Variation**

Measures of variability are the measures of the spread of the data. Measures of variability can be range, interquartile range, variance, standard deviation, and more.

**Descriptive Statistics: Variability - Range**

Range provides the highest and the lowest value

**Quantile**

This provide 0, 25, 50, 75 and 100 quartile. In this:

• 0% = Lowest value

• 25% = First quartile (Q1)

• 50% = Median (Q2)

• 75% = Third quartile (Q3)

• 100% = Highest value

**Descriptive Statistics: Variability - Interquartile Range**

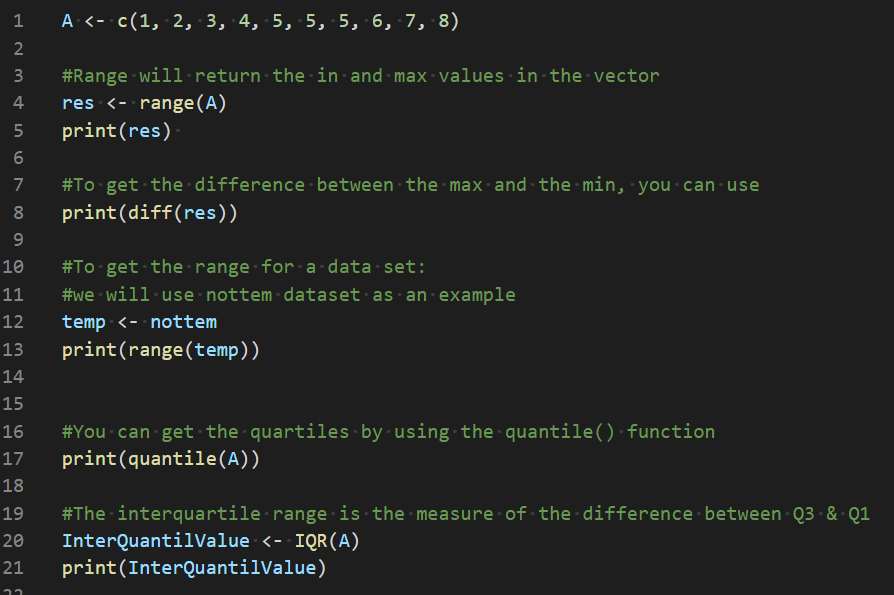
The interquartile range is the measure of the difference between the 75 percentile or third quartile and the 25 percentile or first quartile.

To get the interquartile range, you can use the IQR() function

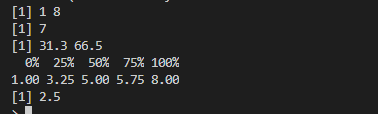
**You should be in:**

* Desktop/CS251/hop04-YourUserName
* In **Module4** project folder, create new file MesurVar.R

- Type the following code in MesurVar.R file



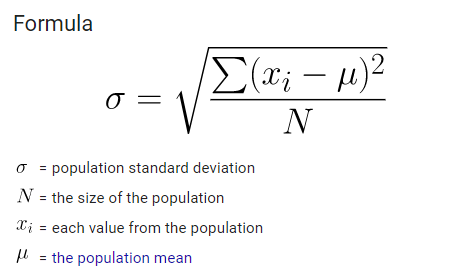
-Save & Run your code

-o/p 

**Descriptive Statistics: Variability – Standard Deviation**

In statistics, the standard deviation is a measure of the amount of variation or dispersion of a set of values. A low standard deviation indicates that the values tend to be close to the mean of the set, while a high standard deviation indicates that the values are spread out over a wider range.

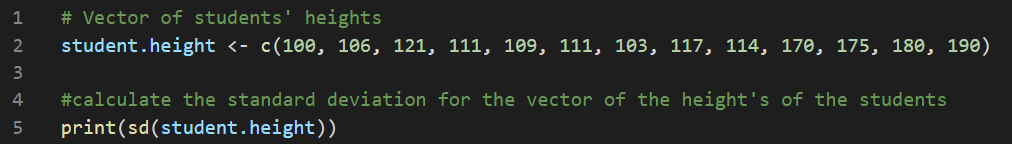
**Standard deviation equation:**



**You should be in:**

* Desktop/CS251/hop04-YourUserName
* **In Module4** project folder, create new file StandrDev.R

**Add the following to StandrDev.R**

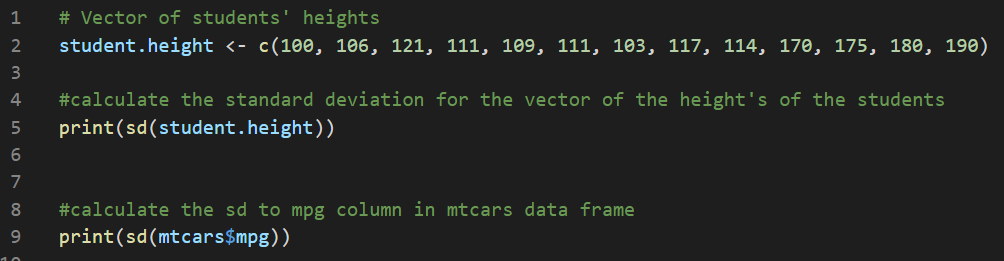


**Run your code**

**Output:** 

**We also can calculate the standard deviation for a column in a data frame**

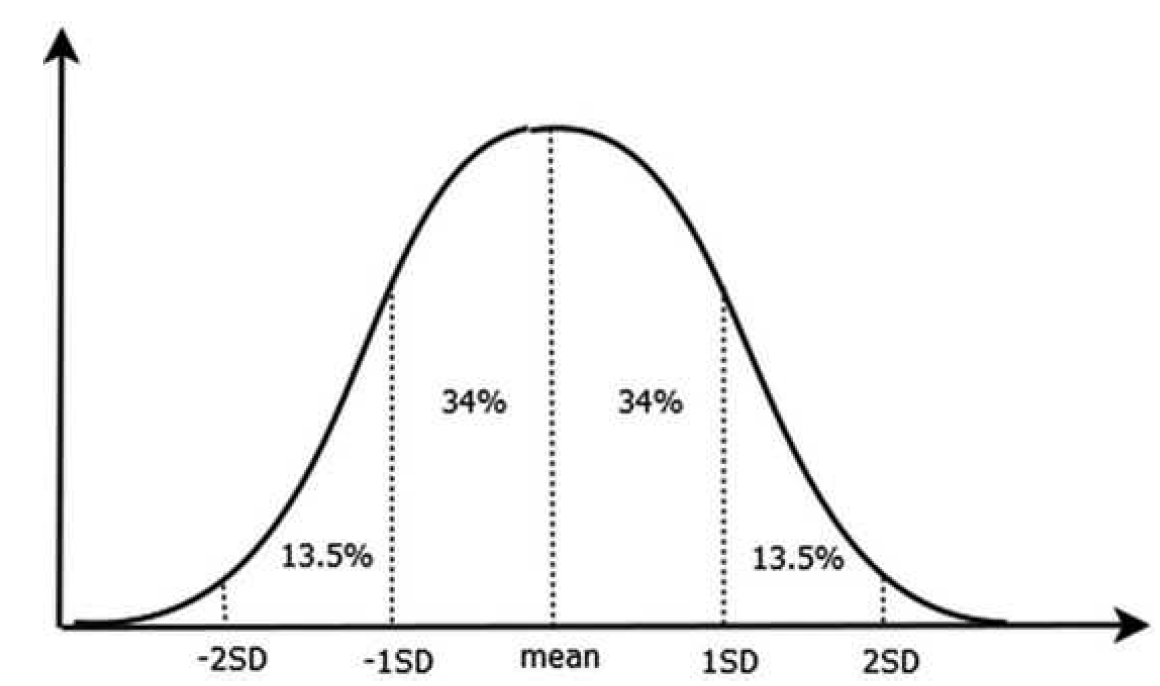
* Add the following to StandDev.R



Run your code

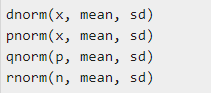
Output: 

**Section2: Normal Distribution**



In a random collection of data from independent sources, it is generally observed that the distribution of data is normal. Which means, on plotting a graph with the value of the variable in the horizontal axis and the count of the values in the vertical axis we get a bell shape curve. The center of the curve represents the mean of the data set. In the graph, fifty percent of values lie to the left of the mean and the other fifty percent lie to the right of the graph. This is referred as normal distribution in statistics.

R has four in built functions to generate normal distribution. They are described below.

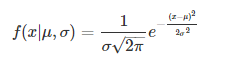


The following is the description of the parameters used in above functions:

* **x** is a vector of numbers.
* **p** is a vector of probabilities.
* **n** is the number of observations (sample size).
* **mean** is the mean value of the sample data. Also, its default value is zero.
* **sd** is the standard deviation. Its default value is 1.

**Normal Distribution – dnorm()**

As we all know the probability density for the normal distribution is:



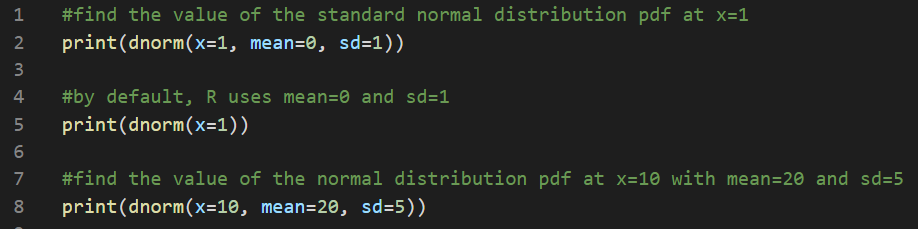
The function dnorm returns the value of the probability density function (pdf) of the normal distribution given a certain random variable x, a population mean μ and population standard deviation σ. The syntax for using dnorm is as follows:

**dnorm(x, mean, sd)**

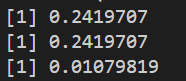
**You should be in:**

* Desktop/CS251/hop04-YourUserName
* **In Module4** project folder, create new file NDdnorm.R

Add the following to NDdnorm.R



Run your code.

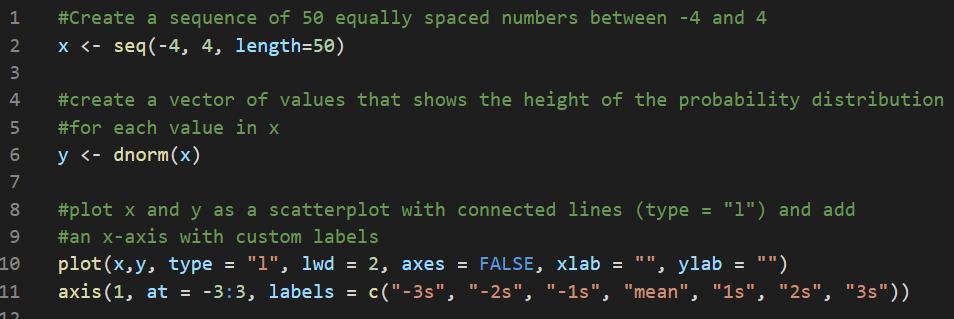
Output

Typically when you’re trying to solve questions about probability using the normal distribution, you’ll often use pnorm instead of dnorm. One useful application of dnorm, however, is in creating a normal distribution plot in R. The following code illustrates how to do so:

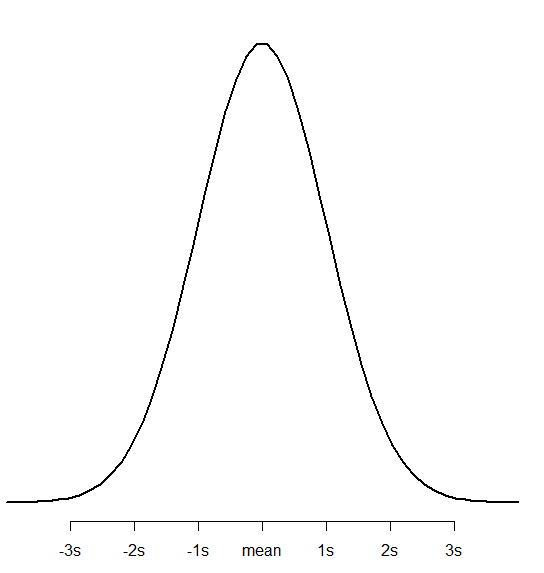
**You should be in:**

* Desktop/CS251/hop04-YourUserName
* **In Module4** project folder, create new file NDdnormPlot.R

Add the following to NDdnormPlot file



Run your code

Output 

**Normal Distribution – pnorm()**

The function pnorm returns the value of the cumulative density function (cdf) of the normal distribution given a certain random variable q, a population mean μ and population standard deviation σ. The syntax for using pnorm is as follows:

**pnorm(q, mean, sd)**

**Please watch the following video to understand the concept of pnorm & qnorm**

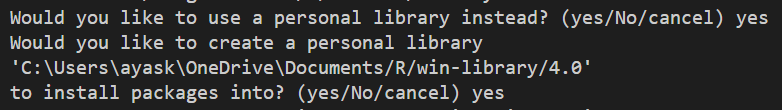
(help) Using pnorm and qnorm. (2017). Retrieved from: <https://www.youtube.com/watch?v=M2v1JCjdrgQ>

**Install the following package so we can use pnormGC**

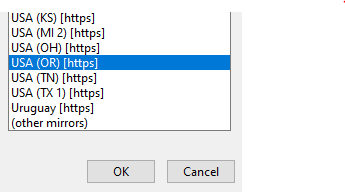
install.packages("tigerstats")



**If it asks you the following, just type yes**



**Choose the any of the USA CRAN**



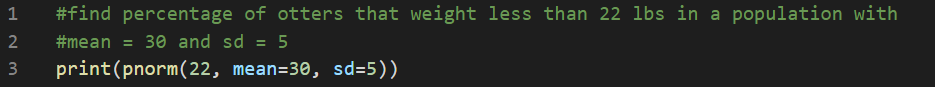
**You should be in:**

* Desktop/CS251/hop04-YourUserName
* **In Module4** project folder, create new file NDpnorm.R

**Question**: Suppose the weight of a certain species of otters is normally distributed with a mean of μ=30 lbs and a standard deviation of σ = 5 lbs. Approximately what percentage of this species of otters weight less than 22 lbs?

**Answer:**

Add the following to NDpnorm.R file

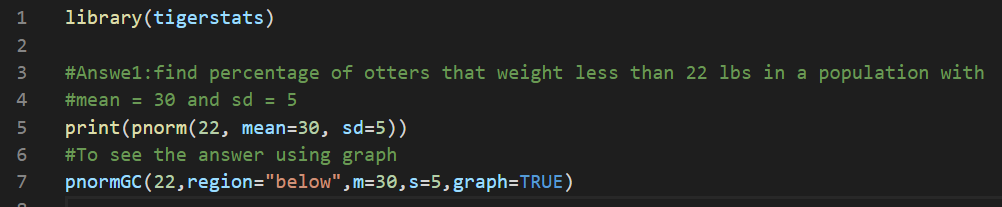


Run your code

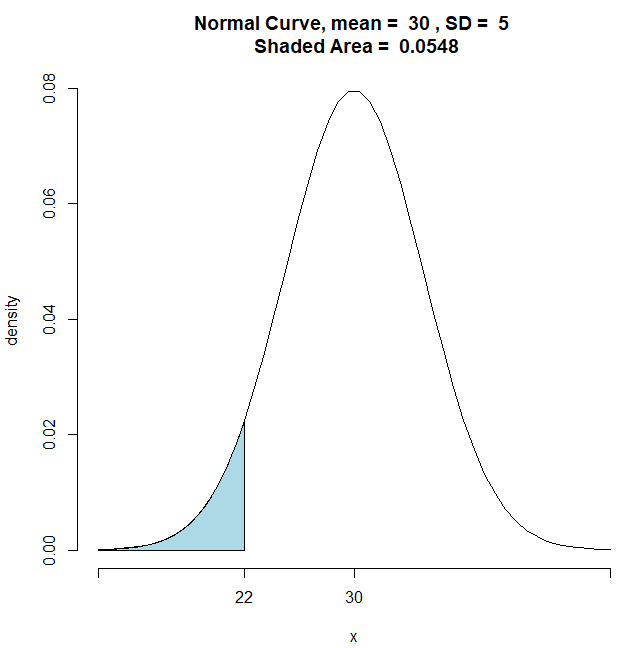
Output: 

This means that approximately 5.4799% of this species of otters weigh less than 22 lbs.

**Add the following to see the answer using graph.**



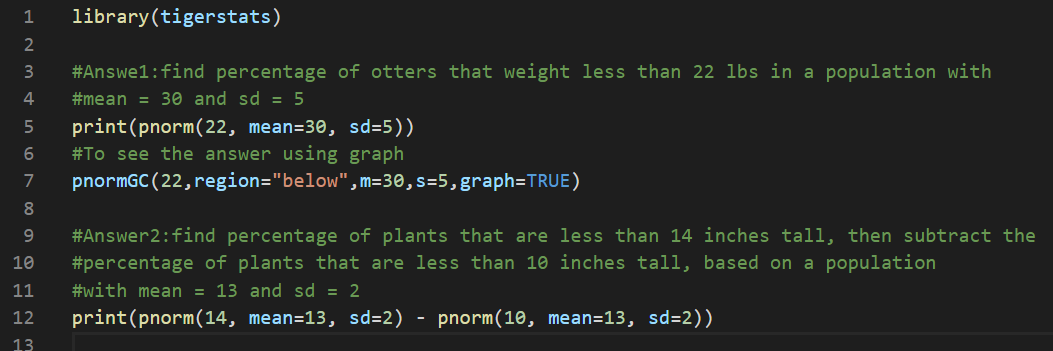
Run your code

Output: 

**Question**: Suppose the height of plants in a certain region is normally distributed with a mean of μ=13 inches and a standard deviation of σ = 2 inches. Approximately what percentage of plants in this region are between 10 and 14 inches tall?

**Answer**:

Add the following to update NDpnorm.R file

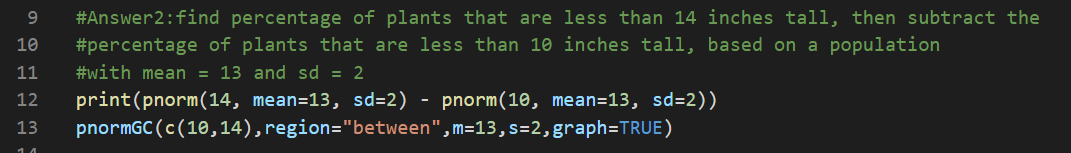


Run your code.

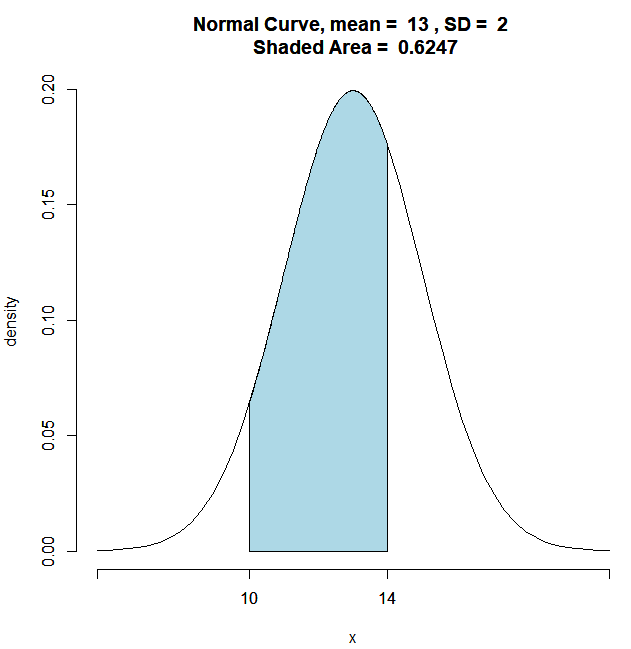
Output: 

This means that approximately 62.4655% of plants in this region are between 10 and 14 inches tall.

**Add the following to see the answer using graph.**



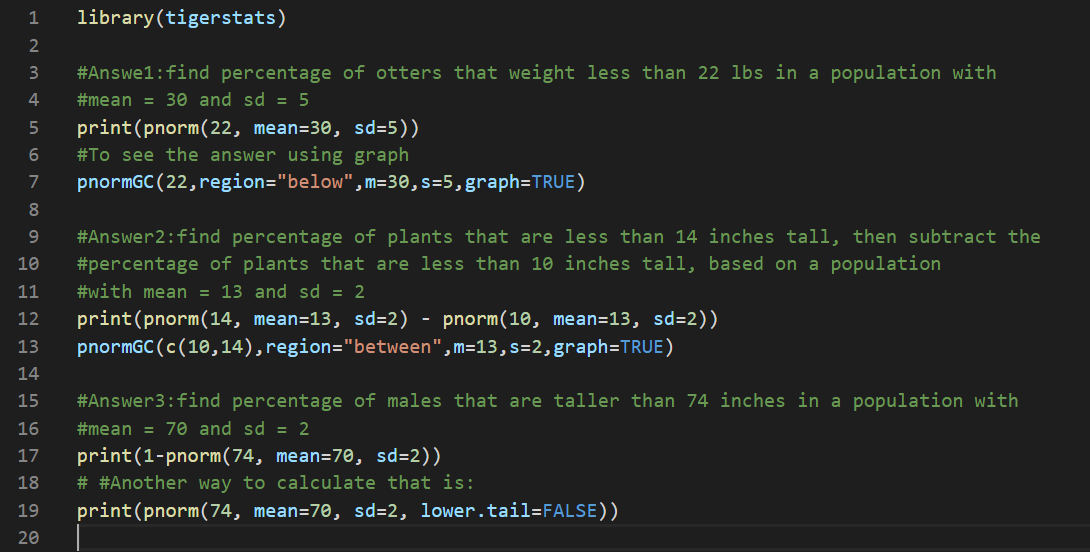
Run your code

Output: 

**Question**: Suppose the height of males at a certain school is normally distributed with a mean of μ=70 inches and a standard deviation of σ = 2 inches. Approximately what percentage of males at this school are taller than 74 inches?

**Answer**:

Add the following to update NDpnorm.R file

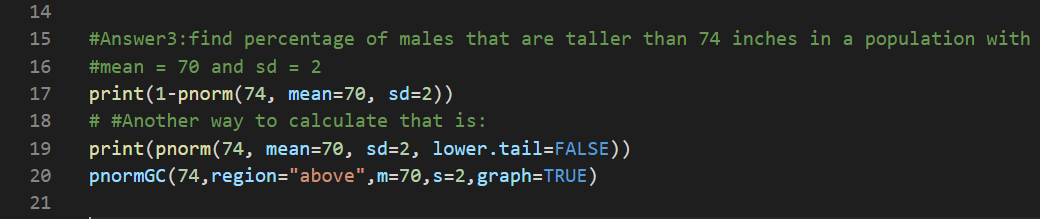


Run your code:

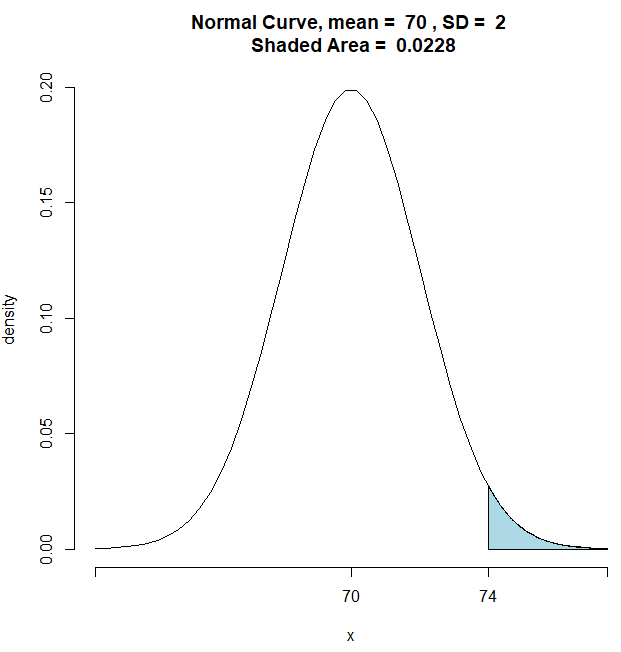
Output: 

This means, at this school, 2.275% of males are taller than 74 inches.

**Add the following to see the answer using graph.**



Run your code

Output: 

**Normal Distribution – qnorm()**

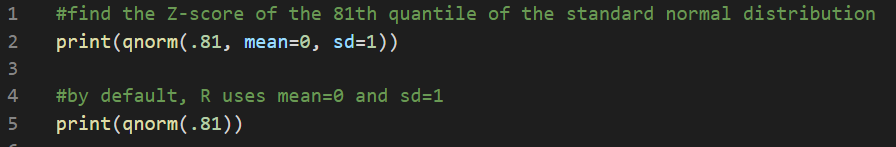
The function qnorm returns the value of the inverse cumulative density function (cdf) of the normal distribution given a certain random variable p, a population mean μ and population standard deviation σ. The syntax for using qnorm is as follows:

**qnorm(p, mean, sd)**

**You should be in:**

* Desktop/CS251/hop04-YourUserName
* **In Module4** project folder, create new file NDqnorm.R

**Add the following to NDqnorm.R**

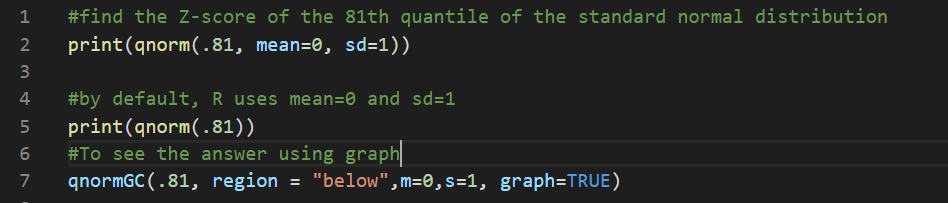


Run your code.

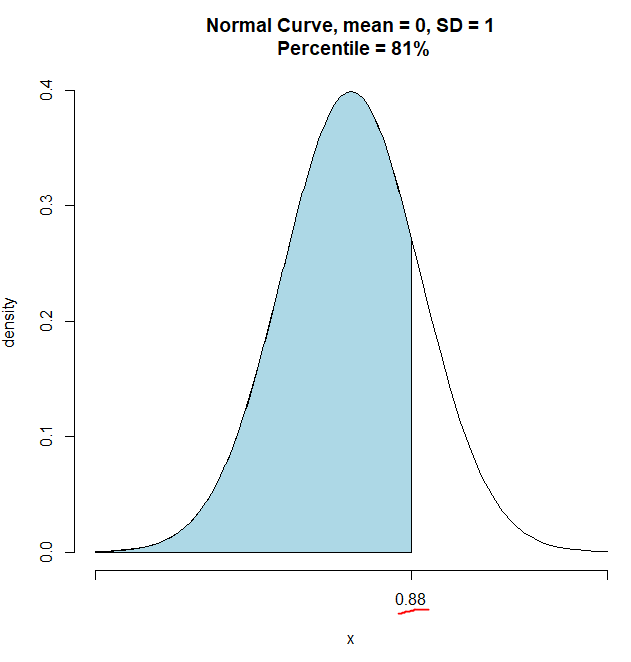
Output: 

This means that the 0.8778963 is the boundary value that determines this area.

Type the following to see the answer using graph



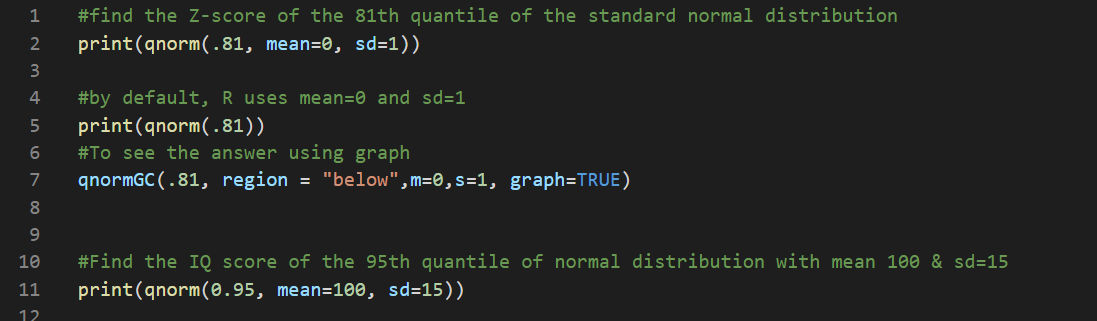
Run your code

Output: 

**Question**: Suppose IQ scores are normally distributed with mean 100 and standard deviation 15. What is the 95th percentile of the distribution of IQ scores?

**Answer:**

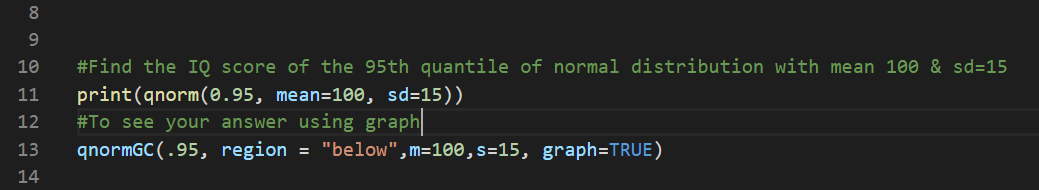
**Add the following to NDqnorm.R**



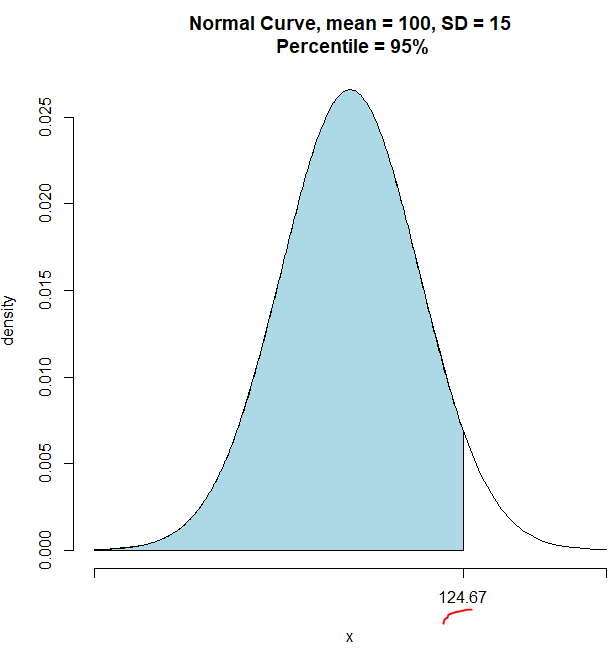
**Run your code**

**Output** 

Update your code like the following to see the answer using graph



Output



**Normal Distribution – rnorm()**

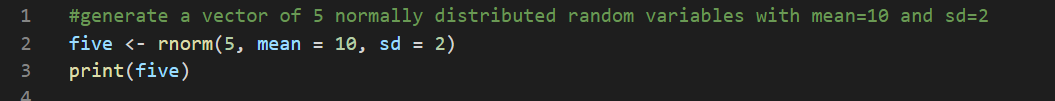
The function rnorm generates a vector of normally distributed random variables given a vector length n, a population mean μ and population standard deviation σ. The syntax for using rnorm is as follows:

**rnorm(n, mean, sd)**

**You should be in:**

* Desktop/CS251/hop04-YourUserName
* **In Module4** project folder, create new file NDrnorm.R

**Add the following to NDrnorm.R**



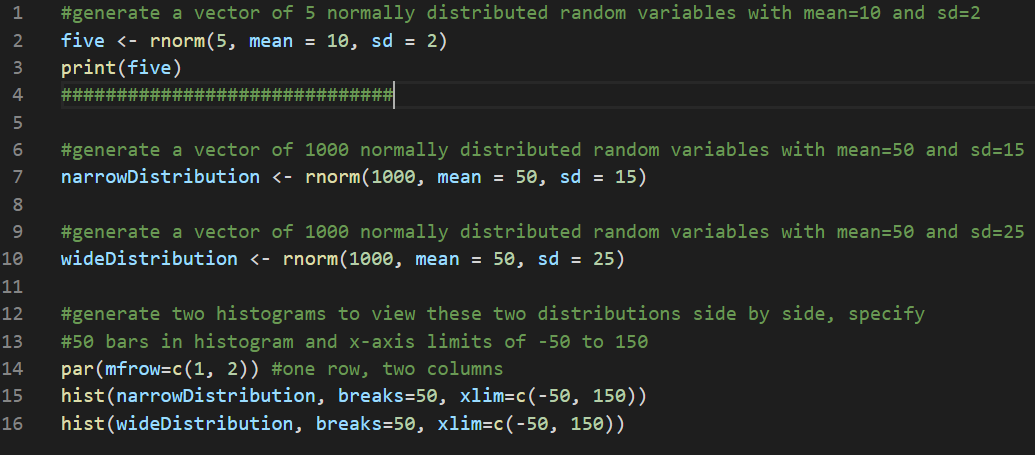
**Run your code**

**Output** 

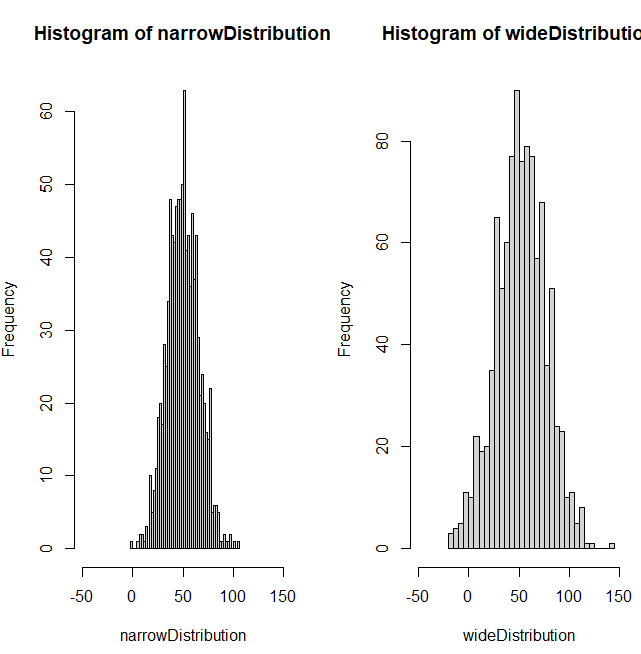
Note: Your output will be different than my output because rnorm generate random numbers

Let’s do another example that shows the random points

**Add the following to update NDrnorm.R**



**Run your code**

**Output:** 

**Push your work to GitHub**

**Make sure you are in**

Desktop/CS251/hop04-YourUserName

Run the following commands to push your work to the GitHub repository:

Open the terminal from the VSCode by hit the **control + ~** key and type the following command:

>>> git add .

>>> git commit -m “Submission for Module 4”

>>> git push origin master